GENERATION OF VIRTUAL 3D CONCRETE MICROSTRUCTURES APPLIED TO MODELING CREEP AND RELAXATION

A Dissertation

by

CHRISTA E. TORRENCE

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2020

Major Subject: Materials Science & Engineering

ABSTRACT

Nearly all nuclear power plants in the United States are operating past their intended lifetimes or are requesting lifetime extensions. Therefore, understanding changes to the concrete containment structure over time is crucial to evaluate the structure's continued viability. Concrete materials are heterogeneous particulate composites that exhibit viscoelastic material properties, which can lead to slow deformation over time, causing stress redistribution and the potential for creep cracking. A code to generate random, 3D concrete microstructures has been developed with a novel overlap detection algorithm that reduces the computational time to generate a microstructure with a realistic volume fraction by 85%. This code is paired with finite element analysis to predict the long-term viscoelastic properties of concrete, where data from these simulations are used to develop constitutive equations for the viscoelastic behavior of the homogenized concrete. To validate this work, the simulated creep behavior of concrete is compared to 800 days of experimental data that has been extended to 27 years of data using the Time-Temperature superposition (TTS) principal. Excellent agreement between the simulation results and experimental data is seen. Additionally, two-dimensional (2D) and three-dimensional (3D) simulations were compared to determine the impact of simulation dimensionality on predicted creep behavior. Boundary conditions, both morphological and mathematical, are investigated for their influence on concrete creep predictions. Lastly, short-term mortar stress relaxation is simulated to demonstrate the application of this framework to short term and early age creep The codes in this work are used to virtualize laboratory experiments, to obtain long-term creep data in a faster, cheaper manner.

DEDICATION

To my parents

ACKNOWLEDGMENTS

First, I'd like to express my appreciation and gratitude towards my advisor, Dr. Zachary Grasley. His guidance, patience, and support of me and my research throughout the past four years has been invaluable and has made my graduate school experience an enjoyable one. I would also like to thank my two external committee members, Dr. Edward J. Garboczi and Dr. William B. Lawrimore, for sharing their expertise and investing their time into helping shape and improve my research. My TAMU committee members, Dr. Joe Bracci, Dr. Patrick Shamberger, and Dr. Bjorn Birgisson have also each played an important role in the creation of this dissertation by providing helpful feedback and encouragement along the way.

I'd like to thank my labmate Aishwarya Baranikumar for providing experimental data on mortar and concrete for use in this work, as well as being a lovely officemate. Additionally, Cesario Tavares has provided collaboration ideas and experimental data used in this work as well.

I am immensely grateful for my parents, Mom, Dad, Bud, and Dani, for being my biggest cheerleaders and always supporting my goals (even when it meant moving away). None of this would have been possible without you.

To my friends, both near and far, I appreciate each one of you dearly. I am so happy to be surrounded by such wonderful, thoughtful, and inspiring people. You all have supported me in so many ways and for that I am forever grateful.

Finally, a special thank you to my fiance Connor, for being an unwavering source of support throughout this journey. Thank you for the many words of encouragement, always making me laugh, and for being a great cat dad.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee chaired by Professor Zachary Grasley of Civil Engineering and Materials Science & Engineering. The committee members include Professor Patrick Shamberger of Materials Science & Engineering, Professor Joseph Bracci of Civil Engineering, Professor Bjorn Birgisson of Civil Engineering, Dr. Edward J. Garboczi of the National Institute of Standards and Technology (NIST), and Dr. William B. Lawrimore of the U.S. Army Engineering Research and Development Center (ERDC).

The Anm coefficients used to reconstruct the aggregate shapes, presented in Chapter 3 was generated and analyzed by Dr. Edward J. Garboczi. The mortar and concrete experimental data presented in Chapters 5 and 6 was collected and analyzed by Aishwarya Baranikumar. The cement paste and mortar experimental data in Chapter 7 was collected and analyzed by Cesario Tavares.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by the College of Engineering Graduate Engineering Merit fellowship from Texas A&M University. This research was also supported by a Nuclear Energy University Partnership (NEUP) grant 16-10457 through US DOE, Office of Nuclear Energy, Light Water Reactor Sustainability (LWRS) Program. Additional support came from the US Army Engineering Research and Development Center (ERDC) grant W912HZ19C0042. Portions of this research were conducted with high performance research computing resources provided by Texas A&M University (https://hprc.tamu.edu).

NOMENCLATURE

2D	Two-Dimensional
3D	Three-Dimensional
ASTM	American Society for Testing and Materials
DEM	Discrete Element Method
EDF	Électricité de France
ERDC	Engineering Research and Development Center
FEA	Finite Element Analysis
FEM	Finite Element Method
FFT	Fast Fourier Transform
NIST	National Institute of Standards and Technology
PBC	Periodic Boundary Conditions
RVE	Representative Volume Element
TAMU	Texas A&M University
TTS	Time-Temperature Superposition
XCT	X-Ray Computed Tomography

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Motivation and Problem Statement

In the United States, 96 nuclear reactors are in operation across 58 commercially operating nuclear power plants [11]. These plants are initially designed with a 40 year operating lifespan, however 47 reactors are operating at ages between 40 and 60 years. The Nuclear Regulatory Commission grants 20 year licence renewals to reactors that pass extensive safety and environmental impact reviews. In December 2019, the NRC granted its first subsequent license renewal to two reactors, allowing them to operate until they reach 80 years of age. As shown in Figure 1.1, the fleet of reactors in the US is quickly aging, with many reactors applying for for subsequent license renewals in the coming years.

Within these nuclear power plants, concrete plays a crucial role by serving as the biological shield surrounding the reactor, blocking radiation from escaping. While over 50% of operating nuclear reactors are safely in use past their initial design life, not all reactors have had such success. In 2009 during scheduled refueling and maintenance work at the Crystal River power plant in Florida, cracks formed throughout the 42 inch thick concrete containment structure [12]. Additional damage to the containment structure occurred two years later during an attempt to repair the initial damage [13]. The estimated cost of repairs exceeded $3 billion dollars, leading to permanent closure of the plant [12].

To ensure the safe operation of aging nuclear power plants, as well as other concrete structures, understanding the changes to the concrete components over time is crucial. Concrete is a viscoelastic material, meaning under a constant load the strain will slowly increase, a phenomena called creep that continues for the duration of loading [14]. The importance of understanding long-term concrete creep stems from how long-term deformations associated with creep lead to loss of prestress, stress redistribution, and the potential for creep cracking. As prestress post-tensioned concrete

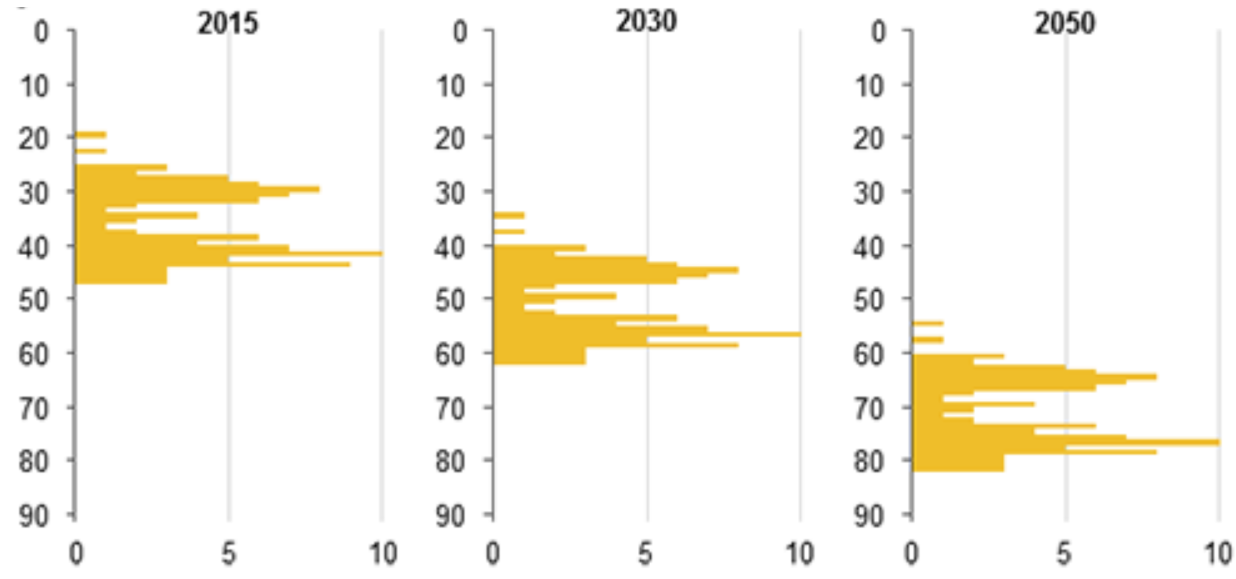


Figure 1.1: Age distribution of nuclear reactors at selected dates [1].

have a wide variety of infrastructure applications, understanding the changes to the loss of prestress and post-tensioning is crucial to assess the concrete structure's ongoing viability and service life. Modeling of concrete creep has been researched for decades, both analytically [15, 16, 17, 18] and through numerical simulations [4, 19]. Due to the lengthy time required to measure creep in an experimental setting, a computational alternative that is both faster and cheaper is highly attractive.

A computational approach to model concrete creep has several notable difficulties, including the complexity of concrete creep and the accurate morphological representation of concrete as a composite material. In regards to the former, a variety of mechanisms are involved in the creep of concrete, including drying creep, aging, temperature dependence, and basic creep [14]. Considering the application at hand, nuclear containment structures, several of these mechanisms can be mitigated. Most NPPs in the US have a steel liner that prevents drying from occurring from the internal surface. Experiments performed on 30 year old concrete in a nuclear containment wall with a liner showed that internal relative humidity (RH) was above 80% across much of the wall's thickness [20]. Thus, drying creep can be considered a minor component of the overall creep of concrete in NPP containment walls. Aging effects of concrete have the biggest impact

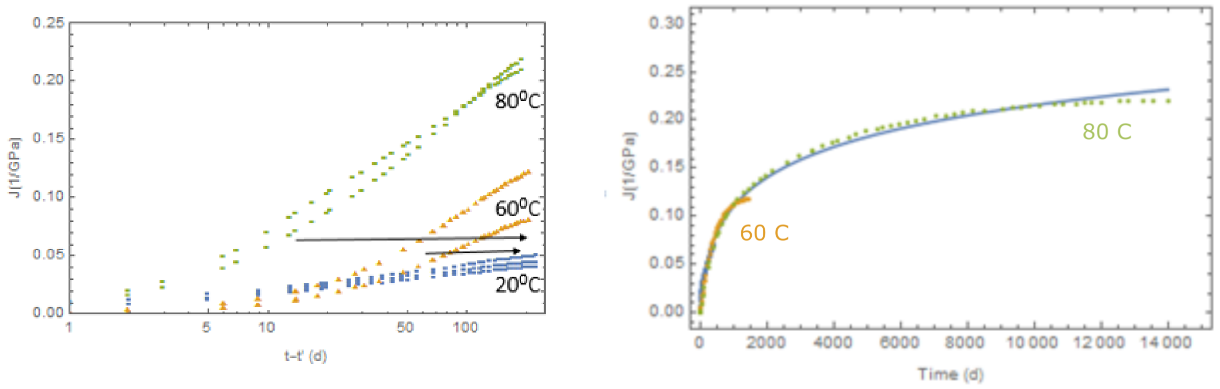


Figure 1.2: Creep compliance data on log time scale (left) used to create master curve for creep compliance (right) using the TTS principle [2]

during the early stages of curing, specifically the first 28 days. This can then treated as a a minimal factor of structural concrete creep because NPP concrete is loaded at ages greater than 28 days. The dependence of concrete creep on temperature is leveraged as an advantage through the use of the Time-Temperature Superposition (TTS) principle. TTS is a concept that has been applied to temperature-dependent creep in other materials, particularly polymers. Concurrent research has been performed on the creep of mortar specimens at three different temperatures. The TTS principal is used to generate a master curve depicting room-temperature creep data over several decades as seen in Figure 1.2 [2, 21].

The second substantial hurdle to modeling concrete creep is the generation of a representative virtual concrete sample. As concrete is a particulate composite composed of aggregates suspended in a cement paste matrix, the development of a virtual microstructure is akin to a particle packing or particle placement problem. As particle packing has applications in a vast array of industries and fields of research, a wide number of approaches have been developed. An important distinction between the concrete application at hand and a majority of the particle packing codes and approaches is the morphology of the particles, which in this case are the aggregates in concrete. Sphere packing is the most well-studied [22, 23, 24, 25, 26, 27], followed by other regularly shaped particles including ellipsoids [28, 29, 30, 31], and cylinders [32, 33, 34]. Irregular shapes add a substantial layer of complexity to placement and packing codes, as detecting overlap between irregularly shaped particles is challenging to implement. As two particles, or aggregates, cannot coexist in the same physical space at the same time, preventing virtual particle-particle overlap is an extremely important aspect of placement and packing codes. Retaining the morphology of real aggregates is important, as aggregate shape has been shown to influence many physical [35, 36, 37, 38, 4] and chemical [39, 40] properties of concrete.

Concrete creep has been studied for many years both experimentally and analytically in efforts to predict it [15, 16, 17]. As shown by Brooks (2005), concrete continues indefinitely, therefore measuring it in an experimental setup poses many challenges to obtaining long-term data [41]. On the other hand, computational methods to model viscoelasticity in concrete is not well studied and remains a large gap in the literature. Aside from the obvious time savings, computational approaches allow for the study of how various parameters impact the creep behavior, such as boundary conditions, aggregate properties, and the size of a sample.

1.1.1 Research Scope and Objectives

The overarching objective of this research is to build an experimentally validated computational framework to predict creep in concrete over several decades. This project includes the development of a random, virtual microstructure generation code, based on the Anm model [42, 43], that generates 3D concrete microstructures using real aggregate shapes obtained from X-Ray Computed Tomography (XCT) data [7] made available for this work by collaborators Edward J. Garboczi (National Institute of Standards and Technology) and Jeffery Bullard (Texas A&M University). The code will also allow for flexibility in aggregate size distribution in order to match sieve curves from experimental concrete mix designs. Additionally, the microstructure generation code can handle various boundary conditions, including periodic, cast, and cored boundaries. A large focus is placed on improving the efficiency of the detection of aggregate-aggregate intersection. The analytical equation describing the surface of the aggregate is leveraged in a novel overlap detection algorithm that successfully reduces the computational expense of generating a microstructure.

Generated concrete microstructures will then be converted to a finite element mesh using OOF3D [44]. The finite element solver Abaqus will then be used to simulate the viscoelastic response of the concrete over time. For simplicity, a stress relaxation simulation will be performed in Abaqus. The relationship between the relaxation modulus and creep compliance is simple in the Laplace domain, making the conversion between the two constituitive equations straightforward. Simulations utilizing the virtual microstructures will be performed on a variety of microstructure groups to compare and assess the influence of various factors including two-dimensional (2D) vs three-dimensional (3D) simulations and assessment of boundary conditions. To validate the finite element analysis results, comparisons to experimental concrete creep tests conducted in concurrent research is performed [45]. These experimental creep tests are conducted at two temperatures, producing data that will be combined using the TTS principal to extend the data into a long-term master curve.

Lastly, short-term simulations of mortar relaxation are performed to model fourteen days of mortar creep. Two mortar mix designs are assessed - one loaded at early age and one loaded at maturity. Experimental data for the associated mortars has been made available for validation of the simulations [46].

The specific tasks of this proposed research are summarized below:

1. Write an efficient particle placement code, based on the Anm model [42, 43], to pack reconstructed aggregates into a domain at volume fractions of 38%. The code shall have the ability to generate microstructres in both two and three dimensions using cast, cored, and periodic boundary conditions.

2. Develop a novel overlap detection algorithm that utilizes the asperities on the aggregate's surface. Quantify time savings provided by this novel asperity overlap detection method on several different aggregate volume fractions and size gradations.

3. Use finite element analysis (FEA) to simulate decades of basic creep of concrete by upscaling

mortar experimental creep data paired with virtual concrete microstructures.

4. Perform RVE size and finite element mesh density convergence studies to determine the optimal domain size and number of finite elements

5. Validate finite element analysis results with experimental concrete creep data

6. Determine the viability of using 2D simulations to model viscoelastic behavior

7. Assess the similarities and differences in viscoelastic behavior of 3D periodic, cast, and cored concrete microstructures

8. Model short-term mortar viscoelastic behavior through the use of virtual mortar microstructures and experimental cement paste data

1.1.2 Research Outline

This dissertation is organized as follows. Chapter 2 discusses the previous and current publications relevant to this work. Section 2.1 provides an overview of published concrete microstructure generation codes, while Section 2.2 reviews a variety of computational models used to simulate the mechanical behavior of concrete.

Chapter 3 walks through the code developed to generate the virtual microstructures by randomly placing reconstructed aggregates into a domain without intersecting one another. The novel overlap detection algorithm is described in Chapter 4, where the approach to identifying and using an aggregate's surface asperities to determine if it overlaps with another aggregate is described in detail. The improvements in efficiency are demonstrated in Section 4.3.

The generated microstructures are put to use in FEA simulations in Chapters 5 - 7. Chapter 5 describes the use of 2D and 3D periodic microstructures to simulate 10,000 days of concrete creep, showcasing the results alongside experimental concrete creep data. The impact of boundary conditions, both during the microstructure generation process and the imposition of periodic stress and strain fields is explored in Chapter 6. Finally, the simulation of mortar stress relaxation is presented in Chapter 7.

2. LITERATURE REVIEW

This chapter discusses the background information and published research in the areas of study related to the objectives of this dissertation. The primary areas covered in this chapter are the generation of virtual concrete microstructures, numerical analysis of cement-based material to measure and predict mechanical properties, influence of dimensionality, simulation of viscoelastic behavior, and impact of RVE size and boundary conditions on the predicted mechanical behavior.

2.1 Generation of Random Concrete Microstructures

The precursor to any numerical simulation is a recreation of the material or structure to be studied. Composite materials present challenges when depicted virtually due to the complexity of accurately representing a heterogeneous material. As a particulate composite material composed of aggregates suspended in a cementitous binder, concrete presents challenges when generating an accurate computational representation due to the irregular shapes of aggregates, their random spatial distribution, and size gradation.

A wide variety of approaches have been taken to represent concrete in numerical simulations. Yang et. al described how these approaches can be divided into two general methods: image-based or computer generated, as shown in Figure 2.1 [3]. Image-based methods include XCT scans and series of optical images. The quality of the reconstructed microstructure depends heavily on the resolution of the the instruments used to image the material. Computer-generated models, on the other hand, alleviate this limitation by providing an analytical description of the aggregate's shape. Computer generated models are the focus of the work presented in this dissertation.

2.1.1 Particle Packing Algorithms

Particle packing and spatial arrangement play important roles in many fields of research and industrial applications. Codes generating virtual representations or particles and composite materials have gained immense popularity for use in simulating mechanical behavior [47, 48, 4], fluid dy-

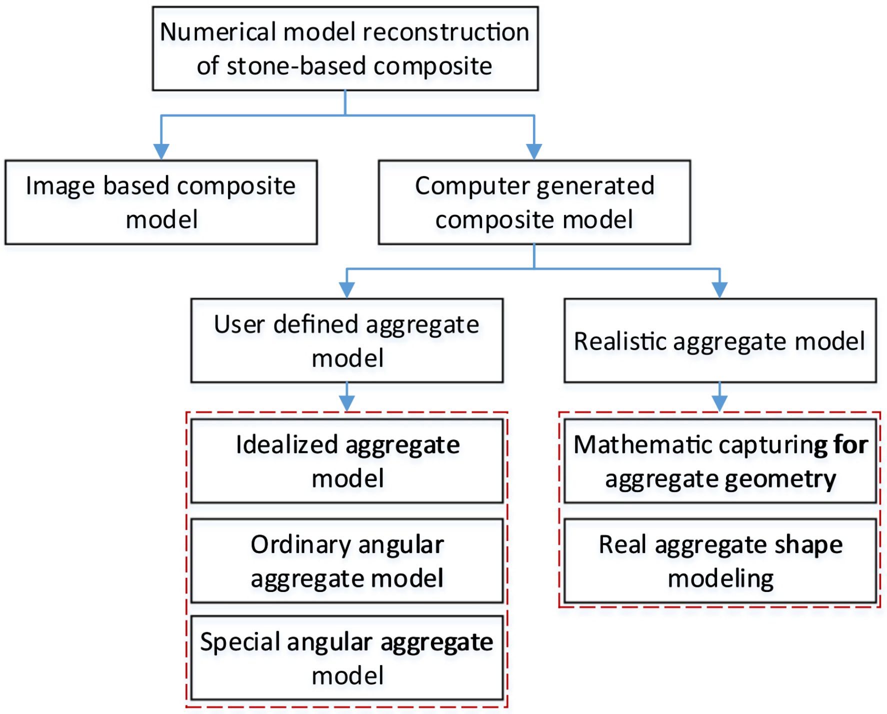

Numerical model reconstruction of stone-based composite
Image based composite model
Computer generated composite model
User defined aggregate model
Realistic aggregate model
Idealized aggregate model
Ordinary angular aggregate model
Special angular aggregate model
Mathematic capturing for aggregate geometry
Real aggregate shape modeling

Figure 2.1: Categories of numerical model recreations of composites [3]

namics [32, 49], permeability, and thermal conductivity [47, 50, 51, 22, 23]. The random packing and placement of regularly shaped particles such as spheres [22, 23, 24, 25, 26, 27], cubes [52], cylinders [32, 33, 34], and ellipsoids [28, 29, 30, 31] have been widely studied.

Efficient particle packing algorithms can be classified into several general groups:

1. Ballistic algorithms: Particles placed along a trajectory [53, 54, 55]

2. Random placement algorithms: Particles placed in a randomly selected location in the domain [56, 57, 43, 58]

3. Growth algorithms: Particles placed randomly, then slowly expanded until they touch or intersect another particle [59, 60].

4. Domain shrinkage algorithms: Particles plotted in a large domain, then moved closer to the

center as the domain decreases in size [61]

5. Advancing front algorithms: Particles are placed next to each other starting at a given location and subsequent particles are placed at the edge of the particle conglomeration [62, 63, 64, 65]

A review of the literature indicates that the packing of highly realistic aggregate shapes is primarily accomplished with random placement and growth algorithms [60, 57, 43]. The other methods presented above are generally used for packing regular shapes. In 2014, Qian et. al published the Anm model, a code that placed these reconstructed aggregates into a cubic domain [57]. This code was improved by the addition of parallelization in a second paper [43]. The Anm model generates microstructures representing cast samples and RVEs with periodic boundary conditions by randomly rotating and placing the aggregates into a cubic domain.

2.1.2 Overlap Detection

As in real life, two particles cannot exist in the same space at the same time, so prevention of virtual particle-particle overlap is an important aspect of particle placement codes. Particle-particle contact and collision has been extensively studied in for the application of discrete element method (DEM) analysis, as DEM measures interparticle forces [66]. Representations of particles in DEM typically utilizes sphere clusters [6, 67] or superquadric functions to describe regular shapes such as ellipsoids, cubes, and cylinders [68, 69]. Detection of overlap between spheres is the most straightforward, warranting only a comparison of the distance between the centers to the sum of the radii. Furthermore, overlap between cubes can be determined by evaluating if any of the vertexes or line segments lie within another cube. In the case of cylinder-cylinder overlap detection, a variety of approaches are used - including projection on a line, separation axis tests, and three-dimensional parallel testing [70, 71]. Contact between two ellipses can be determined by solving the quartic equation derived from combining the equations of two ellipses. Minimization of the geometric potential of the two eclipses can be used to locate intersection points, described in [72]. Additional contact and overlap detection algorithms for ellipsoids can be found in the literature

[30, 73, 74, 75, 76].

When placing or packing irregularly shaped particles, detecting overlap becomes challenging. Some applications combine regular shapes to form more complex particle morphologies - such as such as tablets [77, 78], clusters of spheres [6, 67, 68], and sphere-cylinder combinations [70, 77, 78]. Other methods of irregular particle shape representation include X-Ray Computed Tomography (XCT) scans [4], laser scans [79], random Voronoi polygons [80, 81, 82], and spherical harmonics [7, 83]. A review of published literature has shown the main approach to overlap detection of non-spherical particles is calculating the minimum distance between the surfaces of two particles [83, 61]. This provides accurate detection of overlap, but is computationally expensive. An alternative method involves representing particle shapes as a cluster of component spheres of differing sizes. Each sphere composing the particle in question is checked for overlap with the spheres composing the nearby particles, the accuracy of this method is dependent on the conformity of the sphere cluster to the original particle shape [84, 85]. Another method is to represent each particle's surface as a collection of planes, where the space enclosed by the intersecting planes represents the particle. The overlap test then checks if all the points on other particles lie on opposing sides of each of these planes from the particle in question [86]. This approach requires the use of particles that are able to be defined by a set of planes - a criteria that fits random polygons and similar shapes well, but does not suit some classes of particle morphology including non-convex particles, curved particles, or very irregular particles. Voxel based approaches treat particles as 3D images, composed of cubic voxels. Overlap detection is fast, however there is a tradeoff between the resolution of the particle's surface characteristics and reducing the number of voxels used [87]. A mathematically driven approach can be used to determine exactly where the two particle surfaces intersect, however this is computationally expensive [43, 88, 89]. In this dissertation, the focus is on determining *if* two particles overlap, not precisely *where* they overlap.

2.2 Computational Modeling of Concrete Mechanical Behavior

2.2.1 Impact of Aggregate Shapes

Aggregates in virtual concrete microstructures have been represented by a number of geometries, including the regular shapes such as circles [90, 91, 18, 92, 93, 94, 95] and spheres [96, 97, 98, 99, 100, 101], ellipsoids [102, 90, 103], polygons and irregular shapes [90, 104, 105, 106, 99, 107]. as well as shapes reconstructed from XCT scans [4, 42, 43]. An illustrative example from Bernachy-Barbe et. al (2019) shows concrete microstructures generated from a range of aggregate morphology representations in Figure 2.2.

The shape of aggregates has been found to influence a variety of concrete material properties, including diffusivity [39, 40], fracture patterns [35, 37], and mechanical behavior [35, 36]. Idealized aggregate models describe aggregates as smooth spheres or ellipsoids. While this is the simplest model for aggregate shape, the downfall of this methodology is that the surface texture and angularity is not captured by smooth, rounded aggregate representations. Research modeling aluminum alloys with silicon carbide (SiC) particle inclusions compared the representation of the SiC particles as spheres, ellipsoids, and real SiC particle shapes extracted from image analysis. Differences in the stress-strain curves are shown in Figure 2.3 [5]. In a step towards representing irregular shapes, clustered spheres have been used to represent irregular aggregate shapes in discrete element method (DEM) analysis techniques, such as the microstructure seen in Figure 2.4 [6, 68, 67]. Fu and colleagues found that the clustering technique of representing aggregates provided a much better prediction of shear strength and strain than individual spheres, confirming the need for representative aggregate morphologies. As such, this dissertation work employs the use of real aggregate shapes, recreated using spherical harmonics and XCT data [7]. Garboczi (2002), developed a code in FORTRAN to assess 3D micro-XCT data of both coarse and fine aggregate particles, calculating the coefficients of the spherical harmonics equations that could then be used to recreate the shape of the scanned aggregate on nearly any computer.

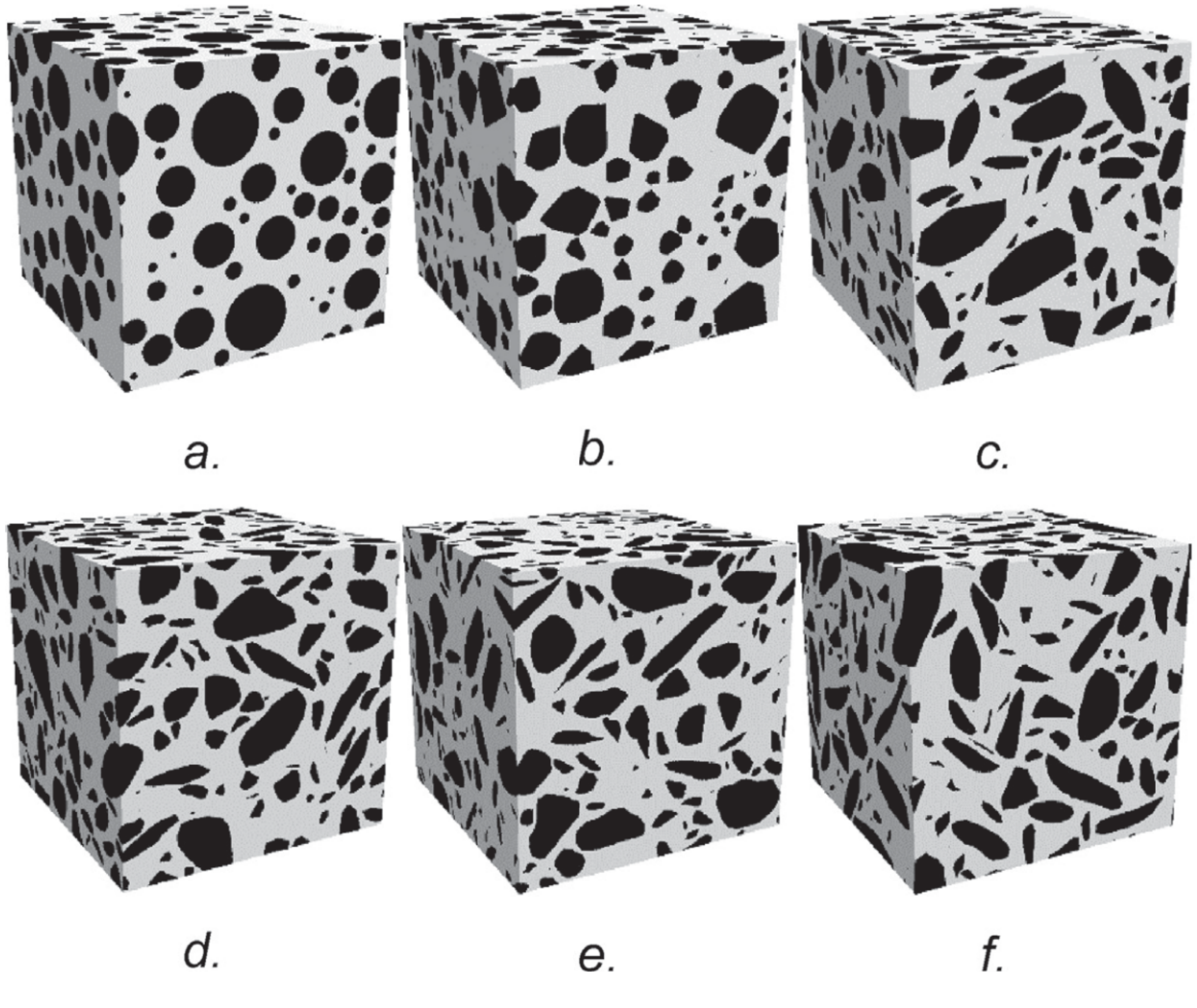


Figure 2.2: Concrete microstructures generated with a) spheres, b) Vornoi polygons, c) flattened Voronoi polygons, and d-f) real aggregate shapes extracted from XCT scans [4]

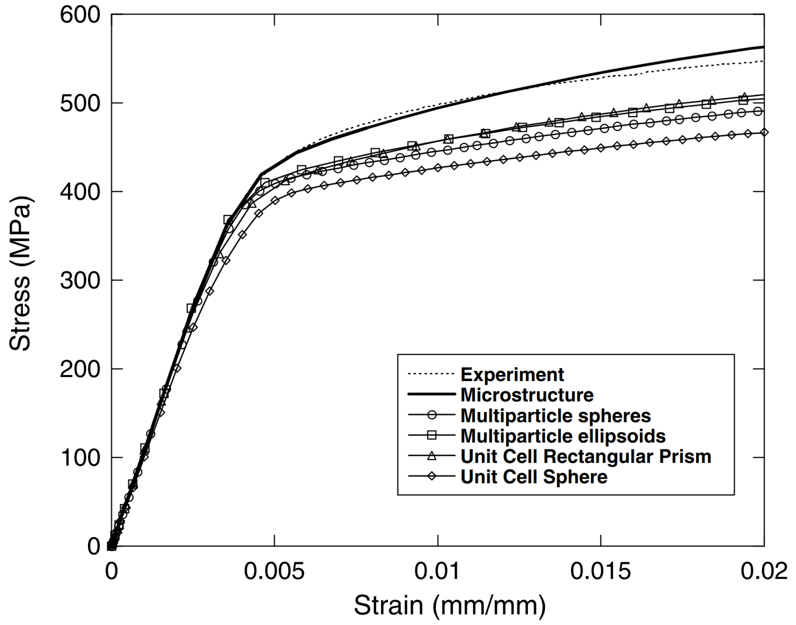


Figure 2.3: Modeling of metal matrix composite with real inclusions shapes (labelled 'microstructure'), spheres, and ellipsoidal inclusions [5]

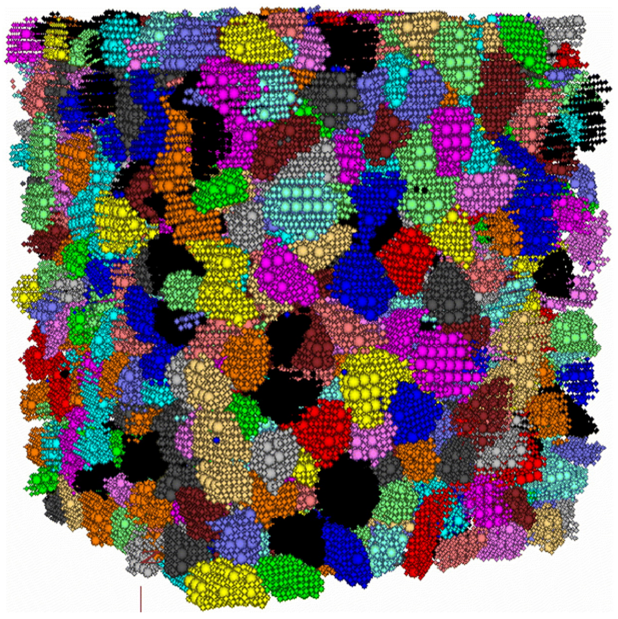


Figure 2.4: Example of clustered spheres of varying sizes used to represent aggregates [6]

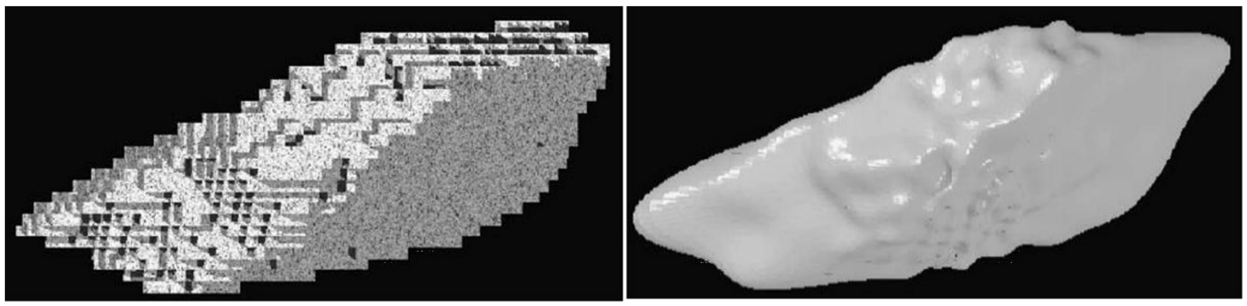


Figure 2.5: 3D digital image taken from XCT data (left), processed and reconstructed with spherical harmonics (right) [7]

In 2002, the concept of pairing spherical harmonics with XCT scan data was first published in Garboczi (2002). In this study, a code was written in FORTRAN to assess 3D micro-XCT data of both large and fine aggregate particles. Algorithms written in this code calculated the coefficients of the spherical harmonics equations that could then be used to recreate the shape of the scanned aggregate on nearly any computer [7]. Additional work has been done using aggregate shapes extracted from XCT data that shows the tomographic aggregates give an excellent prediction of experimental data, while isotropic and convex aggregate representations show softer behavior and lead to notable differences on both the macroscopic and local scale [4].

2.2.2 Two-Dimensional vs Three-Dimensional Modeling

Two-dimensional simulations have the advantage of short computational time, however finite element method (FEM) simulations have shown mixed results when predicting the stress-strain response of concrete [108, 19, 109, 110, 103]. Published literature has indicated that 2D simulations both do [110, 103, 111] and do not accurately predict the elastic properties of composites [108, 109, 8, 112]. It was hypothesized by Huang and colleagues that 2D microstructures do not capture the out-of-plane aggregate interlock effects that impact the material's response under loading [19]. As seen in Figure 2.6, the 2D images taken from a 3D microstructure do not accurately represent the Young's modulus when simulated in FEA [8].

Furthermore, computational research on materials other than concrete revealed that 2D simula-

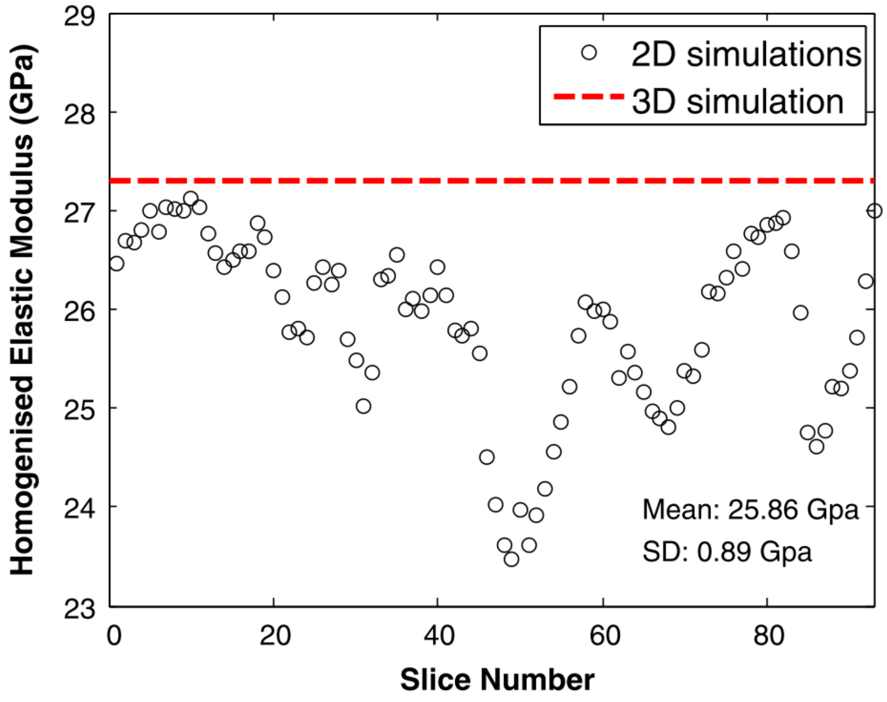


Figure 2.6: The Young's modulus calculated from an FEA simulations of a 3D microstructure and the stack of 2D microstructures taken from slices of the 3D microstructure. [8]

tions do not capture the changes in the stress field [111], defect prediction [113], fracture modeling [109], among others. Thus, additional work is needed to determine whether 2D or 3D simulations can be used to accurately model concrete creep behavior over extended periods of time. Additionally, given that concrete in nuclear containment structures undergoes a 3D state of stress, a 2D modeling technique may not be suitable. To the author's best knowledge, a comparison of viscoelastic behavior simulated in 2D and 3D has not been published at the time of this dissertation. Thus, additional work is needed to determine whether 2D or 3D simulations can be used to accurately model concrete creep behavior over extended periods of time.

2.2.3 Viscoelasticity

Viscoelastic materials exhibit both elastic and viscous behavior, giving their mechanical behavior a time-dependent component. The classical linear theory of viscoelasticity was first described by Ludwig Boltzmann for cases of three-dimensional isotropic materials [114]. Creep is a phenomenon that causes viscoelastic materials such as concrete to continue to deform over time while subjected to a constant stress. Concrete creep has been an area of study for many decades, over which time several studies have aimed to predict concrete creep [15, 16, 17]. Brooks (2005) showed that creep continues indefinitely, meaning computational approaches to measuring and predicting concrete creep are highly attractive [41].

Concrete creep is a combination of basic creep and drying creep [14]. Drying creep, also called the Pickett effect, results from the dehydration of the pores in the cement paste [115]. Basic creep occurs due to the viscous nature of the material, independent of the relative humidity profile in the sample.

Linear viscoelastic materials maintain a linear relationship between stress and strain at all points in time. Concrete remains a viscoelastic material until it is loaded past it's elastic strain limit. Creep compliance, $J(t)$, can be characterized in a variety of ways; such as a Kelvin-Voigt chain is used to describe the creep of concrete. The relationship between creep compliance, $J(t)$, and strain, $\epsilon(t)$, is given in Equation 2.1,

$$\epsilon(t) = \int_0^t J(t-t')\,\frac{\delta\sigma(t')}{\delta t'}dt, \tag{2.1}$$

where t is the present time and t' is the dummy time variable.

Viscoelasticity in concrete using three-dimensional simulations is not well researched in the published literature, a majority of the revelant publications are very recent. A study by Bernachy-Barbe and Bary in 2019 showed that 3D aggregate shapes extracted from XCT scans showed that the XCT aggregates showed the most realistic creep behavior when compared to experimental data. The spherical aggregates showed the most creep, followed by the Voronoi polygons, then finally the XCT aggregates [4]. Aydin et. al (2007) found that the use of irregularly shaped aggregates resulted in a higher resistance to creep than smooth, regular shaped aggregates in 2D concrete simulations, confirming the findings of Bernachy-Barbe and Bary (2019) [38]. On the other hand, Lavergne et. al (2015) found that spherical aggregates and random polygon shaped aggregates resulted in creep compliance predictions that were not statistically significant [116]. Barnachy-Barbe and Bary (2019), Aydin et. al (2007) and Lavergne et. al (2015) all performed simulations on cubic microstructures. A lone publication simulating cylinders was located. Wang et. al (2019) simulated 90 days of creep 3D cast cylinders filled with spherical aggregates [117]. Good agreement with experimental data was shown, however the source or procedures used to obtain the experimental data was not reported. Given the minimal literature available on this topic and the disagreement in the results, more study is needed.

2.2.4 Influence of Boundary Conditions on Mechanical Properties

Cored and cast cylinders have been studied in numerous publications [118, 119]. Additionally, cylinder and cube concrete specimens have been contrasted in a number of papers [119, 120, 121, 122, 123, 124, 125]. Although the topic of experimental sample preparation has been an area of focus in the civil engineering community, study regarding the RVE size and impact of sample geometry on viscoelastic behavior has not been well studied. Published work regarding the RVE size and mechanical properties of cast and cored concrete samples is discussed further in following two sections.

