A COMPUTATIONAL METHODOLOGY FOR
MODELING LARGE SCALE SUBLEVEL CAVING
WITH A THREE-DIMENSIONAL
DISCRETE ELEMENT METHOD

by

Andrew Iver Hustrulid
A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Engineering Systems).

Golden, Colorado
Date 4/2/97

Signed: Andrew Iver Hustrulid

Approved: Dr. Graham G. W. Mustoe
Thesis Advisor

Golden, Colorado
Date 4/2/97

Dr. Joan Gosink
Professor and Head
Engineering Division
ABSTRACT

As the scale of the sublevel caving geometry at Loussavaare-Kiirunavaara AB (LKAB) in Kiruna, Sweden has increased, the assumptions made in the historical material flow studies have become unacceptable. New ideas and methods are required to help understand the blasting and flow mechanics of the caved ore and waste rock. Two research tools are being developed; (1) numerical modeling – specifically the application of the Discrete Element Method (DEM), and (2) in-situ analytical measurements. This thesis develops a computational methodology for simulating large scale sublevel caving with a three-dimensional discrete element method.

To apply the discrete element method to this large problem several technological advances are made. The two-dimensional discrete element implementation started by Zhang (1993) and Mathews (1994) is extended to three dimensions. An improved grid method is used for efficient contact searching allowing simulations to be run in a reasonable length of time. Developmental research includes parallel and object oriented implementations of the discrete element method -- advancing knowledge in these fields. Visualization techniques that take advantage of today’s computer technology are exploited to dramatically present simulation results. The accumulation of these contributions is brought together in the development of the methodology or computational steps for modeling large scale sublevel caving.
# TABLE OF CONTENTS

**ABSTRACT** ................................................................................................................................. III

**TABLE OF CONTENTS** ....................................................................................................................... IV

**LIST OF FIGURES** ............................................................................................................................ VI

**ACKNOWLEDGMENTS** ....................................................................................................................... X

1. **INTRODUCTION** .......................................................................................................................... 1
   1.1 **Problem Statement** .................................................................................................................. 1
   1.2 **Summary of Research** ............................................................................................................. 2
   1.3 **Literature Review** .................................................................................................................... 3

2. **Sublevel Caving** ........................................................................................................................... 10
   2.1 **The Sublevel Caving Mining Method** .................................................................................... 10
   2.2 **Large Scale Sublevel Caving** ................................................................................................ 26
   2.3 **Analytical Tests** ..................................................................................................................... 30
       2.3.1 **Siege Analysis** .................................................................................................................. 30
       2.3.2 **Järnhalmsätare** .............................................................................................................. 31
       2.3.3 **Weighing the LHD Scoop** ............................................................................................. 33
       2.3.4 **Video Recording** ........................................................................................................... 33
       2.3.5 **Marker Recovery** .......................................................................................................... 35
       2.3.6 **Scale Model Tests** ......................................................................................................... 35

3. **The Discrete Element Method** .................................................................................................... 36
   3.1 **Determination and Searching for Contacts** ......................................................................... 40
       3.1.1 **Determining if Two Spheres are in Contact** .................................................................. 41
       3.1.2 **Determining if a Sphere and a Boundary are in Contact** .............................................. 44
       3.1.3 **Locating Possible Contacts** ............................................................................................ 55
   3.2 **Contact Mechanics** .............................................................................................................. 76
   3.3 **Time Integration** ................................................................................................................... 86
   3.4 **Parallel Implementation of the Discrete Element Method** .................................................. 94
       3.4.1 **General Parallel Issues** ................................................................................................. 95
       3.4.2 **The Parallel DEM Algorithm** ......................................................................................... 96
       3.4.3 **Implementation on a 64 Processor Transputer from Alta Technology** .......................... 104
       3.4.4 **Performance Comparison Between Serial and Parallel Implementations** ................ 112
       3.4.5 **Conclusions for Parallel Processors** ............................................................................. 119
   3.5 **Object-Oriented Implementation of the Discrete Element Method** ................................... 120
       3.5.1 **The Object Model** ......................................................................................................... 121
       3.5.2 **The Dynamic Model** ..................................................................................................... 125
4. MODELING SUBLEVEL CAVING WITH THE DISCRETE ELEMENT METHOD ........................................... 138
  4.1 DEFINING THE BOUNDARY CONDITIONS, GEOMETRY, AND CONTACT PARAMETERS .................. 138
  4.2 MODELING STEPS OF THE DEM SUBLEVEL CAVING SIMULATION ........................................... 139
    4.2.1 Step 1 - Generating the Ore in the Ring ............................................................................ 140
    4.2.2 Step 2 - Reducing the Pressures on the Ore Particles ....................................................... 144
    4.2.3 Step 3 - Generating the Caved Waste Material ................................................................. 150
    4.2.4 Step 4 - Settling and Preparing the Ore and Waste Material ............................................. 156
    4.2.5 Step 5 - Blasting the Ring ................................................................................................ 164
    4.2.6 Step 6 - Mining the Material ............................................................................................ 171

5. CONCLUDING REMARKS AND FURTHER WORK ........................................................................... 182
  5.1 CONCLUSIONS ......................................................................................................................... 182
  5.2 FUTURE RESEARCH .............................................................................................................. 184

6. REFERENCES CITED ....................................................................................................................... 186
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Sublevel Caving</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Drift Development</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Drilling the Rings</td>
<td>13</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Drill Pattern</td>
<td>14</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Detonation Sequence</td>
<td>17</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Explosive Loading</td>
<td>18</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Explosive Loading Pattern</td>
<td>20</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Blasting a Ring</td>
<td>21</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Mining the Ore</td>
<td>22</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Muck Pile at the Angle of Repose and Rankine Angle</td>
<td>23</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Comparison of Old and Current Sublevel Caving Geometry</td>
<td>26</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Additional Available Swell Volume</td>
<td>28</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Discrete Element Computation Cycle</td>
<td>37</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Development Areas of the Discrete Element Method</td>
<td>39</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Two Spheres in Contact</td>
<td>41</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Contact Elimination Using Cubes</td>
<td>43</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Contact Between a Sphere and a Boundary</td>
<td>45</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Contact Regions</td>
<td>46</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Convex Boundary Divided into Triangles</td>
<td>47</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Definition of Beta Unit Vectors</td>
<td>48</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Area Coordinate System</td>
<td>49</td>
</tr>
<tr>
<td>Figure</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>22</td>
<td>Contact Location Using Area Coordinates</td>
<td>50</td>
</tr>
<tr>
<td>23</td>
<td>Correct Regions for Contact Location</td>
<td>51</td>
</tr>
<tr>
<td>24</td>
<td>Sphere Center Location with Respect to an Edge</td>
<td>53</td>
</tr>
<tr>
<td>25</td>
<td>Corner Contact</td>
<td>54</td>
</tr>
<tr>
<td>26</td>
<td>Simulation Extents Divided into Cells</td>
<td>59</td>
</tr>
<tr>
<td>27</td>
<td>Example Particle Configuration</td>
<td>60</td>
</tr>
<tr>
<td>28</td>
<td>Tagged Grid Cells</td>
<td>63</td>
</tr>
<tr>
<td>29</td>
<td>Cells Checked for Contact</td>
<td>64</td>
</tr>
<tr>
<td>30</td>
<td>Sample Arrangement of Spheres</td>
<td>66</td>
</tr>
<tr>
<td>31</td>
<td>First Particle Distribution</td>
<td>68</td>
</tr>
<tr>
<td>32</td>
<td>Second Particle Distribution</td>
<td>69</td>
</tr>
<tr>
<td>33</td>
<td>Third Particle Distribution</td>
<td>71</td>
</tr>
<tr>
<td>34</td>
<td>Cell Size Set to Maximum Boundary Extents</td>
<td>73</td>
</tr>
<tr>
<td>35</td>
<td>Cells Tagged for Boundary Contact with Traditional Grid Method</td>
<td>74</td>
</tr>
<tr>
<td>36</td>
<td>Cells Tagged for Boundary Contact with Improved Grid Method</td>
<td>75</td>
</tr>
<tr>
<td>37</td>
<td>Contact Model</td>
<td>76</td>
</tr>
<tr>
<td>38</td>
<td>Maximum Expected Velocity</td>
<td>77</td>
</tr>
<tr>
<td>39</td>
<td>Damping Between Two Discrete Elements</td>
<td>79</td>
</tr>
<tr>
<td>40</td>
<td>Global Viscous Damping</td>
<td>88</td>
</tr>
<tr>
<td>41</td>
<td>Single Degree of Freedom System</td>
<td>90</td>
</tr>
<tr>
<td>42</td>
<td>Velocity Extrapolation</td>
<td>93</td>
</tr>
<tr>
<td>43</td>
<td>Parallel Flow Diagram</td>
<td>98</td>
</tr>
<tr>
<td>44</td>
<td>Geometrical Division</td>
<td>101</td>
</tr>
<tr>
<td>45</td>
<td>Processor Communication Scheduling</td>
<td>103</td>
</tr>
<tr>
<td>46</td>
<td>Parallel Processing Lab at Colorado School of Mines</td>
<td>105</td>
</tr>
</tbody>
</table>
Numerous people have supported this investigation; technologically, financially, and emotionally. To those individuals, I extend my gratitude. I would particularly like to thank:

Ginger for her loving support, faith, and editorial help though out this investigation.

Dr. Graham G.W. Mustoe for his ideas, encouragement, flexibility, and friendship over the past 7+ years.

LKAB for financial support through research contracts, CA01665 and KD96120343.

Dr. William Hustrulid and Dr. Carlos Quintera from LKAB for their patience, ideas, and support.

Dr. Rudolf and Dagmier Kvatil for being an “extended family” while I was in Tucson.

Mr. Erik Anderson and Mr. Grzegorz Dewicki for being my friends.
Chapter 1

INTRODUCTION

An important class of industrial problems involves the control and prediction of solid material flow that behaves in a highly discontinuous manner. Examples of such flow are solid material in conveyor transfer stations, hoppers, silos, and caving methods in mining. In order to solve material flow problems in terms of engineering data that can be used in the design process, computational and experimental methods need to be developed and validated.

The major objective of the work in this thesis is the development and application of a robust and efficient numerical modeling procedure based upon the Discrete Element Method (DEM) that can be employed as a computational tool in the solution of the material flow in sublevel caving operations. DEMs are a family of numerical modeling techniques specifically designed to solve problems in engineering and applied science that exhibit gross discontinuous mechanical behavior. The DEM explicitly models the dynamic motion and mechanical interaction of each body or particle throughout a simulation, and provides a detailed description of the positions, velocities and forces acting on each body or particle at discrete points in time, called time steps, during the analysis. This research includes both experimental and analytical work. The advanced development of the DEM technique requires the application of several computation tools including parallel processing, object-oriented programming, sophisticated data structures, and scientific visualization techniques for the processing of the data generated in discrete element analyses.

1.1 Problem Statement

A particular example of material flow problems of industrial significance is the ore extraction associated with the sublevel caving mining method. Material flow problems
associated with ore extraction in the material handling and mining disciplines require sophisticated computer simulation tools for solution. Many flow problems are characterized by highly discontinuous three-dimensional flow behaviors that cannot be easily characterized with in situ or laboratory measured data.

For underground mining at LKAB, increases in the geometry of the sublevel caving pattern has changed the behavior of the material flow. Scale model tests have failed to reproduce field measurements. The difference resulting from the increases in the caving geometry is believed to be in the material compaction and fragmentation resulting from the blasting with limited swell volume. A new method is developed to help study and understand the material flow in the increased geometry of the sublevel caving.

The sublevel caving problem and others of a similar material flow nature can potentially be modeled accurately with the DEM. Current computer algorithms based upon the DEM, however, require significant development before they can be used effectively to handle problems of this magnitude and complexity.

1.2 Summary of Research

Two main research focuses have been performed to meet the overall objective of this work. These areas are: (1) the development of a three-dimensional DEM computer algorithm, and (2) application of the DEM model to the sublevel caving problem.

A brief summary of the description of these two areas are given below:

   - Development of DEM in three dimensions.
   - Fast contact checking algorithm for computational speed.
   - Object Oriented implementation for maintainability.
   - Parallel Implementation for required speed.
   - Physical based contact mechanics for accurate simulation.
• Visualization tools to present and help interpret numerical data.

2. Application and development of the DEM to Sublevel Caving.

• Understand the procedure of the sublevel caving mining method as practiced by LKAB.

• Obtain as much information as possible from in-situ tests.

• Generate the sublevel caving geometries, initial blasting conditions, and material extraction sequence.

• Draw conclusions about the modeling of sublevel caving mining with the discrete element method and suggest further research directions.

1.3 Literature Review

The basis of the discrete element method was developed in the early 1970’s (Cundall 1971a; Burman 1971; Rodriguez-Ortiz 1974; Hocking 1977). Of these initial authors, Peter Cundall has been the most active. This section presents an overview of the discrete element method from its inception to Ph.D. theses completed during the last year. The objective of this review is to present the different areas of discrete element research. This objective will help define how this research adds to the current level of discrete element technology.

Two conferences have been held specifically on the Discrete Element Method (Mustoe et al. 1989; Williams et al. 1993b). The proceedings from these conferences are excellent sources of information on the discrete element method.

The basis of the Discrete Element Method was first presented in Peter Cundall’s Ph.D. thesis at the Imperial College in London (Cundall 1971a). A paper published that same year presents a two-dimensional model with polyhedral and cylindrical elements (Cundall 1971b). The discrete element method was generalized at the University of Minnesota through research supported by the U.S. Army Corps of Engineers as part of their efforts in the Rational Design of Tunnel Supports (Cundall 1974; Cundall et al. 1975). The numerical method was referred to
as the “Interactive Graphics Method” resulting from the use of a Cathode Ray Tube (CRT) for the input and output of the model data. The computational portion of the implementation was written in assembly language for efficiency. The numerical method referred to as the “Discrete Block Method” is used to begin modeling material compaction type problems in Cundall (1976). Three papers that provide an overview of the DEM and its application to rock mechanics and geotechnical engineering are (Cundall 1987; Hart 1991; Hart et al. 1992). A paper at the First U.S. Conference on the Discrete Element Method presents a comprehensive overview of the discrete element method, a literature survey, and proposed future developments (Cundall 1989a). The work by Cundall on discrete elements can be divided into two fields; (1) polyhedral blocks in two and three dimensions, and (2) particles modeled as discs and spheres. These areas are now presented separately.

A technical report for the U.S. Army Engineer Waterways Experiment Station by Dames and Moore includes several advancements to the discrete element method. (Maini et al. 1978). The work includes a FORTRAN implementation of the original assembly language code, simple deformable objects, crackable objects, a discussion on rock joints, and the introduction of fully deformable two-dimensional blocks. The numerical method is referred to as the “Distinct Element Method” in this report. The Universal Distinct Element Code (UDEC) was developed under a contract from the U.S. Army (Cundall 1980b) to model the behavior of jointed rock. The model was further refined in a project for the U. S. Army Engineer Waterways Experiment Station (Cundall et al. 1985). A three-dimensional version of UDEC called 3DEC is also introduced in this report. Lemos applied the UDEC program in his Ph.D. work in analyzing concrete gravity dams on jointed rock foundations subject to earthquakes (Lemos 1987). The contact detection scheme and associated data structure for systems composed of many polyhedral blocks are presented in Part I of a paper on the formulation of a three-dimensional distinct element model (Cundall 1988b). Part II of the paper covers the calculations for motion and interaction of the polyhedral blocks (Hart et al. 1988). These advances are incorporated in the UDEC and 3DEC programs that are commercially available from the Itasca Consulting Group.
A major research investigation was done at the University of Minnesota and Dames and Moore funded by the National Science Foundation on the development of discrete element algorithm to model particles. Part I of the report (Strack et al. 1978) develops a two-dimensional model with cylindrical discs. The section describing the numerical model was reprinted in Geotechnique (Cundall et al. 1979b) and is commonly referenced. The computer source code for the two-dimensional model, BALL, is included in the appendix to the NSF report in a Technical Note by Dames and Moore (Cundall 1978). Part II of the report continues the development on the model by introducing porosity and fabric calculations for two-dimensional particulate models (Cundall et al. 1979c). The introduction of a three-dimensional particulate model with spheres, TRUBAL, in included with source code in this NSF report. A paper on the development of the constitutive laws for soil using the distinct element includes most of the development and results of the research for NSF (Cundall et al. 1979a). A second NSF research project was undertaken from 1980-1982 in an attempt to understand the basic micro-mechanics of granular media by analyzing results of numerical experiments on assemblies of discs, using the computer program BALL (Cundall et al. 1983a). Several published papers on this topic are included in the appendices of the NSF report (Cundal et al. 1982; Cundall et al. 1983b; Cundal 1980a; Cundall 1983a; Cundall et al. 1983c; Cundall 1983b; Marti 1980; Cundall 1983c; Strack 1980; Drescher 1982). The BALL and TRUBAL discrete element programs have continued to be developed at the Itasca Consulting Group under the direction of Dr. Cundall. Both programs are available commercially as Particle Flow Code in Two Dimensions - PFC2D (Itasca 1995) and PFC3D.

Dr. Graham G.W. Mustoe, Dr. John Williams, and Dr. Grant Hocking worked together on the discrete element method in the early and mid 1980's for Intera in the USA (Williams et al. 1985; Hocking et al. 1987; Mustoe et al. 1987). Their work addresses the prediction of ice forces on offshore oil exploration structures and ships. These developments included modeling generalized three-dimensional contact of polyhedral blocks, fracture of brittle plates, and the generation of ice force-displacement laws for ice-structure interaction problems. This work continues at Intera today (Worgan et al. 1989; Worgan et al. 1993).
Mustoe continued this work on the discrete element method with research on plate and beam elements that are used to model deep-ocean pipes in two dimensions (Mustoe 1989; Mustoe et al. 1992; Mustoe 1992c). The beam formulation was later used by Hustrulid (1993) to model conveyor belt systems in two dimensions in his M.Sc. thesis under Mustoe, and by Zhang (1996) to model flexible boundaries for material compaction in two dimensions for a Ph.D. under Dr. Stein Sture. A similar discrete element formulation is used to model a rockfall fence (Mustoe et al. 1993; Mustoe 1993b). Greening, Mustoe and DePoorter have worked on modeling ceramic processing with the discrete element method (Greening et al. 1989). This work recently resulted in Greening’s Ph.D. under Mustoe and DePoorter (Greening 1996). Two M.Sc. theses have been completed under Mustoe on a two-dimensional discrete element method with circular disks to model material flow (Zhang 1993; Mathews 1994). Zhang and Mustoe used this model to attempt to simulate hydraulic flow phenomena (Zhang et al. 1993). The two-dimensional particle model has been advanced to three dimensions to model material flow at conveyor belt transfer stations (Hustrulid et al. 1996). This thesis documents the advancement to three dimensions and the application to large scale sublevel caving. With Ph.D. student, Scott Oelfke, Mustoe has also developed discrete-finite element hybrid algorithms that model pillar strength and floor heave in underground mines (Oelfke et al. 1994; Oelfke et al. 1995; Oelfke et al. 1996). Another area of research by Mustoe has been in irregular shaped objects such as superquadrics (Mustoe et al. 1993a). Mustoe recently worked with Dr. Xiaoshan Lin, a research fellow at Colorado School of Mines, and Dr. Nakagawa, on three-dimensional compaction problems using ellipsoids and two-dimensional flow is hoppers with a discrete element model (Lin et al. 1995a; Lin et al. 1996; Nakagawa et al. 1996).

