Title
Development of a Coupled 3-D DEM-LBM Model for Simulation of Dynamic Rock-Fluid Interaction

Author
Gardner, Michael Henry

Publication Date
2018

Peer reviewed|Thesis/dissertation

Development of a Coupled 3-D DEM-LBM Model for Simulation of Dynamic Rock-Fluid Interaction

by

Michael Henry Gardner

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Nicholas Sitar, Chair
Professor Jonathan Bray
Associate Professor Per-Olof Persson

Summer 2018

Development of a Coupled 3-D DEM-LBM Model for Simulation of Dynamic Rock-Fluid Interaction

Abstract

Development of a Coupled 3-D DEM-LBM Model for Simulation of Dynamic Rock-Fluid Interaction

by

Michael Henry Gardner

Doctor of Philosophy in Engineering – Civil and Environmental Engineering

University of California, Berkeley

Professor Nicholas Sitar, Chair

Scour of rock is a challenging and interesting problem that combines rock mechanics and hydraulics of turbulent flow. On a practical level, rock erosion is a critical issue facing many of the world's dams at which excessive scour of the dam foundation or spillway can compromise the stability of the dam resulting in significant remediation costs, if not direct personal property damage or even loss of life. The most current example of this problem is Oroville Dam in Northern California where massive scour damage to both the service and emergency spillways during the flood events of February 2017 led to the evacuation of more than 188,000 people living downstream of the dam.

This research is specifically aimed at developing the ability to numerically evaluate rock-water interaction, building upon the experimental and analytical work by George and Sitar [[30],[28], [26], [27], [29]]. The focus is on producing simulation techniques capable of considering the interaction between three-dimensional polyhedral rock blocks interacting with fluid such that the complex shape of the blocks is captured in both the fluid and solid numerical models. Accounting for the rock block geometry and orientations is essential in capturing the correct kinematic response.

To this end, a three-dimensional, open-source program to generate the fractured rock mass was developed based on a linear programming approach. The application runs on Apache Spark which enables it to run locally, on a computer cluster or on the Cloud. The program automatically maintains load balance among parallel processes and can be scaled up to meet computational demands without having to make any changes to the underlying source code. This enables the program to generate real-world scale block systems containing millions of blocks in minutes.

The second stage of this research effort focused on developing a new open-source Discrete Element Method (DEM) program capable of analyzing the kinematic response of fractured rock. The contact detection computations for DEM are also based on a linear programming approach such that similar logic and data structures can be used in both the block generation and DEM code, though the DEM code is written in C++. The program was validated against

analytical solutions as well as other numerical solutions and has been shown to accurately capture the kinematic response of three-dimensional polyhedral rock blocks.

The DEM formulation was then extended to perform coupled fluid-solid interaction analyses by coupling it with the weakly compressible Lattice Boltzmann Method (LBM). A new algorithm, which extends the partially saturated approach, was developed to consider three-dimensional convex polyhedra moving through the fluid domain. The algorithm uses both linear programming and simplex integration for the coupling process. The LBM code and the new fluid-solid coupling algorithm were validated against experimental data and the capabilities of the new coupled DEM-LBM implementation were explored by evaluating the performance of the program in simulating several different problems involving fluid-solid interaction. The results show that the program is able to accurately capture the interaction between polyhedral rock blocks and fluid; however, further performance improvements are necessary to simulate realistic, field scale problems. Particularly, adaptive mesh refinement and multigrid methods implemented in a parallel computing environment will be essential for capturing the highly computationally intensive and multiscale nature of rock-fluid interaction.

In memory of Major Splinter, Captain Soelzer, Sergeant Biskie and Specialist Creamean

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I am deeply grateful to my doctoral advisor, Professor Nicholas Sitar, for giving me the opportunity to pursue my graduate studies. His guidance and support have made this experience the most professionally rewarding I've had. Nick gave me the freedom to decide how I would like to pursue the research while always being available to provide input and discuss what the best path forward was. His mentorship has been invaluable and it always felt as though Nick was not only interested in the research, but also my personal development and well-being. The numerous international trips are also well appreciated!

It is my pleasure to thank Professor Robert Kayen for donating his time and expertise to do LIDAR and drone surveys at Spaulding Dam. Thank you for trusting me with your extremely expensive toys and always having time to talk. Additionally, I would like to thank Professor Kenichi Soga for including me in the large-scale DEM-LBM workshop and giving me the opportunity to meet and interact with so many talented researchers.

Prior to attending university, I had the honor to serve with soldiers from Company C, 5th Engineer Battalion, 1st Engineer Brigade. This was a formative experience and I thank my comrades for teaching me to love the suck—a necessity in both the Army and research. In particular, I would like to express my gratitude to Staff Sergeant Gregory Rogoff who is not only an excellent leader, but a loyal friend. Thank you for all your support during the good and the bad times, both during and after our service.

I would like to thank all my colleagues and friends at Berkeley. In particular, Mike George for the helpful discussions on rock scour and his phenomenal experimental work as well as the opportunity to help with field work at Spaulding Dam. Also, Robert Lanzafame, Julien Waeber and Nathaniel Wagner for the great conversations accompanied by coffee and Top Dogs. I would also like to acknowledge my ski camping buddies, Matthew Zahr and Devon Laduzinsky, with whom a good mix of cold bourbon and hot beans always made for excellent times. I would like to thank John "Jack" Kolb for his guidance in code development and the many runs in the Berkeley hills and Marin County where he never failed to illustrate his superior fitness. Additionally, I would like to thank Oliver Ricard and Krishna Kumar, both extraordinarily busy individuals, for always having time to answer coding-related questions.

This research was supported in part by the National Science Foundation, under Grant No. CMMI-1363354 "Dynamics of Water-Rock Interaction in Rock Scour", and the Edward G. Cahill and John R. Cahill Endowed Chair funds. I am grateful to these funding sources for making this research possible.

Finally, I would like to thank my family. Specifically my mom, dad and brothers for their love and support. Their encouragement and guidance throughout my life, even when they are on the other side of the world, have made this work possible. The biggest thank-you is reserved for my wonderful wife, Grace. During the time it took me to complete this degree, she managed to give birth to two children while still working full-time to support the family. Her patience with and willingness to tolerate my wildly swinging moods is mystifying. I am forever grateful for her unwavering support during this entire process. Lastly, I would like to thank my children, Jake and Claire. They are a constant source of joy and I look forward to

all the adventures we will have together before they realize their dad is a nerd that they'd rather not hang out with.

Chapter 1

Introduction

Erosion of rock by water is fundamental in the natural evolution of the landscape; however, the erosive capacity of water is equally applied to engineered structures. Dam spillways, bridge abutments and tunnels are all subject to water erosion and damage to the structures' foundations or the structures themselves can cost millions of dollars to repair and, in the worst case, cause loss of life.

A sobering example of this is Oroville Dam. Located in the foothills of the Sierra Nevada in Northern California, the dam sits along the Feather River east of the city of Oroville, California. Between 6 to 10 February, 2017, approximately 12.8 inches of rain fell in the Feather River Basin [18], causing much greater than predicted inflows into Lake Oroville. This led to increased discharge over the service spillway which first showed signs of damage on February 7. Spillway gates were closed to allow for damage inspection, the reservoir level increasing all the while. On February 11, water flowed over the emergency spillway for the first time in the history of the structure. Flow channels were quickly eroded into the natural material and aggressively headcut towards the spillway crest, as shown Figure 1.1. This is what triggered the emergency evacuation of approximately 188,000 residents downstream of the dam. The service spillway gates were re-opened on February 12 and water was allowed to spill periodically up until mid-May to manage reservoir levels. Figure 1.2 shows the significant damage caused to the service spillway during this period of time.

Apart from the near-miss in terms of the dam failure, Oroville Dam also highlights the changing risk associated with large infrastructure located in the near vicinity of expanding urban environments. The hazard associated with dam failure is much different today than it was when the structure was first built and this change is not unique to Oroville Dam. This incident has made dam owners painfully aware of the risk that their structures pose to surrounding communities and the need to re-evaluate the safety of these structures considering the evolution of the hazard given a dam failure. Re-evaluating these structures requires new tools that can evaluate the possible failure mechanisms that previous analyses were not able to capture. As outlined by George [28], overly simplifying the three-dimensional nature of fractured rock for scour analyses can be problematic and may miss the influence of geologic structure on rock mass erodibility.

Figure 1.1: Erosion of the emergency spillway at Oroville Dam in Northern California, 13 February 2017 [82]

To this end, this research builds on the experimental and analytical work by George and Sitar [[30],[28], [26], [27], [29]] specifically aiming to develop the ability to numerically evaluate rock-water interaction. The focus herein is on producing simulation techniques capable of considering three-dimensional polyhedral blocks interacting with and moving through fluid. The natural jointing within rock leads to fractured rock masses comprised of irregularly shaped polyhedral rock blocks and the orientation of these blocks relative to the geometry of the slope governs their kinematic response [36]. Therefore, it is essential that the model for the solid phase is capable of accurately representing the rock block shapes and that the coupling between the fluid and solid models allows the shape of the blocks to be accurately captured in the fluid solution.

1.1 Modeling Fractured Rock

The mechanical behavior of fractured rock is governed by the discontinuities within the rock mass: displacements occur along fractures and joints and the strength of these discontinuities is much lower than that of the surrounding competent rock. Additionally, the orientation of the discontinuities relative to slope geometry dictates the kinematic response of the rock mass and accurate representation of the discontinuities is essential in identifying the kinematically correct failure mode [95]. Given these observations, in order to correctly model fractured rock requires 1.) A numerical technique that can consider the discontinues nature of the rock and, 2.) A three-dimensional rock mass model generated based on field

Figure 1.2: Scour damage to service spillway at Oroville Dam in Northern California, 26 February 2017 [58]

observations that accurately captures the jointing within the rock and its orientation relative to slope geometry.

Modeling the mechanical behavior of the rock mass requires integrating the partial differential equations describing its motion—essentially Newton's second law of motion. Continuum techniques such as the Finite Element Method (FEM) [121] and Finite Difference Method (FDM) [65] provide a means to numerically evaluate partial differential equations and have achieved great success in a wide scope of problems. However, a fundamental assumption in these methods is that the medium being modeled is continuous. Techniques do exist to account for discontinuities [[37], [31], [111], [87], [22]], but the use-case for these methods is aimed at problems where the domain is primarily continuous with only a few discontinuities. To address this shortcoming, numerical methods have been developed that explicitly consider the interaction between discrete particles in a discontinuous system. Specifically, the Discrete Element Method (DEM) [[11], [13], [41]] and Discontinuous Deformation Analysis (DDA) [[93], [92]] have gained popularity within geomechanics to model the particulate nature of geomaterials. DDA parallels FEM in that the blocks are described through a system of equations where each element is a real isolated block [92]. For dynamic analyses, the system of equations is solved implicitly. DEM, on the other hand, considers the motion of each particle individually using explicit time integration. The motion of the particle is described by Newton's second law while the interaction of the particle with its neighbors is captured through explicitly considering particle-particle forces based on a contact law.

The explicit, decoupled nature of DEM makes it attractive for parallel computations since only information on the particles' immediate neighbors is required for calculating contact forces after which the motion of each block can be updated independently. Additionally, the extension of DEM to three dimensions is relatively straightforward. Therefore, DEM was selected to model the mechanical behavior of rock.

A three-dimensional rock mass model is required as an input for the mechanical simulations. Generation of this model can loosely be thought of as mesh generation for discontinuous methods; however, the "mesh" is dictated by the physical orientation of the jointing and fractures within the rock mass. The model is generated based on field observations, specifically, the number of joint sets with their strike, dip and spacing as well as their orientation relative to slope geometry. A challenge in generating a fractured rock mass model is that all observations are made on the surface and the internal structure of the rock is inferred based on these surface manifestations. This is a fundamental topic in rock mechanics and many algorithms have been developed to generate fracture networks and subdivide the rock mass into individual blocks [[109], [108], [45], [53], [68], [56], [8]]. One particular class of block cutting algorithms uses linear programming to generate the individual rock blocks [8]. This method simplifies the implementation of the block cutting algorithm in terms of computer code development since the blocks and discontinuities are described simply as linear inequalities. Additionally, the contact detection problem for the mechanical model can be solved using linear programming [7]. This means the blocks generated using the linear programming approach will already be in a format that can be fed into the mechanical model and much of the code developed for the fractured rock mass generation can be re-purposed in the mechanical model contact detection. With this in mind, the fractured rock mass generated as input to the mechanical model was generated using a linear programming approach.

1.2 Rock-Water Interaction

Modeling the interaction between the blocky rock mass and the water flowing over and through it requires coupling between the numerical models for the solid and the fluid phases. This means the hydrodynamic forces and moments exerted on the particles need to be accounted for when integrating the equations of motion for the solid phase while the effect of the particles in and moving through the fluid also needs to be incorporated into the fluid solver. Simulations that capture this interaction generally follow two approaches. The first approach incorporates fluid-solid interaction based on a locally-averaged interaction between the two phases [[2], [105], [112], [104], [71]] while the second approach approach directly simulates hydrodynamic forces on the solid particles [[78], [48], [81], [16]]. In the locally-averaged approach, the fluid-solid coupling is done by averaging the interactions over a representative volume and all particles within a local region experience the same hydrodynamic forces. This makes the method less computationally expensive compared to direct simulation since the number of solid particles is greater than the number of fluid cells. This approach may be appropriate in certain applications, but it does not offer sufficient resolution when trying to

establish the hydrodynamic interaction for individual blocks. Direct simulation of the fluid-solid interaction attempts to overcome this shortcoming by having a much higher density fluid mesh compared to the number of solid particles. This approach is able to capture the variation in hydrodynamic forces on individual particles, but it does come at a much higher computational cost.

Since the individual behavior of rock blocks is important for the kinematic response of fractured rock, the direct simulation of rock-fluid interaction is necessary. Methods conventionally used in computational fluid dynamics (CFD), such as FEM [120] and the Finite Volume Method (FVM) [66] are able to directly solve the hydrodynamic forces and moments acting on solid particles and, in the case of FEM, can accommodate higher order methods relatively easily. Both of these methods are able to handle the complex shapes of rock blocks, but require remeshing if the blocks are allowed to move through the fluid mesh. This can be computationally prohibitively expensive. In comparison, the Lattice Boltzmann Method (LBM) [[70], [97]] allows for fluid-solid coupling in a less computationally expensive fashion. Additionally, LBM is localized in its formulation, making it amenable to adaptive remeshing and parallel computing—a necessity for direct simulation of fluid-solid interaction. Consequently, LBM was selected to model rock-water interaction.

1.3 Research Outline

The inherent geometric complexity and discontinuous nature of fractured rock as well the high computational cost of direct modeling of rock-fluid interaction necessitate numerical methods that can accurately and efficiently simulate the hydrodynamics of rock erosion. In terms of modeling the solid phase, DEM is well suited to describe the kinematic response of fractured rock. The coupling process between DEM and LBM when the rock blocks are allowed to move through the fluid mesh is comparatively simpler and less computationally expensive than other CFD methods. The inherently localized nature of both DEM and LBM makes them attractive candidates for parallel computing.

To this end, a three-dimensional, open-source program to produce the fractured rock mass was developed based on a linear programming approach as described in Chapter 2. Next, a three-dimensional DEM program for the analyses of blocky rock mass kinematics was developed and validated as shown in Chapter 3. Finally, the DEM program was expanded to perform a coupled solid-fluid interaction analysis using a weakly compressible, three-dimensional LBM formulation. A new coupling algorithm that is able to consider three-dimensional convex polyhedra moving through the LBM mesh was developed and validated as described in Chapter 4. The LBM computations were accelerated using the C++ library `Kokkos` [15] which allows for shared memory parallelism on both central processing units (CPUs) and graphics processing units (GPUs). The capabilities of the new coupled DEM-LBM implementation were explored by evaluating the performance of the program in modeling different types of problems involving solid-fluid interaction as shown in Chapter 5. Chapter 6 concludes with a summary and opportunities and suggestions for future research.

Chapter 2

Fractured Rock Mass Generation

The contents of this chapter are primarily from a journal article published in Computers and Geotechnics in May of 2017 by Michael Gardner, John Kolb and Nicholas Sitar entitled "Parallel and Scalable Block System Generation" [24].

2.1 Introduction

Generating a realistic representation of the fractured rock mass is the first step in many analyses—whether evaluating the interaction of fractured rock with water, evaluating the mechanical behavior of the rock mass itself or considering the fracture network within the rock for seepage analyses. The orientation and spacing of the joint sets, how persistent they are and how they intersect each other will fundamentally affect how the rock mass behaves.

This is not a new topic and many researchers have developed algorithms to generate fracture networks and subdivide the rock mass into individual blocks. Warburton [[108], [109]] presents a methodology for generating a blocky rock mass through sequential subdivision. His method uses a three-level data structure for describing the blocks—the faces, edges and vertices describe the geometry of the generated blocks. The scheme developed by Heliot [45] can generate a rock mass containing non-convex blocks by representing blocks as an assemblage of smaller, convex blocks. This method uses a two level data structure containing vertices and faces to represent the individual blocks. Ikegawa and Hudson [53] developed the directed body approach in which all discontinuities are introduced simultaneously. The individual blocks are then extracted from the vertices edges and faces. Additionally, other researchers [[68], [56]] have developed methods that are able to deal with more complex geometry using principles of combinational topology; however, these techniques come at the expense of significant "bookkeeping". Recently, Boon et al [8] introduced an algorithm based entirely on linear programming. Instead of explicitly calculating the face, edges and vertices where discontinuities intersect, the problem is cast as a linear optimization. This makes it possible to represent the blocks using a single level data structure only describing the faces that comprise the block. The simplicity and efficiency of this method makes it an

attractive candidate for large scale parallel computations. The algorithm itself is entirely decoupled—once a block has been subdivided into two new blocks, the further subdivision of these blocks can proceed independently. Consequently, this method is naturally parallel and multiple cuts can be made simultaneously without the need to share information among processes.

2.2 Apache Spark

The sequential subdivision of the rock mass is an iterative process on the same set of data, making the parallel, open-source framework Apache Spark [113] an ideal platform for this approach. Spark can run on many platforms ranging from laptops and personal workstations with multicore processors to Cloud based computing platforms such as Amazon Elastic Cloud Compute (EC2). This scalability in computing power without having to make any changes to the underlying code allows for the analysis of very large problems requiring large amounts of memory and computing power.

The fundamental abstraction in Spark is Resilient Distributed Datasets (RDDs) [114] that allow it to keep large data sets in memory and perform computations in a fault tolerant manner. Spark is able to do iterative transformations on the data extremely quickly since it avoids writing to disk. Fault tolerance is achieved by tracking the lineage of RDDs—all operations applied to the RDD are represented through a lineage graph. When a new operation is applied to the RDD, a new link is added to the graph. Additionally, RDDs are evaluated lazily: only when a result is requested does Spark execute the transformations described by the lineage graph to actually materialize the current RDD. In this way, if a process unexpectedly fails the current state can be quickly reconstructed from the lineage contained in the graph.

For large scale problems, Spark clusters can be deployed on EC2 using Amazons Elastic MapReduce (EMR) framework, which automatically allocates and configures a cluster of EC2 instances to execute Spark tasks submitted by the user. Amazon EC2 falls under the greater umbrella of Cloud Computingapplications delivered as services over the Internet and the associated software and hard ware that provide those services [3]. Running large scale computations on the Cloud offers users several advantages. First, resources can be scaled on demand to meet the computing requirements of the problem at hand. Second, it is no longer necessary to invest large amounts of capital in computational hardware and the associated management and maintenance. Lastly, usage can be scaled up or down as needed so users only pay for what they use and only use what they need. This makes it possible for anyone to run large scale computations since it is no longer necessary to physically own a computer cluster.

2.2.1 SparkRocks

By taking advantage of Sparks ability to run on any computer system and the scalability of Cloud Computing, a parallel block cutting program, `SparkRocks` [1] was developed. The program is capable of generating large numbers of blocks very quickly. The code is open source and the necessary inputs to generate a fractured rock mass are based on parameters that are obtained from field observations, allowing users to quickly translate field measurements into a three-dimensional model. The program was evaluated on different systems—a laptop, desktop workstation, and Amazon EC2—to illustrate its ability run on different platforms and to verify its scalability. Results show that approximately 8 million blocks can be generated in roughly 9 minutes when executing on Amazon EC2.

2.3 Block Cutting Algorithm

In the sequential subdivision approach based on linear programming optimization introduced by Boon et al. [8], each discontinuity is introduced individually and checked for intersection. If it intersects the block, two new blocks are generated. The process continues sequentially until all discontinuities have been introduced, yielding a representation of the fractured rock mass. Many block cutting algorithms require a significant amount of bookkeeping in terms of vertices, edges, faces and how all of these elements are connected. From an implementation perspective, this can be extremely tedious and may not be as robust in terms of floating point error. The linear programming optimization approach introduced by Boon et al. greatly simplifies how block cutting is implemented and how each block is represented in terms of data structure. Here, only a brief overview of this rock cutting algorithm is given since the details are presented in [8].

The orientations of joints in a fractured rock mass are described by strike and dip, as shown in Figure 2.1. The block cutting algorithm uses the normal vector of the plane containing the joint and the distance of that plane to some origin. The strike and dip define the normal vector of the joint. The distance of the joint plane from the origin is determined by projecting a vector connecting the origin to a point in the joint plane onto the normal vector. The global $+x$, $+y$ and $+z$ axes are oriented North, West and upward.

Using only the joint normal and distance, it is possible to completely describe the volume defining a polyhedral block. The volume defining a block bounded by N planes is described by the following equation:

$$a_i x + b_i y + c_i z \leq d_i,\ i = 1, \ldots, N \tag{2.1}$$

The coefficients (a_i, b_i, c_i) represent the normal vector to the i^{th} plane bounding the block and d_i is the distance of that plane from some local origin. In vector notation this becomes:

$$\boldsymbol{a}_i^T \boldsymbol{x} - d_i \leq 0,\ i = 1, \ldots, N \tag{2.2}$$

[1] Available at https://github.com/cb-geo/spark-rocks

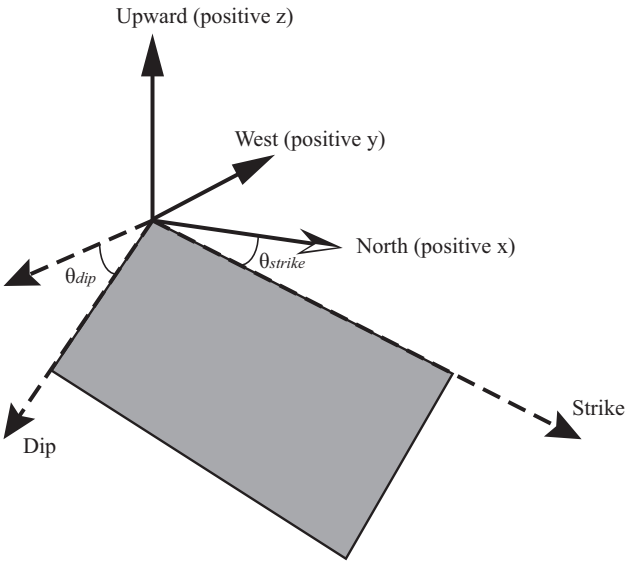


Figure 2.1: Description of strike and dip, illustrating relationship to global coordinates

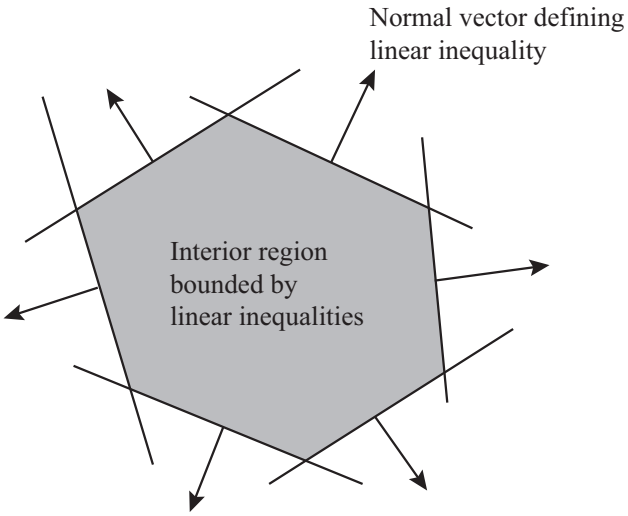


Figure 2.2: Volume bounded by set of inequalities. The shaded region represents the block that is defined by this set of linear inequalities (after ([8]).

In order to subdivide a block, it is necessary to establish whether the block is intersected by the discontinuity being considered. The novelty in the algorithm presented in [8] is recasting this problem as a linear program:

$$\begin{aligned} &\text{minimize } s \\ &\boldsymbol{a}_i^T \boldsymbol{x} - d_i \leq s, \ i = 1, ..., N \\ &\boldsymbol{a}_{\text{new}}^T \boldsymbol{x} - d_{\text{new}} = 0 \end{aligned} \tag{2.3}$$

Here N represents the number of planes that define the block and the discontinuity being considered is represented by the equality. If $s < 0$, there is an intersection and the parent block is split into two child blocks. The child blocks inherit all the parent block's planes as well as the intersecting discontinuity with opposite signs for the discontinuity normal vector for each child block.

As the subdivision continues, some of the discontinuities may become geometrically redundant. It is not necessary to remove these redundancies after each intersection check; instead they can be removed at a later time as discussed in Section 2.4.2.3. Again, this can be done by solving a linear program:

$$\begin{aligned} &\text{maximize } \boldsymbol{c}^T\boldsymbol{x} \\ &\boldsymbol{a}_i^T\boldsymbol{x} \leq d_i,\ i = 1, ..., N \end{aligned} \tag{2.4}$$

Here, $\boldsymbol{c}$ is the normal vector specific to the discontinuity being checked for redundancy and with associated distance d. If $|\boldsymbol{c}^T\boldsymbol{x} - d| < \varepsilon$ the discontinuity is not redundant, where ε represents a numerical tolerance close to zero.

Additionally, `SparkRocks` takes advantage of two major optimizations to the block cutting process that are presented in [8]. The first optimization draws on an idea common to contact detection in particle methods (for example, see [74]): the complex geometry of the polyhedral blocks is enclosed in a simpler shape, in this case a bounding sphere. This enables a simple and fast check for intersection to determine if a more thorough but computationally expensive check is necessary. The second optimization is to control the size and aspect ratio of the blocks that are generated during the slicing process. Unrealistically small or thin blocks can contaminate the generated blocks, leading to undesirable side effects in the subsequent analyses that use the fractured rock mass as an input. Both of these optimizations involve the construction and solution of linear programs.

2.4 Implementation on Spark

2.4.1 Translation Into a Parallel Problem

As already stated, the serial approach to cutting blocks can be modified to run in parallel. Once two child blocks are cut from their parent by a particular joint, they can be treated independently for the remainder of the block cutting process. In other words, one child's intersection with subsequently introduced joints and future subdivisions into further child blocks has no effect on the subdivision matters of its sibling. This gives rise to a tree structure of relationships between blocks, depicted in Figure 2.3. Therefore, while processing each joint in a rock mass, the joint's intersection with each block cut so far can be computed independently. We take advantage of this property to construct and solve the linear programs described in the previous section in parallel and independently on each processor. It is important to note that the tree structure described is not unique to [8], and any other block cutting algorithm with this property would lend itself to parallelization.

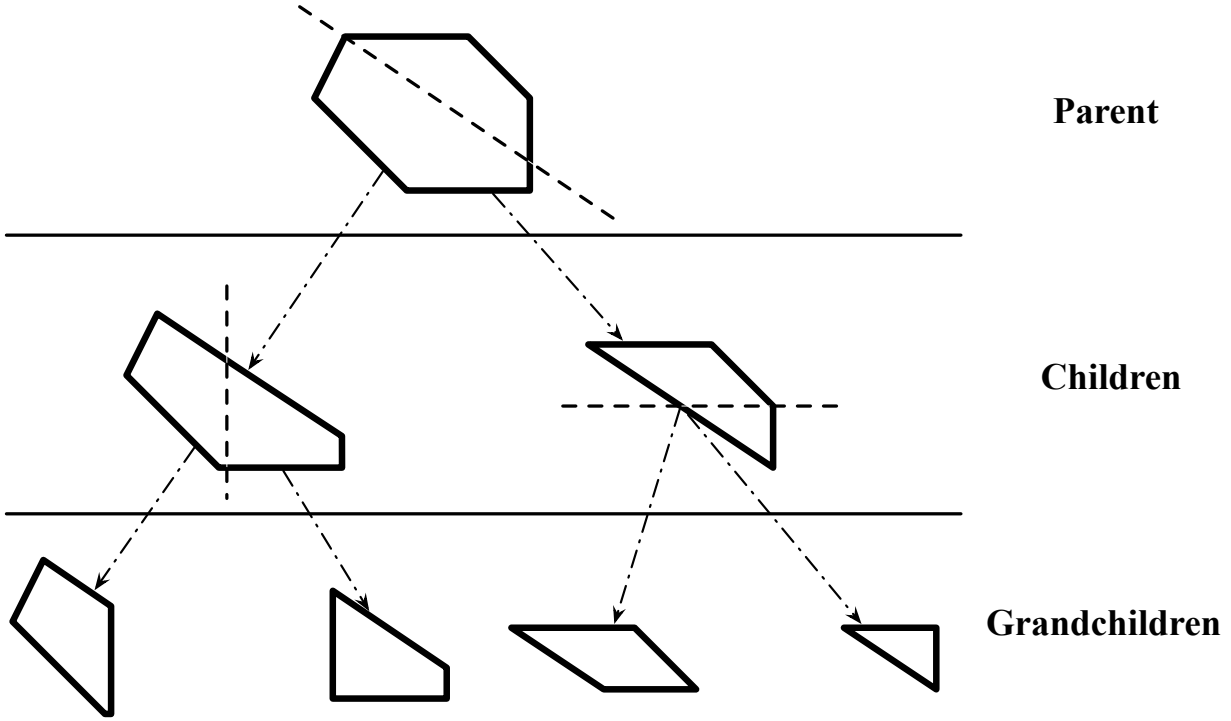


Figure 2.3: Three Generations of Blocks Cut from a Common Ancestor

In practice, it is necessary to effectively distribute work to the multiple central processing unit (CPU) cores and nodes that are available. By expressing the current set of blocks cut from a rock mass as an RDD in the Spark context, it is possible to seamlessly perform parallel operations on these blocks and scale the associated computation to different quantities of CPU cores and nodes without changing any of the underlying rock slicing logic. To split up responsibilities for all of the required rock slicing, we select a small subset of joints to break the overall rock volume into a group of initial blocks of roughly equal volume. Each block, and all of that block's descendants, are then processed independently as illustrated from a high level in Figure 2.4.

2.4.2 Implementation

A direct translation of the method described in [8] into code does not lead to an efficient parallel implementation. This section describes important features and refinements necessary to achieve good performance when dealing with problems at a large scale.

2.4.2.1 Load Balance

Load balance among parallel processes as well as minimizing communication between them is of primary concern in achieving efficient parallel execution. A solution that achieves

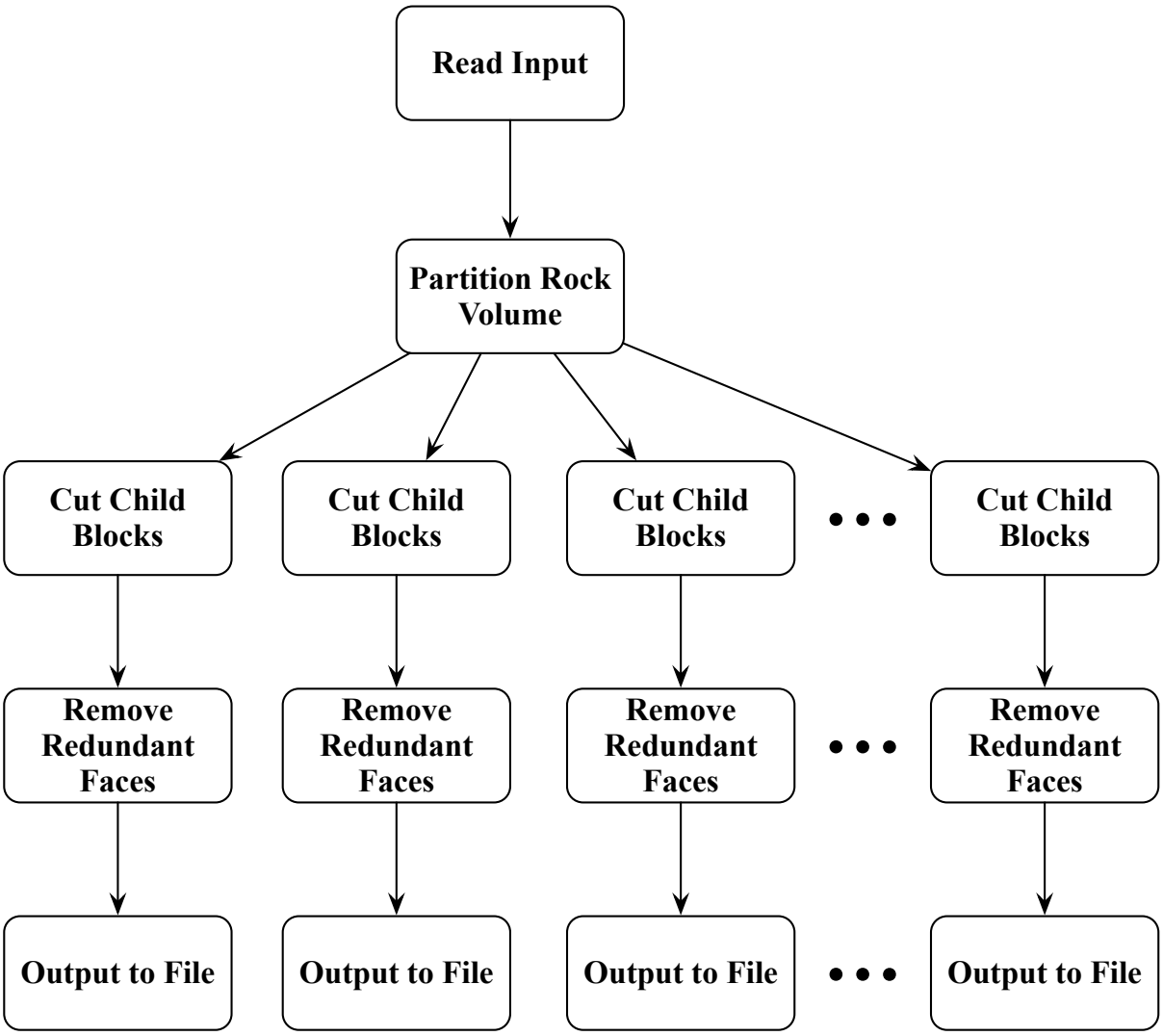


Figure 2.4: Steps in the Parallel Rock Slicing Process

good load balance may not necessarily feature a reasonable level of communication overhead, while another solution may have low communication overhead but poor balancing of work among parallel processes. It is therefore necessary to find a strategy that achieves a balance between these two demands by taking into account the characteristics of the underlying framework on which computations are done.

In this case, the initial thought was to focus on load balance. Artificial joints were introduced to divide the rock volume into equal pieces so that near-perfect load balance is achieved between parallel processes. However, in order to remove the artificial joints at the end of the slicing process, all blocks sharing an artificial joint must be recombined in order to remove that artificial joint from the final rock mass. This involves an exchange of blocks

over the network among all nodes, which induces a high amount of communication overhead that slows down the overall rock slicing process and offsets the gains achieved through load balancing. Currently Spark does not have the necessary communication primitives to directly manage communication which, in this case, leads to excessive communication to the point that the majority of the computation time is spent on the removal of artificial joints rather than the introduction of real joints.

Given these constraints, some of the load balance can be sacrificed in order to minimize communication. Since the block cutting process is entirely decoupled and can be done independently, once each node receives a portion of the initial rock volume it can complete the cutting process without communicating with other nodes. We exploit this by selecting joints from the input joint sets that divide the initial rock volume into approximately equal volumes. The blocks generated by cutting the initial rock volume with the selected joints are used to seed the initial RDD. This idea is illustrated in Figure 2.5. The different colors represent which portions of the rock mass were processed by which node. In this example four nodes were used. Each node processed one portion of approximately equal volume and the last, much smaller volume was processed by the node that completed its subdivision first.