2.2.4.1 Representative Volume Element (RVE) Size

In regards to concrete strength, it is widely understood that cored cylinders have a lower compressive strength than cast cylinders [118, 119]. The rule of thumb is that the width or diameter of the concrete sample should be at least 3x larger than the largest aggregate size [14]. While this guideline has been followed for many years, little work has been performed to determine the minimum sample size needed to accurately assess Young's modulus, and even fewer publications have assessed other mechanical properties, such as viscoelastic behavior.

2.2.4.2 Mechanical Properties

As shown in Figure 2.7, a different exists in the measured compressive strength between cast and cored concrete cylinders. Cored samples have been found to be equivalent to cast samples tested at a younger age [118], resulting in compressive strengths up to 30% lower than the cast

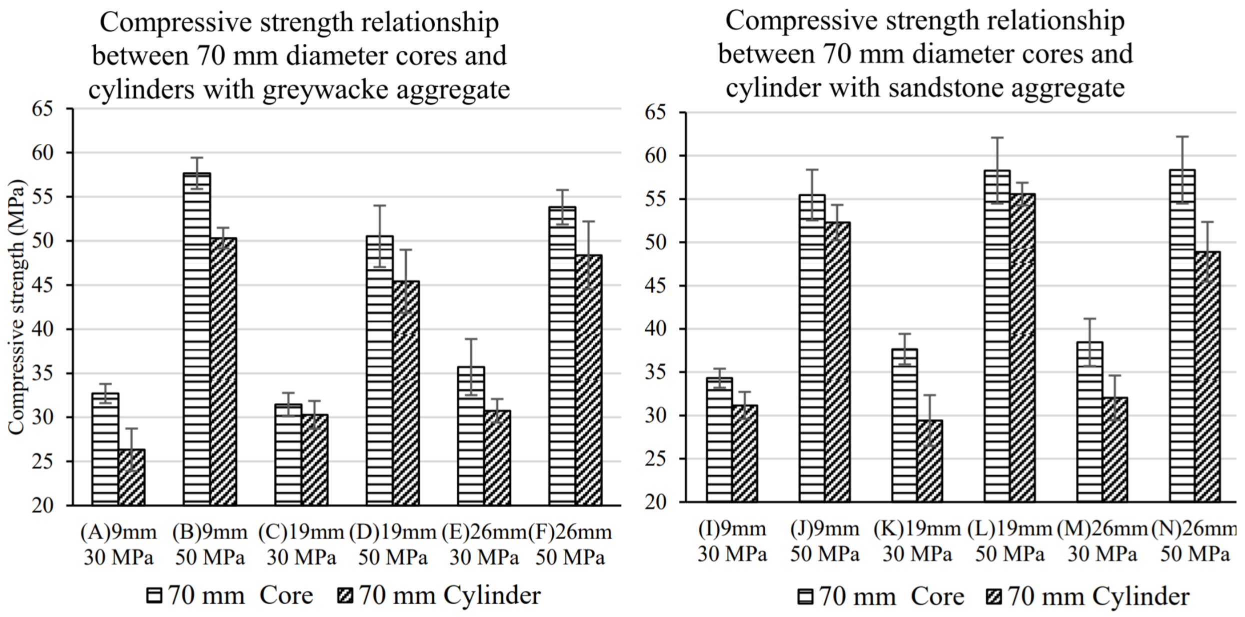


Figure 2.7: Relation between cast and cored cylinder compressive strength [9]

cylinder counterparts. Cast samples will have wall effect, as seen in Figure 2.8, meaning that the volume fraction of aggregates is low around the surfaces of the sample, which may contribute to the difference in compressive strength between cored and cast samples. When comparing the strength of cylindrical cored samples to cube samples, there was no significant different in the compressive strength [9]. A study by Ergun et. al (2012) found that a 19% difference in the compressive strength between cored and cast cylindrical samples, hypothesizing the difference was caused by damage inflicted on the cored samples during the drilling process [119]. This same study found a 16% difference between cube and cast cylindrical specimens [119]. It was noted in Smith (2017) that as the diameter of the cylinders increased the compressive strength began to converge [9]. Although the topic of experimental sample preparation has been an area of focus in the civil engineering community, study regarding the RVE size and impact of sample geometry on the stiffness and viscoelastic behavior has not been well studied.

The relationship between these methods of sample preparation and viscoelastic behavior remains a gap in the literature. To the author's best knowledge, publications on this topic could not be

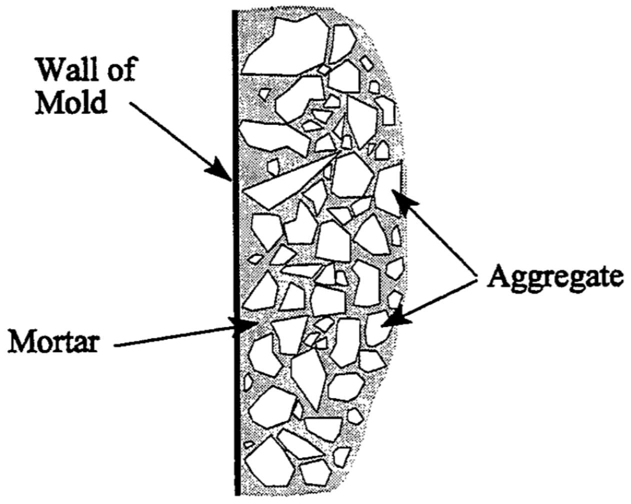


Figure 2.8: Wall effect observed in cast samples [10]

located.

2.3 Gaps in the Current Literature

From a review of the current scientific literature, a few notable gaps can be identified. These gaps are addressed in the subsequent chapters of this dissertation.

The generation of virtual concrete microstructures is not a new area of research, however the packing of highly realistic, reconstructed gravel shapes presents a novel challenge of rapidly identifying particle-particle overlap. Given the evidence that aggregate shape plays an important role in the simultion of mechanical properties, it is important to incorporate real aggregate shapes into virtual concrete microstructures. A vast array of approaches to particle-particle overlap have been developed and utilized in the literature, however they are largely unsuitable in the application at hand. A computationally inexpensive, accurate method to determine overlap between two virtual aggregate particles is needed.

The next gap is the simulation of concrete viscoelastic behavior. As concrete creep is a very slow and continual process [41], measurements collected via computational methods have many advantages, however, only a handful of publications have been located that perform simulations to measure concrete creep [38, 4, 116, 117]. The impact of simulation dimensionality on the creep prediction is unclear, as it has not been demonstrated in the published literature, to the best of the author's knowledge.

Sample preparation's impact on concrete strength is extremely well documented [119, 120, 121, 122, 123, 124, 125], however no studies could be found that compared the creep or relaxation of concrete samples prepared via cast and cored methods. =

These gaps can be summarized into the following main points:

- Rapid test for overlap between two irregularly shaped virtual particles
- Modeling of concrete viscoelastic behavior via virtual concrete microstructures
- Appropriateness of 2D and 3D computational models to predict creep
- Impact of boundary conditions on concrete creep

3. GENERATION OF RANDOM, VIRTUAL CONCRETE MICROSTRUCTURES

The precursor to many numerical simulations is a virtual recreation or development of a digital twin or statistically representative surrogate of the material or structure to be studied. Particles and particulate composite materials present challenges when depicted virtually due to the complexity of accurately representing the morphology of the particulate components. As discussed in Chapter 2, the morphology of particles has been shown to play a role in numerous material properties, thus real aggregate shapes are utilized in this work. This chapter describes the process of reconstructing real aggregate shapes and using them to generate realistic cubic and cylindrical concrete microstructures.

3.1 Reconstruction of Aggregates

Realistic three-dimensional aggregate shapes are reconstructed from micro-XCT data using a mathematical approach termed spherical harmonics, an approach first utilized in Garboczi 2002 [7]. The data for these reconstructions was shared for use in this work by Garboczi. This is a technique that uses associated Legendre functions and cosines to mathematically deform a sphere in a spherical coordinate system. The spherical harmonics series is defined as

$$Y_n^m(\theta, \phi) = \sqrt{\frac{(2n+1)(n-m)!}{4\pi(n+m)!}} P_n^m(cos(\theta))e^{im\phi}, \tag{3.1}$$

where the term P_n^m represents the associated Legendre functions and i is the square root of -1, also termed the imaginary number. In Garboczi 2002, spherical harmonics was applied to XCT scans of individual aggregates using a FORTRAN code that calculated the coefficients to fit Equation (3.1) to the morphological shape in the XCT scan [7]. These coefficients, a_{nm}, are used to define the radius in a spherical coordinate system through the relationship:

$$r(\theta, \phi) = \sum_{n=1}^{n_{max}} \sum_{m=-n}^{n} a_{nm} Y_n^m(\theta, \phi), \tag{3.2}$$

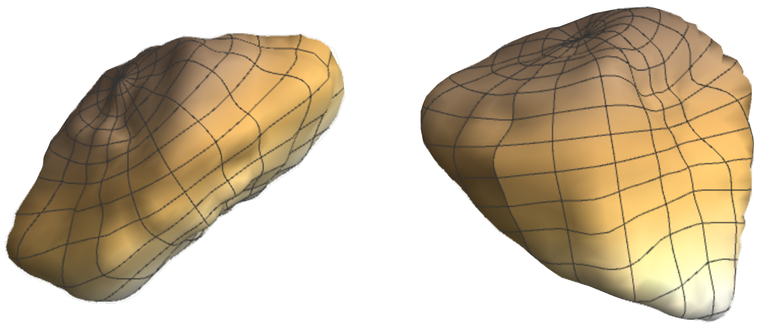


Figure 3.1: Three-dimensional aggregates reconstructed from XCT data using spherical harmonics

where the azimuth angle, θ, ranges from 0 to 2π and the polar angle, ϕ, ranges from 0 to π. It is recommended that Equation (3.2) is carried out to an n_{max} value of approximately 18 to 26. Additional summation terms will not generate an appreciable level of additional detail and may even go beyond the level of detail captured in the XCT scans [7]. Spherical harmonics represents the three-dimensional morphology of irregular, star-shaped particles through the a_{nm} coefficients, which are unique for each n and m pair. To be considered star-shaped, a particle must contain at least one interior point, such as the center of mass, from which a line segment connecting any surface point to this point is entirely inscribed within the particle. Each aggregate reconstructed and used in this work is star-shaped. The aggregates reconstructed from spherical harmonics are defined in three-dimensions at any location in a spherical coordinate system. Three reconstructed aggregates are shown in Figure 3.1. To obtain aggregates in two-dimensions, the value of θ in Equation (3.2) is set to $\frac{\pi}{2}$, thus defining the surface of the aggregate in the $\theta = \frac{\pi}{2}$ plane

3.1.1 Pre-Rotation of Aggregates

To reduce the computational time needed to generate a microstructure, a stockpile of pre-rotated aggregates was created. Each aggregate was used to make 10 pre-rotated copies of itself. This was carried out by rotating the a_nm coefficients by a random degree around each the X-, Y-, and Z-axis. The a_nm were rotated using the code provided in Appendix A.1. With the new coefficients, the pre-rotated aggregates were then re-constructed using Equation (3.2).

3.2 Particle Placement Algorithm

An aggregate's length is defined by the longest line segment connecting two points on the surface that lies entirely within the aggregate. The width is the longest axis perpendicular to the length axis, as described in ASTM D4791 [126]. Using the size of the openings on standard aggregate sieves, aggregates are sorted by their width into sieve size groups [14]. A sieve curve, also called the sieve gradation, provides information about the percent of aggregates in the stockpile that pass through each sieve. The mix design for concrete is the source of information regarding the relative percentages of the fine aggregates and coarse aggregates in the mix. This particle placement code makes use of experimental sieve curves by calculating the volume fraction of coarse aggregate that come from each sieve size range. The largest aggregate size used in this work measures approximately 30mm.

To generate a sample of virtual concrete, the sample must be 'mixed', this process involves placing coarse aggregates randomly into a square or cubic domain and ensuring that they do not overlap with any previously-placed aggregates. The 2D and 3D microstructures are generated using the same procedures, the difference being the 2D aggregates are slices of the 3D aggregates at $\theta = \frac{\pi}{2}$.

The code starts with the largest sieve size and works down the sieve gradation curve to sequentially smaller sieves. During the particle placement loop, the instantaneous volume fraction is used to determine which sieve an aggregate will be randomly selected from. Center coordinates are randomly selected to fall within the domain, chosen by Matlab's internal random number generator. Using these values, the aggregate is translated to it's potential location in the domain. From there, the aggregate is checked for overlap with neighboring aggregates, a process described in more detail in the Chapter 4. The microstructure generation code exists in several versions, based on the desired boundary conditions, as described in the subsequent sections.

Concrete test samples prepared in a standard laboratory environment are typically made by either casting concrete into a mold or coring a sample from a larger piece of concrete. Generated

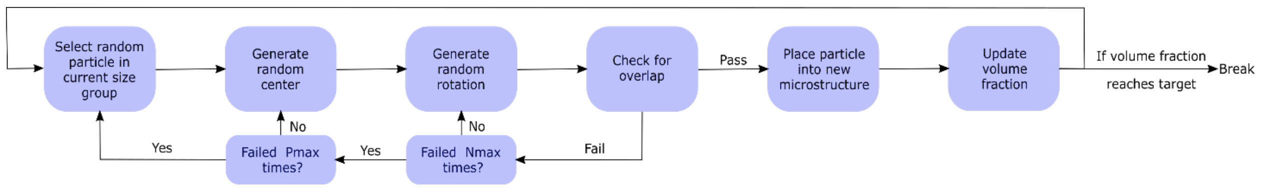


Figure 3.2: Particle Packing Algorithm

microstructures used for the bulk of this research are plotted with periodic boundary conditions, which reduces the size of the RVE, but is not physically replicable. In the interest of studying the potential difference in viscoelastic behavior between cast and cored samples, the plotting code is adjusted to add these two boundary condition options. The cast microstructure plots will forbid aggregates from contact with the domain boundaries. Cored microstructures will allow aggregates to intersect the boundary but the portion of the aggregate that lies outside the domain will be deleted. This is described in the proceeding sectiosn

3.2.1 Periodic Boundary Conditions

Periodic boundary conditions are used in this work to prevent wall effects and minimize the size of the representative volume element (RVE). A periodic RVE is the smallest volume that represents the morphology in an infinite pattern [127]. An important aspect of periodic RVEs is that opposing boundaries are identical to one another. This is achieved by cutting any aggregate that intersects the boundary and translating the external piece to the internal face of the opposing boundary.

Periodic microstructures enforce periodicity, meaning the microstructure is continuous between opposing boundaries. Aggregates that intersect a boundary are cut and the external piece(s) are translated to the internal side of the opposing boundary or boundaries. An example periodic boundary in two dimensions is demonstrated in Figure 3.3.

An example of completed 2D and 3D periodic microstructures are seen in Figure 3.4.

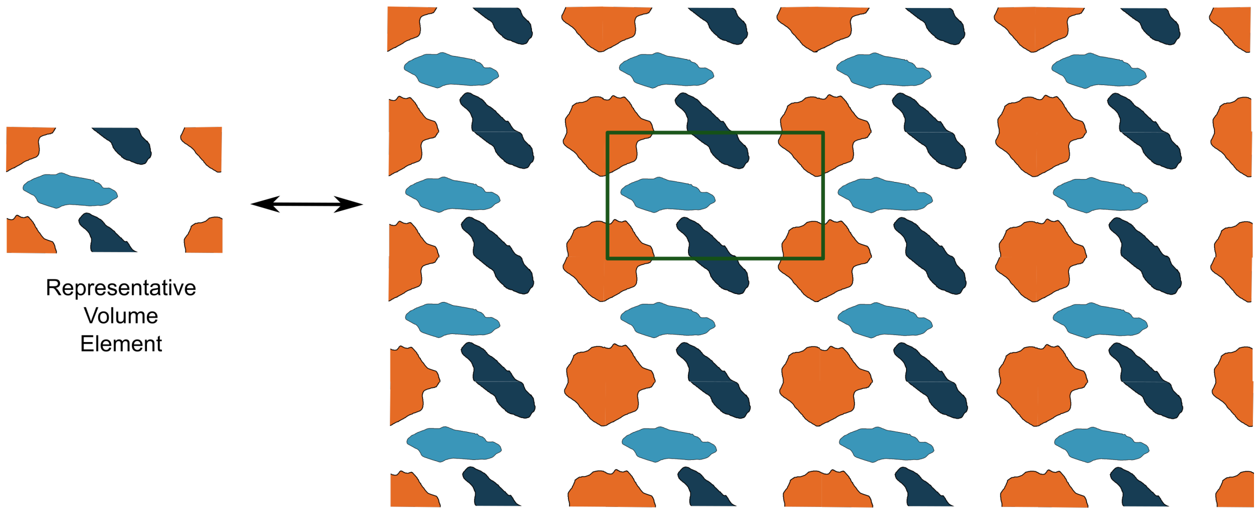


Figure 3.3: Visualization of periodic boundary conditions and a representative volume element (RVE)

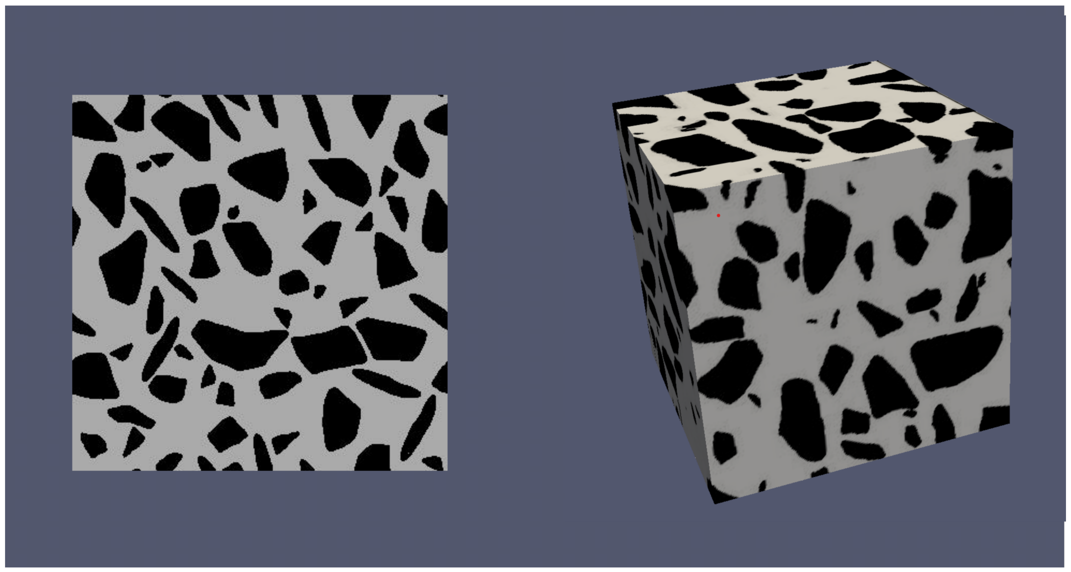


Figure 3.4: A 2D microstructure (left) and a 3D microstructure (right) with periodic boundary conditions composed of 38% coarse aggregates by area and volume, respectively.

3.2.2 Cored Boundary Conditions

The cored microstructures permit intersection between the aggregates and the walls of the cylinder. As each aggregate is placed into the domain, it is 'cored', meaning any external portions of the aggregate surface are cut and deleted. In cubic domains, this simply involves removing portions of the surface that contain coordinates in any direction that are less than zero or greater than the domain edge length. Figure 3.5 shows an example of both a 2D and 3D cored microstructure.

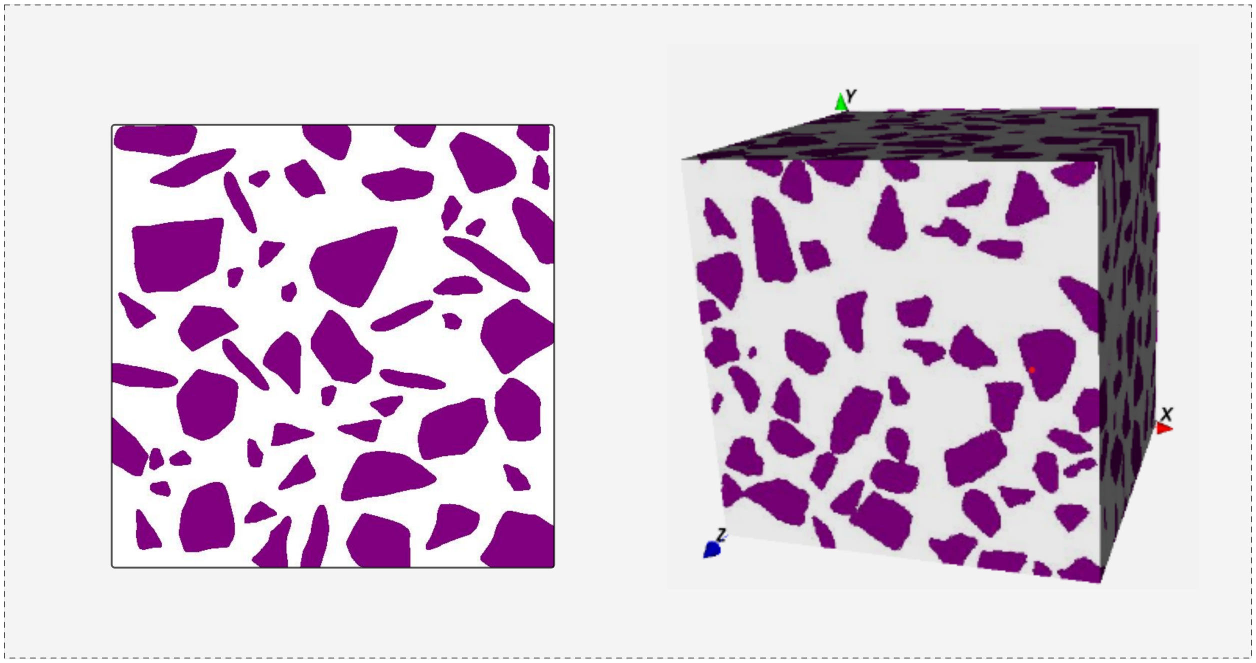


Figure 3.5: A 2D microstructure (left) and a 3D microstructure (right) with cored boundary conditions composed of 38% coarse aggregates by area and volume, respectively.

3.2.3 Cast Boundary Conditions

As many concrete samples prepared in both laboratory and field settings are cast cylinders and cubes, a version of the code was developed to generate microstructures that represent these samples. Cast microstructures do not permit any aggregates intersecting the boundaries of the domain. Intersection with the boundaries is assessed in the same manner as the cored microstructure generation, however instead of erasing the external portion of the aggregate, a flag is raised and the aggregate in question is translated to a new, random location in the domain and are checked for intersection with the boundary again. This process repeats until the aggregate is placed in a location where it is fully contained within the domain. In order to minimize the number of tries to find center coordinates where the aggregate is fully contained in the domain, aggregates are subjected to an additional constraint when center coordinates are randomly selected. The inscribed sphere radius, also referred to as the minimum radius, is the closest the center coordinates can be to the domain boundaries, because if the distance from the center of the aggregate to the boundary is less than the minimum radius, there is no change the aggregate is fully contained in the domain. An example of cast boundary conditions applied to both a 2D and 3D microstructure is seen in Figure 3.6.

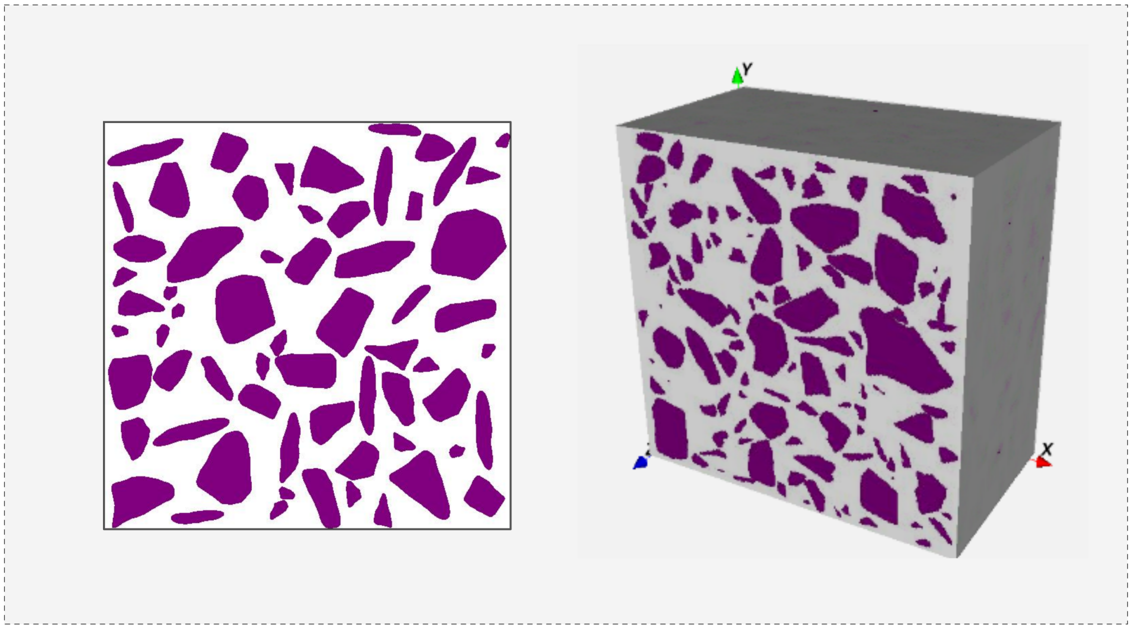


Figure 3.6: A 2D microstructure (left) and a 3D microstructure (right) with cast boundary conditions composed of 38% coarse aggregates by area and volume, respectively.

3.2.4 Cylindrical Microstructures

An additional version of the generation code was created to produce cylindrical microstructures in order to represent the cylinder samples commonly used in laboratory and field testing. In order to do this, the code was adjusted to consider the domain as a cylinder, rather than a cube. The center axis of the cylinder is taken to be a line parallel to the Z-axis. A user-defined diameter is used to set the height-to-diamter ratio. Periodicity in a cylindrical system is not possible, so the code prescribes either cast or cored boundary conditions.

To generate a cylindrical microstructure, a rectangular cuboid is generated as the initial domain, where the height of the cylinder is equal to the height of the cuboid. A user-defined radius delineates the outline of a cylinder centered in the XY plane around an axis parallel to the Z-direction. As center coordinates for the aggregates are randomly chosen, they are restricted to fall only within this cylinder.

To generate a cored cylinder, the aggregate placement process proceeds as it does with the cubic microstructure generation, however after an aggregate is successfully placed, it is ’cored’ by removing any part of the surface that is further from the center axis than the defined radius. The volume of the aggregate is then re-calculated from the remaining portion using Matlab’s built-in function, alphaShape. The volume fraction is then updated and the code proceeds to place the next aggregate. Cylindrical microstructures require two assessments to locate surface points outside the domain - the first determines the distance in the XY plane from the center of the cylinder, followed by the second assessment that checks if the Z-coordinates fall between zero and the domain height.

In the case of cast cylinders, during the aggregate’s random rotation and translation, the code checks if any points on the surface exist outside the cylindrical boundary. If this is found to be true, the aggregate is then rotated and/or translated again until it can find a location where it lies entirely in the cylindrical domain, then it is assessed for overlap. When randomly selecting the center coordinates for the aggregates, the code restricts the center to fall within the cylinder, but also that the center coordinates of the aggregate have a minimum distance from the cylinder’s surface equal to that of the aggregate’s minimum radius. This is because if the center is closer than this minimum radius, there’s no possibility that it lies entirely within the domain. This is a quick check that helps improve the efficiency of the cast microstructure generation code. An example of aggregates packed into cylindrical microstructure is seen in Figure 3.7.

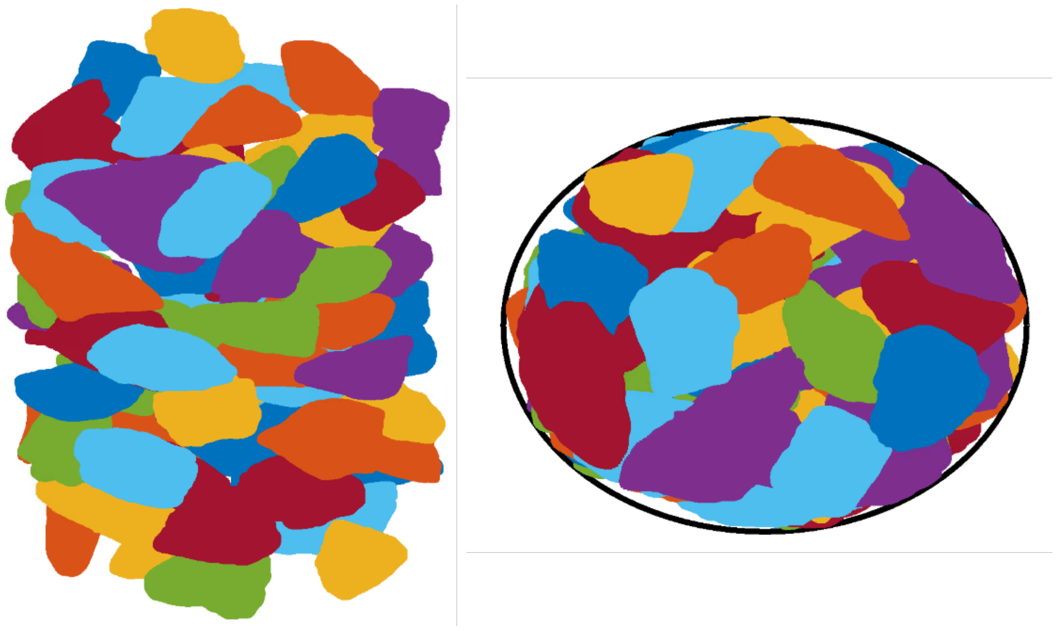


Figure 3.7: A cast cylindrical microstructure viewed from the side (left) and from the top (right), where all aggregates can be seen to be fully contained within the cylindrical domain

3.3 Summary

2D and 3D microstructures are generated by randomly placing virtual aggregates into a domain. Specifically, 2D microstructures can be generated using a square domain with either periodic, cored, or cast boundary conditions. Similarly, versions of the code exist to generate 3D cubic domains with periodic, cored, or cast boundary conditions. The codes to generate these microstructures can be found in Appended A. A 3D cylindrical generation code was developed as well, utilizing either cast or cored boundaries with a user-defined height to diameter ratio.

Much flexibility in this code can be found through various domain shapes and boundary conditions allowing for the generation and study of a wide variety of concrete microstructures. The development of a novel overlap detection method used to increase the efficiency of the microstructure generation code is presented in the next chapter. The succeeding chapters of this dissertation are investigations that stem from the work presented in this chapter by utilizing the generated microstructures. Impact of dimensionality and morphological boundary conditions are studied in

Chapters 5 and 6, respectively. Additionally, mortar microstructures are generated and simulated in Chapter 7.

4. ASPERITY-BASED VIRTUAL PARTICLE-PARTICLE OVERLAP DETECTION ALGORITHM

Detection of overlap between two particles is not a new area of research, as it has applications in the fields of molecular dynamics, DEM, video games, and more, as discussed in Chapter 2. However, detection of intersection of two irregularly shaped particles remains a challenge. In this section, a novel approach to detect overlap between two irregularly shaped virtual particles is described. This method identifies a priori the asperities on each aggregate's surface, corresponding to the most likely points on the aggregates where overlap will occur. Discussion and demonstration of this method are applied to coarse aggregates in this dissertation, however this approach can be implemented in any system involving particles with surfaces described by an analytical function.

4.1 Asperity Detection

The first and second spatial derivatives of the analytical function describing the surface of an aggregate, given in Equation (3.2), are used to identify the local maxima in the particle's radii on the particle's surface, called asperities. These maxima can be detected by first determining the locations where the first derivatives with respect to ϕ and θ are equal to zero. For brevity, the parameter f_{nm},

$$f_{nm} = \sqrt{\frac{(2n+1)(n-m)!}{4\pi(n+m)!}}, \tag{4.1}$$

is used to represent the factorials common to all the derivative equations [7]. The first derivative with respect to ϕ,

$$r_\phi = \sum_{n=1}^{n_{max}} \sum_{m=-n}^{n} (im)a_{nm}Y_n^m(\theta,\phi), \tag{4.2}$$

and the first derivative with respect to θ,

$$r_\theta = \sum_{n=1}^{n_{max}} \sum_{m=-n}^{n} \frac{-a_{nm}f_{nm}}{sin(\theta)}[(n+1)cos(\theta)P_n^m - (n-m+1)P_{n+1}^m]e^{im\phi}, \tag{4.3}$$

are equal to zero at locations of minima and maxima in the radius. Next, the second derivative with respect to ϕ,

$$r_{\phi\phi} = \sum_{n=1}^{n_{max}} \sum_{m=-n}^{n} (-m^2) a_{nm} Y_n^m(\theta, \phi) \tag{4.4}$$

and the second derivative with respect to θ,

$$\begin{aligned} r_{\theta\theta} = \sum_{n=1}^{n_{max}x} \sum_{m=-n}^{n} (\frac{a_{nm} f_{nm}}{sin^2(\theta)}[(n+1+(n+1)^2 cos^2(\theta))P_n^m \\ -2cos(\theta)(n-m+1)(n+2)P_{n+1}^m + \\ (n-m+1)(n-m+2)P_{n+2}^m]e^{im\phi}, \end{aligned} \tag{4.5}$$

are assessed [7]. The second angular derivatives are positive in minima locations and negative in maxima locations. Locations in the spherical coordinate system where the first derivatives are zero and the second derivatives are negative are identified as asperities and are stored for each particle. The asperities are unchanging on the particle surface, therefore the identification of the asperity locations is only performed once. That is, coordinates of the asperities are properties of the individual particle *not* the microstructure, and can thus be identified prior to any microstructure generation. The location data are stored and recalled as needed during the particle placement process.

Due to the computational expense of solving the equations analytically, the asperities on the surface of each particle are detected through numerical assessment, performed in Mathematica, of incremental values of θ and ϕ. A small tolerance above and below zero is used when analyzing the first derivative values, as the θ and ϕ increments may not fall on the precise location where the first derivatives are equal to zero. Identified asperities are saved in a text file for recall during the particle placement process. An example of an aggregate with identified asperities is seen in Figure 4.8.

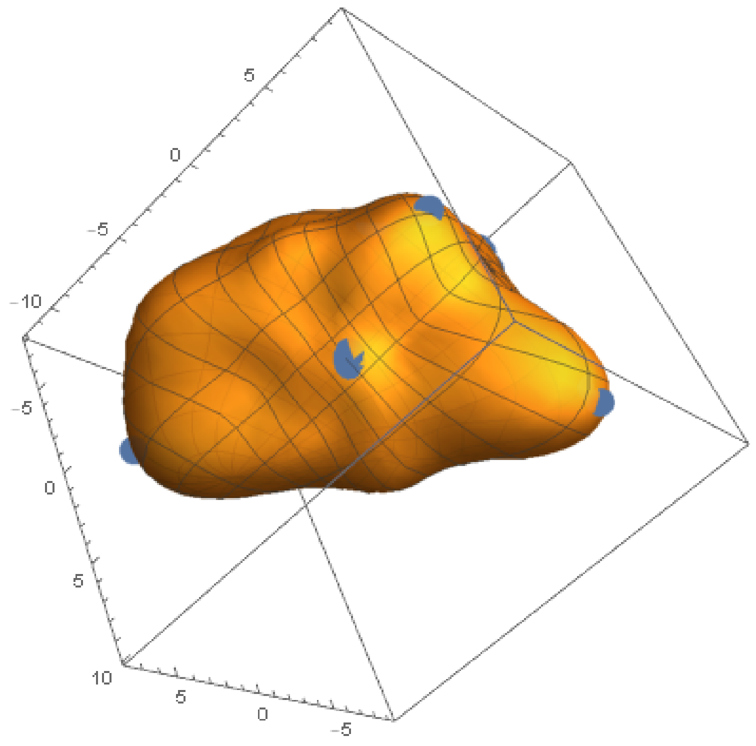


Figure 4.1: Asperities identified on an particle

4.2 Overlap Check

Detection of overlap between two virtual aggregates in this work involves a three-step approach. The overlap check function is divided into three filters. The first, coarse filter is to compare the distance between the centroid of the particle in question and the centroid of a previously placed particle to the sum of the two particles' maximum radii. An enclosing sphere, as opposed to an enclosing box used in [43], was chosen due to compatibility with a spherical coordinate system. This approach is very fast, however, it is also very coarse and if used alone it leads to false detection of overlap because it treats the particles as spheres. Next, the asperities on the surface of the two aggregates are assessed, followed by the third filter that applies a brute force check if necessary. This process is visualized in Figure 4.2

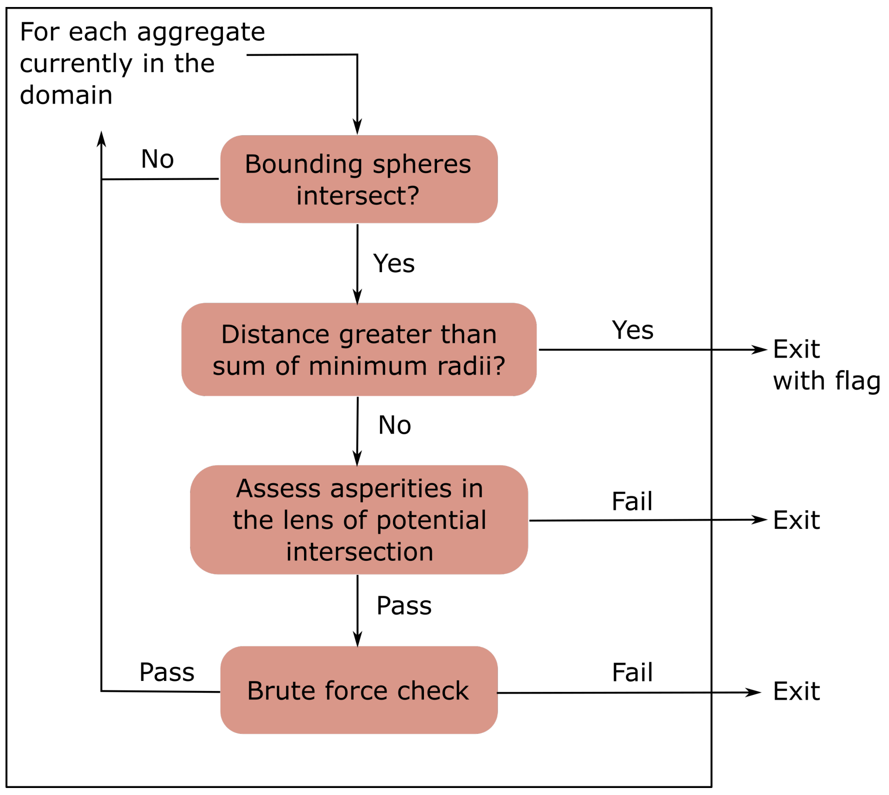

For each aggregate currently in the domain
Bounding spheres intersect?
No
Yes
Distance greater than sum of minimum radii?
Yes
Exit with flag
No
Assess asperities in the lens of potential intersection
Fail
Exit
Pass
Brute force check
Pass
Fail
Exit

Figure 4.2: Overlap check algorithm

4.2.1 Radii Comparison

Each particle has an imaginary bounding sphere with a radius equal to its maximum radii and a minimium inscribed sphere with a radius equal to its minimum radii. The bounding sphere can be imagined as the smallest sphere centered at the aggregate's center of mass that fully contains the particle. The inscribed sphere is the largest sphere that lies entirely in the interior of the aggregate's surface while also being centered at the aggregate's center of mass. If the distance between the centroids is greater than the sum of the maximum radii, the two particles can't possibly overlap. On the other hand, if the distance between the centroids is less than the sum of the maximum radii of the respective particles, the function computes the centroid distance to the minimum radii. If the distance between the two centers is less than the sum of their minimum radii, it is not possible that the two particles are free of overlap and a flag is raised. This flag is used to bypass rotation of the particle in question and jump to translating it to a new, random location within the domain. If the minimum radii assessment is passed, the algorithm advances to the second filter, described in the proceeding section.

4.2.2 Asperity Check

The overlap between the two enclosing spheres forms a lens of potential intersection as seen in Figure 4.3. This lens represents the only 3D space where overlap between the two particles is possible. Only the asperities located in the lens of potential intersection are assessed in this filter.

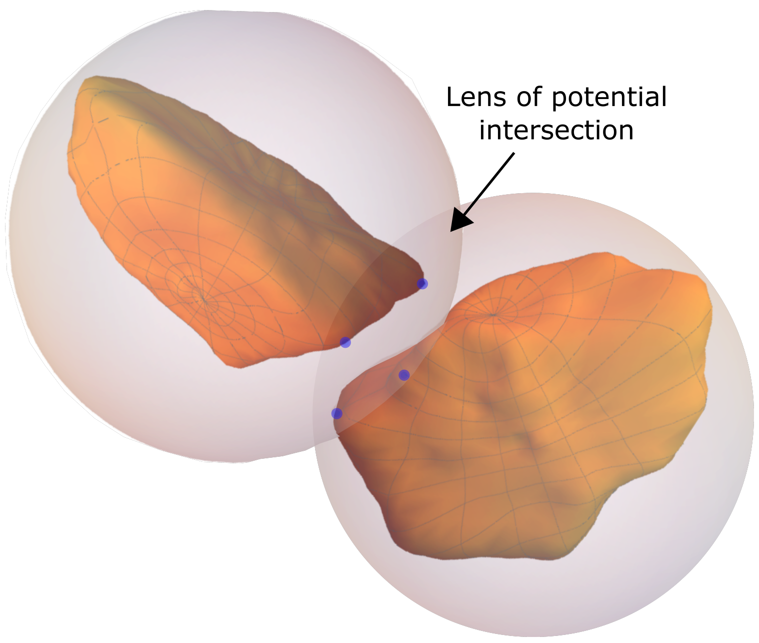


Figure 4.3: Lens of potential intersection formed by intersecting bounding spheres. Only asperities that lie in this lens are checked for overlap with the neighboring particle. In this example, the asperities that lie in this lens are identified with blue markers.