Pande et al. (1990) provide a comprehensive overview of modeling rigid bodies with the discrete element method. The use of superquadrics and hyperquadrics has been pioneered by Williams and Pentland (1989b) at the Massachusetts Institute of Technology (MIT). Williams et al. (1989a) has continued modeling ice islands impacts with arctic offshore production structures. Williams and O’Connor have developed a contact detection scheme based on a heap sorting algorithm, and an object representation method called the discrete
function representation (DFR) to significantly improve the contact resolution performance between irregular shaped objects (O’Connor et al. 1993; Williams et al. 1995a). Much of this work including a distributed discrete element method is included in O’Connor’s Ph.D. work (O’Connor 1996). Williams and Rege (1996) have investigated the formation of coherent structures within a deforming granular material using the MIMES (Williams et al. 1995b) computer program at MIT. Williams and Armatunga (1993) have introduced the use of wavelets for hierarchical geometry representation in the discrete element method.

Dr. Xiaoshan Lin completed his Ph.D. (Lin 1995) under Dr. Tang-Tat Ng at the University of New Mexico. Dr. Lin’s work is on compaction of ellipsoids in three dimensions. He recently completed a research fellowship at Colorado School of Mines with Dr. Graham Mustoe incorporating the ellipsoid contact models into the TRUBAL program maintained at the University of Aston by Lian, Thorton, and Kafui (1994).

The researchers at the University of Aston have been developing advanced contact and adhesion models between elements. This work includes the conglomeration and fracture of conglomerate particles.

A detailed study on modeling soil mechanics with a two-dimensional discrete element model with circular discs in presented by Ting et al. (1989). To avoid the problem of excessive rotation of circular discs a suite of three programs to preprocess, solve, and post process discrete element simulations of two-dimensional ellipses is presented by Ting and Corkum (1992). The methodology for determining contacts between two-dimensional ellipses is presented in detail by Ting (1992). The contact determination between an ellipse and a boundary is presented by Ting et al. (1993a). The effects of particle shape, based on different aspect ratio ellipses, subject to compaction is presented by Ting et al. (1993b) and later expanded upon by Ting et al. (1995). Several earlier papers by Ting and Corkum have not been reviewed but are included in the references for interested readers.

Mishra and Rajamai at the University of Utah have applied the discrete element method to the simulation of ball mills (Mishra et al. 1992a; Mishra et al. 1992b). Their work has included the effects of lifter bars on the charge motion (Mishra et al. 1993). To attempt to
validate the numerical simulations a computer analysis of videos of material motion in scale model physical tests have been completed (Mishra et al. 1994a). Initial studies of ball mills with a three-dimensional discrete element model are also reported (Mishra et al. 1994b). Several earlier publications by Mishra and Rajamai have not been reviewed but are included in the references section for the interested reader.

A general overview of the Distinct Motion Code (Taylor et al. 1989a) including a stick limited friction model is presented by Taylor and Preece (1989b) at the First U. S. DEM Conference. To simulate the bulking of blasted material the radius of circular particles are increased as they rotate. This dilatation of material is compared with experimental results (Preece et al. 1990). Preece (1991) presents a porosity calculation method for a two-dimensional discrete element model with discs. The porosity after blasting for a two-level oil shale retort is calculated. In the numerical modeling of blasting, the porosity calculation is used to couple the discrete element method with a gas flow calculation using a finite difference method. This work presents promising results in two dimensions (Preece et al. 1992). The coupled gas flow and rock motion has also been used to model bench blasting at surface coal mines (Preece et al. 1993b). The influence of the material packing angle on bench blasting simulating in two dimensions is investigated by Preece and Burchell (1993a).

Walton and Braun have modeled the flow behavior of inelastic, frictional particles in rotating drums. Their work includes the clustering of spheres to form irregular shaped objects (Walton et al. 1993). The flow of spheres in chutes is investigated by Walton (1993) and Drake and Walton (1995).

An initial DEM study of sublevel caving in a nickel mine was done for INCO by Itasca using the PFC2D DEM computer program (McKinnon 1995). Only two-dimensional DEM models of sublevel caving were run. The particles were generated using a random number generator and allowed to settle. There was no pre-stress on the blasted ore. The particles were mucked out and the waste concentration measured for each scoop. The conclusions of the Itasca report are that shorter, fatter, rings will result in the highest extraction efficiency and that
some improvement may be made by studying the mechanics of the extraction of several rings in succession.

The report by Itasca is an initial step in numerical modeling of sublevel caving, however, there are several areas where improvements can be made:

- The simulations were done in two-dimensions, ignoring the three-dimensional influences on the flow.

- No steps of inducing a pre-stress resulting from the blasting of the ore ring are performed.

- The ore is drawn out in very large scoops taken in rapid succession. The combination of the large scoops and the continuous dynamic motion of the material draw may be significantly influencing the modeling results and consequent conclusions.
Chapter 2

SUBLEVEL CAVING

The Sublevel Caving (SLC) mining method is the primary extraction technique used by Loussavaara-Kiirunavaara AB (LKAB) in Kiruna, Sweden. Increases in the scale of SLC geometry, favorable from an economic standpoint, have changed the assumptions supporting the traditional studies on the mechanics of SLC. The original techniques used to study the mechanical behavior of SLC cannot handle the current large scale SLC boundary conditions, therefore, new, different, mechanical modeling methods are needed.

This research is part of a major effort towards a better understanding of sublevel caving at LKAB; including a better understanding of dilution for the process of extracting ore. This chapter introduces the reader to the sublevel caving mining method, the differences in the geometry and swell volume defining “large scale” sublevel caving, and the analytical tests that can be performed in the mine. The author participated in many of these tests during the summer of 1995 to gain a better understanding of the sublevel caving mining method as implemented by LKAB.

2.1 The Sublevel Caving Mining Method

The sublevel caving mining method, depicted in Figure 1 (Hamrin 1986), is the primary ore extraction method used at LKAB’s iron ore mines in Kiruna and Malmberget, Sweden. The nickel mines owned and operated by INCO in Sudbury, Canada also use sublevel caving.
Figure 1 Sublevel Caving.
Mining with the sublevel caving mining method can be broken down into four steps.

1. Drift development
2. Fan drilling
3. Explosive loading and blasting
4. Ore extraction

The steps are described in a general manner so that readers without a mining background can understand the sublevel caving mining method.

The first step is the development of the drifts through the ore body. The drifts are driven horizontally in a checkerboard pattern as shown in Figure 2. Figures 2, 3, 8, and 9 are reproduced from a LKAB flyer.

![Figure 2 Drift Development.](image)

At LKAB the drifts are nominally 7 m wide and 5 m high. The drifts are spaced at 25 m horizontally with 27 m between each level. The cost of drifting is approximately 10,000 SEK/m or 62 SEK/tonne assuming drift dimensions of 7m x 5m and a material density of 4.6 tonne/m³.

After a drift is developed, a fan pattern, as shown in the right half of Figure 3, is drilled upwards. The fan pattern is repeated every 3 meters along the length of the drift creating separate rings of ore. The burden of each ring is said to be 3 meters. Each fan is made up of ten holes with a diameter of 115 mm. The fan pattern is angled at 80°, as shown in the left half
of Figure 3, for two reasons; (1) to improve the ore recovery during extraction, and (2) to keep the water and rock from coming straight down on the machinery during drilling.

Figure 3 Drilling the Rings.

The fan pattern varies at the beginning and the ends of the drift. The geometry of the surrounding drifts and geology also influences the fan pattern. Different drill machines will also require variations in the drilling patterns. The modeling in this thesis focuses on the “typical” ring in the center of the drift with uniform surround drift geometry and geology. An example drill pattern is shown in Figure 4.
Figure 4 Drill Pattern.
The location of each drill hole is specified with the boom angle, the offset from the drift centerline and the distance above the floor of the drift. The distance above the floor of the drift is 1.45 m for the example drill patterns used with single boom and dual boom drill rigs listed in Table 1.

Table 1 Drill Hole Data.

<table>
<thead>
<tr>
<th>Hole</th>
<th>Single Boom Drill Machine</th>
<th>Dual Boom Drill Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boom Angle (deg)</td>
<td>Offset (m)</td>
</tr>
<tr>
<td>1</td>
<td>60</td>
<td>-1.5</td>
</tr>
<tr>
<td>2</td>
<td>69</td>
<td>-1.2</td>
</tr>
<tr>
<td>3</td>
<td>76</td>
<td>-0.8</td>
</tr>
<tr>
<td>4</td>
<td>83</td>
<td>-0.5</td>
</tr>
<tr>
<td>5</td>
<td>87.7</td>
<td>-0.2</td>
</tr>
<tr>
<td>6</td>
<td>92.3</td>
<td>0.2</td>
</tr>
<tr>
<td>7</td>
<td>97</td>
<td>0.5</td>
</tr>
<tr>
<td>8</td>
<td>104</td>
<td>0.8</td>
</tr>
<tr>
<td>9</td>
<td>111</td>
<td>1.2</td>
</tr>
<tr>
<td>10</td>
<td>120</td>
<td>1.5</td>
</tr>
</tbody>
</table>

The holes are drilled to a depth that reaches a damage or safety zone of the surrounding drifts. Drill holes 1 and 10 are drilled to the centerline of the adjacent drifts. Holes 2, 3, 8, and 9 are drilled until they meet a safety zone that is 2 m thick surrounding the adjacent drift. Holes 4, 5, 6, and 7 are drilled until the distance of the end, measured perpendicular to the outermost holes drilled from the drift above, is 1.5 m. This is also a type of safety boundary. In practice the long center holes, numbers 4-7, in the ring pattern are not drilled straight and wander some. However, all of the drill holes are considered to be straight in this analysis.

Before a ring of ore can be mined is must first be fractured and broken with explosive. LKAB uses and explosive called Kimulux with 5% aluminum added. The physical properties of the Kimulux with 5% aluminum are listed in Table 2.
Table 2 Properties of Kimulux with 5% Aluminum.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density</td>
<td>1200   kg/m³</td>
</tr>
<tr>
<td>Oxygen Balance</td>
<td>-0.94  %</td>
</tr>
<tr>
<td>Gas Volume (NTP)</td>
<td>828.93 L/kg</td>
</tr>
<tr>
<td>Explosion Pressure</td>
<td>29921.72 ATM</td>
</tr>
<tr>
<td>Explosion Temperature</td>
<td>2638.39 K</td>
</tr>
<tr>
<td>Heat of Explosion</td>
<td>907.69 kcal/kg</td>
</tr>
<tr>
<td></td>
<td>3.80 MJ/kg</td>
</tr>
<tr>
<td>Heat of Reaction (Constant Pressure)</td>
<td>885.79 kcal/kg</td>
</tr>
<tr>
<td></td>
<td>3.71 MJ/kg</td>
</tr>
<tr>
<td>Weight Strength Rel. LFB</td>
<td>0.80</td>
</tr>
<tr>
<td>Rel. ANFO</td>
<td>0.95</td>
</tr>
<tr>
<td>Rel. DXM</td>
<td>0.86</td>
</tr>
<tr>
<td>Velocity of Detonation (Given)</td>
<td>5500 m/sec</td>
</tr>
<tr>
<td>Velocity of Detonation (Measured)</td>
<td>5100 m/sec</td>
</tr>
</tbody>
</table>

The Kimulux is not detonated directly. To initiate the Kimulux a booster, blasting cap, and prima cord are used. The prima cord is lit in the drift and burns at a velocity of detonation of approximately 7000 m/sec until it reaches a blasting cap attached at the other end. A delay number specifying a multiple of a 25 msec delay is associated with the blasting cap. For example, a #8 blasting cap has a delay of 200 msec. The blasting cap is inserted into a booster. When the blasting cap detonates the booster also detonates. The detonation of the booster has enough power to initiate the Kimulux. The system of detonation from the prima cord to the Kimulux is depicted in Figure 5.
Figure 5 Detonation Sequence.

The booster, blasting cap, and prima cord are pushed into the drill holes with an explosive loading machine. After the booster, blasting cap, and prima cord are located in the drill hole the explosive loading machines loads the drill hole with the Kimulux explosive. The photograph in Figure 6 shows the explosive loading machine about to push the booster, blasting cap, and prima cord into hole number 10 of the drill pattern. The Kimulux will then be loaded into the hole with the same machine through the hose also illustrated in Figure 6.
Figure 6 Explosive Loading.
Each drill hole will have one or two detonators and primers. The delays on the
detonators are different for each hole. To protect the brow of the drift the ends of the drill
holes entering the drift are left unloaded for some distance. The distance left unloaded, the
number and locations of the boosters, and the delays of the boosters vary depending on the
geology, fan pattern, and the personnel loading the explosive. The unloaded length, number of
boosters, and delay and location of the booster for an example explosive loading pattern are
listed in Table 3. The unloaded length and booster locations are measured along the length of
the hole from the top of the drift.

Table 3 Explosive Loading Pattern.

<table>
<thead>
<tr>
<th>Hole</th>
<th>Unloaded Length (m)</th>
<th>Number of Boosters</th>
<th>Booster 1 Location (m)</th>
<th>Booster 1 Delay (#/msec)</th>
<th>Booster 2 Location (m)</th>
<th>Booster 2 Delay (#/msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>12/300</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>2</td>
<td>7</td>
<td>11/275</td>
<td>20</td>
<td>11/275</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>2</td>
<td>13</td>
<td>10/250</td>
<td>20</td>
<td>10/250</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>9/225</td>
<td>20</td>
<td>9/225</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>2</td>
<td>11</td>
<td>8/200</td>
<td>20</td>
<td>8/200</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>2</td>
<td>11</td>
<td>8/200</td>
<td>20</td>
<td>8/200</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>9/225</td>
<td>20</td>
<td>9/225</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>2</td>
<td>13</td>
<td>10/250</td>
<td>20</td>
<td>10/250</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>2</td>
<td>7</td>
<td>11/275</td>
<td>20</td>
<td>11/275</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>12/300</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

The explosive pattern listed in Table 3 is depicted in Figure 7.
Figure 7 Explosive Loading Pattern.
In practice two or three consecutive fan patterns in a drift will be loaded with explosive at a time. Typically when mining only a single ring at a time is detonated and then mined out. A ring being blasted and the resulting fractured and broken ore are depicted in Figure 8.

![Blasting a Ring](image)

Figure 8 Blasting a Ring.

After the ring is blasted the ore is ready to be mined out. LKAB uses 13 and 15 tonne diesel and electric Load-Haul-Dump (LHD) machines to mine out the ore. The LHD takes a scoop from the muck pile of material and then transports the material to an ore pass where the material is dumped. The material in the ore pass is loaded on an underground train, transported to a crusher and crushed, and finally hoisted to the surface for further processing. The cost of mining ore from the ring is approximately 35 SEK/tonne. The mining process with the LHD is illustrated in Figure 9.
After the blast, the angle of the surface of muck pile is at the angle of repose. LKAB has measured the angle of repose to be in the range of 38° - 40° for the material in the Kiruna mine. The Load-Haul-Dump (LHD) machine then comes in and takes a scoop of material from the muck pile. The angle of the muck pile surface gets steeper but no flow occurs. The LHD continues to take scoops from the muck pile with no significant flow occurring until the slope of the muck pile reaches the Rankine angle. The Rankine angle is the angle at which the slope of the muck pile becomes unstable and fails, causing a significant amount of material to flow. LKAB has measured the Rankine angle to be in the range of 45° - 48° for the muck piles in the Kiruna mine. When the Rankine angle is reached, material flow occurs until the slope of the muck pile reaches the angle of repose. Figure 10 shows the muck pile at the angle of repose and Rankine angle.
Figure 10 Muck Pile at the Angle of Repose and Rankine Angle.
The volumes of the LHD scoops are reported by the manufacturer to be 5.75 m³ for the 13.5 tonne LHD and 10.0 m³ for the 25 tonne LHD. It is not currently known whether these volumes are the heaped or struck capacities. The LHD scoop volume can also be estimated using the mass per scoop and ore/waste concentration in the scoop. The intact ore and waste densities are about 4600 kg/m³ and 2700 kg/m³ respectively. The swell is about 50% for both so the density for the broken ore is about 3000 kg/m³ and for the waste 1800 kg/m³. With the weighing device on a 13.5 tonne LHD the average load per bucket is measured at 15 tonne when it is mostly ore and about 12 tonne when it is perhaps 50% waste by weight. If one assumes that the average waste is 25% then the average weight per scoop is 13 tonne. Hence, it is assumed that the average 13.5 tonne scoop takes 5.0 m³ of material and the 25 tonne scoop takes 9.26 m³ of material.

The number of scoops between the failure of the muck face slope producing large flows represents the highest natural frequency of the sublevel caving mucking process. The highest natural frequency is calculated ignoring the effects of hang-ups and large boulders. Hang-ups and boulders will decrease the natural frequency. In calculating the highest frequency, it is also assumed that the loads are taken equally from the left and the right side of the muck face. The highest frequency of the sublevel caving mucking process can be calculated knowing the scoop volume, drift dimensions, angle of repose, and the Rankine angle with equation [2.1].

\[ f_n = \frac{wh^2}{2} \left( \frac{1}{\tan \theta_1} - \frac{1}{\tan \theta_2} \right) \frac{\text{ScoopVolume}}{\text{ScoopVolume}} \]

(2.1)

where

- \( w \) = the width of the drift (7 m)
- \( h \) = the height of the drift (5 m)
- \( \theta_1 \) = angle of repose (38° - 40°)
- \( \theta_2 \) = Rankine Angle (45° - 48°)

The highest natural frequency is used to estimate the minimum required frequency for sample collection. Too sparse of a sample frequency will miss important events and may lead
to inaccurate conclusions. Sampling every scoop requires increased work in collection and analysis and slows the loading process. The Nyquist sample criterion states that the sample rate should be selected to be at least twice the highest-frequency component present in the signal (Rizzoni 1993). A range of highest natural frequencies and minimum sampling frequencies are given for the 13 tonne and 25 tonne LHD's in Table 4.