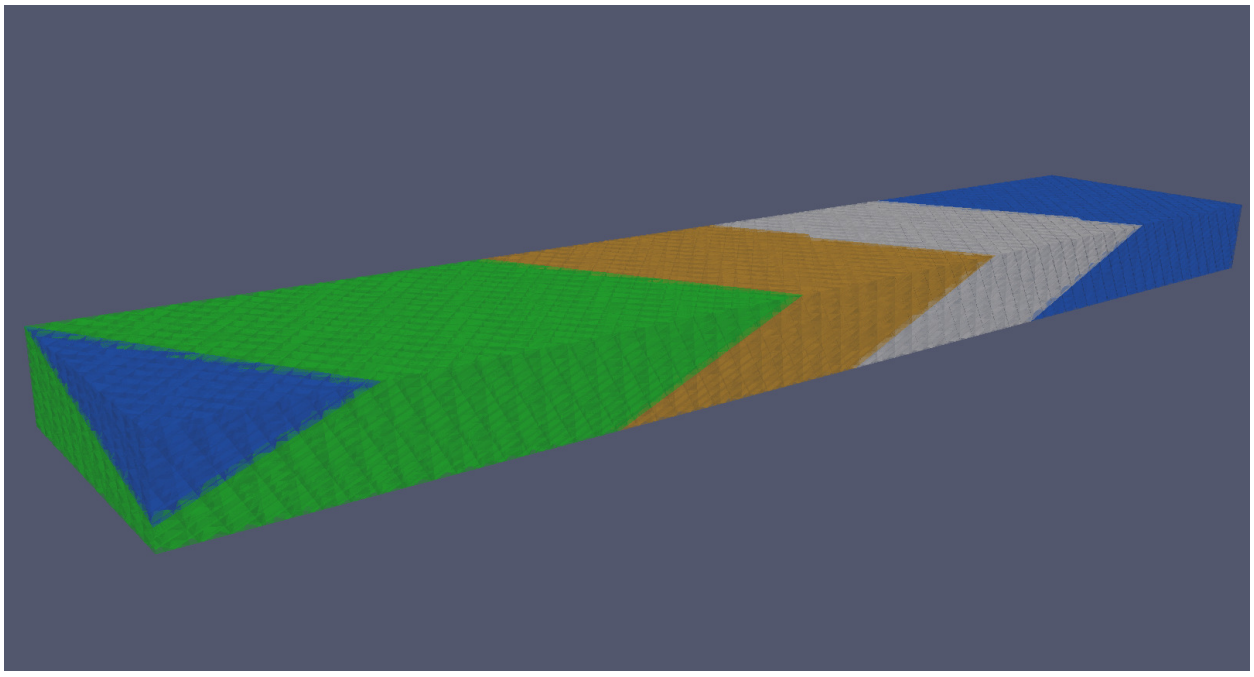


Figure 2.5: Example of load balance. Here, at least 4 partitions were requested. The different colors represent which pieces were processed by which node.

Since the joints used to generate the seed blocks are real joints taken from the input joint sets, it is not possible to achieve perfect load balance. In some instances it may not be possible to find adequate seed blocks from a single joint set, so it becomes necessary to

select joints from multiple joint sets. This complicates finding the exact same number of seed blocks as the number of nodes in the analysis. In most cases, more seed blocks are generated than what is requested. This is especially the case when selecting joints from multiple joint sets. Since Spark performs dynamic load balancing internally, having more seed blocks provides flexibility in managing and maintaining load balance.

2.4.2.2 Lineage

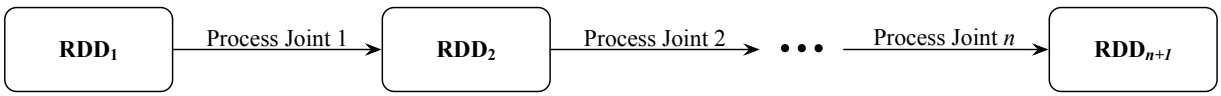
Spark internally tracks the transformations applied to each RDD in a lineage graph. This allows it to defer the materialization of an RDD until its contents are actually needed by traversing a path from a previously materialized RDD to the required RDD, applying the necessary transformations along the way. However, Spark fails when lineage chains in this graph grow too long. Specifically, this occurred in the initial version of the rock slicing code depicted in Figure 2.6a, which iterated through each joint in the rock mass, checked for intersections with any members of the current block RDD, and produced a new RDD in which any blocks intersecting the joint were cut into two child blocks. Thus, a new RDD was created for each joint, and a lineage chain formed with a length proportional to the total number joints, which becomes unwieldy when the number of joint is large.

We resolved this issue by taking advantage of Spark's `fold` primitive, as shown in Figure 2.6b. This operation individually examines each joint and creates an intermediate collection of blocks that are the cumulative result of processing all joints seen so far. The operation repeats until all joints have been processed, and only the final result is retained. In more detail, `fold` starts with an initial element (the seed blocks used for load balancing), and element i is produced by applying an operation to element $i-1$ and the next joint, which in this case is an intersection check and the necessary slicing of parent blocks into child blocks. Spark treats a `fold` as a single transformation and therefore a single link in the lineage chain. This replaces the original lineage chain, with a length proportional to the number of joints, with a lineage chain of length one.

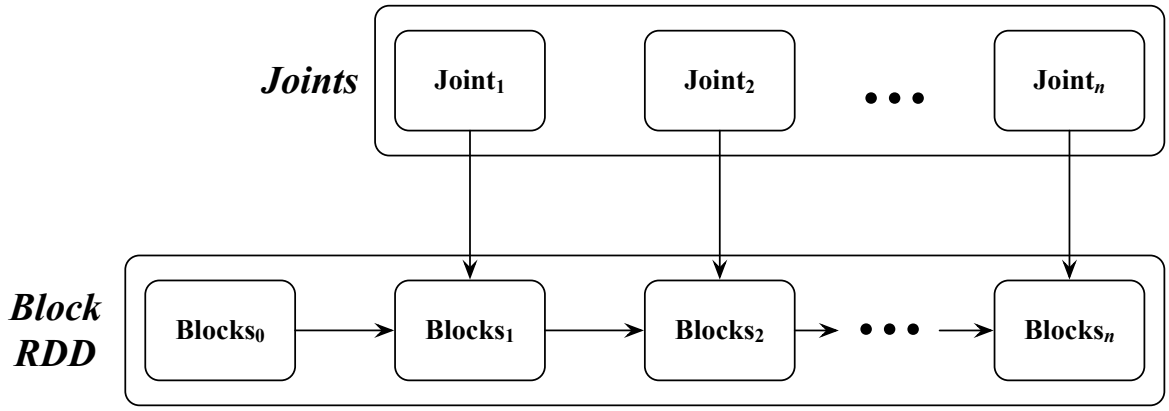
2.4.2.3 Redundant Faces

Two child blocks that are cut from a parent inherit all of the parent's faces as well as a face along the discontinuity that separates the blocks from each other. Many of these faces are geometrically redundant and can removed without compromising the integrity of the block. Boon et al. [8] advocate deferring the removal of geometrically redundant faces until all blocks have been cut. However, retaining a large number of redundant faces in blocks during the slicing process increases the size of the linear programs that must be solved when checking for intersection between blocks and joints. The increase in linear program size leads to degraded performance as more child blocks are cut from the initial rock volume.

To avoid this deterioration in performance, we periodically remove redundant faces during the block cutting process. The frequency at which this removal is performed represents a tradeoff. Removing redundant faces is somewhat expensive because it involves solving a



(a) Iteratively Creating RDDs



(b) Using `fold` to Process all Joints

Figure 2.6: Lineage Chains in Spark

linear program for each face of a block, so it cannot be done too often. We found that eliminating redundant faces for every 200 joints processed keeps the linear programs reasonably sized without adding excessive overhead from geometric redundancy checking.

An important consideration in this scheme is the fact that many, if not most, blocks will not change when additional joints are introduced. Therefore, examining these blocks is unnecessary work and a source of significant inefficiency. To address this problem, we index all joints by the order in which they are processed, i.e. the first joint checked for intersection against all blocks is assigned index 1, the second joint checked is assigned index 2, and so on. Each block is augmented to track its *generation*—the index of the joint that cut the block from its parent, and blocks that were not cut from their parents by any of the 200 most recently introduced joints are skipped.

2.5 Performance

Performance evaluation was done on Amazon EC2 with Amazon Elastic MapReduce (EMR). EMR can seamlessly configure a Spark cluster on EC2, which makes deploying applications written for Spark easy to run. All testing was done on compute-optimized instance types, which are specifically designed for compute-intensive HPC applications. An instance

in the context of EC2 is a single node. For example, a four node cluster comprises four instances. The different instance types give the user a choice in the hardware configuration of the nodes. Testing was done with `c3.xlarge`, `c3.4xlarge` and `c3.8xlarge` instances. These three instance types span a range of computational power and hardware configurations, shown in Table 2.1.

	`c3.xlarge`	`c3.4xlarge`	`c3.8xlarge`
vCPU	4	16	32
Memory (GB)	7.5	30	60
SSD Storage (GB)	2 x 40	2 x 160	2 x 320

Table 2.1: Amazon EC2 Instance Types Used in Testing

In order to maintain control over the exact number of blocks that were generated, the input rock volume consisted of a rectangular prism and input joints were defined such that they divide the rock volume into a specific number of cubes. This allowed easy interpretation of how well `SparkRocks` scales.

2.5.1 Partitioning

Load balance among the different nodes is maintained by partitioning the initial rock mass into approximately equal volumes, as described in Section 2.4.2.1. The number of partitions - seed blocks - used in block cutting has a significant impact on the efficiency. Figures 2.7 and 2.8 show the total elapsed times for the three different instance types with 4,000 blocks and 32,000 blocks per node, respectively. Instances with 64,000 blocks per node showed similar trends. The most important result is that efficiency is highly sensitive to the number of initial partitions used to seed the RDD. Seeding the RDD with more partitions gives Spark more freedom in managing parallel execution, as indicated earlier. This trend is observed independent of the number of nodes. Each node can execute computations in parallel locally; however, if it receives only a single partition, computations will be serial as demonstrated by the runs executed on a single node.

When more than one node is used with too few partitions, Spark cannot effectively share the computational load across all members of the cluster, and some nodes end up doing much more work than others. By seeding the RDD with more partitions initially, each node will receive many seed blocks. This allows Spark to locally exploit parallelism and greatly increase efficiency. This is seen more clearly for tests with more blocks, such as shown in Figure 2.8 where execution times for large node counts are slower than smaller node counts for the same number of partitions.

However, when the input data set is too small, larger clusters perform poorer regardless of the number of partitions. For example, when each node only has 4,000 blocks, as shown in Figure 2.7, the data set is too small to benefit from the greater computational power

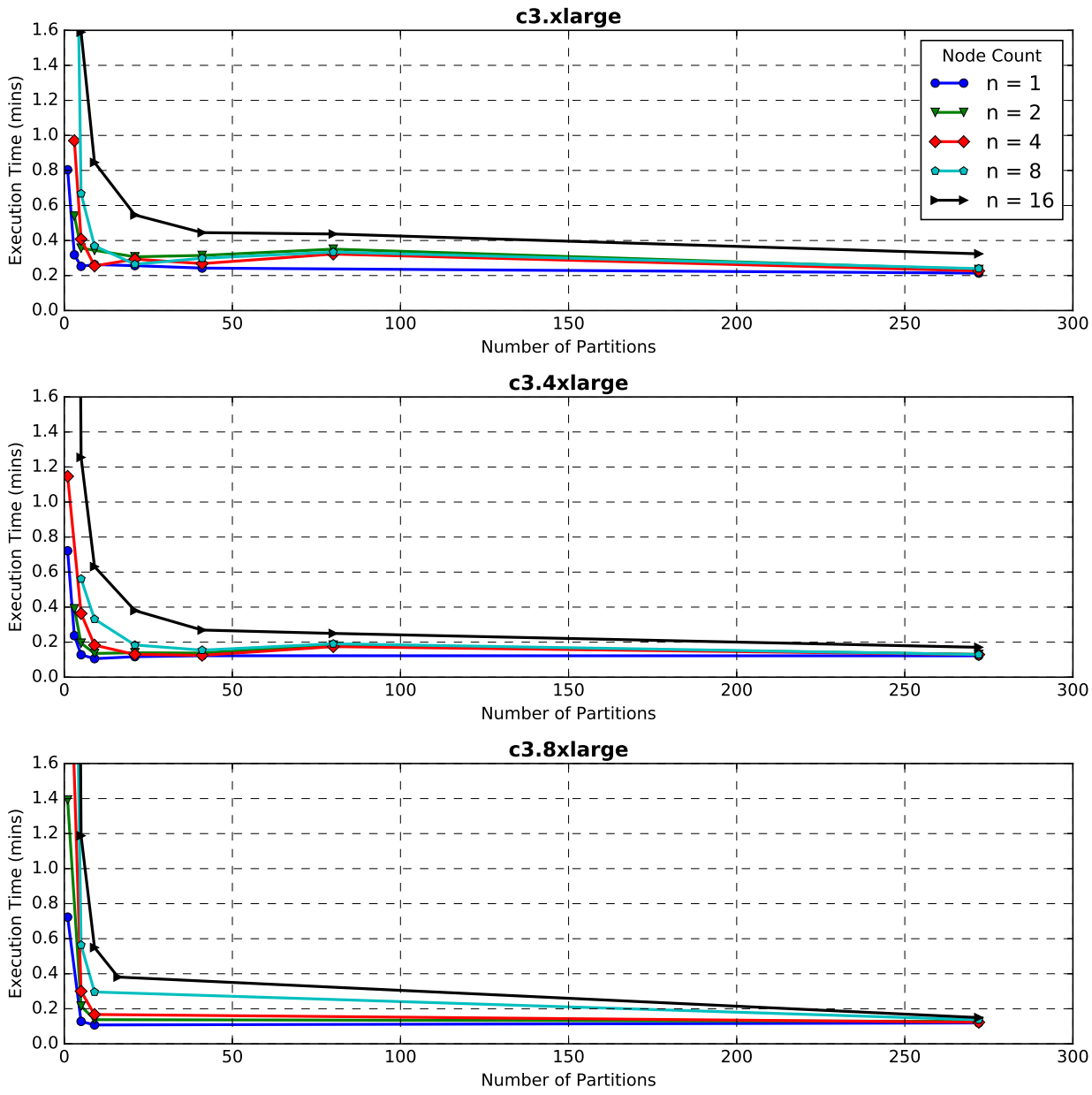


Figure 2.7: Execution Time vs. Number of Initial Partitions in the Rock Volume - 4,000 Blocks per Node

of larger clusters. The communication overhead required to manage more nodes dominates total execution time.

Interestingly, in some instances, there is a slight increase in execution time for larger partition counts. Apparently, with too many partitions the cost of communication among the nodes begins to outweigh the load balancing benefits, leading to higher execution times.

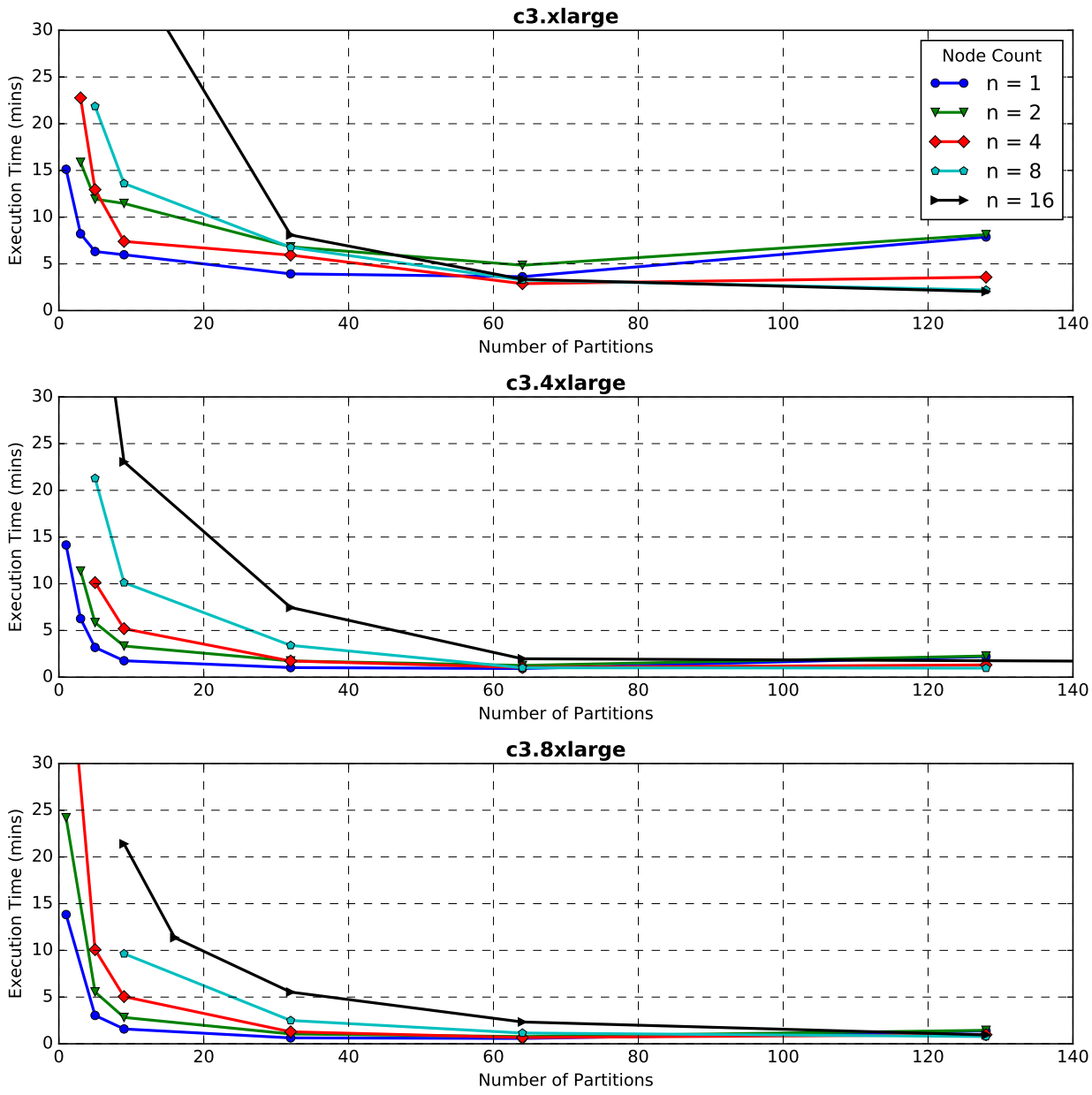


Figure 2.8: Execution Time vs. Number of Initial Partitions in the Rock Volume - 32,000 Blocks per Node

However, as can be seen, great speedup is attainable even if the most optimal partition count is not selected.

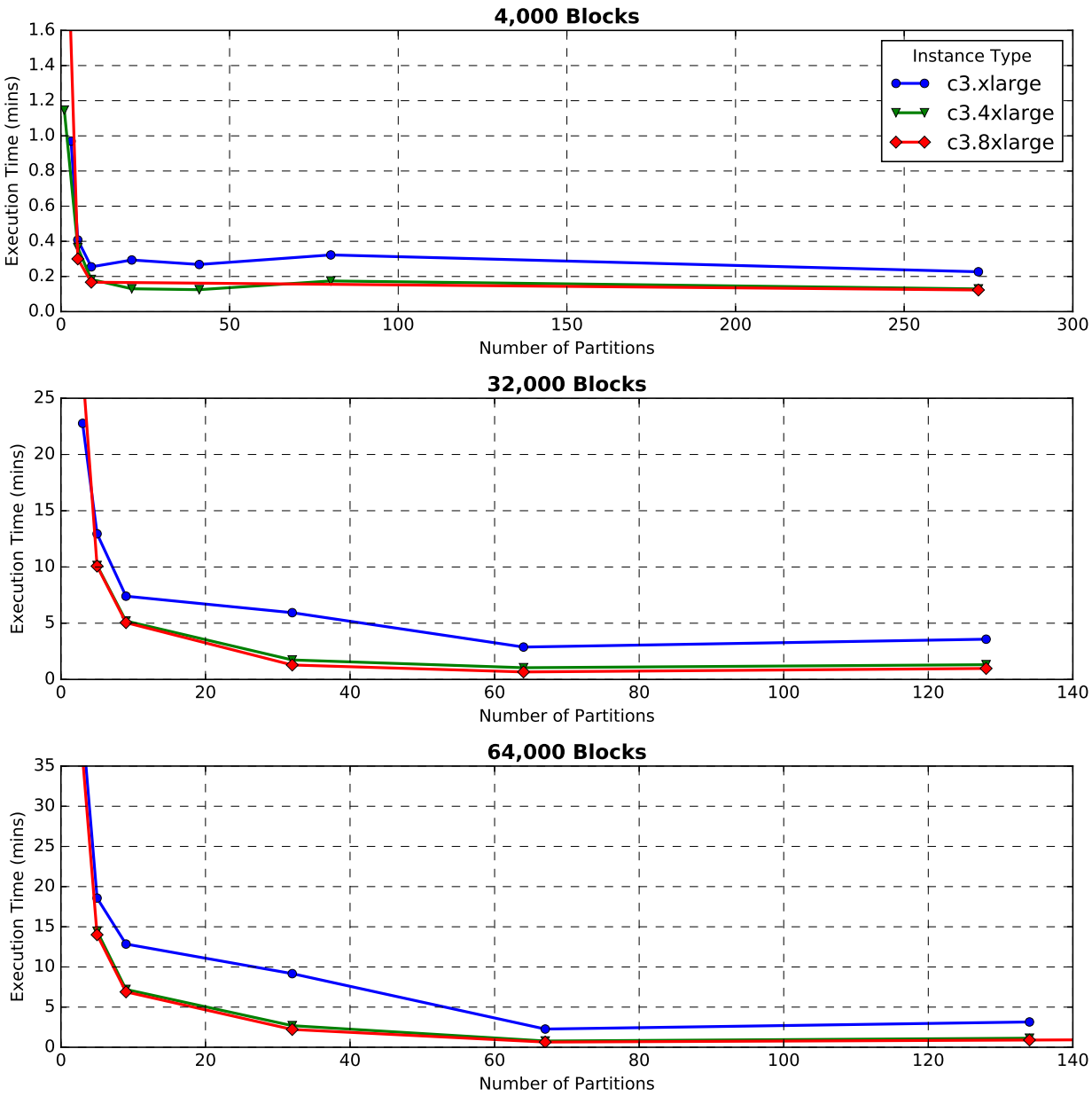


Figure 2.9: Execution Time vs. Number of Initial Partitions in the Rock Volume on a Four-Node Cluster

2.5.2 Instance Type

Figure 2.9 shows the total execution time of the rock slicing process on a four-node cluster for the three EC2 instance types and for three different problem sizes (4,000 blocks per node, 32,000 blocks per node, and 64,000 blocks per node). Clusters of two, eight, and sixteen nodes exhibited similar results. The least powerful instance, `c3.xlarge`, is affected

by small initial partition counts far more than the other types. A low partition count prevents all nodes in the cluster from fully participating, which accentuates the disparity in computational power between the `c3.xlarge` and the other instance types. As the partition count increases, the different instance types begin to yield more comparable performance, although `c3.xlarge` clusters still remain noticeably worse than the alternatives. Interestingly, the `c3.4xlarge` and `c3.8xlarge` demonstrate very similar performance characteristics, not just at high partition counts but for all partition counts. This implies that there are diminishing returns to running a well-tuned deployment on more powerful EC2 nodes, and this has important consequences for users seeking to perform large scale rock slicing on the Cloud. While Amazon's price for a `c3.8xlarge` instance is double that of a `c3.4xlarge` instance, one can use the latter without suffering a compromise in performance.

2.5.3 Weak Scaling

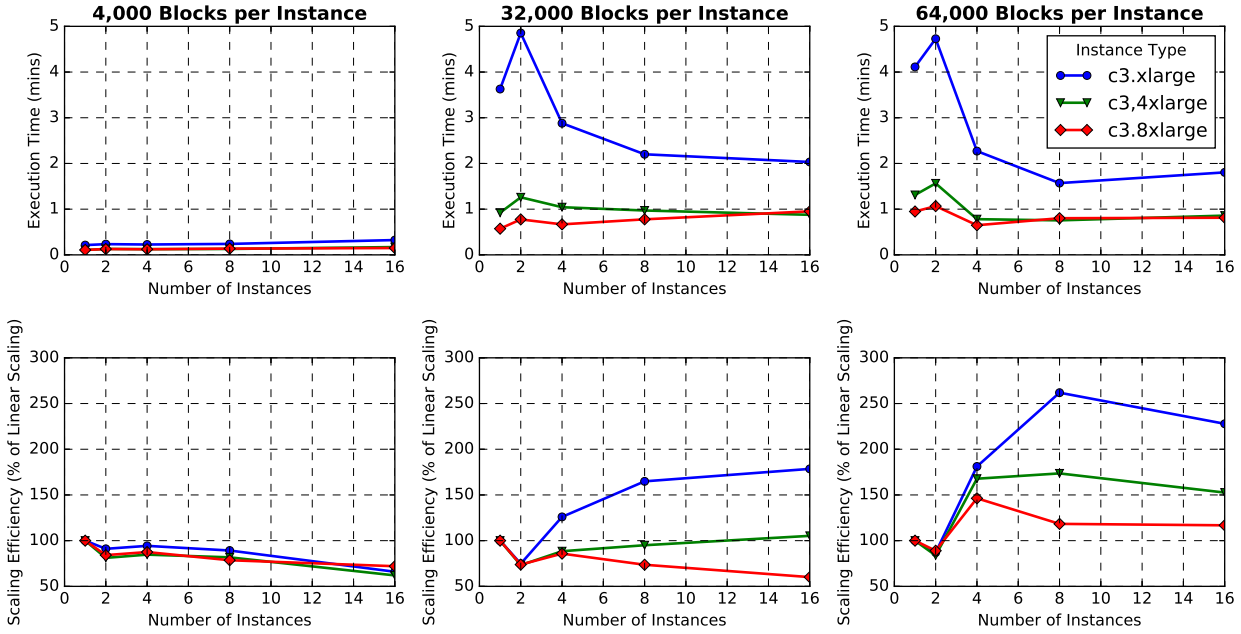


Figure 2.10: Execution time and scaling efficiency as cluster size increases. All experiments shown here used 64 partitions for all cluster sizes.

The *weak scaling* of a parallel program is its ability to maintain a constant level of efficiency while increasing the number of nodes involved in its computations. The problem size per node is kept constant, so each node performs the same amount of work as new nodes are added. In the ideal case, the execution time should remain constant as the number of nodes increases. To test the weak scaling capabilities of `SparkRocks`, we performed rock slicing on clusters of increasing size while proportionally increasing the total number of blocks that are sliced, e.g. a cluster with twice as many nodes slices twice as many

blocks. The relevant results are included in Figure 2.10 in terms of total execution time and scaling efficiency. When processing 4,000 blocks per node, `SparkRocks` demonstrates good weak scaling behavior, although it can be argued that there are not enough blocks in these experiments to seriously challenge the system's scaling abilities. Total execution time slowly increases as cluster size increases, probably due to the additional communication costs that are introduced by adding more nodes. Again, `c3.4xlarge` clusters achieve performance that is comparable to that of `c3.8xlarge` clusters.

When processing 32,000 and 64,000 blocks per node, the results become more complicated. As with 4,000 blocks per node, there is only a small performance difference between `c3.8xlarge` clusters and `c3.4xlarge` clusters, particularly at larger cluster sizes. Moreover, the performance gap between these two instance types and the `c3.xlarge` clusters also decreases as clusters become larger. Larger cluster sizes are therefore able to mask some of the differences in the capabilities of the underlying hardware. Diminishing scaling returns begin to appear at the larger cluster sizes, where execution time either decreases very slightly or increases. This is probably because communication costs start to become the dominant factor in scaling behavior, as is typical for parallel computing applications.

Figure 2.10 also illustrates a pattern in which execution time *decreases* in certain places as cluster size increases, e.g. when moving from two to four nodes. In some sense, this is better than "perfect" scaling where execution time remains constant as the number of nodes increases. This behavior is particularly difficult to analyze because Spark gives the user little control over how their jobs are executed on the underlying cluster of machines. Spark divides a job into a group of tasks, each of which is completed by an *executor* – an abstraction for an independent unit of processing. On Amazon's Elastic MapReduce platform, Spark by default dynamically assigns tasks to executors and increases or decreases the number of executors devoted to a job based on internal heuristics. The improved performance when increasing cluster size is likely due to the fact that with more machines, and therefore with more blocks to partition among these machines, Spark has more freedom to balance load across the cluster and is consequently able to achieve better execution times.

2.5.4 Strong Scaling

Strong scaling is measure of the speedup efficiency when increasing the number of nodes for a fixed problem size - using more nodes should yield shorter execution times. Based on the above mentioned results, strong scaling tests were only performed using `c3.4xlarge` and `c3.8xlarge`. These instance types clearly outperformed `c3.xlarge` and would be reasonable to use when attempting larger analyses. Figure 2.11 shows the results of these tests, both in terms of execution time and scaling efficiency. When comparing the execution times, it is clear that both instances perform equally well with more nodes. The largest difference in execution times is seen when using 1 and 2 nodes. This makes sense since the problem size is getting sufficiently large to accentuate the difference in computational power between `c3.4xlarge` and `c3.8xlarge` instances. Using fewer nodes limits parallelism and the more powerful instance wins out. However, considering the difference in cost and the fact that

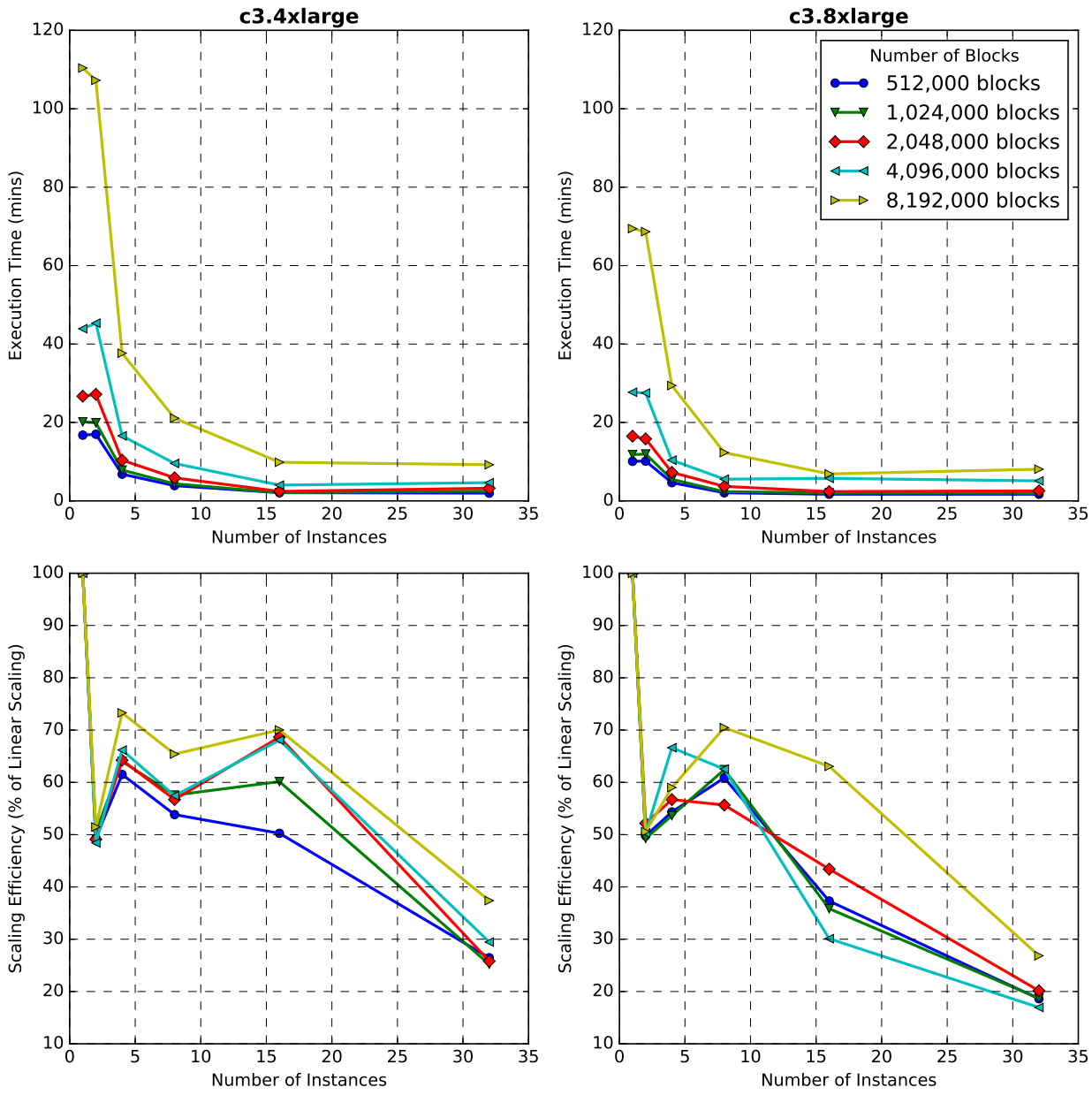


Figure 2.11: Execution Time and Scaling Efficiency as Cluster Size Increases - 134 Partitions

performance is very similar when using 4 nodes or more, `c3.4xlarge` seems to be a better starting choice in terms of instance type.

In terms of efficiency, both instance types exhibit similar trends though `c3.4xlarge` is somewhat more efficient. If `SparkRocks` had perfect strong scaling, the speedup would be equal to the number of nodes used. Most likely, the increase in communication overhead required to manage the cluster offsets much of the gain in additional resources. Also, the

strong scaling tests were performed with the same number of initial partitions, regardless of the cluster size. More in-depth optimization would most likely reveal better strong scaling with respect to varied initial partitioning.

2.5.5 Practical Implications

From a practical perspective, the results presented in Figure 2.11 reveal that, for most cases, using the less expensive `c3.4xlarge` instance type will yield very comparable speedup at a lesser price - even for as many as 8 million blocks. For greater problem sizes, where more memory and computational power are necessary, `c3.8xlarge` is available.

In this context, the computational speed of a single core of a 3.1GHz Intel Core-2-Duo CPU presented in Boon et al. [8] was compared to `SparkRocks` running on a laptop with an Intel Core i7-4720HQ (2.6 GHz) CPU with 4 cores and 10GB of memory, a workstation with two Intel Xeon E5-2630 v2 (2.3 GHz) CPUs with 12 cores and 20GB of memory, and an eight-node cluster of `c3.4xlarge` EC2 instances, using the best partition count for each problem size. The EC2 cluster features a total of 240GB of memory and 128 vCPUs.

Figure 2.12 is a plot of execution time against problem size. It shows that the parallel implementation offers orders of magnitude speedup compared to the serial implementation featured in [8] as the number of available cores and memory increases. In particular, it was observed that when the execution of `SparkRocks` is moved from a single desktop to an EC2 cluster, running times decrease by about an order of magnitude for problems of the same size, while the cluster can also accommodate much larger problems than the lone server. Running times on the EC2 cluster do not begin to significantly increase until the problem size becomes quite large. If even larger problem sizes were tested, running time on EC2 should scale similarly to the running times seen on the laptop and desktop deployments. Overall, performance on EC2 conformed to expectations, as the cluster represents about an order of magnitude increase in CPU and memory resources compared to the desktop, and it generally was observed that the execution of `SparkRocks` is CPU-bound. While the parallel processes running on different machines within the cluster now have to communicate over the network, the rock slicing algorithm implementation minimizes this communication, keeping overhead small and allowing `SparkRocks` to take nearly full advantage of the additional resources.