Starting with the particle in question, the asperities present in the lens of potential intersection are identified from the previously stored list of all the surface asperities. The locations are retrieved and the vector between the asperity and the center of the other particle is established. This vector's magnitude is compared to the radius of the other particle at the exact location, as shown in Figure 4.4. If the radius is shorter than the vector's length, there is no overlap at that asperity and the code

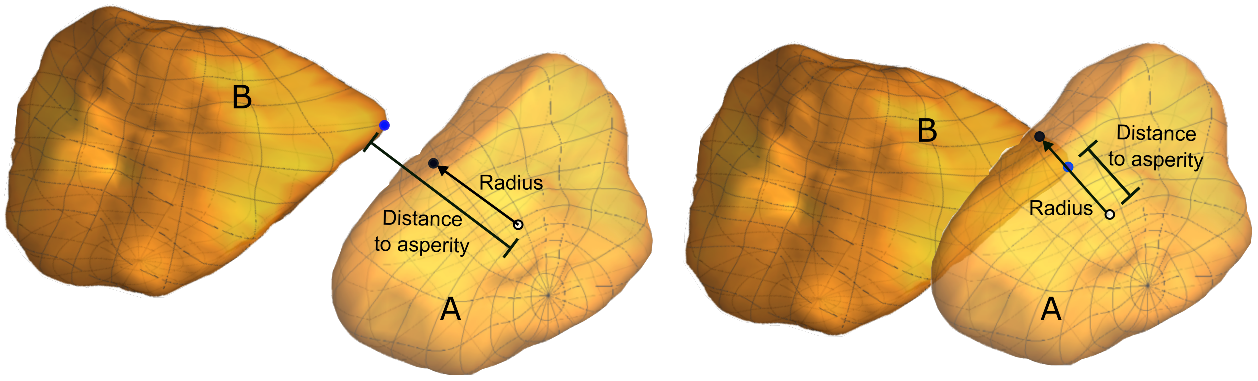


Figure 4.4: Assessing overlap by comparing the distance from the center of particle A (identified with a white marker) to the asperity on particle B (identified with a blue marker) with the radius of particle A along the same vector (identified with a black marker). On the left, the radius is shorter than the distance to the asperity, indicating there is not overlap with the asperity point, while on the right the radius is longer, leading to the detection of overlap.

moves on to assessing the remaining asperities in the lens of potential intersection. If the radius is longer, overlap has been detected and the function exits. This radii-distance comparison was initially seen in [43], however it was used to sweep the entire region of potential intersection, whereas this approach targets specific, predetermined locations within the lens of potential intersection. If all the asperities from the first particle are checked and no overlap is found, the function begins checking the asperities on the second particle. If overlap is detected the function immediately stops and exits.

A similar approach is found in [83], where the L_2 norm of Equation (3.2), is used to determine the minimum distance between two particles. This differs from the approach in this paper in that in [83] the radius equations for each pair of analyzed particles are assessed to find the minimum distance each time they are checked for overlap. In this work, the approach uses the radius equation's spatial derivatives to identify local maximas in the radii and stores the information for recall during future overlap checks.

4.2.3 Brute Force Check

If all asperities pass the previous assessment, the code applies the third filter - a brute force comparison of the surfaces. The brute force method assesses the point clouds of the two aggregates to determine if a point on one aggregate lies within the surface of the other aggregate. This approach is the most robust and will catch all instances of overlap, however it is computationally intensive. The brute force check operates with two filters. The first checks if any points on the surface of aggregate A lie within the inscribed sphere of aggregate B. If any points on aggregate A lie within this sphere, there is overlap between the two particles and the function exits. Otherwise, the function advances to the second filter, which filter relies on MATLAB's built-in function "alphaShape" (https://www.mathworks.com/help/matlab/ref/alphashape.html), which creates a bounding region around a set of points. As the number of points used to define the surface of each 3D aggregate is high, 28,800 points, specifically, the bounding region is very closely aligned with the morphology of the aggregate. The subfunction, "inShape" assesses whether a specified point or set of points lie within the alphaShape. The most straightfoward approach to using the alphaShape objects is to generate an alphaShape of the aggregate in question, herein referred to as aggregate A, and assess if the surface points of nearby aggregates lie within the alphaShape of aggregate A. While this catches *most* of the overlap between aggregate A and neighboring aggregates, one important exception exists - when aggregate A lies entirely inside another aggregate, referred to as aggregate B, such as the example shown in Figure 4.5. This scenario is caught and prevented during the asperity check, because for any asperity on aggregate A, the radius of B will be larger than the distance to the asperity - resulting in detection of overlap. However the control copy of the code needs an additional layer of consideration, because if aggregate A lies entirely within aggregate B, the points on the surface of aggregate B will not be detected as 'in' the alphaShape of aggregate A. In order to detect this, the code checks if all points on aggregate A lie within the bounding sphere of aggregate B. If all of the points on aggregate A fall in the bounding sphere, the code generates an alphaShape of aggregate B and assesses if the points on A are located within it. In the case where part of aggregate A lies outside of the bounding sphere, there is no possibility of aggregate

A being fully enclosed within aggregate B, thus the code uses the previously generated alphaShape of aggregate A to assess if any points on B lie within the alphaShape of A. Each neighboring aggregate is assessed in this manner in order to perform only one alphaShape assessment per pair of aggregates.

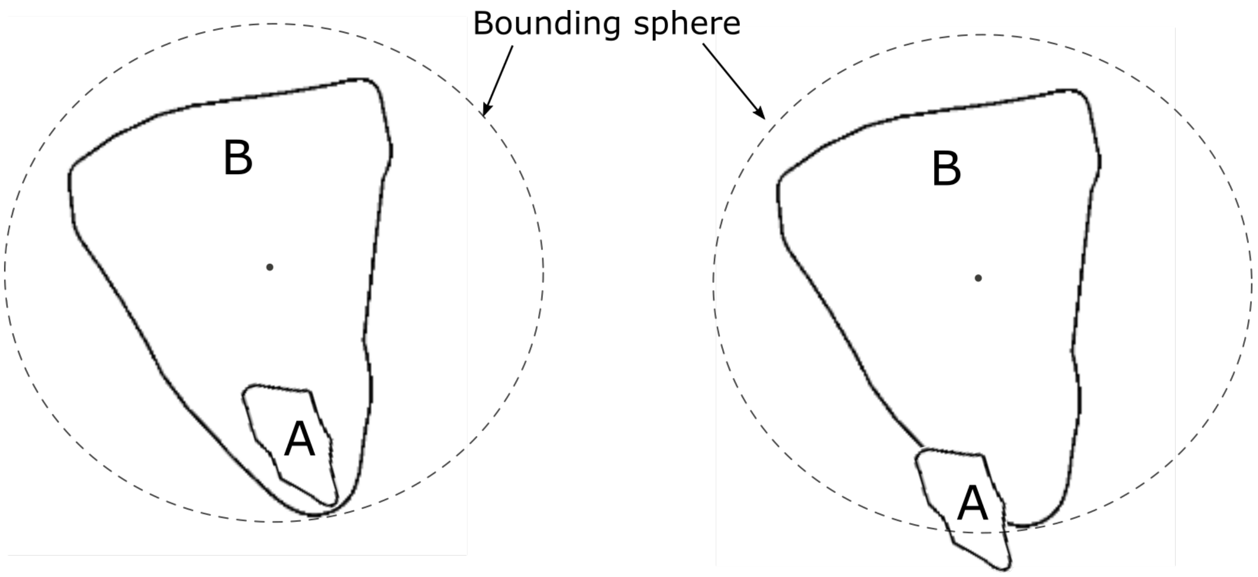


Figure 4.5: Two arrangements of of aggregate A with respect to aggregate B - fully contained in the bounding sphere of aggregate B (left) and intersecting the bounding sphere of aggregate B (right)

4.3 Aggregate Shape Analysis

To provide context for the aggregates used in this work, several shape parameters are assessed and presented here. The ratio of the length and thickness of the particle, generally referred to as the aspect ratio, is assessed alongside the length-width ratio, and the ratio of the maximum and minimum radii. Additionally, the Wadell sphericity parameter, defined as

$$S_w = \frac{\text{Surface area of particle}}{\text{Surface area of volume-equivalent sphere}}, \tag{4.6}$$

is used to describe the roundness of a particle on a scale of 0 to 1, where a S_w value of 1.0 is equivalent to a perfect sphere [128]. As seen in Figure 4.6, the shape parameters are relatively consistent between the four size gradations, indicating that the four gradations assessed in the next section can be compared simply without needing to account for differing aggregate shape ratios in the resulting microstructures.

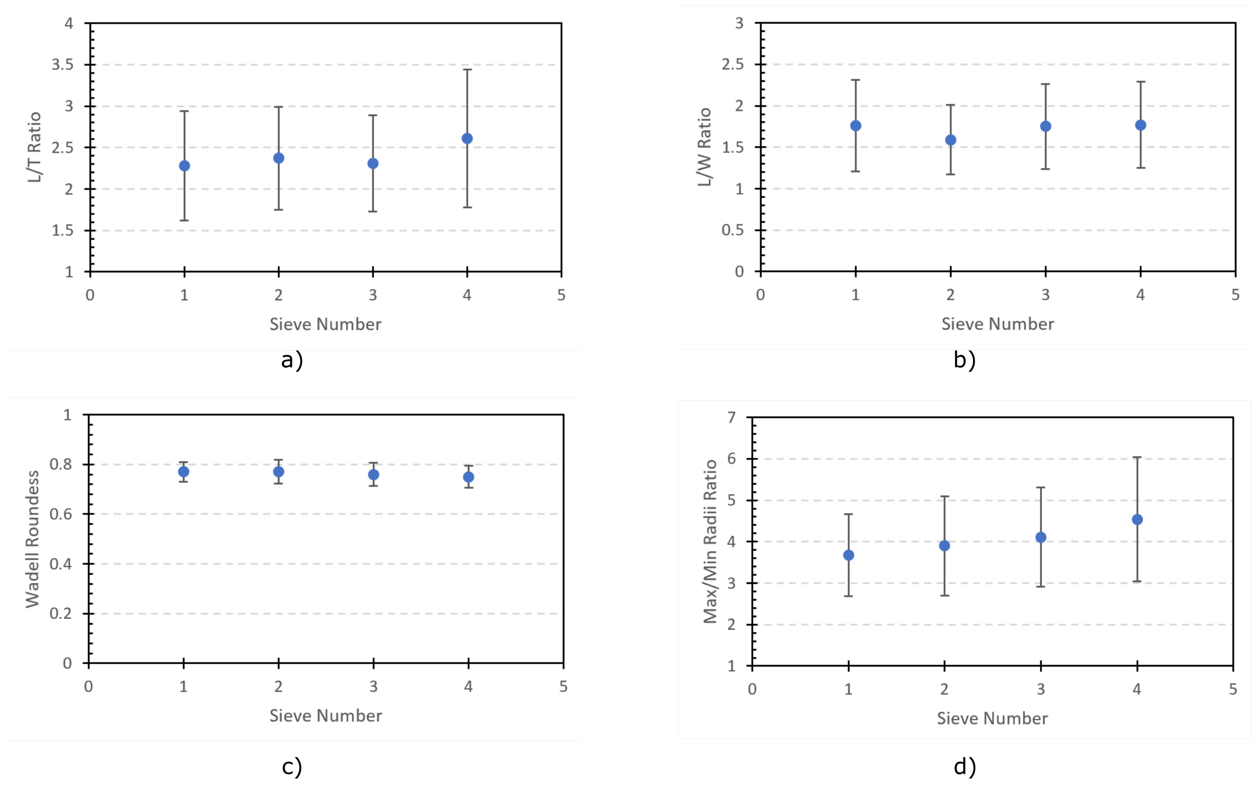


Figure 4.6: Shape parameters for each of the four sieve sizes: a) length-thickness (L/T) ratio, b) length-width (L/W) ratio, C) Wadell sphericity, and d) maximum-minimum radius ratio. Error bars represent ± one standard deviation

4.4 Results

4.4.1 Aggregate Size Gradations

Four particle size distributions were studied to determine the impact of the asperity check on computation time in a variety of packing scenarios. The particles are divided into four size groups based on their width. These gradation curves are seen in Figure 4.7. Gradations A and B contain greater proportions of larger particles, whereas gradations C and D contain a finer mix of particles. Gradation curve C is based on the Fuller curve, where the percentage of aggregate, Y, passing through a sieve with diameter, D, is given to be

$$Y = 100 * \frac{D}{D_{max}^{1/2}}, \tag{4.7}$$

where D_{max} is the diameter of the largest aggregate [129].

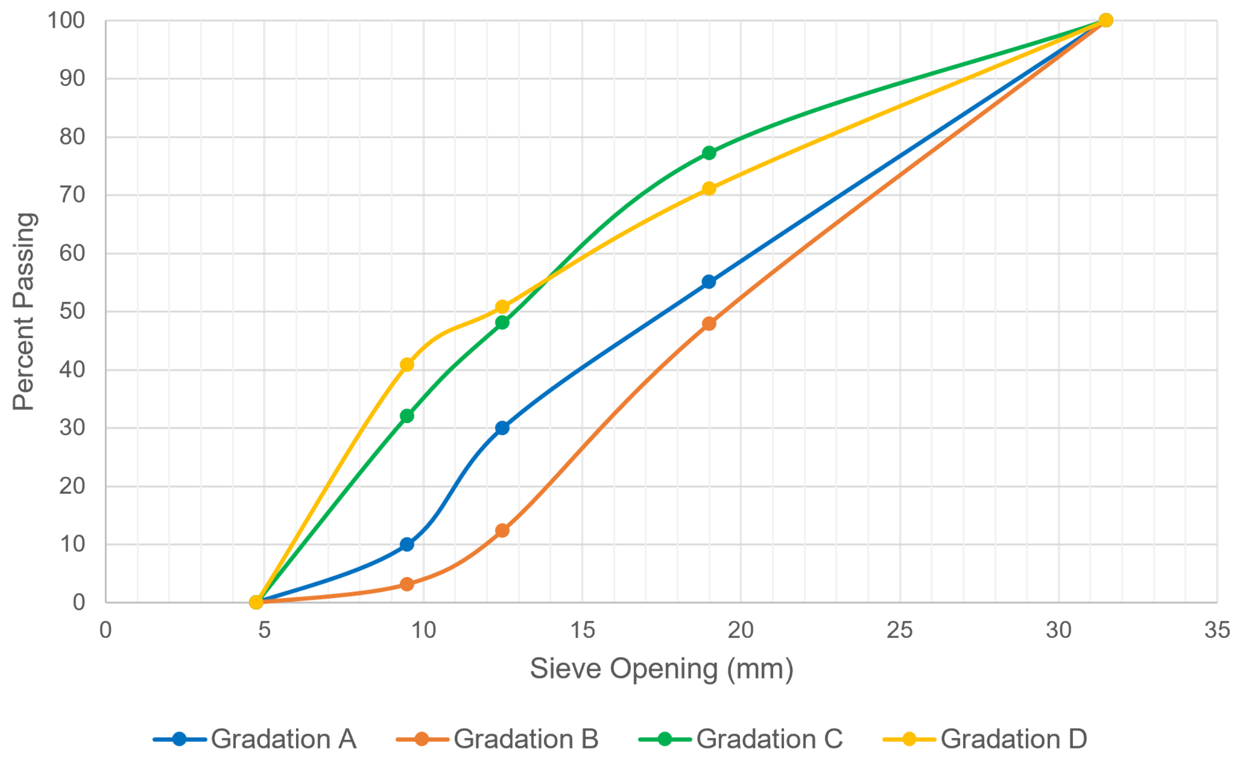


Figure 4.7: The four gradation curves studied - with two coarse gradations (A and B), alongside two finer gradations (C and D)

All microstructure plots were carried out using Matlab R2018a on the Texas A&M University High Performance Computing Center's supercomputer Ada, a hybrid cluster from IBM with Intel Ivy Bridge processors and a Mellanox FDR-10 Infiniband interconnect. The plotting code is carried out serially, placing one particle at a time, on a single node with 2500 MB of allotted memory. For each gradation, three microstructures are plotted at each 20 %, 30 %, and 40 % particle volume fractions. The control code is a copy of the plotting code with the asperity check removed and the additional consideration to determine if an aggregate lies entirely within another. All other parameters and functions in the codes remain the same.

4.4.2 Influence of n_{max} Value

As the asperities of interest are the major maxima in the radii, not the surface detail, a reduced number of summations can be used in the derivative equations. In Figure 4.8, an example of asperity detection using a variety of n_{max} values is depicted.

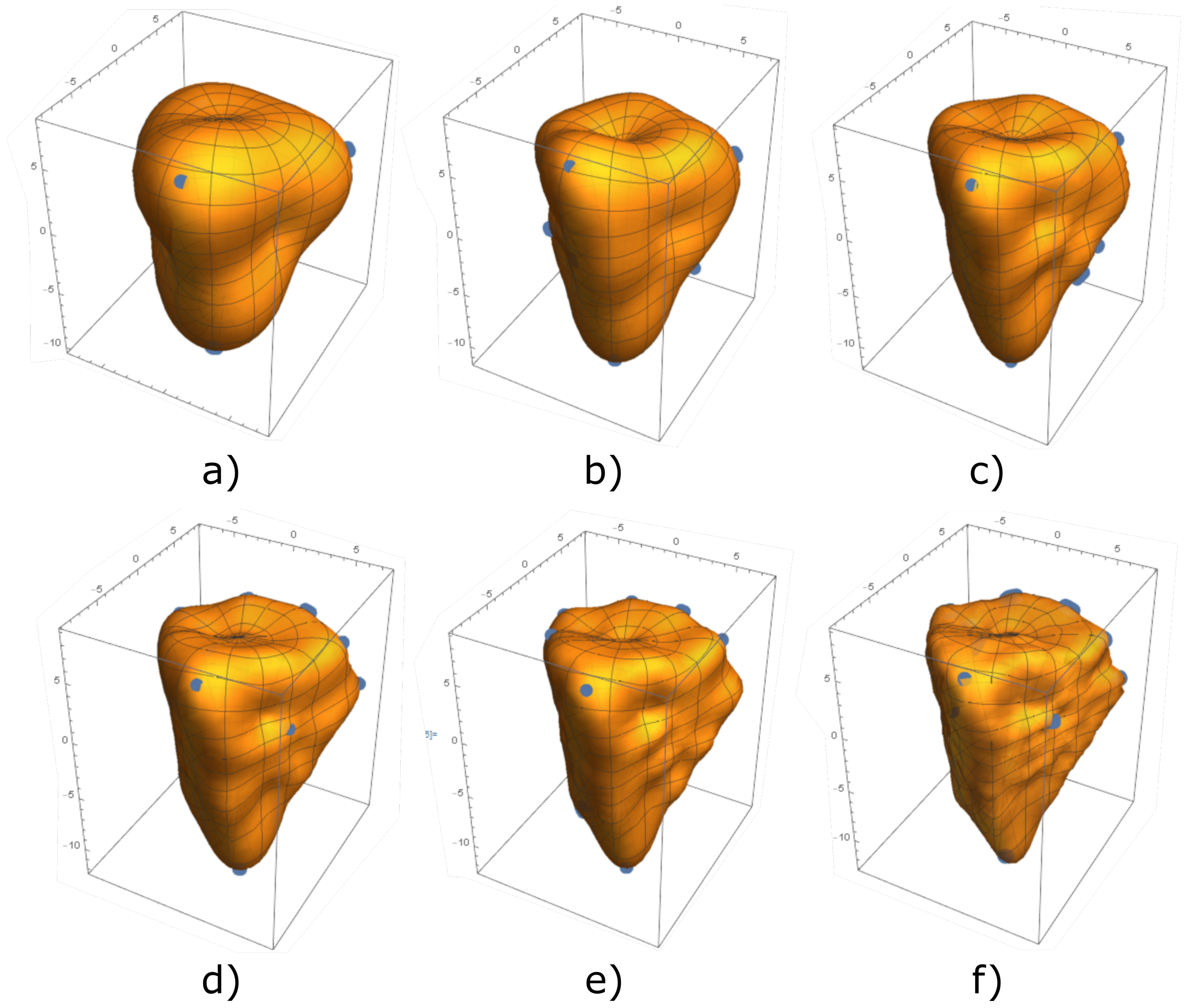


Figure 4.8: Asperities identified on a particle plotted and analyzed with an n_{max} value of a) 4, b) 6, c) 8, d) 10, e) 12, f) 20. The largest asperities are identified on all, with increasingly subtle asperities detected as n_{max} increases

Table 4.1: Mean and median number of asperities identified on each aggregate for varying n_{max} values.

n_{max} Value	Mean	Median
6	18.2	16
8	23.2	21
10	28.3	26
12	33.1	31

To quantify the level of influence that the n_{max} parameter used during asperity detection has, the set of pre-rotated aggregates was run through the asperity detection code four times - using n_{max} values of 6, 8, 10, and 12. The n_{max} value of 4 was not considered due to the notable morphological difference between the aggregate generated with an n_{max} value of 4 and the aggregates generated with higher n_{max} values in Figure 4.8.

As the mean and median number of asperities detected by the varying n_{max} values in Table 4.1 varies by up to 15 asperities per aggregate - a test determine the optimal n_{max} value to use during the asperity detection process was conducted. Three microstructures of 40% volume fraction coarse aggregate were then generated using each set of asperities. The average runtime of the microstructure generation for each n_{max} value using gradation A is shown in Figure 4.9.

It was found that the n_{max} values used during asperity detection did not cause a statistically significant difference in the overlap detection time, as each of the error bars overlapped. This is likely due to the major asperities being visible and detected at an n_{max} value as low as 4. As seen in Figure 4.8, the morphology of the aggregate generated with an n_{max} value of 4 is notably morphologically smoother and simpler than the aggregates with a higher n_{max} value. In this work, the asperities are detected with an n_{max} value of 10, as this value provides a balance of computational speed and morphological detail during the asperity detection process. Aggregates are generated for particle packing with an n_{max} value of 18 to 26 based on the recommendation attached the data set.

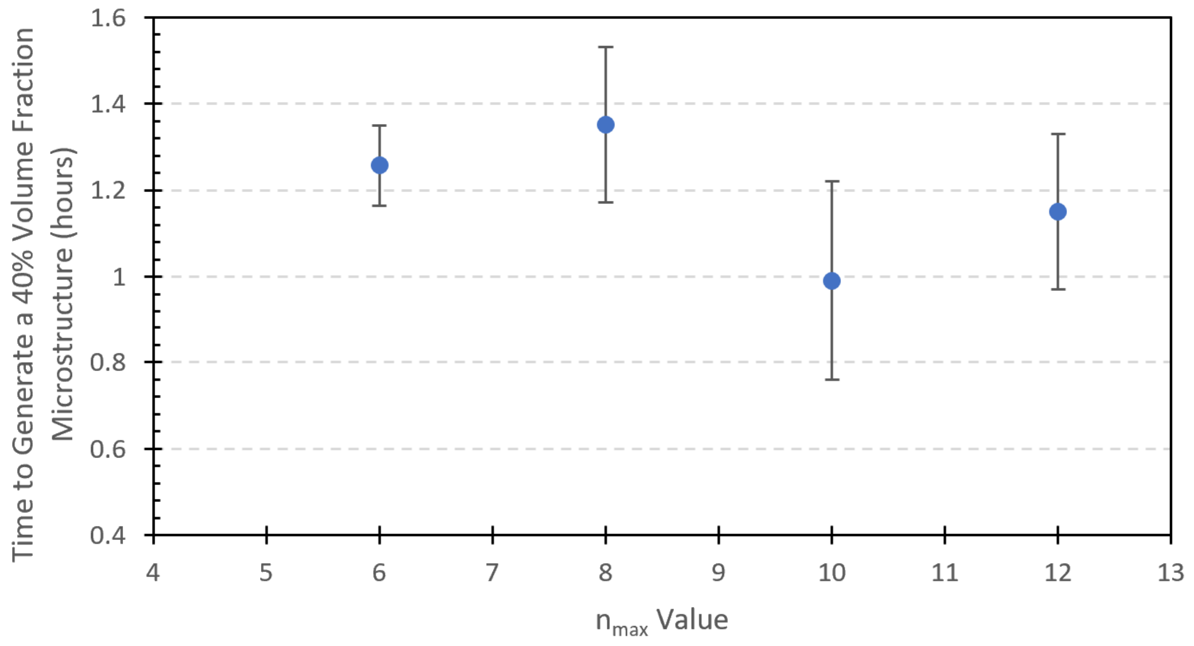


Figure 4.9: The time to generate a 40% volume fraction microstructure, using gradation A, with asperities identified using an n_{max} value of 6, 8, 10, and 12. Each data point represents the average runtime of generating three microstructures. Error bars represent $\pm$ one standard deviation

4.4.3 Efficiency Improvements

Tables 4.2 thru 4.5 shows the runtime data differences between microstructure plotting with and without the asperity check, giving the average runtime $\pm$ the standard deviation of the three runs. Speedup is a parameter describing the improvement in code performance, according to

$$\text{Speedup} = \frac{\text{Average Runtime of Control Code}}{\text{Average Runtime of Code with Asperity Check}}. \tag{4.8}$$

Table 4.2: Runtime and improvement results for each volume fraction using Gradation A

Volume Fraction	Avg. Runtime ± Std. Dev. (hrs)	Avg. Runtime Control ± Std. Dev. (hrs)	Avg. Speedup
20%	0.05 ± 0.0001	0.19 ± 0.009	3.7
30%	0.12 ± 0.0009	0.94 ± 0.16	7.6
40%	1.45 ± 0.55	12.24 ± 3.1	8.5

Table 4.3: Runtime and improvement results for each volume fraction using Gradation B

Volume Fraction	Avg. Runtime ± Std. Dev. (hrs)	Avg. Runtime Control ± Std. Dev. (hrs)	Avg. Speedup
20%	0.03 ± 0.001	0.11 ± 0.02	3.4
30%	0.16 ±0.02	1.08 ± 0.14	6.7
40%	2.92 ± 1.1	36.9± 8.5	12.6

Table 4.4: Runtime and improvement results for each volume fraction using Gradation C

Volume Fraction	Avg. Runtime ± Std. Dev. (hrs)	Avg. Runtime Control ± Std. Dev. (hrs)	Avg. Speedup
20%	0.11 ± 0.003	0.42 ±0.015	3.8
30%	0.54 ± 0.11	4.3 ± 0.89	7.9
40%	10.37 ± 1.8	91.3 ± 25.3	8.8

Table 4.5: Runtime and improvement results for each volume fraction using Gradation D

Volume Fraction	Avg. Runtime ± Std. Dev. (hrs)	Avg. Runtime Control ± Std. Dev. (hrs)	Avg. Speedup
20%	0.15 ± 0.004	0.52 ± 0.02	3.44
30%	0.68 ± 0.014	4.6 ± 0.77	6.7
40%	9.52 ± 0.2	98.9 ± 7.5	10.3

The speedup values for each volume fraction and gradation are shown together in Figure 4.10. The average speedup demonstrated across the four gradations is 3.6, 7.2, and 10.0 for the 20%, 30%, and 40% volume fractions, respectively. While the plotting code executes serially, the plotting time is, on average, cut by an over 70% in the lowest volume fraction and over 85% in the higher volume fractions by utilizing an asperity-based overlap detection algorithm. It can be seen that the average time to plot increases with the fineness of the particle size gradation, which can be directly linked to the number of particles placed in order to reach the desired volume fraction. Furthermore, the relative speedup values for the higher volume fractions are greater than the lower volume fractions, the result of more particles plotted, and thus more overlap checks, to reach the desired volume fraction.

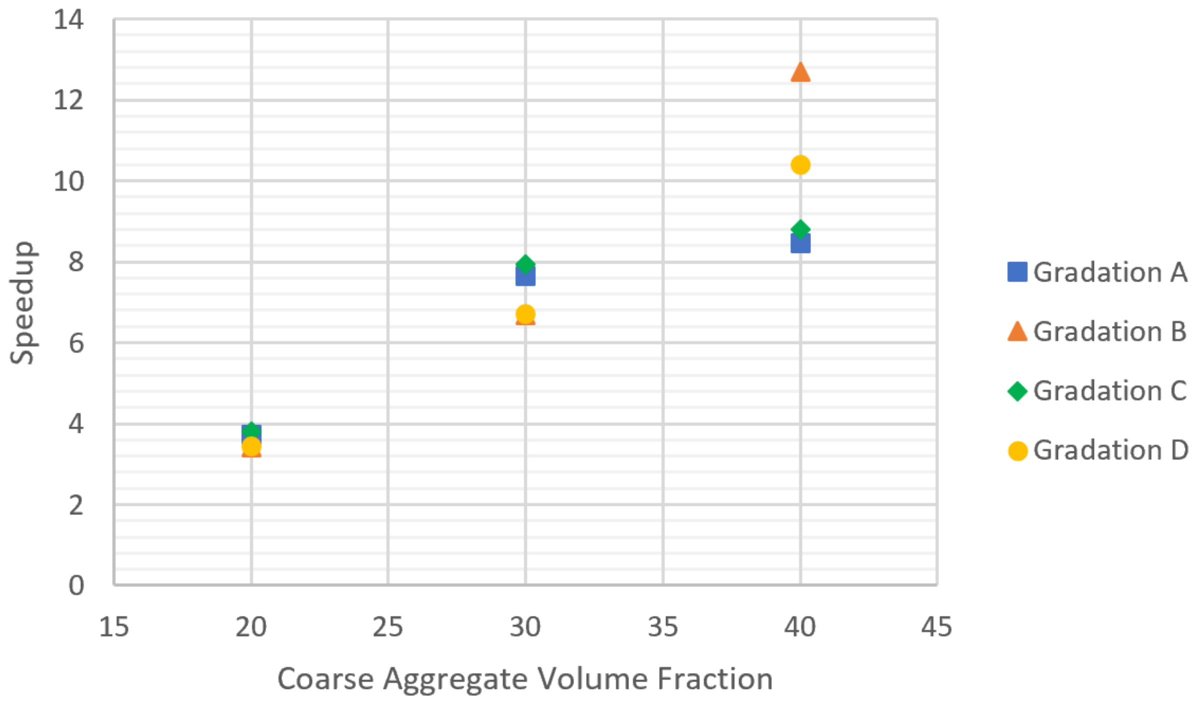


Figure 4.10: Combined speedup data of each volume fraction and sieve gradation

When reaching high percentages of runtime reduction, a very small percentage difference can result in a large difference in speedup values. For example, consider the average speedup values of gradations A and B for the 40% microstructure generations. Gradation A has an average speedup value of 8.5, while gradation B averages at 12.6. Table 4.6 shows the percentage reduction in the runtime, where it can be seen that gradation A's average reduction for a 40% microstructure is 88.2%, while gradation B's average is 92.1%. This data is presented in Figure 4.11 to visualize the similarity in the percentage reductions for each gradation and volume fraction. This data is the same as that of Figure 4.8, however it is presented in terms of percentage reduction in runtime to provide perspective on the gaps in the speedup values between the gradations.

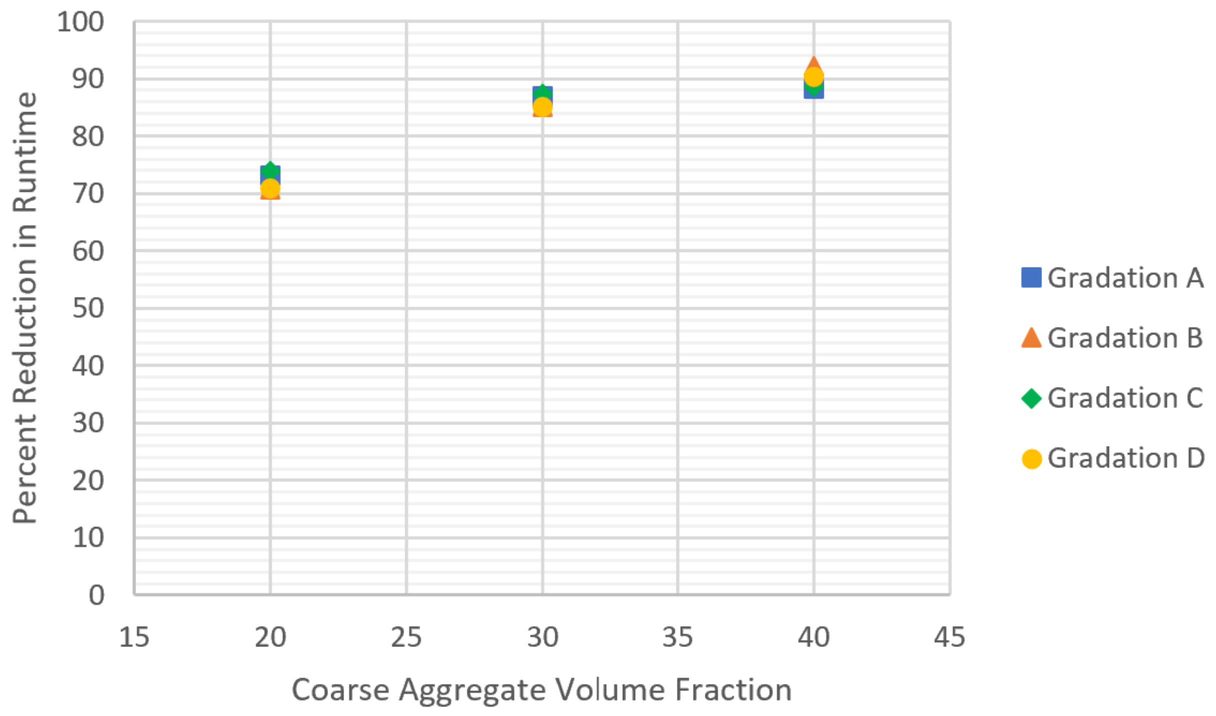


Figure 4.11: Average percentage reduction in runtime for each combination of volume fraction, 20%, 30% and 40%, and sieve gradation, A-D

4.4.4 Proportion of Overlap Detection Methods

While the improvements in efficiency due to the addition of the asperity check are clear, to determine the proportion of overlap detected by each of the three filters, counters were added to the

Table 4.6: Average percentage reducting in time to generate a microstructure for each volume fraction and sieve gradation combination

Gradation	20%	30%	40%
A	73.1%	86.9%	88.2 %
B	70.1%	85.0 %	92.1 %
C	73.4%	87.4 %	88.6 %
D	70.9%	85.1%	90.4 %

Table 4.7: Average proportion of overlap detected by each filter for a 40% volume fraction microstructure generated with gradations A-D

Gradation	Minimum Radii	Asperity Check	Brute Force
A	13.1%	86.4 %	0.5 %
B	13.3%	86.5 %	0.2 %
C	10.7%	89.2 %	0.1 %
D	12.8%	86.6 %	0.6 %

code to track each instance of overlap detection. The data collected by these counters can be seen in Figure 4.12 for a 40% microstructures generated with gradations A-D.

Table 4.7 gives the average proportion of overlap detected by each of the three overlap detection methods implemented in this code. The data is taken as a simple average from the raw data shown in Figure 4.12. The minimum proportion of overlap detected through the asperity-radii comparison is 86.4 %, confirming that this approach is highly effective. The proportion of overlap that makes its way to the brute force check is very low, at a maximum of 0.6%. This explains why the reductions in runtime given in Figure 4.11 are so substantial.

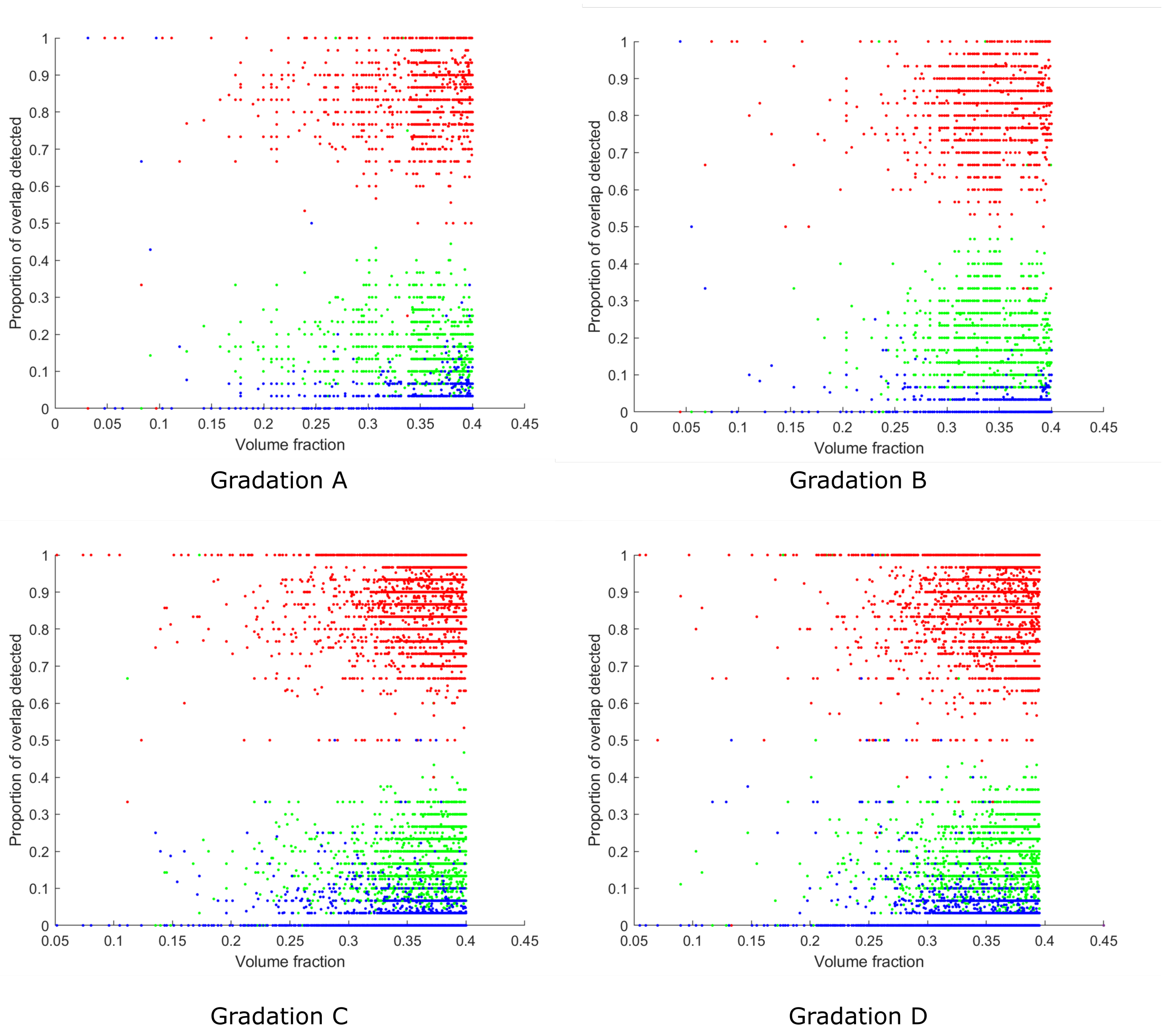


Figure 4.12: Proportion of overlap detected by each of the three filters as a function of volume fraction in a 40% volume fraction microstructure for gradations A-D. Overlap detected by the minimum diameter is labelled in green, the asperity check is labelled red, and the brute force is marked in blue. Each data point represents the proportions of overlap detected during each aggregate's total number of placement attempts.

4.5 Summary

As demonstrated, the asperity check reduces the time to plot a microstructure, on average, over 85% for realistic volume fractions. The particles used in this study were reconstructions of gravel, however this approach is useful in any type of particle described by any analytical function with a real second derivative or has otherwise identifiable surface asperities. The code used in this work places particles serially; additional improvements in efficiency would be seen if the code was executed in parallel. Additional future work could look at the impact of various shape parameters on the efficiency improvements to determine what kinds of aggregates are best suited to this method. The value of n_{max} used during asperity detection was shown to not play an impactful role on the level of efficiency improvement but more work needed to more fully understand if there is a link between aggregate morphology and level of efficiency improvement. The following overarching conclusions can be drawn from this chapter:

- Particles described by analytical functions can be assessed for the detection of surface asperities
- To determine overlap between a surface asperity and a neighboring aggregate, the angles between the two aggregates are calculated and used to determine the radius of the neighboring particle in the direction of the surface asperity. If the radius is longer than the distance to the asperity, overlap is detected.
- In the spherical harmonics reconstruction of the aggregates used in this work, the n_{max} value used during asperity detection did not play a large role in the computational runtime during the generation of 40% volume fraction microstructure.
- The improvements in efficiency were above 85% for microstructures containing greater than 30% volume fraction coarse aggregates.
- The speedup values increase with increasing volume fraction. The average speedup values are 3.6, 7.2, and 10.0 for the 20%, 30% and 40% volume fractions, respectively.
- The asperity check detected between an average of 85% and 90 % of all aggregate-aggregate overlap in 40% microstructures of each gradation. Less than 1% of overlap made it to the brute force check
- Future work should assess additional aggregate sets with differing shape parameters to quan-

tify the efficiency improvement's linkage to aggregate morphology.

5. LONG-TERM CREEP AND RELAXATION MODELING OF CONCRETE

The periodic microstructures generated from the methods described in Chapters 3 and 4 are used in finite element analysis (FEA) to simulate the long-term viscoelastic behavior of the virtual concrete. Cast and cored microstructures are assessed in the following chapter. For validation purposes, comparison to experimental data is performed out to 10,000 days. Variables including the dimensionality of the microstructure and the Poisson's ratio are investigated for their impact on the prediction of concrete creep.

5.1 Finite Element Mesh Generation

To convert the virtual microstructures into a finite element mesh, this work utilizes software packages called OOF2 and OOF3D, developed by the National Institute of Standards and Technology (NIST). In preparation, the 2D microstructures are saved as an image and the 3D microstructures are 'sliced' into a stack of images. Image slicing is done by sectioning the microstructure in 0.3mm increments up the Z-axis and saving each slice increment as an image. The image stack is then loaded into OOF3D, where the images are re-stacked up the Z-axis, forming voxels by assigning each pixel a height of 0.3mm. A burn algorithm is used to identify the matrix and classify it as a different material from the un-selected aggregate regions.

With the pixels and voxels categorizes as matrix and aggregate regions, OOF3D moves to creating the finite element mesh. The 3D microstructures are meshed by starting with a uniform tetrahedral mesh composed of elements with an edge length of 3mm and a volume of 3.1 mm^3. 2D microstructures begin with a uniform triangular mesh with element edge lengths of 3mm. In both 2D and 3D, the uniform mesh is then subjected to several rounds of adaptive refinement to subdivide heterogenous elements along matrix-aggregate interfaces in order to retain the aggregate's original morphology. In the case of 2D microstructures, the refinement functions introduce quadrilateral elements, making the resulting mesh a mixture of triangular and quadrilateral elements, while the

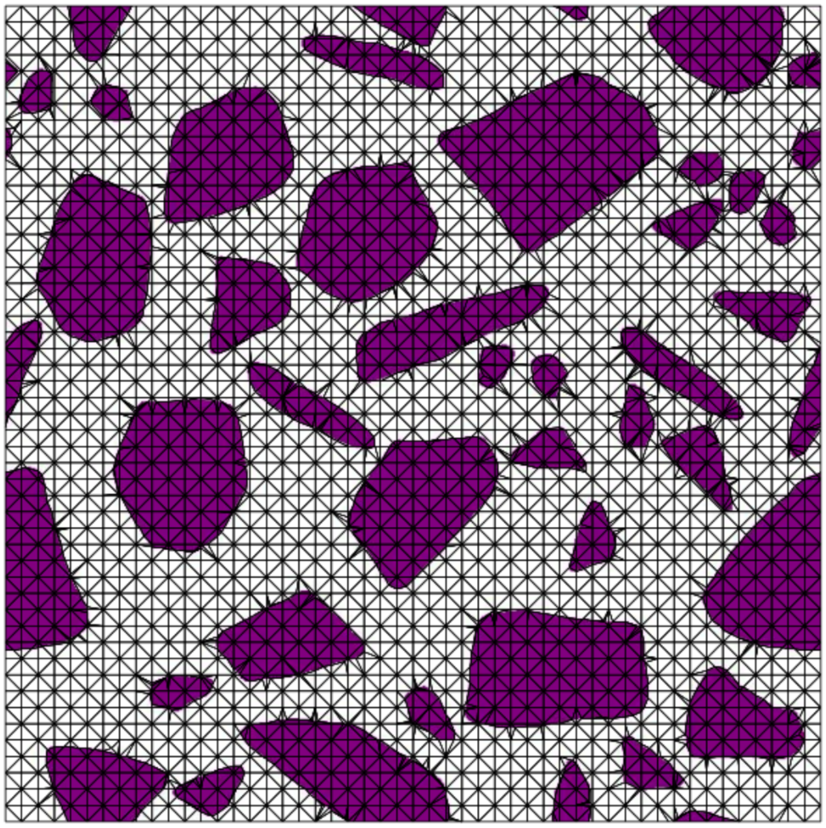


Figure 5.1: A 2D microstructure meshed with OOF2.