Table 4 Minimum Sampling Frequencies.

<table>
<thead>
<tr>
<th>Angle of Repose</th>
<th>Rankine Angle</th>
<th>13.5 Tonne LHD</th>
<th>25 Tonne LHD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Natural Frequency (scoops/sample)</td>
<td>Minimum Sample Frequency (scoops/sample)</td>
<td>Natural Frequency (scoops/sample)</td>
</tr>
<tr>
<td>Worst Case</td>
<td>40°</td>
<td>45°</td>
<td>3.356</td>
</tr>
<tr>
<td>Best Case</td>
<td>38°</td>
<td>48°</td>
<td>6.642</td>
</tr>
<tr>
<td>Average Case</td>
<td>39°</td>
<td>46.5°</td>
<td>5.004</td>
</tr>
</tbody>
</table>

Sampling every scoop or every other scoop may be impractical because of the impacts on the production. Several factors should be considered in applying the minimum sampling frequencies in Table 4. If hang-ups or large boulders occur, the natural frequency will decrease resulting in a decreased sampling frequency. Also once the slope of the pile reaches the Rankine angle the LHD operator may have the opportunity to take an additional scoop beyond the Rankine angle as the slope fails. The additional scoop increases the volume taken between large flows and decreases the required sampling frequency. The natural frequency of the sublevel caving system can also be estimated by watching the mining process for the large flows of material. The ore/waste sampling should be taken at least twice as often as the occurrences of large material flows.
2.2 Large Scale Sublevel Caving

The scale of sublevel caving geometry practiced by LKAB has increased greatly over the past few years. Increasing the distances between drifts reduces the number of drifts that need to be developed and most importantly the developmental costs. Today the sublevel interval at LKAB is 27m, the sublevel drift spacing is 25m, and the burden is 3m. The front drift inclination is 80 degrees. Figure 11 shows one of the older sublevel caving geometries in comparison to the current geometry. Figure 11 also shows the 20% and 50% swell volumes for the two geometries.

![Current Geometry](image)

![Old Geometry](image)

Figure 11 Comparison of Old and Current Sublevel Caving Geometry.

The swell volume levels indicate the volume of material that will have the opportunity to not only fracture but break apart and loosen during blasting. The 50% swell volume level assumes that the volume after blasting will be 150% of the original volume. In practice the
swell volume will not be constant as assumed in Figure 11 but will vary with the most swelling occurring near the drift and decreasing with increasing height in the ring.

The swell volumes indicated in Figure 11 assume that the angle of the front of the muck pile is equal before and after the blast and that the only volume available for material to swell into during blasting is the drift. In practice, additional swell volume is available in the lower regions of the ring as shown in Figure 12. Leaving the muck pile at an angle greater than the angle of repose prior to blasting will also create additional swell volume.
Figure 12 Additional Available Swell Volume.
The change in the volume of material that has the ability to swell is one of the defining characteristics of Large Scale Sublevel Caving. In the old geometry, shown in Figure 58, the majority of the ore in the ring had the opportunity to break apart and loosen, with the current sublevel caving geometry only the lower portion of the ring can loosen up. This change in the initial conditions of extracting the ore with the LHD is what defines Large Scale Sublevel Caving and necessitates this research.

The traditional theory of the flow of the caved material developed by Janelid and Kvapil (1966) was developed using small scale models. These models were constructed by carefully filling the models with loose material. These models and the resulting “ellipsoid of flow” have worked well in predicting the flow of traditional sublevel caving. Yenge (1980) provides a complete review of the flow theories of loosely packed irregular shaped caved material. The applicability of these models to sublevel caving depends on the assumption that the material was well fragmented and loose in the ring. This was true for the older geometry.

Observations made during the loading of the blasted rock at LKAB suggest a pulsating type of caved waste rock inflow. The working premise is that the pulsation of caved waste rock is the result of the formation and destruction of arches between the intact mining front and the caved rock. These arches control the downward flow of the fragmented material (ore) in the blasted ring. The change in the amount of material capable of swelling in the ring is believed to increase the likelihood of the formation of these arches.

It appears that continuing with the classical bin flow types of laboratory experiments will not lead to the desired pulsation rock flow results and new approaches are needed that handle the effects of blasting. Current approaches make specific observations in the mine and couple this to the development and calibration of numerical models that describe the material flow. With these calibrated models, different design/operational alternatives can be evaluated.

LKAB’s goals for sublevel caving are to; (1) achieve high ore extraction from the slice, (2) minimize the amount of waste rock during ore extraction, and (3) keep the different ore qualities separate. An alternative to keeping the different ore qualities separate, is to know what material is flowing out of the draw point in terms of quality and quantity at any time. Undesired
flow of waste rock into the draw point presents a series of problems that must be dealt with by the mine and processing plant. Waste rock is transported to the surface and is magnetically separated at the plant from the ore. The high quality, low phosphorous ore cannot be separated from the low quality, high phosphorous ore by the current separation process. Therefore, measures to decrease dilution by waste rock at the source are being examined.

2.3 Analytical Tests

There is a limited amount of experimentation that can be done to understand the mechanics of sublevel caving. This includes in-situ marker recovery, scale model tests, numerical models, analytical models, exploratory drilling and drifting, and extraction point observations.

During the summer of 1995, the extraction of two full rings were observed and tested. The tests included video taping the complete extraction process and measuring the ore/waste concentration with two independent methods.

2.3.1 Sieve Analysis

Determining the ore/waste ratio in each scoop is an important issue. Theoretically, it is possible to crush the contents of each scoop of material and magnetically separate the ore from the waste. This would give an exact estimate of the ore and the waste ratio, but is impractical.

In practice, a sieve sample of 3 mm to 10 mm material (weighing 0.5 to 1.0 kg) is collected from the scoop of the LHD. The ore/waste concentration of this sample is then taken to be representative of the ore/waste ratio in the entire scoop. This is the traditional method used by LKAB. Sieving material is also used on the train level to measure the concentrations coming out of the ore passes.

When a sample is taken from the LHD scoop it is screened and only material between 3 and 10 mm is collected. This material is then heated to drive off the moisture and the ore is magnetically separated from the waste. The amount of ore and waste is then weighed. The ore can then be crushed and ground to a powder and put through a x-ray analysis. The x-ray
analysis produces the atomic breakdown of the ore. The components of the ore can include MgO, Al₂O₃, SiO₂, P, K₂O, CaO, TiO₂ and Fe. By knowing the original detailed geology of the in situ rock, and closely watching the concentrations of each of these components, it is possible to use them as “geological markers”. This type of analysis can take a lot of time, in sample collections, sample analysis, and in the interpretation of the data.

2.3.2 Järnhaltsmätare

Another method of determining the ore/waste ratio in the LHD scoop is to take advantage of the magnetic properties of the ore. The “Järnhaltsmätare” or “Iron-Content-Measuring Instrument” measures the capacitive changes in an electric coil brought into close proximity to the muck. If the muck has a high concentration of iron, the current in the coil will significantly increase. If the muck is all waste (no iron content), the current in the coil will not be affected. LKAB has had the järnhaltsmätare for several years, but its use has been limited due to poor correlation with sieve measurements. To get better correlation between the järnhaltsmätare and the sieve analysis the measurements from the järnhaltsmätare must be adjusted for the fact that the järnhaltsmätare is giving a value of iron per unit volume and the lab analysis is weight percent iron. If the value given by the järnhaltsmätare is converted from a % iron by volume to a % iron by mass there is very good correlation with the sieve measurements.

The results of the tests with the järnhaltsmätare show good correlation with the sieve sampling method. This is an indication that the sieve sample size of 3-10 mm is a good representation of the total concentration of ore and waste in the scoop. The järnhaltsmätare does not discriminate on particle size. While using the järnhaltsmätare several observations were made which may be useful for future development.

- The järnhaltsmätare provided instant feedback on the ore/waste content of the muck. This aided in observing when the ore and waste rock flowed.
• The loader operators commented that it would be very helpful to have an instrument mounted in the bucket as an aid to determine when to stop loading.

• The indicator's finest markings are at 5% intervals. This allows for readings with an accuracy of ±2.5%.

• The järnhaltsmätare is too big and clumsy.

• After zeroing, the indicator would read 10% Fe with the spool in the air.

• The measurement taken could vary by ±5% depending on the size distribution of the material (i.e. course material - poor contact, fines - good contact).

• The measurement technique allowed for lots of operator "interpretation".

Based on these observations several recommendations are made.

• The järnhaltsmätare is very old. The possibility of a newer järnhaltsmätare being built or a different instrument being used should be considered. Some other possible instruments that could be used are: (1) beachcomber, (2) airport security hand held metal detector, or (3) an electronic stud finder. If these do not work it should not be very difficult to re-engineer the current järnhaltsmätare. The instrument has four basic electronic parts: (1) charging unit, (2) on/off switch timer circuit, (3) calibration circuit, and (4) measuring circuit. The first two parts should be very basic and easy to separate out leaving only the critical section (the calibration and measuring circuit) which should be relatively straightforward to understand.

• A finer graduated analog or digital readout would provide a more precise measurement reading.

• A smaller hand held unit would be more flexible and possibly be influenced less by material size.
• For standardized use, operator interpretation needs to be removed. Mounting the sensor in the LHD scoop would do this.

• The järnhaltsmätare measures % Fe. It would be more useful if it could measure % malm, i.e. % ore + % waste = 100%, because the loader operators stop loading based on % waste.

• The järnhaltsmätare indicates ≈72% on a piece of ore and ≈30% on a piece of waste. This is a range of 42%. With the analog indicator only readable to ± 2.5% this does not provide very high precision. Increasing the range from 0% for waste to 100% for ore and using either a finer graduated analog or digital display would increase the readable precision.

The järnhaltsmätare is potentially a very useful piece of equipment that, with a minimal amount of development, could be put into widespread use.

2.3.3 Weighing the LHD Scoop

The significant density differences between the ore (4600 kg/m³) and waste (2700 kg/m³) can be used to determine the ore/waste concentration in a LHD scoop. To use this method the volume in each scoop must be known. In practice it should be acceptable to assume that each scoop volume of individual operators are different but on average fairly constant. That is to say, that each operator fills the LHD scoop with the same volume each time. The weight of each scoop is measured electronically. A scoop full of ore will be very heavy and a scoop full of waste will be comparatively light. This method requires little to no interruption in the mining process and can be monitored continuously.

2.3.4 Video Recording

The entire loading process was recorded on videotape during the tests run during the summer of 1995 at LKAB. The video camera was mounted to the drift roof with a quick
release so it could be quickly installed and removed. A cable ran from the camera to a video recorder and monitor that was located on a table by the mouth of the drift. The setup was wired in such a way that the display would only show a picture when everything was working. In future tests it is recommended to place the video recorder and monitor in a closed box to minimize dust exposure to the recorder. The lens of the video camera was cleaned periodically for better clarity. Four 500 Watt Halogen lamps were used to light up the area. Two lamps were mounted on the same structure as the camera, approximately 30 cm below and to the right and left of the camera. Two other lights were mounted on the walls of the drift. Having the lamps too close to the camera appeared to light up dust particles directly in front of the camera and reduced the sharpness of the video image.

The video recorder ran continuously during the test period. Approximately 68 hours of videotape for one ring was produced. The video recorded continuously to avoid the hassle of starting and stopping the recorder between loads and to eliminate the possibility of missing data.

While performing the tests, material could be heard “rolling” down the roof and then seen speeding out of the top of the drift opening. This suggests that there is some free space against the face of the next ring and the loose material. This observation was made after at least 100 scoops and not at the beginning of the draw process. Two methods could be used to estimate the distance of open space against the ring face. (1) If a timer and a distance scale were placed on the video film, (this is possible with most modern cameras) it would be possible to calculate the velocity of the material as it exited into the drift. Assuming the rock had been in a state of free fall, the height of the “open” space in the ring could be estimated with a velocity measurement. (2) Use a police type radar gun to measure the rock speed.

The final value of the video taping is that it is a good record of what actually happened. How often did the loader load from the right and the left of the muck pile? When did large boulders come in? The videotape can be frequently referenced and these questions can be easily answered.
2.3.5 Marker Recovery

Using markers is currently the only method of measuring the flow of the caved ore in situ. It involves placing markers in the ore prior to blasting and locating them as they are extracted. LKAB is using 1m long pieces of electrical cable (about 2” in diameter) as markers. The markers do not have the same density as the broken ore but are somewhat lighter. The markers must be designed to anchor well in the material. Marker recovery provides the researcher with the initial location of the marker and when and where it is extracted (if it is found). Markers currently offer no information on the path traveled throughout the fragmented rock.

2.3.6 Scale Model Tests

Scale model tests have provided most of the information on the mechanics of sublevel caving at present. Originally derived from models measuring material flow in silos, scale models have provided the current understanding of material flow in sublevel caving. Scale models have not been able to incorporate the stresses/material compaction resulting from blasting. Without overcoming these limitations, scale model test may not provide any further insight. These scale model models continue to be used particularly by the University of Luleå, the Mining Institute in Novosibirsk, and INCO in a continuing effort to try and better understand the sublevel caving process.

A large number of laboratory studies have been made by both the University of Luleå and the Mining Institute in Novosibirsk to study sublevel caving. The classic type of experiments use granular materials poured into a bin type of model box. The broken ore is represented by one type of material and the caved waste by another. The development of the draw bodies is then studied under a variety of conditions. These laboratory studies have not reproduced the pulsation currently found in the mine.
Chapter 3

THE DISCRETE ELEMENT METHOD

The discrete element method (DEM) is a numerical technique that solves engineering problems modeled as a large system of distinct interacting general shaped (deformable or rigid) bodies or particles that are subject to gross motion. Engineering problems that exhibit such large scale discontinuous dynamic or static behavior cannot be solved with a conventional continuum based procedure such as the Finite Element Method. The discrete element procedure is used to determine the dynamic contact topology of the bodies. It can account for the complex non-linear interaction phenomena between bodies and numerically solves the equations of motion. Fundamentally, the discrete element method is simple to understand. However, implementing the discrete element method correctly in an elegant and efficient manner is more difficult.

The solution steps for problems solved using the DEM are:

1. Preprocessor – Define the boundaries, initial discrete element locations, and material properties for the system.
2. Solver – Integrate the equations of motion and contact laws for all the discrete elements in the system for the duration of the simulation time.
3. Postprocessor – Interpret the results by generating graphs and visualizations of the material flow.

Ideally, these three steps should be integrated in a totally interactive environment. Unfortunately, step 2 – the solver phase of the discrete element method – often takes hours to weeks of computer time. This interactive environment, therefore, is still a few years off for most problems.
The DEM explicitly models the dynamic motion and mechanical interactions of each body or particle in the physical problem throughout a simulation. The model provides a detailed description of the positions, velocities, and forces acting on each body or particle at discrete points in time. The spacing between each of the discrete points in time is called a time step. Step 2 – “Integrate the equations of motion and contact laws for all the discrete elements in the system” – is the heart of the discrete element method. The computational tasks involved in step 2 are essentially an endless loop as shown in Figure 13. This loop updates the physical parameters within the particle system for on time step.

![Figure 13 Discrete Element Computation Cycle.](image)

Recent research on the discrete element method generally focuses in the following categories:

- Contact Searching
- Contact Determination
- Contact Mechanics
- Time Integration
• Computer Science - Programming Constructs

• Scientific Visualization

• Applications

A pictorial representation of the different research areas is given in Figure 14. The research included in this thesis is represented by the shaded boxes and the “crystal ball” in the center offers suggestions on what the future may behold.
Figure 14 Development Areas of the Discrete Element Method.
Each problem solved with the discrete element method requires different combinations of the available technology. Two examples that illustrate this statement are: (1) A compaction problem used to predict the macroscopic constitutive equations for a granular material will have a minor amount of motion, a size distribution of irregular shaped objects, require accurate contact mechanics and will be run for a relatively short amount of time. (2) A simulation of the material flow at a transfer station will have massive amounts of motion and will use a relaxed contact stiffness and simpler discrete elements to optimize computational performance.

This wide variation in problem types makes it difficult and computationally inefficient to develop a general purpose discrete element program. Instead, the different discrete element technologies should be combined in different ways to best solve a particular problem or class of problems.

The DEM technologies developed in this thesis are selected for the large scale sublevel caving simulation investigated for LKAB in Chapter 4. In the large scale sublevel caving models, the driving characteristics are the size of the model and the physical time of the simulations. To accommodate these conditions; three dimensions are used to capture the true flow mechanics, the discrete elements are spheres and planar boundaries whose edges form convex polygons, a limited particle size distribution is used, and a very efficient contact searching algorithm is developed. Object oriented and parallel algorithms are also developed to improve the maintainability and speed of the implementation. These developments are discussed in full in this chapter.

3.1 Determination and Searching for Contacts

Contact searching and determination are important parts of the discrete element method. This section presents how to determine if two spheres are in contact and if a sphere and a “convex planar boundary” in three dimensions are in contact. A “convex planar boundary” is defined as a boundary whose vertices lie in a single plane and whose edges form
convex polygons. An improved grid contact searching algorithm is also presented that avoids the inefficiencies of traditional grid method implementations.

3.1.1 Determining if Two Spheres are in Contact

The algorithm for determining if two spheres are in contact is simple. Two spheres are considered to be in contact if the distance between their centers is less than the sum of their radii. Two spheres in contact are illustrated in Figure 15.

The description of any geometrical point in the following sections of this thesis will be described with a shorthand notation. This notation defines the geometrical point by simply writing it’s position vector at the location of the geometrical point.