Overall, the parallel implementation in `SparkRocks` is capable of generating 8,192,000 blocks in roughly 9 minutes on EC2, while a serial analysis running on a desktop CPU is able to slice only 60,000 blocks in roughly ten minutes [8]. This speed and scalability is made possible by the use of Spark and the abstractions it provides. Expressing the rock slicing process as a series of transformations on a resilient distributed dataset of blocks allows the work to be spread and to scale to all of the nodes and CPU cores available. All of this was done without changing any of the actual rock slicing code. For the problem sizes typically seen in engineering practice and research, the `SparkRocks` parallel implementation can generate full block systems in a matter of minutes. Historically, access to the kinds of computing clusters that can provide this level of performance has been prohibitively

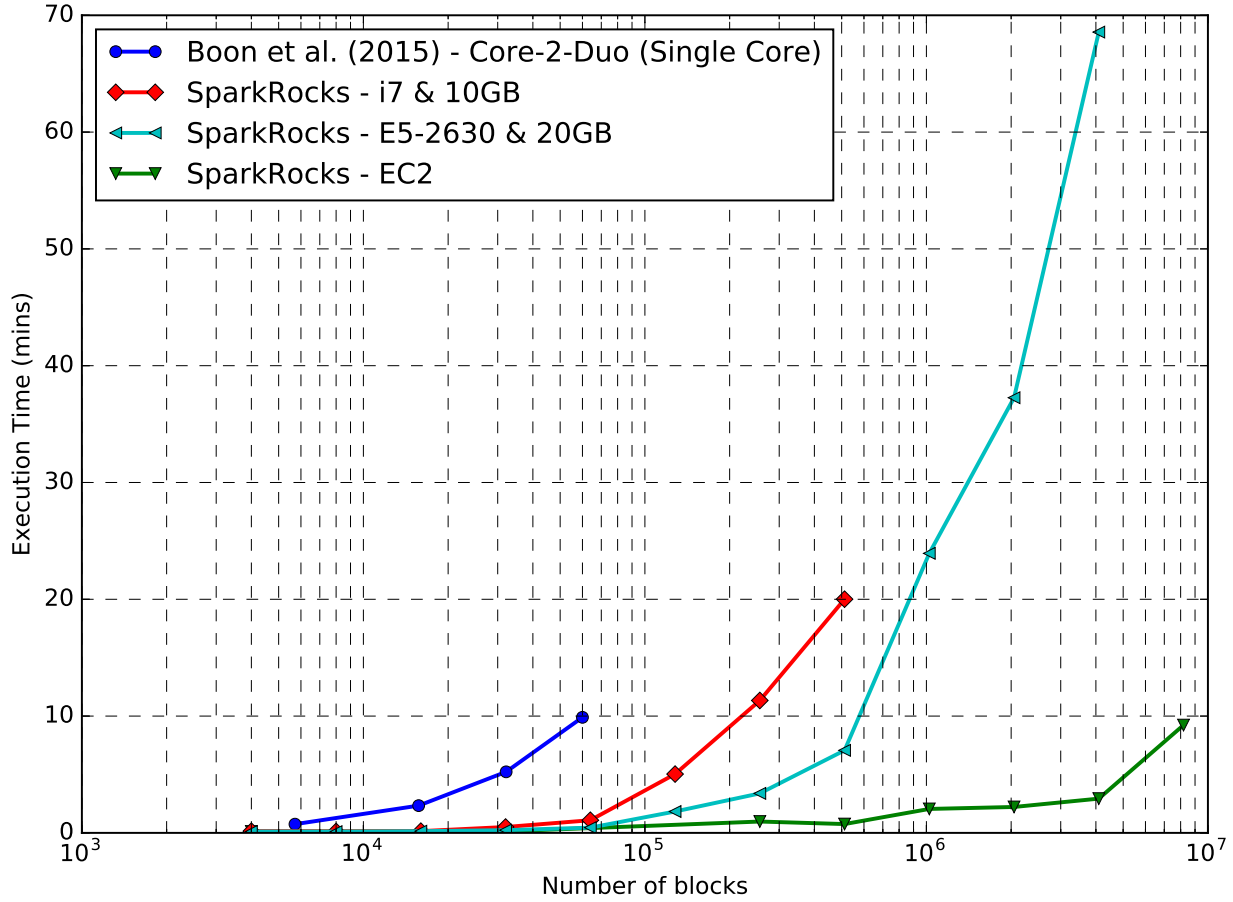


Figure 2.12: Execution Time vs. Problem Size for `SparkRocks` and Boon et. al. (2015) [8]. Note: x-axis is in log scale

expensive for many. However, with the relatively recent advent of Cloud Computing, users can forego provisioning their own clusters and access cloud resources instead; paying only for what they actually use. Thus, these results demonstrate that the computational resources are no longer a limitation and the solution of real-world scale problems is well within reach.

2.6 Summary

A parallel, scalable open-source application, `SparkRocks`, was developed that runs on Apache Spark to allow fast, parallel block system generation. Testing on different systems, ranging from multiprocessor workstations to Amazon EC2, shows that the parallel implementation offers orders of magnitude speedup for the solution of large problems. Moreover, the ability to take advantage of Cloud Computing greatly increases the scale of analyses that can be attempted. Real-world, large-scale block systems comprising millions of blocks can

be generated in a matter of minutes. Cloud Computing makes this scale of analysis available to any user since Cloud resources can be rented as-needed, negating the need to maintain a local computing cluster. Users only pay for what they use and only use what they need.

Chapter 3

3D Discrete Element Method for Analysis of the Mechanical Behavior of Jointed Rock Masses

3.1 Introduction

The mechanical behavior of fractured rock is governed by the discontinuities within the rock mass—displacements occur along fractures and joints and the strength of these discontinuities is significantly lower than that of the surrounding competent rock. Given this inherent discontinuous nature of the rock and the highly localized displacements along discontinuities, continuum based methods cannot capture the full kinematics of rock mass response. To address this shortcoming, numerical methods have been developed to explicitly account for the discrete, particulate nature of fractured rock, such as the Discrete Element Method (DEM)[11] and Discontinuous Deformation Analysis (DDA)[93]. These methods are well suited to describe the kinematic interaction between individual rock blocks and, both DEM, as expanded in [13] and [41], and DDA are able to explicitly model the polyhedral shape of the individual blocks. In this study DEM was selected to model the fractured rock mass because its explicit time integration and localized mechanical computations make it an attractive candidate for parallel computations.

3.2 Contact Detection

The contact detection phase is the most computationally expensive portion of DEM simulations, accounting for approximately 80% of the total simulation time [50]. Contact detection can be divided into two separate phases: neighbor search and contact resolution. During the neighbor search, a block's nearest neighbors—blocks that are close enough to possibly be in contact within a given time period or step—are identified. Once this phase

is completed, each of the nearest neighbors is checked to resolve if the blocks are actually physically in contact. The following sections describe each of these phases in more detail.

3.2.1 Neighbor Search

During simulations, only blocks in the same vicinity have the possibility of being in contact, so checking each block against every other block for contact is unnecessary and inefficient. For example, if the simulation contains N blocks, a naïve implementation would result in $O(N^2)$ operations. To avoid doing this, it is necessary to establish which blocks are "close enough" to warrant a more detailed, computationally intensive contact check. This is the purpose of the neighbor search.

Neighbor search algorithms can be divided into two classes: tree-based algorithms and binning algorithms [110]. Tree-based algorithms [[79], [17], [83]] generally are $O(Nln(N))$ while binning algorithms [[73], [110], [59]], also called spatial hashing algorithms, are $O(N)$. The fundamental idea in the neighbor search is to divide the domain into cells—with the size of cells set for each block individually, as in tree-based approaches, or a tunable parameter, as in binning approaches—and then map each of the blocks in the domain to the cells that they occupy [13]. For non-spherical particles, such as blocks, the shape of the particle is simplified to a bounding box—the smallest box that can contain the particle. This bounding box is what is used to establish which cells the block is mapped to. Figure 3.1 illustrates the concepts of both cell mapping and the bounding box as applied in spatial hashing. For the neighbor search phase in our software, we have implemented the CGRID algorithm [110]. CGRID, compared to the NBS algorithm [73], is able to maintain performance even when the sizes of particles in the simulation differ significantly, as is often the case for fractured rock. Additionally, the method can be extended to three dimensions relatively easily.

Once all blocks have been mapped to cells and the neighbors of each block have been established, the fine-grained contact resolution phase establishes whether blocks are physically in contact.

3.2.2 Contact Resolution

The neighbor search establishes which blocks possibly could be in contact, but it is then necessary to establish which of these neighboring blocks are actually in contact. Here, we make use of the block's bounding sphere to first check whether the bounding spheres of the neighboring blocks overlap. If they do, further contact resolution is required. If not, the blocks are not in contact.

In the case where further contact resolution is required, several algorithmic approaches are available. One approach is the common plane algorithm [13]. In this approach the contact between two neighboring blocks is established based on the relationship of the blocks to the common plane and not directly to each other. The fast common plane [76] and shortest link method [75] build upon this approach. Another class of contact resolution algorithms are the so-called direct search methods [[116], [118], [115], [117]]. While the common plane

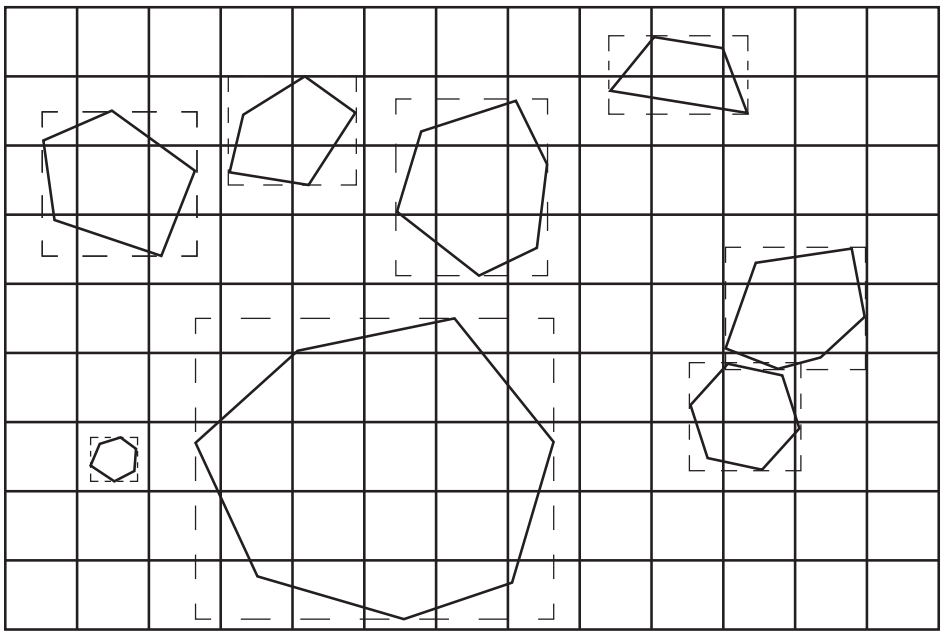


Figure 3.1: Blocks, with bounding boxes shown, moving through background cells. Bounding box is used to determine which cells block is mapped to.

approach avoids much of the "bookkeeping" in terms of tracking vertices, edges and faces, both common plane and direct search method suffer from sudden changes in contact type: If two blocks are initially in a vertex-to-vertex contact, one of the block may move such that the vertex that was in contact displaces just enough so that the contact is reclassified as vertex-to-face or vertex-to-edge. This can lead to undesired jumps in the contact force and sudden, unrealistic changes in the contact normal direction. Another approach that avoids these undesired effects recasts the contact detection problem as a convex optimization [7]. In this approach, the particle shape is described using only the faces of the blocks and their distance from some origin as previously described in Section 2.3. Describing the contact between two blocks using only the faces is attractive since floating point errors associated with tracking individual vertices and how they feature in contacts is partially alleviated—floating point errors are unavoidable, but the algorithm is more robust in terms of accounting for them. More importantly, the algorithm does not suffer from the sudden transitions in contact types since "potential particles" [51] are used as a proxy for the particle shape which allows for smooth transitioning of the contact normal in the vicinity of vertices and edges.

The following is an overview of the contact resolution process, including necessary details on how it is implemented. A full description of the algorithm is given in [7].

3.2.2.1 Establishing Contact

Following the approach outlined in [7] where the blocks are described using only their facets, contact is established by solving the following linear program:

$$\begin{gathered} \text{minimize s} \\ \boldsymbol{a_i}^T \boldsymbol{x} - d_i \leq s, \ i = 1, ..., N_A + N_B \end{gathered} \tag{3.1}$$

Where N_A and N_B represent the number of facets of the two neighboring blocks. The two blocks are in contact if $s < -\epsilon$ where ϵ is a specified numerical tolerance.

3.2.2.2 Contact Point

Once it has been verified that two blocks are actually in contact, it is necessary to calculate the contact point—the location at which the interaction forces between the two blocks are applied. As described in [7], the analytic center of the volume of the region of overlap is taken the contact point. This is illustrated in Figure 3.2. The analytic center is then solved for by minimizing the following:

$$\text{minimize} \quad t\boldsymbol{c}^T\boldsymbol{x} - \sum_{i=1}^{N} log(d_i - \boldsymbol{a}_i^T\boldsymbol{x}) \tag{3.2}$$

Equation 3.2 is solved repeatedly using the log-barrier method with Newton's method to find the optimal point—the analytic center—which is then used as the contact point between the two contacting blocks.

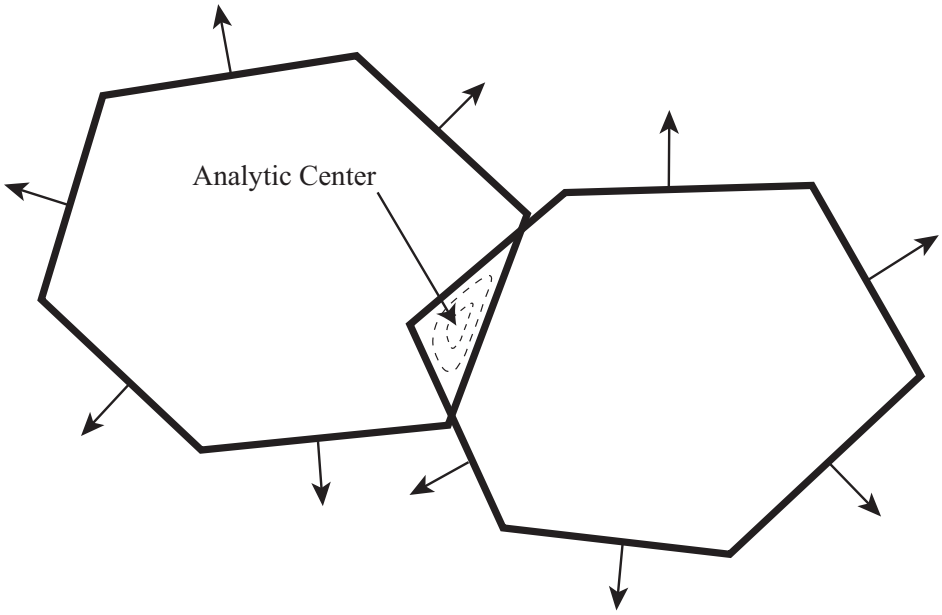


Figure 3.2: Two colliding blocks with analytic center taken as contact point. Arrows indicate the direction of normal vectors to block faces. (modified, based on [7])

3.2.2.3 Contact Normal

To avoid the previously mentioned issues and ambiguities in terms of establishing the type of contact and the associated contact normal, [7] utilized the concept of "potential particles" [51]. These potential particles are located entirely inside the block and are used to efficiently calculate the contact normal regardless of the location of the contact relative to vertices and edges. Figure 3.3 illustrates the concept of potential particles and how the normal is calculated for a particular contact point. The potential particle is defined by:

$$f = \sum_{i=1}^{N} \langle a_i x + b_i y + c_i z - d_i + r \rangle^2 \tag{3.3}$$

where the values a_i, b_i and c_i are the components of the normal vectors to the facets of the block and r is the radius of curvature of the corners. The Macaulay brackets are defined such that $\langle x \rangle = 0$ and $\langle -x \rangle = 0$. Given this definition, it is important to note that r should be selected such that it is greater than the maximum overlap expected between any two particles during the simulation. Generally, the constraints on the time step in terms of numerical stability enforce that the overlap is significantly smaller than the value of r. The definition of the potential particle can also be viewed as a distance function where the distance is set to zero inside the particle and is positive outside the particle. Then, the normal to the particle is given by the gradient of f as defined in Equation 3.3:

$$\begin{aligned} &\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right) \\ &\text{where} \\ &\frac{\partial f}{\partial x} = 2 \sum_{i=1}^{N} a_i \langle a_i x + b_i y + c_i z - d_i + r \rangle \\ &\frac{\partial f}{\partial y} = 2 \sum_{i=1}^{N} b_i \langle a_i x + b_i y + c_i z - d_i + r \rangle \\ &\frac{\partial f}{\partial z} = 2 \sum_{i=1}^{N} c_i \langle a_i x + b_i y + c_i z - d_i + r \rangle \end{aligned} \tag{3.4}$$

The contact normal is taken as the average of the contact normals to the potential particles of the two colliding blocks. The contact overlap is then calculated by finding the length of the line passing through the contact point in the contact normal direction that terminates on the surfaces of the two colliding blocks.

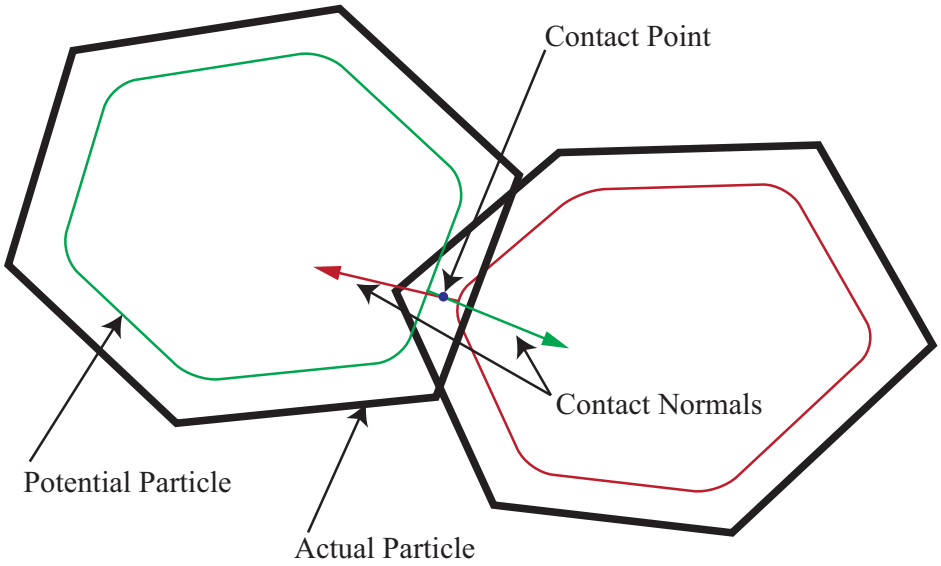


Figure 3.3: Two colliding blocks with contact normals based on each blocks potential particle shown. Contact normals pass through contact point. (modified, based on [7])

3.3 Contact Forces and Moments

Once contact between two block has been established, the next step is to describe how they interact with each other—what are the forces and moments between the two blocks. The contact normal and overlap calculated in the contact resolution phase serve as inputs to establish the interaction forces between the blocks. In its basic formulation, DEM is modular in terms of how the contact forces are described and many different formulations exist for calculating contact forces. In the simplest case, the contact between two blocks is described as linear elastic in the contact normal direction and frictional, using Coulomb friction with cohesion, in the tangential direction [41]. More involved models [[107], [103], [86]] are based on elastic theory and can be considered as variants of [46] for the normal direction and [72] for the tangential direction. However, these models are primarily applicable to spherical particles. In the case of polyhedral rock blocks, models have been developed that describe the behavior of rock joints [[35], [4], [84], [1], [55]] which are more applicable than the models based on more spherical particle geometry.

For this research, the focus is on the hydromechanical coupling between a single block and surrounding fluid. Therefore, the use of a highly specialized contact model is not the primary concern and will not have an effect on the hydrodynamics of the fluid-solid coupling. Consequently, a linear elastic contact model and Coulomb friction with cohesion were used to describe the contact forces between the block and boundaries. However, the software implementation of DEM has been developed in a modular fashion such that more intricate contact models can be added for future research.

3.3.1 Forces and Moments

Following the procedure outlined by [41], the translational velocity of the contact—the velocity of block B relative to block A at the contact point—is calculated by:

$$\boldsymbol{v}_{contact} = \boldsymbol{v}_B + \boldsymbol{\omega}_B \times (\boldsymbol{x}_{contact} - \boldsymbol{x}_B) + \boldsymbol{\omega}_A \times (\boldsymbol{x}_{contact} - \boldsymbol{x}_A) \tag{3.5}$$

where $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ are the position vectors of the centroids of the two colliding blocks, $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$ are the translational velocity of the blocks, $\boldsymbol{\omega}_A$ and $\boldsymbol{\omega}_B$ are the angular velocities of the two colliding blocks and $\boldsymbol{x}_{contact}$ is the contact point established in the contact resolution phase.

The displacement at the contact is then calculated as:

$$\Delta\boldsymbol{u}_{contact} = \boldsymbol{v}_{contact}\Delta t \tag{3.6}$$

which is then resolved into normal and tangention components relative to the contact normal:

$$\begin{aligned} \Delta u_n &= \boldsymbol{u}_{contact} \cdot \hat{\boldsymbol{n}}_{contact} \\ \Delta\boldsymbol{u}_t &= \boldsymbol{u}_{contact} - (\hat{\boldsymbol{n}}_{contact} \otimes \hat{\boldsymbol{n}}_{contact})\, \boldsymbol{u}_{contact} \end{aligned} \tag{3.7}$$

The contact normal is updated at every time step, so existing tangential forces need to be updated to ensure they still act along the tangential direction of the contact. The tangential force is updated as:

$$\boldsymbol{F}_t^{updated} = \boldsymbol{F}_t^{old} - \boldsymbol{F}_t^{old} \times (\hat{\boldsymbol{n}}_{old} \times \hat{\boldsymbol{n}}_{updated}) \tag{3.8}$$

The contact displacements are used to calculate the elastic force increments at the contact.The normal force increment is calculated as:

$$\Delta F_n = -K_n \Delta u_n \tag{3.9}$$

and the tangential force increment is:

$$\Delta\boldsymbol{F}_t = -K_t \Delta\boldsymbol{u}_t A_{contact} \tag{3.10}$$

where K_n is the contact normal stiffness, K_t is the tangential stiffness and $A_{contact}$ is the contact area. Note that compressive forces are taken as positive. The contact area is calculated by passing a plane through the contact point with the contact normal. The vertices at the intersection of this plane and the faces defining the contacting blocks are used to calculate the contact area by simplex integration [91].

The total normal and tangential forces are then updated as:

$$\begin{aligned} F_n^{new} &= F_n^{old} + \Delta F_n \\ \boldsymbol{F}_t^{new} &= \boldsymbol{F}_t^{old} + \Delta\boldsymbol{F}_t \end{aligned} \tag{3.11}$$

It is necessary to check whether the total normal force is tensile. If it is greater than the specified cohesion or there is no cohesion, the normal force is set to zero. In the tangential direction, the shear force is checked against the maximum values:

$$F_{max} = cA_{contact} + F_n tan\phi \tag{3.12}$$

The magnitude of the tangential force is simply the norm $\boldsymbol{F}_t$. If the norm F_t is greater than $\boldsymbol{F}_{max}$, then the tangential force is set to:

$$\boldsymbol{F}_t = F_{max} \frac{\boldsymbol{F}_t}{\|\boldsymbol{F}_t\|} \tag{3.13}$$

The contact force vector is taken as:

$$\boldsymbol{F}_{contact} = -F_n \hat{\boldsymbol{n}}_{contact} - \boldsymbol{F}_t \tag{3.14}$$

The total forces and moments acting on each of the contacting blocks are then updated:

$$\begin{aligned}
&\text{Block A:} \\
&\boldsymbol{F}_A^{updated} = \boldsymbol{F}_A^{previous} - \boldsymbol{F}_{contact} \\
&\boldsymbol{M}_A^{updated} = \boldsymbol{M}_A^{previous} - (\boldsymbol{x}_{contact} - \boldsymbol{x}_A) \times \boldsymbol{F}_{contact} \\
&\text{Block B:} \\
&\boldsymbol{F}_B^{updated} = \boldsymbol{F}_B^{previous} + \boldsymbol{F}_{contact} \\
&\boldsymbol{M}_B^{updated} = \boldsymbol{M}_B^{previous} + (\boldsymbol{x}_{contact} - \boldsymbol{x}_B) \times \boldsymbol{F}_{contact}
\end{aligned} \tag{3.15}$$

The total forces and moments calculated above are then used to update the block position as described in Section 3.3.2.

3.3.2 Time Integration

The equations of translational and rotational motion for an individual rock block are:

$$\begin{aligned}
\ddot{\boldsymbol{x}}_i + \alpha \dot{\boldsymbol{x}}_i &= \frac{\boldsymbol{F}_i}{m} + \boldsymbol{g}_i \\
\dot{\boldsymbol{\omega}}_i + \alpha \boldsymbol{\omega}_i &= \frac{\boldsymbol{M}_i}{I}
\end{aligned} \tag{3.16}$$

where $\ddot{\boldsymbol{x}}_i$ and $\dot{\boldsymbol{\omega}}_i$ are the translational and rotational acceleration of block i; $\boldsymbol{F}_i$ and $\boldsymbol{M}_i$ are the total force and moment acting on block i; α is a damping constant; and $\boldsymbol{g}_i$ is the gravitational acceleration. The block motion is updated by integrating the equations of motion.

The block translational motion is integrated using a velocity Verlet finite difference approach [101] which is $O(h^2)$ in time. First, the velocity at time $t + \Delta t$ is calculated as:

$$\boldsymbol{v}_i^{t+\frac{\Delta t}{2}} = \boldsymbol{v}_i^{t-\frac{\Delta t}{2}} + \boldsymbol{a}_i^t \Delta t \tag{3.17}$$

where $\boldsymbol{a}_i^t$ is the acceleration of block i at time t based on the total force, $\boldsymbol{F}_i^{Total}$, acting on the block:

$$\boldsymbol{a}_i^t = \frac{\boldsymbol{F}_i^{Total} - \alpha |\boldsymbol{F}_i^{Total}| sgn\left(\boldsymbol{v}_i^t\right)}{m_i} \tag{3.18}$$

where the acceleration is damped proportional to the magnitude of the unbalanced force acting on the block [12]. The block position is then updated using the velocity at $t + \frac{\Delta t}{2}$:

$$\boldsymbol{x}_i^{t+\Delta t} = \boldsymbol{x}_i^t + \Delta t \boldsymbol{v}_i^{t+\frac{\Delta t}{2}} \tag{3.19}$$

The rotational motion is updated using a quaternion-based approached developed by [57]. This approach guarantees orthonormality of the resulting rotation and also overcomes gimbal lock of the Euler angles. The rotations are integrated using a fourth-order Runga-Kutta scheme. The rotations are based on a local coordinate system that is centered at the block's center of mass. The orientation of the coordinate system is fixed to the coordinate axes of the principal moments of inertia of the block. For general polyhedra, these are calculated using simplex integration [91] and then finding the eigenvalues and eigenvectors of the resulting inertia tensor—the eigenvalues represent the principal moments of inertia and the eigenvectors are the axes along which they act.

This local coordinate system is established at the beginning of simulations after which the rotations are updated as follows. For the first approximation, the rotation is assumed as:

$$\langle q_0, \boldsymbol{q} \rangle_{(1)} = \frac{\Delta t}{2} \langle \omega_0, \boldsymbol{\omega} \rangle_t \tag{3.20}$$

where the $\boldsymbol{q}$ and $\boldsymbol{\omega}$ represent the position and angular velocity quaternions—the first quantity in the the angle brackets is the scalar value of the quaternion while the second quantity is the vector portion. The angular velocity is then updated as:

$$\begin{aligned} \langle \omega_0, \boldsymbol{\omega} \rangle_{(2)} =& \widetilde{E} \left(R\left(\langle q_0, \boldsymbol{q} \rangle_{(1)} \circ \langle r_0, \boldsymbol{r} \rangle_t \right) \hat{J}^{-1} R\left(\langle q_0, \boldsymbol{q} \rangle_{(1)} \circ \langle r_0, \boldsymbol{r} \rangle_t \right)^T \right) \\ & \times \left(\left(R\left(\langle r_0, \boldsymbol{r} \rangle_t \right) \hat{J} R\left(\langle r_0, \boldsymbol{r} \rangle_t \right)^T \right) \boldsymbol{\omega}_t + \boldsymbol{M}_t^{Total} \Delta t \right) \end{aligned} \tag{3.21}$$

where $\widetilde{E}$ represents an operator for constructing a quaterion from a rotation matrix and R represents an operator for constructing a rotation matrix from a quaternion. The quaternion $\langle r_0, \boldsymbol{r} \rangle_t$ is the current orientation of the block and $\hat{J}$ are the principal moments of inertia of the block. The second approximation of the rotation then becomes:

$$\langle q_0, \boldsymbol{q} \rangle_{(2)} = \frac{\Delta t}{2} \langle \omega_0, \boldsymbol{\omega} \rangle_{(2)} \tag{3.22}$$

which is used to update the approximation of the angular velocity:

$$\begin{aligned}\langle \omega_0, \boldsymbol{\omega} \rangle_{(3)} =& \widetilde{E} \left(R \left(\langle q_0, \boldsymbol{q} \rangle_{(2)} \circ \langle r_0, \boldsymbol{r} \rangle_t \right) \hat{J}^{-1} R \left(\langle q_0, \boldsymbol{q} \rangle_{(2)} \circ \langle r_0, \boldsymbol{r} \rangle_t \right)^T \right) \\ & \times \left(\left(R \left(\langle r_0, \boldsymbol{r} \rangle_t \right) \hat{J} R \left(\langle r_0, \boldsymbol{r} \rangle_t \right)^T \right) \boldsymbol{\omega}_t + \boldsymbol{M}_t^{Total} \Delta t \right)\end{aligned} \tag{3.23}$$

The third approximation of the rotation is:

$$\langle q_0, \boldsymbol{q} \rangle_{(3)} = \frac{\Delta t}{2} \langle \omega_0, \boldsymbol{\omega} \rangle_{(3)} \tag{3.24}$$

which gives an updated an angular velocity:

$$\begin{aligned}\langle \omega_0, \boldsymbol{\omega} \rangle_{(4)} =& \widetilde{E} \left(R \left(\langle q_0, \boldsymbol{q} \rangle_{(3)} \circ \langle r_0, \boldsymbol{r} \rangle_t \right) \hat{J}^{-1} R \left(\langle q_0, \boldsymbol{q} \rangle_{(3)} \circ \langle r_0, \boldsymbol{r} \rangle_t \right)^T \right) \\ & \times \left(\left(R \left(\langle r_0, \boldsymbol{r} \rangle_t \right) \hat{J} R \left(\langle r_0, \boldsymbol{r} \rangle_t \right)^T \right) \boldsymbol{\omega}_t + \boldsymbol{M}_t^{Total} \Delta t \right)\end{aligned} \tag{3.25}$$

The fourth approximation of the rotation is then given by:

$$\langle q_0, \boldsymbol{q} \rangle_{(4)} = \frac{\Delta t}{2} \langle \omega_0, \boldsymbol{\omega} \rangle_{(4)} \tag{3.26}$$

The final approximation of the rotation is taken as:

$$\langle q_0, \boldsymbol{q} \rangle = \frac{1}{6} \left(\langle q_0, \boldsymbol{q} \rangle_{(1)} + 2 \langle q_0, \boldsymbol{q} \rangle_{(2)} + 2 \langle q_0, \boldsymbol{q} \rangle_{(3)} + \langle q_0, \boldsymbol{q} \rangle_{(4)} \right) \tag{3.27}$$

The block's new orientation and angular velocity are then set as:

$$\begin{aligned}\langle r_0, \boldsymbol{r} \rangle_{t+\Delta t} =& \langle q_0, \boldsymbol{q} \rangle \circ \langle r_0, \boldsymbol{r} \rangle_t \\ \langle \omega_0, \boldsymbol{\omega} \rangle_{t+\Delta t} =& \frac{1}{6} \left(\langle \omega_0, \boldsymbol{\omega} \rangle_t + 2 \langle \omega_0, \boldsymbol{\omega} \rangle_{(2)} + 2 \langle \omega_0, \boldsymbol{\omega} \rangle_{(3)} + \langle \omega_0, \boldsymbol{\omega} \rangle_{(4)} \right)\end{aligned} \tag{3.28}$$

The steps for calculating the displacement and velocity—both translational and rotational—are repeated at each time step until the desired simulation length it reached.

3.3.3 Numerical Stability

The velocity Verlet method is an explicit method and therefore only conditionally stable. For a single-degree-of-freedom system with zero damping, [74] shows that the system will be stable when the time step is smaller than:

$$\Delta t_{critical} \leq \frac{2}{\sqrt{\frac{K}{m}}} \tag{3.29}$$

where m is the mass of the particle and K is its stiffness. In a system with many particles, the critical time step decreases as the number of contacts per particle increases [80]. Since it is not possible to know the number of contacts each particle will experience throughout

the simulation, it is not possible to analytically determine what the value for K should be. [38] suggests the following for the critical time step:

$$\Delta t_{critical} \leq frac\, 2\sqrt{\frac{m_{min}}{K_{max}}} \tag{3.30}$$

where m_{min} is the mass of the smallest block, K_{max} is the maximum contact stiffness, and *frac* is a user defined value that is set to 0.1 by default. This value accounts for the fact that one block may be in contact with several other blocks simultaneously, effectively increasing the contact stiffness, K_{max}.

3.4 Validation

The Discrete Element Method was implemented in C++ using the algorithms described above. All of the individual pieces of functionality in the code were unit tested using `CxxTest` [106]—an open-source unit testing framework for C++. Beyond simple unit tests, integration testing was also done to verify that the software is able to match analytical solutions as well as reproduce the kinematics observed in the field for more complicated slope configurations.

3.4.1 Sliding Block Analysis

For polyhedra, the simplest analytical test case is that of a block sliding down an inclined plane. This tests the ability of the software to properly capture the friction force and the gravitational acceleration. The free-body diagram for a block sitting on an inclined plane is shown in Figure 3.4.