3D meshes are composed solely of tetrahedral elements. A meshed 2D microstructure can be seen in Figure 5.1. An area of future work is to mesh the microstructures in a non-imaged based manner.

5.2 Implementation of Periodic Boundary Conditions

Periodic boundary conditions in the morphology of microstructures has been previously discussed, however periodicity applies to the stress and strain fields as well, meaning that opposing boundaries should be identical to one another. To enforce periodicity in these fields, the surface nodes are each paired with their respective opposite nodes on the opposing boundary. Here, the term 'opposing boundary' refers to the corners, edges, and walls on opposite sides of an RVE that touch when the RVE is used to reconstruct the large, infinite pattern it represents. For example, nodes on the $\pm$ X-Z plane walls are paired together. In order to pair nodes on opposing boundaries, each node on the a boundary is assessed to find it's most closely aligned node on the opposite boundary. These node pairing are used to write the constraint equations into the Abaqus input file. Each pair of nodes, given as node a and node b, are constrained such that their displacement, u, in each direction is bound by the relationships:

$$\begin{aligned} u_x^a - u_x^b &= 0 \\ u_y^a - u_y^b &= 0 \\ u_z^a - u_z^b &= 0, \end{aligned} \tag{5.1}$$

in the x, y, and z directions respectively.

To ensure the periodic boundary condition constraints are correctly implemented, the stress and strain along the elements compositing the top and bottom edges of a two-dimensional microstructure at time = 0 are plotted together. True periodicity is seen when the stress and strain values are identical along opposing boundaries. As seen in Figure 5.2, the X-direction of the strain field in the elements along the two edges is nearly equal, indicating the periodicity in the strain field has been properly enforced. stress values are also assessed for periodicity, as depicted in Figure 5.3. Slight differences in the curves can be attributed to differences in the finite element mesh and edge element construction at the top and bottom edges of the microstructure. Furthermore, the stress and strain values are taken as the stress of the element along the edge, meaning that the differences in element sizes along the two edges imposes differences in the strain data along the top and bottom edges in Figures 5.2 and 5.3.

To ensure the stress and strain fields are also periodic in the 3D microstructure simulations, as well as over time, two opposing edges in the X-direction on the X-Z plane are assessed for their stress and strain values after 10,000 simulated days. Two edges, labelled the top and bottom, were assessed by evaluating the stress and strain in the edge elements from x = 0 mm to x = 90 mm with a constant Z-coordinate of 90 mm and 0 mm for the top and bottom edges respectively. Much like the 2D results, the strain and stress are found to be periodic and the values at opposing edges nearly equal to each other, with the results shown in Figure 5.4 and Figure 5.5 for the strain and stress

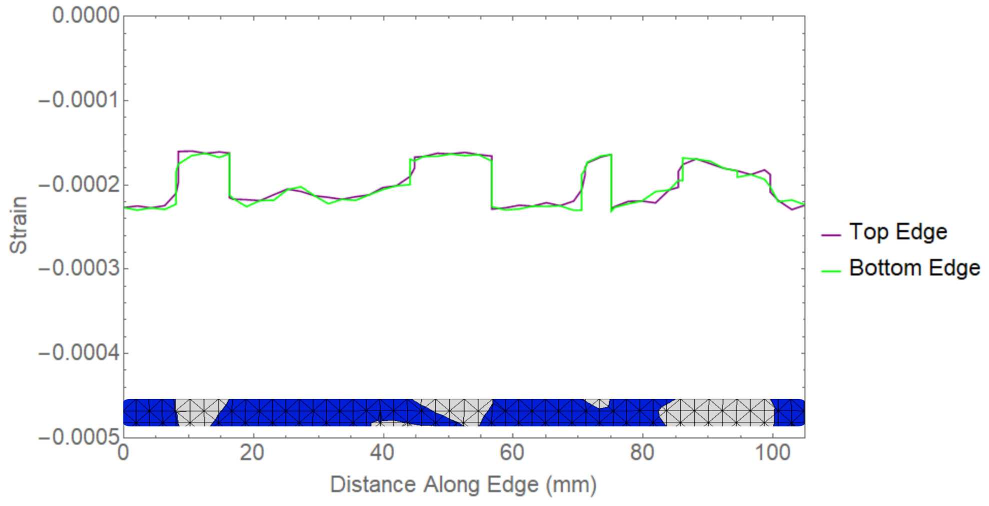


Figure 5.2: Strain along the edges of a 105mm square two-dimensional microstructure after 0 days, in X-direction with the edge of the microstructure overlaid below the curves to indicate composition along the lines. Mortar is identified in blue and aggregates in gray. Deviations between the two lines is the result of differences in the finite element mesh between the two edges.

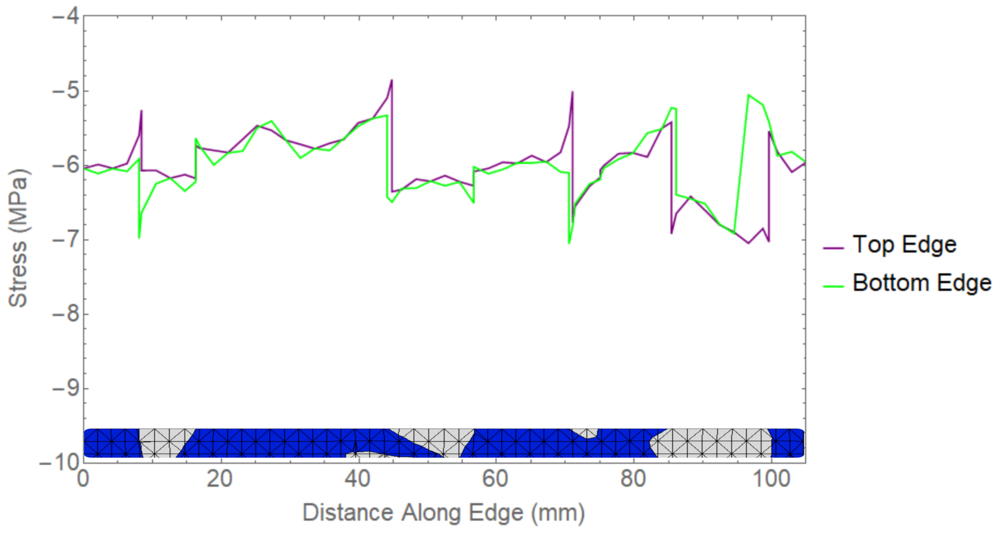


Figure 5.3: Stress along the edges of a 105mm square two-dimensional microstructure after 0 days, in X-direction with the edge of the microstructure overlaid below the curves to indicate composition along the lines. Mortar is identified in blue and aggregates in gray. Deviations between the two lines is the result of differences in the finite element mesh between the two edges.

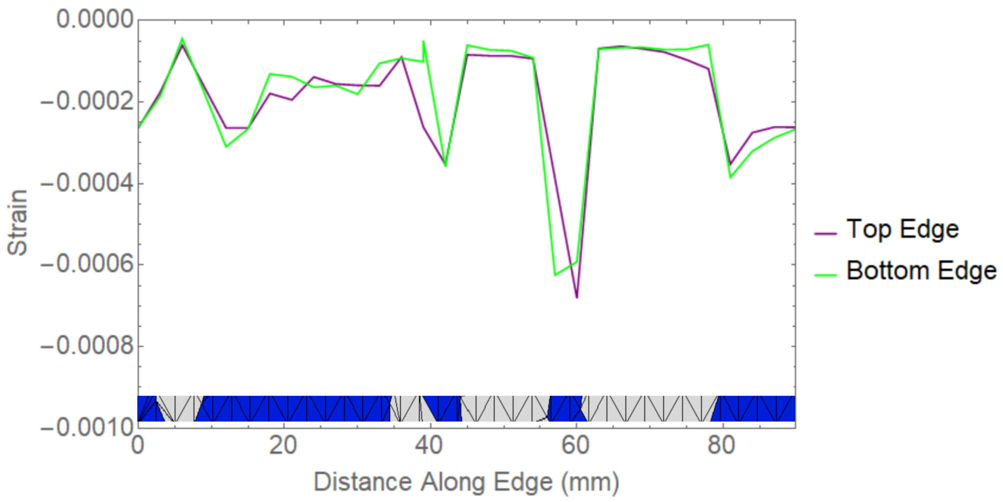


Figure 5.4: Strain along the edges of a 90mm cubic three-dimensional microstructure after 7,084 days, in X-direction with the edge of the microstructure overlaid below the curves to indicate composition along the lines. Mortar is identified in blue and aggregates in gray. Deviations between the two lines is the result of differences in the finite element mesh between the two edges.

respectively. The differences between the stress and strain two opposing edges is more pronounced in the 3D microstructures than in the 2D, as OOF3D does not have the ability to generate periodic finite element meshes in 3D, as it does in 2D. This adds an additional layer of difference to the opposing boundaries in 3D because while in 2D the nodes were periodic but the element size and shape differed, in 3D the nodes are not periodic. The outcome of this is that the finite elements composing opposing boundaries in 3D have additional mesh-based differences that translate to nearly, but not perfectly periodic stress and strain fields.

5.3 Description of Viscoelastic Mortar

Experimental mortar and concrete creep tests were performed in concurrent research [45, 21, 2]. The mixture design, setup, and data is briefly presented here for context.

5.3.1 Experimental Sample Preparation

The VeRCoRs concrete mix design from Électricité de France (EDF) serves as the basis for the concrete mix used in this work. The concrete samples are a mixture of Type I/II cement, fine aggregates that pass a 4 mm sieve, intermediate aggregates that are sieved to pass between 8 mm and

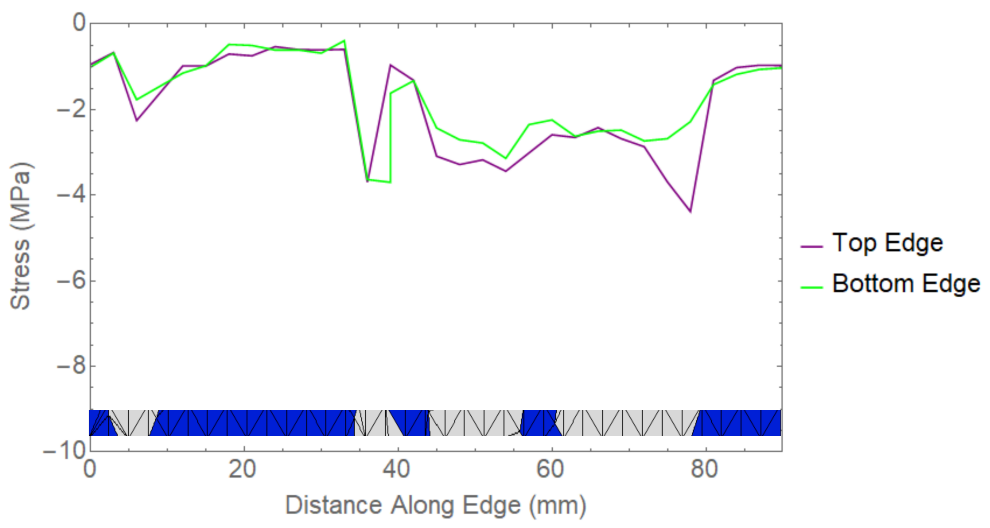


Figure 5.5: Stress along the edges of a 90mm cubic three-dimensional microstructure after 7,084 days, in X-direction with the edge of the microstructure overlaid below the curves to indicate composition along the lines. Mortar is identified in blue and aggregates in gray. Deviations between the two lines is the result of differences in the finite element mesh between the two edges.

4 mm sieves, and coarse aggregates that pass between the 16 mm and 8 mm sieves [130]. Aggregates were dried in an oven for 24 h prior to mixing. Pozzolith 80, a water reducing admixture, was added at a proportion of 4.22 mL per kilogram of cementitious material to improve the workability. The resulting water to cement (w/c) ratio in this concrete mix is 0.52 by mass in the saturated surface dry SSD condition. Table 5.1 gives the proportions of the concrete mix components. For the purposes of virtual microstructure generation, the sum of the intermediate and coarse aggregate volume fractions, 38%, is used as the target aggregate volume fraction of aggregates in the generate microstructures.

The concrete was mixed using the procedure outlined in ASTM C192 and immediately cast into 101.6 mm by 203.2 mm (4 in by 8 in) cylindrical molds [131]. Embedded strain gauges were suspended from fishing line that were positioned axially in the center of the molds. Filling of the molds was done in three increments with tapping in between to minimize air voids in the samples. The concrete remained in the molds for 28 days to prevent moisture loss prior to testing. Once the samples reached an age of 28 days, they were demolded and sealed with one layer of adhesive-

Table 5.1: Concrete mix design used for experimental concrete creep samples, based on the EDF Vercors concrete mix.

Materials	Quantity
Cement (Type I/II)	320 kg/m^3
Fine Aggregate	837 kg/m^3
Intermediate Aggregate	456 kg/m^3
Coarse Aggregate	560 kg/m^3
Water	137 kg/m^3
Pozzolith 80 (Water-reducing admixture)	1.35 l/m^3

backed aluminum foil to minimize drying effects.

The samples were divided into two groups, the first to be used for the uniaxial creep tests, and the second used measure free strain shrinkage. The two groups were then each divided into two temperature groups: 20^o C and 60^o C. The 60^o C creep test samples were heated to this temperature before beginning the test to prevent thermal strains from building up during creep. The creep test samples were loaded with a uniaxial compressive load of 5300 N (1200 lbs), corresponding to 20% of the 28 days compressive strength. This value was chosen as it allowed the concrete to be considered as a non-aging linearly viscoelastic material; higher loads risk entering the non-linear viscoelastic regime [14]. The free strain samples were left untouched at their respective temperatures .

Due to the short time to needed apply the compressive load to the creep samples, the application of the load was approximated as a Heaviside step function. Data from the embedded strain gauges was read and recorded by a datalogger. For both the creep and free strain samples, the data was read and recorded immediately upon loading, followed by daily readings for one week, then weekly readings for one month, and finally monthly readings thereafter. Additionally, detachable mechanical (DEMEC) strain gauges were attached to the external surface of the samples to collect strain data on the outer surfaces of the cylinders.

Despite sealing the samples with adhesive-backed aluminum foil, a small amount of drying creep was observed. This creep was separated from the total creep to divide the creep data into drying creep and basic creep. More information on how this was done can be found in Baranikumar et. al (2020) [45]. Only the basic creep is assessed in this work.

5.3.2 Experimental Data Analysis

Creep tests ideally consist of a sample under a constant load over time, however in reality it is difficult to maintain an unchanging load. To account for changes in the applied compressive stress, the load history is accounted for through $\sigma(t)$, which is the stress as a function of time, t. Measured creep data over time is represented by $\epsilon(t)$. The constitutive relationship between the stress history, $\sigma(t)$, the strain, $\epsilon(t)$, and creep compliance, $J(t)$ is

$$\epsilon(t) = \int_0^t J(t-t')\,\frac{\delta\sigma(t')}{\delta t'}dt, \tag{5.2}$$

where t is time in days and t' is the dummy time variable. $J(t)$ is assumed to be of the form

$$\begin{aligned} J(t) = J_0 + J_1 \cdot Ln[1+\frac{-t}{\tau_1}] + J_2 \cdot Ln[1+\frac{-t}{\tau_2}] + J_3 \cdot Ln[1+\frac{-t}{\tau_3}] \\ +J_4 \cdot Ln[1+\frac{-t}{\tau_4}] + J_5 \cdot Ln[1+\frac{-t}{\tau_5}] + J_6 \cdot Ln[1+\frac{-t}{\tau_6}]\frac{1}{GPa}, \end{aligned} \tag{5.3}$$

where J_0, J_1, J_2, J_3, J_4, J_5, and J_6 are coefficients with units GPa^{-1}. The relaxation times τ_0, τ_1, τ_2, τ_3, τ_4, τ_5 and τ_6 are in units of days and are chosen to be 1, 10, 100, 1,000, 10,000, and 100,000 respectively. To determine the coefficients in the $J(t)$ equation, Equation (5.2) is integrated and fit to the $\epsilon(t)$ data. The values of the fitted coefficients are then plugged into Equation (5.3) to obtain $J(t)$. Mortar creep data from experimental work is used to calculate the mortar's bulk and shear compliance functions. This data originates in Baranikumar et. al (2020), where further details regarding the sample preparation, experimental procedure, and application of TTS are described

[45]. The mortar creep compliance data is given by the equation $J(t)$,

$$\begin{aligned} J(t) = 0.039 + 1.86 \cdot 10^{-3} \cdot Ln[1+t] - 5.19 \cdot Ln[1+\frac{t}{10}] \\ +3.85 \cdot 10^{-2} \cdot Ln[1+\frac{t}{100}] + 8.56 \cdot 10^{-3} \cdot Ln[1+\frac{t}{1000}] \\ +2.58 \cdot 10^{-9} \cdot Ln[1+\frac{t}{100000}]\ \frac{1}{GPa}. \end{aligned} \tag{5.4}$$

The creep compliance is directly related to the bulk modulus, $B(t)$, given by the relationship,

$$B(t) = \frac{J(t)}{3 \cdot (1 - 2 \cdot \nu)}, \tag{5.5}$$

where ν is the poisson's ratio. The Poisson's ratio was calculated confined mortar creep tests and found to be nearly constant over time [21], which allows for the simple interconversion between the bulk, shear, and elastic moduli. As a result, this work is assuming a constant Poisson's ratio. The shear compliance, $G(t)$, is found through a similar relationship,

$$G(t) = \frac{J(t)}{2 \cdot (1 + \nu)}. \tag{5.6}$$

The bulk and shear compliance functions are then normalized by their values at t = 0 days, and used to generate a data file containing the normalized data. This file is read by ABAQUS, which then fits the data to a Prony series, representing the dimensionless relaxation modulus.

5.4 Simulation Data Analysis

The software ABAQUS was selected to be the FEA solver for this research. To analyze the meshed microstructures, a simulated stress relaxation simulation experiment was chosen. The coarse aggregates are assumed to be linear elastic, while the mortar phase is described by the data collected from laboratory mortar creep experiments [45]. The microstructure is subjected to a constant compressive strain of 0.02% applied in the X-direction and held for the remainder of the simulation. The $\pm$ Y-Z planes where the displacement is applied are not permitted to translate in the Y- or Z-directions or rotate. A stress relaxation simulation was chosen over a creep simulation due to the ease of applying a constant strain boundary condition. Moreover, the stress relaxation modulus, $E(t)$, and the creep compliance are inversely related in the Laplace domain,

$$\overline{E(s)} = \frac{1}{s^2\overline{J(s)}}, \tag{5.7}$$

where s is the Laplace transformed time variance, $\overline{J(s)}$ is the Laplace transformed $J(t)$, and $\overline{E(s)}$ is the Laplace transformed $E(t)$, allowing for simple interconversion between the creep compliance and stress relaxation equations.

5.5 Convergence Studies

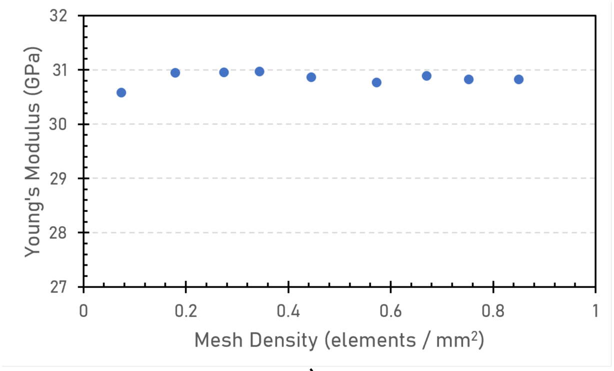
In this section, the influence of the finite element mesh and the microstructure domain size are assessed. In order to ensure the simulations are run with the optimal microstructure size and density of the finite element mesh, convergence studies were performed to locate the lowest values that yielded results consistent with higher values.

5.5.1 Finite Element Mesh Density

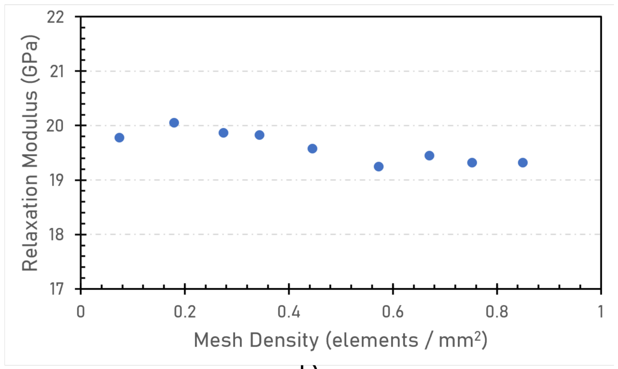
Increasing finite element mesh density directly raises the computational expense of the finite element simulation, due to increasing the number of equations to be solved in FEA. As such, minimizing the number of finite elements is advantageous. For both 2D and 3D, a single microstructure was converted to a finite element mesh with varying values of mesh densities, defined as

$$\text{Mesh Density} = \frac{\text{Number of Elements}}{\text{Area or Volume of Microstructure}}. \tag{5.8}$$

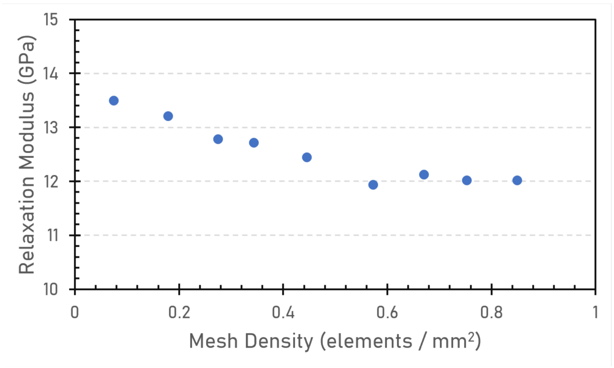
Each finite element mesh generated was used to run identical stress relaxation simulations; the resulting relaxation modulus was plotted against the mesh density at four time points. In Figure 5.6, convergence can be seen above 0.6 elements/mm^2 for a two-dimensional microstructure. Similarly, in the three-dimensional case, convergence can be seen above 0.6 elements/mm^3, as shown in Figure 5.7. Mesh densities above 0.8 elements/mm^3 were found to have poor mesh quality and thus were not included in this paper. In both Figure 5.6 and Figure 5.7, the convergence is most



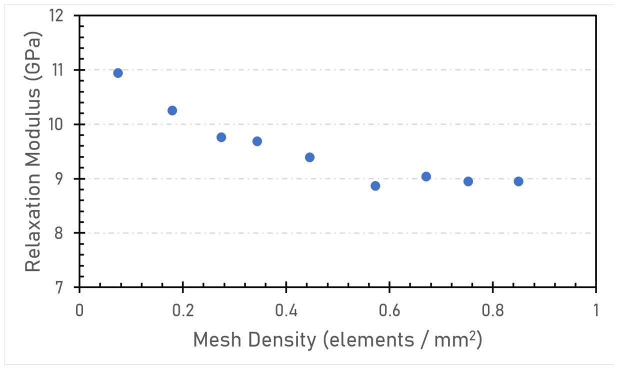
a)



b)



c)



d)

Figure 5.6: The Young's modulus a), and relaxation modulus at b) 100 days, c) 1,000 days, and d) 5,000 days of a two-dimensional square microstructure with an edge length of 90 mm plotted against the density of finite elements to determine convergence at a 38% coarse aggregate volume fraction

prominently seen in the relaxation moduli at 1,000 and 5,000 days, indicating that the mesh density becomes more influential at later time steps.

To further assess convergence, the L_2 norm of the strain tensor error was assessed, where the strain error was taken to be

$$\epsilon_u = \epsilon - \epsilon_a, \tag{5.9}$$

where ϵ_u is the strain error tensor, ϵ is the strain tensor, and ϵ_a is the analytical solution for the strain. As this doesn't problem doesn't have an analytical solution, the value of the analytical strain tensor is taken to be that of the strain using the highest mesh density. The ϵ_u tensor was used

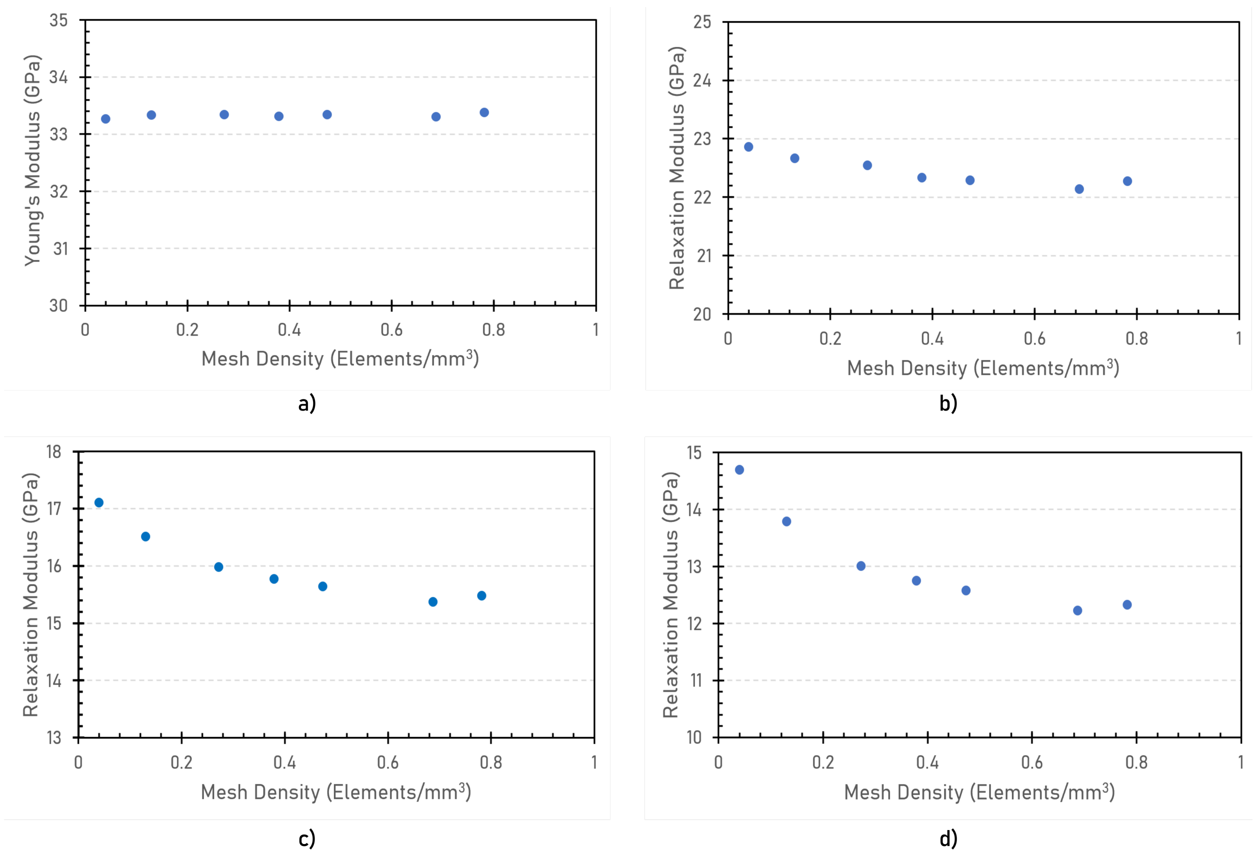


Figure 5.7: The Young's modulus a), and relaxation modulus at b) 100 days, c) 1,000 days, and d) 5,000 days of a three-dimensional cubic microstructure with an edge length of 75 mm plotted against the density of finite elements to determine convergence at a 38% coarse aggregate volume fraction

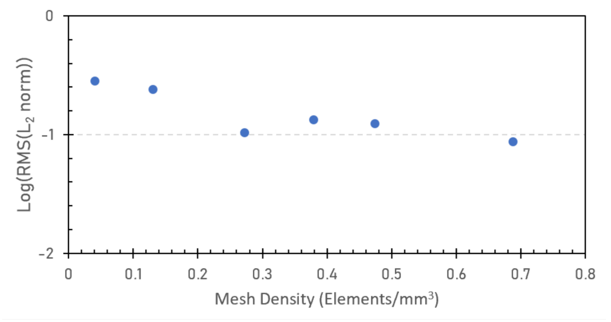
to calculate the L_2 norm,

$$||\epsilon_u||_{L_2} = \sqrt{\oint_V \epsilon_u \cdot \epsilon_u \, dV}. \tag{5.10}$$

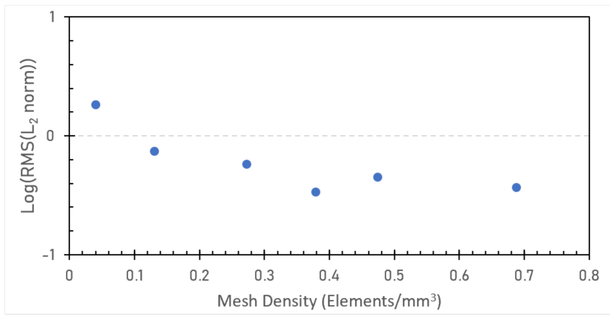
The root mean square (RMS) of the L_2 norm of the strain error tensor,

$$RMS = \sqrt{\frac{\sum_{n=1}^{N} ||\epsilon_u(i)||_{L_2}^2}{N}}, \tag{5.11}$$

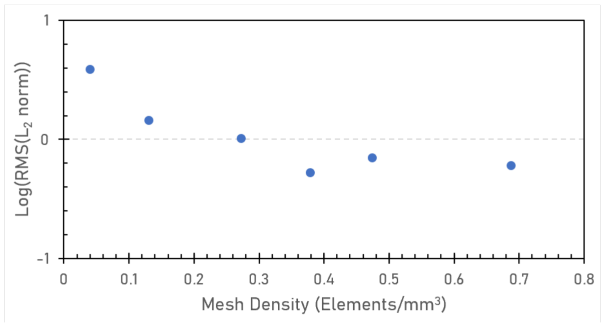
where N is the number of entries in the strain tensor. The RMS of the strain error tensor was plotting against the mesh density in Figure 5.8, where the convergence can be seen more clearly at earlier ages. The convergence is seen around 0.4 elements/mm^3 in Figure 5.8, lower than the value of approximately 0.6 elements/mm^3 seen in Figure 5.7. Figure 5.9 shows the data from Figure 5.8 plotted against the number of degrees of freedom in each mesh. Each node in the finite element mesh has six degrees of freedom - three translational degrees of freedom and three rotational. The RMS of the L_2 norm of the strain error tensor of the highest mesh density, approximately 0.8 elements/mm^{3}is zero, and thus the log value is negative infinity, and as such, it is not included on the graphs in Figure 5.8. To err on the side of caution, the convergence of the finite element mesh density will be taken as 0.6 elements/mm^3 moving forward.



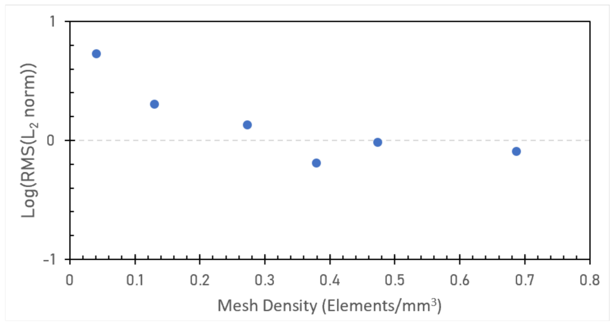
a)



b)



c)



d)

Figure 5.8: The RMS of the L_2 norm of the strain error at a) 0 days, b) 100 days, c) 1,000 days, and d) 5,000 days of a stress relaxation simulating using a three-dimensional cubic microstructure with an edge length of 75 mm plotted against the density of finite elements to determine convergence at a 38% coarse aggregate volume fraction

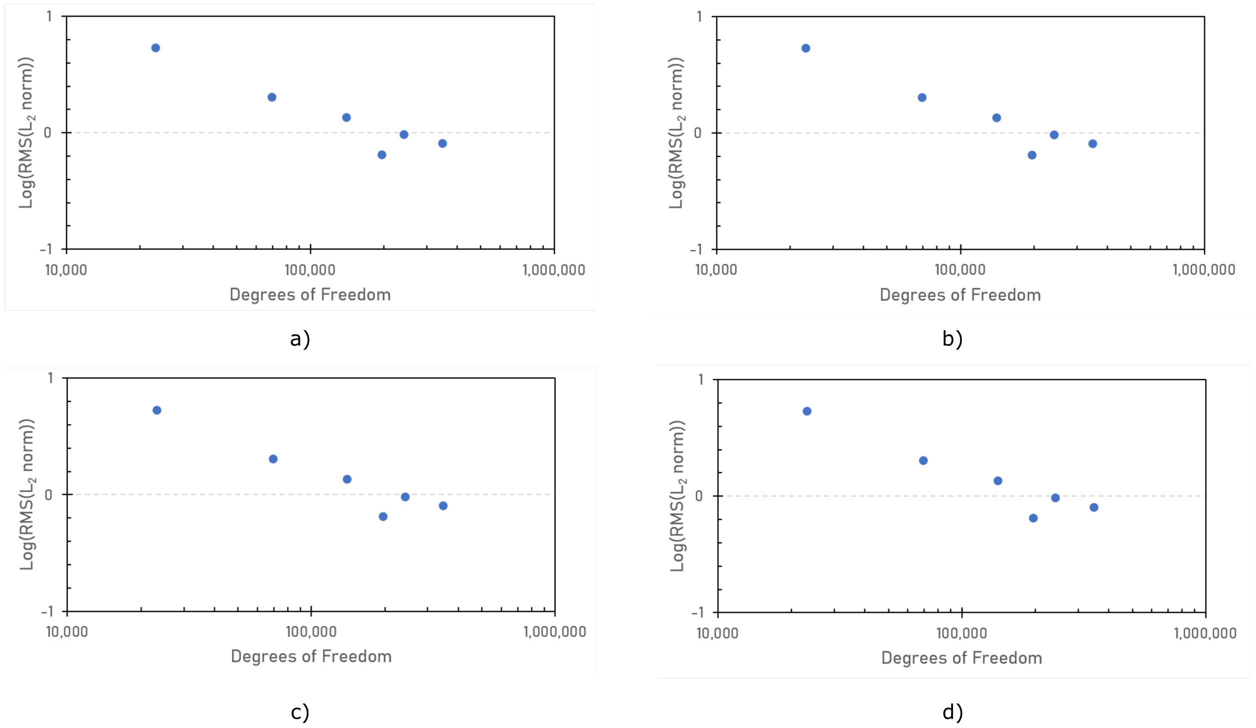


Figure 5.9: The RMS of the L_2 norm of the strain error at a) 0 days, b) 100 days, c) 1,000 days, and d) 5,000 days of a three-dimensional cubic microstructure with an edge length of 75 mm plotted against the number of degrees of freedom (DOF) on a log scale to determine convergence at a 38% coarse aggregate volume fraction

5.5.2 RVE Size

Five microstructures of 38% volume fraction were generated for each RVE size in both two- and three-dimensions. A finite element mesh density of 0.6 elements/mm^2 and 0.6 elements/mm^3 was used for the 2D and 3D microstructures respectively. Each was subjected to a FEM stress relaxation simulation, using a strain of 0.02%, the results were averaged across the five simulations and the resulting relaxation moduli were plotted at four time points to determine the RVE size at which convergence begins. In regard to the 2D microstructures, the RVE size doesn't show as clear a convergence trend as the mesh density convergence study did. Figure 5.10 shows the relaxation modulus at four different time points ranging from 0 days (the Young's modulus) to 5,000 days, showing the maximum difference between the data points to be about 1 GPa. On the other hand, the 3D simulations show a clear convergence at a domain size of 90 mm, as visible in Figure 5.11.

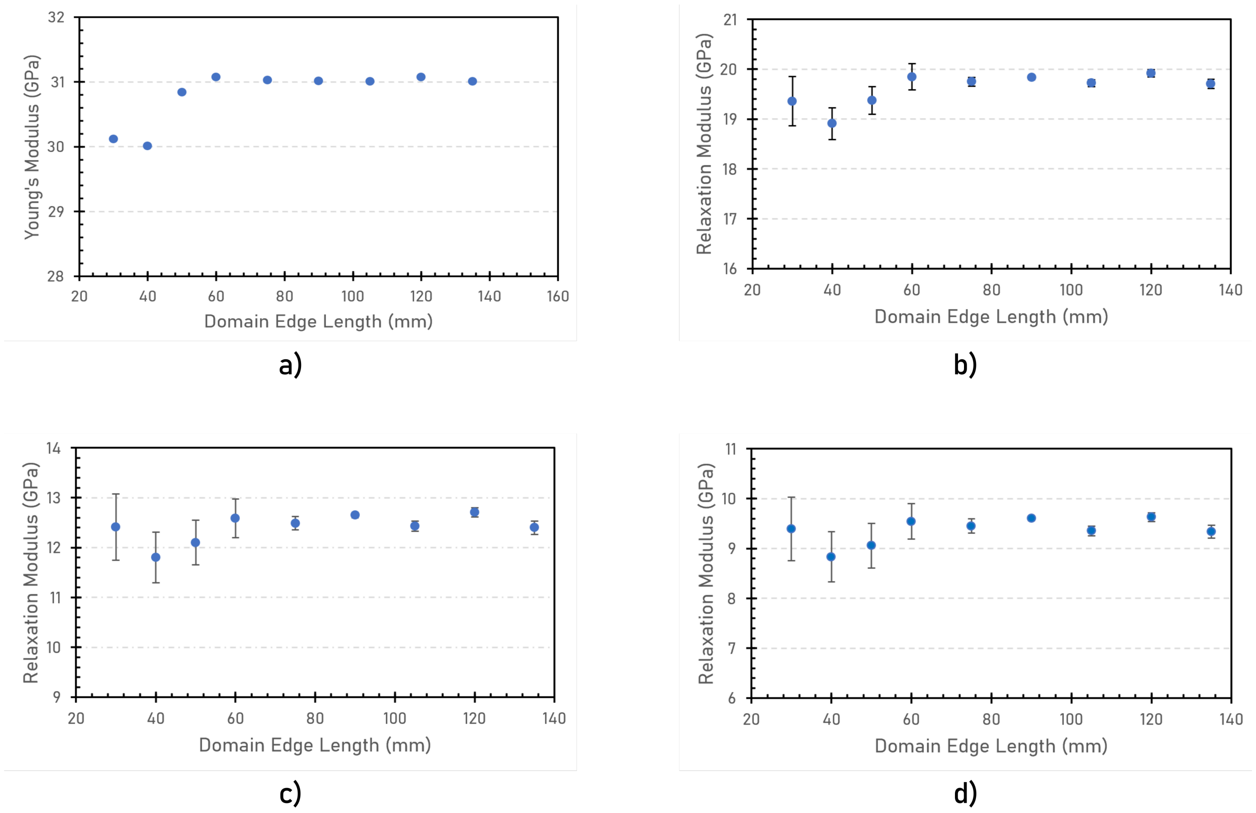


Figure 5.10: The Young's modulus a), and relaxation modulus at b) 100 days, c) 1,000 days, and d) 5,000 days of a two-dimensional square microstructure plotted against the domain edge length to determine convergence at a 38% coarse aggregate volume fraction and a finite element mesh density of 0.6 elements/mm^2. Error bars on plots b-d represent $\pm$ one standard deviation.

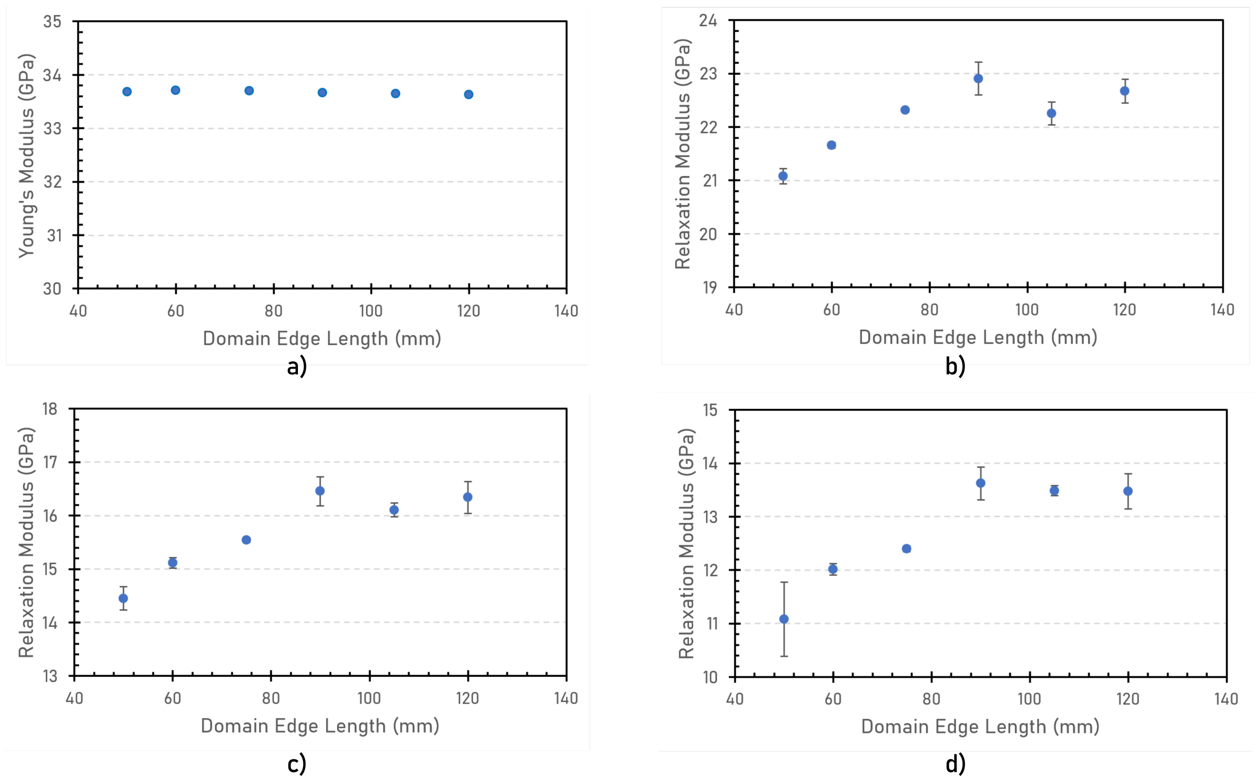


Figure 5.11: The Young's modulus a), and relaxation modulus at b) 100 days, c) 1,000 days, and d) 5,000 days of a three-dimensional cubic microstructure plotted against the domain edge length to determine convergence at a 38% coarse aggregate volume fraction and a finite element mesh density of 0.6 elements/mm^3. Error bars on plots b-d represent $\pm$ one standard deviation

5.6 Results

5.6.1 Comparison to Experimental Data

Comparison to experimental data began with plotting 2D and 3D simulation results with experimental creep compliance data. The standard error, defined as,

$$\text{Standard Error} = \frac{\text{Standard Deviation}}{\sqrt{\text{Number of Samples}}}, \tag{5.12}$$

is used to construct error bars on the experimental data to determine how the simulation data does or does not agree with the experimental data. As seen in Figure 5.12, data from both the internal strain gauges and the DEMEC strain gauges is presented alongside results from 2D and 3D simulations. The DEMEC strain gauge data represents only basic creep, as the B3 model was used to subtract off creep due to surface drying [132]. Creep compliance predicted by the 3D simulations falls within the experimental data bounds throughout all 800 days, while the 2D simulation results increasingly deviate from the experimental data after 300 days.