Mathematically the contact check is represented as:

$$\delta = R_1 + R_2 - |\bar{u}_2 - \bar{u}_1|$$

(3.1)

where $\delta < 0$ indicates the particles are not in contact and $\delta \geq 0$ indicates the particles are in contact. $\bar{u}_1$, $\bar{u}_2$ are the position vectors with respect to global axes of the centroids of spheres 1 and 2 respectively where:

$$\bar{u}_i = u_{i,x}\hat{i} + u_{i,y}\hat{j} + u_{i,z}\hat{k}$$
\[ \vec{u}_2 = u_{2,x} \hat{i} + u_{2,y} \hat{j} + u_{2,z} \hat{k} \]

The normal direction at the point of contact is:

\[ \hat{n}_c = \frac{\vec{u}_2 - \vec{u}_1}{|\vec{u}_2 - \vec{u}_1|} \]  \hspace{1cm} (3.2)

and the point of contact is located at:

\[ \vec{P}_c = \vec{u}_1 + \left( R_1 - \frac{\delta}{2} \right) \hat{n}_c \]  \hspace{1cm} (3.3)

or

\[ \vec{P}_c = \vec{u}_2 - \left( R_2 - \frac{\delta}{2} \right) \hat{n}_c \]  \hspace{1cm} (3.4)

The normal direction and the point of contact are illustrated in Figure 15.

The amount of overlap or penetration ($\delta$), the unit normal direction vector ($\hat{n}_c$), and the point of contact ($\vec{P}_c$), are used in the contact mechanics calculations.

During a simulation the contact check between two spheres can be performed several million times, so it is important that it is programmed as computationally efficient as possible. The expansion of equation [3.1] that will be executed by the computer is shown in equation [3.5].

\[ \delta = R_1 + R_2 - \sqrt{(u_{2,x} - u_{1,x})^2 + (u_{2,y} - u_{1,y})^2 + (u_{2,z} - u_{1,z})^2} \]  \hspace{1cm} (3.5)

The most “expensive” portion of this calculation is the square root. To reduce the expense of the square root, at the price of some accuracy and extra storage space, a fast square root function can be used. (Lalonde and Dawson 1990; Hill 1994) The fast square root function replaces the square root function with a lookup table containing pre-calculated square root values. The size of the lookup table can be reduced because the range of square roots that will need to be looked up can be estimated based on the sizes of the discrete elements in the simulation and the contact searching algorithm used.
Another option to reducing the expense of this calculation is to minimize the number of square root operations performed. One method is to compare the square of the sum of the spheres radii with the square of the distance between the sphere centers as shown in equation [3.6].

\[
dist^2 = |\vec{u}_2 - \vec{u}_1|^2 = (u_{2,x} - u_{1,x})^2 + (u_{2,y} - u_{1,y})^2 + (u_{2,z} - u_{1,z})^2
\]

where:

\[
dist^2 > (R_1 + R_2)^2 \quad \text{particles not in contact}
\]

\[
dist^2 \leq (R_1 + R_2)^2 \quad \text{particles in contact}
\]

This extra step adds an additional square function to all contact checks and eliminates the square root function for spheres that are not in contact.

A second method of reducing the contact checking expense is to do a coarser contact check prior to the square root operation. Checking the contact between two cubes that surround the spheres can quickly eliminate spheres that are not in contact. This operation is depicted in two dimensions in Figure 16.

![Diagram of spheres and cubes](image-url)

**Figure 16 Contact Elimination Using Cubes.**

Spheres that are "not in contact" will be eliminated prior to the square root operation. This procedure is computationally expressed in equation [3.7].
If \[ |u_{2x} - u_{1x}| \leq (R_1 + R_2) \text{AND} |u_{2y} - u_{1y}| \leq (R_1 + R_2) \text{AND} |u_{2z} - u_{1z}| \leq (R_1 + R_2) \]\n
Possibly in contact

Else

Not in contact \hfill (3.7)

This method introduces some additional temporary local storage requirements and up to three additional comparative operations per contact check.

Which of these operations should be used to reduce the computational expense of the contact check between spheres depends on several factors. These factors include the type of problem being modeled – specifically the amount of motion occurring and the size distribution of the particles, the contact searching algorithm used, and differences in compilers and CPUs.

3.1.2 Determining if a Sphere and a Boundary are in Contact

An algorithm for determining if a sphere is in contact with a planar convex boundary is presented in this section. If the sphere is in contact with the boundary, the point of contact \( \vec{P}_c \) on the boundary, the amount of penetration \( \delta \), and the unit normal direction vector \( \vec{n}_c \) at the point of contact are calculated for use in the contact mechanics calculations.

A sphere is defined as being in contact with a boundary if the distance from the center of the sphere to the closest point on the boundary is less than the radius of the sphere. When in contact the boundary and the sphere penetrate each other. A sphere in contact with a boundary is depicted in two dimensions in Figure 17.
Figure 17 Contact Between a Sphere and a Boundary.

The point of contact is defined as the point on the boundary closest to the center of the sphere. The amount of penetration is defined as the radius of the sphere minus the distance from the center of the sphere to the point of contact. Thus, the amount of penetration is positive if the sphere and the boundary are in contact. The unit normal at the point of contact is a unit vector in the direction from the point of contact on the boundary towards the center of the sphere.

Determining the point of contact on the boundary takes several steps. The point of contact can lie in one of three regions: (1) inside the boundary, (2) along an edge, or (3) at a corner of the boundary. These regions are depicted in Figure 18.
Figure 18 Contact Regions.
To make contact determination easier the boundaries are divided into triangles. A boundary with N vertices is divided into N-2 triangles as depicted in Figure 19. The corner points of the boundary are defined as $\bar{u}_i$ with $i = 0, 1, \ldots, N-1$.

![Diagram of boundary divided into triangles](image)

Figure 19 Convex Boundary Divided into Triangles.

Each triangle is treated as an individual boundary with vertices numbered 0 to 2. The direction of the numbering of the vertices, clockwise or counterclockwise is not important. Both sides of the triangle are checked for contact.

The problem of determining contact between a sphere and a boundary is reduced to determining the contact between a sphere and a triangle. To simplify calculations a local coordinate system is defined at the first vertex in the plane of the triangle. The axes of the local orthogonal coordinate system are defined with three beta ($\hat{\beta}_k, k = 1, 2, 3$) unit vectors as shown in Figure 20, and the origin of the local coordinate system is at the point defined by the vector $\bar{u}_0$. For the sake of brevity, the superscripts are dropped from the triangle vertices $(\bar{u}_0, \bar{u}_1, \bar{u}_2 = \bar{u}_0, \bar{u}_1, \bar{u}_2)$.
The unit vectors $\mathbf{\hat{\beta}_k}$, $k=0,1,2$ have components in the global system that are defined by

$$\mathbf{\hat{\beta}_k} = \beta_{k,x}\mathbf{i} + \beta_{k,y}\mathbf{j} + \beta_{k,z}\mathbf{k}$$

The vector, $\mathbf{\hat{\beta}_2}$, is normal to the plane of the triangle. The beta unit vectors are calculated as:

$$\mathbf{\hat{\beta}_0} = \frac{\mathbf{\hat{u}_1} - \mathbf{\hat{u}_0}}{|\mathbf{\hat{u}_1} - \mathbf{\hat{u}_0}|} \quad (3.8)$$

$$\mathbf{\hat{\beta}_2} = \frac{\mathbf{\hat{\beta}_0} \times (\mathbf{\hat{u}_2} - \mathbf{\hat{u}_0})}{|\mathbf{\hat{\beta}_0} \times (\mathbf{\hat{u}_2} - \mathbf{\hat{u}_0})|} \quad (3.9)$$

$$\mathbf{\hat{\beta}_1} = \mathbf{\hat{\beta}_2} \times \mathbf{\hat{\beta}_0} \quad (3.10)$$

Vectors or points in the local coordinate systems are designated with a single quote (i.e. Point $\mathbf{P}$ in the global coordinate systems is $\mathbf{P}'$ in the local coordinate system). A point $\mathbf{P}$ in the global coordinate system is translated and rotated to the local coordinate system with the following equation:
\[ \bar{P}' = \begin{bmatrix} P'_x \\ P'_y \\ P'_z \end{bmatrix} = \begin{bmatrix} \beta_{0,x} & \beta_{0,y} & \beta_{0,z} \\ \beta_{1,x} & \beta_{1,y} & \beta_{1,z} \\ \beta_{2,x} & \beta_{2,y} & \beta_{2,z} \end{bmatrix} \begin{bmatrix} P_x - u_{0,x} \\ P_y - u_{0,y} \\ P_z - u_{0,z} \end{bmatrix} \] (3.11)

The distance between the center of the sphere, \( \bar{P} \), and the plane of the boundary is calculated as:

\[ \text{dist} = |P'_z| \] (3.12)

The absolute value accounts for the sphere being on either side of the boundary. If the distance is greater than the radius of the sphere, then the boundary and the sphere are not in contact, and no further contact checks are needed. If the distance is less than the radius of the sphere, then the boundary and the sphere may be in contact and the point of contact must be found.

To determine if the point of contact lies within the boundary, along an edge, or at a corner of the boundary an area coordinate system is introduced. See Zienkiewicz and Taylor (1989) for a complete definition of an area coordinate system. The area coordinate system is shown in Figure 21.

![Area Coordinate System](image)

Figure 21 Area Coordinate System.

The point, \( (P'_x, P'_y) \), is the projection of the sphere’s center on the plane of the triangle. The area coordinates \( L_0, L_1, \) and \( L_2 \) represent the areas of each associated triangle.
shown in Figure 21. The area coordinates of the sphere center projected on the plane of the boundary are calculated as:

\[ L_i = \frac{a_i + b_i P'_x + c_i P'_y}{2A} \]  \hspace{1cm} (3.13)

where:

\[ i = 0 \ldots 2 \]

\[ A = \text{area of triangle (012)} \]

and the constants \( a_i \), \( b_i \), and \( c_i \) are calculated as:

\[ a_0 = u'_1 x u'_2 y - u'_2 x u'_1 y \]  \hspace{1cm} (3.14)

\[ b_0 = u'_1 y - u'_2 y \]  \hspace{1cm} (3.15)

\[ c_0 = u'_2 x - u'_1 x \]  \hspace{1cm} (3.16)

with cyclic rotation on the indices 0, 1, and 2. The constants \( a_i \), \( b_i \), and \( c_i \) only need to be calculated once and can be stored with the rest of the boundary data.

Initially it was thought that binary combinations of area coordinate signs would indicate the side or corner of the contact point. This method is depicted in Figure 22 and summarized in Table 5.

![Contact Region Diagram](image)

\[ \text{Contact Region} \]

Figure 22 Contact Location Using Area Coordinates.
Table 5 Contact Location Using Area Coordinates.

<table>
<thead>
<tr>
<th>$L_0$</th>
<th>$L_1$</th>
<th>$L_2$</th>
<th>Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\geq 0$</td>
<td>$\geq 0$</td>
<td>$\geq 0$</td>
<td>Within Boundary</td>
</tr>
<tr>
<td>$\geq 0$</td>
<td>$\geq 0$</td>
<td>$&lt; 0$</td>
<td>On edge ①</td>
</tr>
<tr>
<td>$\geq 0$</td>
<td>$&lt; 0$</td>
<td>$\geq 0$</td>
<td>On edge ②</td>
</tr>
<tr>
<td>$\geq 0$</td>
<td>$&lt; 0$</td>
<td>$&lt; 0$</td>
<td>On corner ③</td>
</tr>
<tr>
<td>$&lt; 0$</td>
<td>$\geq 0$</td>
<td>$\geq 0$</td>
<td>On edge ④</td>
</tr>
<tr>
<td>$&lt; 0$</td>
<td>$\geq 0$</td>
<td>$&lt; 0$</td>
<td>On corner ⑤</td>
</tr>
<tr>
<td>$&lt; 0$</td>
<td>$&lt; 0$</td>
<td>$\geq 0$</td>
<td>On corner ⑥</td>
</tr>
<tr>
<td>$&lt; 0$</td>
<td>$&lt; 0$</td>
<td>$&lt; 0$</td>
<td>Never Possible</td>
</tr>
</tbody>
</table>

Unfortunately, this method does not completely define the region of contact. The shortcoming is in the fact that the areas of corner contact are defined by regions drawn perpendicular to the boundaries. This difference is depicted in Figure 23.

![Diagram](image)

○ Contact Region

△ Correct Corner Contact Regions

Figure 23 Correct Regions for Contact Location.

The signs of the area coordinates, however, can still provide information helpful in locating the contact region. The correct possible contact regions based on the signs of the area coordinates are listed in Table 6.
Table 6 Correct Regions for Contact Location.

<table>
<thead>
<tr>
<th>$L_0$</th>
<th>$L_1$</th>
<th>$L_2$</th>
<th>Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\geq 0$</td>
<td>$\geq 0$</td>
<td>$\geq 0$</td>
<td>Within Boundary</td>
</tr>
<tr>
<td>$\geq 0$</td>
<td>$\geq 0$</td>
<td>$&lt; 0$</td>
<td>On edge 1, corner 3, or corner 5</td>
</tr>
<tr>
<td>$\geq 0$</td>
<td>$&lt; 0$</td>
<td>$\geq 0$</td>
<td>On edge 2, corner 3, or corner 6</td>
</tr>
<tr>
<td>$\geq 0$</td>
<td>$&lt; 0$</td>
<td>$&lt; 0$</td>
<td>On corner 3, edge 1, or edge 2</td>
</tr>
<tr>
<td>$&lt; 0$</td>
<td>$\geq 0$</td>
<td>$\geq 0$</td>
<td>On edge 4, corner 5, or corner 6</td>
</tr>
<tr>
<td>$&lt; 0$</td>
<td>$\geq 0$</td>
<td>$&lt; 0$</td>
<td>On corner 5, edge 1, or edge 4</td>
</tr>
<tr>
<td>$&lt; 0$</td>
<td>$&lt; 0$</td>
<td>$\geq 0$</td>
<td>On corner 6, edge 2, or edge 4</td>
</tr>
<tr>
<td>$&lt; 0$</td>
<td>$&lt; 0$</td>
<td>$&lt; 0$</td>
<td>Never Possible</td>
</tr>
</tbody>
</table>

If $L_0$, $L_1$, and $L_2$ are all greater than or equal to zero the sphere and the boundary are in contact, the point of contact is within the boundary, and the location is found with equation [3.17].

$$\vec{P}_c = L_0\vec{u}_0 + L_1\vec{u}_1 + L_2\vec{u}_2$$ (3.17)

The point on contact is calculated in global coordinates. The normal at the point of contact is the same as the normal for the plane of the boundary and is calculated as shown in equation [3.18].

$$\vec{n}_c = \text{sign}(P'_z)\vec{\beta}_2$$ (3.18)

where

$$\text{sign}(x) = \begin{cases} 1 & x > 0 \\ 0 & if \ x = 0 \\ -1 & x < 0 \end{cases}$$

The sign of $P'_z$ accounts for which side of the plane the sphere is in contact with.

If only one of the area coordinates is negative, the sphere may be in contact with an edge or one of the adjacent corners. The point of contact with position vector $\vec{P}_c$ along an edge is depicted in Figure 24.
The distance $b$ is calculated in equation (3.19).

$$ b = \left( \vec{P}_s - \vec{u}_0 \right) \cdot \frac{\vec{u}_i - \vec{u}_0}{|\vec{u}_i - \vec{u}_0|} \quad (3.19) $$

New area coordinates of the contact point, $\vec{P}_c$, are calculated by dividing $b$ by the length of the edge. For the edge between vertices $\vec{u}_0$ and $\vec{u}_i$ the new area coordinates are calculated as:

$$ L[1] = \frac{b}{|\vec{u}_i - \vec{u}_0|} \quad (3.20) $$

$$ L[0] = 1.0 - L[1] \quad (3.21) $$

$$ L[2] = 0.0 \quad (3.22) $$

To fix the point of contact at the corners $L[1]$ is set to 0.0 if it is less than 0.0 or 1.0 if $L[1]$ is greater than 1.0. Forcing the value of $L[1]$ to this range handles the corners. With the corrected area coordinates, the point of contact is calculated with equation (3.17).

If two of the area coordinates are negative, the sphere may be in contact with a corner or one of the two connecting edges. The point of contact is depicted in Figure 25.
The distances along both edges are calculated using equations [3.23] and [3.24].

\[ b_1 = \left( \frac{\vec{p}_s - \vec{u}_0}{|\vec{u}_1 - \vec{u}_0|} \right) \cdot \frac{\vec{u}_1 - \vec{u}_0}{|\vec{u}_1 - \vec{u}_0|} \quad (3.23) \]

\[ b_2 = \left( \frac{\vec{p}_s - \vec{u}_0}{|\vec{u}_2 - \vec{u}_0|} \right) \cdot \frac{\vec{u}_2 - \vec{u}_0}{|\vec{u}_2 - \vec{u}_0|} \quad (3.24) \]

If both \( b_1 \) and \( b_2 \) are negative, the point of contact is the corner and the area coordinates are calculated as:

\[ L[0] = 1.0 \quad (3.25) \]

\[ L[1] = 0.0 \quad (3.26) \]

\[ L[2] = 0.0 \quad (3.27) \]

If \( b_1 \) or \( b_2 \) is positive, the point of contact is on the edge associated with the positive b value. The area coordinates are calculated following equations [3.20 - 3.22]. Both \( b_1 \) and \( b_2 \) should never both be positive. With the corrected area coordinates, the point of contact can be calculated with equation [3.17].

Once the point of contact is found it can then be determined if the sphere and boundary are actually in contact. The amount of penetration is calculated with equation [3.28].
\[ \delta = R - |\vec{P}_s - \vec{P}_c| \]  

(3.28)

where \( \delta < 0 \) = not in contact

\( \delta \geq 0 \) = in contact

If the sphere and boundary are in contact, the normal at the point of contact is calculated with equation [3.29].

\[ \hat{n}_c = \frac{\vec{P}_s - \vec{P}_c}{|\vec{P}_s - \vec{P}_c|} \]  

(3.29)

An additional benefit to using area coordinates to determine contact between a sphere and a boundary is the ability to assign additional properties for the boundaries. One example is to assign a temperature to each of the vertices, \( T_0, T_1, \) and \( T_2. \) The temperature at the point of contact, \( T_c, \) is then calculated as:

\[ T_c = L_0 T_0 + L_1 T_1 + L_2 T_2 \]  

(3.30)

3.1.3 Locating Possible Contacts

One of the most important aspects of a discrete element implementation is the algorithm that locates possible contacts. The contact searching algorithm takes all of the discrete elements in the system and determines which elements (spheres) are likely to be in contact. The algorithm should be conservative; meaning that elements that are in contact must never be missed.