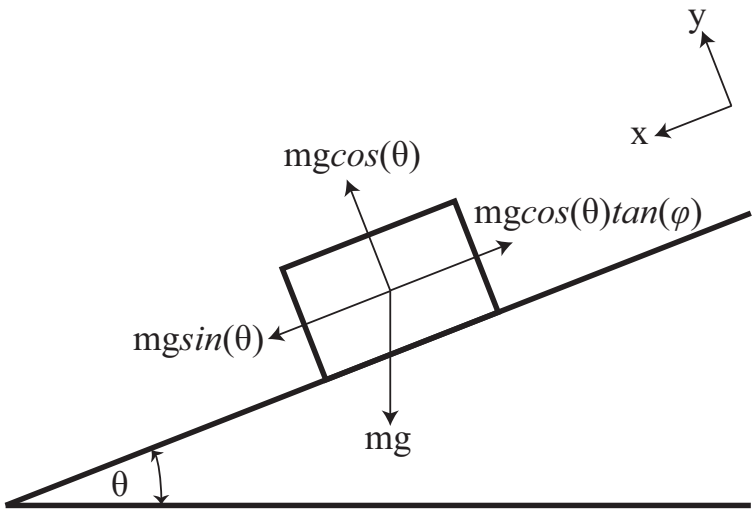


Figure 3.4: Free body diagram for block with mass, m, on inclined plane under gravitational acceleration, g

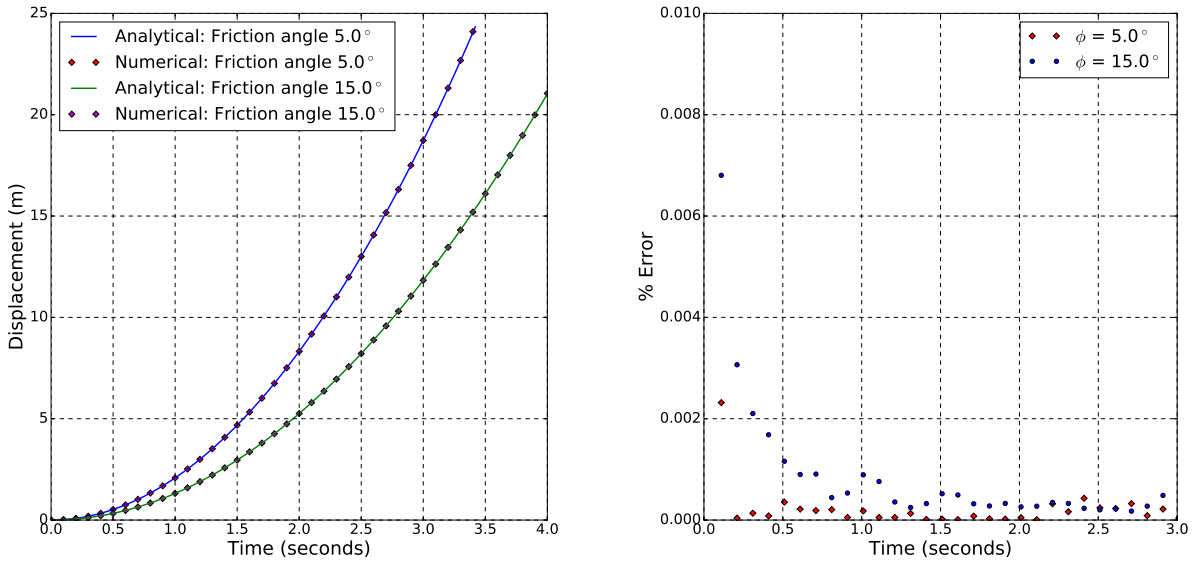
For a given gravitational force, the inclination of the plane and the angle of friction between the block and sliding surface will dictate whether the block slides down the plane and, if it does, at what rate it will accelerate. In the case where sliding does occur, the position of the block on the plane is given by:

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 + \boldsymbol{v}_0 t + \frac{1}{2}\boldsymbol{a}t^2 \tag{3.31}$$

where $\boldsymbol{x}$ is the position of the block at time t, $\boldsymbol{x}_0$ and $\boldsymbol{v}_0$ are the initial position and velocity of the block and $\boldsymbol{a}$ is the constant gravitational force acting on the block. The acceleration is given as a function of the slope and friction angles by:

$$\boldsymbol{a} = (sin(\theta) - tan(\phi)\, cos(\theta))\, \boldsymbol{g} \tag{3.32}$$

where θ is the slope angle and ϕ is the angle of friction between the block and the sliding plane.



(a) Block Displacement vs Time

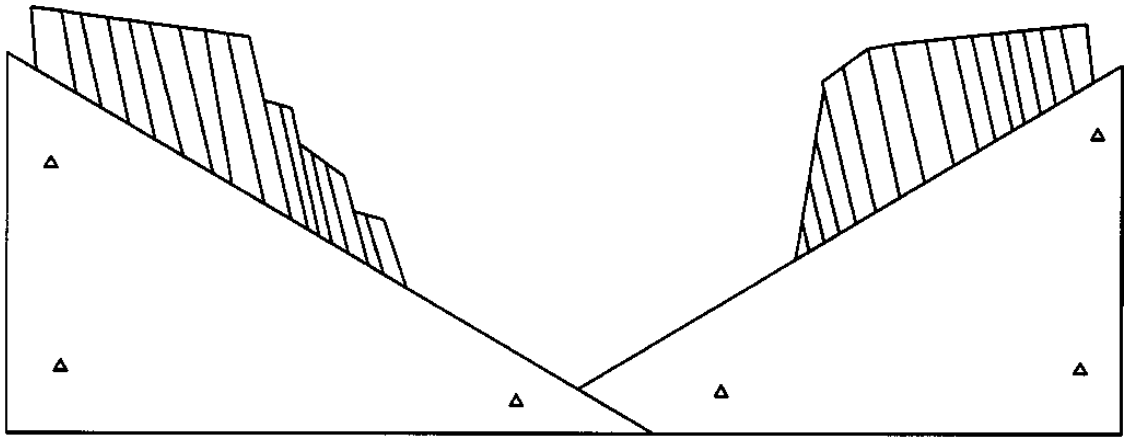
(b) Percent Error vs Time

Figure 3.5: Comparison of Numerical Results with Analytical Solution for Sliding Block

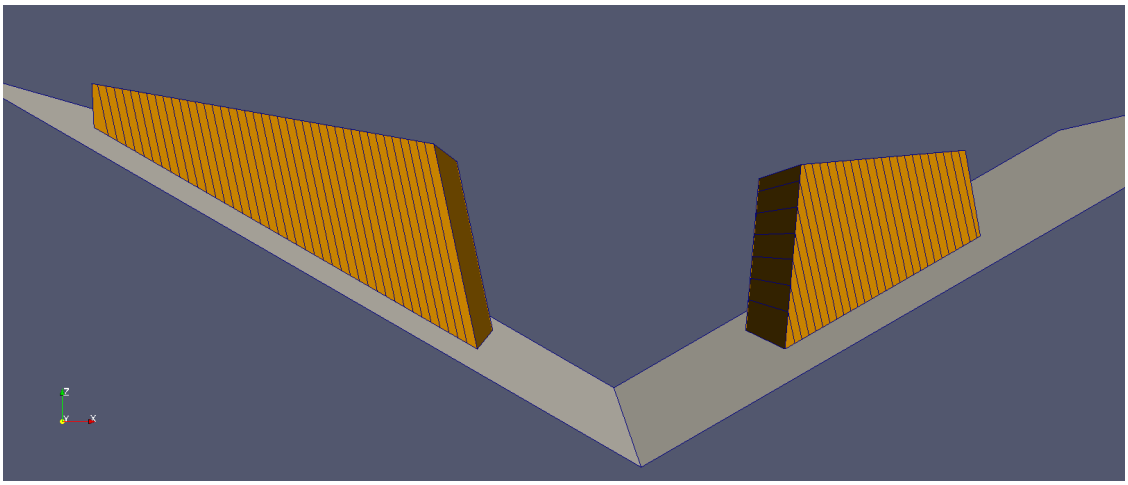
Figure 3.5 shows the comparison of the numerical results with the analytical solution. The numerical results match the analytical solution very well. The case where the slope angle and friction angle match was also tested—in this case no sliding should occur—and sliding did not occur in the numerical analysis. All these tests used a linear elastic contact model with Coulomb friction for the contact between the block and sliding plane.

3.4.2 Toppling and Slumping Slope Failure

The orientation of fractures and discontinuities within rock relative to slope configuration govern how failure modes develop [36]. Any discontinuous numerical model needs to be able capture this phenomenon if it is to be used for analyzing fractured rock. In order to ensure our implementation of DEM functions correctly, we compared the results from a three-dimensional DEM analysis with that of a two-dimensional Discontinuous Deformation Analysis (DDA) [95] that illustrates how geometry of discontinuities affects kinematic behavior of rock slopes.



(a) Initial configuration from [95]

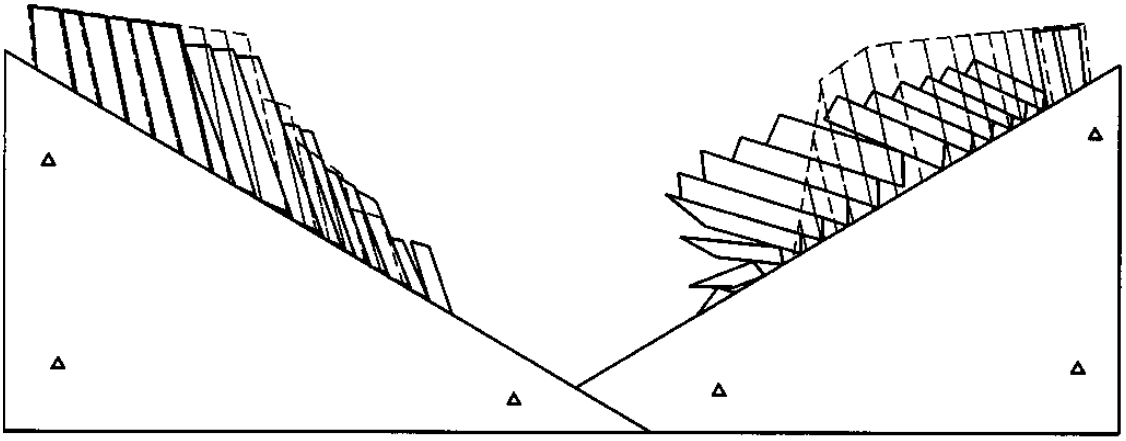


(b) Initial configuration for this analysis

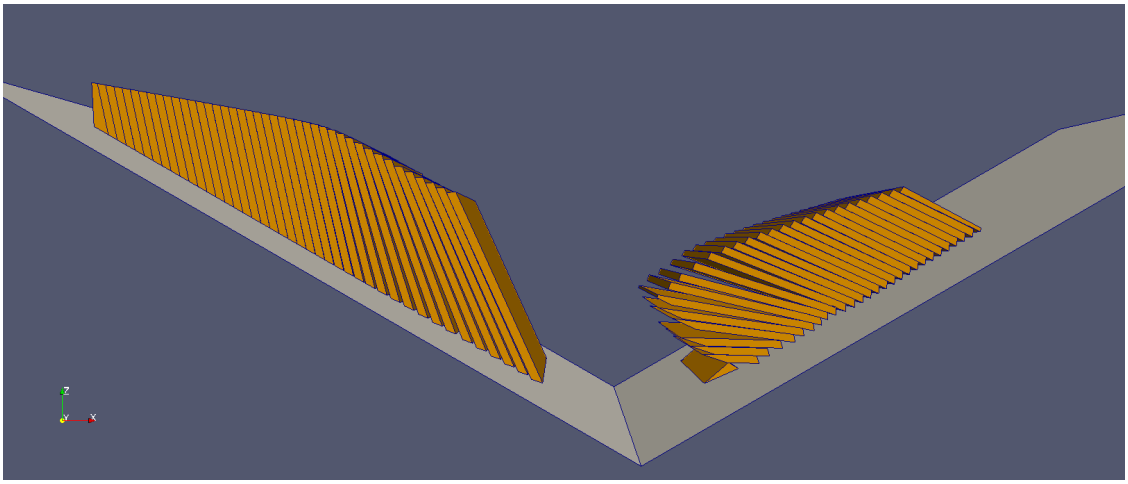
Figure 3.6: Initial configuration for failure mode analysis

In this analysis, a valley or cut in pervasively jointed rock is shown to have two different failure modes on either side of the valley—the failure mode is governed by the orientation of the joints relative to the failure plane. Figure 3.6 shows the initial configuration from [95]

alongside the initial configuration used in this analysis. Through-going joints dip at 30°, forming the sliding plane on either side of the valley. The friction angle along the sliding plane is 32° and the friction angle between the individual rock blocks is 22°. If the rock mass was considered as a single wedge on either side of the valley, the analysis would conclude that both slopes are safe against against sliding. However, as shown in Figure 3.7, the slope on the right fails by toppling while the slope on the left fails by slumping [95]. Both of these modes are captured in a single analysis, illustrating the capability of DEM to capture the kinematics governing failure for different slope configurations. This is an essential feature for analyzing more complex landslides where different failure modes may initiate as the slope failure progresses.



(a) Failure modes from [95]



(b) Failure modes for this analysis

Figure 3.7: Slumping (left) and toppling (right) failure modes

3.4.3 Wedge Sliding

Wedge sliding is a typical failure mode in blocky rock masses. For this type of failure, as show in Figure 3.8, a rock block slides without rotation along two non-parallel planes along their line of intersection. This type of failure is typical in blocky rock masses with multiple, continuous, non-parallel joint sets [36]. Block theory [34] offers an analytical solution for determining whether a block will fail in this mode. The force F required to stabilize a block sliding along two planes with normals $\hat{n}_i$ and $\hat{n}_j$ due to the resultant $\boldsymbol{r}$ of all forces acting on the block is given by:

$$\begin{aligned} F = \frac{1}{\|\hat{n}_i \times \hat{n}_j\|^2}[&|\boldsymbol{r} \cdot (\hat{n}_i \times \hat{n}_j)| \cdot \|\hat{n}_i \times \hat{n}_j\| \\ &- |(\boldsymbol{r} \times \hat{n}_j) \cdot (\hat{n}_i \times \hat{n}_j)| tan\phi_i \\ &- |(\boldsymbol{r} \times \hat{n}_i) \cdot (\hat{n}_i \times \hat{n}_j)| tan\phi_j] \end{aligned} \tag{3.33}$$

where ϕ_i and ϕ_j are the angles of friction on the two sliding planes. When F is negative the block is stable, when F is positive the block is unstable. If F is zero, the block is in equilibrium and the factor of safety is essentially 1.0. When only gravitational loading is considered, $\boldsymbol{r} = \langle 0.0, 0.0, -W \rangle$ where W is the weight of the block.

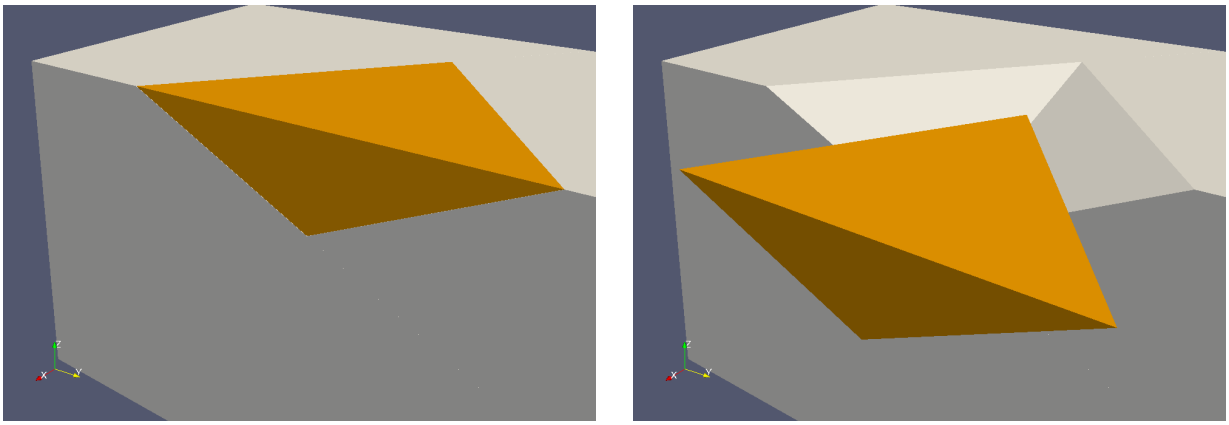


Figure 3.8: Blocky rock wedge failing in sliding. Block translates along line of intersection of sliding planes.

To verify the ability of the DEM implementation to correctly capture the three-dimensional kinematic behavior of blocky rocks, the results from several wedge failure analyses were compared with the analytical solution from block theory. Table 3.1 shows the comparison of the numerical results with the predicted values from Equation 3.33. The same set of joints was used for all analyses while modifying the angle of friction on the sliding planes. The numerical analysis captures the transition from stable to sliding very well and matches the results from block theory. For the cases where F was very small and negative—indicating the block is stable, but not by a great amount—the block displaced only very slightly at the

beginning of the simulations. However, once enough frictional resistance was mobilized the block stopped sliding and remained stationary for the remainder of the simulation time.

$\hat{n}_i$	$\hat{n}_j$	ϕ_i	ϕ_j	$F[N]$ (Equation 3.33)	DEM Result
$\langle 0.321, -0.383, 0.866 \rangle$	$\langle 0.321, 0.383, 0.866 \rangle$	30.0	30.0	-5311.0	No sliding
		19.0	19.0	-34.3	No sliding
		18.9	18.9	9.9	Sliding
		15.0	15.0	1695.3	Sliding
		15.0	30.0	-1807.9	No sliding
		6.5	30.0	-64.1	No sliding
		6.0	30.0	35.9	Sliding
		3.0	30.0	632.5	Sliding

Table 3.1: Wedge Sliding: Comparison of numerical results and block theory

3.5 Summary

A three-dimensional DEM program capable of modeling polyhedral particles was developed in C++. The contact detection computations for DEM are based on a linear programming approach such that similar data structures and logic can be used in both the DEM program and the block generation application described in Chapter 2. The results of the validation analyses demonstrate that the program is able to accurately capture the kinematic response of three-dimensional polyhedral rock blocks.

Chapter 4

Coupled Discrete Element and Lattice Boltzmann Methods for Modeling Rock-Water Interaction

4.1 Introduction

Modeling the interaction between rock and water, requires coupling of the numerical model for the solid phase—DEM in this case—with a model for the fluid phase. In terms of the coupling process, there are two primary considerations that need to be taken into account: 1.) the ability of the fluid model to capture the boundary between the fluid and solid phases and, 2.) the efficiency of the fluid model when the solid phase is allowed to move through the fluid domain as the solution progresses.

Methods conventionally used in computational fluid dynamics (CFD), such as the Finite Element Method (FEM) [120] and Finite Volume Method (FVM) [66], directly solve the Navier-Stokes equations to describe the evolution of the macroscopic variables of the fluid, such density and velocity. These methods are able to capture complex boundary shapes through unstructured and irregular grids and, in the case of FEM, are amenable to higher order methods. However, the mesh generation can be quite complicated and computationally intensive which can be prohibitive if there are solid particles moving through the fluid domain—generally necessitating remeshing at every time step.

The Lattice Boltzmann Method (LBM) arrives at the solution of the Navier-Stokes equations in a much different fashion compared to conventional CFD. LBM solves the mesoscopic behavior of the fluid in that it describes the behavior of distributions of particles, not the macroscopic velocity and density nor the behavior of individual particles. The physical basis of LBM is rooted in the Boltzmann equation, but it can be linked to the macroscopic behavior of fluid, in effect solving the weakly compressible Navier-Stokes equations. What makes LBM particularly attractive for modeling fluid-solid interaction is the localized nature of the method and, more importantly, the ease with which complex shapes moving through

the fluid domain can be accommodated. As a solid particle moves through the fluid domain, the state of the nodes that the solid interacts with are updated. Based on the status of the node, the presence of the solid is accounted for in the fluid solution and the effect of the fluid on the solid is also considered. Typically, there is no need for remeshing and the change in the status of each node is incorporated in the computations locally. Given the relative ease of this coupling process compared to other methods used in CFD, LBM was chosen in this research to model the fluid-solid interaction.

4.2 The Lattice Boltzmann Method

The Lattice Boltzmann Method solves a discrete form of the Boltzmann equation—the so-called Lattice Boltzmann Equation (LBE)—to arrive at the solution for various problems in fluid dynamics. The Lattice Boltzmann Equation [70] was first developed as a response to the main drawback—statistical noise—of its predecessor Lattice Gas Cellular Automata [19]. The LBE is derived starting from the force-free Boltzmann equation:

$$\frac{\partial f}{\partial t} + \xi_\beta \frac{\partial f}{\partial x_\beta} = \Omega(f) \tag{4.1}$$

Equation 4.1 describes the advection of the distribution function f with particle velocity $\boldsymbol{\xi}$ [60]. The source term Ω, called the collision operator, on the right hand side of the equation represents the redistribution of f due to interparticle collisions. The Lattice Boltzmann Equation is obtained by discretizing Equation 4.1 in velocity space, physical space and time:

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i \Delta t, t + \Delta t) = f_i(\boldsymbol{x}, t) + \Omega_i(\boldsymbol{x}, t) \tag{4.2}$$

where $\boldsymbol{c}_i$ is the discrete set of velocities which limits the continuous particle velocity $\boldsymbol{\xi}$ to a carefully selected subset, as described in Section 4.2.1. This equation describes particles $f_i(\boldsymbol{x}, t)$ moving with velocity $\boldsymbol{c}_i$ to a neighboring point located at $\boldsymbol{x}+\boldsymbol{c}_i\Delta t$, as shown in Figure 4.1. This is known as the *streaming* step. Additionally, the collision operator redistributes particles among the populations f_i at each point—this redistribution models particle collisions. This is known as the *collision* step and is discussed in more detail in Section 4.2.2. Together the streaming and collision steps are the fundamental concept of the LBE.

The basic variable in LBM, as shown in Equation 4.2, is the discrete-velocity distribution function, $f_i(\boldsymbol{x}, t)$. This distribution function represents the density of particles with velocity $\boldsymbol{c}_i$ at time t and position $\boldsymbol{x}$. The macroscopic fluid mass density and momentum are calculated through weighted sums in velocity space, known as moments, of f_i:

$$\begin{aligned} \rho(\boldsymbol{x}, t) &= \sum_i f(\boldsymbol{x}, t) \\ \rho \boldsymbol{u}(\boldsymbol{x}, t) &= \sum_i \boldsymbol{c}_i f(\boldsymbol{x}, t) \end{aligned} \tag{4.3}$$

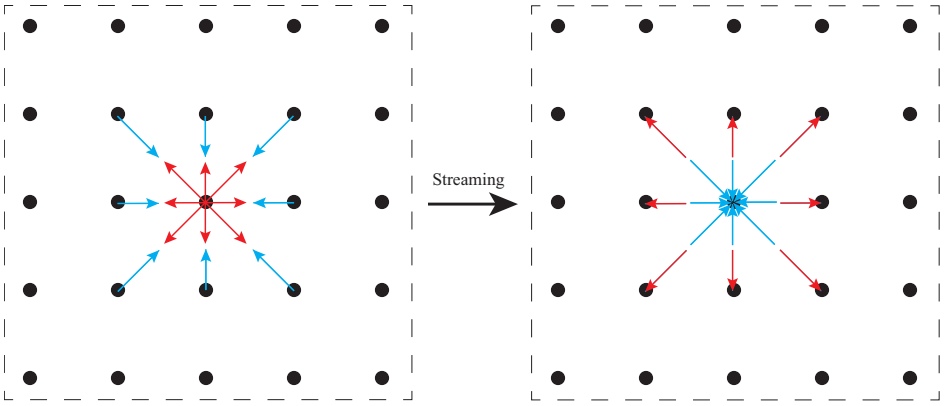


Figure 4.1: Distributions streaming between central node and neighbors. The figure on the left shows distributions prior to streaming, while the figure on the right shows the distributions post-streaming (After [60])

The connection between the Navier-Stokes equations and the LBE can be determined through the Chapman-Enskog analysis [9]. Through this analysis, it can be shown that the LBE models the macroscopic behavior of the Navier-Stokes equations when the kinematic shear viscosity is related to the relaxation parameters of the collision model as well as the spatial and temporal discretization.

4.2.1 Velocity Sets

The continuous particle distribution function $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ spans a seven-dimensional space—x, y, z, ξ_x, ξ_y, ξ_z and t—which may initially seem prohibitively computationally intensive and mathematically daunting. However, the moments of the Boltzmann equation give the correct conservation laws for mass, momentum and energy; the underlying physics is not relevant if only the correct macroscopic behavior is desired. The moments are weighted integrals of the particle distribution function in velocity space, so discretization on the velocity space can be made such that the macroscopic fluid behavior is maintained without having to solve the full continuous Boltzmann equation.

The discretization of velocity space can be done either through Mach number expansion [42] or Hermite series expansion [90], though both approaches give the same form of equilibrium as Navier-Stokes. The resulting discrete velocities, $\boldsymbol{c}_i$, and corresponding weights, w_i, comprise a velocity set. The naming of the velocity sets that discretize velocity space take the form DdQq where d is the number of spatial dimensions and q is the number of discrete velocities used in the set. Commonly used velocity sets are D1Q3, D2Q9, D3Q19 and D3Q27. Figure 4.2 shows the velocities for D2Q9 and D3Q27, while Table 4.1 shows the velocity components and corresponding weights for these two velocity sets. For this research, the D3Q27 velocity set was used. This velocity set is more computationally and memory intensive than the D3Q15 and D3Q19 velocity sets, but [94] show the D3Q15 and D3Q19

velocity sets are not rotationally invariant while the D3Q27 velocity set is. When modeling high Reynolds number flows this shortcoming can spoil the results [99], making the D3Q27 velocity set the better option for modeling turbulent flow.

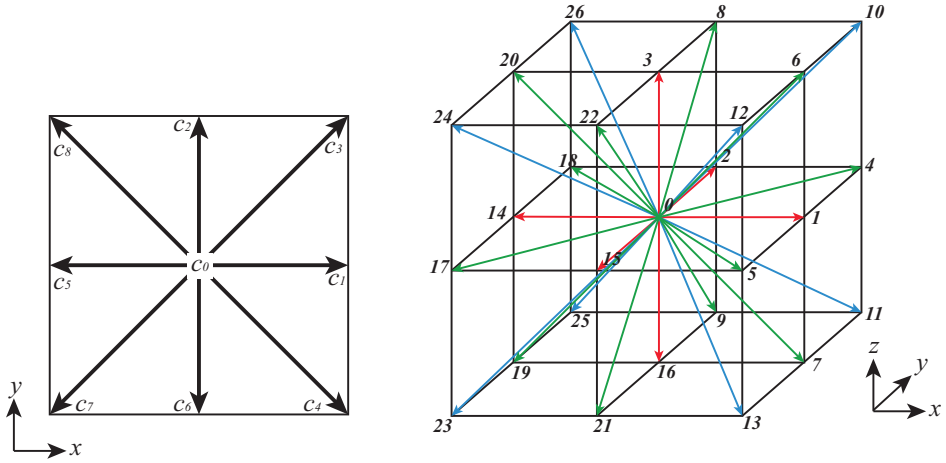


Figure 4.2: D2Q9 and D3Q27 velocity sets

Name	**Velocities $\boldsymbol{c}_i$**	**Number**	**Weight w_i**
D2Q9	$(0,0)$	1	4/9
	$(\pm1,0)$, $(0,\pm1)$	4	1/9
	$(\pm1,\pm1)$	4	1/36
D3Q27	$(0,0,0)$	1	8/27
	$(\pm1,0,0)$, $(0,\pm1,0)$, $(0,0,\pm1)$	6	2/27
	$(\pm1,\pm1,0)$, $(\pm1,0,\pm1)$, $(0,\pm1,\pm1)$	12	1/54
	$(\pm1,\pm1,\pm1)$	8	1/216

Table 4.1: D2Q9 and D3Q27 Velocity Sets

In addition, each velocity set has a constant c_s given by:

$$c_s = \frac{1}{\sqrt{3}}\frac{\Delta x}{\Delta t} \tag{4.4}$$

where Δx is the spatial discretization and Δt is the time step. For isothermal LBM, c_s determines the equation of state:

$$p = c_s^2 \rho \tag{4.5}$$

where p is the pressure and ρ is the fluid density. In this case c_s represents the model's speed of sound. This constant effectively sets an upper limit on what the maximum simulated fluid velocity can be. Krüger et al. [60] recommend that the simulated fluid velocity should be less than approximately $0.12c_s$.

4.2.2 Collision Operator

The collision operator models inter-particle collisions by redistributing particles among the different populations f_i at each point. Though the collision operator does not retain all the underlying microscopic physics, it still respects mass and momentum conservation and recovers the correct macroscopic behavior. The most common and simple collision operator is the Bhatnagar-Gross-Krook (BGK) collision operator [6].

$$\Omega_i(\boldsymbol{x}, t) = -\frac{f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t)}{\tau} \tag{4.6}$$

This equation implies that all populations f_i decay to their equilibrium state f_i^{eq} at the same rate τ. The parameter τ represents the typical time scale at which different hydrodynamic modes relax to local equilibrium [98] which is why it is called the relaxation time. The discrete form of the equilibrium distribution function f_i^{eq} is [69]:

$$f_i^{eq}(\boldsymbol{x}, t) = w_i \rho \left[1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{(\boldsymbol{c}_i \cdot \boldsymbol{u})^2}{2c_s^4} - \frac{\|\boldsymbol{u}\|^2}{2c_s^2}\right] \tag{4.7}$$

where $\boldsymbol{u}$ and ρ are the fluid velocity and density, $\boldsymbol{c}_i$ and w_i together are the discrete velocity set and c_s is the speed of sound. The macroscopic Navier-Stokes behavior is recovered when the kinematic shear viscosity ν is related to the relaxation time τ:

$$\nu = c_s^2\left(\tau - \frac{\Delta t}{2}\right) \tag{4.8}$$

Note the presence of c_s—the sound speed associated with the velocity set. This means the kinematic shear viscosity, relaxation time as well as the temporal and spatial discretization are all interdependent.

4.2.2.1 Multiple-Relaxation-Time Collision Operators

The simplicity of the BGK collision operator comes at a cost: reduced accuracy, particularly at large viscosities, and stability, particularly at small viscosities [60]. The BGK collision operator relaxes all moments at the same rate, but in principle the moments can all be relaxed at different rates. This is the idea behind the multiple-relaxation-time (MRT) collision operator [14]—all the different moments can be relaxed at different time scales to achieve better stability and accuracy. In order to do this, all the populations f_i must first be transformed to moment space, in which the collision step is performed, and then transformed back to population space, where the streaming step is performed. The MRT LBE is then:

$$|f(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t)\rangle - |f(\boldsymbol{x}, t)\rangle = -\boldsymbol{M}^{-1}\hat{\boldsymbol{S}}\boldsymbol{M}\left[|f(\boldsymbol{x}, t)\rangle - |f^{eq}(\boldsymbol{x}, t)\rangle\right] \tag{4.9}$$

where $\boldsymbol{M}$ is the transformation matrix that transforms the distributions functions from velocity space to moments space, and $\hat{\boldsymbol{S}}$ is the diagonal collision matrix: $\hat{\boldsymbol{S}} = diag(s_0, s_1, ..., s_{q-1})$. The values along the diagonal of $\hat{\boldsymbol{S}}$ are relaxation parameters for the different moments.

The transformation matrix $\boldsymbol{M}$ is a linear transformation from population space to moment space. Based on the ordering of the discrete velocities for the D3Q27 lattice used in this research, the orthogonal moment set was calculated following [25]:

$$\boldsymbol{M} = \begin{bmatrix} \langle 1| \\ \langle c_{ix}| \\ \langle c_{iy}| \\ \langle c_{iz}| \\ -2\langle| + \langle\|\boldsymbol{c}_i\|^2| \\ 2\langle c_{ix}^2| - \langle c_{iy}^2 + c_{iz}^2| \\ \langle c_{iy}^2 - c_{iz}^2| \\ \langle c_{ix}c_{iy}| \\ \langle c_{iy}c_{iz}| \\ \langle c_{ix}c_{iz}| \\ -4\langle c_{ix}| + 2\langle c_{ix}c_{iy}^2 c_{ix}c_{iz}^2| \\ -4\langle c_{iy}| + 2\langle c_{iy}c_{iz}^2 c_{iy}c_{ix}^2| \\ -4\langle c_{iz}| + 2\langle c_{iz}c_{ix}^2 c_{iz}c_{iy}^2| \\ 4\langle c_{ix}| - 6\langle c_{ix}c_{iy}^2 + c_{ix}c_{iz}^2| + 9\langle c_{ix}c_{iy}^2 c_{iz}^2| \\ 4\langle c_{iy}| - 6\langle c_{iy}c_{ix}^2 + c_{iy}c_{iz}^2| + 9\langle c_{iy}c_{ix}^2 c_{iz}^2| \\ 4\langle c_{iz}| - 6\langle c_{iz}c_{ix}^2 + c_{iz}c_{ix}^2| + 9\langle c_{iz}c_{iy}^2 c_{iz}^2| \\ 4\langle 1 - \|\boldsymbol{c}_i\|^2| + 3\langle c_{ix}^2 c_{iy}^2 + c_{iy}^2 c_{iz}^2 + c_{ix}^2 c_{iz}^2| \\ -8\langle 1| + 12\langle\|\boldsymbol{c}_i\|^2| - 18\langle c_{ix}^2 c_{iy}^2 + c_{iy}^2 c_{iz}^2 + c_{ix}^2 c_{iz}^2| + 27\langle c_{ix}^2 c_{iy}^2 c_{iz}^2| \\ 2\langle c_{iz}^2 + c_{iy}^2| + 3\langle c_{ix}^2 c_{iy}^2 + c_{ix}^2 c_{iz}^2| - 4\langle c_{ix}^2| - 6\langle c_{iy}^2 c_{iz}^2| \\ 2\langle c_{iz}^2 - c_{iy}^2| + 3\langle c_{ix}^2 c_{iy}^2 - c_{ix}^2 c_{iz}^2| \\ -2\langle c_{ix}c_{iy}| + 3\langle c_{ix}c_{iy}c_{iz}^2| \\ -2\langle c_{iy}c_{iz}| + 3\langle c_{ix}^2 c_{iy}c_{iz}| \\ -2\langle c_{ix}c_{iz}| + 3\langle c_{ix}c_{iy}^2 c_{iz}| \\ \langle c_{ix}c_{iy}^2 - c_{ix}c_{iz}^2| \\ \langle c_{iy}c_{iz}^2 - c_{iy}c_{ix}^2| \\ \langle c_{iz}c_{ix}^2 - c_{iz}c_{iy}^2| \\ \langle c_{ix}c_{iy}c_{iz}| \end{bmatrix} \tag{4.10}$$

Appendix A contains the calculated values for $\boldsymbol{M}$ and $\boldsymbol{M}^{-1}$. The values along the diagonal of $\hat{\boldsymbol{S}}$ relax the moments to equilibrium at different time scales. For the D3Q27 lattice, this gives:

$$\begin{aligned} \hat{\boldsymbol{S}} = diag(&0, 0, 0, 0, s_4, s_5, s_5, s_7, s_7, s_7, s_{10}, s_{10}, s_{10}, s_{13}, s_{13}, s_{13}, \\ &s_{16}, s_{17}, s_{18}, s_{18}, s_{20}, s_{20}, s_{20}, s_{23}, s_{23}, s_{23}, s_{26}) \end{aligned} \tag{4.11}$$

In this research the parameters proposed by [99] were used:

$$\begin{aligned} &s_4 = 1.54, \quad s_{10} = 1.5, \quad s_{13} = 1.83, \quad s_{16} = 1.4, \\ &s_{17} = 1.61, \quad s_{18} = s_{20} = 1.98, \quad s_{23} = s_{26} = 1.74 \end{aligned} \tag{4.12}$$

As with the BGK collision operator, it is necessary to relate the kinematic viscosity ν to the the corresponding components of $\hat{\boldsymbol{S}}$ [99]:

$$\nu = c_s^2 \left(\frac{1}{s_5} - \frac{1}{2} \right) \Delta t = c_s^2 \left(\frac{1}{s_7} - \frac{1}{2} \right) \Delta t \tag{4.13}$$

4.2.3 Body Forces

Forces can be included in the LBE by discretizing the continuous Boltzmann equation with a forcing term. By discretizing velocity space, physical space and time, the LBE with forces included then becomes:

$$f_i \left(\boldsymbol{x} + \boldsymbol{c}_i \Delta t, t + \Delta t \right) - f_i \left(\boldsymbol{x}, t \right) = \Omega_i \left(\boldsymbol{x}, t \right) + S_i \left(\boldsymbol{x}, t \right) \tag{4.14}$$

where the inclusion of a body force manifests itself as a source term, S_i. Huang et al. [52] compared several different forcing schemes [[89], [69], [43], [39], [62], [61]] for including body forces in LBM and proved they are identical up to second order. Additionally, the different schemes demonstrated comparable accuracy for single-phase flows. Given these observations, the forcing scheme proposed in [39] was used in this research as it was the simplest to build into the simulation software. Following this scheme, the equilibrium velocity is defined as:

$$\boldsymbol{u}_{eq} = \frac{1}{\rho} \sum_i f_i \boldsymbol{c}_i + \frac{\boldsymbol{F} \Delta t}{2\rho} \tag{4.15}$$

where $\boldsymbol{F}$ is the force density; $\boldsymbol{F} = \rho \boldsymbol{g}$ in the case of a gravitational force. The forcing source term takes the following form:

$$S_i = \left(1 - \frac{\Delta t}{2\tau} \right) w_i \left(\frac{\boldsymbol{c}_i - \boldsymbol{u}}{c_s^2} + \frac{\left(\boldsymbol{c}_i \cdot \boldsymbol{u} \right) \boldsymbol{c}_i}{c_s^4} \right) \cdot \boldsymbol{F} \tag{4.16}$$

where $\boldsymbol{u}$ is the fluid velocity and τ is the BGK relaxation time. The inclusion of relaxation parameters based on an MRT collision operator was done based on [67]. In the forcing scheme proposed by Guo et al. [39], the macroscopic fluid velocity is calculated by:

$$\boldsymbol{u} = \frac{1}{\rho} \sum_i f_i \boldsymbol{c}_i + \frac{\boldsymbol{F} \Delta t}{2\rho} \tag{4.17}$$

4.2.4 Boundary Conditions

Inclusion of boundary conditions is a necessity for solving partial differential equations. Though they only apply over a small portion of the domain, their presence may affect the entire solution. Boundary conditions in the Lattice Boltzmann Method do not manifest as naturally as they do in other CFD methods such as FEM, since the mesoscopic distribution

functions f_i must be specified based on macroscopic variables ρ and $\boldsymbol{u}$. In the case of the D3Q27 lattice, this means that 27 degrees of freedom are specified based on only 4 input variables. Due to this under-specified nature at boundaries, there are many implementations of boundary conditions available in LBM.