Moving forward, simulations of 27+ years of concrete creep were conducted using 2D square and 3D cubic microstructures, all with an edge length of 90mm and a finite element mesh density of 0.6. Although the 2D simulations did not show good agreement with experimental data in Figure 5.12, they were used in the long-term simulations in order to assess the level of overprediction over time. The TTS principle was used to construct a concrete creep compliance master curve from the experimental creep data collected from the DEMEC strain gauges at 20^o C and 60^o C. The shift factor for the 60^o C data is 12.5, thus the master curve for creep at 20^o C extends to 10,000 days, or approximately 27.4 years. Further information on the application of the TTS principle and its application to the creep of cementitious materials can be found in Baranikumar et. al (2020) [45]. Figure 5.15 displays the master curve, in black, plotted with the average of five simulations, where excellent agreement can be seen with the 3D simulations.

The close agreement between the extended experimental concrete data and the 3D simulations

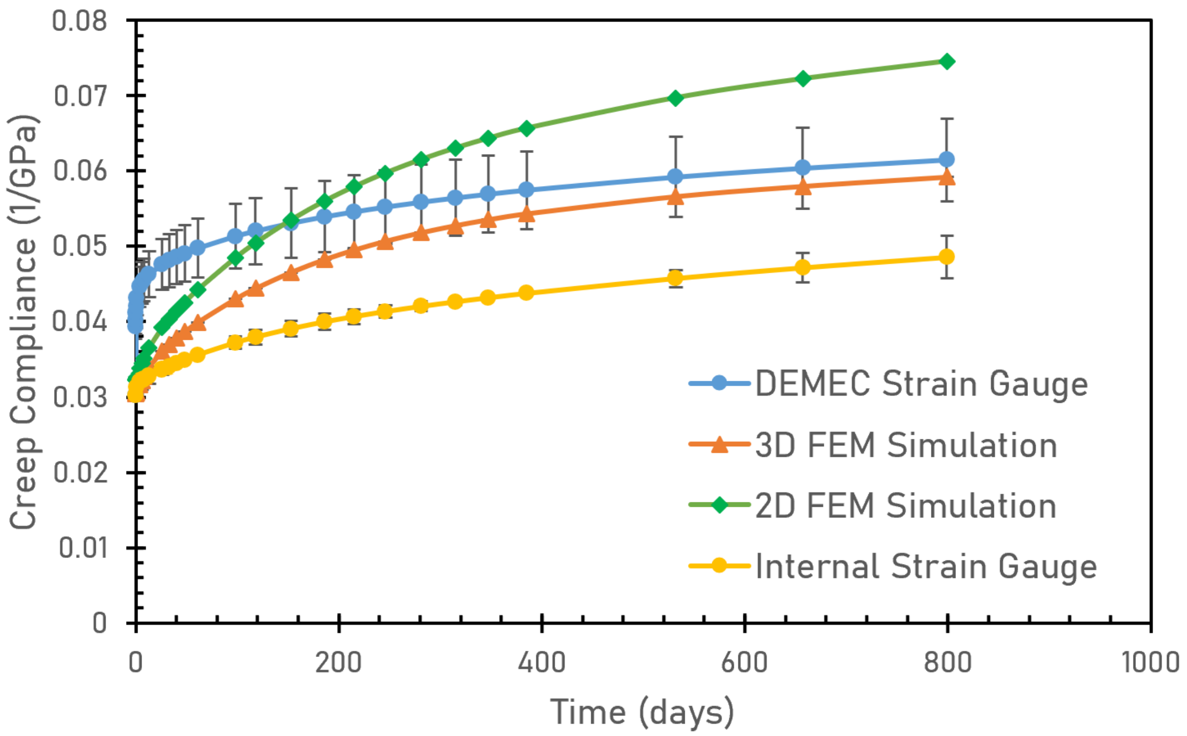


Figure 5.12: Creep compliance data at 20^o C over 800 days, showing a comparison of the experimental concrete creep data to the 2D and 3D simulations. Error bars represent the standard error of the experimental data. The 2D and 3D simulation data points are averages of five 90mm edge length square and cubic microstructures, meshed with a density of 0.6 elements/mm^2 and elements/mm^3 respectively.

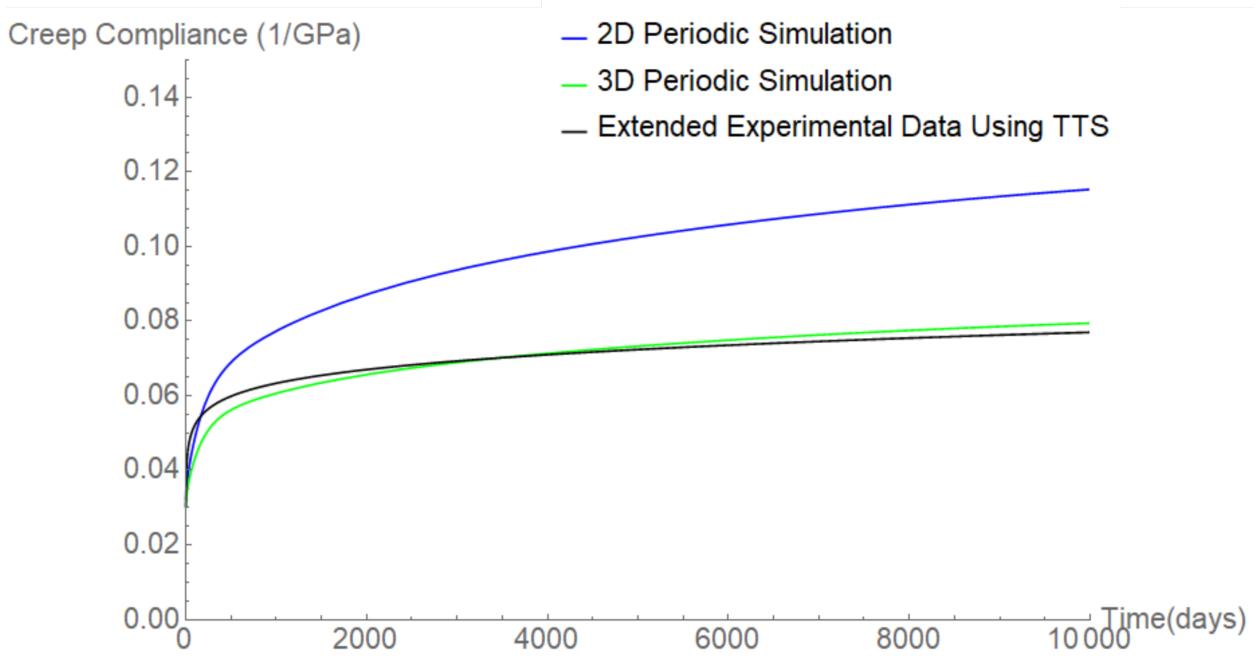


Figure 5.13: The creep compliance over 10,000 days modeled in 2D and 3D, plotted with experimental data extended using the TTS principle.The 2D and 3D simulation data points are averages of five 90mm edge length square and cubic microstructures meshed with a density of 0.6 elements/mm^2 and elements/mm^3 respectively.

indicates the assumptions to treat the aggregates as linear elastic and neglect the ITZ did not impact the accuracy of the 3D concrete creep compliance simulations. This agrees with the conclusions of Barnachy-Bary et. al (2015), which suggested the ITZ has little influence in creep simulations at this scale [133], at least at the scale of the coarse aggregate.

5.6.2 Influence of Dimensionality

The 2D simulations provide a reasonable prediction of the Young's modulus, falling on average 6-7 % lower than the experimental values. The Young's modulus is found to be 33 GPa experimentally, and the 2D and 3D simulations predict a Young's modulus of 31 and 33 GPa respectively. As time goes on, the gap between the 2D simulations' creep compliance and the experimental data continues to grow, reaching a difference of 49% at 10,000 days, as shown in Figure 5.15. This is an important finding, as it shows that the 2D simulations may be useful in approximating the Young's modulus of concrete but, are likely unsuitable to predict viscoelastic behavior. On the other hand,

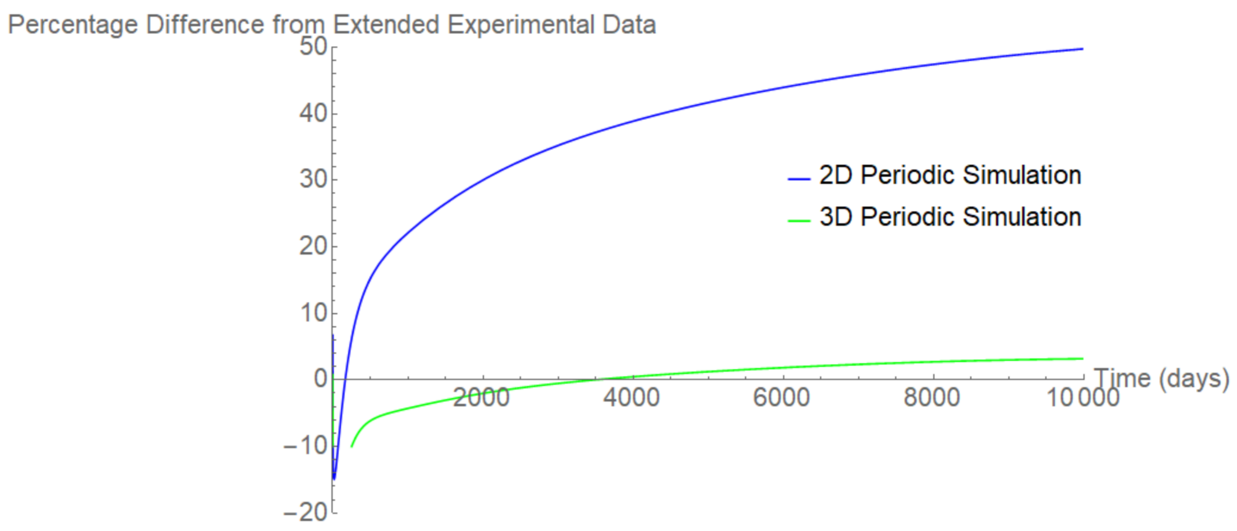


Figure 5.14: The percentage difference between the 2D and 3D simulations from the extended experimental data over 10,000 d

the 3D simulations remain in close agreement with the extended experimental data, with a difference between the two curves of 3% at 10,000 days. Figure 5.14 shows the percentage difference between the 2D and 3D simulation results and the concrete master curve over time. Beyond 4,000 days, the 2D simulation diverges from the concrete master curve at a rate of approximately 0.66% per year. Major differences in the creep compliance predictions between the 2D and 3D simulations indicate that the viscoelastic behavior is not accurately captured by a two-dimensional stress state. Much like the findings of Huang et. al (2016) and Shen (2007), the 2D simulations in this work do not accurately represent the stress-strain behavior, whereas the 3D counterpart simulations do [134, 111]. Huang et. al (2016) also showed that the 2D concrete simulations underpredict the Young's modulus on average by 5.6%, echoing the results presented in this chapter. The consequences of this finding include the demonstrated need to model viscoelasticity of concrete in 3D, even though it is slower and computationally more expensive than 2D models.

5.6.3 Influence of the Mortar's Poisson's Ratio

Viscoelastic Poisson's ratio (VPR) of concrete has been published in scientific literature with a wide variety of results - studies showing it increases [], decreases [], and remains constant [**?**] over

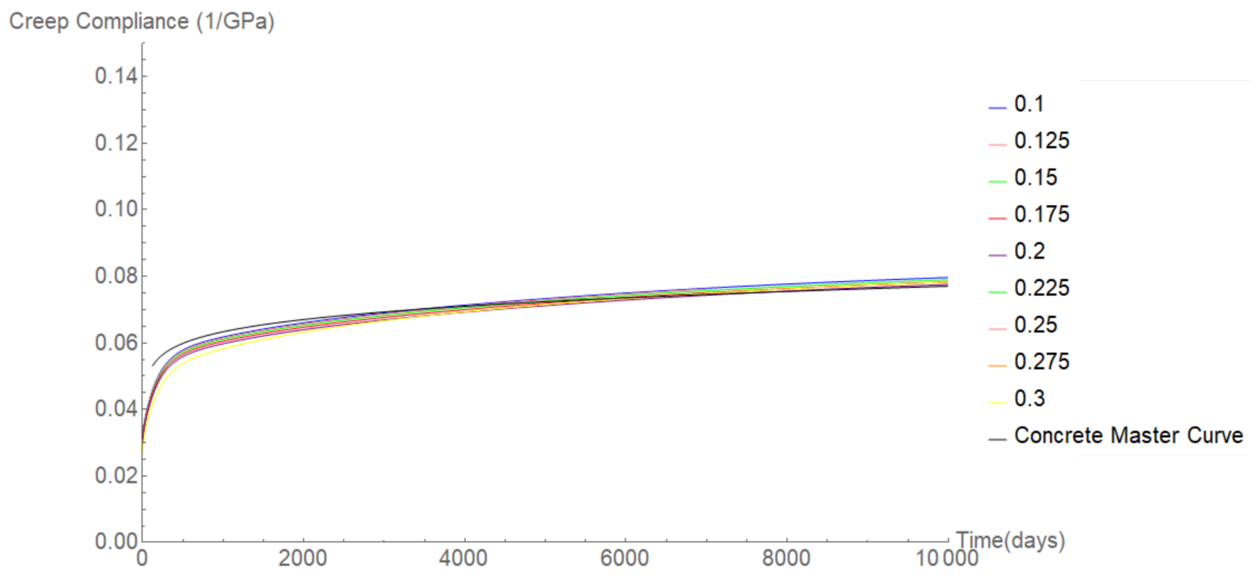


Figure 5.15: The creep compliance over 10,000 days modeling using a 75mm cubic microstructure with periodic boundary conditions. The Poisson's ratio of the mortar phase was altered in increments from 0.1 to 0.3

time. The experimental mortar data showed the Poisson's ratio to be nearly constant over time through the use of confined creep tests, however in order to determine the influence that the VPR plays on the magnitude of concrete creep, the Poisson's ratio of the mortar phase was altered over nine increments ranging from 0.1 to 0.3. The maximum difference between the highest and lowest creep compliance value at 10,000 days was 7.9%. This indicates that there is some influence of the VPR on the unixial creep compliance, however it is relatively minor. A key takeaway from these results is that whether the Poisson's ratio remains the same, or changes with time, it is a small factor in predicting the overall creep of concrete. Thus, an estimation of a Poisson's ratio, or even simply an assumption that it remains constant could be considered acceptable for many applications.

5.7 Summary

Simulation tools can be very useful in the world of concrete, and composites in general, to obtain information on viscoelastic behavior. As lab tests on materials with long-term viscoelasticity, such as concrete, are very slow to generate data, the methods presented in this paper offer a good alternative to experiments by demonstrating the process to generate morphologically representative microstructures and using FEM to predict long-term viscoelastic behavior. The following conclusions can be drawn from this work:

- Periodic boundary conditions were successfully applied both the aggregate geometry and mathematically. The geometrical periodicity was exact, while the stress and strain along opposite edge elements on 2D and 3D microstructures were compared and found to have good, but not perfect, agreement, indicating approximate periodicity. Small differences in the stresses and strains resulted from the differences in size and shape of the edge elements between the opposite edges.

- Convergence of the finite element mesh density was determined to be approximately 0.6 elements/mm^2 and 0.6 elements/mm^3 for 2D and 3D microstructures respectively. Simulations performed with lower mesh densities showed up to 20 % difference from the converged results, showing the importance of using a proper finite element mesh.

- Convergence of the RVE size was found to be at approximately 60 mm for 2D and 90 mm 3D microstructures, corresponding to an edge length approximately 2x and 3x longer than the largest aggregate's length for the 2D and 3D microstructures, respectively.

- Both the convergence of the RVE size and the finite element mesh density show a time-dependence of the value of convergence, indicating the need for convergence studies of time dependent properties to assess multiple time points. Additionally, the convergence of the Young's modulus does not represent the convergence of the creep/relaxation behavior, further reinforcing the need to assess the convergence over time.

- Both 2D and 3D simulations accurately predict Young's modulus within 7 % and 3 %, respectively. However, the 2D simulations greatly overpredict creep, with the overprediction of the creep compliance increasing over time to about 50 % after 10,000 days. Therefore, one should not use 2D models to predict long-term creep, but rather make the computational time and resource investment needed to solve 3D models.

- The treatment of aggregates as linear elastic and the decision to neglect the ITZ regions around the coarse aggregates in this work appear to be valid simplifications given the close agreement between the 3D simulation results and experimental data. Any significant effect of the ITZ regions around sand grains were not studied but were swept into the mortar matrix creep data.

- The Poisson's ratio was found to have an impact on the creep compliance predictions, however the difference between the largest, 0.3, and smallest, 0.1, Poisson's ratio simulated was under 8% after 10,000 days of creep. The implications of this finding are that if the Poisson's ratio of a sample are not known, the value can be estimated with a non-negligible but relatively small risk of error.

- Due to the time-dependent nature of concrete creep, it is challenging to collect creep behavior data in a timely fashion. This approach has successfully upscaled experimental mortar creep data, which is significantly easier to test than concrete, using the TTS principle. This upscaled mortar data can then be used in the concrete models to give long-term predictions.

6. INFLUENCE OF BOUNDARY CONDITIONS ON MODELING OF CONCRETE CREEP AND RELAXATION

Although the topic of experimental sample preparation has been an area of focus in the civil engineering community, study regarding the RVE size and impact of sample geometry on the stiffness and viscoelastic behavior has not been well studied. This is discussed further in following two sections.

6.1 Microstructure Generation

The cast, cored, and periodic microstructures utilized in the simulations presented in this chapter were generated using the methods described in Chapter 3. The cast microstructures induced a ’wall effect’, where the aggregates pack less-densely in regions close to the boundary [10]. This phenomenon can be seen in Figure 6.1, where the volume fraction of a cubic microstructure of each boundary condition category is plotted along the Z-axis. In Figure 6.1, the volume fraction of aggregate drops as the Z-coordinate approaches the top or bottom of the cube.

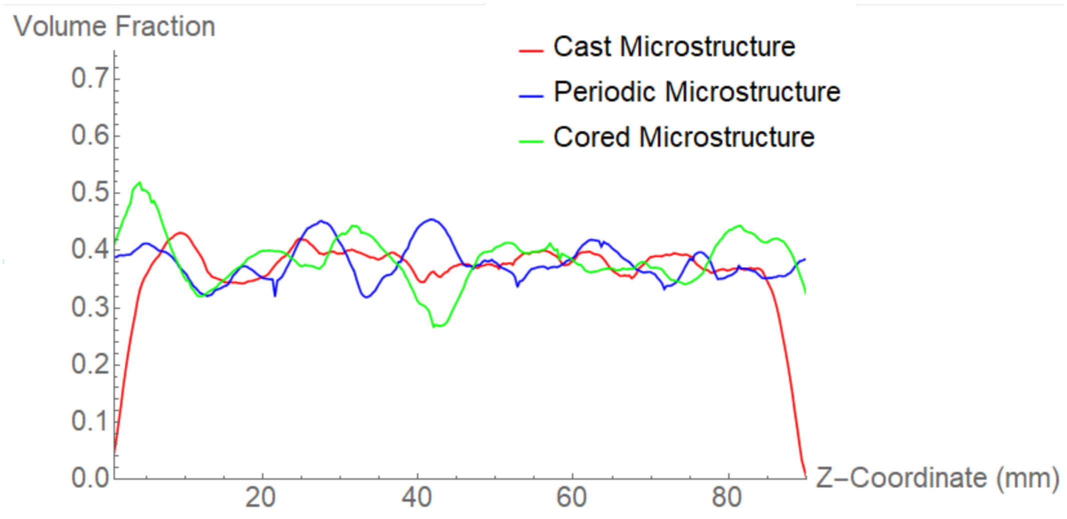


Figure 6.1: Volume fraction of a 38% volume fraction 90mm cubic periodic, cast, and cored microstructure as a function of the Z-coordinate

6.2 Convergence Studies

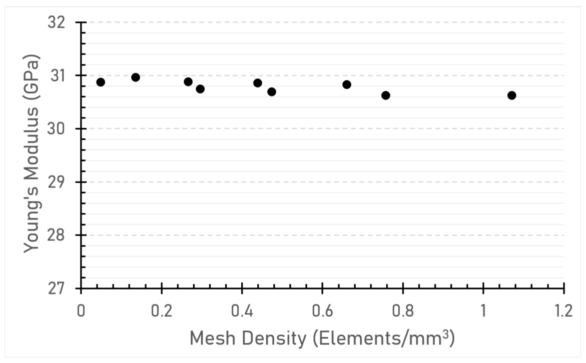
In this section, the influence of the finite element mesh and the microstructure domain size are assessed in the same manner presented in Chapter 5.

6.2.1 Finite Element Mesh Density

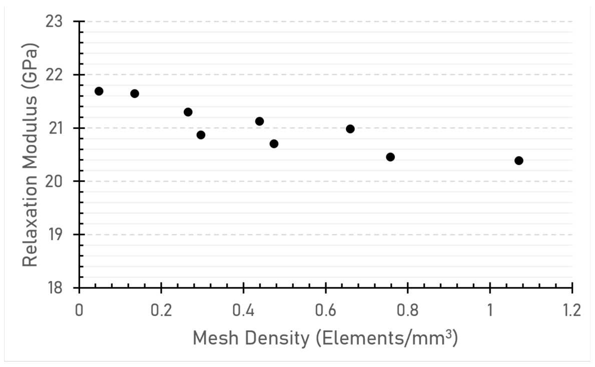
As discussed in previous chapters, using a proper finite element mesh with a density above the convergence value is important, particularly in creep behavior past 5,000 days. In Chapter 5, the coarsest and finest meshes predicted up to a 20% difference in creep compliance. A mesh density convergence study was performed on a 90mm cored microstructure of 38% coarse aggregate volume fraction. In Figure 6.2 convergence can be seen above a density of 0.75 elements/mm^3. Similar to the mesh density convergence study in Chapter 5, the difference in the simulated creep compliance between the finest and coarsest mesh is over 20%.

A mesh density convergence study was also performed on a cast cubic microstructure with an edge length of 105mm. As seen in Figure 6.3, the convergence trend is not as pronounced as the periodic and cored mesh density convergence studies. In Figure 6.3 d), the data can be seen to level off around 0.45 elements/mm^3, potentially as early as 0.3 elements/mm^3, depending on interpretation.

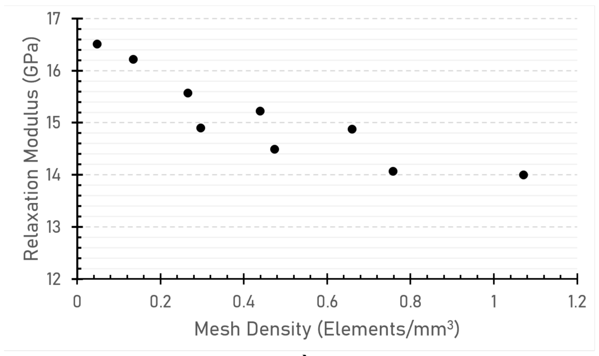
In Chapter 5, the periodic microstructures showed convergence at a density of about 0.6 elements/mm^3. The cored microstructures may contain more aggregate-boundary intersections because during the particle placement process, the aggregate does not return on the opposing boundary - meaning it does not need to fit within the space associated with the opposing boundary. As a result, the higher number of aggregate-boundary intersection would increase the number of elements needed to properly discretize the interface between the aggregate and the domain boundary(ies). Cast microstructures, on the other hand, are composed of aggregates that are fully contained within the domain, thus do not have aggregate - domain boundary interfaces. Additional study is needed to confirm or deny these hypotheses.



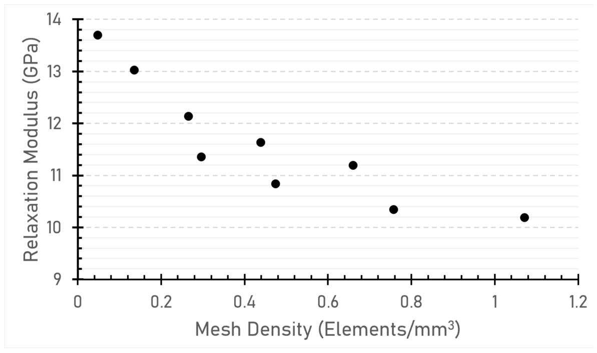
a)



b)



c)



d)

Figure 6.2: The Young's modulus a), and relaxation modulus at b) 100 d, c) 1,000 d, and d) 10,000 d of a 3D cored cubic microstructure with an edge length of 90 mm plotted against the density of finite elements to determine convergence at a 38% coarse aggregate volume fraction

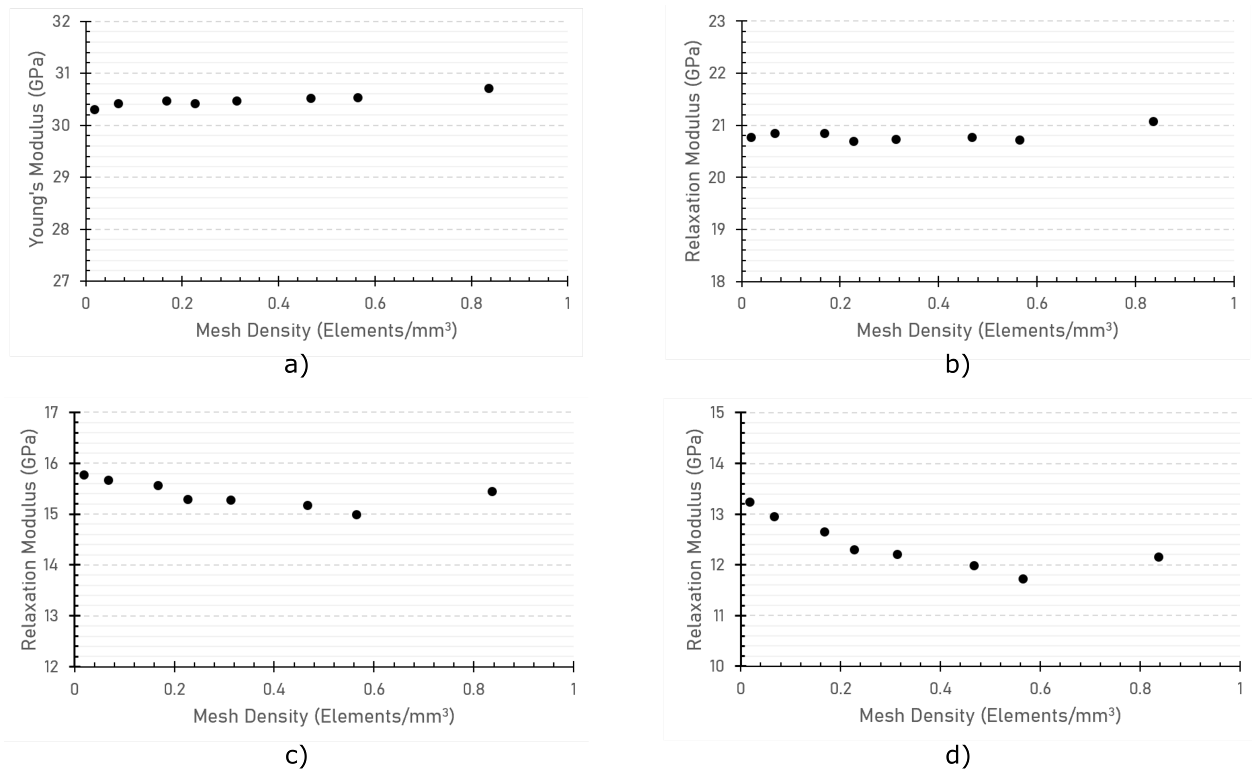
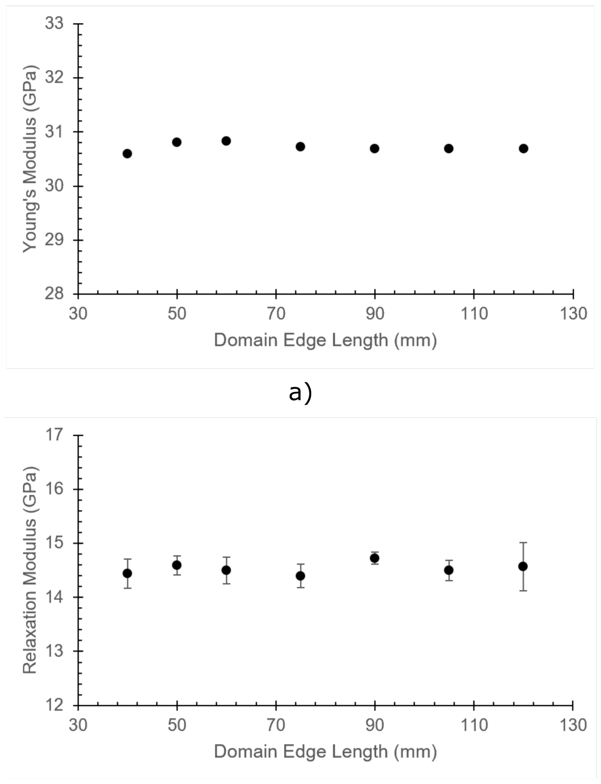


Figure 6.3: The Young's modulus (equivalent to the relaxation modulus at $t = 0$) a), and relaxation modulus at b) 100 d, c) 1,000 d, and d) 10,000 d of a 3D cast cubic microstructure with an edge length of 105 mm plotted against the density of finite elements to determine convergence at a 38% coarse aggregate volume fraction

6.2.2 RVE Size

To determine the appropriate RVE size for the cored and cast cubic microstructures, RVE convergence studies were carried out. Figure 6.4 shows the relaxation modulus at four different time points plotted against the domain edge length for a variety of cored microstructure sizes. Unlike the results of the periodic microstructure RVE size in Figure 5.11, a clear convergence trend is not seen in the cored RVE convergence study. Looking closely, the data appears to level out around 90mm, therefore this will be considered the cored cubic microstructure's RVE size moving forward.

An RVE size convergence study was also performed on a cast cubic microstructure, results shown in Figure 6.5. Much like the cored RVE convergence study, a distinct convergence is not seen in the data presented in Figure 6.5. Micorstructures with an edge length smaller than 50mm were unable to be generated at a 38% volume fraction, as the code stalled at lower volume fractions due to difficulty of placing aggregates into a small domain without intersecting one another or the boundaries of the domain. Improvements in the code to overcome this limitation is an area of future work.


a) b) c) d)

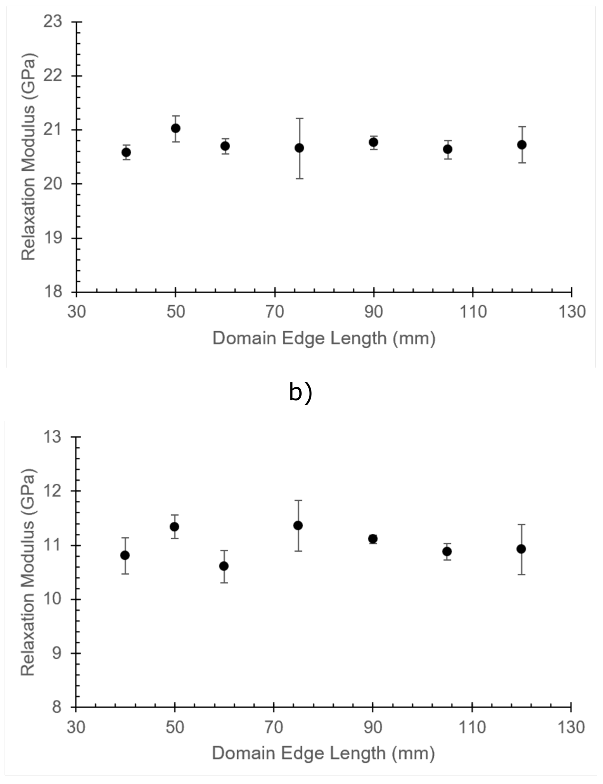
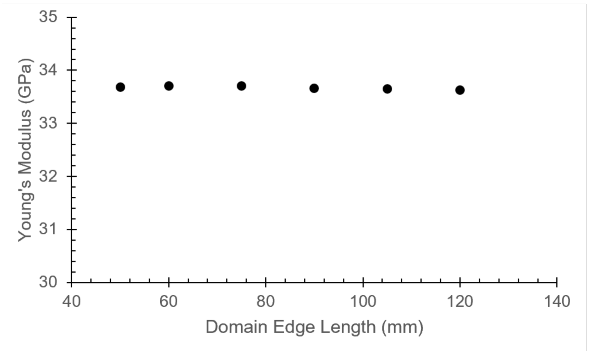
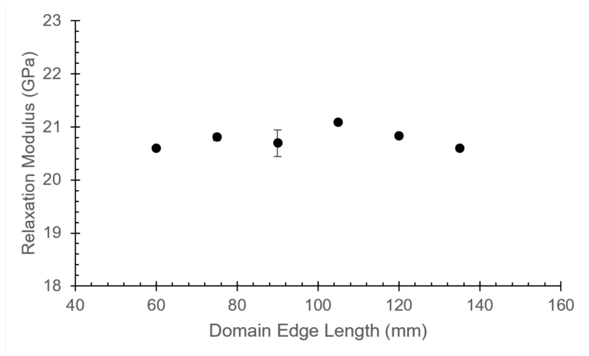


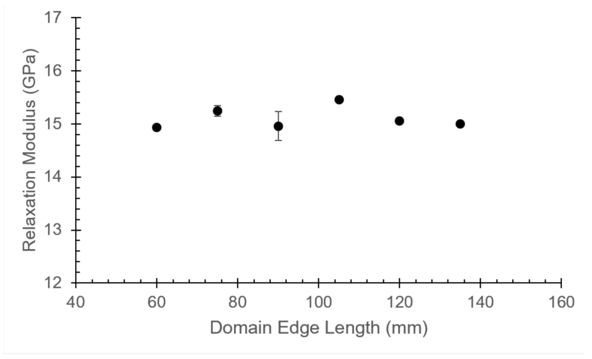
Figure 6.4: The Young's modulus a), and relaxation modulus at b) 100 d, c) 1,000 d, and d) 10,000 d of a 3D cored cubic microstructures plotted against the domain edge length to determine convergence at a 38% coarse aggregate volume fraction, meshed with a density above 0.8 elements/mm^3. Each data point is the average of three simulations, with error bars representing $\pm$ one standard deviation.



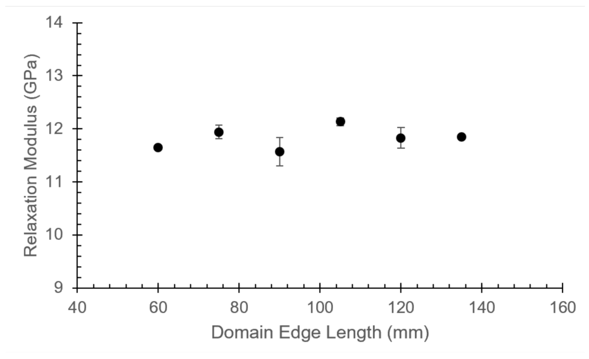
a)



b)



c)



d)

Figure 6.5: The Young's modulus a), and relaxation modulus at b) 100 d, c) 1,000 d, and d) 10,000 d of a 3D cast cubic microstructures plotted against the domain edge length to determine convergence at a 38% coarse aggregate volume fraction meshed with a density above 0.5 elements/mm^3. Each data point is the average of three simulations, with error bars representing $\pm$ one standard deviation.

6.3 Results

6.3.1 Influence of Morphological Boundary Conditions

The simulation results for the cast, cored, and periodic cubic microstructures are plotted next to the experimental data extended with the TTS principle [45] in Figure 6.6. The periodic microstructures, as discussed in Chapter 5 shows the closest agreement with the experimental data, followed by the cast simulations. The cored simulations display a notably higher level of creep compliance, measuring 13.9% higher creep compliance than the extended experimental data at 10,000 days. The cast microstructures predict a compliance value 5.5% higher. Error bars are not included on Figure 6.6 because the simulations are very consistent, with a standard deviation of less than 0.5% at 10,000 days for both the cast and cored simulations, resulting in error bars that are difficult to see on the graph.

The cast microstructures predict the concrete creep compliance behavior well, which is expected because the experimental samples were cast, albeit into cylinders. Experimental data comparing creep behavior of cubes vs cylinders has not been located during an extensive literature review. The cored samples, on the other hand, have a notably higher over prediction of creep. As previously discussed in Section 6.2.2, it is hypothesizes that the cored samples have more aggregates that intersect the boundary, and thus more boundary-aggregate interfaces. The creep of the microstructures may be sensitive to the number of aggregate-boundary intersections. Additional research is needed to more fully investigate this theory.

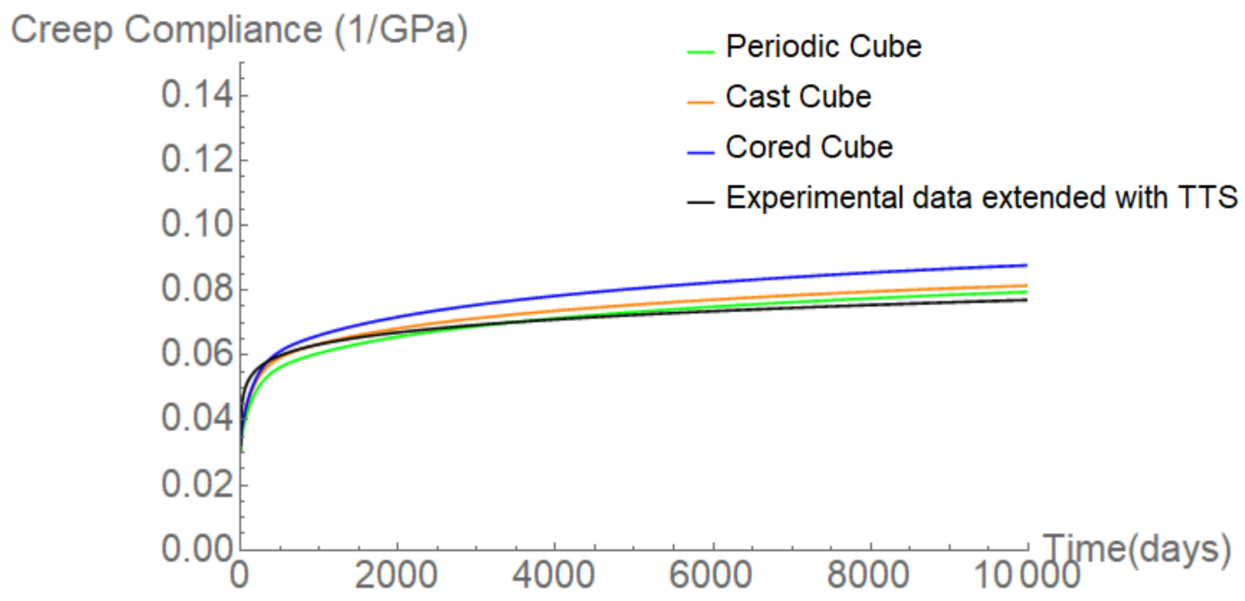


Figure 6.6: The creep compliance over 10,000 d modeled with 3D periodic, cast, and cored boundary conditions, plotted with experimental data extended using the TTS principle. Each curve is generated from the average of three simulations for each microstructure category, where the periodic and cored microstructures have an edge length of 90mm and the cast microstructures have an edge length of 105mm.

6.3.2 Influence of Periodic Boundary Condition Constraint Equations

Initially discussed in Chapter 5, the software used to generate the finite element mesh, OOF3D doesn't have the capability to generate a periodic mesh, therefore even though the microstructure is perfectly periodic, variations and differences between the meshes on opposing boundaries exists. A MATLAB script was written to automatically find node pairings on opposing boundaries and write constraint equations into the Abaqus input file. This code can be found in Appendix B.1. In order to asses the influence of these periodic boundary condition constraint equations imposed on the periodic microstructures, a 90mm periodic cubic microstructure was simulated with and without the constraint equations. The resulting creep compliance data is shown in Figure 6.7.

The differences between the simulations with and without periodic constraint equations can be seen more visibly on a log time scale, shown in Figure 6.8.