The steamroller approach of finding contacts, involving checking every discrete element against every other discrete element, is a very expensive O(N^2) operation. Fortunately, algorithms that are more intelligent are available to reduce this computational burden. The choice of the algorithm is influenced by the following factors:

- The type of discrete elements in the model.
- The amount of motion in the system.
• Size distribution of the discrete elements.

• Additional memory required by the searching algorithm.

• The portability to parallel or distributed computer architectures.

• Complexity of implementing the algorithm.

A simulation with very complicated discrete elements, such as superquadrics or polyhedrals, should use an algorithm that only locates elements having a very high probability of being in contact. Some computational expense for determining this high probability can be tolerated for complicated elements. For simpler discrete elements, such as spheres or ellipsoids, a faster algorithm that returns a higher percentage of elements not in contact is used. Systems that experience minimal amounts of motion can take advantage of algorithms that do not need to update the contact topology in the entire system every time step. Systems with a significant amount of motion should use an algorithm that updates the possible contacts very quickly. In simulations where there is a large size distribution in the discrete elements, algorithms that are not penalized by the size distribution should be used.

Some contact searching algorithms require additional memory to help sort the elements. With the decreasing cost of memory, memory limitations are becoming less of an issue. Certain contact searching algorithms are more adaptable to parallel implementation and parallel and distributed computer architectures are becoming available to the general public. The complexity of implementing different algorithms in computer code is also an issue.

In the large scale sublevel caving model investigated in this thesis an enhanced derivative of the traditional grid method is implemented. This contact searching algorithm is selected because (1) the size of the spherical discrete elements are nearly uniform, (2) the elements – modeled as spheres are very simple, (3) there are periods of time when significant motion of the particles occurs, and (4) the algorithm is implemented on a parallel machine in two dimensions. This section presents the traditional grid method, the advancements to the
grid method used in this thesis, a theoretical calculation of the performance improvement, and an explanation on how to handle boundaries with the grid method.

3.1.3.1 Traditional Grid Method

The grid method was introduced by Cundall (1974) and continues to be used by researchers today (Mathews 1994; Zhang 1996; Greening 1996). The grid method is comparable to a bucket sort and has an $O(N)$ computational cost.

As the name implies the problem domain is divided into cells, forming a grid. The discrete elements are mapped, or sorted, into one or several cells. Contact checking is reduced to checking for contact with other discrete elements located in the current or adjacent cells. The performance of the algorithm is influenced by the size of the grid cells and how the discrete elements are mapped or sorted into these cells. This section discusses the traditional grid method, highlighting several inefficiencies. Section 3.1.3.2 presents an improved grid method that overcomes these inefficiencies. The improved grid method is used in the simulations presented in this thesis.

The problem domain is divided into equal size square or cubic cells. The extents of the problem domain can be adjusted as a simulation proceeds based on the locations of the bounding discrete elements or it can be fixed at the start of the simulation. Adjusting the grid as the simulation runs requires dynamic allocation and de-allocation of memory. To avoid this extra memory management the problem domain is fixed at the beginning of the simulation in this thesis. If and when discrete elements move outside the problem domain they are deleted from the simulation.

The choice of the grid cell size affects the performance of the grid method. If large cells are used there will be many discrete elements in each cell and in the limit the search approaches the steamroller approach of $O(N^2)$. If small cells are used there will be more cells in the system that must be searched for contacts, hence, processing time increases. With smaller cells, each discrete element may be mapped to several cells and having more cells also increases the memory requirements.
In Cundall's early work, he reports using a grid with 208 cells (16x13) to cover the problem domain (Cundall 1974). This number was reached by "divine inspiration or something similar" but is more likely the result of memory limitations, octal numbers, and the screen resolution of the computer. Cundall conceded that there might be a more optimal grid size. In the BALL discrete element program a grid size of at least the size of the largest particle is used to ensure a particle would only map into at most four cells (Cundall 1978).

Taylor and Preece (1989a), used a grid size based on the largest particle diameter in the system. No explanation is given if this is an optimal size and appears to be chosen so that each particle can be located in, at most, four grid cells. Greening (1996) follows this method in the three-dimensional work done in his Ph.D. thesis.

Zhang's (1993) suggests that the optimum grid size should be arrived at by running a number of simulations with different grid sizes and finding the minimum CPU time. With simulations taking over 100 hours of dedicated CPU time the test process was impractical and a cell size of 2 times the largest particle diameter was arrived at by "logical analysis". The grid method Zhang uses requires that the cell size be at least as large as the largest particle diameter. This methodology and cell size is also used in Zhang's Ph.D. thesis work (Zhang 1996).

In Mathews' (1994) work, a continuation of Zhang's (1993) research, he uses a grid cell size of 2.5 times the smallest particle radius. This size constraint is chosen because it limits the maximum number of particles to occupy any given cell at any given time to nine. Mathews' work overcame the minimum cell size requirement in Zhang's (1993, 1996) algorithm.

No conclusions can be drawn on the optimum cell size for these grid methods. In this thesis a grid size equal to the maximum particle diameter is used. This cell size is chosen to simplify the algorithms and is the minimum required cell size in the improved grid technique presented in Section 3.1.3.2.

The simulation extents are defined with minimum (xmin, ymin, zmin) and maximum (xmax, ymax, zmax) coordinates. This volume is divided into geometric cubic cells (boxels) which have a width of Δ. Figure 26 shows the geometric grid in two dimensions.
The number of grid cells in the x, y, and z directions are calculated so that the grid totally encompasses the simulation extents. The number of grid cells in the x direction, denoted by GNX, is calculated as

$$GNX = \left\lceil \frac{x_{\text{max}} - x_{\text{min}}}{\Delta} \right\rceil$$

(3.31)

where

$[x]$ denotes the smallest integer greater than or equal to x.

The number of grid cells in the y and z directions are calculated in a similar manner.

Each discrete element in the system is mapped into the grid structure in the following manner:

1. A bounding box is defined for each discrete element.
2. The grid cells that the bounding box overlaps are assigned a pointer to the discrete element.

If the cell width is selected to be greater than the largest particle diameter, each discrete element will map into a maximum of four grid cells. If a smaller cell size is used each element can map into several cells. In the computer implementation, each cell in the grid data structure that contains a discrete element receives a pointer to that discrete element.
For each time step, after the discrete element velocities and positions have been updated, their locations in the grid structure need to be checked and corrected if necessary. This update can be avoided every time step by adding a buffer zone around each particle in the sorting process. The discrete elements in the data structure then only need to be updated when one particle in the system has moved the buffer distance. This method is introduced by Cundall (1978), presented in detail by Mathews (1994) and duplicated in Zhang (1996). Mathews uses a buffer region of 20% of the largest particle radius while Zhang uses a buffer region of 20% of the largest particle diameter.

Figure 27 shows an example configuration of discrete elements. All of the discrete elements are the same size and a buffer region of 20% of the particle radius is shown.

Figure 27 Example Particle Configuration.

Table 7 shows how this configuration of discrete elements maps into the grid data structure.
Table 7 Particle Locations in the Grid Data Structure.

<table>
<thead>
<tr>
<th>Cell</th>
<th>Discrete Elements</th>
<th>Cell</th>
<th>Discrete Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C, E</td>
<td>11</td>
<td>D</td>
</tr>
<tr>
<td>2</td>
<td>C, E</td>
<td>12</td>
<td>B, D</td>
</tr>
<tr>
<td>3</td>
<td>None.</td>
<td>13</td>
<td>B</td>
</tr>
<tr>
<td>4</td>
<td>None.</td>
<td>14</td>
<td>A, F</td>
</tr>
<tr>
<td>5</td>
<td>None.</td>
<td>15</td>
<td>A, F</td>
</tr>
<tr>
<td>6</td>
<td>C, E</td>
<td>16</td>
<td>D</td>
</tr>
<tr>
<td>7</td>
<td>B, C, E</td>
<td>17</td>
<td>D</td>
</tr>
<tr>
<td>8</td>
<td>B</td>
<td>18</td>
<td>None.</td>
</tr>
<tr>
<td>9</td>
<td>F</td>
<td>19</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>F</td>
<td>20</td>
<td>A</td>
</tr>
</tbody>
</table>

After all the discrete elements are sorted into the grid structure the contact checking begins. Every cell in the data structure is checked. If the cell contains two or more discrete elements, all of the elements in the cell are checked for contact with one another. This is an $O(n^2)$ operation where $n$ is the number of discrete elements in the cell.

It is possible and likely that two discrete elements will be mapped into more than one cell for contact checking. To avoid adding the contact forces to the discrete elements several times when this duplication occurs, the point of contact must be found. If the point of contact is not in the current cell being checked, the contact forces are not applied to the elements. A summary of the contact checking operations for the configuration of particles in Figure 27 is given in Table 8.
Table 8 Particle Configuration Contact Checking Operations.

<table>
<thead>
<tr>
<th>Cell</th>
<th>Check Contact</th>
<th>In Contact</th>
<th>Contact Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C-E</td>
<td>Yes</td>
<td>In Cell</td>
</tr>
<tr>
<td>2</td>
<td>C-E</td>
<td>Yes</td>
<td>Not In Cell</td>
</tr>
<tr>
<td>3</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>C-E</td>
<td>Yes</td>
<td>Not In Cell</td>
</tr>
<tr>
<td>7</td>
<td>B-C</td>
<td>Yes</td>
<td>In Cell</td>
</tr>
<tr>
<td>7</td>
<td>B-E</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>C-E</td>
<td>Yes</td>
<td>Not In Cell</td>
</tr>
<tr>
<td>8</td>
<td>Single</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Single</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Single</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Single</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>B-D</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Single</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>A-F</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>A-F</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Single</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Single</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Single</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Single</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Some disadvantages of this form of the grid method are:

1. All of the cells in the grid structure must be checked for particles in contact. This can waste a significant amount of processor time for sparsely populated systems and becomes intolerable in three dimensions.

2. The contact between discrete elements is checked multiple times each time step. Only the contact forces with the point of contact in the current cell are added to the elements.

3. The discrete elements are sorted into several cells requiring more memory.

4. Sorting the discrete elements into several cells creates more data that must be shared between processors in a parallel implementation.
Mathews (1994) introduces an improvement to the grid method in two dimensions that helps reduce the computational burden of searching all the grid cells for particles in contact. The improvement is very problem dependent and does not scale well to three dimensions. In section 3.1.3.2 an improved grid method is derived that avoids these disadvantages.

3.1.3.2 Improved Grid Method

The improved grid contact searching method is designed to avoid the inefficiencies of the traditional grid method. By making a few changes to the grid method it is made significantly faster, easier to implement and maintain, and requires less communication between processors in a parallel implementation. This section presents the improved grid method and gives an example of a contact checking problem for a sample configuration of discrete elements.

The fundamental change from the traditional grid method is from tagging all the grid cells a discrete element overlaps to only tagging the one cell where the center of the discrete element is located. This change is illustrated in Figure 28.

![Diagram of Traditional and Improved Grid Methods]

Figure 28 Tagged Grid Cells.
Each discrete element in the simulation is located in only one cell. The cell that contains the
discrete element is determined by the center location of the discrete element. The grid indices
are denoted as i, j, and k in the x, y, and z directions respectively. The i index is calculated as:

\[ i = \left\lfloor \frac{x_c - x_{\text{min}}}{\text{SphereDiameter}_{\text{max}}} \right\rfloor \]  

(3.32)

where

\[ \lfloor x \rfloor \] denotes the largest integer less than or equal to x.

where \( x_c \) is the x coordinate of the center of the discrete element being considered. The j and
k indices are calculated in a similar manner. Each cell contains a linked list of the discrete
elements that are mapped into that cell.

Having each discrete element mapped to only one cell decreases the amount of
memory required. In the traditional grid method, each discrete element could map to six or
more cells depending on the cell size.

The real advantage of each discrete element mapping to only one cell is in the contact
checking. Figure 29 shows a “Current Cell” that contains a discrete element being interrogated
for contact.

![Figure 29 Cells Checked for Contact.](image)

If the cell size, \( \Delta \), is set to the largest particle diameter or larger, only the current and adjacent
cells need to be evaluated for discrete elements in contact. In this thesis, the cell size is set to
the largest particle diameter. It is believed that this is the optimum cell size. Also only half of
the adjacent cells are checked for discrete elements in contact as shown in Figure 29. The other
half of the cells will be checked if and when they become the "Current Cell". This contact checking method completely eliminates the multiple checks for contact checks between discrete elements. In the traditional grid method the cell containing the location of the point of contact has to be found to determine which contact check is the correct one to use.

One of the major inefficiencies of the traditional grid method is having to interrogate the entire grid structure looking for possible contacts. This is especially true in three-dimensional and in sparsely populated simulations. Because each particle is mapped to only one cell, a separate data structure containing only the discrete elements can be kept. In this thesis, a linked list data structure is used for flexibility. For improved performance, an array could be used. During the contact checking, the data structure of discrete elements is iterated through. The cell of the current discrete element is found and the discrete element is checked for contact with discrete elements in the current and adjacent cells. The next discrete element in the data structure is then checked in a similar matter and the process repeats. This method completely eliminates the geometrical influence on the performance of the grid method. The size of the simulation extents and the number of grid cell do not impact the performance.

In a parallel implementation that divides the problem domain geometrically among processors the improved grid method requires less data be communicated between processors. This is a direct result of each discrete element mapping onto only one cell. A parallel implementation of the improved grid method is discussed in section 3.4.

To illustrate how the improved grid method works, the contact checks for an example arrangement of discrete elements is traced.
In Figure 30 the numbers inside each sphere represent the sphere’s order in the large linked list. The letters represent the sphere’s order in an individual cell’s linked list. The linked list of discrete elements is looped through. For each discrete element: (1) all of the other discrete elements in the current cell (in the linked list after the current discrete element) are checked for contact and then (2) all of the spheres in half the adjacent cells are checked for contact. The order in which the adjacent cells are checked is based on how the three-dimensional grid is stored in memory to maximize performance. The contact checking between spheres would proceed in the order shown in Table 9.

Table 9 Contact Checking Order.

<table>
<thead>
<tr>
<th>Active Sphere</th>
<th>Check For Contact With Sphere</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-A</td>
<td>None.</td>
</tr>
<tr>
<td>2-A</td>
<td>3-B</td>
</tr>
<tr>
<td>2-A</td>
<td>4-A</td>
</tr>
<tr>
<td>3-B</td>
<td>4-A</td>
</tr>
<tr>
<td>4-A</td>
<td>None.</td>
</tr>
<tr>
<td>5-A</td>
<td>2-A</td>
</tr>
<tr>
<td>5-A</td>
<td>3-B</td>
</tr>
<tr>
<td>6-A</td>
<td>1-A</td>
</tr>
</tbody>
</table>
The number of operations in Table 9 should be compared to Table 8 which lists the contact checking operations for the traditional grid method for the same configuration of discrete elements.

After the discrete elements have been moved each time step, their location in the grid cells must be checked and re-mapped. If a discrete element has moved between cells, the appropriate linked lists in the cells must be updated. The only maintenance done on the linked list holding the elements is adding and deleting discrete elements. There may be some performance benefit resulting from memory access times if the list is ordered occasionally to better reflect the geometric ordering of the discrete elements.

3.1.3.3 Performance Comparison Between the Traditional and New Grid Methods

The improvement in the contact search algorithm is clearly seen by comparing the number of contact checks for several different particle configurations. The amount of improvement is dependent on the particle configuration, however, as the following examples show the improvement is significant in all of the examples considered. Two measurements for each algorithm are made: (1) the total number of contact checks for a system of N particles arranged in a grid format, and (2) the number of contact checks performed on a single particle located in the center of the pattern. These calculations are simple for the improved grid method but for the original grid method they get more difficult as the particle density becomes higher.

The first particle configuration is shown in Figure 31. The number of particles has been selected so the $\sqrt{N}$ is an integer (N=25). Lines drawn connecting the particles represent contact checks that will be performed. The contact checks for the new grid algorithm are shown on the left and on the right for the original algorithm.
The number of contact checks using the improved grid algorithm for \( N \) particles is:

\[
NC = 0
\]  
(3.33)

For a particle located the center of the pattern the number of contact checks performed is also:

\[
NC = 0
\]  
(3.34)

For the traditional algorithm the number of contact checks for \( N \) particles is:

\[
NC = \left[ (\sqrt{N} - 1)(\sqrt{N}) \right] \cdot 2 \cdot 3 + (\sqrt{N} - 1)(\sqrt{N} - 1) \cdot 2 \cdot 1 = 8 \cdot N - 10 \cdot \sqrt{N} + 2
\]  
(3.35)

and for a single particle in the center of the pattern:

\[
NC = 4 \cdot 3 + 4 \cdot 1 = 16
\]  
(3.36)

In the second particle distribution, the possible contact checks that are performed once and three times in the traditional algorithm have been separated out to help clarify the figure. The second distribution is shown in Figure 32.
Figure 32 Second Particle Distribution.
For the improved grid algorithm the number of contact checks for $N$ particles is:

$$NC = \left[ (\sqrt{N} - 1) \cdot 2 + 1 \right] (\sqrt{N} - 1) = 2N - 3\sqrt{N} + 1$$  \hspace{1cm} (3.37)

The number of contact checks for a particle in the center of the configuration is:

$$NC = 4$$  \hspace{1cm} (3.38)

With the traditional grid algorithm the number of contact checks is:

$$NC = \left[ (\sqrt{N} - 1) \cdot 2 + 1 \right] (\sqrt{N} - 1) \cdot 4 + \left[ (\sqrt{N} - 1) \sqrt{N} + (\sqrt{N} - 2) \sqrt{N} \right] \cdot 3$$

$$+ \left( \sqrt{N} - 1 \right) (\sqrt{N} - 2) \cdot 2 \cdot 1 = 16 \cdot N - 25 \cdot \sqrt{N} + 6$$  \hspace{1cm} (3.39)

and for a particle in the center:

$$NC = 4 \cdot 4 + 4 \cdot 3 + 4 \cdot 1 = 32$$  \hspace{1cm} (3.40)

Finally in the third particle distribution, as shown in Figure 33, the contact checks that are performed once and three times are separated out in the traditional algorithm.
Figure 33 Third Particle Distribution.
For the new contact algorithm the number of contact checks is:

\[ NC = (\sqrt{N} - 1)\sqrt{N} \cdot 2 + (\sqrt{N} - 1)(\sqrt{N} - 1) \cdot 2 = 4N - 6\sqrt{N} + 2 \]  \hspace{1cm} (3.41)

and for a particle in the center of the distribution with the new algorithm is:

\[ NC = 8 \]  \hspace{1cm} (3.42)

For the traditional contact algorithm the number of contact checks is:

\[ NC = \left(\sqrt{N} - 1\right)\sqrt{N} \cdot 2 \cdot 6 + (\sqrt{N} - 1)(\sqrt{N} - 1) \cdot 2 \cdot 4 + 2\left(\sqrt{N} - 2\right)(\sqrt{N} - 1) \cdot 4 \]

\[ + 4\left(\frac{\sqrt{N}}{2} - 1\right)\sqrt{N} \cdot 2 \cdot 3 + \left(\frac{\sqrt{N}}{2} - 1\right)(\sqrt{N} - 1) \cdot 2 \right] = 36N - 72\sqrt{N} + 32 \]  \hspace{1cm} (3.43)

and:

\[ NC = 4 \cdot 6 + 4 \cdot 4 + 4 \cdot 3 + 8 \cdot 2 + 4 \cdot 1 = 72 \]  \hspace{1cm} (3.44)

To come up with the equation of the number of contact checks for the traditional grid method a grid that is 6x6 is used. The 5x5 grid illustrated in Figure 33 proved to be too difficult because the \[ \frac{\sqrt{N}}{2} \] was not an integer.