4.2.4.1 Periodic Boundaries

Periodic boundaries are the simplest type of boundary condition within LBM. This type of boundary is useful in minimizing the size of the simulation domain when isolating repeating flow patterns and, from a more pragmatic perspective, simulating two-dimensional flow using three-dimensional software—for example, Couette or Poiseuille flow. For periodic boundaries, flow leaving one side of the domain instantaneously reenters the domain on the opposite side. As such, both mass and momentum are conserved.

Implementing periodic boundaries in LBM is relatively straightforward [[98], [100]]: the collision step is unchanged and streaming is achieved similarly to the bulk domain. For the streaming step, nodes on a periodic boundary exchange populations with interior nodes as they normally would; however, the populations that stream out of the domain are exchanged with nodes on the opposite side of the domain.

4.2.4.2 Solid Boundaries

Solid boundaries mark the interface between fluid and solid, and most commonly this interface is described using a no-slip condition. In LBM, this type of boundary is modeled using the so-called bounce-back method [[10], [119], [32], [63], [44]]. The basic principle is that populations hitting a solid boundary are reflected, or bounced back, in the direction in which they originally came from. This concept is illustrated in Figure 4.3. For full-way bounce-back [[100], [60]], the collision step for solid boundaries is then described by:

$$f_{-i}\left(\boldsymbol{x}, t\right) = f_i\left(\boldsymbol{x}, t\right) \tag{4.18}$$

where f_{-i} indicates the population is reflected back with velocity $-\boldsymbol{c}_i$. The location of the boundary is midway between the boundary and interior nodes, though this location is only approximate when using the BGK collision operator—the exact location of the boundary is viscosity dependent [60]. In order to avoid this viscosity dependence, an MRT collision operator is used instead of the BGK model which has only one relaxation time.

4.2.4.3 Open Boundaries

Open boundaries comprise inlets, where fluid enters the domain, and outlets, where fluid exits the domain. These types of boundaries are useful in shrinking the size of the domain that is to be simulated—a velocity or pressure profile can be introduced that imposes the expected upstream behavior and allows fluid to leave the domain downstream.

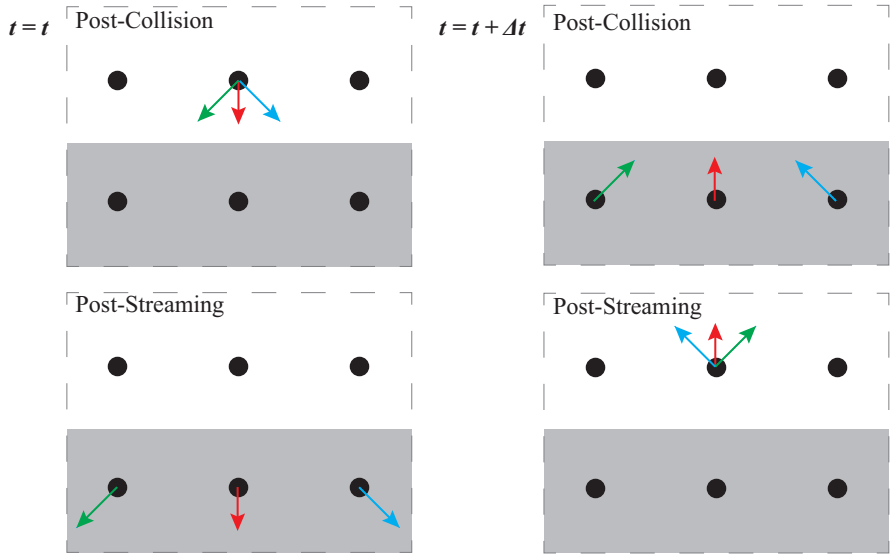


Figure 4.3: Illustration of the bounce-back boundary condition. Boundary located midway between fluid and solid nodes. Shaded areas indicate solid region. (Modified, based on [100] and [60])

Velocity Boundaries

The fluid velocity at a boundary can be specified using the bounce-back method [62] as follows:

$$f_{-i}\left(\boldsymbol{x}_b, t+\Delta t\right) = f_i\left(\boldsymbol{x}_b, t\right) - 2w_i\rho_b\frac{\boldsymbol{c}_i\cdot\boldsymbol{u}_b}{c_s^2} \tag{4.19}$$

where $\boldsymbol{u}_b$ is the velocity of the fluid at the boundary and ρ_b is the density of the fluid at the boundary. Generally, if the velocity is specified the density will be unknown—standard LBM is weakly compressible. To solve this problem, the density at the boundary can either be taken as the system's average density or as the local fluid density [60]. In this research, the local density was approximated using extrapolation.

Pressure Boundaries

Pressure boundaries are specified using the anti-bounce-back method [33]:

$$f_{-i}\left(\boldsymbol{x}_b, t+\Delta t\right) = -f_i\left(\boldsymbol{x}_b, t\right) + 2w_i\rho_b\left[1 + \frac{\left(\boldsymbol{c}_i\cdot\boldsymbol{u}_b\right)^2}{2c_s^4} - \frac{\|\boldsymbol{u}_b\|^2}{2c_s^2}\right] \tag{4.20}$$

where ρ_b is the density of the fluid at the boundary and $\boldsymbol{u}_b$ is the velocity of the fluid at the boundary. The specified pressure is related to the density using the equation of state $p = c_s^2\rho$. The input pressure is understood as a difference from a reference pressure,

therefore the density calculated from the equation of state is added to the average density at the reference pressure. Since the pressure is specified, the fluid velocity is generally not known and poses a similar problem as seen with velocity boundaries. In this research, the fluid velocity is estimated using extrapolation.

Non-Reflecting Boundaries

In its formulation, the LBM is weakly compressible—the macroscopic behavior of the model follows the compressible Navier-Stokes equations. As a consequence of this formulation, initialization of the velocity and density fields may introduce undesired pressure waves in the solution domain. Enforcing the velocity or pressure at the boundaries using the simple bounce-back techniques will trap these initial transients, reflecing the waves back into the domain and potentially spoiling the entire solution. This is especially apparent at outlets.

Characteristic boundary conditions adapted for LBM [[54], [47]] minimize this reflection of pressure waves back into the domain. For this research, the methodology presented in [47] has been extended to consider a three dimensionsal domain as well as the presence of a body force. The effect of viscosity is neglected near the boundaries, yielding the three-dimensional Euler conservation equations with an external force:

$$\partial_t \boldsymbol{m} + \boldsymbol{A}\partial_x \boldsymbol{m} + \boldsymbol{B}\partial_y \boldsymbol{m} + \boldsymbol{C}\partial_z \boldsymbol{m} = \boldsymbol{g} \tag{4.21}$$

where $\boldsymbol{m}^T = (\rho, u_x, u_y, u_x)$ is used for the characteristic variables, $\boldsymbol{g}^T = (0, g_x, g_y, g_z)$ for the external force and the coefficient matrices:

$$\boldsymbol{A} = \begin{bmatrix} u_x & \rho & 0 & 0 \\ \frac{c_s^2}{\rho} & u_x & 0 & 0 \\ 0 & 0 & u_x & 0 \\ 0 & 0 & 0 & u_x \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} u_y & 0 & \rho & 0 \\ 0 & u_y & 0 & 0 \\ \frac{c_s^2}{\rho} & 0 & u_y & 0 \\ 0 & 0 & 0 & u_y \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} u_z & 0 & 0 & \rho \\ 0 & u_z & 0 & 0 \\ 0 & 0 & u_z & 0 \\ \frac{c_s^2}{\rho} & 0 & 0 & u_z \end{bmatrix}, \tag{4.22}$$

Equation 4.21 is a hyberbolic system of equations, therefore the coefficient matrices are diagonalizable:

$$\boldsymbol{\Omega_x} = \boldsymbol{S}\boldsymbol{A}\boldsymbol{S}^{-1}, \quad \boldsymbol{\Omega_y} = \boldsymbol{T}\boldsymbol{B}\boldsymbol{T}^{-1}, \quad \boldsymbol{\Omega_z} = \boldsymbol{U}\boldsymbol{C}\boldsymbol{U}^{-1} \tag{4.23}$$

where Ω_x, Ω_y and Ω_z contain the eigenvalues of $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$:

$$\begin{aligned} \boldsymbol{\Omega_x} &= diag\left(\lambda_{x,1}, \lambda_{x,2}, \lambda_{x,3}, \lambda_{x,4}\right) = diag\left(u_x - c_s, u_x, u_x, u_x + c_s\right), \\ \boldsymbol{\Omega_y} &= diag\left(\lambda_{y,1}, \lambda_{y,2}, \lambda_{y,3}, \lambda_{y,4}\right) = diag\left(u_y - c_s, u_y, u_y, u_y + c_s\right), \\ \boldsymbol{\Omega_z} &= diag\left(\lambda_{z,1}, \lambda_{z,2}, \lambda_{z,3}, \lambda_{z,4}\right) = diag\left(u_z - c_s, u_z, u_z, u_z + c_s\right) \end{aligned} \tag{4.24}$$

One set of diagonalization matrices, essentially extending the choice of [47] to three-dimensions, is:

$$\boldsymbol{S} = \begin{bmatrix} c_s^2 & -c_s\rho & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ c_s^2 & c_s\rho & 0 & 0 \end{bmatrix}, \quad \boldsymbol{S^{-1}} = \begin{bmatrix} \frac{1}{2c_s^2} & 0 & 0 & \frac{1}{2c_s^2} \\ \frac{-1}{2c_s\rho} & 0 & 0 & \frac{1}{2c_s\rho} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

$$\boldsymbol{T} = \begin{bmatrix} c_s^2 & 0 & -c_s\rho & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ c_s^2 & 0 & c_s\rho & 0 \end{bmatrix}, \quad \boldsymbol{T^{-1}} = \begin{bmatrix} \frac{1}{2c_s^2} & 0 & 0 & \frac{1}{2c_s^2} \\ 0 & 1 & 0 & 0 \\ \frac{-1}{2c_s\rho} & 0 & 0 & \frac{1}{2c_s\rho} \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{4.25}$$

$$\boldsymbol{U} = \begin{bmatrix} c_s^2 & 0 & 0 & -c_s\rho \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ c_s^2 & 0 & 0 & c_s\rho \end{bmatrix}, \quad \boldsymbol{U^{-1}} = \begin{bmatrix} \frac{1}{2c_s^2} & 0 & 0 & \frac{1}{2c_s^2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{-1}{2c_s\rho} & 0 & 0 & \frac{1}{2c_s\rho} \end{bmatrix}$$

This diagonalization allows incoming and outgoing waves to be separated based on the eigenvalues and eigenvectors; the separation makes it possible extinguish incoming waves. The meaning of the diagonalization and how it is used to eliminate incoming waves can be seen more clearly when considering a particular case along a z-boundary. In this case, the spatial derivates become:

$$\boldsymbol{C}\partial_z\boldsymbol{m} = \boldsymbol{U}^{-1}\boldsymbol{\Omega}_z\boldsymbol{U}\partial_z\boldsymbol{m} = \boldsymbol{U}^{-1}\mathscr{L}_z \tag{4.26}$$

where

$$\mathscr{L}_z = \begin{pmatrix} \mathscr{L}_{z,1} \\ \mathscr{L}_{z,2} \\ \mathscr{L}_{z,3} \\ \mathscr{L}_{z,4} \end{pmatrix} \tag{4.27}$$

With this description, the components $\mathscr{L}_{z,i} = \lambda_{z,i}S_{ij}\partial_z m_j$ express the characteristic wave amplitude variations [85]. The sign of $\lambda_{z,i}$ indicates the direction of the wave—$\lambda_{z,1} = u_z - c_s$ is outgoing on the negative z-boundary and incoming on the positive z-boundary. Following the idea of [102] as outlined in [47], incoming waves are annihilated by setting their wave amplitude variations to zero:

$$\mathscr{L}_{z,i} = \begin{cases} \lambda_{z,i}S_{ij}\partial_z m_j & \text{for outgoing waves} \\ 0 & \text{for incoming waves} \end{cases} \tag{4.28}$$

The values of $\boldsymbol{m}$ are then obtained by solving a modified version of Equation 4.21:

$$\begin{cases} \partial_t \boldsymbol{m} + \gamma \boldsymbol{B}\partial_y \boldsymbol{m} + \gamma \boldsymbol{C}\partial_z \boldsymbol{m} = -\boldsymbol{S}^{-1}\mathscr{L}_x + \boldsymbol{g} \text{ along the x-boundary} \\ \partial_t \boldsymbol{m} + \gamma \boldsymbol{A}\partial_x \boldsymbol{m} + \gamma \boldsymbol{C}\partial_z \boldsymbol{m} = -\boldsymbol{T}^{-1}\mathscr{L}_y + \boldsymbol{g} \text{ along the y-boundary} \\ \partial_t \boldsymbol{m} + \gamma \boldsymbol{A}\partial_x \boldsymbol{m} + \gamma \boldsymbol{B}\partial_y \boldsymbol{m} = -\boldsymbol{U}^{-1}\mathscr{L}_z + \boldsymbol{g} \text{ along the z-boundary} \end{cases} \tag{4.29}$$

where γ is a factor introduced by [47]. When $\gamma = 1$, the two-dimensional case recovers [102] while $\gamma = 0$ will give the one-dimensional approach of [54]. [47] found $\gamma = 3/4$ to be superior; this is the value that was used in this research. In the case of edges or corners, the characteristic analysis is done for two or three directions, respectively. For example, Equation 4.21 becomes:

$$\partial_t \boldsymbol{m} + \gamma \boldsymbol{C}\partial_z \boldsymbol{m} = -\boldsymbol{S}^{-1}\mathscr{L}_x \quad -\boldsymbol{T}^{-1}\mathscr{L}_y + \boldsymbol{g} \tag{4.30}$$

along an x-y edge and

$$\partial_t \boldsymbol{m} = -\boldsymbol{S}^{-1}\mathscr{L}_x \quad -\boldsymbol{T}^{-1}\mathscr{L}_y - \boldsymbol{U}^{-1}\mathscr{L}_z + \boldsymbol{g} \tag{4.31}$$

at the corners.

Before the required values of the density and velocity at the next time step can be calculated, the spatial derivatives $\partial_x \boldsymbol{m}$, $\partial_y \boldsymbol{m}$ and $\partial_z \boldsymbol{m}$ must first be approximated. Second-order finite differences are used based on neighboring values in the lattice[47]:

$$\partial_x m_i(x) \approx \frac{\mp m_i(x) \pm 4m_i(x \pm \Delta x) \mp m_i(x \pm 2\Delta x)}{2\Delta x} \tag{4.32}$$

for derivatives perpendicular to the boundary. The upper signs are for a forward difference and the lower signs are for a backward difference. Derivatives parallel to the boundary are approximated by:

$$\partial_x m_i(x) \approx \frac{m_i(x + \Delta x) - m_i(x - \Delta x)}{2\Delta x} \tag{4.33}$$

With this it is now possible to calculate an estimate of $\partial_t \boldsymbol{m}$. The macroscopic variables at the next time step are then estimated using a forward Euler time integrator:

$$m_i(\boldsymbol{x}, t + \Delta t) \approx m_i(\boldsymbol{x}, t) + \Delta t \partial_t m_i(\boldsymbol{x}, t) \tag{4.34}$$

Though it is possible to implement a higher order time integrator, it has been shown that higher order methods [47] perform nearly identically to the less computationally expensive forward Euler—the error for non-reflecting characteristic boundary conditions is dominated by other sources.

With the known values of ρ and $\boldsymbol{u}$ at the next time step, the values of the distribution functions f_i can be set accordingly. This is implemented as follows: During the collision step, no actual collision occurs. Instead, the values for ρ and $\boldsymbol{u}$ for the next time step are estimated

after which the values of f_i are set according to their equilibrium values determined by the pre-calculated ρ and $\boldsymbol{u}$.

4.2.5 Turbulence

The standard LBM is restricted to flows with low Reynolds numbers and requires special treatment if modeling turbulent flows. In the case of turbulent flows, the grid scale may not be sufficiently small to capture all scales of different flow features. Large Eddy Simulation (LES) is a popular approach to account for this large difference in the scale of flow structures—eddies greater than the grid scale are solved for directly while subgrid-scale flow structures are accounted for through a subgrid-scale (SGS) eddy viscosity, ν_{SGS}. This additional turbulent viscosity allows for energy to be transferred between resolved and unresolved scales. In this research, ν_{SGS} is calculated using the Wall-Adapting Local Eddy-visocity (WALE) model [77] which is based on the square of the velocity gradient tensor. Compared to the classical Smagorinsky approach [96], the WALE model includes both stain and rotation rates and the eddy viscosity goes naturally to zero in the vicinity of walls.

For the D3Q27 MRT model used in this research, the subgrid-scale eddy viscosity is added to the kinematic viscosity as follows [99]:

$$\nu + \nu_{SGS} = c_s^2 \left(\frac{1}{s_5} - \frac{1}{2} \right) \Delta t = c_s^2 \left(\frac{1}{s_7} - \frac{1}{2} \right) \Delta t \tag{4.35}$$

The subgrid-scale eddy viscosity based on the WALE model is:

$$\nu_{SGS} = (C_w \Delta x)^2 \frac{\left(\mathscr{S}_{ij}^d \mathscr{S}_{ij}^d \right)^{3/2}}{\left(\overline{S}_{ij} \overline{S}_{ij} \right)^{5/2} + \left(\mathscr{S}_{ij}^d \mathscr{S}_{ij}^d \right)^{5/4}} \tag{4.36}$$

where the eddy viscosity constant C_w is 0.32 and the overbar denotes a filtered velocity field. As shown in Equation 4.36, the SGS eddy viscosity depends on the deformation tensor

$$\overline{S}_{ij} = \frac{1}{2} \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right) \tag{4.37}$$

and the traceless symmetric part of the square of the velocity gradient tensor

$$\mathscr{S}_{ij}^d = \frac{1}{2} \left(\overline{g}_{ij}^2 + \overline{g}_{ji}^2 \right) - \frac{1}{3} \delta_{ij} \overline{g}_{kk}^2 \tag{4.38}$$

where $\overline{g}_{ij} = \frac{\partial \overline{u}_i}{\partial x_j}$ is the velocity gradient tensor and $\overline{g}_{ij}^2 = \overline{g}_{ik} \overline{g}_{kj}$.

4.2.6 LBM Implementation Verification

The Lattice Boltzmann method and all of the above-mentioned features were implemented in C++. Since the resolution required to resolve the fluid pressures acting on individual blocks requires many more fluid nodes than solid particles, at present only the fluid

computations have been accelerated using parallel computing. The parallelization was performed using `Kokkos` [15], a C++ library aimed at performance portability. By abstracting both parallel execution and memory layout, the same source code can be compiled to target a particular computing architecture based on the machine where the program will be executed. If the underlying limitations of graphics processing units (GPUs) are respected in the way that the code is written, code accelerated with `Kokkos` can be executed in parallel on both central processing units (CPU) and GPUs.

As with the DEM portion of the software, all individual pieces of the code were unit tested using `CxxTest` [106]. Integration testing was then performed to verify the software is able to match analytical solutions in fluid dynamics for both transient and steady-state conditions.

4.2.6.1 Couette Flow

Couette flow is a particular case of simple parallel-plate flow where one plate is fixed while the other plate moves with a constant velocity while a viscous fluid fills the space between the two plates. Figure 4.4 illustrates this situation and shows the steady-state solution. Couette flow demonstrates shear-driven fluid motion and, by modeling this numerically, the ability of the LBM implementation to correctly capture the viscous behavior of fluid.

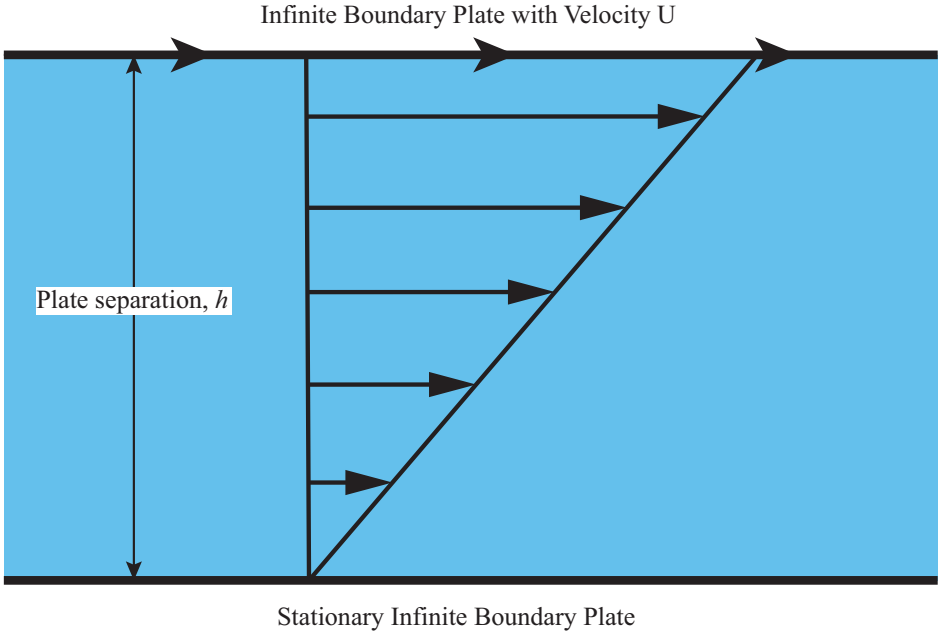


Figure 4.4: Steady-state velocity profile for Couette flow with two infinite plates separated by distance h. The top plate moves with constant horizontal velocity U while the bottom plate is a stationary, no-slip boundary.

The PDE describing the evolution of velocity between the two parallel plates is [5]:

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial y^2} \tag{4.39}$$

with initial condition and boundary conditions:

$$\begin{aligned} u(y,0) &= 0, \;\; 0 < y < h \\ u(0,t) &= 0, \;\; t \geq 0 \\ u(h,t) &= U, \;\; t \geq 0 \end{aligned} \tag{4.40}$$

The solution to this PDE is given by [64]:

$$u\left(x,t\right) = -\frac{2U}{h} \sum_{n=1}^{\infty} \frac{h}{n\pi} e^{-\nu\left(\frac{n\pi}{h}\right)^2 t} sin\left(\frac{n\pi}{h} y\right) + \frac{U}{h}\left(h - y\right) \tag{4.41}$$

Figure 4.5 shows the comparison of numerical results with the analytical solution at different times. The numerical results match the analytical solution very well and illustrate the ability of the LBM implementation to capture shear-driven flow correctly.

4.2.6.2 Gravity-Driven Plane Poiseuille Flow

The validity of the body force implementation is tested by modeling gravity-driven plane Poiseuille flow. For this type of flow two infinite plate are separated by a constant distance, as shown in Figure 4.6, with a body force $\boldsymbol{g}$ acting on the fluid. The PDE describing this type of flow is given by:

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial y^2} + \boldsymbol{g} \tag{4.42}$$

with initial condition and boundary conditions:

$$\begin{aligned} u(y,0) &= 0, \;\; 0 < y < h \\ u(0,t) &= 0, \;\; t \geq 0 \\ u(h,t) &= 0, \;\; t \geq 0 \end{aligned} \tag{4.43}$$

The solution for gravity-driven plane Poiseuille flow is given by[21]:

$$u^*(y^*,t^*) = \frac{P^*}{2}\left[\left(1 - y^{*2}\right) - 4\sum_{n=0}^{\infty}(-1)^n \frac{cos\left(\pi y^*\left(n+\frac{1}{2}\right)\right)}{\pi^3\left(n+\frac{1}{2}\right)^3} e^{-\pi^2 t^*\left(n+\frac{1}{2}\right)^2}\right] \tag{4.44}$$

where:

$$t^* = \frac{t\nu}{h^2}, \;\; u^* = \frac{u}{U}, \;\; y^* = \frac{y}{h}, \;\; P^* = \frac{gh^2}{\nu U} \tag{4.45}$$

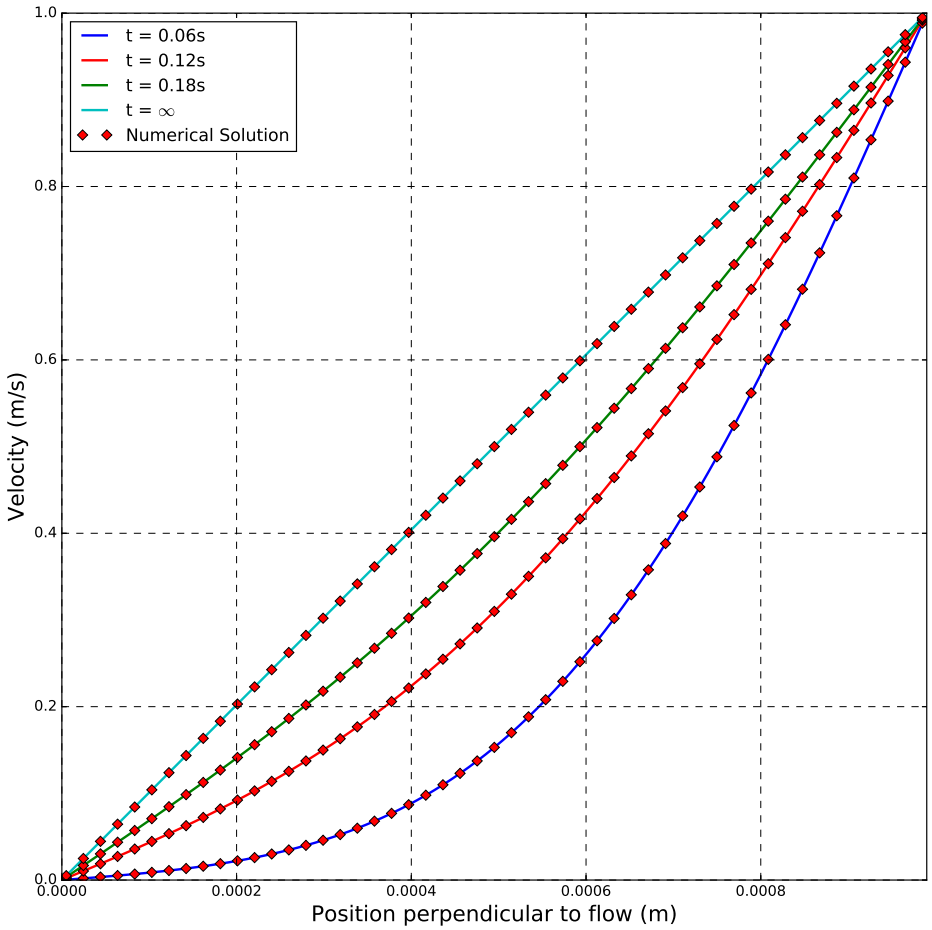


Figure 4.5: Comparison of numerical results with analytical solution for Couette flow. Solid lines indicate analytical solution at different times while red diamonds show numerical results at corresponding times.

The comparison between the analytical solution and numerical results are shown in Figure 4.7. This shows excellent agreement between the numerical results and the analytical solution at various stages as the flow evolves.

4.3 Fluid-Solid Coupling

Describing the interaction between the fluid and solid phases requires the two separate models, DEM and LBM, to exchange information. The presence of solids in the fluid mesh has to be factored into the fluid response, while the hydrodynamic forces and moments acting on the solids need to be included in the equations of motion for the individual blocks. The boundary conditions described in Section 4.2.4 are generally applicable for more simple geometries that do not move through the domain and would not be able to model com-

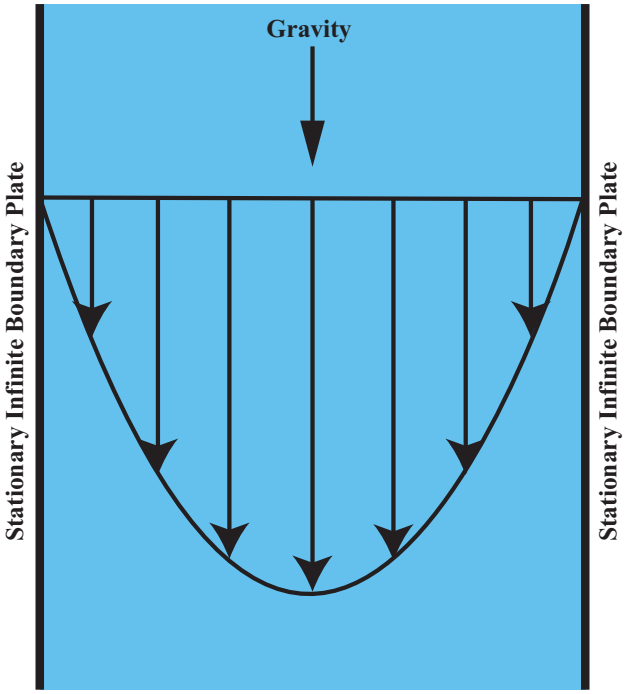


Figure 4.6: Steady-state velocity profile for gravity-driven plane Poiseuille flow with two infinite plates separated by distance h. Both boundary plates are stationary, no-slip boundary conditions

plex polyhedral blocks moving through the fluid domain. Instead, the coupling process is achieved though the introduction of a special boundary condition—partially saturated bounce-back—that allows the polyhedral blocks to move through the fluid mesh while still maintaining a similar form of the LBE.

First introduced by Noble and Torczynski [78], the partially saturated method or gray LBM accounts for the presence of complex shaped solids within the fluid mesh by considering the volumetric solid content of each of the lattice cells. As a rock block moves through the fluid mesh, it may partially or completely cover fluid cells, as shown in Figure 4.8. The LBE with the BGK collision operator is modified to accommodate the solid phase by introducing an additional solid collision operator:

$$f_i\left(\boldsymbol{x}+\boldsymbol{c}_i\Delta t, t+\Delta t\right) = f_i\left(\boldsymbol{x}, t\right) + \left[1-\sum_s B\left(\epsilon_s,\tau\right)\right]\Omega_i^{BGK} + \sum_s B\left(\epsilon_s,\tau\right)\Omega_i^s \tag{4.46}$$

where ϵ_s is the volumetric solid fraction for each block intersecting the fluid node and $B\left(\epsilon_s,\tau\right)$ is a weighting function. $B\left(\epsilon_s,\tau\right)$ ranges from 0 (pure fluid) to 1 (pure solid). When $B\left(\epsilon_s,\tau\right)=0$ the standard LBE is recovered, while for $B\left(\epsilon_s,\tau\right)=1$ only the solid collision operator participates in the collision step. The collision operator for solid nodes is:

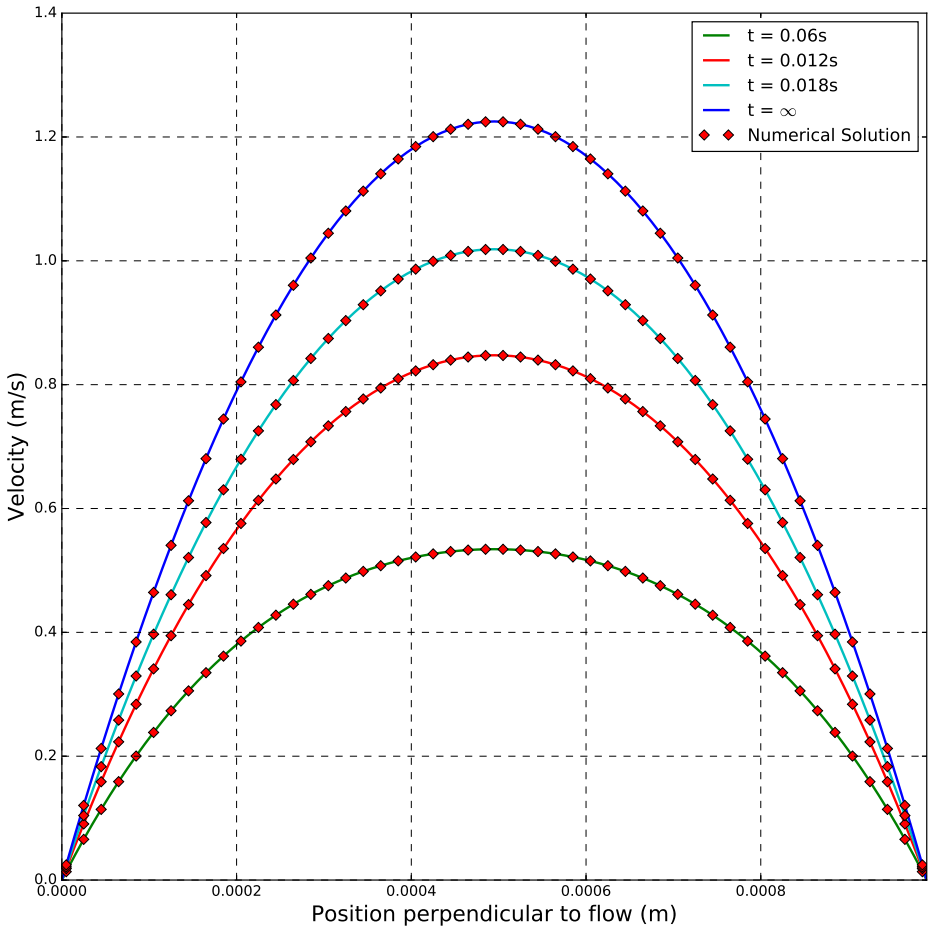


Figure 4.7: Comparison of numerical results with analytical solution for gravity-driven plane Poiseuille flow. Solid lines indicate analytical solution at different times while red diamonds show numerical results at corresponding times.

$$\Omega_i^s = f_{-i}\left(\boldsymbol{x}, t\right) - f_i\left(\boldsymbol{x}, t\right) + f_i^{eq}\left(\rho, \boldsymbol{u}_s\right) - f_{-i}^{eq}\left(\rho, \boldsymbol{u}\right) \tag{4.47}$$

where $\boldsymbol{u}_s$ is the velocity of the solid particle at time $t + \Delta t$ at the node. This form of Ω_i^s is based on the bounce-back of the non-equilibrium portion of the particle distributions [122]. The weighting function is expressed as:

$$B\left(\epsilon_s, \tau\right) = \frac{\epsilon_s\left(\tau/\Delta t - 1/2\right)}{\left(1 + \epsilon_s\right) + \left(\tau/\Delta t - 1/2\right)} \tag{4.48}$$

The hydrodynamic force and torque acting on a block moving through the fluid mesh is calculated as:

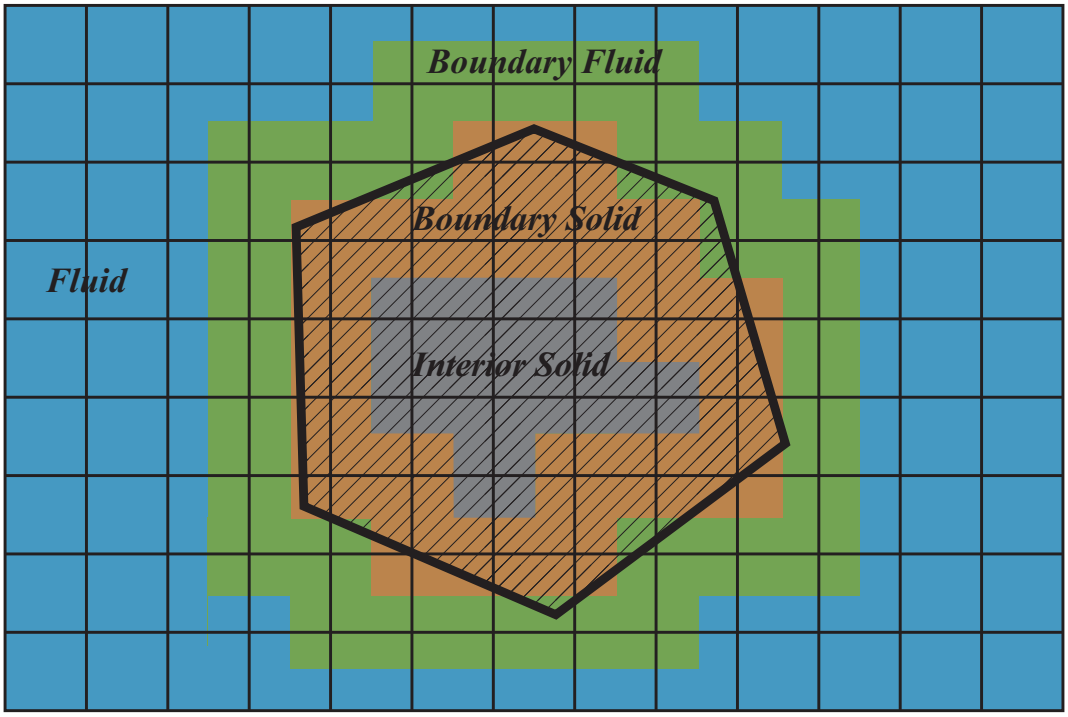


Figure 4.8: Polyhedral block moving through fluid mesh. Background grid indicates lattice nodes which are at the center of fluid cells. Fluid cells are pure fluid, interior solid cells are pure solid. Boundary fluid and boundary solid cells are some proportion of fluid and solid

$$\begin{aligned} \boldsymbol{F} &= \frac{\Delta x^3}{\Delta t} \sum_{\boldsymbol{x}_n} B\left(\boldsymbol{x}_n\right) \left(\sum_i \Omega_i^s \boldsymbol{c}_i \right) \\ \boldsymbol{T} &= \frac{\Delta x^3}{\Delta t} \sum_{\boldsymbol{x}_n} \left(B\left(\boldsymbol{x}_n\right) \left(\boldsymbol{x}_n - \boldsymbol{x}_{CM}\right) \times \sum_i \Omega_i^s \boldsymbol{c}_i \right) \end{aligned} \tag{4.49}$$

where $\boldsymbol{x}_n$ are all the lattice nodes that are interacting with the block and $\boldsymbol{x}_{CM}$ is the location of the center of mass of the block. The summation i runs over all directions of the particular lattice velocity set in use—27 in the case of this research.