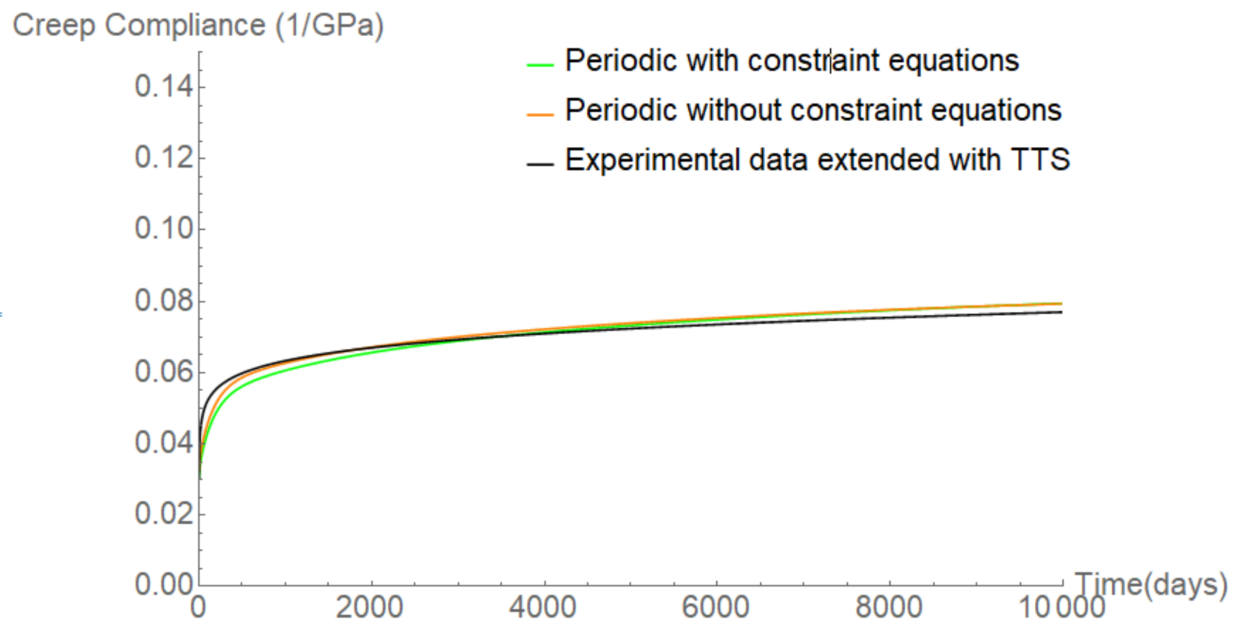


Figure 6.7: The creep compliance over 10,000 days simulated from a 90mm cubic periodic microstructure with and without periodicity constraint equations, plotted alongside experimental data extended using the TTS principle.

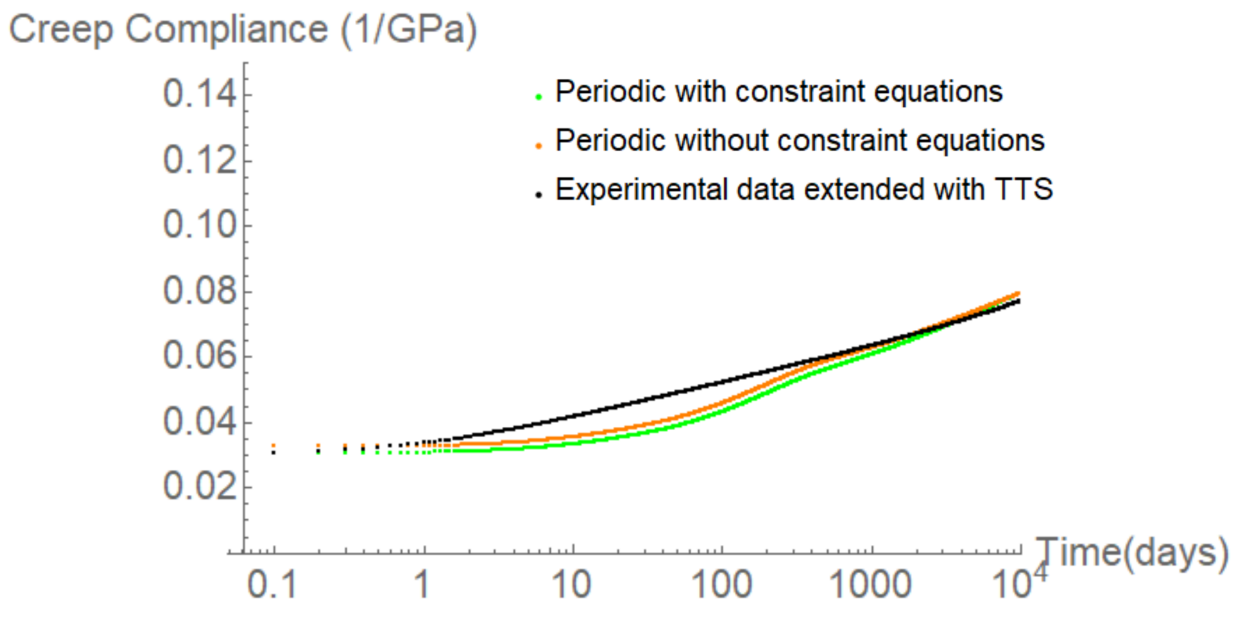


Figure 6.8: The creep compliance over 10,000 days simulated from a 90mm cubic periodic microstructure with and without periodicity constraint equations, plotted alongside experimental data extended using the TTS principle on a logarithmic time scale.

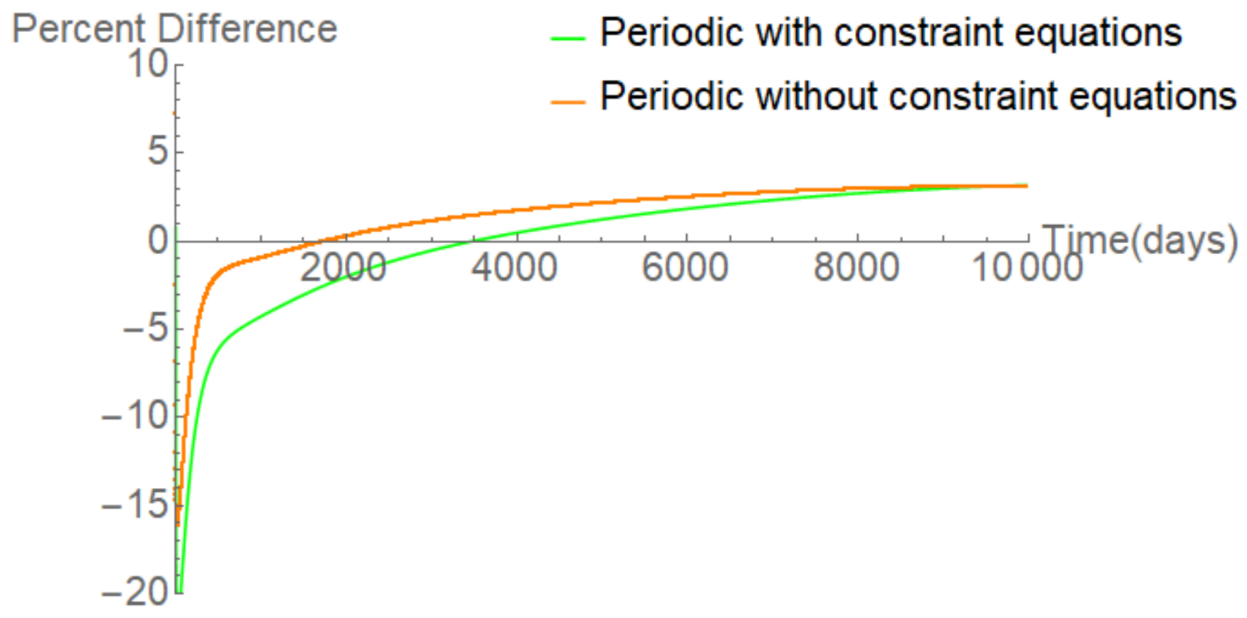


Figure 6.9: The creep compliance over 10,000 days simulated from a 90mm cubic periodic microstructure with and without perioidicity constraint equations, plotted alongside experimental data extended using the TTS principle.

The Young's modulus is found to be 33.5 GPa and 30.8 GPa for the simulation with and without the periodicity constraint equations, respectively. Figure 6.9 shows the percentage difference between the two periodic simulations, with and without constraint equations, and the extended experimental data over 10,000 days. Interestingly, the periodic simulation without constraint equations shows closer agreement with the extended experimental data in the first 3,000 days. This is believed to be the result of the imposed periodicity constraints equations constraining the microstructure's relaxation, as the un-constrained simulation immediately has a higher level of creep. This higher level of creep in the first 3,000 days matches the experimentally measured behavior closely. Beyond 3,000 days, agreement switches, where the simulation with constraint equation shows closer agreement to the extended experimental data. Ultimately the two data sets converge to nearly the same value, with the difference in the creep compliance between the simulations with and without the constraint equations being 0.2% at 10,000 days.

These results indicate that the inclusion of the periodicity constraint equations improves the accuracy of the Young's modulus prediction, however it doesn't play a significant role in the long-term prediction of creep behavior. For modeling situations where periodic boundary condition constraint equations are difficult to implement, this data indicates the constraints *may* be neglected, especially if the simulation is long-term.

6.4 Summary

In this chapter, cast and cored cubic microstructures were assessed through RVE size and mesh density convergence studies. The simulated creep compliance was subsequently compared to the results of periodic microstructure simulations from Chapter 5 and extended experimental data. Additionally, the periodic microstructures were simulated both with and without the periodic boundary condition constraints written in to the Abaqus input file. The following conclusions can be drawn from the data presented in this chapter:

- The mesh density convergence studies revealed that the cast microstructures converge at a slightly lower value, 0.5 elements/mm^3, than periodic microstructures, while the cored microstructures converge at a value closer to 0.75 elements/mm^3.

- The cored microstructures over predict creep by about 13-14%. The cast microstructures also over predict, albeit a lower value of 5-6%.

- More study is need to understand the underlying mechanisms that prevent a clear RVE convergence trend in cored and cast cubic microstructures

- The periodic constraint equations play a notable role in constraining the early creep/relaxation of the concrete, resulting in a 3 GPa difference in the prediction of Young's modulus. However this difference decreases with time, to a point where the creep compliance at 10,000 days is 0.2% different between the simulations with and without the constraint equations. This suggests that for shorter term simulations, the constraint equations are important to implement, but they could potentially be neglected in long term creep modeling.

- Future work should include the simulation of cylindrical microstructures to further the understanding of geometry and boundary conditions in this context. At the time of this dissertation, a suitable finite element meshing approach for the cylindrical microstructures has not been found, however performing both cylindrical and cubic simulations would provide insight into the comparison of different experimental sample geometries.

7. SHORT-TERM RELAXATION MODELING OF MORTAR

Given the success of upscaling mortar experimental data to concrete data through computational approaches, described in Chapters 5 and 6, these concepts were applied to upscaling cement paste data to mortar level data. In this approach, mortar microstructures are generated by randomly placing virtual fine aggregates into a continuous cement past matrix. These microstructures are subsequently subjected to a FEA stress relaxation simulation, where experimental cement paste data is used as input. The results of these simulations are then used to describe the mortar material behavior in a concrete microstructure simulation, which is carried out in a similar manner to the simulations described in Chapter 5.

7.1 Mortar Microstructure Generation

As the convergence studies performed in Chapters 5 and 6 showed convergence of an RVE size of approximately 3x the largest aggregate, this rule of thumb is used in this section. A set of fine aggregates were reconstructed for use in generating mortar microstructures, with the largest aggregate measuring about 0.65mm, thus a 2mm domain was chosen for this work. The reconstructed sand grains are then used to generate a mortar microstructure by randomly placing them into a periodic, cubic domain until the desired volume fraction is reached, using the same approach outlined in Chapter 3.

In this work, two different mortars are simulated - one with a w/c ratio of 0.4 and volume fraction of 47% fine aggregates, loaded 3 days after mixing and the other with a w/c ratio of 0.5 and a volume fraction of 50% fine aggregates, loaded 28 days after mixing. Corresponding cement paste experimental data is used as input to describe the microstructure matrix [46]. Assessing two different kinds of mortars will improve the understanding of how this model and approach can be applied to modeling mortar viscoelasticity at both early and mature ages of mortar. A completed mortar microstructure of 47% volume fraction is seen in Figure 7.1.

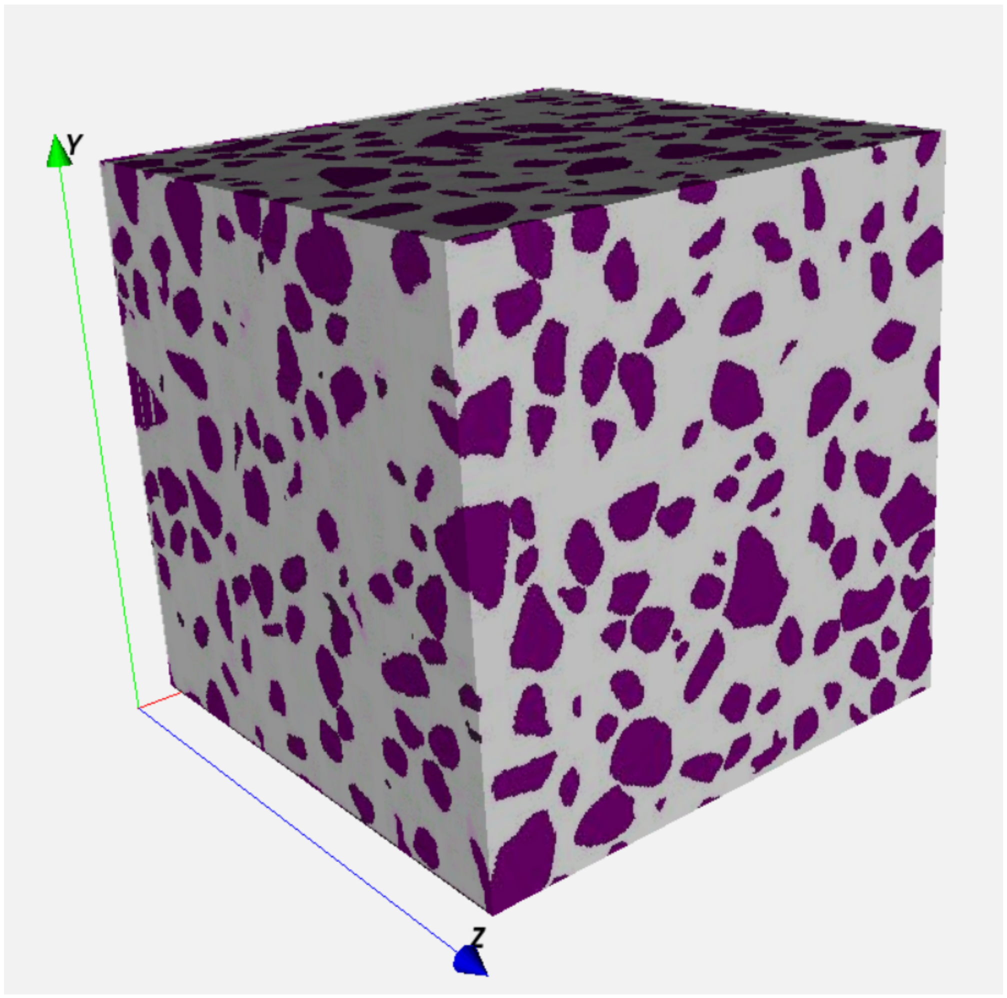


Figure 7.1: A mortar microstructure.

7.2 Description of Viscoelastic Cement Paste

Cement paste creep data from experimental work described in the previous section is used to calculate the cement paste's stress relaxation function, and the associated bulk and shear relaxation functions. This data originates in Tavares and Grasley (2020), where further details regarding the sample preparation and experimental procedure are described. [46]. The experimental procedures are briefly described here for context.

7.2.1 Experimental Sample Preparation

Cement pastes and mortars with w/c equal to 0.40 and 0.50, respectively, were batched in a 6-Quart stand mixer with a stainless-steel container, using Portland Cement Type I/II and tap water [130]. The mortar mixes included a river sand sieved to the #16 sieve with a sand to cement mass ratio (s/c) of 2.0.

Mortar samples were cast in 76 x 156 mm cylinders to be tested for Young's modulus and compressive strength. Samples for stress relaxation tests were cast in plastic tubes with 11 x 203 mm dimensions, from which 22 mm long segments were cut. Three replicate samples were tested for each experiment at each age. Once filled with fresh material (and before being cut) the top ends of the tubes were sealed with aluminum foil and electric tape as the tubes were placed horizontally in a roller and subjected to a slow rotation, promoting a curing process that enables a homogeneous w/c ratio distribution throughout the tube. Sealing the tubes ensures that fresh material stays inside the mold during rotation while simultaneously preventing moisture loss during hydration.

All samples were cured at room temperature for 24 hours. The exposed surfaces of the 76 x 152 mm cylinder molds were wrapped with plastic to prevent moisture loss. After the first 24 hours, all specimens were moved to an environmental chamber with 98% relative humidity at a temperature of 23^{o}C for a controlled curing process. All specimens were soaked in water for 24 hours prior to testing. The cylinders made for the stress relaxation tests were sealed along their lateral surface with aluminum foil, prior to loading, to prevent moisture loss during the tests. Furthermore, the top end of each specimen received a very thin layer of heavy-duty epoxy to account for the surface imperfections.

The stress relaxation experimental setup consists of multiple steel frames equipped with load cells at the base that are connected to a single sensor/modular device that collects the data from all load cells and transfers it to a computer, in which the data is converted to load units and stored. The conversion of the signals to load units occurs through the LabView software, following a script written on the background of the program interface. Each specimen is placed between two steel plates located in the body of the frame. These plates have free vertical movement, while being constrained from horizontal motion. The load is slowly applied by manually tightening a small screw positioned at the top of each frame. The element containing the screw is constrained by two leveled bolts. The rotation of the screw results in a vertical pressure that forces the upper plate to move downwards, applying a displacement to the specimen.

Upon testing, the applied force is monitored in real time at the program interface, and slowly increased until reaching the desired stress level. As mentioned previously, the caliper was used to measure the vertical displacement between the steel plates located at the ends of the specimen. In order to account for any free deformation of the samples as a function of time, the deformation of untested companion specimens, identical to the relaxation specimens, was measured. The free strains were subtracted from the strains measured in the relaxation tests in order to isolate mechanically induced strains.

7.3 Mortar Simulation

The periodic, cubic mortar microstructures are sliced into a stack of images in the Z-direction, where each image represents a 6.6 micrometer thickness. These images are stacked meshed using OOF3D in the same manner as the concrete microstructures using the process described in Chapter 5. In consistency with the periodic concrete simulations, periodic boundary condition constraint equations are applied to the mortar microstructures. A constant, uniaxial compressive strain of 0.02% is applied to the microstructures for 14 simulated days. The sand grains are considered to be linear elastic and are assigned a Young's modulus value of 30 GPa. The cement paste is assigned a Young's modulus of 13.02 GPa and viscoelastic material properties, described in the next section.

7.3.1 Description of Viscoelastic Cement Paste

The cement paste stress data and strain data from Tavares and Grasley (2020) is processed in the same manner that the stress relaxation simulation data is, as described in Chapter 5. Two cement pastes are assessed in this work, one with a w/c ratio of 0.4 loaded 3 days after mixing, and the other with a w/c ratio of 0.5 loaded 28 days after mixing.

Stress relaxation tests are performed by subjecting a sample to a constant strain, maintaining a perfectly constant strain is difficult to do in laboratory conditions, so to account for changes in the applied strain, the strain history is accounted for through $\epsilon(t)$, which is the strain as a function of time, t. Measured relaxation data over time is represented by $\sigma(t)$. The constitutive relationship between the strain history, $\epsilon(t)$, the stress, $\sigma(t)$, and relaxation modulus, $E(t)$ is

$$\sigma(t) = \int_0^t E(t-t')\,\frac{\delta\epsilon(t')}{\delta t'}dt, \tag{7.1}$$

where t is time in d and t' is the dummy time variable. $E(t)$ is assumed to be of the form

$$E(t) = E_0 + E_1 \cdot e^{\frac{-t}{\tau_1}} + E_2 \cdot e^{\frac{-t}{\tau_2}} + E_3 \cdot e^{\frac{-t}{\tau_3}} + E_4 \cdot e^{\frac{-t}{\tau_4}} + E_5 \cdot e^{\frac{-t}{\tau_5}} + E_6 \cdot e^{\frac{-t}{\tau_6}} \text{ GPa}, \tag{7.2}$$

where E_0, E_1, E_2, E_3, E_4, E_5, and E_6 are coefficients with units GPa. The relaxation times τ_0, τ_1, τ_2, τ_3, τ_4, τ_5 and τ_6 are in units of days. To determine the coefficients in the $E(t)$ equation, Equation (7.1) is integrated and fit to the $\sigma(t)$ data. The values of the fitted coefficients are then plugged into Equation (7.2) to obtain $E(t)$. The E(t) function calculated for the cement paste with a w/c ratio of 0.4 loaded at 3 days is

$$\begin{aligned} E(t) = 0.898 + 2.910 \cdot e^{\frac{-t}{0.05}} + 7.269 \cdot e^{\frac{-t}{0.25}} + 5.262 \cdot e^{\frac{-t}{1.25}} - 7.585 \cdot e^{\frac{-t}{2.5}} \\ +10.375 \cdot e^{\frac{-t}{5}} - 5.472 \cdot e^{\frac{-t}{10}} \text{ GPa}, \end{aligned} \tag{7.3}$$

where the τ_0, τ_1, τ_2, τ_3, τ_4, τ_5 and τ_6 values are chosen to be 0.05, 0.25, 1.25, 2.5, 5, and 10 respectively. The E(t) function calculated for the cement paste with a w/c ratio of 0.5 loaded at 28 days is

$$\begin{aligned} E(t) = 1.01 + 2.568 \cdot e^{\frac{-t}{0.05}} - 0.208 \cdot e^{\frac{-t}{0.25}} + 12.306 \cdot e^{\frac{-t}{1.25}} - 6.109 \cdot e^{\frac{-t}{2.5}} \\ +8.069 \cdot e^{\frac{-t}{5}} - 4.037 \cdot e^{\frac{-t}{10}} \text{ GPa}, \end{aligned} \tag{7.4}$$

where the $\tau_0, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5$ and τ_6 values are also chosen to be 0.05, 0.25, 1.25, 2.5, 5, and 10 respectively. Both Equations (7.3) and (7.4) are the average $E(t)$ from three experiments, respectively.

7.3.2 Mesh Density Convergence Study

Following the same procedures as the mesh density convergence study in Chapter 5, a single microstructure was meshed mutliple times at varying mesh densities. Each finite element mesh was subjected to identical stress relaxation simulations. Two time points, t = 0 days and t = 14 days, are used to assess the resulting relaxation modulus, where the relaxation modulus at t = 0 is equivalent to the Young's modulus. Given that the mortar microstructures are very similar, only set apart by their volume fraction, only one mesh density convergence study was conducted. The 50% volume fraction microstructure were selected for the study because they contain a higher number of aggregates and, therefore, a suitable mesh density for the higher volume fraction set will be suitable for the 47% volume fraction set as well. Figure 7.2 shows the corresponding data for the mortar simulated with a 50% volume fraction and a w/c ratio of 0.4. In 7.2, the convergence is most pronounced at t = 14 days, matching the findings in Chapters 5 and 6 that the mesh density becomes more impactful on the results at later time steps. Clear convergence at a value around 0.000075 elements/micrometer^3 can be seen in Figure 7.2.

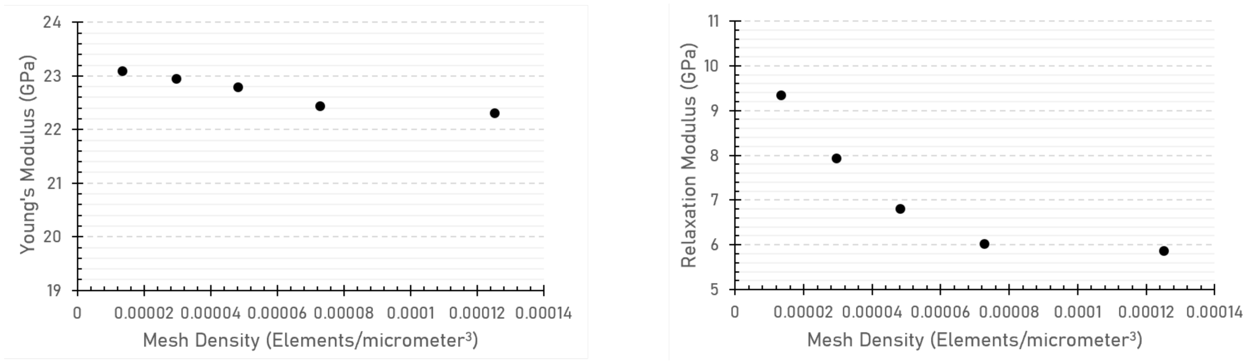


Figure 7.2: The Young's modulus a) and relaxation modulus at 14 days b) of a three-dimensional cubic mortar microstructure with an edge length of 2 mm plotted against the density of finite elements to determine convergence at a 50% fine aggregate volume fraction with a w/c ratio of 0.4

7.3.3 Results

Figure 7.3 shows the relaxation modulus E(t) of mortar with a w/c of 0.50 loaded age 28 days. The Poisson's ratio for cement paste in this simulation was assumed to be constant and equal to 0.235 based on the results from a study by Li et al [21]. Although that study showed that the Poisson's ratio is not constant during relaxation tests, the change with time is typically minor.

The instantaneous Young's modulus obtained for this mortar following ASTM E111-17 [135] and testing three replicate samples is 28.8 $\pm$ 0.53 GPa, while simulation results showed an instantaneous Young's modulus of 24.7 GPa.

Figure 7.4 shows the relaxation modulus of mortar with a w/c of 0.40 loaded 3 days after mixing. The Young's modulus obtained for this mortar is 26.6 $\pm$ 0.45 GPa and 22.9 GPa for the experimental and simulated data, respectively. Similar to the first simulation, the assumed Poisson's ratio is also equal to a constant value of 0.235.

As seen in Figures 7.4 and 7.3, the simulation results show the same relaxation trends, however

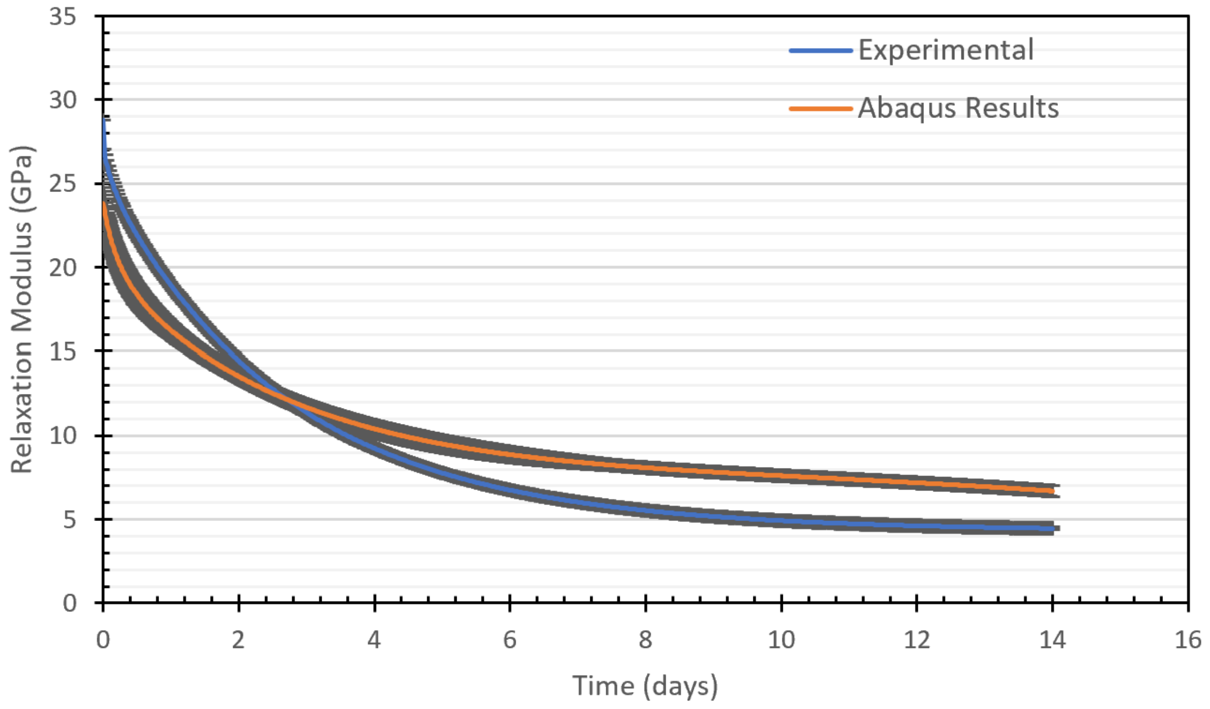


Figure 7.3: Experimental and simulated mortar stress relaxation data, loaded 28 days after mixing with a w/c of 0.50. The gray band around each curve represnts ± the standard error.

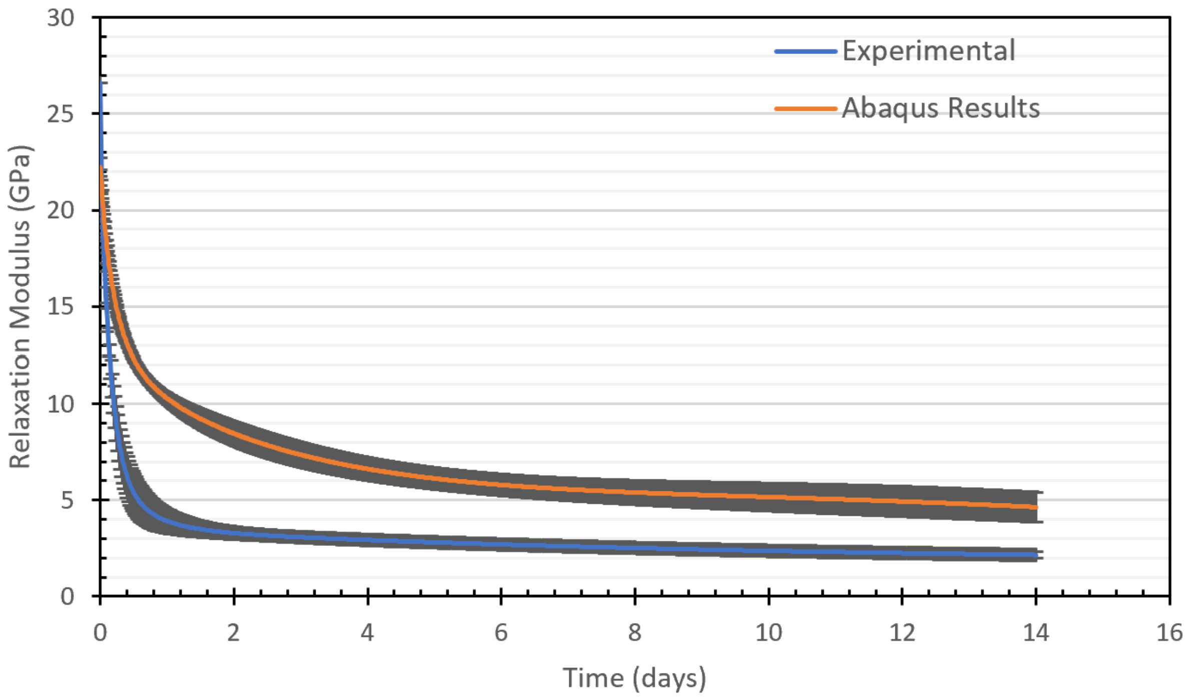


Figure 7.4: Experimental and simulated mortar stress relaxation data, loaded 3 days after mixing with a w/c of 0.40. The gray band around each curve represnts ± the standard error.

the agreement is not outstanding. This is hypothesized to be the result of the following potential factors:

- Influence of ITZ on creep
- Aging behavior of cement paste at early ages
- Difficulty obtaining detailed strain history data in cement paste experiments

A publication by Lavergne et. al (2015) indicated that the ITZ impacts the creep of mortars, providing one potential explanation for the differences between the experimental and simulation results. As the ITZ is a porous region with a lower stiffness that the surrounding cement and aggregates, it is theorized that the ITZ has a lower resistance to creep and stress relaxation, thus increasing the level of creep/relaxation in mortars and concrete. The microstructures in this dissertation doesn't include an ITZ region. In Chapter 5, it was shown that neglecting the ITZ in these microstructures was a valid assumption, as the simulation and experimental concrete creep data showed excellent agreement. However, the important difference between the work presented in Chapter 5 and this chapter is that the mortar experimental data used to describe the viscoelastic behavior of the matrix included the impact of the ITZ region surrounding the sand grains in the mortar. In the mortar models presented in this chapter, the matrix is described by cement paste experimental data that does not have any aggregates, and thus doesn't include ITZ effects.

The viscoelastic behavior of the cement paste has been described by linear viscoelasticity in this work, however since linear viscoelasticity does not include the aging behavior of early-age cement paste and mortar, the FEA model does not include the aging behavior either. Differences in the experimental and simulated data are believed to be the result of neglecting the aging component in the viscoelastic model. This idea is supported by the closer agreement between the experimental and simulation data for the mortar loaded at age 28 days compared to the results for the mortar loaded at age 3 days. The cement paste sample loaded at age 3 days is not as far along in the hydration process as the 28 days sample, making it more susceptible to the influence of aging.

Furthermore, Poisson's ratio of the cement paste is assumed to be constant in the FEA simulations, however literature has shown this assumption is not always valid [21]. Finally, the accuracy of the strain history can also influence the shape and magnitude of the simulated E(t), as shown by C. Tavares et. al [22]. Cement paste relaxation data has only been collected for 14 days, limiting the length of time that of the simulations. Collection of longer-term cement paste creep data was planned; however, this effort was interrupted due to COVID-19 and data beyond 14 days are not available at this time.

The third and final hypothesis for the differences between the simulation and experimental relaxation data is the measurement of the strain in the cement paste experiments. In a stress relaxation experiment, the ideal goal is to maintain a perfectly constant strain, however this is difficult to obtain in real life. Thus, the strain history is accounted for by considering the strain to be a function of time. Strain measurements were taken by hand with a caliper, and due to the strains being small, error in measurement could have a notable impact. These strains were fit to an equation that was then used to calculate the relaxation modulus, $E(t)$, therefore error in the measurements is propagated into the simulation by way of the $E(t)$ equation.

7.4 Summary

The research presented in this chapter investigates a method of upscaling cement paste creep data to mortar creep behavior in an effort to further upscale these simulations to the concrete level in future works, alleviating the challenges involving concrete creep measurements. Mortar simulations were shown in this paper with reasonable agreement to experimental data. At the time of this dissertation, suitable concrete data was unavailable for comparison, therefore the mortar results were not upscaled to concrete as planned. The following conclusions can be drawn from the work presented in this chapter:

- Convergence of the finite element mesh density was determined to be approximately 0.000075 elements/micrometer^3 for the 3D mortar microstructures Simulations performed with lower mesh densities showed up to 35 % difference from the converged results, showing the importance of using a proper finite element mesh.

- The simulation of 14 days of mortar relaxation was performed with good agreement using two different cement pastes - one with a w/c ratio of 0.4 loaded at 3 days, and the second with a w/c ratio of 0.5 loaded at 28 days.
- Small differences between the simulation and experimental mortar relaxation data are thought to be due to the lack of ITZ. In Chapter 5, the mortar experimental data included any influence that the ITZ region around the sand particles had on the creep behavior, whereas the cement paste samples don't contain any aggregates and thus, no ITZ regions.
- The upscaling approach presented here and in Chapter 5 shows the ability to upscale both cement paste and mortar experimental data

8. SUMMARY AND CONCLUSIONS

8.1 Dissertation Summary

In this dissertation, a microstructure generation code was developed and presented. Real aggregate shapes were reconstructed from XCT data analyzed with spherical harmonics. This code has the ability to model a variety of boundary conditions in both 2D and 3D, as well as generate cubic and cylindrical microstructures. A wide range of microstructures have been generated using these codes, the details of which can be found in the Appendices. Great attention was paid to improving the efficiency of the microstructure generation. Notable approaches include the pre-rotation of aggregates and the asperity based overlap detection algorithm.

Asperities on an aggregate's surface were identified through the first and second derivatives of the radius equation in spherical coordinates. These locations were stored and used in a novel overlap detection algorithm that compares the distance from an aggregate's center to an asperity on another aggregate with the radius of the aggregate in the same direction. Substantial time savings of over 85% for realist volume fractions resulted from this algorithm.

The virtual concrete samples were simulated and compared to experimental data that has been extended using the TTS principle to create a long-term master creep compliance curve. Simulation results using a periodic cubic microstructure showed excellent agreeement with the master curve. On the other hand, it was shown that 2D periodic simulations greatly overpredict the creep of concrete by 50% after 27 years of creep. Mortar microstructures were also assessed using two cement pastes, one loaded at 3 days of age and the other loaded at 28 days old. The mortar simulations showed good agreement with the general trend of the experimetnal stress relaxation data, however the simulations underpredict the magnitude of relaxation, likely due to the lack of an ItZ phase in the model.

8.2 Primary Conclusions

Simulation tools can be very useful in the world of concrete, and composites in general, to obtain information on viscoelastic behavior. As lab tests on materials with long-term viscoelasticity, such as concrete, are very slow to generate data, the methods presented in this paper offer a good alternative to experiments by demonstrating the process to generate morphologically representative microstructures and using FEM to predict long-term viscoelastic behavior.

The following conclusions can be drawn from this dissertation work:

- This work utilizes an efficient microstructure generation code that can be adapted to follow different aggregate sieve gradations, utilize different aggregate shapes, etc. to allow for the creation of a highly realistic microstructure.
- The improvements in efficiency resulting from the use of the novel asperity based overlap detection algorithm were above 85% for microstructures containing greater than 30% volume fraction coarse aggregates.
- The asperity check detected between an average of 85% and 90 % of all aggregate-aggregate overlap in 40% volume fraction microstructures of each of the four gradations tested
- Periodic boundary conditions were successfully applied both the aggregate geometry and mathematically. The geometrical periodicity was exact, while the stress and strain along opposite edge elements on 2D and 3D microstructures were compared and found to have good, but not perfect, agreement, indicating approximate periodicity. Small differences in the stresses and strains resulted from the differences in size and shape of the edge elements between the opposite edges.
- Convergence of the RVE size was found to be at approximately 60 mm for 2D and 90 mm 3D periodic microstructures, corresponding to an edge length approximately 2x and 3x longer than the largest aggregate's length for the 2D and 3D microstructures, respectively.

- Both the convergence of the RVE size and the finite element mesh density show a time-dependence of the value of convergence, indicating the need for convergence studies of time dependent properties to assess multiple time points. Additionally, the convergence of the Young's modulus does not represent the convergence of the creep/relaxation behavior, further reinforcing the need to assess the convergence over time.

- Both 2D and 3D simulations accurately predict Young's modulus within 7 % and 3 %, respectively. However, the 2D simulations greatly overpredict creep, with the overprediction of the creep compliance increasing over time to about 50 % after 10,000 days. Therefore, 2D models may be sufficient to predict Young's modulus in some applications, they are not recommended for use in modeling viscoelasticity.

- Two types of cement paste, one loaded at an early age and one loaded at maturity, were upscaled to mortar using the same framework but adapted to simulate a mortar microstructure.

- The treatment of coarse aggregates as linear elastic and the decision to neglect the ITZ regions around the coarse aggregates in concrete microstructures appears to be valid simplifications given the close agreement between the 3D simulation results and experimental data. Any significant effect of the ITZ regions around sand grains were not studied but were swept into the mortar matrix creep data.

- On the other hand, the lack of an ITZ region in the mortar microstructures is believed to be the primary cause of underpredicion of stress relaxation in the simulation of mortars.

- Due to the time-dependent nature of concrete creep, it is challenging to collect creep behavior data in a timely fashion. This dissertation work has successfully upscaled experimental mortar creep data to predict concrete creep behavior.

8.3 Implications & Applications

Presented in this dissertation is an efficient microstructure generation code capable of producing both 2D and 3D microstructures with a variety of boundary conditions and sample geometries. This code can adapt to variety of aggregate gradations, aggregate shapes, and volume fractions. From this, future researchers can generate virtual microstructures that closely resemble any concrete in question. These codes have attracted interest and collaboration with national laboratories that are researching concrete mechanical behavior and fracture, where a highly realistic virtual morphology is useful.

This work has developed a complete framework to predict long-term concrete creep behavior by using virtual concrete microstructures and FEA simulations to upscale experimental mortar creep data. From the start of microstructure generation to the processing of the FEA simulation results, decades of data can now be generated in 1-2 days. This opens the door to much future work and assessment of the influence of a variety of parameters on the mechanical behavior of concrete and mortar, as data can be gathered rapidly and simultaneously.

Applications of this work are numerous and varied. The initial target application was assessment of aging nuclear power plants' concrete containment structures and the assessment of the ongoing viability of those structures. As it is not possible to take a concrete core from a nuclear power plant, this research has demonstrated a method of assessing the level of concrete creep. From this, the concrete creep behavior at nearly any NPP can be evaluated if the concrete mix design is known. This data can be used as input into larger-scale simulation codes or models.

Furthermore, the homogenized concrete creep constitutive relationship was used in a collaboration with Idaho National Lab to describe the concrete viscoelastic behavior in a large-scale simulation in the FEA software Grizzly. This collaboration produced simulations of 40 years of concrete creep in large, reinforced concrete walls - providing a method of using the data collected in this research to make predictions on the structural scale. Use of structural simulation codes presents one method of upscaling the data generated in the framework presented here, as it can be used to describe the material behavior of the concrete in situations with reinforcement, prestress or post-tensioning, and other structural design scenarios.

8.4 Future Work & Next Steps

Many avenues of study may branch off from the research presented in this dissertation. The microstructure generation code paired with the asperity-based overlap detection method opens the door to much research into microstructural factors and their relationship to mechanical behavior. Additionally, with the successful upscaling of both cement paste and mortar experimental data, a number of possibilities for future areas of research have been brainstormed. The following questions, among many others, can be posed:

- What is the impact of aggregate gradation on creep?
- What is the impact of aggregate shape on creep?
- How does the morphological boundary conditions of a cylinder microstructure impact creep behavior? Is it the same relationship as cubic microstructures?
- How does the ratio of diameter to height of cylinders impact simulated Young's modulus and creep?
- How can an ITZ be incorporated into the microstructures and FEA models to better predict creep of mortar?
- Can cement paste data be upscaled all the way to concrete?
- Can this framework incorporate steel reinforcement on a small scale?

REFERENCES

[1] “Will Nuclear Power Plants in the United States Experience Life beyond 60?.” https://www.scottmadden.com/insight/will-nuclear-power-plants-in-the-united-states-experience-life-beyond-60/.

[2] A. Baranikmuar, C. E. Torrence, and Z. Grasley, “Using Time-Temperature Superposition to Preduct Long-Term Creep of Nuclear Concrete,” in *Transactions of 25th Conference on Structural Mechanics in Reactor Technology*, 2019.

[3] X. Yang, Z. You, Z. Wang, and Q. Dai, “Review on heterogeneous model reconstruction of stone-based composites in numerical simulation,” *Construction and Building Materials*, vol. 117, pp. 229–243, 2016.

[4] F. Bernachy-Barbe and B. Bary, “Effect of aggregate shapes on local fields in 3D mesoscale simulations of the concrete creep behavior,” *Finite Elements in Analysis and Design*, vol. 156, no. September 2018, pp. 13–23, 2019.

[5] N. Chawla, R. S. Sidhu, and V. V. Ganesh, “Three-dimensional visualization and microstructure-based modeling of deformation in particle-reinforced composites,” *Acta Materialia*, 2006.

[6] Y. Fu, L. Wang, and C. Zhou, “3D clustering DEM simulation and non-invasive experimental verification of shear localisation in irregular particle assemblies,” *International Journal of Pavement Engineering*, vol. 11, pp. 355–365, Oct. 2010.