The number of contact checks for these different configurations is summarized in Table 10. The number of contact checks for a single particle in the center of the pattern in three dimensions is included.

Table 10 Number of Contact Checks for Traditional and Improved Grid Methods.

<table>
<thead>
<tr>
<th>Element</th>
<th>Two Dimensions</th>
<th>Three Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All Contacts</td>
<td>One Element</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>Old</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8 \cdot N - 10\sqrt{N} + 2</td>
</tr>
<tr>
<td>2</td>
<td>2 \cdot N - 3\sqrt{N} + 2</td>
<td>16 \cdot N - 25\sqrt{N} + 6</td>
</tr>
<tr>
<td>3</td>
<td>4 \cdot N - 6\sqrt{N} + 2</td>
<td>36 \cdot N - 72\sqrt{N} + 32</td>
</tr>
</tbody>
</table>
The conclusion is that for a large number of elements of similar (or uniform) size the improved grid method requires approximately 9 times fewer contact checks than the traditional grid method in two dimensions and 27 times fewer contact checks in three dimensions as the packing density increases.

3.1.3.4 Handling the Boundaries With the Grid Methods

In the presentation of the grid contact searching algorithm no mention of how to handle the boundaries has been made. One possibility is to set the grid cell size to the extents of the largest boundary. This is computationally very simple since all of the discrete elements (the spheres and the boundaries) can be treated the same by the contact searching algorithm. Unfortunately the efficiency of the algorithm approaches \(O(N^2)\) since there can be many discrete elements in each cell. This condition is shown in Figure 34.

![Figure 34 Cell Size Set to Maximum Boundary Extents.](image)

An alternative is to set the cell size based on the particle sizes and "tag" the cells that the boundary occupies. This is the method traditionally used in discrete element
implementations. With the traditional grid method, cells that the boundary intersects are tagged. See Mathews (1994). During contact searching the tagged boundaries are checked for contact with particles in each cell. One particle could be checked for contact with a boundary multiple times. Similar to the contact check between two spheres, only the forces calculated with the point of contact in the current cell are applied to the elements. The cells that need to be tagged for contact with the traditional grid method for a boundary are shown in Figure 35.

![Tagged Grid Cells](image)

Figure 35 Cells Tagged for Boundary Contact with Traditional Grid Method.

With the improved grid method presented in this thesis the cells that are within one cell width of the boundary are tagged. Contacts between the sphere and the boundary are only checked for contact with boundaries in the cell where the sphere is located. The cells requiring tagging with the improved grid method are shown in Figure 36.
How to select the cells that need to be tagged for a boundary is also important. The “smart” way to do it is analytically based on the equation of the boundary. This method is straightforward in two dimensions, but becomes more difficult in three dimensions. If the boundary is moving through space the cells should be tagged using the “smart” method.

If the boundaries are not moving the cells only need to be tagged once at the beginning of the simulation. Since the cells are only tagged once, the computational expense is not as significant and a “simpler” method of tagging the cells can be used. With this method all the cells that are within the extents of the boundary are evaluated. If any of the corners of the cell are within once cell width of the boundary the cell is tagged. This method is inefficient but is easy to implement computationally.
3.2 Contact Mechanics

The contact mechanics between interacting bodies are modeled with a contact force law which has components defined in the normal and shear contact directions. The normal contact force component is generated with a linear elastic restoring component and a viscous damping term to simulate the energy loss in a normal collision. The linear elastic component is modeled with a spring whose coefficient is based upon the maximum allowable penetration of one element into another and the normal viscous damper coefficient is defined in terms of an equivalent coefficient of restitution. The shear contact force component is governed with a Coulomb friction model defined with a friction coefficient. The static and dynamic friction coefficients are set equal (i.e. $\mu_s = \mu_d$). The springs and dashpots modeling the contact law are illustrated in Figure 37.

![Figure 37 Contact Model.](image)

The normal contact stiffness, $k_n$, is calculated based on the maximum allowable overlap between two particles, defined as $\delta_{\text{max}}$, and an estimated maximum possible relative speed between interacting particles and boundaries in the simulation, $v_{\text{max}}$. Equating the
kinetic energy of a discrete element with the potential energy stored in the spring when in full contact results in equation [3.45].

\[
\frac{1}{2} m_{\text{max}} |v_{\text{max}}|^2 = \frac{1}{2} k_n \delta_{\text{max}}^2
\]  

(3.45)

where \( m_{\text{max}} \) is the largest discrete element mass in the simulation. Solving for \( k_n \):

\[
k_n = \frac{m_{\text{max}} |v_{\text{max}}|^2}{\delta_{\text{max}}^2}
\]  

(3.46)

The maximum possible relative speed in a simulation is found by calculating how fast a particle would be traveling if it were allowed to free fall from one extent of the problem domain to the other extent under gravity. Note that any initial particle injection velocity is also accounted for in this calculation. Figure 38 depicts this scenario.

![Diagram](image_url)

Figure 38 Maximum Expected Velocity.

Mathematically the maximum speed is calculated as:
\[
\nu_{\text{max}} = \left| \bar{v}_{\text{max}} \right| = \sqrt{2(\bar{d}_{\text{max}} - \bar{d}_{\text{min}}) \cdot \bar{g}} + \left| \bar{v}_0 \right|^2
\]  
(3.47)

This calculation is conservative for all simulations by assuming that the particles are allowed to free fall unimpeded, the particles are injected at the extent of the problem domain, and the particles are injected in the same direction that gravity is acting.

In Section 3.1.1 Determining if Two Spheres are in Contact, and Section 3.1.2 Determining if a Sphere and a Boundary are in Contact, the equations for the amount of overlap, \( \delta \), and the normal direction at the point of contact, \( \hat{n}_c \), are derived. The normal force is calculated as:

\[
\bar{F}_n = k_\delta \delta \hat{n}_c
\]  
(3.48)

This force is applied to the two discrete elements in contact in accordance with Newtons 3rd Law.

This method calculates the total normal force every time step. Another method used by some researchers is to calculate the normal force incrementally. The contact force generated within a time step can be computed from the incremental movement of the discrete elements in contact. When using the incremental method of calculating the contact forces, the total force is computed with a summation of the incremental forces. When the elements first come into contact the amount of penetration should be calculated exactly.

Using the incremental force calculation provides no additional benefit with the linear spring contact model being used in this thesis. Therefore, the total normal contact force is calculated every time step. The incremental force calculation is needed in contact models that are path dependent.

The contact damping factor is related to the coefficient of restitution. The derivation considers the motion of two elements impacting each other and separating in one dimension. The instant the two particles first contact, defined as \( t=0 \), the displacements and velocities provide boundary conditions defined as:
\[ u_2 - u_1 - (r_1 + r_2) = 0 \]  \hspace{2cm} (3.49)
\[ \dot{u}_2 - \dot{u}_1 = v_2^0 - v_1^0 \]  \hspace{2cm} (3.50)

At the instant the elements release from contact, \( t = t_{rel} \), the boundary conditions are:
\[ u_2 - u_1 - (r_1 + r_2) = 0 \]  \hspace{2cm} (3.51)
\[ \dot{u}_2 - \dot{u}_1 = v_2^f - v_1^f = \eta(v_2^0 - v_1^0) \]  \hspace{2cm} (3.52)

where \( \eta \) in the coefficient of restitution. The particle locations and velocities at these two instances in time are shown in Figure 39.

![Figure 39 Damping Between Two Discrete Elements.](image)

The equations of motion for the two elements, ignoring gravity, are written as:
\[ m_1 \ddot{u}_1 = c(\dot{u}_2 - \dot{u}_1) + k_n(u_2 - u_1 - (r_1 + r_2)) \]  \hspace{2cm} (3.53)
\[ m_2 \ddot{u}_2 = -c(\dot{u}_2 - \dot{u}_1) - k_n(u_2 - u_1 - (r_1 + r_2)) \]  \hspace{2cm} (3.54)

Subtracting equation [3.53] from equation [3.54] results in:
\[ \ddot{u}_2 - \ddot{u}_1 = -\left(\frac{c}{m_1} + \frac{c}{m_2}\right)(\dot{u}_2 - \dot{u}_1) - \left(\frac{k_n}{m_1} + \frac{k_n}{m_2}\right)(u_2 - u_1 - (r_1 + r_2)) \]  \hspace{2cm} (3.55)

The following substitutions are made to simplify the derivation:
\[
y = u_2 - u_1 \left( r_1 + r_2 \right), \quad \dot{y} = \ddot{u}_2 - \ddot{u}_1, \quad \ddot{y} = \dddot{u}_2 - \dddot{u}_1,
\]
\[
\alpha = \left( \frac{c \left( m_1 + m_2 \right)}{2m_1 m_2} \right), \quad \beta = \left( \frac{k_n \left( m_1 + m_2 \right)}{m_1 m_2} \right)
\]

Equation [3.55] becomes:
\[
\ddot{y} + 2\alpha \dot{y} + \beta y = 0 \quad (3.56)
\]

Substituting the general solution of equation [3.56]
\[
y = Ae^\gamma \quad (3.57)
\]
equation [3.56] becomes:
\[
(\gamma^2 + 2\alpha \gamma + \beta)Ae^\gamma = 0 \quad (3.58)
\]
solving for \(\gamma\)
\[
\gamma = -\alpha \pm \sqrt{\alpha^2 - \beta} \quad (3.59)
\]

When \(\alpha^2 = \beta\) the damping factor \(c = 2\sqrt{k_n \frac{m_1 m_2}{m_1 + m_2}}\) and the system is critically damped. Cundall (1974) stops at this point: “In the present program, \(c\) was just set below this critical value for each contact, so as to make the response almost ‘dead beat’ (very slight bounce).” If \(\beta < \alpha^2\) the system is over damped. We are interested in the case when \(\beta > \alpha^2\). Rewriting equation [3.59]:
\[
\gamma = -\alpha \pm i\sqrt{\beta - \alpha^2} \quad (3.60)
\]

and substituting
\[
B = \sqrt{\beta - \alpha^2} \quad (3.61)
\]
equation [3.57] is written as:
\[
y = Ae^{-\alpha t \pm iBt} \quad (3.62)
\]
or
\[
y = Ae^{-\alpha t} \sin(Bt + C) \quad (3.63)
\]

Applying the first set of boundary conditions at \(t=0\):
\[ y(0) = 0 = Ae^{-\alpha_0} \sin(B0 + C) \]  
therefore 
\[ C = 0 \]

The relative velocity is found by taking the derivative of \( y \) with respect to time:
\[ \dot{y} = Ae^{-\alpha t} B \cos(Bt) - A\alpha e^{-\alpha t} \sin(Bt) \]  
(3.65)

Applying the second part of the first boundary condition:
\[ \dot{y}(0) = v_2^0 - v_1^0 = Ae^{-\alpha_0} B \cos(B0) - A\alpha e^{-\alpha_0} \sin(B0) \]  
(3.66)

therefore
\[ A = \frac{v_2^0 - v_1^0}{B} \]  
(3.67)

Equation [3.63] is now written as:
\[ y = \frac{v_2^0 - v_1^0}{B} e^{-\alpha t} \sin(Bt) \]  
(3.68)

The time the particles separate is found from the periodicity of the sin function:
\[ y = 0 \text{ at } t = 0 \text{ or } t = \frac{\pi}{B} \]

therefore
\[ t_{rel} = \frac{\pi}{B} \]  
(3.69)

Applying the second boundary condition of the relative velocities at the instant the elements separate:
\[ \dot{y}(t_{rel}) = v_2^f - v_1^f = \frac{v_2^0 - v_1^0}{B} e^{-\alpha \pi B} B \cos(Bt_{rel}) - \frac{v_2^0 - v_1^0}{B} \alpha e^{-\alpha \pi B} \sin(Bt_{rel}) \]  
(3.70)

and substituting the release time, equation [3.69], into equation [3.70]:
\[ \dot{y}(t_{rel}) = v_2^f - v_1^f = \frac{v_2^0 - v_1^0}{B} e^{-\alpha \pi B} B \cos(\frac{\pi}{B}) - \frac{v_2^0 - v_1^0}{B} \alpha e^{-\alpha \pi B} \sin(\frac{\pi}{B}) \]  
(3.71)

Consolidating terms:
\[ v_2^f - v_1^f = (v_2^0 - v_1^0) e^{-\frac{a}{B}} \]  
(3.72)

and introducing the coefficient of restitution from equation [3.52]

\[ \eta = \frac{v_2^f - v_1^f}{v_2^0 - v_1^0} \]  
(3.73)

where

\[ \eta = 0 \] is critically damped

\[ \eta = 1 \] results in zero damping

Substituting equation [3.72] into equation [3.73] the coefficient of restitution becomes equal to:

\[ \eta = \frac{v_2^f - v_1^f}{v_2^0 - v_1^0} = e^{-\frac{a}{B}} \]  
(3.74)

solving for \( \alpha \)

\[ \alpha = \frac{-B \ln(\eta)}{\pi} \]  
(3.75)

replacing \( B \)

\[ \alpha = \frac{-\sqrt{\beta - \frac{\alpha^2}{\pi^2}}} \ln(\eta) \]  
(3.76)

and again solving for \( \alpha \)

\[ \alpha^2 = \frac{(\beta - \alpha^2) [\ln(\eta)]^2}{\pi^2} \]  
(3.77)

\[ \alpha^2 \left(1 + \frac{[\ln(\eta)]^2}{\pi^2}\right) = \frac{\beta [\ln(\eta)]^2}{\pi^2} \]  
(3.78)

\[ \alpha^2 = \frac{\beta [\ln(\eta)]^2}{\pi^2 + [\ln(\eta)]^2} \]  
(3.79)

\[ \alpha = -\ln(\eta) \frac{\beta}{\sqrt{\pi^2 + [\ln(\eta)]^2}} \]  
(3.80)
replacing $\alpha$ and $\beta$

$$\frac{c}{2} \left( \frac{m_1 + m_2}{m_1 m_2} \right) = -\ln(\eta) \sqrt{\frac{k_n (\frac{m_1 + m_2}{m_1 m_2})}{\pi^2 + \left[ \ln(\frac{1}{\eta}) \right]^2}}$$  \hspace{1cm} (3.81)

and solving for $c$

$$c = 2 \ln\left( \frac{1}{\eta} \right) \sqrt{\frac{k_n m_1 m_2}{\pi^2 + \left[ \ln\left( \frac{1}{\eta} \right) \right]^2}}$$  \hspace{1cm} (3.82)

Computationally equation [3.82] is very expensive. It must be calculated for each new contact every time step. Some help is gained by calculating and storing the natural logarithms at the beginning of the simulation but the remaining computational work is still significant. Another solution has been found that eliminates more of the computational work. A partial damping coefficient is calculated for each particle when they are created. The partial damping coefficients are calculated as:

$$c' = 2 \ln\left( \frac{1}{\eta} \right) \sqrt{\frac{k_n m}{\pi^2 + \left[ \ln\left( \frac{1}{\eta} \right) \right]^2}}$$  \hspace{1cm} (3.83)

When two particles come into contact the correct damping coefficient is then calculated as:

$$c = \frac{c' c'_2}{\sqrt{c'^2 + c''^2}}$$  \hspace{1cm} (3.84)

Unfortunately the square root operation is still present.

When a discrete element comes into contact with a rigid fixed boundary (with infinite mass $m_2 \rightarrow \infty$) the damping coefficient is calculated as:

$$c_b = 2 \ln\left( \frac{1}{\eta} \right) \sqrt{\frac{k_n m_1}{\pi^2 + \left[ \ln\left( \frac{1}{\eta} \right) \right]^2}}$$  \hspace{1cm} (3.85)

which is conveniently the same as equation [3.83]. The derivation for the damping coefficient between a sphere and a boundary is very similar to that between two spheres and is therefore not derived.
The normal damping force is calculated by multiplying the damping coefficient, \( c \), with the relative normal velocity at the point of contact. This is written as:

\[
\vec{F}_d = c\vec{v}_{rel_n}
\]  
(3.86)

The relative velocity at the point of contact is calculated as:

\[
\vec{v}_{rel} = \left( \vec{v}_1 + \left( r_1 - \frac{\Delta t}{2} \right) \vec{n} \times \vec{\omega}_1 \right) - \left( \vec{v}_2 + \left( r_2 - \frac{\Delta t}{2} \right) \vec{n} \times \vec{\omega}_2 \right)
\]  
(3.87)

where \( \hat{n}_c \) is calculated as discussed in sections 3.1.1 and 3.1.2. The relative velocity at the point of contact in the normal direction is then calculated as:

\[
\vec{v}_{rel_n} = (\vec{v}_{rel} \cdot \hat{n}_c) \hat{n}_c
\]  
(3.88)

This damping force is applied to the discrete elements in contact.