4.3.1 Volumetric Solid Fraction

The behavior of the fluid-solid interaction in the case when a fluid cell is neither pure fluid or pure solid—boundary fluid or boundary solid cells as shown in Figure 4.8—is dictated by the weighting factor B. The value of the weighing function is influenced by the collision operator parameters and, to a greater extent, the volumetric solid content of the cell. Therefore, it is important that the volumetric solid content be calculated as accurately and efficiently as possible. In terms of calculating the volumetric solid fraction, current cou-

pling algorithms approximate the shape of the particles in terms of the shape of the region of overlap—[81] for spherical particles and [88] for sphero-polyhedra. For this research, a new methodology was developed to calculate the volumetric solid content analytically for convex polyhedra to determine the value of the weighting function. This new method was applied using an MRT collision operator.

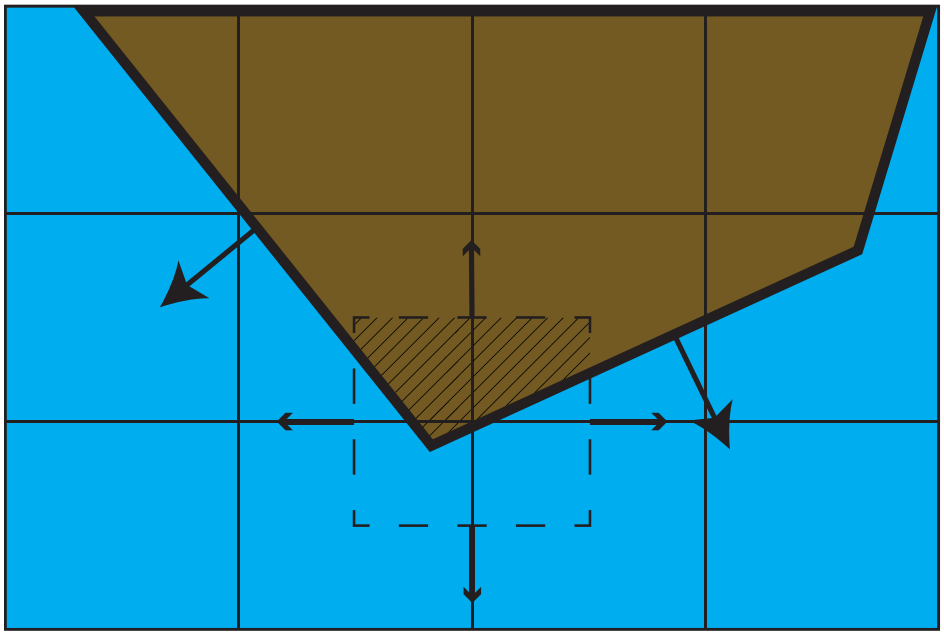


Figure 4.9: Closeup of polyhedral block overlapping fluid cell. The fluid cell is described by the normals to the cell faces and their distance from the lattice node at the center of the cell. The hatched region is where the block and fluid cell overlap—this is the solid content of the fluid cell.

The first step in calculating the volumetric solid content is to determine whether the block and fluid cell overlap, as shown in Figure 4.9. This is essentially the same problem as establishing contact between two polyhedral blocks. Thus, for this phase of the calculations methods used in contact detection for DEM can also be applied to establish whether a block intrudes on a fluid cell. Here, a linear programming approach [7] is used to establish overlap:

$$\begin{gathered} \text{minimize s} \\ \boldsymbol{a_i}^T \boldsymbol{x} - d_i \leq s, \;\; i = 1, ..., N_S + N_F \end{gathered} \tag{4.50}$$

where N_S are the faces that define the block and N_F are the faces that define the fluid cell—four faces for a square in two dimensions and six faces for a cube in three dimensions. $\boldsymbol{a}_i$ is the normal vector to the the face and d_i represents the distance of that face from some local origin. The same tolerance criteria as described in [7] is used here to establish overlap—the block and fluid cell overlap if $s < -\epsilon$ where ϵ is a specified numerical tolerance. If the block and fluid cell do not overlap then the cell is pure fluid with a volumetric solid

content of zero. However, if the block and fluid cell do overlap, it is necessary to do further computations to determine the volumetric solid content.

The region of overlap between the block and fluid cell, shown for the two dimensional case in Figure 4.9, comprises the solid content. This region is described by a subset of the faces that define the block and fluid cell and with this subset it is possible to calculate the solid content. The minimal set of faces that describe this region is established by checking all faces of the block and fluid cell for redundancy. A particular face $\boldsymbol{n}^T\boldsymbol{x} \leq d$ from the set $N_S + N_F$ is checked for redundancy by solving the linear program:

$$\begin{aligned} &\text{maximize } \boldsymbol{n}^T\boldsymbol{x} \\ &\boldsymbol{a}_i\boldsymbol{x} \leq d_i,\ i = 1, ..., N_S + N_F \end{aligned} \tag{4.51}$$

The face is not redundant if $|\boldsymbol{n}^T\boldsymbol{x} - d| < \epsilon$. This approach is similar to the removal of redundant faces during fractured rock mass generation as described in [8].

With this minimal set, it is now possible to calculate the volumetric solid fraction in the fluid cell. The volume of the region described by this minimal set is calculated using simplex integration [91]:

$$\begin{aligned} V_{overlap} &= \int\int\int_V dx\, dy\, dz \\ &= \sum_{i=1}^{s}\sum_{k=1}^{n(i)} S_{P_0P_1^iP_k^iP_{k+1}^i}(0,0,0) \\ &= \frac{1}{6}\sum_{i=1}^{s}\sum_{k=1}^{n(i)} \begin{vmatrix} x_1^i & y_1^i & z_1^i \\ x_k^i & y_k^i & z_k^i \\ x_{k+1}^i & y_{k+1}^i & z_{k+1}^i \end{vmatrix} \end{aligned} \tag{4.52}$$

Equation 4.52 describes the summation of the volumes of tetrahedra S_i—a tetrahedron is a three-dimensional simplex—that together form the three-dimensional block. This assumes that the vertices $P_1, ..., P_n$ describing each of the faces of the block are oriented counter-clockwise relative to the outward normal of the face. All vertices are specified relative to a local origin P_0, in this case set to the location of the center of the fluid cell in question.

After the volume of the region of overlap is calculated, the volumetric solid content is given by $\epsilon_s = V_{overlap}/V_{cell}$ where the volume of the cell is simply the volume of a cube with side length Δx. For the D3Q27 MRT collision operator used in this research, the weighting function is then given by:

$$B = \frac{\epsilon_s\left(\frac{1}{s_5} - \frac{1}{2}\right)}{(1-\epsilon_s) + \left(\frac{1}{s_5} - \frac{1}{2}\right)} = \frac{\epsilon_s\left(\frac{1}{s_7} - \frac{1}{2}\right)}{(1-\epsilon_s) + \left(\frac{1}{s_7} - \frac{1}{2}\right)} \tag{4.53}$$

4.3.2 Coupling Algorithm Validation

The proposed fluid-solid coupling algorithm was implemented in C++. As with all other portions of the software, all individual pieces of the code were unit tested using `CxxTest` [106]. Beyond simple unit tests, the coupling algorithm was tested for physical correctness by comparing numerical results with several different correlations based on physical experiments.

4.3.2.1 Comparison with experimental data

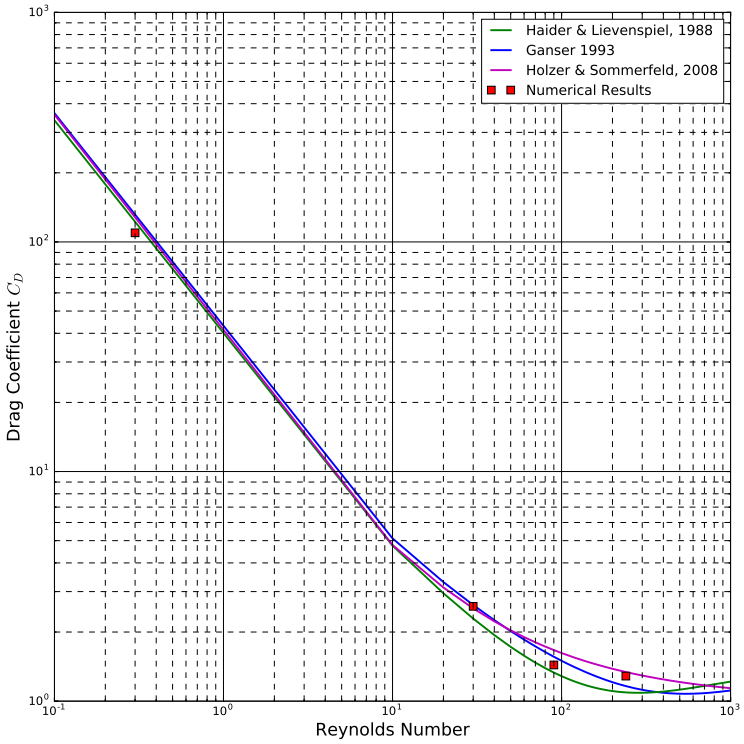


Figure 4.10: Comparison of numerically calculated drag coefficient C_D for a cube in uniform flow with regressed experimental data [[40], [23], [49]]. The upstream face of the cube is oriented perpendicular to flow.

First, the numerically calculated drag coefficient C_D was compared with regressions on numerous experimental results [[40], [23], [49]]. These regressions incorporate both shape of the particle and the Reynolds number. The value of C_D was computed for Reynolds numbers of 0.3, 30, 90 and 240 for a $1 \times 1 \times 1m$ cube in uniform flow. The upstream face of the cube was oriented perpendicular to flow. The Reynolds number is given by:

$$Re = \frac{\|\boldsymbol{u}\| d}{\nu} \tag{4.54}$$

where $\boldsymbol{u}$ is the velocity of the undisturbed fluid and ν is the kinematic viscosity. The characteristic dimension of the the cube d is defined as the diameter of the volume-equivalent sphere. The drag coefficient is defined as:

$$C_D = \frac{F_D}{\frac{1}{2}\rho u^2 A_{proj}} \tag{4.55}$$

where F_D is the force component in the direction of flow, ρ is the fluid density and A_{proj} is the projected frontal area of the volume-equivalent sphere ($A_{proj} = \pi(d/2)^2$).

As shown in Figure 4.10, the calculated drag coefficient from the numerical analyses matches all three correlations well over the tested range of Reynolds numbers. The three solid lines are the curves of regressed experimental data and the red squares are the numerical results.

4.3.2.2 Cube rotated in uniform flow

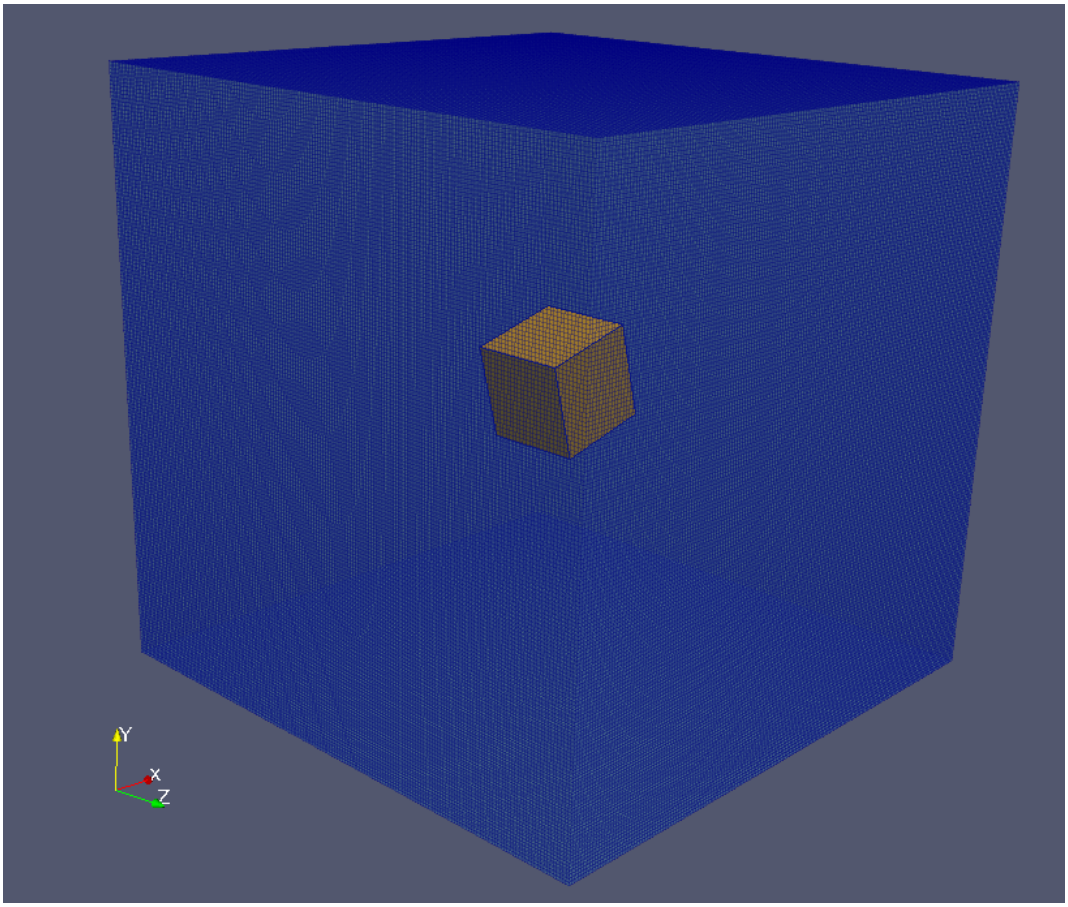
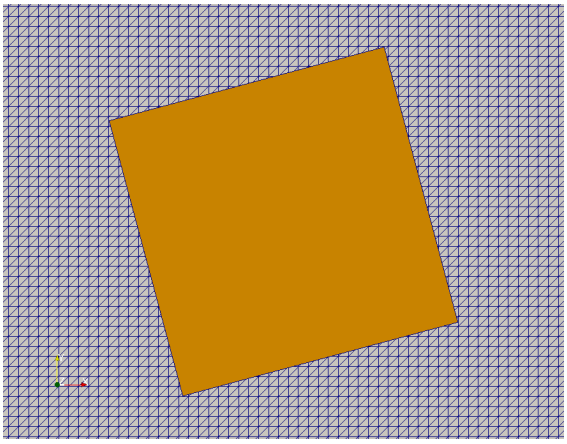
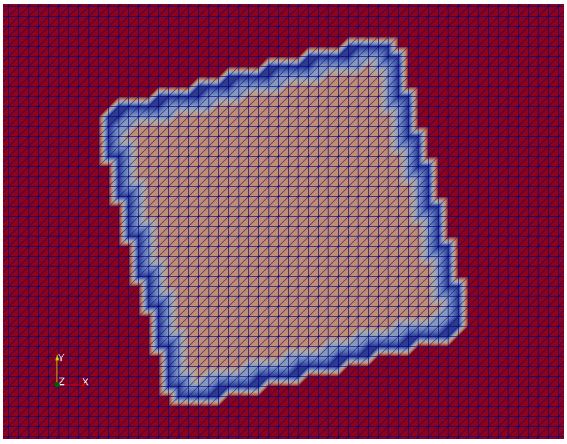


Figure 4.11: Cube embedded in three-dimensional fluid mesh

Next, the orientation of the cube relative to the flow field was changed to check whether the coupling algorithm is able to capture the change in drag coefficient due to the change in projected area of the cube relative to the direction of fluid flow. The upstream boundary was set to a constant velocity ($\boldsymbol{u} = \langle 1.0, 0.0, 0.0 \rangle \ m/s$) while the downstream boundary was a non-reflecting characteristic boundary. All other boundaries were periodic. The cube was rotated around an axis perpendicular to the flow field and simulations were run at rotation angles of 0°, 15°, 30°, 45°, 60°, 75° and 90° at Reynolds numbers 0.3, 30, 90 and 240 respectively. Figure 4.11 shows the three-dimensional mesh for the case where the cube was rotated 15°. Figure 4.12a shows a section through this three-dimensional domain in the near-vicinity of the block to illustrate the mesh density and Figure 4.12b shows the node status for the coupling process. Red nodes are fluid, pink nodes are solid, dark blue nodes are boundary fluid and light blue nodes are boundary solid. The mesh spacing was $0.035m$ in all directions giving a mesh with 5,088,448 nodes. The model time was 6.0 seconds with a time step of 0.0005 seconds. This simulation took approximately 15.7 minutes to complete on a machine with two Intel Xeon E5-2630 v2 (2.3 GHz) CPUs with 12 cores each (24 cores total) and 20GB of memory.



(a) Cube embedded in fluid mesh. Cube is rotated 15° around axis perpendicular to flow direction. Cube side length is $1m$ and mesh spacing is $0.035m$.



(b) Status of fluid nodes for coupling process. Red nodes are fluid, pink nodes are solid, dark blue nodes are boundary fluid and light blue nodes are boundary solid

Figure 4.12: Sections through three-dimensional domain showing mesh density in the vicinity of the cube

Figure 4.13 shows the calculated values of C_D for the tested rotation angles θ. The calculated values for the drag coefficient are mirrored around 45° as is expected since the cube is isometric. The maximum projected cross-sectional area is obtained when the cube is rotated 45° relative to the flow direction, giving the highest drag coefficient. Any rotation away from this orientation leads to a lower projected cross-sectional area and hence a lower drag coefficient with the lowest values at 0° and 90°. The coupling algorithm is able to

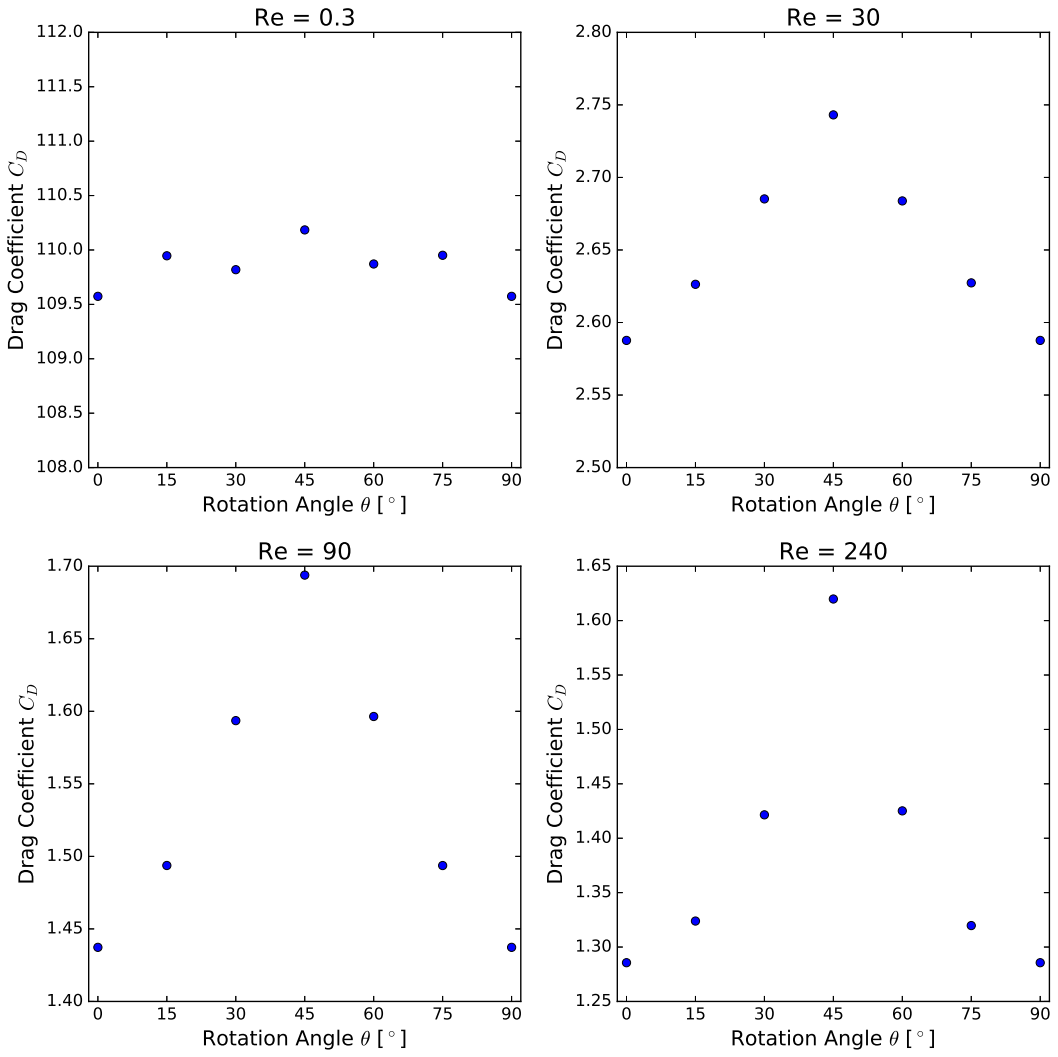
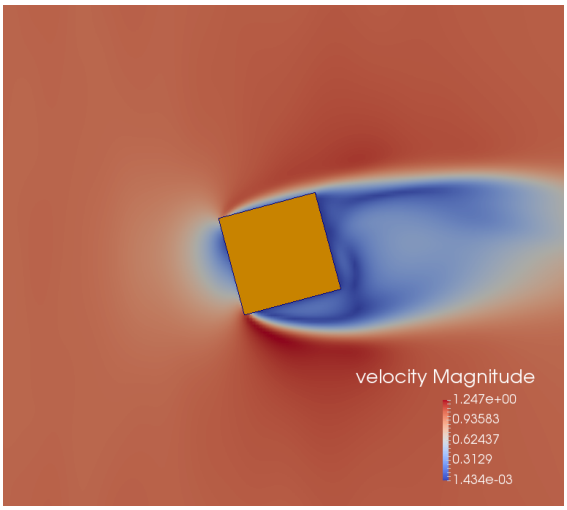
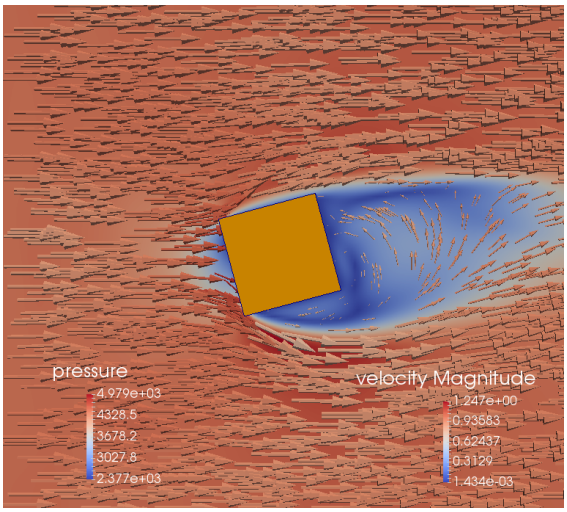


Figure 4.13: Drag coefficient C_D for cubes fixed in uniform flow as a function of rotation of cube relative to flow direction at Reynolds Number = 0.3, 30, 90 and 240

capture this behavior. Additionally, C_D is more sensitive for higher Reynolds numbers which is consistent with observed behavior. Figure 4.14a shows the resulting velocity field for the case where the cube was rotated 15° relative to flow at $Re = 240$. The velocity magnitude is shown in m/s. Figure 4.14b shows this same cube with velocity vectors colored based on pressure.



(a) Velocity field magnitude with included legend given in units of m/s



(b) Velocity field vectors scaled based on magnitude. Arrows are colored based on pressure where included legend is in $Pascals$

Figure 4.14: Cube rotated 15° relative to flow direction. The upstream boundary has constant velocity $\boldsymbol{u} = \langle 1.0, 0.0, 0.0 \rangle m/s$ while the downstream boundary is a non-reflecting characteristic boundary. All remaining boundaries are periodic.

4.4 Summary

The DEM implementation described in Chapter 3 was extended to perform coupled fluid-solid interaction analyses using weakly compressible LBM. A new coupling algorithm, which extends the paritally saturated approach, was developed to consider three-dimensional convex polyhedra moving through the fluid domain. The algorithm uses both linear programming and simplex integration for the coupling process. The LBM implementation and new coupling algorithm were validated against analytical solutions from fluid mechanics and experimental data.

Chapter 5

Performance Evaluation

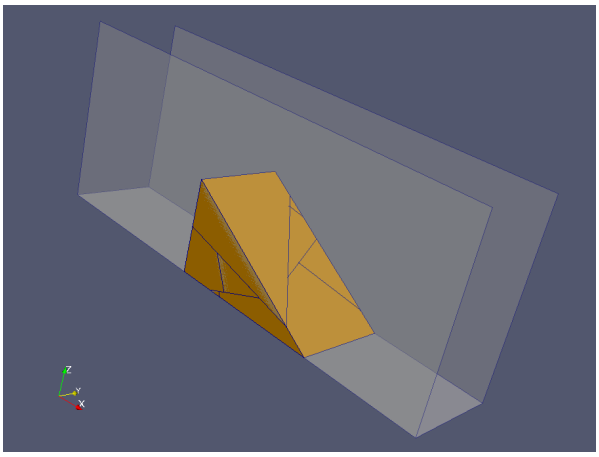
The coupled DEM-LBM program has the potential to be used to analyze a variety of problems involving solid-fluid interaction. To this end, the performance characteristics of the program were tested using several different scenarios to illustrate its capability and to identify its current limitations. Thus, in these examples the focus is on the computational demand for various classes of applications and on identifying where future improvement is warranted or needed.

5.1 Rock Erosion

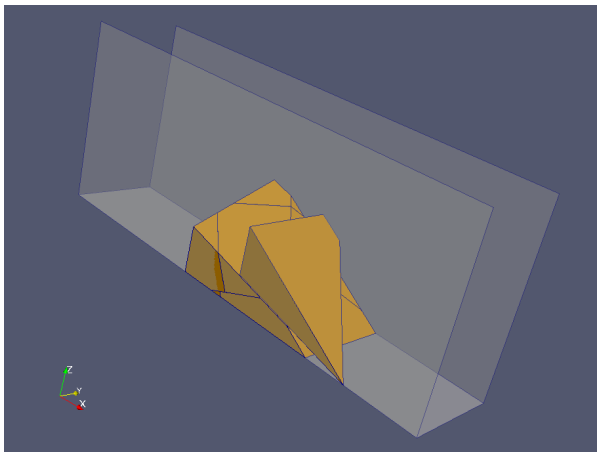
The principal motivation for this research was the development of capability to simulate the erosion of individual rock blocks from a fractured rock mass. As outlined in George [28], previous studies have focused on simplified cubic or rectangular blocks, but these simplifications may lead to misleading results as the kinematic response of the rock mass is greatly influenced by the orientation of the discontinuities. In these simulations, the fractured rock mass was generated using joint set data from an unlined dam spillway in the Sierra Nevada in Northern California to ensure the block shapes were more realistic.

First, the dry condition was tested to ensure that the rock mass was stable prior to any interaction with water. Figure 5.1a shows the initial configuration of the fractured rock where it can be clearly seen that one joint set dips unfavorably downslope. The block on the top of this unfavorable joint was unstable and failed in sliding, as shown in Figure 5.1b, and was removed prior to running analyses with fluid. The remaining rock mass was stable in the dry condition. The bottom and side boundaries were modeled as rigid, frictional boundaries.

With this stable rock mass, a coupled fluid-solid simulation was performed to investigate the potential for further erosion due to hydrodynamic loading. The upstream boundary was held to a constant velocity varying linearly from 0 m/s at the bottom of the channel to 9 m/s at the top of the domain. The downstream boundary was modeled as an outlet using a non-reflecting characteristic boundary. The fluid in the remainder of the domain was initially at rest and was allowed to come to equilibrium due to the velocity boundary on the upstream



(a) Initial configuration of dry rock mass



(b) Block sliding along unfavorable discontinuity in dry condition

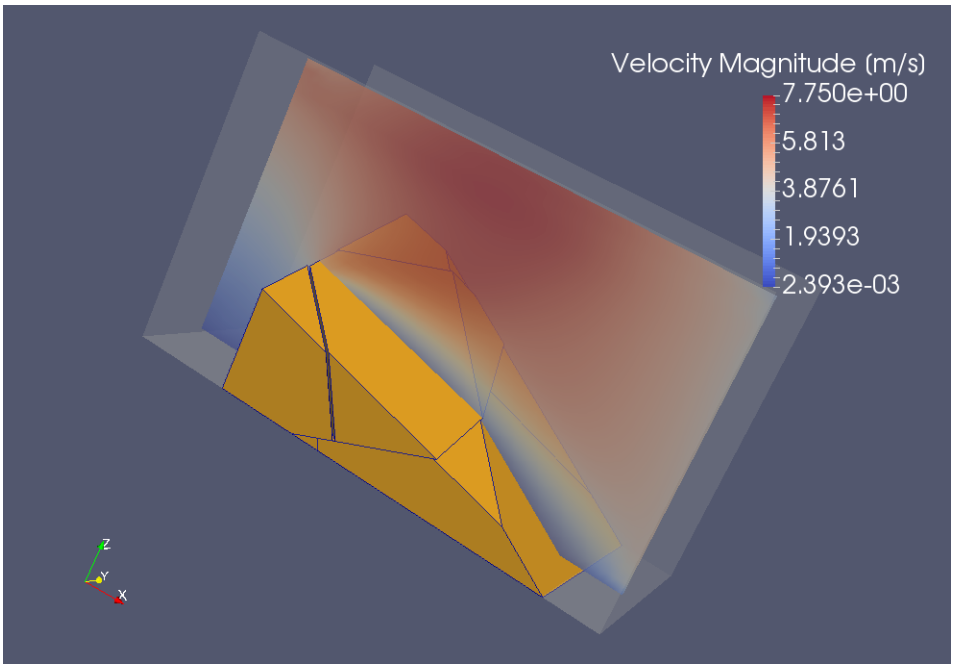
Figure 5.1: Fractured rock mass generated based on joint set data from unlined dam spillway in the Sierra Nevada in Northern California

side and gravitational acceleration as the water flowed downslope. During the course of the simulation, 6.0 seconds model time, one additional block was eroded due to hydrodynamic loading. Figure 5.2 shows the displacement of the block as well as the velocity magnitude along a section of the incline. Figure 5.3 shows a closer view of the opening fractures as the block displaces. The velocity vectors in this figure illustrate how flow through fractures increases as they open up.

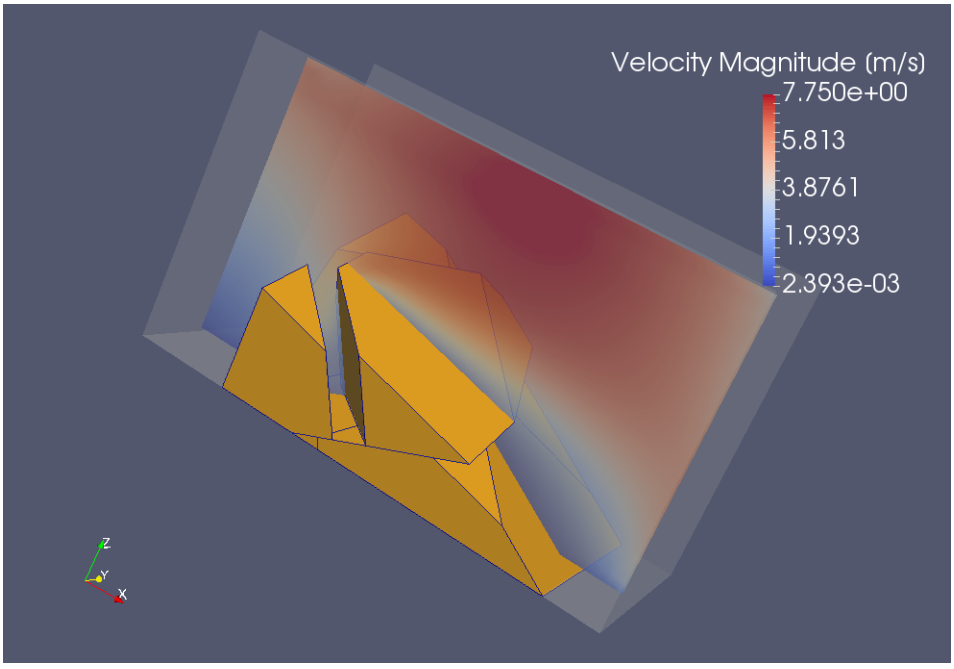
This example problem illustrates the ability of the coupled LBM-DEM program to capture hydrodynamic loading on the fractured rock as it flows over and against the rock. However, it also highlights a significant challenge in modeling this class of problem—the difference in scales required to capture flow both over the rock mass and through the rock mass. Only once the block has sufficiently displaced does water begin to flow through the fractures. This is a numerical issue as water most definitely flows through the fractures long before they have dilated as much as seen in the numerical simulations.

Numerically, blocks are allowed to overlap slightly in the DEM formulation, hence resulting in completely closed fractures. A partial solution to this issue is to define a "virtual fracture aperture" where the block faces are set back by some amount such that the effective sizes of the blocks manifested in the fluid mesh are slightly smaller and the fluid mesh is fine enough for a sufficient number of nodes to be present in the virtual fracture. This approach is effective in the current implementation of the method only when the simulation domain is small enough that gross mesh refinement is able to capture the scale of the fractures before the memory demands for such a fine mesh exceed available hardware capacity.