[7] E. J. Garboczi, “Three-dimensional mathematical analysis of particle shape using X-ray tomography and spherical harmonics: Application to aggregates used in concrete,” *Cement and Concrete Research*, vol. 32, pp. 1621–1638, Oct. 2002.

[8] Y. Huang, D. Yan, Z. Yang, and G. Liu, “2D and 3D homogenization and fracture analysis of concrete based on in-situ X-ray Computed Tomography images and Monte Carlo simulations,” *Engineering Fracture Mechanics*, vol. 163, pp. 37–54, Sept. 2016.

[9] W. P. Y. Smith, *Relating concrete cube, core and cylinder compressive strengths that are cast, cured, prepared and tested in laboratory conditions*. PhD thesis, University of Cape Town, 2017.

[10] A. M. Neville, *Properties of Concrete*. New York: Pearson, 5 ed., 2015.

[11] “U.S. Nuclear Regulatory Commission – NRC Datasets.” https://www.nrc.gov/reading-rm/doc-collections/datasets/, 2019.

[12] “Duke Energy shuts down Crystal River nuclear plant permanently,” *Central Florida 13 News*, Feb. 2013.

[13] “Crystal River nuclear plant to be retired; company evaluating sites for potential new gas-fueled generation,” Feb. 2013.

[14] S. Mindess, F. J. Young, and D. Darwin, *Concrete 2nd Edition*. 2003.

[15] L. G. Tulin, *Creep of a Portland cement mortar as a function of stress-level and time*. PhD thesis, Iowa State University, Digital Repository, Ames, Jan. 1965.

[16] I. J. Jordaan, “A note on concrete creep analysis under static temperature fields,” *Matériaux et Constructions*, vol. 7, pp. 329–333, Sept. 1974.

[17] Z. P. Bazant and L. Panula, “Practical prediction of time-dependent deformations of concrete,” tech. rep., 1978.

[18] S. Scheiner, C. Hellmich, and A. M. Asce, “Continuum Microviscoelasticity Model for Aging Basic Creep of Early-Age Concrete,” *Journal of Engineering Mechanics*, 2009.

[19] Y. J. Huang, Z. J. Yang, X. W. Chen, and G. H. Liu, “Monte Carlo simulations of meso-scale dynamic compressive behavior of concrete based on X-ray computed tomography images,” *International Journal of Impact Engineering*, vol. 97, pp. 102–115, Nov. 2016.

[20] L.-O. Nilsson, “Present limitations of models for predicting chloride ingress into reinforced concrete structures,” *Journal de Physique IV (Proceedings)*, vol. 136, pp. 123–130, Nov. 2006.

[21] A. Baranikmuar, C. E. Torrence, and Z. Grasley, “3D Creep Response of Concrete,” in *Proceedings of The 19th International Conference on Environmental Degradation of Materials in Nuclear Power Systems- Water Reactors*, 2019.

[22] C. Argento and D. Bouvard, “Modeling the effective thermal conductivity of random packing of spheres through densification,” *International Journal of Heat and Mass Transfer*, vol. 39, pp. 1343–1350, May 1996.

[23] A. Jagota and C. Y. Hui, “The effective thermal conductivity of a packing of spheres,” *Journal of Applied Mechanics, Transactions ASME*, vol. 57, no. 3, pp. 789–791, 1990.

[24] L. Y. Yi, K. J. Dong, R. P. Zou, and A. B. Yu, “Coordination number of the packing of ternary mixtures of spheres: Dem simulations versus measurements,” *Industrial and Engineering Chemistry Research*, vol. 50, pp. 8773–8785, July 2011.

[25] L. Y. Yi, K. J. Dong, R. P. Zou, and A. B. Yu, “Radical tessellation of the packing of spheres with a log-normal size distribution,” *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 92, Sept. 2015.

[26] K. Han, Y. T. Feng, and D. R. Owen, “Sphere packing with a geometric based compression algorithm,” *Powder Technology*, vol. 155, pp. 33–41, July 2005.

[27] A. Z. Zinchenko, "Algorithm for Random Close Packing of Spheres with Periodic Boundary Conditions," tech. rep., 1994.

[28] Z. Y. Zhou, R. P. Zou, D. Pinson, and A. B. Yu, "Dynamic simulation of the packing of ellipsoidal particles," *Industrial and Engineering Chemistry Research*, vol. 50, pp. 9787–9798, Aug. 2011.

[29] W. Man, A. Donev, F. H. Stillinger, M. T. Sullivan, W. B. Russel, D. Heeger, S. Inati, S. Torquato, and P. M. Chaikin, "Experiments on random packings of ellipsoids," *Physical Review Letters*, vol. 94, p. 198001, May 2005.

[30] X. Lin and T.-T. Ng, "Contact detection algorithms for three-dimensional ellipsoids in discrete element modelling," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 19, pp. 653–659, Sept. 1995.

[31] X. Lin and T. T. Ng, "A three-dimensional discrete element model using arrays of ellipsoids," *Geotechnique*, vol. 47, no. 2, pp. 319–329, 1997.

[32] F. Dorai, C. Moura Teixeira, M. Rolland, E. Climent, M. Marcoux, and A. Wachs, "Fully resolved simulations of the flow through a packed bed of cylinders: Effect of size distribution," *Chemical Engineering Science*, vol. 129, pp. 180–192, June 2015.

[33] W. Zhang, K. E. Thompson, A. H. Reed, and L. Beenken, "Relationship between packing structure and porosity in fixed beds of equilateral cylindrical particles," *Chemical Engineering Science*, vol. 61, pp. 8060–8074, Dec. 2006.

[34] A. B. Y. R. P. Zou, "Wall effect on the packing of cylindrical particles," 1996.

[35] H.-K. Man and J. G. M van Mier, "Size effect on strength and fracture energy for numerical concrete with realistic aggregate shapes," vol. 154, pp. 61–72, 2008.

[36] E. Piotrowska, Y. Malecot, and Y. Ke, “Experimental investigation of the effect of coarse aggregate shape and composition on concrete triaxial behavior,” *Mechanics of Materials*, 2014.

[37] G. Giaccio and R. Zerbino, “Failure mechanism of concrete: combined effects of coarse aggregates and strength level,” *Advanced Cement Based Materials*, 1998.

[38] A. C. Aydin, A. Arslan, and R. Gül, “Mesoscale simulation of cement based materials' time-dependent behavior,” *Computational Materials Science*, 2007.

[39] J. J. Zheng, X. Z. Zhou, Y. F. Wu, and X. Y. Jin, “A numerical method for the chloride diffusivity in concrete with aggregate shape effect,” *Construction and Building Materials*, 2012.

[40] S. Dehghanpoor Abyaneh, H. S. Wong, and N. R. Buenfeld, “Modelling the diffusivity of mortar and concrete using a three-dimensional mesostructure with several aggregate shapes,” *Computational Materials Science*, 2013.

[41] J. J. Brooks, “30-year creep and shrinkage of concrete,” *Magazine of Concrete Research*, vol. 57, pp. 545–556, Nov. 2005.

[42] Z. Qian, E. J. Garboczi, G. Ye, and E. Schlangen, “Anm: a geometrical model for the composite structure of mortar and concrete using real-shape particles,” *Materials and Structures/Materiaux et Constructions*, vol. 49, pp. 149–158, Jan. 2016.

[43] S. Thomas, Y. Lu, and E. J. Garboczi, “Improved Model for Three-Dimensional Virtual Concrete: Anm Model,” *Journal of Computing in Civil Engineering*, vol. 30, no. 2, p. 4015027, 2015.

[44] “OOF: Finite Element Analysis of Microstructures.” https://www.ctcms.nist.gov/oof/oof3d/index.html.

[45] A. Baranikumar, C. E. Torrence, and Z. Grasley, “Thermo-Rheological Approach to Predict Long-Term Creep of Cement Mortar from Short-Term Tests,” *Mechanics of Time-Dependent Materials*.

[46] C. Tavares and Z. Grasley, “Effect of Strain History and Mixture Proportions on Early Age Cement Paste Mortar and Stress Relaxation,” *Journal of Materials in Civil Engineering*.

[47] J. W. Bullard and E. J. Garboczi, “A model investigation of the influence of particle shape on portland cement hydration,” *Cement and Concrete Research*, vol. 36, no. 6, pp. 1007–1015, 2006.

[48] D. P. Bentz, E. J. Garboczi, C. J. Haecker, and O. M. Jensen, “Effects of cement particle size distribution on performance properties of Portland cement-based materials,” *Cement and Concrete Research*, vol. 29, pp. 1663–1671, Oct. 1999.

[49] L. W. Rong, Z. Y. Zhou, and A. B. Yu, “Lattice-Boltzmann simulation of fluid flow through packed beds of uniform ellipsoids,” *Powder Technology*, vol. 285, pp. 146–156, Nov. 2015.

[50] B. Osbaeck and V. Johansen, “Particle Size Distribution and Rate of Strength Development of Portland Cement,” *Journal of the American Ceramic Society*, vol. 72, pp. 197–201, Feb. 1989.

[51] J. Gan, Z. Zhou, and A. Yu, “Effect of particle shape and size on effective thermal conductivity of packed beds,” *Powder Technology*, vol. 311, pp. 157–166, Apr. 2017.

[52] F. Y. Fraige, P. A. Langston, and G. Z. Chen, “Distinct element modelling of cubic particle packing and flow,” *Powder Technology*, vol. 186, pp. 224–240, Sept. 2008.

[53] M. J. Vold, “THE SEDIMENT VOLUME IN DILUTE DISPERSIONS OF SPHERICAL PARTICLES,” *The Journal of Physical Chemistry*, vol. 64, no. 11, pp. 1616–1619, 1960.

[54] M. D. Webb and I. L. Davis, “Random particle packing with large particle size variations using reduced-dimension algorithms,” *Powder Technology*, 2006.

[55] T. Atmakidis and E. Y. Kenig, “CFD-based analysis of the wall effect on the pressure drop in packed beds with moderate tube/particle diameter ratios in the laminar flow regime,” *Chemical Engineering Journal*, 2009.

[56] C. E. Torrence, A. Baranikumar, Z. Grasley, and E. J. Garboczi, “Homogenization of Concrete Microstructures in Nuclear Power Plants,” tech. rep.

[57] Z. Qian, E. J. Garboczi, G. Ye, and E. Schlangen, “Anm: a geometrical model for the composite structure of mortar and concrete using real-shape particles,” *Materials and Structures/Materiaux et Constructions*, vol. 49, no. 1-2, pp. 149–158, 2014.

[58] M. Salemi and H. Wang, “Image-aided random aggregate packing for computational modeling of asphalt concrete microstructure,” *Construction and Building Materials*, vol. 177, pp. 467–476, 2018.

[59] Y. Wang and J. G. Zhu, “Simulation of realistic particle packing and impact on high-density tape recording,” in *IEEE Transactions on Magnetics*, vol. 45, pp. 3737–3740, Oct. 2009.

[60] H. Zhang, P. Sheng, J. Zhang, and Z. Ji, “Realistic 3D modeling of concrete composites with randomly distributed aggregates by using aggregate expansion method,” *Construction and Building Materials*, vol. 225, pp. 927–940, Nov. 2019.

[61] N. Chernov, Y. Stoyan, and T. Romanova, “Mathematical model and efficient algorithms for object packing problem,” *Computational Geometry: Theory and Applications*, vol. 43, pp. 535–553, July 2010.

[62] C. Recarey, I. Pérez, R. Roselló, M. Muniz, E. Hernández, R. Giraldo, and E. Oñate, “Advances in particle packing algorithms for generating the medium in the Discrete Element

Method," *Computer Methods in Applied Mechanics and Engineering*, 2019.

[63] X. Y. Li, S. H. Teng, and A. Üngör, "Biting: Advancing front meets sphere packing," *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 61–81, Sept. 2000.

[64] Y. T. Feng, K. Han, and D. R. J. Owen, "Filling domains with disks: an advancing front approach," *International Journal for Numerical Methods in Engineering*, vol. 56, pp. 699–713, Feb. 2003.

[65] I. P. Morales, M. M. de Farias, R. R. Valera, C. R. Morfa, and H. E. Martínez Carvajal, "Contributions to the generalization of advancing front particle packing algorithms," *International Journal for Numerical Methods in Engineering*, vol. 107, pp. 993–1008, Sept. 2016.

[66] P. A. Cundall and O. D. L. Strack, "A discrete numerical model for granular assemblies," vol. 29, pp. 47–65, Mar. 1979.

[67] M. Lu and ·. R. Mcdowell, "The importance of modelling ballast particle shape in the discrete element method," vol. 9, pp. 69–80, 2007.

[68] S. Abedi and A. A. Mirghasemi, "Particle shape consideration in numerical simulation of assemblies of irregularly shaped particles," *Particuology*, vol. 9, pp. 387–397, Aug. 2011.

[69] K. Kildashti, K. Dong, and B. Samali, "An accurate geometric contact force model for super-quadric particles," *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112774, Mar. 2020.

[70] M. Kodam, R. Bharadwaj, J. Curtis, B. Hancock, and C. Wassgren, "Cylindrical object contact detection for use in discrete element method simulations. Part I - Contact detection algorithms," *Chemical Engineering Science*, vol. 65, pp. 5852–5862, Nov. 2010.

[71] J. S. Ketchel and P. M. Larochelle, “Collision detection of cylindrical rigid bodies using line geometry,” tech. rep.

[72] J. M. Ting, “A robust algorithm for ellipse-based discrete element modelling of granular materials,” *Computers and Geotechnics*, vol. 13, pp. 175–186, Jan. 1992.

[73] A. Džiugys and B. Peters, “An approach to simulate the motion of spherical and non-spherical fuel particles in combustion chambers,” *Granular Matter*, vol. 3, pp. 231–265, Dec. 2001.

[74] X. Jia, Y. K. Choi, B. Mourrain, and W. Wang, “An algebraic approach to continuous collision detection for ellipsoids,” *Computer Aided Geometric Design*, vol. 28, pp. 164–176, Mar. 2011.

[75] J. W. Perram and M. S. Wertheim, “Statistical mechanics of hard ellipsoids. I. Overlap algorithm and the contact function,” *Journal of Computational Physics*, vol. 58, pp. 409–416, May 1985.

[76] J. M. Ting, M. Khwaja, L. R. Meachum, and J. D. Rowell, “An ellipse-based discrete element model for granular materials,” *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 17, pp. 603–623, Sept. 1993.

[77] Y. Song, R. Turton, and F. Kayihan, “Contact detection algorithms for DEM simulations of tablet-shaped particles,” *Powder Technology*, vol. 161, pp. 32–40, Jan. 2006.

[78] M. Kodam, J. Curtis, B. Hancock, and C. Wassgren, “Discrete element method modeling of bi-convex pharmaceutical tablets: Contact detection algorithms and validation,” *Chemical Engineering Science*, vol. 69, pp. 587–601, Feb. 2012.

[79] Y. Zhou, H. Jin, and B. Wang, “Modeling and mechanical influence of meso-scale concrete considering actual aggregate shapes,” *Construction and Building Materials*, 2019.

[80] K. Dong, C. Wang, and A. Yu, “Voronoi analysis of the packings of non-spherical particles,” *Chemical Engineering Science*, vol. 153, pp. 330–343, Oct. 2016.

[81] F. M. Schallerab, S. C. Kapferac, M. E. Evansa, M. J. Hoffmanna, T. Asted, M. Saadatfare, K. Meckea, G. W. Delaney, and G. E. Schröder-Turka, “Set Voronoi diagrams of 3D assemblies of aspherical particles,” *Philosophical Magazine*, vol. 93, no. 31-33, pp. 3993–4017, 2013.

[82] G. Mollon and J. Zhao, “3D generation of realistic granular samples based on random fields theory and Fourier shape descriptors,” *Computer Methods in Applied Mechanics and Engineering*, vol. 279, pp. 46–65, Sept. 2014.

[83] B. Zhou Ã, J. Wang, and H. Wang, “Three-dimensional sphericity, roundness and fractal dimension of sand particles,”

[84] A. Bertei, C. C. Chueh, J. G. Pharoah, and C. Nicolella, “Modified collective rearrangement sphere-assembly algorithm for random packings of nonspherical particles: Towards engineering applications,” *Powder Technology*, vol. 253, pp. 311–324, 2014.

[85] G. Liu and K. E. Thompson, “Influence of computational domain boundaries on internal structure in low-porosity sphere packings,” *Powder Technology*, vol. 113, no. 1, pp. 185–196, 2000.

[86] R. Zhou, Z. Song, and Y. Lu, “3D mesoscale finite element modelling of concrete,” *Computers and Structures*, vol. 192, pp. 96–113, Nov. 2017.

[87] T. Byholm, M. Toivakka, and J. Westerholm, “Effective packing of 3-dimensional voxel-based arbitrarily shaped particles,” *Powder Technology*, vol. 196, pp. 139–146, Dec. 2009.

[88] C. W. Boon, G. T. Houlsby, and S. Utili, “A new contact detection algorithm for three-dimensional non-spherical particles,” *Powder Technology*, vol. 248, pp. 94–102, Nov. 2013.

[89] E. J. Garboczi and J. W. Bullard, “Contact function, uniform-thickness shell volume, and convexity measure for 3D star-shaped random particles,” *Powder Technology*, vol. 237, pp. 191–201, 2013.

[90] T. Tong, *COMPUTATIONAL MODELLING OF CONCRETE TIME-DEPENDENT MECHANICS AND ITS APPLICATION TO LARGE-SCALE STRUCTURE ANALYSIS*. PhD thesis, University of Pittsburgh Swanson School of Engineering, Pittsburgh, 2016.

[91] M. Sharma and S. Bishnoi, “Influence of properties of interfacial transition zone on elastic modulus of concrete: Evidence from micromechanical modelling,” *Construction and Building Materials*, vol. 246, June 2020.

[92] H. Schorn and U. Rode, “Numerical simulation of crack propagation from microcracking to fracture,” *Cement and Concrete Composites*, vol. 13, no. 2, pp. 87–94, 1991.

[93] Y. Yin, Q. Ren, and L. Shen, “Study on the effect of aggregate distribution on mechanical properties and damage cracks of concrete based on multifractal theory,” *Construction and Building Materials*, vol. 262, Nov. 2020.

[94] M. Q. Thai, B. Bary, and Q. C. He, “A homogenization-enriched viscodamage model for cement-based material creep,” *Engineering Fracture Mechanics*, vol. 126, pp. 54–72, 2014.

[95] S. M. Kim and R. K. Abu Al-Rub, “Meso-scale computational modeling of the plastic-damage response of cementitious composites,” *Cement and Concrete Research*, vol. 41, pp. 339–358, Mar. 2011.

[96] G. Ye, K. Van Breugel, and A. L. Fraaij, “Three-dimensional microstructure analysis of numerically simulated cementitious materials,” in *Cement and Concrete Research*, vol. 33, pp. 215–222, Feb. 2003.

[97] G. Sherzer, P. Gao, E. Schlangen, G. Ye, and E. Gal, “Upscaling cement paste microstructure

to obtain the fracture, shear, and elastic concrete mechanical LDPM parameters," *Materials*, vol. 10, no. 3, 2017.

[98] L. Contrafatto, M. Cuomo, and S. Gazzo, "A concrete homogenisation technique at meso-scale level accounting for damaging behaviour of cement paste and aggregates," *Computers and Structures*, vol. 173, pp. 1–18, Sept. 2016.

[99] T. De Larrard, B. Bary, E. Adam, and F. Kloss, "Influence of aggregate shapes on drying and carbonation phenomena in 3D concrete numerical samples," *Computational Materials Science*, 2013.

[100] T. Honorio, B. Bary, and F. Benboudjema, "Multiscale estimation of ageing viscoelastic properties of cement-based materials: A combined analytical and numerical approach to estimate the behaviour at early age," *Cement and Concrete Research*, vol. 85, pp. 137–155, July 2016.

[101] P. Wriggers and S. O. Moftah, "Mesoscale models for concrete: Homogenisation and damage behaviour," *Finite Elements in Analysis and Design*, vol. 42, no. 7 SPEC. ISS., pp. 623–636, 2006.

[102] F. Xu, Z. Wu, J. Zheng, Y. Hu, and Q. Li, "Experimnetal study on the bond behavior of reinforcing bars embedded in concrete subjected to lateral pressure," *Journal of Materials in Civil Engineering*, vol. 24, no. 1, pp. 125–133, 2012.

[103] Q. Dai and Z. You, "Prediction of creep stiffness of asphalt mixture with micromechanical finite-element and discrete-element models," *Journal of Engineering Mechanics*, vol. 133, pp. 163–173, Feb. 2007.

[104] A. K. Kwan, Z. M. Wang, and H. C. Chan, "Mesoscopic study of concrete II: Nonlinear finite element analysis," *Computers and Structures*, vol. 70, no. 5, pp. 545–556, 1999.

[105] B. Bary, C. Bourcier, T. Helfer, B. Bary, C. Bourcier, and T. Helfer, "Thermoviscoelastic analysis of concrete creep at mesoscale," tech. rep.

[106] Q. X. Meng, D. Lv, and Y. Liu, "Mesoscale computational modeling of concrete-like particle-reinforced composites with non-convex aggregates," *Computers and Structures*, vol. 240, Nov. 2020.

[107] Z. Wu, J. Zhang, H. Yu, and H. Ma, "3D mesoscopic investigation of the specimen aspect-ratio effect on the compressive behavior of coral aggregate concrete," *Composites Part B: Engineering*, vol. 198, Oct. 2020.

[108] K. Hbaieb, Q. X. Wang, Y. H. J. Chia, and B. Cotterell, "Modelling stiffness of polymer/clay nanocomposites," *Polymer*, 2007.

[109] A. Yin, X. Yang, and Z. Yang, "2D and 3D fracture modeling of asphalt mixture with randomly distributed aggregates and embedded cohesive cracks," in *Procedia IUTAM*, vol. 6, pp. 114–122, Elsevier B.V., 2013.

[110] T. Liu, S. Qin, D. Zou, W. Song, and J. Teng, "Mesoscopic modeling method of concrete based on statistical analysis of CT images," *Construction and Building Materials*, vol. 192, pp. 429–441, Dec. 2018.

[111] H. Shen and L. C. Brinson, "Finite element modeling of porous titanium," *International Journal of Solids and Structures*, 2007.

[112] S. Meille and E. J. Garboczi, "Linear elastic properties of 2D and 3D models of porous materials made from elongated objects Related content Modelling drying shrinkage in reconstructed porous materials: application to porous Vycor glass Simulations of glass-estane mock PBXs Linear elas," *Modelling and Simulation in Materials Science and Engineering*, no. 9, pp. 371–390, 2001.

[113] M. Krishnapillai, R. Jones, I. H. Marshall, M. Bannister, and N. Rajic, "NDTE using pulse thermography: Numerical modeling of composite subsurface defects," *Composite Structures*, 2006.

[114] L. Boltzmann, "On the theory of the elastic aftereffect," *Poggendorff's Ann. Phys. Chem*, vol. 7, pp. 624–645, 1876.

[115] G. Pickett, "THE EFFECT OF CHANGE IN MOISTURE-CONTENT ON THE CREEP OF CONCRETE UNDER A SUSTAINED LOAD," *Portland Cement Assoc R & D Lab Bull*, no. 0, 1970.

[116] F. Lavergne, K. Sab, J. Sanahuja, M. Bornert, and C. Toulemonde, "Investigation of the effect of aggregates' morphology on concrete creep properties by numerical simulations," *Cement and Concrete Research*, vol. 71, pp. 14–28, 2015.

[117] Y. Wang, Q. Xu, and S. Chen, "Approaches of concrete creep using mesomechanics: Numerical simulation and predictive model," *Modelling and Simulation in Materials Science and Engineering*, vol. 27, no. 5, 2019.

[118] B. R. H. Campbell and R. E. Tobin, "Core and Cylinder Strengths of Natural and Lightweight Concrete," *ACI Journal Proceedings*, vol. 4, no. 64, pp. 190–195, 1967.

[119] A. Ergün and G. Kürklü, "x," Tech. Rep. 3, 2012.

[120] D. J. Elwell and G. Fu, "Compression Testing of Concrete: Cylinders versus Cubes," tech. rep., United States Department of Transportation, Mar. 1995.

[121] J. R. Del Viso, J. R. Carmona, and G. Ruiz, "Size and Shape Effects on the Compressive Strength of High Strength Concrete," in *International Association of Fracture Mechanics for Concrete and Concrete Structures*, 2007.

[122] M. K. Abd and Z. D. Habeeb, "EFFECT OF SPECIMEN SIZE AND SHAPE ON COMPRESSIVE STRENGTH OF SELF-COMPACTING CONCRETE," *Diyala Journal of Engineering Sciences*, vol. 7, no. 2, pp. 16–29, 2014.

[123] Y. Kusumawardaningsih, E. Fehling, and M. Ismail, "ScienceDirect UHPC compressive strength test specimens: Cylinder or cube?," *Procedia Engineering*, vol. 125, pp. 1076–1080, 2015.

[124] S. Reddy, S. Rao, and S. Shrihari, "Strength Conversion Factors for Concrete Based On Specimen Geometry, Aggregate Size and Direction of Loading," *International Journal of Recent Technology and Engineering*, no. 2, pp. 2277–3878, 2019.

[125] J. N. Pacheco, J. de Brito, C. Chastre, and L. Evangelista, "Probabilistic conversion of the compressive strength of cubes to cylinders of natural and recycled aggregate concrete specimens," *Materials*, vol. 12, Jan. 2019.

[126] ASTM, "ASTM D4791 - Standard Test Method for Flat Particles, Elongated Particles, or Flat and Elongated Particles in Coarse Aggregate," tech. rep., American Sociey for Testing and Materials, 2019.

[127] C. T. Sun and R. S. Vaidya, "Prediction of composite properties from a representative volume element," *Composites Science and Technology*, vol. 56, pp. 171–179, Jan. 1996.

[128] H. Wadell, "Volume, Shape, and Roundness of Rock Particles," *The Journal of Geology*, vol. 40, no. 5, pp. 443–451, 1932.

[129] F. W.B. and T. S.E., "THE LAWS OF PROPORTIONING CONCRETE," Jan. 1907.

[130] ASTM C150/C150M, "Standard Specification for Portland Cement," Tech. Rep. April, 2020.

[131] ASTM C192, "Standard Practice for Sampling Freshly Mixed Concrete," *ASTM International, West Conshohocken, PA*, vol. 04, pp. 8–10, 2007.

[132] Z. P. Bažant and A. Steffens, "Mathematical model for kinetics of alkali-silica reaction in concrete," *Cement and Concrete Research*, vol. 30, pp. 419–428, Mar. 2000.

[133] B. Bary, C. Bourcier, and T. Helfer, "Numerical Analysis of Concrete Creep on Mesoscopic 3D Specimens," *Concreep 10*, 2015.

[134] Y. J. Huang, Z. J. Yang, X. W. Chen, and G. H. Liu, "Monte Carlo simulations of meso-scale dynamic compressive behavior of concrete based on X-ray computed tomography images," *International Journal of Impact Engineering*, vol. 97, pp. 102–115, Nov. 2016.

[135] A. International, "ASTM E111 - 17: Standard Test Method for Young's Modulus, Tangent Modulus, and Chord Modulus," tech. rep., American Society of Testing and Materials, 2017.

APPENDIX A

MICROSTRUCTURE GENERATION CODE

A.1 Aggregate Reconstruction and Pre-Rotation

```
addpath('C:\Users\ctorrence\Documents\Plot_PBC')

cd 'D:\Christa\aggregate\MA108BD-6-coarse'

data = readtable('MA108BD-6-coarse-geom.dat','ReadVariableNames', false);

nlim = data:,13; %'nlim' is the optimal nmax value indicated in the dataset

temp = data(:,1);

s = char(table2cell(temp(:,1)));

volumes = data:,7;

vol = zeros(length(nlim)*10,1);

parfor e=1:1950 %195 aggregates in original file

valu = round((e-5)/10)+1; %Need 10 copies of original aggregate

%'valu' linkes 'e' to original aggregate number

a = dlmread(string(s(valu,:))); %Read anm coefficient file

b = zeros(length(a(:,1)),4); %'b' will hold the new, rotated coefficients

alpha = rand*2*pi; %Random rotation about the X-axis
```

```
beta = rand*2*pi; %Random rotation about the Y-axis

gamma =rand*2*pi; %Random rotation about the Z-axis

cosbeta = cos(beta2);

sinbeta = sin(beta2);

for h=1:((nlim(valu,1)+1)2̂) %'nlim' is the n_max value suggested in the orignal dataset

n=a(h,1); %Column 1 is n value

m=a(h,2); %Column 2 is m value

for mp=n:-1:-n

ddd=sqrt(factorial(n+mp)*factorial(n-mp)/factorial(n+m)/factori al(n-m));

klow=max(0,m-mp);

khigh=min(n-mp,n+m);

total=0.0;

for k=klow:khigh

abc=((-1)(̂k+mp-m))*( factorial(n+m)/factorial(k)/factorial(n+m-k) )*

( factorial(n-m)/factorial(n-mp-k)/factorial(-m+mp+k) ); total=total+

abc*(cosbeta(̂2*n+m-mp-2*k))*(sinbeta(̂2*k+mp-m));

end

ddd=ddd*total*(cos(mp*alpha)-1i*sin(mp*alpha))* (cos(m*gamma) + -1i*sin(m*gamma));
```

```
row = (n*n) + n - m + 1;

row_mp = (n*n) + n - mp + 1;

t = (a(row_mp,3) + 1i*a(row_mp,4))*ddd;

b(row,3) = b(row,3) +real(t);

b(row,4) = b(row,4) + imag(t);

b(row,1) = round(n);

b(row,2) = round(m);

end

end

anm(e).a = b; %Store rotated anm coefficients

anm(e).rot = [alpha, beta, gamma]; %Store random rotation values

end

parfor e=1:1950

j=0;

i=0;

valu = round(e-5/10)+1;

a = anm(e).a; %a is the set of pre-rotated anm coefficients

R=zeros(120,240); % R will store the radius
```

```
Aggtemp = zeros(28800,3); %Aggtemp will store the Cartesian coordinates

AggtempS=zeros(120,240); %AggtempS will store the Spherical coordinates

counter=1;

for t=1:120 %Theta (t is a counter, represents the number of increments of theta)

%If t or p incriments are changed - remember to update pi incriments below

for p=1:240 %Phi (p is a counter, represents the number of increments of phi)

for h=1:((nlim(valu)+1)2̂) %Reads anm file from n=0 n=nlim which includes m=-n to m=n for
each n

n=a(counter,1); %Column 1 is n value

m=a(counter,2); %Column 2 is m value %Columns 3 and 4 are real and imaginary anm coefficients
R(t,p)= (R(t,p)+((a(counter,3)+1i*a(counter,4))*ytotfun(n,m,cos(j),i)));

counter=counter+1;

end

k = (t-1)*240 + p;

j = t*(pi/120);

i = p* (pi/120);

AggtempS(t,p)=R(t,p);

Aggtemp(k,1)=real(R(t,p)*sin(j)*cos(i)); %X value in Cartesian

Aggtemp(k,2)=real(R(t,p)*sin(j)*sin(i)); %Y value in Cartesian
Aggtemp(k,3)=real(R(t,p)*cos(j)); %Z value in Cartesian

k=k+1;

counter=1;

%i=i+(pi/120); %Incriment for i and j should be the same %Bc i needs to go to 2pi, and j to 1pi

end

%i=0; %Set phi back to zero to sample all the phi's again for each theta

%j=(j+(pi/120));

end

AggregateR(e).C=Aggtemp; %Stores aggregate in a structure array

rmax(e) = real(max(max(AggtempS)));

rmin(e) = real(min(min(AggtempS)));

AggregateR(e).rot = anm(e).rot;

vol(e) = volumes(valu);

end
```

A.2 Asperity Identification

```
SetDirectory["D:\\asperities\ma108_rot_anm"];

fileNames = FileNames[];

NNmax = 12;

(*Can use a reduced NNmax than that listed on the data sheet because  we're looking for major
```

asperities, not concerned with the surface detailing provided by NN > 10 Garboczi’s paper gives the details about the equations used here *)

```
kappa[m_, n_] = Sqrt[((2 n + 1) (n - m)!)/(4 Pi (n + m)!)];

Y[Theta_, Phi_, m_, n_] =

Piecewise[(kappa[m, n])* LegendreP[n, m, Cos[Theta ]*Exp[I*m*Phi], m >= 0, (n - Abs[m])!/ (n + Abs[m])!*(-1)Âbs[m] *(kappa[m, n])* LegendreP[n,Abs[m], Cos[Theta]* Exp[I*m*Phi], m < 0];

rowl[m_, n_] = (n*n) + n - m + 1;

ParallelDo[

a = Import[fileNames[[i]], "Table"];

r[Theta_, Phi_] = (

Sum,[ n = 0, NNmax] ( Sum [m = -n, n]((a[rowl[m, n], 3] + I*a[rowl[m, n], 4])* Y[Theta, Phi, m, n]);

rprimeTheta[Theta_, Phi_] =

Re[D[r[Theta, Phi], Theta]];

rprimePhi[Theta_, Phi_] = Re[D[r[Theta, Phi], Phi]];

rdoubleprimeTheta[Theta_, Phi_] =

Re[D[r[Theta, Phi], Theta, 2]];

rdoubleprimePhi[Theta_, Phi_] = Re[D[r[Theta, Phi], Phi, 2]];

h = ;
t1 = 0.5;

t2 = 0;

For[p = Pi/180 ,

p < (2 Pi - ((Pi*5)/180)), (p = p + Pi/180),

For[t = (Pi*5)/90 ,

t < (Pi - ((Pi*5)/90)), (t = t + Pi/90),

If[rprimePhi[t, p] < t1  rprimePhi[t, p] > -t1

rprimeTheta[t, p] < t1  rprimeTheta[t, p] > - t1

rdoubleprimeTheta[t, p] < t2  rdoubleprimePhi[t, p] < t2,

AppendTo[

h, r[t, p], t, p, rdoubleprimeTheta[t, p],

rdoubleprimePhi[t, p] ];

]

]

];

Export["D:\Christa\asperities\ma108_rot_asp2\" <> fileNames[[i]],

N[Re[h]]],

i, 1, Length[fileNames]]
```

(*Adjust the i, start number, end number above to sweep the range

of files desired *)

A.3 Microstructure Generation Main File

```
clear

clc

cd 'C:\Users\ctorrence\Documents\Plot_Pre_Rotated'

domain = 75; %Box edge length in mm

vf = 0.38; %Target volume fraction in decimal form

set_name ='1_38_75'; %Name for the current plot

%open output files, one for data, one for counters [outfile,message] = fopen(strcat(set_name,'.txt'),'wt');

[counterfile,message2] = fopen(strcat(set_name,'_counter.txt'),'wt');

%Load data

load('Aggregate_MA108_Rotated_Asp.mat')

voldata=dlmread('MA108BD-6-coarse-data.dat'); %Another data file

xrot=0;

yrot=0;

zrot=0;

rng('shuffle'); %Shuffle the random number generator seed %This is very important when plotting
multiple microstructures simultaneously
```

```
%The variable 'box' is a structure variable used to store all the info %about each aggregate that is
placed in the domain

box = struct('coords', cell(1, 2120),'center', cell(1, 2120),'asp', cell(1, 2120),'radius', cell(1, 2120),'info',
cell(1, 2120)); box(1).coords = zeros(28800,3); %Stores the point cloud that represents each ag-
gregate

box(1).center = [0,0,0];

box(1).radius = 0; %Maximum radius of the aggregate

box(1).info = [0,0,0,0,0]; %Aggregate number, counter, phi rot, theta rot

box(1).asp = [0,0,0];

box_position = 1;

indiv_counter = 0;

indiv_counter_max = 25; %Max tries for individual aggregate before the code moves on

boxtemp(1).C=zeros(28800,3);

volumesum = 0; %Keeps track of the volume of aggregates already placed

counter = 1; %Keeps track of the number of aggregates placed

rotcounter = 0; %Tallies the number of rotations an aggregate is subjected to

volumes = voldata(:,7);

rsort=voldata(:,15); %Gets the largest width of each aggregate, used to sate

group1 = (find(rsort<15.0rsort>=12.5));
```

```
group2 = (find(rsort<12.5rsort>=9.5));

group3 = (find(rsort<9.5rsort>=4.75));

group4 = (find(rsort<6.75));

total = tic; %Used to time the plotting process

for j=1:15000 % j is max number of aggregate plotting attempts

% j is used to cut off the plotting process if the volume fraction

% can't be reached

aggtime = tic;

asp_counter=0;

bf_counter = 0;

total_counter = 0;

asp_flag = 0;

bf_flag = 0;

diamin_flag_counter = 0;

asp_flag_counter = 0;

bf_flag_counter = 0;

dia_counter = 0;

%s is a scaling variable, not used
```

```
s = 1;

clear boxtemp %Make sure temporary box is empty

boxtemp(1).C = zeros(28800,3); %Set all coordinates to zero to initialize

if (volumesum/domain3̂) > vf %Break j loop if volume fraction is reached

break;

end

%Below is selecting which sieve to grab an aggregate from

%This can be chosen multiple ways, but we use volume fraction as our guide

%Aggregates plotted from largest sieve to smallest

if (volumesum/domain^3) < 0.4 * vf

sieve_level = 1;

select = randi([1,length(group1)]);

num = group1(select);

elseif (volumesum/domain^3) < 0.65 * vf

sieve_level = 2;

select = randi([1,length(group2)]);

num = group2(select);

elseif (volumesum/domain^3) < 0.8 * vf
```

```
sieve_level = 3;

select = randi([1,length(group3)]);

num = group3(select);

elseif (volumesum/domain^3) < vf

sieve_level = 4;

select = randi([1,length(group4)]);

num = group4(select);

R = round(real(AggregateR(num).C),2); %Get cartesian coordinates for aggregate

asp = AggregateR(num).m;

x = rand*domain; %x-offset

y = rand*domain; %y-offset

z = rand*domain; %z-offset

rot_counter = 0;

%Place the aggregate in the domain using center and rotation

[boxtemp,pbc_counter] = PBC_Placement_PreRotated(domain,x,y,z,boxtemp,R,s,asp);

indiv_counter = 0;

if counter > 1 %If more than one aggregate is in the domain, overlap check needed

rotcounter = 0;
```

```
[truefalse,diaminflag,asp_counter,bf_counter,total_counter, asp_flag,bf_flag, dia_counter]=
OverlapCheckPreRotated(box,boxtemp,rmax,rmin,num,box_position,
asp_counter,bf_counter,total_counter, dia_counter);
while truefalse==1 %Overlap detected
indiv_counter = indiv_counter+1;
%If max number of tries exceeded, the aggregate will be discarded
if indiv_counter > indiv_counter_max
break
end
rng('shuffle'); %Shuffle the random number generator seed
x = rand*domain; %x-offset
y = rand*domain; %y-offset
z = rand*domain; %z-offset
clear boxtemp
boxtemp(1).C = zeros(length(R(:,1)),6);
[boxtemp,pbc_counter] = PBC_Placement_PreRotated(domain,x,y,z,boxtemp,R,s,asp);
[truefalse,diaminflag,asp_counter,bf_counter,total_counter, asp_flag,bf_flag, dia_counter]
=OverlapCheckPreRotated(box,boxtemp,rmax,rmin,num,box_position,
asp_counter,bf_counter,total_counter, dia_counter);

if asp_flag ==1

asp_flag_counter = asp_flag_counter + 1;

elseif bf_flag == 1

bf_flag_counter = bf_flag_counter + 1;

end

if diaminflag == 1

diamin_flag_counter = diamin_flag_counter + 1;

end

end

end

if indiv_counter < indiv_counter_max

for box_count=1:length(boxtemp)

box(box_position).coords = boxtemp(box_count).C;

box(box_position).asp = boxtemp(box_count).asp;

box(box_position).radius = s*rsort(num);

box(box_position).center = boxtemp(box_count).cent;

box(box_position).info = [counter,num,xrot,yrot,zrot];
```

```
box_position = box_position+1;

fprintf(outfile,'%d',num,boxtemp(box_count).cent(1),boxtemp(box_count).cent(2),boxtemp(box_count).cent(3),
ot,zrot,pbc_counter);

end

volumesum = volumesum+(volumes(num)*(s^3));

fprintf(counterfile,'1, %f, %d, %d, %d, %d, %d, %d, %d, %f, %d, %d',toc(aggtime), indiv_counter,
total_counter,asp_counter, asp_flag_counter,

bf_counter, bf_flag_counter, (volumesum/domain^3), sieve_level, dia_counter, diamin_flag_counter);

fprintf('END OF PLACEMENT LOOP for aggregate number %d, volume fraction is %f and j is
%d',counter,(volumesum/domain^3), j)

rotcounter=0;

counter = counter+1;

else

fprintf(counterfile,'0, %f, %d, %d, %d, %d, %d, %d, %d, %f, %d, %d',toc(aggtime), indiv_counter,
total_counter,asp_counter, asp_flag_counter,

bf_counter, bf_flag_counter, (volumesum/domain^3), sieve_level, dia_counter, diamin_flag_counter);

end

end

fclose(outfile);

fclose(counterfile);
```

```
total_end = toc(total);

fprintf("total runtime = %f seconds",total_end)
```