The friction forces are calculated incrementally. They cannot be calculated exactly every time step since they are path dependent. Calculating the forces incrementally requires that the information from the previous time step is available. This adds an additional task to determine if the current contact is a new or existing contact. The relative shear displacement is incremented as:

\[
\vec{\delta}_s^{new} = \vec{\delta}_s^{old} + (\Delta t)\vec{v}_{rel_i}
\]  
(3.89)

The relative velocity at the point of contact in the shear direction is calculated as:

\[
\vec{v}_{rel_i} = \vec{v}_{rel} - \vec{v}_{rel_n}
\]  
(3.90)

The shear force is calculated as:

\[
\vec{F}_s = \vec{\delta}_s^{new} k_s
\]  
(3.91)

In this thesis, the shear stiffness is set equal to the normal stiffness.

\[
k_s = k_n
\]  
(3.92)

The magnitude of the shear force is compared to the slip friction force calculated as:

\[
F_{\text{fric\_max}} = |\vec{F}_s| \mu
\]  
(3.93)

where \( \mu \) is the specified coefficient of friction. The damping force is not included in this calculation. If the calculated shear force, equation [3.91], is larger than the maximum friction force, equation [3.93], the friction force is scaled so its magnitude is equal to the maximum
friction force. In this condition the elements are considered to be slipping relative to each other. The shear displacement in the contact spring is adjusted to account for this slip. The updated shear displacement is calculated as:

\[
\delta_s = \frac{\tilde{F}}{k_s} \frac{F_{pr,ma}}{|\tilde{F}|}
\]  

The friction force is added to the forces and moments acting on the elements in contact.

The contact stiffness calculated using the maximum particle overlap is significantly lower than the stiffness calculated with Hertz-Mindlin contact equations using the correct modulus of elasticity for the materials. This commonly raises concern among scientists and engineers using the discrete element method. The reason that the Hertz-Mindlin contact laws are not used in this thesis is that the increased stiffness would extend computer run times from days to weeks or months. The assumption being made to allow the decreased normal stiffness follows Cundall’s original work of the discrete element method.

‘Stress’ and ‘Strain’ are concepts that lose more and more meaning as the rock mass becomes more and more jointed. The truth of this statement is evident from photoelastic stress patterns in highly jointed model tests, which reveal localized stresses of high magnitude. Provided that these stresses do not cause cracking of the intact material, they are irrelevant to the movements in the rock mass that take place as a result of slip, opening and rotation across joints. Whilst stresses and strains can tend to arbitrarily large values, forces and displacements remain finite in jointed media; that is why the present program deals in forces and displacements. (Cundall 1974)

The main physical assumption made in the program is that all deformations occur at block surfaces. This is believed to be valid in situations of low stress where most displacements arise through sliding, separation, rotation and interlocking. Other methods should be used for those situations where the deformations of the intact rock are comparable to the deformations associated with joints. (Cundall 1975)

In the caving problem addressed in this work, the major movements are developed by rigid block motions rather than deformational between blocks. Therefore the earlier rational and assumptions proposed by Cundall seem to be appropriate for the present study.
In presenting the results of the discrete element simulations several quantities are calculated. The stresses acting on spherical discrete element are calculated by applying the Gauss divergence theorem to the equations of motion as (Cundall 1978):

$$\sigma_{ij} = \sum_{k}^{\text{contacts}} \frac{F_{i}^{k} x_{j}^{k}}{\text{Volume}}$$  \hspace{1cm} (3.95)

The particle pressure or mean particle stress is calculated as:

$$P = \frac{\sigma_{11} + \sigma_{22} + \sigma_{33}}{3} = \frac{1}{3} \sum_{k}^{\text{Contacts}} \left( \frac{F_{1}^{k} x_{1}^{k} + F_{2}^{k} x_{2}^{k} + F_{3}^{k} x_{3}^{k}}{\text{Volume}} \right)$$  \hspace{1cm} (3.96)

The kinetic energy of a particle is calculated as:

$$KE = \frac{1}{2} mv^2 + \frac{1}{2} I \omega^2$$  \hspace{1cm} (3.97)

The potential energy of a particle is calculated as:

$$PE = m(\ddot{g} \cdot \ddot{u}) + \sum_{k}^{\text{Contacts}} F^{k} \Delta^{k}$$  \hspace{1cm} (3.98)

For the simulations presented in this thesis a maximum allowable penetration of 10% is used and the coefficient of restitution is set to 20%. Rotational damping is not used. In most cases the angular kinetic energy will be dissipated with the friction forces. Some experimentation has been done with restraining the spheres from rotating. This was done to inhibit the ease of rotation associated with spheres.

3.3 Time Integration

For a system of discrete elements Newton’s second law, $\sum \ddot{F} = m \ddot{a}$ and $\sum \ddot{M} = \frac{d\ddot{H}}{dt}$ where $\ddot{H}_G$ is the angular momentum with respect to the center of mass (point G), is applied to each element. The equations are coupled due to contact stiffness and damping forces between elements. For each element, Newton’s second law is written for translations as:
\[ m^i \ddot{\mathbf{a}}^i + \alpha m^i \ddot{\mathbf{v}}^i = \sum_{j}^{\text{contacts}} c_i (\ddot{\mathbf{v}}^j - \ddot{\mathbf{v}}^i) + k_i (\dddot{\mathbf{u}}^j - \dddot{\mathbf{u}}^i) + \sum F^i \]  

(3.99)

where \( \alpha m^i \ddot{\mathbf{v}}^i \) is a mass proportional damping term and \( j \) is summed over the elements in contact with element \( i \). A similar equation is written for the rotations. These coupled equations can be solved using an implicit numerical method, however, typically with the discrete element method the contact damping and stiffness forces are calculated at the current instant in time and included with the applied forces. This de-couples the equations which can now be solved with either an implicit or explicit numerical method.

Traditionally a simple explicit central difference integration method has been used. (Cundall 1974; Hustrulid 1993; Zhang 1993; Mathews 1994; Lin 1995) However, recently (O’Connor 1996) reports using the Runge-Kutta-Nyström scheme, and (Greening 1996) implemented Newmark’s method. O’Conner and Greening do not indicate why they have chosen these more complicated integration schemes. In this thesis the explicit central difference scheme is used.

The choice to use an explicit numerical method over an implicit method is based on several factors. Using an implicit method can involve slightly more computational work, however, it provides unconditional numerical stability and can allow the use of a larger time step. To take advantage of the larger time step and ensure accuracy the implicit method also requires that the system’s behavior is not highly path dependent. This last requirement is rarely satisfied with systems modeled using the discrete element method. The major physical discontinuous behavior is a result of contact creation, termination, and stick-slip friction. This physical behavior will not be modeled accurately if the time step in an implicit method is greater than the duration of the contact. The original argument made by Cundall for using an explicit numerical technique still holds true.
We have seen that failure in a well-jointed rock mass is preceded by numerous, small shuffling motions of joints as loads are re-distributed. It is not improbable that the particular sequence and magnitude of these motions is of some importance to the conditions precipitating final failure; i.e., the final outcome may depend on the path taken to get there, due to the high degree of geometrical non-linearity present in a jointed mass. In view of this, it seems safer to use an explicit numerical method, which marches time on by very small steps, as opposed to an implicit method, which, although it can take much larger steps, may perhaps skip over some important irreversible effects. (Cundall 1974)

The central difference explicit method for the translational motion is now derived in some detail. The derivation for the rotation of the discrete element is nearly identical.

Equation [3.99] is de-coupled by calculating and including the contact damping and stiffness forces with the applied forces $\vec{F}_r$, so that:

$$m^i \ddot{a}_r^i + \alpha m^i \dot{v}_r^i = \vec{F}_r^i$$  \hfill (3.100)

where

$\alpha$ is a global viscous damping term as shown in Figure 40.

$$\vec{F}_r^i = \sum_{j}^{\text{contact}} \left[ c \left( \dot{v}_r^i - \dot{v}_r^j \right) + k \left( \ddot{u}_r^i - \ddot{u}_r^j \right) \right] + \sum \vec{F}_r^i$$  \hfill (3.101)

$\vec{F}_r^i$ are global applied forces such as gravity $(m^i g)$.

![Figure 40 Global Viscous Damping.](image)

The central difference approximations for acceleration and velocity shown in equations [3.102] and [3.103]:
\[
\bar{a}_r = \frac{\bar{v}_{r+\frac{\Delta}{2}} - \bar{v}_{r-\frac{\Delta}{2}}}{\Delta t} \tag{3.102}
\]
\[
\bar{v}_r = \frac{\bar{v}_{r+\frac{\Delta}{2}} + \bar{v}_{r-\frac{\Delta}{2}}}{2} \tag{3.103}
\]

are substituted into equation [3.100]. Solving for \( \bar{v}_{r+\frac{\Delta}{2}} \) yields:

\[
\bar{v}_{r+\frac{\Delta}{2}} = \frac{(1 - \frac{a \Delta t}{2})\bar{v}_{r-\frac{\Delta}{2}} + \frac{\bar{F}_r}{m} \Delta t}{1 + \frac{a \Delta t}{2}} \tag{3.104}
\]

If the mass proportional damping is constant throughout the simulation equation [3.104] simplifies to:

\[
\bar{v}_{r+\frac{\Delta}{2}} = a\bar{v}_{r-\frac{\Delta}{2}} + b \frac{\bar{F}_r}{m} \Delta t \tag{3.105}
\]

where

\[
a = 1 - \frac{a \Delta t}{2} \tag{3.106}
\]
\[
b = \frac{1}{1 + \frac{a \Delta t}{2}} \tag{3.107}
\]

are calculated at the beginning of the simulation to save on computational time. If there is no mass proportional damping equation [3.105] further reduces to:

\[
\bar{v}_{r+\frac{\Delta}{2}} = \bar{v}_{r-\frac{\Delta}{2}} + \frac{\bar{F}_r}{m} \Delta t \tag{3.108}
\]

The updated positions are calculated by taking a central difference approximation of the velocity:

\[
\bar{v}_{r+\frac{\Delta}{2}} = \frac{\bar{u}_{r+\Delta} - \bar{u}_r}{\Delta t} \tag{3.109}
\]

and solving for the updated position:
\[ \ddot{u}_{r+\Delta t} = \ddot{u}_r + \dddot{v}_{r+\frac{3}{2} \Delta t} \] (3.110)

The discrete elements have now been incremented one time step and the procedure is ready to repeat again.

The time step selected for the explicit central difference numerical technique is very important. Too large a time step results in instability and incorrect solutions. Conversely too small a time step produces correct results but wastes CPU time. Unfortunately calculating the exact critical time step for a system of discrete elements requires finding the eigenvalues of the complete system. With the configuration of particles changing continuously over time the critical time step calculated at one point may be too large an instant later. To avoid this computational expense an estimate of the critical time step is traditionally made at the start of a simulation.

A single degree of freedom system without damping is shown in Figure 41.

![Figure 41 Single Degree of Freedom System.](image)

For this model the equations of motion are solved with a central difference numerical scheme written as:

\[ 0 = u_{r+\Delta t} + \left( \frac{k}{m} \Delta t^2 - 2 \right) u_r + u_{r-\Delta t} \] (3.111)

The difference equation is solved giving stable oscillatory \( u \) for:
\[ \Delta t < 2\sqrt{\frac{1}{k_i}} \]  
and exponentially increasing \( u \) for greater \( \Delta t \). A similar relation also holds for angular motion:
\[ \Delta t < 2\sqrt{\frac{1}{k_i}} \]
where \( k_i \) is the shear stiffness and \( I \) is the moment of inertia. This stability analysis follows Cundall (1974).

The critical time step for a single degree of freedom system provides an upper limit on the time step for a multibody discrete element simulation. In practice a fraction of this critical time step is used. Exactly what fraction is appropriate is open to debate. Theoretically it is based on how many particles one particle could be in contact with at any one time. In a simulation with a significant size distribution of particles this number is essentially unbounded. Cundall (1974) states that “in the present program \( \Delta t \) has been chosen sufficiently small to accommodate most situations”. More specific ranges of appropriate critical time step factors are later defined by Cundall. “A value of FRAC of 0.1 is probably safe for most problems, but 0.2 to 0.5 may be used with caution for loosely-packed assemblies.” (Cundall 1978) Zhang (1993) reports using a time step of 5% of the critical time step. Mathews (1994) suggests that the percentage for stability should be kept below 20%. If damping and friction forces are being modeled the system will appear “stiffer” and the percentage of critical time step should be adjusted appropriately. In this thesis a time step of 10% of the critical time step has provided stable results.

If the time step is too large the total energy in a closed system will increase. The forces acting on the discrete elements are also seen to fluctuate wildly. If these behaviors are observed the simulation should be re-run with a smaller time step.

An interesting observation of the time step in a system of discrete elements with a constant normal stiffness and size distribution from \( r_{\text{min}} \) to \( r_{\text{max}} \) can be made. For simplicity it is assumed all the discrete elements have the same density. From equation [3.112] the critical time step is written as:
\[ \Delta t \propto \sqrt{\frac{m_{\text{min}}}{k}} \]  \hspace{1cm} (3.114)

where \( m_{\text{min}} \) is the mass of the smallest particle. The contact stiffness from equation [3.46] is expanded to:

\[ k = \frac{m_{\text{max}} v_{\text{max}}^2}{\delta_{\text{max}}^2} = \frac{\rho \frac{4}{3} \pi r_{\text{max}}^3 v_{\text{max}}^2}{(\xi r_{\text{min}})^2} \]  \hspace{1cm} (3.115)

where \( \xi \) is the maximum allowed overlap measured in units of the smallest particle. Substituting equation [3.115] into equation [3.114] and defining \( m_{\text{min}} \) equation [3.114] becomes:

\[ \Delta t \propto \sqrt{\rho \frac{4}{3} \pi r_{\text{min}}^3 (\xi r_{\text{min}})^2} \]  \hspace{1cm} (3.116)

Collecting terms:

\[ \Delta t \propto \sqrt{\frac{\xi^2 r_{\text{min}}^5}{r_{\text{max}}^3 v_{\text{max}}^2}} \left( \frac{r_{\text{min}}}{r_{\text{max}}} \right) \]  \hspace{1cm} (3.117)

The observation is that as the limits of the size distribution of discrete elements increase the critical time step becomes significantly smaller.

One solution to this behavior that can be applied to pseudo static simulations such as settling and possibly in compaction problems, is to use inertial density scaling as described by Cundall (1974). In this technique the inertial density of all the discrete elements in the system are set so that the masses (inertia) of all the elements are equal for the time integration. With inertial density scaling equation [3.104] becomes:

\[ \bar{v}_{i,\text{ct}} = \frac{(1 - \frac{\rho_{\text{ct}}^2}{2}) \bar{v}_{i,\text{ct}} + \frac{\bar{F}_{\text{ct}}}{m'} \Delta t}{1 + \frac{\rho_{\text{ct}}^2}{2}} \]  \hspace{1cm} (3.118)

where

\[ m' = 1.0 \] for all discrete elements in the system

The density used for calculating the gravity forces included in \( \bar{F}_{\text{ct}} \) (equation [3.101]) is left unchanged. With inertial density scaling equation [3.117] reduces to:
\[ \Delta t \approx \frac{\xi r_{\min}}{v_{\max}} \]  

(3.119)

which shows the time step is now independent of the discrete element size distribution. Inertial density scaling does not appear applicable to modeling flow problems because of the dynamics of the flow behavior.

An important book keeping note is that in the central difference time marching scheme the discrete element velocities are calculated at the half time steps. The damping calculation requires the velocity at the current instant in time (i.e. at the full time step). The velocity values, however, are computed at half time steps. Cundall (1974) assumed that \( \tilde{v}_{t-\frac{1}{2}} = \tilde{v}_t \) and stated that in pseudo static problems he dismissed the issue of modeling damping accurately. Cundall makes the statement, “as not too important from the point of view of accuracy, since the dashpots are only there to absorb energy, and not to model faithfully irreversible processes at rock contacts”. Zhang (1993) uses a linear extrapolation as shown in Figure 42 to estimate the velocity at the full time step to achieve a more accurate computation of the contact damping forces.

![Figure 42 Velocity Extrapolation.](image)

Mathematically the velocity at time \( t \) is calculated as:
\[ {\bar{v}}_t = 1.5{\bar{v}}_{t-1} - 0.5{\bar{v}}_{t-2} \]  

(3.120)

A higher order extrapolation function could be used, however, this would require additional storage of previous velocities.

3.4 Parallel Implementation of the Discrete Element Method

Implementation of the DEM on a parallel computer brings the opportunity of being able to solve larger, more complex, physical systems today. The DEM algorithm appears to be “inherently parallel”. This section develops a parallel, two-dimensional algorithm and its implementation on a 64 processor Transputer based machine in the Parallel Processing Lab at Colorado School of Mines (CSM). The parallel logic, however, is applicable to other parallel and multiprocessor machines.

The ideas and implementation possibilities of a parallel discrete element algorithm have been presented by several researchers. However, as O’Connor (1996) points out, documented implementations are significantly fewer. Ghaboussi et al. (1993) discuss a parallel algorithm that uses neural networks to avoid or at least minimize the communication between processors during contact checking. Hustrulid (1994) presents a two-dimensional parallel implementation of the DEM that is the basis of this section of the thesis. Schröder (1995) presents a method that is very similar to Hustrulid’s (1994) work. Schröder divides the problem domain into a two-dimensional grid to improve load balancing. Schröder’s work is part of a three year project supported by the EPSRC. O’Connor (1996) reports results from distributed sorting and contact checking used in the Xmal DEM code at MIT. O’Connor uses the Message Passing Interface (MPI) to achieve portability between different machines. The distributed contact checking used by O’Connor divides the problem so that each processor is responsible for checking contacts for N/P particles where N is the total number of particles in the system and P is the number of processors. This division is possible because of the spatial sorting algorithm used by O’Connor. The implementation is somewhat simplified by each processor maintaining a complete set of data for the simulation. This simplification would not be possible on machines with limited resources such as the Transputer. The work by scientists at Los Alamos
on short-range molecular dynamics should also be noted. A simulation with 11.3 million atoms has been run on a 512-processor computer at the Advanced Computing Laboratory (Beazly et al. 1995).

3.4.1 General Parallel Issues

Today there are many different types of parallel computers, each with their specific advantages and disadvantages. Because of the different types of parallel computers available, the development of an optimal, general purpose, parallel, DEM algorithm is not possible. To take full advantage of the different parallel machines, programs and algorithms must be custom tailored to each machine. As new machines are introduced and older designs fall by the wayside new algorithms and implementations must be written. Traditionally this is one of the major disadvantages associated with parallel machines. One method that is being used to avoid continually updating algorithms is to design and write the parallel algorithm for a virtual parallel machine. An interface tool kit is then used to implement the virtual parallel machine on different parallel architectures. The disadvantages of this method are: (1) the resulting algorithm is not optimized for a specific machine, and (2) the extra overhead of the interface tool kit. The implementation in this section is written specifically for the Transputer.

There are many different types of parallel machines available. They are commonly roughly categorized by their level of granularity as being either fine grained, medium grained, or coarse grained.