The computational bottleneck in this type of simulation is the fluid-solid coupling computations—each of the fluid nodes identified as a being on the fluid-solid boundary needs to be checked for volumetric solid content. In the current code implementation the coupling



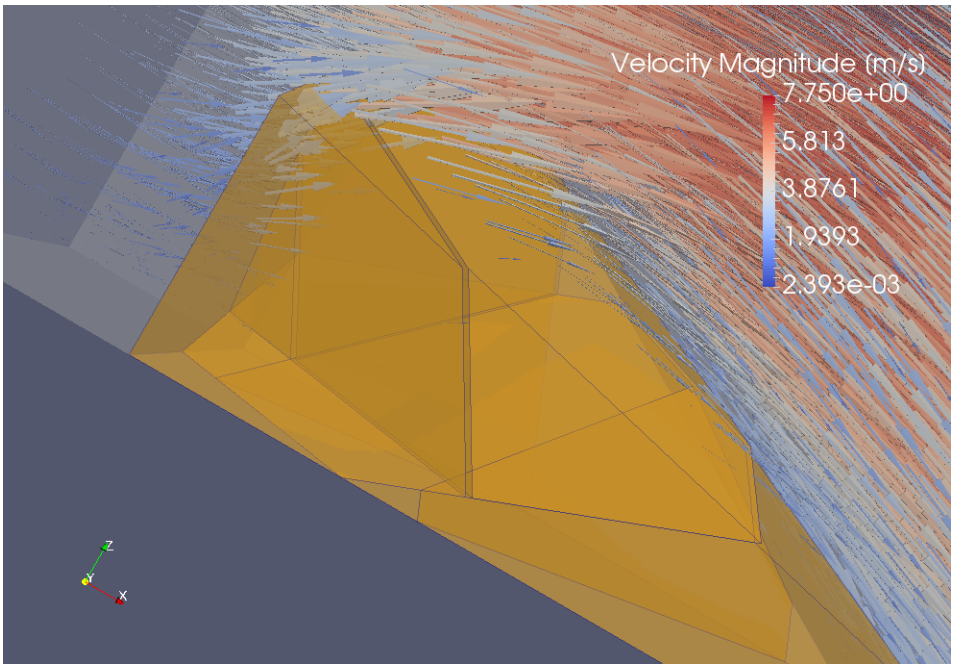
(a) Initial fracture opening



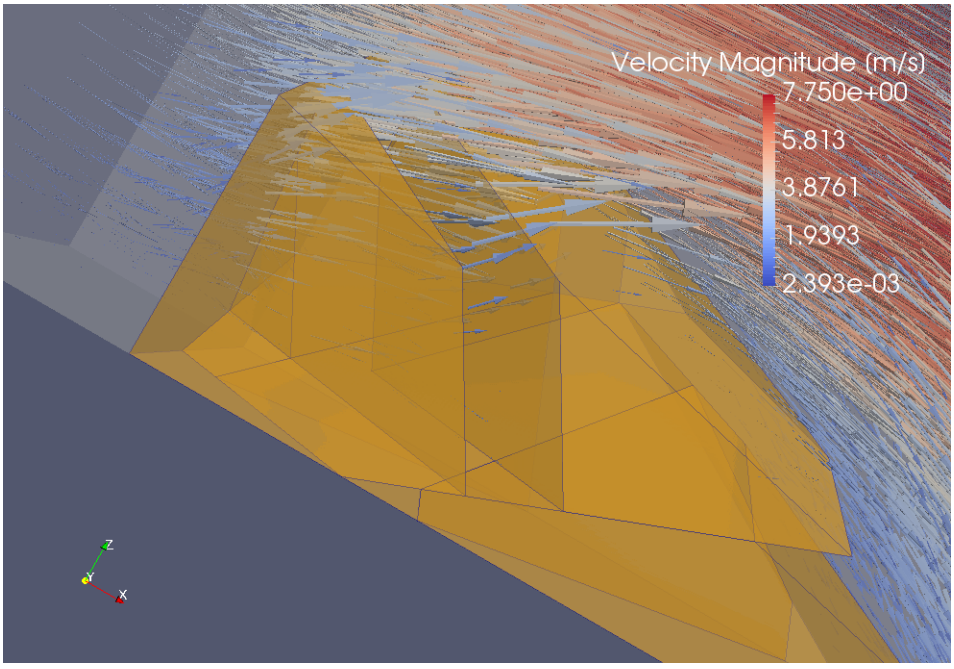
(b) Fractures opening increases as block is eroded by water

Figure 5.2: Rock block sliding on fracture plane due to hydrodynamic loading

computations are still computed in serial while the rest of the fluid computations are executed in parallel. As the fluid mesh is refined or the size or number of the blocks is increased,



(a) Little to no flow inside fractures during initial displacement



(b) Large amount of fluid flowing through fractures as they widen

Figure 5.3: Closer view of fractures and internal rock mass structure during erosion

the number of fluid cells that need to checked for volumetric solid content increases rapidly. As a result, the coupling computations may take anywhere from 30% to 60% of the total

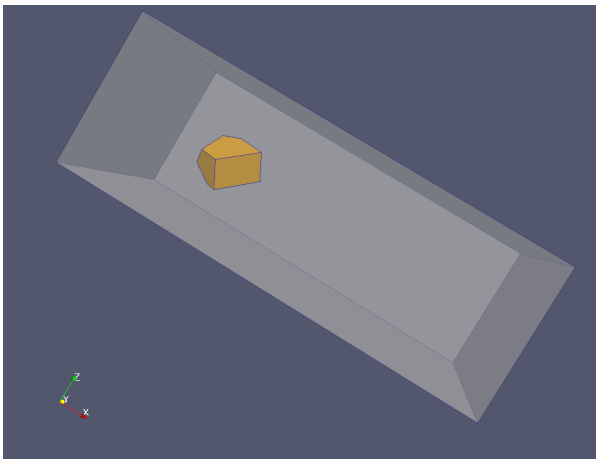
computation time depending on the problem configuration.

5.2 Transport of Polyhedral Particles

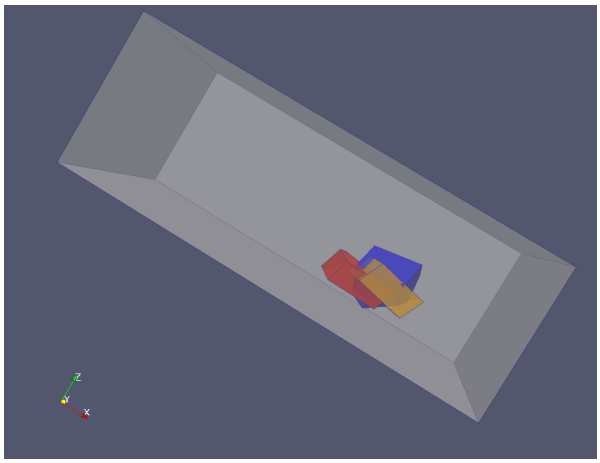
A different type of simulation problem is the transport of polyhedral particles, such as sand transport and the movement of particles along a stream bed. In these simulations a polyhedral block is dropped onto an inclined plane and moves down the plane under the influence of gravity considering three different conditions. First, the block is allowed to roll down the plane without any water present—essentially modeling a rock fall. Next, the block starts from the same initial position, but it is surrounded by stationary fluid. This is achieved through a D3Q27 MRT collision model without a body force acting on the fluid. Lastly, the block is again dropped from its initial position, but this time within a fast-moving fluid that is flowing down the inclined plane with a constant inlet velocity and an outflow boundary at the bottom of the slope. The fluid model in this instance is a D3Q27 MRT collision model with gravitational loading on the fluid that causes it to accelerate down the incline.

Figure 5.4a shows the initial block position and Figure 5.4b shows a comparison of the block displacements for the different cases. The coupling correctly captures the interaction between the block and fluid: The block in stationary fluid rolls down the slope at a slower rate compared to the dry case while the fast-moving fluid carries the block downslope at a faster rate than the block in stationary fluid but it does not bounce as high as the dry case. Figure 5.5 shows how stream tracers bend round the polyhedral block as it rolls and is pushed down slope by the fluid.

Computationally what is important to note from these simulations is the simulation time and mesh density compared to that of the simulations described in Section 5.1. The number of fluid nodes in this instance is 726,291 while the number of fluid nodes for the rock erosion example is 33,201. The computation time required for 0.1 seconds of model time for the block rolling down the incline is approximately 10 minutes while it takes approximately 63 minutes of computation time to simulate 0.1 seconds for the rock erosion example on a computer with 2 Intel Xeon E5-2630 CPUs (6 cores each) and 20GB of memory. This difference in computation time illustrates the computational demand of the fluid-solid coupling computations and the need to accelerate them in the source code. For the block rolling down the incline, doubling the size of the domain increases the total computation time less than if the domain size was held constant but the mesh refined by one order of magnitude. For these two cases the number of nodes is the same, but the number of fluid-solid boundary nodes is greater in the case where the mesh is refined. The computation time increases by a factor of approximately 1.4 for the increased domain size while it increases by a factor of 4.63 for the refined mesh case, though the number of nodes in the simulations is exactly the same. This illustrates that the bottleneck in the fluid-solid coupling computations is exacerbated in cases when a greater fraction of the fluid nodes interact with solids as opposed to just having more fluid nodes overall.



(a) Initial position of the polyhedral block for all test cases



(b) Comparison of block displacements at model time 1.7 seconds. The blue block is the dry case, the orange block is the fast-flowing fluid case and the red block is the stationary fluid case. In all cases the block continues rolling down the incline.

Figure 5.4: Polyhedral block rolling down inclined plane

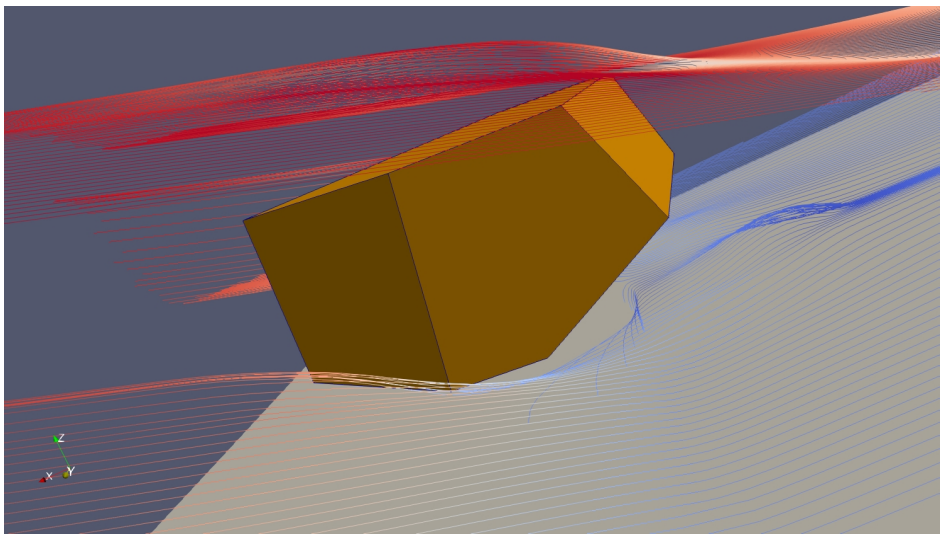


Figure 5.5: Stream tracers bending round the polyhedral block moving through the fluid mesh

5.3 Uplift Forces on Hydraulic Structures

As evidenced by the damage to the service spillway at Oroville dam, hydraulic uplift forces at the cracks and joints on slabs in hydraulic structures can pose a major risk to their

safe and reliable operation. Offsets in the joints or cracks can cause the hydraulic pressures to be transmitted underneath the slabs causing uplift or erosion of the foundation materials [20]. The stagnation pressure and flow patterns associated with offset joints was modeled using the coupled DEM-LBM implementation to illustrate its potential use for this class of problems and to show how it performs when the solid phase is fixed and does not translate through the fluid mesh. For these simulations, a 1/8 inch joint with 1/8 inch offset between slabs was modeled, similar to the physical and numerical experiments performed by Frizell [20]. Figure 5.6 shows the boundaries for the simulations. The two slabs are separated in the center of the domain by the joint where the offset between the two slabs can be clearly seen. The left boundary is the upstream side where fluid enters the domain at a constant velocity and flows toward the right side of the domain where flow is forced upward by the offset joint.

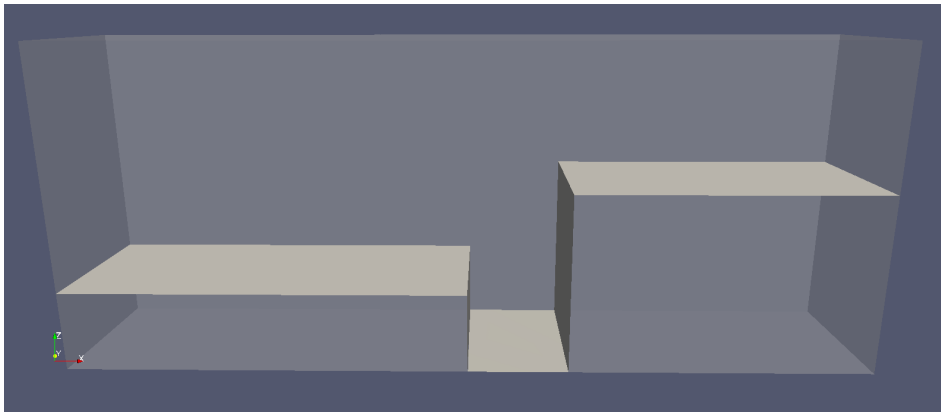


Figure 5.6: Two slabs offset by 1/8 inch separated by 1/8 inch joint

Two different cases were evaluated to show how the flow characteristics change depending on whether the joint is open—water can flow underneath the slabs—or sealed. The resulting pressure and velocity fields with associated stream tracers are shown for the open joint case in Figure 5.7 and for the sealed joint case in Figure 5.8. The location of the stagnation pressure and general flow characteristics agree well with the results presented in [20].

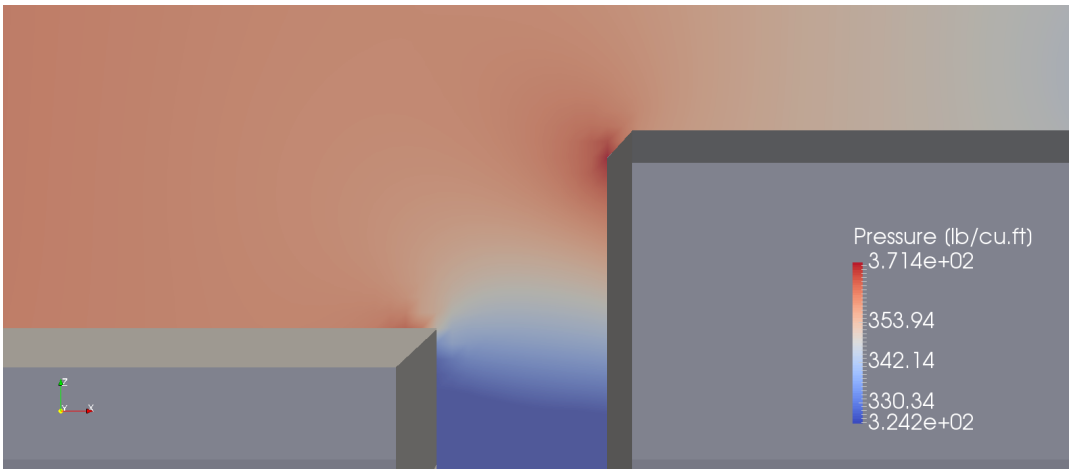
Since the slabs are kept stationary during the simulation, the volumetric solid fraction only needs to be calculated once at the beginning of the simulation. This greatly improves the performance since the volumetric solid does not need to be evaluated at every time step. For the same size time step and approximately the same number of nodes, simulating 0.1 seconds of model time can be completed roughly 20 times faster for these analyses than the simulations of a block rolling down an incline described in Section 5.2.

5.4 Summary

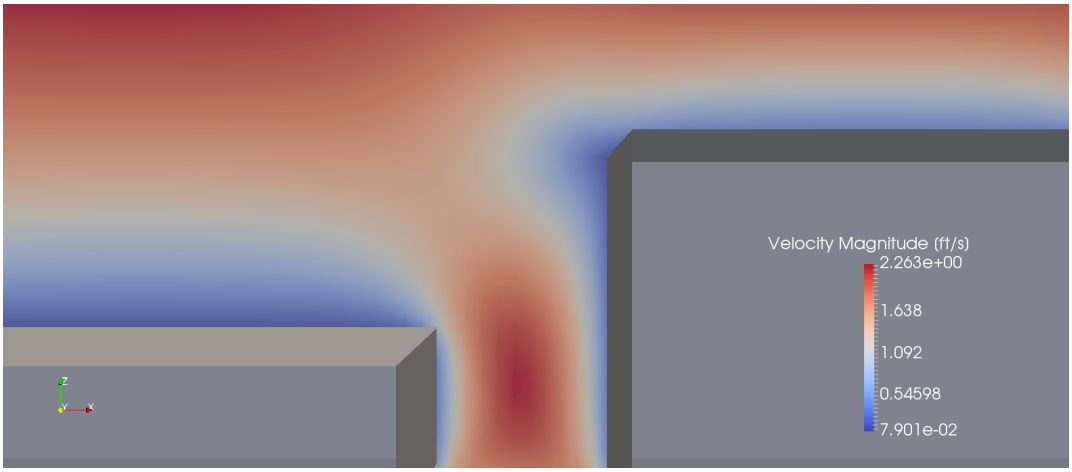
A series of example analyses were performed to assess the performance characteristics of the coupled DEM-LBM program. Table 5.1 contains relevant details about the hardware and simulation configurations for the example analyses. The results of the analyses show that these problems are extremely computationally intensive and further performance improvements will be needed to be able to simulate realistic, field scale problems. In particular, the fluid-solid coupling computations need to be accelerated as they were observed to be a bottleneck in simulations containing large amounts of solid in the fluid mesh. Additionally, in the case where fluid flows both over fractured rock and through the fractures within it, gross mesh refinement will not be able to capture the multiscale nature of the interactions and realistic solution of the interaction will require adaptive meshing and multigrid methods.

Simulation	**Number of Nodes**	**Computation Time per Model Time**	**Solid Fraction in Fluid Mesh**	**Moving/ Fixed Solid?**
Rock Erosion	33,201	43,200	Very High	Moving
Transport of Polyhedral Particles	726,291	5,700	Low	Moving
	5,704,581	8,570	Very Low	Moving
	5,704,581	27,770	High	Moving
Uplift Forces	86,961	330	High	Fixed

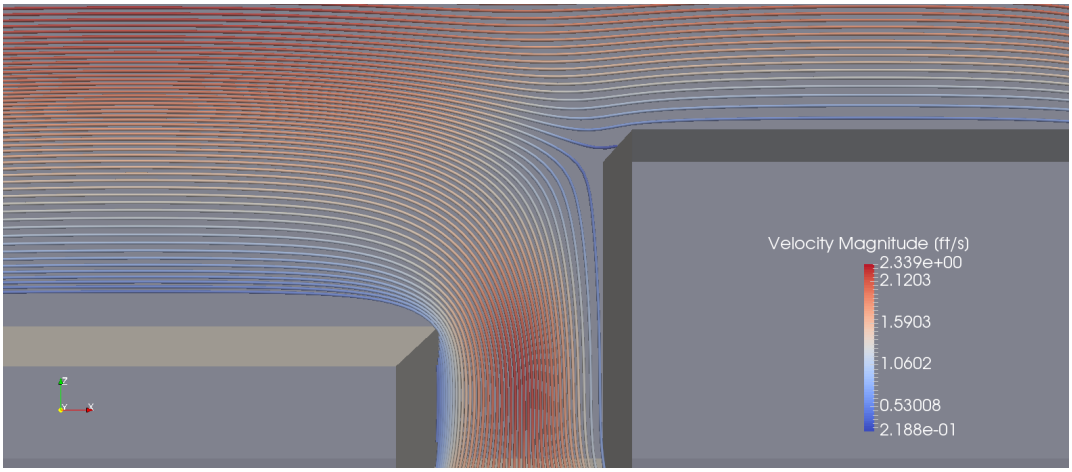
Table 5.1: Example Analyses Configurations. All simulations were executed on a machine with two Intel Xeon E-5-2630 CPUs (12 cores, 24 threads) with 20GB of memory



(a) Pressure field in and around the joint

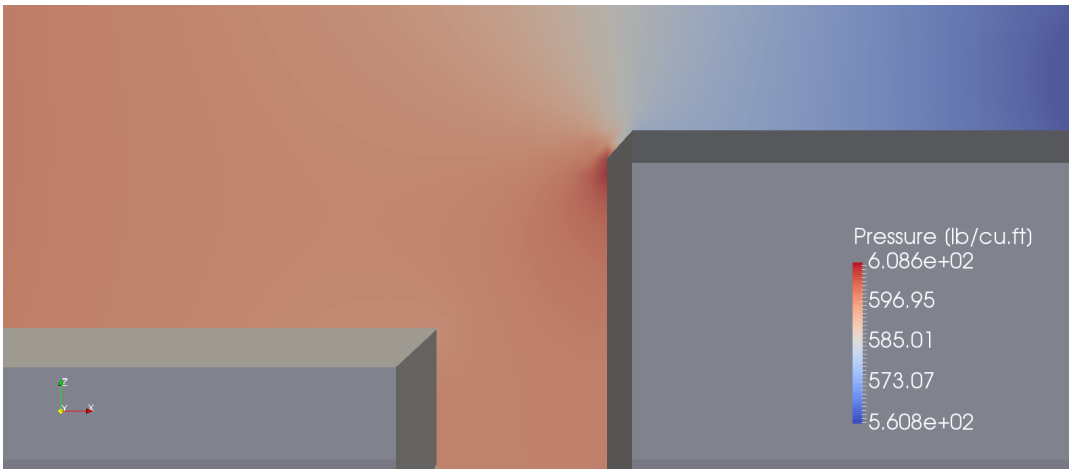


(b) Velocity field magnitude



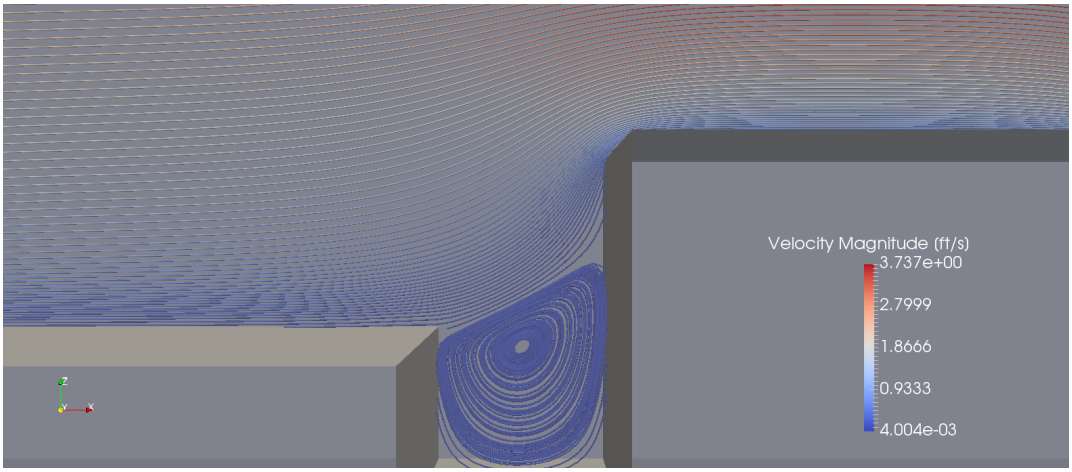
(c) Velocity field tracers

Figure 5.7: Simulation results for 1/8 inch offset slabs with 1/8 inch open joint where water is able to flow through the joint



(a) Pressure field in and around the joint

(b) Velocity field magnitude



(c) Velocity field tracers

Figure 5.8: Simulation results for 1/8 inch offset slabs with 1/8 inch closed joint

Chapter 6

Conclusions and Future Research

The objective of this research was the development of tools for the evaluation and modeling of the potential for rock erosion and rock transport by fast flowing water in rock channels as these processes have been shown to have a significant impact on the performance of unlined dam spillways and other structures. The natural jointing within the rock leads to fractured rock masses comprised of irregularly shaped polyhedral blocks. The orientation of the blocks relative to slope geometry has been shown to govern the kinematic response of the rock mass. This inherent geometric complexity and discontinuous nature of the rock mass requires numerical methods that can model the discrete interactions between individual blocks. Additionally, the shape and orientation of the individual blocks within the rock mass needs to be explicitly accounted for when modeling rock-water interaction—the hydrodynamic forces acting on each block need to be modeled directly—which is highly computationally intensive. Therefore, it is necessary to develop tools that can capture the kinematics of the rock mass response and that take advantage of parallel computing.

To this end, a three-dimensional, open-source program to produce the fractured rock mass was developed based on a linear programming approach. While this program uses previously developed techniques for efficient rock mass generation [8], its implementation is scalable to take advantage of parallel computing based on the computational resources that are available—whether the computations are executed on a desktop or on the Cloud, the program and underlying source code are exactly the same. The program automatically maintains load balance and numerical experiments show that the parallel implementation is able to maintain essentially "perfect" weak scaling. Compared to the serial implementation [8], a speedup of approximately 22 can be obtained on a desktop workstation. When scaling computations up to the Cloud, approximately 8 million blocks can be generated in the same amount of time as it takes the serial implementation to generate 60,000 blocks.

The second stage of this research effort concentrated on the development of a new open-source DEM program for analyzing blocky rock mass kinematics. The contact detection algorithm used for DEM is also based on a linear programming approach [7], which allows for similar data structures and logic in both the DEM and block generation code. The program implementation has been verified against analytical solutions as well as other numerical

solutions and has been shown to accurately capture the three-dimensional kinematic behavior of polyhedral rock blocks.

The DEM program was then extended to perform a coupled fluid-solid interaction analysis using a weakly compressible three-dimensional LBM formulation. A new coupling algorithm was developed that is able to consider three-dimensional convex polyhedra moving through the LBM mesh. The algorithm extends the partially saturated method [78] to analytically calculate the volumetric solid content in a fluid cell using linear programming and simplex integration. The LBM implementation was verified against analytical solutions in fluid mechanics and the coupled DEM-LBM algorithm was validated through comparison with regressed experimental data. Additionally, the fluid computations were accelerated using the C++ library `Kokkos` [15] such that computations can be executed on both the central processing unit (CPU) and graphics processing unit (GPU). The capabilities of the new coupled DEM-LBM implementation were then explored by evaluating the performance characteristics of the program in modeling different types of problems involving solid-fluid interaction.

6.1 Future Research

The fully coupled DEM-LBM software developed as part of this research effort is capable of simulating the complex interaction between fractured rock and water. However, the capability to model full-scale problems is still elusive due to the enormous computational effort dictated by the multiscale nature of such problems. Therefore, significant future effort is required to improve the computational speed and model efficiency to allow realistic representation of natural settings. Thus, future effort should consider incorporating the stochastic nature of discontinuities within the rock mass, better capturing the multiscale aspects of rock-fluid interaction and, more pragmatically, enhancing parallel capability of the simulation software.

In terms of the fractured rock mass generation, the current implementation of the block cutting algorithm generates a fractured rock mass with persistent joints. However, non-persistent joints are a common occurrence in natural rock and should be incorporated into analyzing the behavior of fractured rock. Future work should include a stochastic joint generator that captures the natural variation in strike, dip, spacing and persistence of discontinuities. The current intersection code is able to account for non-persistent joints and the code that generates the joint sets can be expanded to produce stochastic realizations such that the natural variability in the rock mass can be considered. This will make it possible to quantify the uncertainty in the response of the rock mass given its natural variability. Specifically, reliability methods are available to estimate the probability distribution for rock scour with a small number of model evaluations and automatically provide sensitivity of the solution to the uncertain input parameters—discontinuity geometry and strength.

Furthermore, rock scour is a multiscale problem where the size of the rock blocks compared to the fractures varies by orders of magnitude. The required mesh resolution to

accurately resolve hydrodynamic forces of water flowing over and around rock blocks is significantly less than the mesh resolution required to capture the interaction between the fractured rock mass and water flowing through the discontinuities. For example, in the experimental work by George [28] the diameter of the volume equivalent sphere for the blocks tested was approximately $8cm$ with flow depths up to $12cm$ while the joint aperture was only $1.8mm$. Clearly, the size of the mesh within the fractures would have to be much finer compared to the rest of the domain and simply applying that fine of a mesh to the entire simulation domain is not feasible in terms of memory demands. Additionally, once the blocks start displacing the mesh density requirements would change as the fractures open up and the blocks begin to move through the fluid mesh. Adaptive meshing and multigrid methods would be ideal for capturing these different scales of interaction.

Lastly, the high computational and memory demands for three-dimensional direct simulation of rock-water interaction requires parallel computing and efficient use of computing resources. The fluid computations have been accelerated to use shared memory parallelism on both the CPU and GPU; however, the DEM-LBM coupling and DEM computations are currently executed in serial. As shown in Chapter 5, the fluid-solid coupling computations are a bottleneck when there is a substantial amount of solid present in the fluid domain. Accelerating this portion of the computations would greatly improve performance and enhance the capability for simulating fluid-solid interaction at a greater scale. Though the current parallelization has been implemented using shared memory, future acceleration should also consider including distributed memory parallelism such that more computational resources can be brought to bear on increasingly large simulations.

Appendix A

Transformation Matrices for MRT-LBM

The transformation matrix $\boldsymbol{M}$ used in the multiple-relaxation-time collision operator described in Section 4.2.2.1 is calculated from Equation 4.10 with the discrete velocities $\boldsymbol{c}_i$ listed in Table 4.1, as shown in Figure 4.2, for the D3Q27 velocity set. The resulting tranformation matrix is calculated as:

$$\boldsymbol{M} = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\
0 & 0 & 1 & 0 & 1 & -1 & 0 & 0 & 1 & 1 & 1 & 1 & -1 & -1 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & -1 & -1 & -1 & -1 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
-2 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 2 & -1 & -1 & 1 & 1 & 1 & 1 & -2 & -2 & 0 & 0 & 0 & 0 & 2 & -1 & -1 & 1 & 1 & 1 & 1 & -2 & -2 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 1 & -1 \\
0 & -4 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 2 & 2 & 2 & 2 & 4 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & -2 & -2 & -2 & -2 \\
0 & 0 & -4 & 0 & -1 & 1 & 0 & 0 & -1 & -1 & 2 & 2 & -2 & -2 & 0 & 4 & 0 & 1 & -1 & 0 & 0 & 1 & 1 & -2 & -2 & 2 & 2 \\
0 & 0 & 0 & -4 & 0 & 0 & -1 & 1 & -1 & 1 & 2 & -2 & 2 & -2 & 0 & 0 & 4 & 0 & 0 & 1 & -1 & 1 & -1 & -2 & 2 & -2 & 2 \\
0 & 4 & 0 & 0 & -2 & -2 & -2 & -2 & 0 & 0 & 1 & 1 & 1 & 1 & -4 & 0 & 0 & 2 & 2 & 2 & 2 & 0 & 0 & -1 & -1 & -1 & -1 \\
0 & 0 & 4 & 0 & -2 & 2 & 0 & 0 & -2 & -2 & 1 & 1 & -1 & -1 & 0 & -4 & 0 & 2 & -2 & 0 & 0 & 2 & 2 & -1 & -1 & 1 & 1 \\
0 & 0 & 0 & 4 & 0 & 0 & -2 & 2 & -2 & 2 & 1 & -1 & 1 & -1 & 0 & 0 & -4 & 0 & 0 & 2 & -2 & 2 & -2 & -1 & 1 & -1 & 1 \\
4 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\
-8 & 4 & 4 & 4 & -2 & -2 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 & 4 & 4 & 4 & -2 & -2 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\
0 & -4 & 2 & 2 & 1 & 1 & 1 & 1 & -2 & -2 & 0 & 0 & 0 & 0 & -4 & 2 & 2 & 1 & 1 & 1 & 1 & -2 & -2 & 0 & 0 & 0 & 0 \\
0 & 0 & -2 & 2 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -2 & 2 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & -2 & 2 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & 1 & -1 & -1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & 0 & 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & 0 & 0 & 1 & -1 & 1 & -1 \\
0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1
\end{bmatrix} \tag{A.1}$$

and its inverse is calculated as:

$$\boldsymbol{M}^{-1} = \frac{1}{216}\begin{bmatrix}
8 & 0 & 0 & 0 & -24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 24 & -8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 12 & 0 & 0 & -12 & 12 & 0 & 0 & 0 & 0 & -12 & 0 & 0 & 12 & 0 & 0 & 0 & 4 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 12 & 0 & -12 & -6 & 18 & 0 & 0 & 0 & 0 & -12 & 0 & 0 & 12 & 0 & 0 & 4 & 6 & -18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & 12 & -12 & -6 & -18 & 0 & 0 & 0 & 0 & 0 & -12 & 0 & 0 & 12 & 0 & 4 & 6 & 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 12 & 12 & 0 & 0 & 6 & 18 & 18 & 0 & 0 & -3 & -3 & 0 & -6 & -6 & 0 & -6 & -2 & 3 & 9 & -18 & 0 & 0 & 27 & -27 & 0 & 0 \\
8 & 12 & -12 & 0 & 0 & 6 & 18 & -18 & 0 & 0 & -3 & 3 & 0 & -6 & 6 & 0 & -6 & -2 & 3 & 9 & 18 & 0 & 0 & 27 & 27 & 0 & 0 \\
8 & 12 & 0 & 12 & 0 & 6 & -18 & 0 & 0 & 18 & -3 & 0 & -3 & -6 & 0 & -6 & -6 & -2 & 3 & -9 & 0 & 0 & -18 & -27 & 0 & 27 & 0 \\
8 & 12 & 0 & -12 & 0 & 6 & -18 & 0 & 0 & -18 & -3 & 0 & 3 & -6 & 0 & 6 & -6 & -2 & 3 & -9 & 0 & 0 & 18 & -27 & 0 & -27 & 0 \\
8 & 0 & 12 & 12 & 0 & -12 & 0 & 0 & 18 & 0 & 0 & -3 & -3 & 0 & -6 & -6 & -6 & -2 & -6 & 0 & 0 & -18 & 0 & 0 & 27 & -27 & 0 \\
8 & 0 & 12 & -12 & 0 & -12 & 0 & 0 & -18 & 0 & 0 & -3 & 3 & 0 & -6 & 6 & -6 & -2 & -6 & 0 & 0 & 18 & 0 & 0 & 27 & 27 & 0 \\
8 & 12 & 12 & 12 & 12 & 0 & 0 & 18 & 18 & 18 & 6 & 6 & 6 & 3 & 3 & 3 & 6 & 1 & 0 & 0 & 9 & 9 & 9 & 0 & 0 & 0 & 27 \\
8 & 12 & 12 & -12 & 12 & 0 & 0 & 18 & -18 & -18 & 6 & 6 & -6 & 3 & 3 & -3 & 6 & 1 & 0 & 0 & 9 & -9 & -9 & 0 & 0 & 0 & -27 \\
8 & 12 & -12 & 12 & 12 & 0 & 0 & -18 & -18 & 18 & 6 & -6 & 6 & 3 & -3 & 3 & 6 & 1 & 0 & 0 & -9 & -9 & 9 & 0 & 0 & 0 & -27 \\
8 & 12 & -12 & -12 & 12 & 0 & 0 & -18 & 18 & -18 & 6 & -6 & -6 & 3 & -3 & -3 & 6 & 1 & 0 & 0 & -9 & 9 & -9 & 0 & 0 & 0 & 27 \\
8 & -12 & 0 & 0 & -12 & 12 & 0 & 0 & 0 & 0 & 12 & 0 & 0 & -12 & 0 & 0 & 0 & 4 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & -12 & 0 & -12 & -6 & 18 & 0 & 0 & 0 & 0 & 12 & 0 & 0 & -12 & 0 & 0 & 4 & 6 & -18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & -12 & -12 & -6 & -18 & 0 & 0 & 0 & 0 & 0 & 12 & 0 & 0 & -12 & 0 & 4 & 6 & 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & -12 & -12 & 0 & 0 & 6 & 18 & 18 & 0 & 0 & 3 & 3 & 0 & 6 & 6 & 0 & -6 & -2 & 3 & 9 & -18 & 0 & 0 & -27 & 27 & 0 & 0 \\
8 & -12 & 12 & 0 & 0 & 6 & 18 & -18 & 0 & 0 & 3 & -3 & 0 & 6 & -6 & 0 & -6 & -2 & 3 & 9 & 18 & 0 & 0 & -27 & -27 & 0 & 0 \\
8 & -12 & 0 & -12 & 0 & 6 & -18 & 0 & 0 & 18 & 3 & 0 & 3 & 6 & 0 & 6 & -6 & -2 & 3 & -9 & 0 & 0 & -18 & 27 & 0 & -27 & 0 \\
8 & -12 & 0 & 12 & 0 & 6 & -18 & 0 & 0 & -18 & 3 & 0 & -3 & 6 & 0 & -6 & -6 & -2 & 3 & -9 & 0 & 0 & 18 & 27 & 0 & 27 & 0 \\
8 & 0 & -12 & -12 & 0 & -12 & 0 & 0 & 18 & 0 & 0 & 3 & 3 & 0 & 6 & 6 & -6 & -2 & -6 & 0 & 0 & -18 & 0 & 0 & -27 & 27 & 0 \\
8 & 0 & -12 & 12 & 0 & -12 & 0 & 0 & -18 & 0 & 0 & 3 & -3 & 0 & 6 & -6 & -6 & -2 & -6 & 0 & 0 & 18 & 0 & 0 & -27 & -27 & 0 \\
8 & -12 & -12 & -12 & 12 & 0 & 0 & 18 & 18 & 18 & -6 & -6 & -6 & -3 & -3 & -3 & 6 & 1 & 0 & 0 & 9 & 9 & 9 & 0 & 0 & 0 & -27 \\
8 & -12 & -12 & 12 & 12 & 0 & 0 & 18 & -18 & -18 & -6 & -6 & 6 & -3 & -3 & 3 & 6 & 1 & 0 & 0 & 9 & -9 & -9 & 0 & 0 & 0 & 27 \\
8 & -12 & 12 & -12 & 12 & 0 & 0 & -18 & -18 & 18 & -6 & 6 & -6 & -3 & 3 & -3 & 6 & 1 & 0 & 0 & -9 & -9 & 9 & 0 & 0 & 0 & 27 \\
8 & -12 & 12 & 12 & 12 & 0 & 0 & -18 & 18 & -18 & -6 & 6 & 6 & -3 & 3 & 3 & 6 & 1 & 0 & 0 & -9 & 9 & -9 & 0 & 0 & 0 & -27
\end{bmatrix} \tag{A.2}$$

Bibliography

[1] B Amadei and S Saeb. "Constitutive models of rock joints". In: *Rock joints. Proceedings of the international symposium on rock joints, Loen, Norway*. 1990, pp. 4–6.