A.4 Particle Placement

A.4.1 Periodic Placement

```
function [boxtemp,pbc_pass]=PBC_Placement(domain,x,y,z,xrot,yrot,zrot,boxtemp,R)

pbc_counter=0;

pos_x=0;

neg_x=0;

pos_y=0;

neg_y=0;

pos_z=0;

neg_z=0;

boxtemp(1).C=zeros(legnth(R(:,1)),3);

boxtemp(1).rot=[xrot,yrot,zrot];

rotmatrix_x = [1,0,0; 0, cosd(xrot), -sind(xrot); 0 sind(xrot), cosd(xrot)];

rotmatrix_y = [cosd(yrot), 0, sind(yrot); 0 ,1 , 0; -sind(yrot), 0, cosd(yrot)];

rotmatrix_z = [cosd(zrot), -sind(zrot), 0; sind(zrot), cosd(zrot), 0; 0, 0, 1];

for q=1:length(R(:,1)) coords=[R(q,1),R(q,2),R(q,3)];

boxtemp(1).C(q,1:3) = ((coords*rotmatrix_z)*rotmatrix_x)*rotmatrix_y + [x,y,z];
```

```
end

for i=1:length(R(:,1))

if boxtemp(1).C(i,1) < 0

neg_x=1;

elseif boxtemp(1).C(i,1) > domain

pos_x=1;

end

if boxtemp(1).C(i,2) < 0

neg_y=1;

elseif boxtemp(1).C(i,2) > domain

pos_y=1;

end

if boxtemp(1).C(i,3) < 0

neg_z=1;

elseif boxtemp(1).C(i,3) > domain

pos_z=1;

end

end
```

```
boxtemp(1).cent=[x,y,z];

box_storage_counter=2;

if (pos_x==1)

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) - domain);

boxtemp(box_storage_counter).cent=[x-domain,y,z];

box_storage_counter=box_storage_counter+1;

elseif(neg_x==1)

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) + domain);

boxtemp(box_storage_counter).cent=[domain+x,y,z];

box_storage_counter = box_storage_counter+1;

end

if (pos_y==1)

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;
boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) - domain);

boxtemp(box_storage_counter).cent=[x,y-domain,z];

box_storage_counter = box_storage_counter+1;

elseif(neg_y==1)

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) + domain);

boxtemp(box_storage_counter).cent=[x,domain+y,z];

box_storage_counter = box_storage_counter+1;

end

if (pos_z==1)

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) - domain);

boxtemp(box_storage_counter).cent=[x,y,z-domain];

box_storage_counter = box_storage_counter+1;

elseif(neg_z==1)

pbc_counter=pbc_counter+1;
```

```
boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) + domain);

boxtemp(box_storage_counter).cent=[x,y,domain+z];

box_storage_counter = box_storage_counter+1;

end

if pbc_counter>1

if neg_x =0 || pos_x =0

if neg_y =0 || pos_y =0

if neg_x =0

if neg_y =0 %-,-

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) + domain);

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) + domain);

boxtemp(box_storage_counter).cent=[domain+x,domain+y,z]; box_storage_counter = box_storage_counter+1;

elseif pos_y =0 %-,+ pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) + domain);
boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) - domain);

boxtemp(box_storage_counter).cent=[domain+x,y-domain,z];

box_storage_counter = box_storage_counter+1;

end

else if neg_y =0 %+,-

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) - domain);

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) + domain);

boxtemp(box_storage_counter).cent=[x-domain,domain+y,z];

box_storage_counter = box_storage_counter+1;

elseif pos_y =0 %+,+ pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) - domain);

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) - domain);

boxtemp(box_storage_counter).cent=[x-domain,y-domain,z];

box_storage_counter = box_storage_counter+1;

end
```

```
end

end

if neg_z =0 || pos_z =0

if neg_x =0

if neg_z =0 %-,- pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) + domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) + domain);

boxtemp(box_storage_counter).cent=[domain+x,y,domain+z];

box_storage_counter = box_storage_counter+1;

elseif pos_z =0 %-,+ pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) + domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) - domain);

boxtemp(box_storage_counter).cent=[domain+x,y,z-domain];

box_storage_counter = box_storage_counter+1;

end

else
```

```
if neg_z =0 %+,- pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) - domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) + domain);

boxtemp(box_storage_counter).cent=[x-domain,y,domain+z];

box_storage_counter = box_storage_counter+1;

elseif pos_z =0 %+,+

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) - domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) - domain);

boxtemp(box_storage_counter).cent=[x-domain,y,z-domain];

box_storage_counter = box_storage_counter+1;

end

end

end

end

if neg_y =0 || pos_y =0
if neg_y =0

if neg_z =0 %-,- pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) + domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) + domain);

boxtemp(box_storage_counter).cent=[x,domain+y,domain+z];

box_storage_counter = box_storage_counter+1;

elseif pos_z =0 %-,+

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) + domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) - domain);

boxtemp(box_storage_counter).cent=[x,domain+y,z-domain];

box_storage_counter = box_storage_counter+1;

end

elseif pos_y =0

if neg_z =0

%+,-
```

```
pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) - domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) + domain);

boxtemp(box_storage_counter).cent=[x,y-domain,domain+z];

box_storage_counter = box_storage_counter+1;

elseif pos_z =0 %+,+

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C = boxtemp(1).C;

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) - domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) - domain);

boxtemp(box_storage_counter).cent=[x,y-domain,z-domain];

box_storage_counter = box_storage_counter+1;

end

end end

end

if ((neg_x =0 || pos_x =0)  (neg_y =0 || pos_y =0)  (neg_z =0 || pos_z =0))

if neg_x =0
if neg_y =0

if neg_z =0 %-,-,-

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C=boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) + domain);

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) + domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) + domain);

boxtemp(box_storage_counter).cent=[domain+x,domain+y,domain+z];

elseif pos_z =0 %-,-,+

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C=boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) + domain);

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) + domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) - domain);

boxtemp(box_storage_counter).cent=[domain+x,domain+y,z-domain]; %fprintf('+x +y -z')

end

elseif pos_y =0

if neg_z =0 %-,+,-
```

```
pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C=boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) + domain);

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) - domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) + domain);

boxtemp(box_storage_counter).cent=[domain+x,y-domain,domain+z];

%fprintf('+x -y -z') elseif pos_z =0

%-,+,+

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C=boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) + domain);

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) - domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) - domain);

boxtemp(box_storage_counter).cent=[domain+x,y-domain,z-domain]; %fprintf('+x -y -z')

end

end

elseif pos_x =0

if neg_y =0
if neg_z =0 %+,-,-

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C=boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) - domain);

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) + domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) + domain);

boxtemp(box_storage_counter).cent=[x-domain,domain+y,domain+z]; %fprintf('-x +y +z')

elseif pos_z =0 %+,-,+

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C=boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) - domain);

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) + domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) - domain);

boxtemp(box_storage_counter).cent=[x-domain,domain+y,z-domain]; %fprintf('-x +y -z')

end

else

if neg_z =0 %+,+,-

pbc_counter=pbc_counter+1;
```

```
boxtemp(box_storage_counter).C=boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) - domain);

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) - domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) + domain);

boxtemp(box_storage_counter).cent=[x-domain,y-domain,domain+z];

%fprintf('-x -y +z') elseif pos_z =0 %+,+,+

pbc_counter=pbc_counter+1;

boxtemp(box_storage_counter).C=boxtemp(1).C;

boxtemp(box_storage_counter).C(:,1) = (boxtemp(box_storage_counter).C(:,1) - domain);

boxtemp(box_storage_counter).C(:,2) = (boxtemp(box_storage_counter).C(:,2) - domain);

boxtemp(box_storage_counter).C(:,3) = (boxtemp(box_storage_counter).C(:,3) - domain);

boxtemp(box_storage_counter).cent=[x-domain,y-domain,z-domain]; %fprintf('-x -y -z')

end

end

end

end %pbc==3

pbc_pass=pbc_counter;
```

A.4.2 Intersecting/Cored Placement

```
function [boxtemp]=Intersecting_Placement(x,y,z,xrot,yrot,zrot,boxtemp,R,s,asp)
```

```
boxtemp(1).C=zeros(length(R(:,1)),3);

boxtemp(1).asp = zeros(length(asp(:,1)),3);

boxtemp(1).rot = [xrot, yrot, zrot];

rotmatrix_x = [1,0,0; 0, cosd(xrot), -sind(xrot); 0 sind(xrot), cosd(xrot)];

rotmatrix_y = [cosd(yrot), 0, sind(yrot); 0 ,1 , 0; -sind(yrot), 0, cosd(yrot)];

rotmatrix_z = [cosd(zrot), -sind(zrot), 0; sind(zrot), cosd(zrot), 0; 0, 0, 1];

for q=1:length(R(:,1))

coords=[R(q,1),R(q,2),R(q,3)];

boxtemp(1).C(q,1:3) = ((coords*rotmatrix_z)*rotmatrix_x)*rotmatrix_y + [x,y,z];

end

for q=1:length(asp(:,1)) coords=[asp(q,1);asp(q,2);asp(q,3)];

boxtemp(1).asp(q,1:3) = ((coords*rotmatrix_z)*rotmatrix_x)*rotmatrix_y + [x,y,z];

end

boxtemp(1).cent=[x,y,z];
```

A.4.3 Cast Placement

function [boxtemp,pbc_flag]=Cast$_Placement(domain, x, y, z, xrot, yrot, zrot, boxtemp, R, s, Agg)$

```
pbc_flag=0;

boxtemp(1).C=zeros(length(R(:,1)),3);
```

```
rotmatrix_x = [1,0,0; 0, cosd(xrot), -sind(xrot); 0 sind(xrot), cosd(xrot)];

rotmatrix_y = [cosd(yrot), 0, sind(yrot); 0 ,1 , 0; -sind(yrot), 0, cosd(yrot)];

rotmatrix_z = [cosd(zrot), -sind(zrot), 0; sind(zrot), cosd(zrot), 0; 0, 0, 1];

for q=1:length(R(:,1))

coords=[R(q,1),R(q,2),R(q,3)];

coords3 = ((coords*rotmatrix_z)*rotmatrix_x)*rotmatrix_y;

if (s*coords3(1) + x)<=0

pbc_flag = 1;

break;

end

if (s*coords3(1) + x)>domain

pbc_flag = 1;

break;

end

if (s*coords3(2) + y)<=0

pbc_flag = 1;

break;

end
```

```
if (s*coords3(2) + y)>domain

pbc_flag = 1;

break;

end

if (s*coords3(3) + z)<=0

pbc_flag = 1;

break;

end

if (s*coords3(3) + z)>domain

pbc_flag = 1;

break;

end

boxtemp(1).C(q,1) = real(s*coords3(1) + x);

boxtemp(1).C(q,2) = real(s*coords3(2) + y);

boxtemp(1).C(q,3) = real(s*coords3(3) + z);

end

boxtemp(1).cent=[x,y,z];
```

A.5 Overlap Detection with Asperity Check

```
function [truefalse,diaminflag,asp_counter,bf_counter,total_counter, asp_flag,bf_flag, dia_counter]
```

```
=OverlapCheckPreRotatedAsperity(box,boxtemp,rmax,rmin,num,box_position,asp_counter

,bf_counter,total_counter, dia_counter)

truefalse=0;

diaminflag=0;

asp_flag = 0;

bf_flag = 0;

to_check = zeros(box_position,8);

to_check_counter = [0,0,0,0,0,0,0,0];

for u=1:length(boxtemp)%Accounts for PBC aggregates having multiple pieces

x=boxtemp(u).cent(1);

y=boxtemp(u).cent(2);

z=boxtemp(u).cent(3);

for i=1:(box_position-1) %Checks against all previously placed aggregates and their PBC pieces

x0=box(i).center(1);

y0=box(i).center(2);

z0=box(i).center(3);

d1=sqrt((x0-x)^2 + (y0-y)^2 + (z0-z)^2);

%Calculate the distance between the centers of the two aggregates
```

```
diamax=box(i).radius + rmax(num);

diamin=(rmin(num)+rmin(box(i).info(2)));

dia_counter = dia_counter + 1;

%Diamax is sum of maximum radii (which is the radius of the enclosing sphere)

%Diamin is sum of the minimum radii

%If the distance is less than diamin, there is no way the aggregates do not overlap

%Diamin flag tells the main code to forgo trying to rotate the aggregate and it simply translate it

if d1<(diamax)

if d1<(diamin)

truefalse=1;

diaminflag=1;

break

end

if truefalse==1

break

end

to_check_counter(u) = to_check_counter(u) + 1;

to_check(to_check_counter(u),u) = i;
```

```
% If overlap is not detected by the asperities, "to_check" will store the aggregate number to assess
in the brute force check

old_num=box(i).info(2);

asp_counter = asp_counter + 1;

total_counter = total_counter + 1;

%———————————————Loop 1——————————————-

asp_to_check = boxtemp(u).asp(:,:) - [x0, y0,z0];

%Filter the asperity list to remove asperities that do not fall into the overlap lens

asp_to_check(sqrt(asp_to_check(:,1).2̂ + asp_to_check(:,2).2̂+ asp_to_check(:,3).2̂ ) > rmax(old_num),:)
= [];

if  isempty(asp_to_check)

dist_to_check = sqrt(sum(asp_to_check.2̂,2));

g = asp_to_check;

gx = g(:,1)./sqrt(g(:,1).2̂ + g(:,2).2̂);

gy = g(:,2)./sqrt(g(:,1).2̂ + g(:,2).2̂);

gz = g(:,3)./sqrt(g(:,1).2̂ + g(:,2).2̂+ g(:,3).2̂);

phi = rad2deg(acos(gx));

for p=1:length(gy)

if gy(p) >= 0
```

```
phi(p) = rad2deg(acos(gx(p)));

else

phi(p) = rad2deg(2*pi - acos(gx(p)));

end

end

theta = rad2deg(acos(gz));

row=(round((12/18)*theta) - 1)*240 + round((24/36)*phi);

rad = sqrt(sum((box(i).coords(row,:) - [x0, y0, z0]).2̂,2));

tester = rad - dist_to_check;

if length(tester(tester>-0.5))>0

truefalse=1;

asp_flag=1;

break;

end

end

if truefalse==1

break

end
```

```
%————————————Loop 2—————————————————————————————————

asp_to_check2 = box(i).asp(:,:) - [x,y,z];

asp_to_check2(sqrt(asp_to_check2(:,1).2̂ + asp_to_check2(:,2).2̂+ asp_to_check2(:,3).2̂ ) > rmax(num),:)
= [];

if  isempty(asp_to_check2)

dist_to_check = sqrt(sum(asp_to_check2.2̂,2));

g = asp_to_check2;

gx = g(:,1)./sqrt(g(:,1).2̂ + g(:,2).2̂);

gy = g(:,2)./sqrt(g(:,1).2̂ + g(:,2).2̂);

gz = g(:,3)./sqrt(g(:,1).2̂ + g(:,2).2̂+ g(:,3).2̂);

phi = rad2deg(acos(gx));

for p=1:length(gy)

if gy(p) >= 0

phi(p) = rad2deg(acos(gx(p)));

else

phi(p) = rad2deg(2*pi - acos(gx(p)));

end

end

theta = rad2deg(acos(gz));
```

```
row=(round((12/18)*theta) - 1)*240 + round((24/36)*phi);

rad = sqrt(sum((boxtemp(u).C(row,:) - [x, y, z]).2̂,2));

tester = rad - dist_to_check;

if length(tester(tester>-0.5))>0

truefalse=1;

asp_flag=1;

break;

end

end

if truefalse==1

break

end

end

if truefalse==1

break

end

end

if truefalse==1
```

```
break

end

end

if truefalse==0

for u = 1:length(boxtemp)

x = boxtemp(u).cent(1);

y = boxtemp(u).cent(2);

z = boxtemp(u).cent(3);

tempx = boxtemp(u).C(:,1);

tempy = boxtemp(u).C(:,2);

tempz = boxtemp(u).C(:,3);

shp = alphaShape(tempx, tempy, tempz);

for j=1:to_check_counter(u)

%Checks against all previously placed aggregates and their PBC pieces

i = to_check(j,u);

if truefalse==1

break

end
```

x0=box(i).center(1);

y0=box(i).center(2);

z0=box(i).center(3);

d1=sqrt((x0-x)$\hat{2}$ + (y0-y)$\hat{2}$ + (z0-z)$\hat{2}$);

%Calculate the distance between the centers of the two aggregates

diamax=box(i).radius + rmax(num);

if d1<(diamax)

bf_counter = bf_counter + 1;

total_counter = total_counter + 1;

%If the asperity check passes, use the build in function inShape

%The function 'in' checks if a point is within the alphaShape

temp = boxtemp(u).C;

if length(temp((sqrt((temp(:,1) - x0).$\hat{2}$ + (temp(:,2) - y0).$\hat{2}$ + (temp(:,3) - z0).$^{)} < rmin(old\_num)), :$
$)) > 0$

truefalse=1;

bf_flag=1;

break

end

temp = box(i).coords;

```
temp((sqrt((temp(:,1) - x).2̂ + (temp(:,2) - y).2̂ + (temp(:,3) - z).2̂ ) > rmax(num)),:) = [];

in = inShape(shp,temp(:,1),temp(:,2),temp(:,3));

if length(in(in>0))>0

truefalse=1;

bf_flag = 1;

break

end

if truefalse==1

break

end

end

if truefalse==1

break

end

end

end

end %Ends u loop
```

A.6 Generation of Image Stack

```
% List of microstructures to be imaged
```

to_image = ["1_40_b";"1_38_75$_{c}ast”;”1\_38\_90_{c}ast”;”1\_38_{9}0\_cored”$];

addpath(’/scratch/user/ctorrence/plotting/prerot’);

%List domain sizes of the microstructures to be imaged

domainlist = [75,75,75,75,90,90];

%Load appropriate aggregate set load(’Aggregate_Rotated_N10.mat’)

for k=1:length(to_image)

%Set domain to proper value

domain = domainlist(k);

%Read microstructure data file

plotdata = dlmread(strcat([to_image(k)],’.txt’));

%Create directory to store images

mkdir([char(to_image(k))]);

% Enter the directory cd ([char(to_image(k))]);

% Recreate microstructure for j=1:length(plotdata)

num = plotdata(j,1);

R=real(AggregateR(plotdata(j,1)).C);

x = plotdata(j,2); %x-offset

y = plotdata(j,3); %y-offset

```
z = plotdata(j,4); %z-offset

xrot = plotdata(j,5);

yrot = plotdata(j,6);

zrot = plotdata(j,7);

%If using pre-rotated aggregates - xrot, yrot, and zrot will be 0

rotmatrix_x = [1,0,0; 0, cosd(xrot), -sind(xrot); 0 sind(xrot), cosd(xrot)];

rotmatrix_y = [cosd(yrot), 0, sind(yrot); 0 ,1 , 0; -sind(yrot), 0, cosd(yrot)];

rotmatrix_z = [cosd(zrot), -sind(zrot), 0; sind(zrot), cosd(zrot), 0; 0, 0, 1];

boxtemp(1).C=zeros(28800,3);

R=real(AggregateR(num).C);

for q=1:length(R(:,1))

coords=[R(q,1),R(q,2),R(q,3)];

coords3 = ((coords*rotmatrix_z)*rotmatrix_x)*rotmatrix_y;

boxtemp(1).C(q,1) = (coords3(1) + x);

boxtemp(1).C(q,2) = (coords3(2) + y);

boxtemp(1).C(q,3) = (coords3(3) + z);

end

box(j).coords=boxtemp(1).C;
box(j).center=[plotdata(j,2),plotdata(j,3),plotdata(j,4)];

end

corners=ones(2,2);

corners(1,1)=-25;

corners(1,2)=-25;

corners(2,1)=domain+25;

corners(2,2)=domain+25;

for b=1:(domainlist(k)/0.3) %Each image represents a 0.3mm thickness

f = (domain/(domainlist(k)/0.3))*b; %determine the z value of the image to be plotted

for i=1:length(plotdata)

slice2=zeros(100000,2); %Will store the points that fall into the z-slice

count=1;

%For each aggregate, check if it’s possible to have points in the z-slice

if abs(box(i).center(3) - f) < max(rmax)

%If it is possible, find the points that fall in the z-slice

for j=1:length(box(i).coords(:,1))

if box(i).coords(j,3)<=(f+(domain/round(domainlist(k)/0.3))) box(i).coords(j,3)>=(f-(domain/round(domainlist(k

%Store said points in a temporary variable
```

```
slice2(count,1)=box(i).coords(j,1);

slice2(count,2)=box(i).coords(j,2);

count=count+1; end end

if count > 4

%Plot the points in a cumulative plot

r = boundary(slice2(1:count-1,1),slice2(1:count-1,2));

x = slice2(1:count,1);

y = slice2(1:count,2);

plot(x(r),y(r),'k-')

drawnow

hold on; end

end

%Plot the control points

hold on;

daspect([1 1 1]);

hold on;

drawnow;

end
```

```
%Add buffer points to standardize the image plot size

plot(-2*max(rmax),-2*max(rmax),'.');

plot(domain+2*max(rmax),domain+2*max(rmax),'.');

fig = gcf;

fig.PaperUnits = 'inches';

fig.PaperPosition = [0 0 15 15];

filename = char(strcat(num2str(b+100),'.png'));

print(filename,'-dpng','-r0');

%Close the cumulative image to start fresh for the next z-slice

close all; end

cd ..

end
```

APPENDIX B

ABAQUS INPUT FILE GENERATION CODE

B.1 Matlab Code to Generate Abaqus Input File and Apply Periodic Boundary Condition Constraint Equations

```
filename = ’5_38_90.inp’;

disp = 0.03; % Amount of strain to apply

fid = fopen(strcat(filename,’.inp’),’rt’) ;

S = textscan(fid,’%s’,’Delimiter’,’\n’);

S = S{1} ;

[outfile,message]= fopen(strcat(filename,’.inp’),’wt’);

% The idx variables are Get the line numbers that mark the beginning and end of sections, such as
the node list, element list, surface node sets, etc

idxS = strfind(S, ’*NODE’);

idx1 = find(not(cellfun(’isempty’, idxS)));

%idx1 = start of node list;

idxS = strfind(S, ’** The OOF3D element type is TET4_4. The ABAQUS element type will’);

idx2 = find(not(cellfun(’isempty’, idxS)));

idxS = strfind(S, ’** Point boundaries in OOF3D’);
```

```
idx3 = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=XminYmin');

idx_xminymin = find(not(cellfun('isempty', idxS)));

idx_xminymin = idx_xminymin(3);

idxS = strfind(S, '*NSET, NSET=YminZmin');

idx_yminzmin = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=YmaxZmax');

idx_ymaxzmax = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=YminZmax');

idx_yminzmax = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=XminZmax');

idx_xminzmax = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=XminYmax');

idx_xminymax = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=XmaxZmin');

idx_xminymax = idx_xminymax(3);

idx_xmaxzmin = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=XminZmin');
```

```
idx_xminzmin = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=XmaxYmax');

idx_xmaxymax = find(not(cellfun('isempty', idxS)));

idx_xmaxymax = idx_xmaxymax(3);

idxS = strfind(S, '*NSET, NSET=YmaxZmin');

idx_ymaxzmin = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=XmaxYmin');

idx_xmaxymin = find(not(cellfun('isempty', idxS)));

idx_xmaxymin = idx_xmaxymin(3);

idxS = strfind(S, '*NSET, NSET=XmaxZmax');

idx_xmaxzmax = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=Zmax');

idx_zmax = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=Ymax');

idx_ymax = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=Xmax');

idx_xmax = find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*NSET, NSET=Xmin');
```

```
idx_xmin = find(not(cellfun('isempty', idxS)));

idx_xmin = idx_xmin(length(idx_xmin));

idxS = strfind(S, '*NSET, NSET=Ymin');

idx_ymin = find(not(cellfun('isempty', idxS)));

idx_ymin = idx_ymin(length(idx_ymin));

idxS = strfind(S, '*NSET, NSET=Zmin');

idx_zmin= find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*ELSET, ELSET=Aggregate');

idx_aggstart= find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*SOLID SECTION, ELSET=Aggregate, MATERIAL=Aggregate');

idx_aggend= find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*ELSET, ELSET=Mortar');

idx_mortarstart= find(not(cellfun('isempty', idxS)));

idxS = strfind(S, '*SOLID SECTION, ELSET=Mortar, MATERIAL=Mortar');

idx_mortarend= find(not(cellfun('isempty', idxS)));

nodes = S(idx1+1:idx2-1) ;

nodes = cell2mat(cellfun(@str2num,nodes,'UniformOutput',false));

elements = S(idx2+3:idx3-1) ;
elements = cell2mat(cellfun(@str2num,elements,'UniformOutput',false));

xminymin_nodelines = S(idx_xminymin+1:idx_yminzmin-1) ;

xminymin_nodelines = cell2mat(cellfun(@str2num,xminymin_nodelines','UniformOutput',false));

yminzmin_nodelines = S(idx_yminzmin+1:idx_ymaxzmax-1) ;

yminzmin_nodelines = cell2mat(cellfun(@str2num,yminzmin_nodelines','UniformOutput',false));

ymaxzmax_nodelines = S(idx_ymaxzmax+1:idx_yminzmax-1) ;

ymaxzmax_nodelines = cell2mat(cellfun(@str2num,ymaxzmax_nodelines','UniformOutput',false));

yminzmax_nodelines = S(idx_yminzmax+1:idx_xminzmax-1) ;

yminzmax_nodelines = cell2mat(cellfun(@str2num,yminzmax_nodelines','UniformOutput',false));

xminzmax_nodelines = S(idx_xminzmax+1:idx_xminymax-1) ;

xminzmax_nodelines = cell2mat(cellfun(@str2num,xminzmax_nodelines','UniformOutput',false));

xminymax_nodelines = S(idx_xminymax+1:idx_xmaxzmin-1) ;

xminymax_nodelines = cell2mat(cellfun(@str2num,xminymax_nodelines','UniformOutput',false));

xmaxzmin_nodelines = S(idx_xmaxzmin+1:idx_xminzmin-1) ;

xmaxzmin_nodelines = cell2mat(cellfun(@str2num,xmaxzmin_nodelines','UniformOutput',false));

xminzmin_nodelines = S(idx_xminzmin+1:idx_xmaxymax-1) ;

xminzmin_nodelines = cell2mat(cellfun(@str2num,xminzmin_nodelines','UniformOutput',false));

xmaxymax_nodelines = S(idx_xmaxymax+1:idx_ymaxzmin-1) ;
xmaxymax_nodelines = cell2mat(cellfun(@str2num,xmaxymax_nodelines','UniformOutput',false));

ymaxzmin_nodelines = S(idx_ymaxzmin+1:idx_xmaxymin-1) ;

ymaxzmin_nodelines = cell2mat(cellfun(@str2num,ymaxzmin_nodelines','UniformOutput',false));

xmaxymin_nodelines = S(idx_xmaxymin+1:idx_xmaxzmax-1) ;

xmaxymin_nodelines = cell2mat(cellfun(@str2num,xmaxymin_nodelines','UniformOutput',false));

xmaxzmax_nodelines = S(idx_xmaxzmax+1:idx_zmax-2) ;

xmaxzmax_nodelines = cell2mat(cellfun(@str2num,xmaxzmax_nodelines','UniformOutput',false));

zmax_nodelines = S(idx_zmax+1:idx_ymax-1) ;

zmax_nodelines = cell2mat(cellfun(@str2num,zmax_nodelines','UniformOutput',false));

ymax_nodelines = S(idx_ymax+1:idx_zmin-1) ;

ymax_nodelines = cell2mat(cellfun(@str2num,ymax_nodelines','UniformOutput',false));

zmin_nodelines = S(idx_zmin+1:idx_xmax-1) ;

zmin_nodelines = cell2mat(cellfun(@str2num,zmin_nodelines','UniformOutput',false));

xmax_nodelines = S(idx_xmax+1:idx_xmin-1) ;

xmax_nodelines = cell2mat(cellfun(@str2num,xmax_nodelines','UniformOutput',false));

xmin_nodelines = S(idx_xmin+1:idx_ymin-1) ;

xmin_nodelines = cell2mat(cellfun(@str2num,xmin_nodelines','UniformOutput',false));

ymin_nodelines = S(idx_ymin+1:idx_aggstart-1) ;
ymin_nodelines = cell2mat(cellfun(@str2num,ymin_nodelines','UniformOutput',false));

agg_elements = S(idx_aggstart+1:idx_aggend-1) ;

agg_elements = cell2mat(cellfun(@str2num,agg_elements','UniformOutput',false));

mortar_elements = S(idx_mortarstart+1:idx_mortarend-1) ;

mortar_elements = cell2mat(cellfun(@str2num,mortar_elements','UniformOutput',false));

% Remove nodes common to multiple opposing boundaries - Abaqus will exit if a node is con-
strained to more than one other node in the same degree of freedom

remove_zmax = unique(horzcat(ymaxzmax_nodelines,yminzmax_nodelines,xminzmax_nodelines,xmaxzmax_no

remove_zmin = unique(horzcat(yminzmin_nodelines, xmaxzmin_nodelines, xminzmin_nodelines,
ymaxzmin_nodelines));

remove_ymax = unique(horzcat(xminymax_nodelines, xmaxymax_nodelines));

remove_ymin = unique(horzcat(xminymin_nodelines, xmaxymin_nodelines));

zmax_nodes = zeros(length(zmax_nodelines) - length(remove_zmax),4);

counter = 1;

for i=1:length(zmax_nodelines)

if ismember(zmax_nodelines(i), remove_zmax) == 0

zmax_nodes(counter,:) = nodes(zmax_nodelines(i),:);

counter = counter + 1;

end
```

```
end

ymax_nodes = zeros(length(ymax_nodelines) - length(remove_ymax),4);

counter = 1;

for i=1:length(ymax_nodelines)

if ismember(ymax_nodelines(i), remove_ymax) == 0

ymax_nodes(counter,:) = nodes(ymax_nodelines(i),:);

counter = counter + 1;

end

end

xmax_nodes = zeros(length(xmax_nodelines),4);

for i=1:length(xmax_nodelines)

xmax_nodes(i,:) = nodes(xmax_nodelines(i),:);

end

zmin_nodes = zeros(length(zmin_nodelines) - length(remove_zmin),4);

counter = 1;

for i=1:length(zmin_nodelines)

if ismember(zmin_nodelines(i), remove_zmin) == 0

zmin_nodes(counter,:) = nodes(zmin_nodelines(i),:);
```

```
counter = counter + 1;

end 0 . end

ymin_nodes = zeros(length(ymin_nodelines) - length(remove_ymin),4);

counter = 1;

for i=1:length(ymin_nodelines)

if ismember(ymin_nodelines(i), remove$_ymin) == 0$

ymin_nodes(counter,:) = nodes(ymin_nodelines(i),:);

counter = counter + 1;

end

end

xmin_nodes = zeros(length(xmin_nodelines),4);

for i=1:length(xmin_nodelines)

xmin_nodes(i,:) = nodes(xmin_nodelines(i),:);

end

y_pairs = zeros(length(ymax_nodes(:,1)),2);

z_pairs = zeros(length(zmax_nodes(:,1)),2);

for i=1:length(ymax_nodes(:,1))

min_distance = 50;
```

```
j=1;

for j=1:length(ymin_nodes(:,1))

distance = sqrt((ymax_nodes(i,2) - ymin_nodes(j,2))2̂ + (ymax_nodes(i,4) - ymin_nodes(j,4))2̂);

if distance < min_distance

min_distance = distance;

number = ymin_nodes(j,1);

end

end

y_pairs(i,1) = ymax_nodes(i,1);

y_pairs(i,2) = number;

end

for i=1:length(zmax_nodes(:,1))

min_distance = 50;

j=1;

for j=1:length(zmin_nodes(:,1))

distance = sqrt((zmax_nodes(i,3) - zmin_nodes(j,3))2̂ + (zmax_nodes(i,2) - zmin_nodes(j,2))2̂);

if distance < min_distance

min_distance = distance;
```

```
number = zmin_nodes(j,1);

end end

z_pairs(i,1) = zmax_nodes(i,1);

z_pairs(i,2) = number; end

fprintf(outfile,"*Heading\n**\n*Preprint, echo=NO, model=NO, history=NO, contact=NO\n");

fprintf(outfile,"**PARTS\n*Part, name=Microstructure\n*Node\n");

for i=1:length(nodes)

fprintf(outfile,"%d, %f, %f, %f \n",nodes(i,1),nodes(i,2),nodes(i,3),nodes(i,4));

end

fprintf(outfile,"*Element, type=C3D4\n");

for i=1:length(elements)

fprintf(outfile,"%d, %d, %d, %d, %d \n",elements(i,1),elements(i,2),elements(i,3),elements(i,4),elements(i,5));

end

fprintf(outfile,"*Elset, elset=AGGREGATE\n");

for i=1:length(agg_elements)

fprintf(outfile,"%d,\n", agg_elements(i)) ;

end

fprintf(outfile,"*Elset, elset=Mortar\n");
```

```
for i=1:length(mortar_elements)

fprintf(outfile,"%d,\n", mortar_elements(i)) ;

end

fprintf(outfile,"** Section: Section-1-MORTAR\n*Solid Section, elset=MORTAR, material=MORTAR\n,\n");

fprintf(outfile,"** Section: Section-2-AGGREGATE\n*Solid Section, elset=AGGREGATE,

material=AGGREGATE\n,\n");

fprintf(outfile,"**\n*End Part\n**\n**\n** ASSEMBLY\n**\n*Assembly, name=Assembly\n**
\n");

fprintf(outfile,"*Instance, name=PART-1-1, part=Microstructure\n*End Instance\n");

fprintf(outfile,"*Nset, nset=Xmin, instance=PART-1-1\n");

for i=1:length(xmin_nodes(:,1))

fprintf(outfile,"%d,\n", xmin_nodes(i,1)) ;

end

fprintf(outfile,"*Nset, nset=Ymin, instance=PART-1-1\n");

for i=1:length(ymin_nodes)

fprintf(outfile,"%d,\n", ymin_nodes(i,1)) ;

end

fprintf(outfile,"*Nset, nset=Zmin, instance=PART-1-1\n");

for i=1:length(zmin_nodes)
```

```
fprintf(outfile,"%d,\n", zmin_nodes(i,1)) ;

end

fprintf(outfile,"*Nset, nset=Xmax, instance=PART-1-1\n");

for i=1:length(xmax_nodes)

fprintf(outfile,"%d,\n", xmax_nodes(i,1)) ;

end

fprintf(outfile,"*Nset, nset=Ymax, instance=PART-1-1\n");

for i=1:length(ymax_nodes)

fprintf(outfile,"%d,\n", ymax_nodes(i,1)) ;

end

fprintf(outfile,"*Nset, nset=Zmax, instance=PART-1-1\n");

for i=1:length(zmax_nodes)

fprintf(outfile,"%d,\n", zmax_nodes(i,1)) ;

end

for i=1:length(x_pairs)

fprintf(outfile,"*Nset, nset=X%d, instance=PART-1-1\n %d\n", x_pairs(i,1), x_pairs(i,1));

fprintf(outfile,"*Nset, nset=XC%d, instance=PART-1-1\n %d\n", x_pairs(i,2), x_pairs(i,2));

end
```

```
for i=1:length(y_pairs)

fprintf(outfile,"*Nset, nset=Y%d, instance=PART-1-1\n %d\n", y_pairs(i,1), y_pairs(i,1));

fprintf(outfile,"*Nset, nset=YC%d, instance=PART-1-1\n %d\n", y_pairs(i,2), y_pairs(i,2));

end

for i=1:length(z_pairs)

fprintf(outfile,"*Nset, nset=Z%d, instance=PART-1-1\n %d\n", z_pairs(i,1), z_pairs(i,1));

fprintf(outfile,"*Nset, nset=ZC%d, instance=PART-1-1\n %d\n", z_pairs(i,2), z_pairs(i,2));

end

for i=1:length(y_pairs)

fprintf(outfile,"*Equation\n");

fprintf(outfile,"2\nY%d, 1, 1.0\n YC%d, 1, -1.0\n**\n",y_pairs(i,1),y_pairs(i,2));

fprintf(outfile,"*Equation\n");

fprintf(outfile,"2\n Y %d, 2, 1.0\n YC %d, 2, -1.0\n**\n",y_pairs(i,1),y_pairs(i,2));

fprintf(outfile,"*Equation\n");

fprintf(outfile,"2\n Y %d, 3, 1.0\n YC %d, 3, -1.0\n**\n",y_pairs(i,1),y_pairs(i,2));

end

for i=1:length(z_pairs)

fprintf(outfile,"*Equation\n");
fprintf(outfile,"2\n Z %d, 1, 1.0\n ZC %d, 1, -1.0\n**\n",z_pairs(i,1),z_pairs(i,2));

fprintf(outfile,"*Equation\n");

fprintf(outfile,"2\n Z%d, 2, 1.0\nZC %d, 2, -1.0\n**\n",z_pairs(i,1),z_pairs(i,2));

fprintf(outfile,"*Equation\n");

fprintf(outfile,"2\n Z %d, 3, 1.0\n ZC %d, 3, -1.0\n**\n",z_pairs(i,1),z_pairs(i,2));

end

fprintf(outfile,"*End Assembly\n");

fprintf(outfile,"** MATERIALS\n**\n*Material, name=AGGREGATE\n*Elastic\n3.6e+10, 0.22\n");

fprintf(outfile,"*Material, name=MORTAR\n*Elastic, moduli=INSTANTANEOUS\n 2.559e+10,
0.15\n");

fprintf(outfile,"*Viscoelastic, time=CREEP TEST DATA\n*Combined Test Data \n");

visco = dlmread('basic_creep.txt');

for i=1:length(visco(:,1))

fprintf(outfile,"%f, %f, %f \n",visco(i,1),visco(i,2),visco(i,3)) ;

end

fprintf(outfile,"** ————————————————————————————-\n");

fprintf(outfile,"** \n** STEP: Loading\n** \n*Step, name=Loading, nlgeom=NO, amplitude=STEP\n");

fprintf(outfile,"*Static\n0.0001, 0.0001, 1e-07, 0.0001\n**\n** BOUNDARY CONDITIONS\n**\n");

fprintf(outfile,"** Name: Disp-BC-1 Type: Displacement/Rotation\n*Boundary\n");
fprintf(outfile,"XMAX, 1, 1, -% disp\nXMAX, 2, 2\nXMAX, 3, 3\n",disp );

fprintf(outfile,"XMAX, 4, 4\nXMAX, 5, 5\nXMAX, 6, 6\n");

fprintf(outfile,"** Name: Disp-BC-2 Type: Displacement/Rotation\n*Boundary\n");

fprintf(outfile,"XMIN, 1, 1, % disp \nXMIN, 2, 2\nXMIN, 3, 3\n", disp);

fprintf(outfile,"XMIN, 4, 4\nXMIN, 5, 5\nXMIN, 6, 6\n");

fprintf(outfile,"**\n** OUTPUT REQUESTS\n**\n*Restart, write, frequency=0\n");

fprintf(outfile,"**\n** FIELD OUTPUT: F-Output-1\n**\n*Output, field, variable=PRESELECT\n");

fprintf(outfile,"**\n** HISTORY OUTPUT: H-Output-1\n**\n*Output, history, variable=PRESELECT\n");

fprintf(outfile,"*End Step\n");

fprintf(outfile,"** ——————————————————————————-\n");

fprintf(outfile,"**\n** STEP: Relaxation\n**\n*Step, name=Relaxation, nlgeom=NO, inc=100000\n");

fprintf(outfile,"*Visco, cetol=0.00001\n0.01, 17000., 0.00175, 125.\n**\n");

fprintf(outfile,"** OUTPUT REQUESTS\n**\n*Restart, write, frequency=0\n**\n");

fprintf(outfile,"** FIELD OUTPUT: F-Output-2\n**\n*Output, field, variable=PRESELECT\n");

fprintf(outfile,"**\n** HISTORY OUTPUT: H-Output-2\n**\n");

fprintf(outfile,"*Output, history, variable=PRESELECT\n*End Step\n");

fclose(outfile);
```

APPENDIX C

DATA ANALYSIS CODE

C.1 ODB File Reader

The following is a Python script that reads the ODB file generated by Abaqus. It calculates the spatially averaged stress and strain in the XX-direction for each time step and outputs the data to two text files, one for the strain data and one for the stress. The files are named "filename_E11.txt" and "filename_S11.txt" for the strain and stress data, respectively.

```
import sys

from odbAccess import *

from abaqusConstants import*

from types import IntType

import numpy as np

odb_names = ['odb_file_1','odb_file_2']

for i in range(0,len(odb_names)):

# desired name for the results file

results_name=odb_names[i]

#Initializing Arrays

DAT = []
```

```
DAT2 = []

#Opening the odb

odb = openOdb(odb_names[i]+’.odb’, readOnly=True)

assembly = odb.rootAssembly

instance = assembly.instances.keys()[0]

Extracting Step

step1 = odb.steps.values()[1]

print(step1)

#Creating a for loop to iterate through all frames in the step

for x in odb.steps[step1.name].frames:

####Setting temporary array to empty

temp = []

temp2 = []

####Progress report

print(’Extracting from Frame:’+str(x))

####Reading stress and strain data from the model

odbSelectResults = x.fieldOutputs[’S’]

odbSelectResults2 = x.fieldOutputs[’E’]
```

field1 = odbSelectResults

field2 = odbSelectResults2

####Reading COORDS data from the top node set

####Storing Stress and strain values for the current frame

# stress values

stress_temp = []

for s in field1.values:

stress_temp.append(s.data[0]) the values in s.data are organized as follows:

[S11,S22,S33,S12,S13,S23]

temp.append(x.frameValue)

temp.append(sum(stress_temp)/ len(stress_temp))

# strain values

strain_temp = []

for e in field2.values:

strain_temp.append(e.data[0]) the values here are organized similarly to the stress values. This grabs E33

temp2.append(x.frameValue)

temp2.append(sum(strain_temp)/ len(strain_temp))

DAT.append(temp)

```
DAT2.append(temp2)

####Writing to a .csv file

with open(results_name+'_S11.txt', 'w') as f:

np.savetxt(f,DAT,delimiter=' ')

with open(results_name+'_E11.txt', 'w') as f:

np.savetxt(f,DAT2,delimiter=' ')

#Close the odb

odb.close()
```

C.2 Constituitive Viscoelastic Equation Calculation

The following is a script, written in Mathematica that is used to analyze the output data from the ODB file reader, given in the previous section. Both the relaxation modulus and creep compliance equations are calculated.

```
SetDirectory[thisDir];

Strain = Import["C:/Users/Christa/Google Drive/TAMU/Data/1_38_105_cast_E11.txt,"Table""];

Stress = Import["C:/Users/Christa/Google Drive/TAMU/Data/1_38_105_cast_S11.txt,"Table""];

ϵ[t_, a_, b_, c_, d_] = a*Exp[-t/1] + b*Exp[-t/10] + c*Exp[-t/1000] + d

nlm = FindFit[Strain, {[t, a, b, c, d], {-10 < a < 10}, {-10 < b < 10} , {-10 < c < 10}, {-10 < d < 10}},
 {a, b, c, d}, t]

ϵ_fitted[t_] = epsilon[t, a, b, c, d]*(1 - E(̂-t/.001)) /. nlm
```

```
Show[Plot[ϵ_fitted[t], t, 0, 17000, PlotRange -> 0, 17000, 0, -0.0005], ListPlot[Strain]]

H[t_] = E0 + E1*Exp[-(t)/1] + E2*Exp[-(t)/10] + E3*Exp[-(t)/100] + E4*Exp[-(t)/1000] + E5*Exp[-
(t)/10000] + E6*Exp[-(t)/100000]

$Assumptions -> t ∈ Reals, t > tp, tp >= 0.01;

σ=[t_, E0_, E1_, E2_, E3_, E4_, E5_, E6_] = Integrate[H[t - tp]*D[ϵ_fitted[tp], tp], {tp, 0, t}]

nlm2 = FindFit[ Stress, σ[t, E0, E1, E2, E3, E4, E5, E6], { E0, E1, E2, E3,E4, E5, E6}, t]

Efitted[t_] = H[t] /. nlm2

Plot[Efitted[t]/109̂, {t, 0.01, 17000}, PlotRange -> {{0, 17000}, {40, 0}}, Frame -> True, Frame-
Label -> {"Time (d)", "E (GPa)"}]

Relax[t_] = Efitted[t]*(1/109̂);

poisson = 0.2;

ELaplace[s_] = LaplaceTransform[Relax[t], t, s];

JLaplace[s_] = 1/(s*s*ELaplace[s]);

CreepCompliance[t_] = Simplify[ InverseLaplaceTransform[JLaplace[s], s, t]]
```