[2] T. B. Anderson and Roy Jackson. "Fluid Mechanical Description of Fluidized Beds. Equations of Motion". In: *Industrial & Engineering Chemistry Fundamentals* 6.4 (1967), pp. 527–539. DOI: 10.1021/i160024a007. eprint: https://doi.org/10.1021/i160024a007. URL: https://doi.org/10.1021/i160024a007.

[3] Michael Armbrust et al. *Above the Clouds: A Berkeley View of Cloud Computing*. Feb. 2009. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[4] N Barton and SC Bandis. "Rock joint model for analyses of geologic discontinua". In: *Proc. 2rid Int. Conf. on Constitutive Laws for Engng Materials*. 1987.

[5] George Keith Batchelor. *An introduction to fluid dynamics*. Cambridge university press, 2000.

[6] P. L. Bhatnagar, E. P. Gross, and M. Krook. "A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems". In: *Phys. Rev.* 94 (3 May 1954), pp. 511–525. DOI: 10.1103/PhysRev.94.511. URL: https://link.aps.org/doi/10.1103/PhysRev.94.511.

[7] C.W. Boon, G.T. Houlsby, and S. Utili. "A new algorithm for contact detection between convex polygonal and polyhedral particles in the discrete element method". In: *Computers and Geotechnics* 44 (2012), pp. 73–82. ISSN: 0266-352X. DOI: https://doi.org/10.1016/j.compgeo.2012.03.012. URL: http://www.sciencedirect.com/science/article/pii/S0266352X12000535.

[8] C.W. Boon, G.T. Houlsby, and S. Utili. "A new rock slicing method based on linear programming". In: *Computers and Geotechnics* 65 (Apr. 2015), pp. 12–29. DOI: 10.1016/j.compgeo.2014.11.007.

[9] Shiyi Chen and Gary D. Doolen. "Lattice Boltzmann Method for Fluid Flows". In: *Annual Review of Fluid Mechanics* 30.1 (1998), pp. 329–364. DOI: 10.1146/annurev.fluid.30.1.329. eprint: https://doi.org/10.1146/annurev.fluid.30.1.329. URL: https://doi.org/10.1146/annurev.fluid.30.1.329.

[10] R. Cornubert, D. d'Humières, and D. Levermore. "A Knudsen layer theory for lattice gases". In: *Physica D: Nonlinear Phenomena* 47.1 (1991), 241–259. ISSN: 0167-2789. DOI: https://doi.org/10.1016/0167-2789(91)90295-K. URL: http://www.sciencedirect.com/science/article/pii/016727899190295K.

[11] P. A. Cundall and O. D. L. Strack. "A discrete numerical model for granular assemblies". In: *Gèotechnique* 29.1 (1979), pp. 47–65. DOI: 10.1680/geot.1979.29.1.47. eprint: http://dx.doi.org/10.1680/geot.1979.29.1.47. URL: http://dx.doi.org/10.1680/geot.1979.29.1.47.

[12] P.A. Cundall. "Analytical and Computational Methods in Engineering Rock Mechanics". In: ed. by E.T. Brown. Allen and Unwin, 1987. Chap. Distinct element models for rock and soil structure, pp. 129–163.

[13] P.A. Cundall. "Formulation of a Three-dimensional Distinct Element Model—Part I. A Scheme to Detect and Represent Contacts in a System Composed of Many Polyhedral Blocks". In: *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts* 25.3 (1988), pp. 107–116. DOI: 10.1016/0148-9062(88)92293-0.

[14] Dominique d'Humières et al. "Multiple–relaxation–time lattice Boltzmann models in three dimensions". In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 360.1792 (2002). Ed. by Irina Ginzburg et al., pp. 437–451. ISSN: 1364-503X. DOI: 10.1098/rsta.2001.0955. eprint: http://rsta.royalsocietypublishing.org/content/360/1792/437.full.pdf. URL: http://rsta.royalsocietypublishing.org/content/360/1792/437.

[15] H. Carter Edwards, Christian R. Trott, and Daniel Sunderland. "Kokkos: Enabling manycore performance portability through polymorphic memory access patterns". In: *Journal of Parallel and Distributed Computing* 74.12 (2014). Domain-Specific Languages and High-Level Frameworks for High-Performance Computing, pp. 3202–3216. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2014.07.003. URL: http://www.sciencedirect.com/science/article/pii/S0743731514001257.

[16] Strack O. Erik and Cook Benjamin K. "Three-dimensional immersed boundary conditions for moving solids in the lattice-Boltzmann method". In: *International Journal for Numerical Methods in Fluids* 55.2 (2007), pp. 103–125. DOI: 10.1002/fld.1437. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.1437. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.1437.

[17] Y.T. Feng and D.R.J. Owen. "A spatial digital tree based contact detection algorithm". In: *Proc. 4th Int. Conf. on Analysis of Discontinuous Deformation (ICADD-4)*. 2001.

[18] John W. France et al. *Independent Forensic Team Report: Oroville Dam Spillway Incident*. Tech. rep. California Department of Water Resources, 2018.

[19] Uriel Frisch, Brosl Hasslacher, and Yves Pomeau. "Lattice-gas automata for the Navier-Stokes equation". In: *Physical review letters* 56.14 (1986), p. 1505.

[20] Warren Frizell. *Uplift and Crack Flow Resulting from High Velocity Discharges over Open Offset Joints—Laboratory Studies*. Tech. rep. DSO-07-07. United States Bureau of Reclamation, 2007.

[21] Tsugio Fukuchi. "Numerical calculation of fully-developed laminar flows in arbitrary cross-sections using finite difference method". In: *AIP Advances* 1.4 (2011), p. 042109. DOI: 10.1063/1.3652881. eprint: https://doi.org/10.1063/1.3652881. URL: https://doi.org/10.1063/1.3652881.

[22] Bfer G. "An isoparametric joint/interface element for finite element analysis". In: *International Journal for Numerical Methods in Engineering* 21.4 (1985), pp. 585–600. DOI: 10.1002/nme.1620210402. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620210402. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620210402.

[23] Gary H. Ganser. "A rational approach to drag prediction of spherical and nonspherical particles". In: *Powder Technology* 77.2 (1993), pp. 143–152. ISSN: 0032-5910. DOI: https://doi.org/10.1016/0032-5910(93)80051-B. URL: http://www.sciencedirect.com/science/article/pii/003259109380051B.

[24] Michael Gardner, John Kolb, and Nicholas Sitar. "Parallel and scalable block system generation". In: *Computers and Geotechnics* 89 (2017), pp. 168–178. ISSN: 0266-352X. DOI: https://doi.org/10.1016/j.compgeo.2017.05.001. URL: http://www.sciencedirect.com/science/article/pii/S0266352X17301143.

[25] Martin Geier et al. "The cumulant lattice Boltzmann equation in three dimensions: Theory and validation". In: *Computers & Mathematics with Applications* 70.4 (2015), pp. 507–547. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2015.05.001.

[26] MF George, N Sitar, L Sklar, et al. "Experimental Evaluation of Rock Erosion in Spillway Channels". In: *49th US Rock Mechanics/Geomechanics Symposium*. American Rock Mechanics Association. 2015.

[27] MF George, N Sitar, et al. "Mechanics of 3D rock block erodibility". In: *50th US Rock Mechanics/Geomechanics Symposium*. American Rock Mechanics Association. 2016.

[28] Michael F. George. "3D block erodibility: Dynamics of rock-water interaction in rock scour". English. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2016-05-31. PhD thesis. 2015, p. 462. ISBN: 9781339593296. URL: https://search-proquest-com.libproxy.berkeley.edu/docview/1778844728?accountid=14496.

[29] Michael F. George and Nicholas Sitar. "System reliability approach for rock scour". In: *International Journal of Rock Mechanics and Mining Sciences* 85 (2016), pp. 102–111. ISSN: 1365-1609. DOI: https://doi.org/10.1016/j.ijrmms.2016.03.012. URL: http://www.sciencedirect.com/science/article/pii/S1365160916300399.

[30] Michael F George, Nicholas Sitar, and Armen Der Kiureghian. "System reliability analysis of rock scour". In: *Dams & extreme events e reducing risk of aging infrastructure under extreme conditions. Proceedings of the 34th Annual USSD Conference, San Francisco, CA. Denver: US Society on Dams*. 2014, pp. 477–94.

[31] Jamshid Ghaboussi, Edward L Wilson, and Jeremy Isenberg. "Finite element for rock joints and interfaces". In: *Journal of Soil Mechanics & Foundations Div* 99.Proc Paper 10095 (1973).

[32] I Ginzbourg and PM Adler. "Boundary flow condition analysis for the three-dimensional lattice Boltzmann model". In: *Journal de Physique II* 4.2 (1994), pp. 191–214. DOI: 10.1051/jp2:1994123.

[33] Irina Ginzburg, Frederik Verhaeghe, and Dominique d'Humières. "Two-relaxation-time lattice Boltzmann scheme: About parametrization, velocity, pressure and mixed boundary conditions". In: *Communications in computational physics* 3.2 (2008), pp. 427–478.

[34] R. Goodman and G. Shi. *Block theory and its applications to rock engineering*. Prentice-Hall, Inc., 1985.

[35] Richard E Goodman. *Methods of geological engineering in discontinuous rocks*. St. Paul: West Pub. Co, 1976.

[36] Richard E. Goodman and D. Scott Kieffer. "Behavior of Rock in Slopes". In: *Journal of Geotechnical and Geoenvironmental Engineering* 126.8 (2000), pp. 675–684. DOI: 10.1061/(ASCE)1090-0241(2000)126:8(675). eprint: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%291090-0241%282000%29126%3A8%28675%29. URL: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%291090-0241%282000%29126%3A8%28675%29.

[37] Richard E Goodman, Robert L Taylor, and Tor L Brekke. "A model for the mechanics of jointed rock". In: *Journal of Soil Mechanics & Foundations Div* (1968).

[38] Itasca Consulting Group. *3DEC Version 4.1*. Minneapolis, Minnesota, 2007.

[39] Zhaoli Guo, Chuguang Zheng, and Baochang Shi. "Discrete lattice effects on the forcing term in the lattice Boltzmann method". In: *Physical Review E* 65.4 (2002), p. 046308.

[40] A. Haider and O. Levenspiel. "Drag coefficient and terminal velocity of spherical and nonspherical particles". In: *Powder Technology* 58.1 (1989), pp. 63–70. ISSN: 0032-5910. DOI: https://doi.org/10.1016/0032-5910(89)80008-7. URL: http://www.sciencedirect.com/science/article/pii/0032591089800087.

[41] R. Hart, P.A. Cundall, and J. Lemos. "Formulation of a three-dimensional distinct element model—Part II. Mechanical calculations for motion and interaction of a system composed of many polyhedral blocks". In: *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts* 25.3 (1988), pp. 117–125. ISSN: 0148-9062. DOI: https://doi.org/10.1016/0148-9062(88)92294-2. URL: http://www.sciencedirect.com/science/article/pii/0148906288922942.

[42] Xiaoyi He and Li-Shi Luo. "Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation". In: *Phys. Rev. E* 56 (6 Dec. 1997), pp. 6811–6817. DOI: 10.1103/PhysRevE.56.6811. URL: https://link.aps.org/doi/10.1103/PhysRevE.56.6811.

[43] Xiaoyi He, Xiaowen Shan, and Gary D. Doolen. "Discrete Boltzmann equation model for nonideal gases". In: *Phys. Rev. E* 57 (1 Jan. 1998), R13–R16. DOI: 10.1103/PhysRevE.57.R13. URL: https://link.aps.org/doi/10.1103/PhysRevE.57.R13.

[44] Xiaoyi He et al. "Analytic solutions of simple flows and analysis of nonslip boundary conditions for the lattice Boltzmann BGK model". In: *Journal of Statistical Physics* 87.1 (Apr. 1997), pp. 115–136. ISSN: 1572-9613. DOI: 10.1007/BF02181482. URL: https://doi.org/10.1007/BF02181482.

[45] D. Heliot. "Generating a blocky rock mass". In: *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts* 25.3 (June 1988), pp. 127–138. DOI: 10.1016/0148-9062(88)92295-4.

[46] Heinrich Rudolf Hertz. "Über die Berührung fester elastischer Körper". In: *Verhandlung des Vereins zur Beforderung des GewerbefleiBes, Berlin* (1882), p. 449.

[47] Daniel Heubes, Andreas Bartel, and Matthias Ehrhardt. "Characteristic boundary conditions in the lattice Boltzmann method for fluid and gas dynamics". In: *Journal of Computational and Applied Mathematics* 262 (2014). Selected Papers from NUMDIFF-13, pp. 51–61. ISSN: 0377-0427. DOI: https://doi.org/10.1016/j.cam.2013.09.019. URL: http://www.sciencedirect.com/science/article/pii/S0377042713004743.

[48] D. J. Holdych. "Lattice Boltzmann methods for diffuse and mobile interfaces". PhD thesis. University of Illinois at Urbana-Champaign, 2003.

[49] Andreas Hölzer and Martin Sommerfeld. "New simple correlation formula for the drag coefficient of non-spherical particles". In: *Powder Technology* 184.3 (2008), 361–365. ISSN: 0032-5910. DOI: https://doi.org/10.1016/j.powtec.2007.08.021. URL: http://www.sciencedirect.com/science/article/pii/S0032591007004792.

[50] David A. Horner, John F. Peters, and Alex Carrillo. "Large Scale Discrete Element Modeling of Vehicle-Soil Interaction". In: *Journal of Engineering Mechanics* 127.10 (2001), pp. 1027–1032. DOI: 10.1061/(ASCE)0733-9399(2001)127:10(1027). eprint: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9399%

282001%29127%3A10%281027%29. URL: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9399%282001%29127%3A10%281027%29.

[51] G.T. Houlsby. "Potential Particles: a method for modelling non-circular particles in DEM". In: *Computers and Geotechnics* 36 (2009), pp. 953–959. DOI: 10.1016/j.compgeo.2009.03.001.

[52] Haibo Huang, Manfred Krafczyk, and Xiyun Lu. "Forcing term in single-phase and Shan-Chen-type multiphase lattice Boltzmann models". In: *Phys. Rev. E* 84 (4 Oct. 2011), p. 046710. DOI: 10.1103/PhysRevE.84.046710. URL: https://link.aps.org/doi/10.1103/PhysRevE.84.046710.

[53] Y. Ikegawa and J.A. Hudson. "A novel automatic identification system for three-dimensional multi-block systems". In: *Engineering Computations* 9.2 (1992), pp. 169–179. DOI: 10.1108/eb023856.

[54] Salvador Izquierdo and Norberto Fueyo. "Characteristic nonreflecting boundary conditions for open boundaries in lattice Boltzmann methods". In: *Phys. Rev. E* 78 (4 Oct. 2008), p. 046707. DOI: 10.1103/PhysRevE.78.046707. URL: https://link.aps.org/doi/10.1103/PhysRevE.78.046707.

[55] L Jing, Erling Nordlund, and Ove Stephansson. "A 3-D constitutive model for rock joints with anisotropic friction and stress dependency in shear stiffness". In: *International journal of rock mechanics and mining sciences & geomechanics abstracts*. Vol. 31. 2. Elsevier. 1994, pp. 173–178.

[56] Lanru Jing. "Block system construction for three-dimensional discrete element models of fractured rocks". In: *International Journal of Rock Mechanics and Mining Sciences* 37.4 (June 2000), pp. 645–659. DOI: 10.1016/S1365-1609(00)00006-X.

[57] Scott M. Johnson, John R. Williams, and Benjamin K. Cook. "Quaternion-based rigid body rotation integration algorithms for use in particle methods". In: *International Journal for Numerical Methods in Engineering* 74.8 (2007), pp. 1303–1313. DOI: 10.1002/nme.2210. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2210. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2210.

[58] Dale Kolke. *An aerial view of the damaged Oroville Dam spillway*. California Department of Water Resources. 26 February 2017. URL: https://www.facebook.com/CADWR/photos/a.10154476288967449.1073742016.95205192448/10154476289472449/?type=3&theater.

[59] Dinant Krijgsman, Vitaliy Ogarko, and Stefan Luding. "Optimal parameters for a hierarchical grid data structure for contact detection in arbitrarily polydisperse particle systems". In: *Computational Particle Mechanics* 1.3 (Sept. 2014), pp. 357–372. ISSN: 2196-4386. DOI: 10.1007/s40571-014-0020-9. URL: https://doi.org/10.1007/s40571-014-0020-9.

[60] Timm Krüger et al. *The Lattice Boltzmann Method*. Springer, 2017.

[61] A.L. Kupershtokh, D.A. Medvedev, and D.I. Karpov. "On equations of state in a lattice Boltzmann method". In: *Computers & Mathematics with Applications* 58.5 (2009). Mesoscopic Methods in Engineering and Science, pp. 965–974. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2009.02.024. URL: http://www.sciencedirect.com/science/article/pii/S0898122109001011.

[62] A. J. C. Ladd and R. Verberg. "Lattice-Boltzmann Simulations of Particle-Fluid Suspensions". In: *Journal of Statistical Physics* 104.5 (Sept. 2001), pp. 1191–1251. ISSN: 1572-9613. DOI: 10.1023/A:1010414013942. URL: https://doi.org/10.1023/A:1010414013942.

[63] Anthony JC Ladd. "Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation". In: *Journal of Fluid Mechanics* 271 (1994), pp. 285–309. DOI: 10.1017/S0022112094001771.

[64] Chen Lei et al. "Transient stage comparison of Couette flow under step shear stress and step velocity boundary conditions". In: *International Communications in Heat and Mass Transfer* 75 (2016), pp. 232–239. ISSN: 0735-1933. DOI: https://doi.org/10.1016/j.icheatmasstransfer.2015.10.011. URL: http://www.sciencedirect.com/science/article/pii/S0735193315002183.

[65] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. Vol. 98. Siam, 2007.

[66] Randall J LeVeque. *Finite volume methods for hyperbolic problems*. Vol. 31. Cambridge university press, 2002.

[67] Huina Li, Chongxun Pan, and Cass T. Miller. "Pore-scale investigation of viscous coupling effects for two-phase flow in porous media". In: *Phys. Rev. E* 72 (2 Aug. 2005), p. 026705. DOI: 10.1103/PhysRevE.72.026705. URL: https://link.aps.org/doi/10.1103/PhysRevE.72.026705.

[68] D. Lin, C. Fairhurst, and A.M. Starfield. "Geometrical Identification of Three-Dimensional Rock Block Systems Using Topological Techniques". In: *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts* 24.6 (1987), pp. 331–338. DOI: 10.1016/0148-9062(87)92254-6.

[69] Li-Shi Luo. "Unified Theory of Lattice Boltzmann Models for Nonideal Gases". In: *Phys. Rev. Lett.* 81 (8 Aug. 1998), pp. 1618–1621. DOI: 10.1103/PhysRevLett.81.1618. URL: https://link.aps.org/doi/10.1103/PhysRevLett.81.1618.

[70] Guy R McNamara and Gianluigi Zanetti. "Use of the Boltzmann equation to simulate lattice-gas automata". In: *Physical review letters* 61.20 (1988), p. 2332.

[71] Furuichi Mikito and Nishiura Daisuke. "Robust coupled fluid-particle simulation scheme in Stokes-flow regime: Toward the geodynamic simulation including granular media". In: *Geochemistry, Geophysics, Geosystems* 15.7 (2014), pp. 2865–2882. DOI: 10.1002/2014GC005281. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2014GC005281. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014GC005281.

[72] R.D. Mindlin and H. Deresiewicz. "Elastic spheres in contact under varying oblique forces". In: *J. Applied Mech.* 20 (1953), pp. 327–344.

[73] A. Munjiza and K.R.F. Andrews. "NBS contact detection algorithm for bodies of similar size". In: *International Journal for Numerical Methods in Engineering* 43.1 (), pp. 131–149. DOI: 10.1002/(SICI)1097-0207(19980915)43:1<131::AID-NME447>3.0.CO;2-S. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291097-0207%2819980915%2943%3A1%3C131%3A%3AAID-NME447%3E3.0.CO%3B2-S. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-0207%2819980915%2943%3A1%3C131%3A%3AAID-NME447%3E3.0.CO%3B2-S.

[74] Antonio A Munjiza. *The combined finite-discrete element method*. John Wiley & Sons, 2004. ISBN: 9780470020180. DOI: 10.1002/0470020180.

[75] E.G. Nezami et al. "Shortest link method for contact detection in discrete element method". In: *International Journal for Numerical and Analytical Methods in Geomechanics* 30.8 (2006). cited By 70, pp. 783–801. DOI: 10.1002/nag.500. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-33745912823&doi=10.1002%2fnag.500&partnerID=40&md5=437b53f851dfdabfa576e65f3302c05b.

[76] Erfan G. Nezami et al. "A fast contact detection algorithm for 3-D discrete element method". In: *Computers and Geotechnics* 31.7 (2004), pp. 575–587. ISSN: 0266-352X. DOI: https://doi.org/10.1016/j.compgeo.2004.08.002. URL: http://www.sciencedirect.com/science/article/pii/S0266352X04001016.

[77] F. Nicoud and F. Ducros. "Subgrid-Scale Stress Modelling Based on the Square of the Velocity Gradient Tensor". In: *Flow, Turbulence and Combustion* 62.3 (Sept. 1999), pp. 183–200. ISSN: 1573-1987. DOI: 10.1023/A:1009995426001. URL: https://doi.org/10.1023/A:1009995426001.

[78] D. R. Noble and J. R. Torczynski. "A Lattice-Boltzmann Method for Partially Saturated Computational Cells". In: *International Journal of Modern Physics C* 09.08 (1998), pp. 1189–1201. DOI: 10.1142/S0129183198001084. eprint: https://doi.org/10.1142/S0129183198001084. URL: https://doi.org/10.1142/S0129183198001084.

[79] Ruaidhri M. O'Connor. "A Distributed Discrete Element Modeling Environment: Algorithms, Implementation and Applications". AAI0577090. PhD thesis. Cambridge, MA, USA, 1996.

[80] Catherine O'Sullivan and Jonathan D. Bray. "Selecting a suitable time step for discrete element simulations that use the central difference time integration scheme". In: *Engineering Computations* 21.2/3/4 (2004), pp. 278–303.

[81] D. R. J. Owen, C. R. Leonardi, and Y. T. Feng. "An efficient framework for fluid-structure interaction using the lattice Boltzmann method and immersed moving boundaries". In: *International Journal for Numerical Methods in Engineering* 87.1-5 (2010), 66–95. DOI: 10.1002/nme.2985. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2985. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2985.

[82] Randy Pench. *Eroded landscape just below the Oroville Dam Emergency Spillway*. Sacramento Bee. 13 February 2017. URL: http://www.latimes.com/local/california/la-live-updates-oroville-dam-view-from-1487016427-htmlstory.html.

[83] Eric Perkins and John R. Williams. "A fast contact detection algorithm insensitive to object sizes". In: *Engineering Computations* 18.1/2 (2001), pp. 48–62. DOI: 10.1108/02644400110365770. eprint: https://doi.org/10.1108/02644400110365770. URL: https://doi.org/10.1108/02644400110365770.

[84] Michael E. Plesha. "Constitutive models for rock discontinuities with dilatancy and surface degradation". In: *International Journal for Numerical and Analytical Methods in Geomechanics* 11.4 (1987), pp. 345–362. DOI: 10.1002/nag.1610110404. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nag.1610110404. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nag.1610110404.

[85] T.J Poinsot and S.K Lelef. "Boundary conditions for direct simulations of compressible viscous flows". In: *Journal of Computational Physics* 101.1 (1992), pp. 104–129. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(92)90046-2. URL: http://www.sciencedirect.com/science/article/pii/0021999192900462.

[86] Loc Vu-Quoc and Xiang Zhang. "An accurate and efficient tangential force-displacement model for elastic frictional contact in particle-flow simulations". In: *Mechanics of Materials* 31.4 (1999), pp. 235–269. ISSN: 0167-6636. DOI: https://doi.org/10.1016/S0167-6636(98)00064-7. URL: http://www.sciencedirect.com/science/article/pii/S0167663698000647.

[87] Desai C. S. et al. "Thin-layer element for interfaces and joints". In: *International Journal for Numerical and Analytical Methods in Geomechanics* 8.1 (1984), pp. 19–43. DOI: 10.1002/nag.1610080103. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nag.1610080103. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nag.1610080103.

[88] S.A. Galindo-Torres. "A coupled Discrete Element Lattice Boltzmann Method for the simulation of fluid-solid interaction with particles of general shapes". In: *Computer Methods in Applied Mechanics and Engineering* 265 (2013), 107–119. ISSN: 0045-

7825. DOI: https://doi.org/10.1016/j.cma.2013.06.004. URL: http://www.sciencedirect.com/science/article/pii/S0045782513001564.

[89] Xiaowen Shan and Hudong Chen. "Lattice Boltzmann model for simulating flows with multiple phases and components". In: *Physical Review E* 47.3 (1993), p. 1815.

[90] Xiaowen Shan, Xue-Feng Yuan, and Hudong Chen. "Kinetic theory representation of hydrodynamics: a way beyond the Navier-Stokes equation". In: *Journal of Fluid Mechanics* 550 (2006), pp. 413–441. DOI: 10.1017/S0022112005008153.

[91] Gen-Hua Shi. *Working Forum on Manifold Method of Material Analysis, Volume 2. The Numerical Manifold Method and Simplex Integration*. Tech. rep. DTIC Document, 1997.

[92] Gen-Hua Shi and Richard E Goodman. "Discontinuous deformation analysis-a new method forcomputing stress, strain and sliding of block systems". In: *The 29th US Symposium on Rock Mechanics (USRMS)*. American Rock Mechanics Association. 1988.

[93] G.H. Shi. "Discontinuous deformation analysis—a new model for the statics and dynamics of block systems". PhD thesis. University of California, Berkeley, 1988.

[94] Goncalo Silva and Viriato Semiao. "Truncation errors and the rotational invariance of three-dimensional lattice models in the lattice Boltzmann method". In: *Journal of Computational Physics* 269 (2014), pp. 259–279. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2014.03.027. URL: http://www.sciencedirect.com/science/article/pii/S0021999114002083.

[95] Nicholas Sitar, Mary M. MacLaughlin, and David M. Doolin. "Influence of Kinematics on Landslide Mobility and Failure Mode". In: *Journal of Geotechnical and Geoenvironmental Engineering* 131.6 (2005), pp. 716–728. DOI: 10.1061/(ASCE)1090-0241(2005)131:6(716). eprint: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%291090-0241%282005%29131%3A6%28716%29. URL: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%291090-0241%282005%29131%3A6%28716%29.

[96] J. Smagorinsky. "General circulation experiments with the primitive equations". In: *Monthly Weather Review* 91.3 (1963), pp. 99–164. DOI: 10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2. eprint: https://doi.org/10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2. URL: https://doi.org/10.1175/1520-0493(1963)091%3C0099:GCEWTP%3E2.3.CO;2.

[97] S. Succi, E. Foti, and F. Higuera. "Three-Dimensional Flows in Complex Geometries with the Lattice Boltzmann Method". In: *EPL (Europhysics Letters)* 10.5 (1989), p. 433. URL: http://stacks.iop.org/0295-5075/10/i=5/a=008.

[98] Sauro Succi. *The lattice Boltzmann equation: for fluid dynamics and beyond*. Oxford University Press, 2001.

[99] K. Suga et al. "A D3Q27 multiple-relaxation-time lattice Boltzmann method for turbulent flows". In: *Computers & Mathematics with Applications* 69.6 (2015), pp. 518–529. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2015.01.010. URL: http://www.sciencedirect.com/science/article/pii/S0898122115000346.

[100] MC Sukop and DT Thorne Jr. *Lattice Boltzmann Modeling: An Introduction for Geoscientists and Engineers*. Springer, 2006.

[101] William C. Swope et al. "A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters". In: *The Journal of Chemical Physics* 76.1 (1982), pp. 637–649. DOI: 10.1063/1.442716. eprint: https://doi.org/10.1063/1.442716. URL: https://doi.org/10.1063/1.442716.

[102] Kevin W Thompson. "Time dependent boundary conditions for hyperbolic systems". In: *Journal of Computational Physics* 68.1 (1987), pp. 1–24. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(87)90041-6. URL: http://www.sciencedirect.com/science/article/pii/0021999187900416.

[103] C. Thornton and K.K. Yin. "Impact of elastic spheres with and without adhesion". In: *Powder Technology* 65.1 (1991). A Special Volume Devoted to the Second Symposium on Advances in Particulate Technology, pp. 153–166. ISSN: 0032-5910. DOI: https://doi.org/10.1016/0032-5910(91)80178-L. URL: http://www.sciencedirect.com/science/article/pii/003259109180178L.

[104] Takuya Tsuji, Akihito Ito, and Toshitsugu Tanaka. "Multi-scale structure of clustering particles". In: *Powder Technology* 179.3 (2008). WCPT5, pp. 115–125. ISSN: 0032-5910. DOI: https://doi.org/10.1016/j.powtec.2007.07.003. URL: http://www.sciencedirect.com/science/article/pii/S0032591007003348.

[105] Y. Tsuji, T. Kawaguchi, and T. Tanaka. "Discrete particle simulation of two-dimensional fluidized bed". In: *Powder Technology* 77.1 (1993), pp. 79–87. ISSN: 0032-5910. DOI: https://doi.org/10.1016/0032-5910(93)85010-7. URL: http://www.sciencedirect.com/science/article/pii/0032591093850107.

[106] Erez Volk. *CxxTest GitHub Repository*. 2018. URL: https://github.com/CxxTest/cxxtest (visited on 06/20/2018).

[107] Otis R. Walton and Robert L. Braun. "Viscosity, granular-temperature, and stress calculations for shearing assemblies of inelastic, frictional disks". In: *Journal of Rheology* 30.5 (1986), pp. 949–980. DOI: 10.1122/1.549893. eprint: https://doi.org/10.1122/1.549893. URL: https://doi.org/10.1122/1.549893.

[108] P.M. Warburton. "A computer program for reconstructing blocky rock geometry and analyzing single block stability". In: *Computers and Geosciences* 11.6 (1985), pp. 707–712. DOI: 10.1016/0098-3004(85)90013-5.

[109] P.M. Warburton. "Applications of a new computer model for reconstructing blocky rock geometry - analysing single block stability and identifying keystones". In: *Proceedings of the 5th International Congress on Rock Mechanics* (1983), F225–F230.

[110] John R. Williams, Eric Perkins, and Ben Cook. "A contact algorithm for partitioning N arbitrary sized objects". In: *Engineering Computations* 21.2/3/4 (2004), pp. 235–248. DOI: 10.1108/02644400410519767. eprint: https://doi.org/10.1108/02644400410519767. URL: https://doi.org/10.1108/02644400410519767.

[111] EL Wilson. "Finite elements for foundations, joints and fluids". In: *Finite elements in geomechanics* (1977), pp. 319–350.

[112] BH Xu and AB Yu. "Numerical simulation of the gas-solid flow in a fluidized bed by combining discrete particle method with computational fluid dynamics". In: *Chemical Engineering Science* 52.16 (1997), pp. 2785–2809.

[113] Matei Zaharia et al. *Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing*. Tech. rep. UCB/EECS-2011-82. EECS Department, University of California, Berkeley, July 2011. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-82.html.

[114] Matei Zaharia et al. "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing". In: *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX, 2012, pp. 15–28. ISBN: 978-931971-92-8. URL: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia.

[115] Hong Zhang et al. "A new algorithm to identify contact types between arbitrarily shaped polyhedral blocks for three-dimensional discontinuous deformation analysis". In: *Computers and Geotechnics* 80 (2016), pp. 1–15. ISSN: 0266-352X. DOI: https://doi.org/10.1016/j.compgeo.2016.06.007. URL: http://www.sciencedirect.com/science/article/pii/S0266352X16301331.

[116] Hong Zhang et al. "Detection of contacts between three-dimensional polyhedral blocks for discontinuous deformation analysis". In: *International Journal of Rock Mechanics and Mining Sciences* 78 (2015), pp. 57–73. ISSN: 1365-1609. DOI: https://doi.org/10.1016/j.ijrmms.2015.05.008. URL: http://www.sciencedirect.com/science/article/pii/S1365160915001367.

[117] Fei Zheng et al. "A fast direct search algorithm for contact detection of convex polygonal or polyhedral particles". In: *Computers and Geotechnics* 87 (2017), pp. 76–85. ISSN: 0266-352X. DOI: https://doi.org/10.1016/j.compgeo.2017.02.001. URL: http://www.sciencedirect.com/science/article/pii/S0266352X17300277.

[118] Fei Zheng et al. "Object-Oriented Contact Detection Approach for Three-Dimensional Discontinuous Deformation Analysis Based on Entrance Block Theory". In: *International Journal of Geomechanics* 17.5 (2017), E4016009. DOI: 10.1061/(ASCE)GM.1943-5622.0000718.

[119] Donald P. Ziegler. "Boundary conditions for lattice Boltzmann simulations". In: *Journal of Statistical Physics* 71.5 (June 1993), pp. 1171–1177. ISSN: 1572-9613. DOI: 10.1007/BF01049965. URL: https://doi.org/10.1007/BF01049965.

[120] O. C. Zienkiewicz, Robert L. Taylor, and Perumal Nithiarasu. *The finite element method for fluid dynamics.* Oxford : Butterworth-Heinemann, 2014, 2014. ISBN: 9780080951379. URL: https://libproxy.berkeley.edu/login?qurl=http%3a%2f%2fsearch.ebscohost.com%2flogin.aspx%3fdirect%3dtrue%26db%3dcat04202a%26AN%3ducb.b22584019%26site%3deds-live.

[121] Olgierd Cecil Zienkiewicz et al. *The finite element method.* Vol. 36. McGraw-hill London, 1977.

[122] Qisu Zou and Xiaoyi He. "On pressure and velocity boundary conditions for the lattice Boltzmann BGK model". In: *Physics of Fluids* 9.6 (1997), pp. 1591–1598. DOI: 10.1063/1.869307. eprint: https://doi.org/10.1063/1.869307. URL: https://doi.org/10.1063/1.869307.