Fine grained machines, like a Connection Machine, do small, simple, calculations on each processor. They are fast at performing large matrix and vector operations, such as fast Fourier transforms, in parallel. Commonly fine grained machines are used to solve Finite Element problems in parallel. With the DEM, fine grained machines could possibly be taken advantage of in calculating the contact between very complicated discrete elements.

Medium grained machines are characterized by rather powerful processors, all on a local bus. Each processor has its own memory or access to shared memory. Examples of medium grained parallel machines are the Transputer, a multi-processor Intel Pentium Pro, and
the SGI Power Challenge. The author believes that medium grained machines are best suited for a DEM algorithm because of their flexibility for individual processors to perform different tasks and the high speed communication links between processors. The parallel algorithm presented in this section is intended for medium grained parallel machines.

Course grain parallel machines are very common. They are defined by individual computers linked to each other via a LAN, WAN, Intranet, or even the Internet. The shortcoming that limits their use with DEMs is the relatively slow communication links between computers. This shortcoming will be minimized or possibly eliminated with increases in communication speeds. DEM simulations where communication bottlenecks are less significant, such as those with a large number of very complicated elements, may be able to take advantage of large grain parallel machines. Course grained machines could also be used with the DEM if the goal of implementation is to solve P slightly different simulations in the time required to solve one simulation in a serial mode.

The parallel algorithm is described using threads. Threads are paths of program execution that represent concurrency in the algorithm. Threads are used to describe the parallel algorithm to accommodate symmetric and asymmetric parallel machines. In an asymmetric parallel machine each processor is assigned and executes one individual thread. It is the programmers responsibility to insure that each processor receives an equivalent amount of work for good load balancing among the processors. A symmetric multiprocessor machine handles multiple threads on each processor. The operating system distributes the threads among the processors to ensure load balancing.

3.4.2 The Parallel DEM Algorithm

The goal of developing a parallel algorithm on a machine with P processors must be defined. Two possible goals are to (1) solve one simulation P times faster, or (2) solve P slightly different simulations in the time required to solve one simulation in a serial mode. In this research the goal is to solve one simulation P times faster.
The current parallel implementation of the DEM is structured for a medium grained parallel machine. The design highlights all of the major areas of the method that can be performed in parallel. Just because an area can be performed in parallel does not mean it should be. Deciding which areas of the method are done in parallel is influenced by the targeted parallel machine and the potential time savings.

The characteristics of a system modeled using the discrete element method lead to two physical ways of dividing the problem up to be run on a parallel machine. One method is to have an individual thread for each discrete element. These threads are referred to as “DE Threads”. The second method is to divide the problem up geometrically and have a unique thread responsible for each portion of the modeling space. These threads are referred to as “Geometry Threads”.

Figure 43 shows the discrete element algorithm broken down into concurrent processes that can be executed in parallel. The algorithm incorporates DE Threads and Geometry Threads and shows how they can work together.
Figure 43 Parallel Flow Diagram.
There are two concerns when using DE Threads: (1) The number of DE threads can be much greater than the number of processors on a machine. If a “Fork Always” type procedure is used, the overhead of managing the threads can be greater than the concurrency savings. “Fork When Idle” and “Work Crew” type procedures can be used to manage thread creation and execution to handle this concern. (2) The amount of time spent executing the functions in the DE Threads is significantly less than the amount of time spent doing contact checking. Significant time savings can not be found by only using DE Threads.

With Geometric Threads there are three concerns that relate to load balancing and communication between threads. These concerns are: (1) Effort needs to be taken to ensure the amount of particles in each Geometry Thread remains balanced. This load balancing must be done on a problem specific basis. (2) The influence of gravity on the nature of the problem should be considered in breaking the problem up geometrically. (3) The amount of required communication between Geometry Threads needs to be taken into consideration. One method of minimizing the amount of communication required is to minimize the areas of the shared boundaries between geometry volumes.

The amount of time required to setup the DEM model is very small in comparison to the remainder of the algorithm. The setup is also performed only once in contrast to the thousands of times the remaining portion of the algorithm is executed. For these reasons the setup of the DEM model is not done in parallel.

In certain DEM implementations there is the possibility of taking advantage of parallelism in areas not included in Figure 43. These include the steps “Locate DE’s in Data Structure” and “Find Contacts, Calculate, and Apply Contact Forces to All DE’s”. O’Connor implements a parallel sorting algorithm in the “Locate DE’s in Data Structure” step in MIT's Xmal program (O’Connor 1996).

The parallel algorithm implemented breaks the problem up geometrically. Different geometric areas of the problem domain are handled by different processors. The question then becomes how to break up the problem geometrically. The obvious choices are in rows, columns, or in a grid structure. The breakup should result in an even distribution of bodies
among the processors. Looking at the types of models used in the DEM, the bodies tend to head for the bottom of the modeling area due to gravity. Dividing the problem into rows or a grid would leave areas without bodies and processors without work to do. To avoid poor load balancing, the problem is divided into \( P \) columns, where \( P \) is the number of processors.

An example of the division of a problem onto three processors is depicted in Figure 44. In this example the domain is divided into nine columns and one row. Each processor is responsible for the discrete elements in three columns.
Figure 44 Geometrical Division.
After the problem is divided among processors the discrete elements in columns on processor boundaries are shared (communicated) with adjacent processors. Each processor now has enough information to find, calculate, and apply the contact forces for all of the discrete elements the processor is responsible for. After the contact forces are calculated the discrete elements in the buffer regions of each processor are dropped (deleted). Finally the velocities and positions of the discrete element are updated and moved between processors if necessary.

Communication between the processors is needed to share information about particles which are on or near the geometrical division boundaries. The method of scheduling the communications between adjacent processors is important. The initial impulse of having each processor read from the “left”, write to the “right”, read from the “right”, and then write to the “left” is easy to implement but very inefficient. When this method is used the communication becomes serial with each processor waiting on an adjacent processor. A more efficient communication scheme can be used with the following steps:

1. Odd processors write to the “right”, even processors read from the “left”
2. Even processors write to the “left”, odd processors read from the “right”
3. Even processors write to the “right”, odd processors read from the “left”
4. Odd processors write to the “left”, even processors read from the “right”

These two different communication schemes are depicted in Figure 45.
Figure 45 Processor Communication Scheduling.
The communication time is $2(P - 1)$ where $P$ is the number of processors for the inefficient communication method and 4 for the efficient communication method independent of the number of processors.

3.4.3 Implementation on a 64 Processor Transputer from Alta Technology

The Parallel Processing Lab at the Colorado School of Mines, shown in Figure 46, has two parallel computers. The cubic box under the power outlets is a P-Cube from Parsytec and contains 10 T805 processors. The box just to the right of the P-Cube is the parallel computer from Alta Technologies which has 64 T805 processors installed.
Figure 46 Parallel Processing Lab at Colorado School of Mines.
3.4.3.1 Introduction to the Transputer

The two-dimensional parallel algorithm is implemented on is a 64 node Transputer from Alta Technologies. Each node is a 32 bit T805 Transputer with its own memory and high speed communication links. The Transputer chip was developed by Inmos Ltd. “The name Transputer was derived from TRANSistor and comPUTER, since the component was to be a basic building block like a transistor but a complete computer on a chip.” (Hull et al. 1994)

Each T805 Transputer is a full 32-bit processor with an on-chip floating point unit, 4 kbytes of on-chip ram, and 4 Mbytes of external ram. The T805 processor is rated at 30 MIPS. Each processor also has 4 bi-directional communication links which can transmit data at rates up to 20 Mbits/second. The communication links are designated as 0 through 4 (Hull et al. 1994).

The 64 node Transputer at the Colorado School of Mines is configured into two banks of 32 processors each so that two users can be working on the parallel computer simultaneously. The 32 Transputers are physically connected in a two-dimensional grid structure, with three external links to a Sun workstation. This “interconnectivity” is defined in a “network configuration file”. Each set of 32 processors has its own network configuration file.

An understanding of how the 32 processors are physically connected can be gained by looking at the processor connections as shown in Figure 47.
Figure 47 Transputer Network Configuration.

- **Links to Sun Workstation (T225)**
- **Transputer (T805)**
The 4x8 grid layout of the Transputers is easily seen. Inside the Transputer case there are physically four cards, each containing eight Transputers, that are connected in this manner. How the Transputers are connected to the external Sun workstation is important. A linear “pipeline”, via the communication links, can be constructed with this layout.

3.4.3.2 Implementation using Virtual Channels

Ideally any two processors in a parallel computer could send and receive data between each other regardless of the processor network’s physical topology. Unfortunately each T805 processor is limited to communicate only with the processors that are physically connected to it.

To allow greater flexibility with the T805, software virtual channels have been developed for the T805 processor. These software virtual channels handle a message passage system on each processor. This allows the user to define a virtual connectivity between the processors. Because virtual channels are implemented with software on the T805, there is a performance penalty.

In the implementation of the DEM, virtual channels were initially used because of their flexibility in implementation and their ability to directly output information to the screen from any processor. This ability to output information to the screen from any processor also greatly aids in debugging. Once the implementation was working using virtual channels, the communication links were changed to real channels to increase performance.

The breakup of the problem onto the parallel machine is shown in Figure 48.
The thickness of the communication links connecting the processors represent the expected amount of data to be transferred. The thick lines represent links where the majority of the communication is expected and the thin lines represent links with minimal communication. When implementing virtual channels the expected amount of communication for each channel can be assigned with weighted values, 1 being low and 100 being high. The network loading pattern and the virtual channel connection are defined in a "network information file".

3.4.3.3 Implementation using Actual Channels

Actual channels are the physical links between the processors. Using actual channels eliminates the overhead of the virtual channels and results in a more efficient implementation. When using actual channels rather than virtual channels, each processor can only communicate with the processors which are directly attached to it. Great care must be taken in the
implementation of an algorithm on a particular processor layout. Additionally, only the processor which is connected to the Sun workstation can output information to the screen and handle disk input/output.

The major "challenge" of using actual channels is the successful mapping of the desired processor configuration onto the physical layout of parallel machine. With the implementation of the DEM model it is not obvious how to map the layout shown in Figure 48 onto the Transputer configuration shown in Figure 47. Fortunately, the major communication links will be between adjacent processors, which aids in the mapping to the Transputer. The mapping of the DEM onto the Transputer is shown in Figure 49.
Figure 49 DEM Actual Channel Transputer Configuration.
The communication link 1 for each processor is always connected to communication link 2 of the adjacent processor.

3.4.4 Performance Comparison Between Serial and Parallel Implementations

A parallel implementation of the discrete element method has been developed so that large models can be run in a minimal amount of time. Ideally using twice as many processors would get the job done in half of the time. Unfortunately, due to losses resulting from the breakup of the problem onto the multiprocessors, communication overhead, and uneven load balancing among the processors, the resulting speedup is less than ideal. This section develops an estimate of the theoretical speedup expected from the parallel implementation of the DEM and compares it with the actual speedup obtained from a simple problem.

3.4.4.1 Theoretical Speedup

The theoretical speedup gives an estimation of how much faster a program will run on P processors compared with how fast a program runs on a single processor. For example, linear speedup would be defined as \( S(P) = P \). An estimate of the speedup expected from this implementation of the DEM is presented in this section.

Reviewing Figure 48, one processor is dedicated to managing the file input and output and distributing the model to the other processors. This processor is termed the managing processor and does not do any calculations except when it is the only processor in the system. When more than one processor is in use, the managing processor will only handle the system administration. Therefore, the speed of using two processors should be equivalent to the speed of using only one processor. It is assumed that the managing processor never becomes a bottleneck for the system – limiting the system performance.

The load balancing within the system will affect the speedup of this algorithm. Load balancing refers to the amount of work each processor has to perform. Ideally, all of the processors will have the same amount of work, finish each step at the same time, and not have to wait for the other processors to complete their work. In the DEM, good load balancing is
obtained by having each processor responsible for the same number of discrete elements. The processor with the most discrete elements will be the limiting processor. A simple load balancing method has been used in this implementation -- gravity. By dividing the problem into columns, as opposed to rows, gravity will tend to equalize the number of particles in each column. A more complex, dynamic load balancing method could be developed.

In the serial algorithm the majority of the processor time is spent on determining contacts and calculating the resulting contact forces between bodies. The way the DEM problem has been divided to run on separate processors has increased the number of contact checks performed. The increase in the number of contact checks, resulting from duplicate contact checks occurring interface of geometry regions, is a function of the number of processors and the specific problem. The number of additional contacts cannot be determined exactly, however, an approximate relationship can be obtained by making several assumptions:

- The problem is relatively well load balanced among the processors. This means that each processor has approximately the same number of particles and grid columns.

- The number of columns in the grid structure is proportional to the number of contact checks.

The problem domain is broken up into a number of columns which are particle diameter width. The number of original columns is designated as GNX. Therefore, there will be GNX-1 contact checks between columns. For two processors, one managing and one calculating, the time required, T(2), is defined as T(2) = G NX-1. As the problem is broken into more pieces (data is sent to more processors), the time complexity is defined by the processor with the most columns. An example of the breakup of the problem onto 2, 3, and 5 processors is shown in Figure 50.
Number of Processors
P = 2
Number of Columns
GNX = 20
Maximum Number of Columns on a Single Processor
\((\text{GNX})/(\text{P}-1)\) = 20
Time Complexity on P Processors
\(T(P) = T(2) = (\text{GNX})/(\text{P}-1) = 19\)

Number of Processors
P = 3
Number of Columns
GNX + (P-1) = 22
Maximum Number of Columns on a Single Processor
\([\text{GNX} + (\text{P}-1)]/(\text{P}-1)\) = 11
Time Complexity on P Processors
\(T(P) = T(3) = [\text{GNX} + (\text{P}-1)]/(\text{P}-1) = 10\)

Number of Processors
P = 5
Number of Columns
GNX + 2*(P-2) = 26
Maximum Number of Columns on a Single Processor
\([\text{GNX} + 2*(\text{P}-2)]/(\text{P}-1)\) = 7
Time Complexity on P Processors
\(T(P) = T(5) = [\text{GNX} + 2*(\text{P}-2)]/(\text{P}-1) = 6\)

Original Column  Original Column  Contact Checks Between Columns

Figure 50 Break up of Problem onto P Processors.
The time complexity when using $P$ processors becomes:

$$T(P) = \frac{GNX + P - 3}{P - 1} \quad (3.121)$$

The balance of the CPU time in the serial algorithm is spent moving the particles, updating the grid structure, and outputting information to an output file. The time required for moving the particles and updating the grid structure should be inversely proportional to the number of processors. The time required for outputting information to an output file is independent of the number of processors.

A final influence on the theoretical speedup of the DEM is the communication between adjacent processors. This time complexity is not a function of the number of processors and is highly problem specific. No estimation of the time for communication versus the time for computation has been computed, but it is assumed that the time required for computations is much larger than the time for communications.

To calculate a rough estimate of the theoretical speedup, it is assumed that the speedup depends entirely on the number of contact checks. The speedup is the time complexity for one processor, $T(1)$, divided by the time complexity for $P$ processors, $T(P)$, becomes $S(P)$. This is written as:

$$S(P) = \frac{GNX - 1}{\frac{GNX + P - 3}{P - 1}} \quad (3.122)$$

To obtain the most economical speedup it is desirable to have $GNX$ much larger than the number of processors, however, in a physical modeling situation $GNX$ will normally be dictated by the problem's geometry and maximum particle size.

An important assumption that the processors have balanced loads has been made. The problem has been broken up into processors with columns to take advantage of the natural load balancing which will occur from gravity. Gravity will ensure that a flat level of particles which will balance the particles. If a model without gravity is run, the load balancing will need to be addressed. A future enhancement of the current algorithm could be to develop a dynamic load balancing scheme which varies the number of columns sent to each processor.
3.4.4.2 Measured Speedup

A two-dimensional simulation with 625 spherical discrete elements is used to evaluate the performance of the parallel two-dimensional DEM algorithm. In the simulation a block of elements is dropped into a container and allowed to settle as shown in Figure 51. The shading of the elements represents the mean stress or pressure.
Figure 51 Two-dimensional Simulation.
This 10 second simulation took approximately half of an hour to run on one Transputer. The model was run on 1 to 32 T805 processors and on an SGI Indigo² Extreme workstation. The speedup curve is show in Figure 52.

![Graph showing speedup of 625 particles over 10 seconds with GNX = 63](image)

**Figure 52 Measured Speed Up.**

The model was run using both actual and virtual communication links to obtain a comparison in the performance. It can be seen that the realized speedup is well below the theoretical speedup for this model. An important note is that with 32 processors and only 63 columns each processors is only responsible for at most two columns. This distribution forces inter-processor communication to be the limiting factor affecting performance. A more representative view of the speedup is seen by looking at only 1-10 processors as shown in Figure 53.
With ten processors a speedup of 60% of theoretical is obtained. It is also interesting to note that the shape of the speedup curves in Figure 52 and Figure 53 have similar characteristics to the theoretical speedup.

3.4.5 Conclusions for Parallel Processors

A parallel algorithm for the DEM has been successfully developed and implemented using 32 nodes of a 64 node T805 Transputer. The speedup, while less than ideal, shows significant improvement over a serial implementation. By using 32 processors a speed increase of nearly 8 times was obtained for the example problem.

Communication seems to be the limiting factor for the algorithm developed. Methods of reducing the amount of communication between processors should be investigated. A DEM implementation which uses more complicated bodies would greatly increase the amount of computation time on each processor and, therefore, negate the communication limitations. Larger models will also have better parallel performance over smaller models.
While the parallel algorithm was implemented on a Transputer, the algorithm should be portable to a number of Single Program Multiple Data (SPMD) distributed memory machines with minimal changes. A slightly different algorithm could be developed for a shared memory machine which would eliminate the communication limitations.

Initial results indicate that the realized speedup is less than ideal. Testing shows that utilizing 32 processors on the Transputer results in a simulation speed similar to a SGI Indigo² Extreme workstation. The similar speed of the 32 processor Transputer with the SGI, the increased difficulty and temperament of using the Transputer, and the lack of a C++ compiler for the Transputer resulted in the decision not to develop a three-dimensional DEM algorithm for the Transputer.

3.5 Object-Oriented Implementation of the Discrete Element Method

In this work Object-Oriented modeling and design are used to implement the DEM. The Object-Oriented technique elegantly encapsulates the data and behavior of the discrete elements and the contacts within a simulation. There are several different techniques for designing with and documenting Object-Oriented implementations. This research uses the Object Modeling Technique (OMT) (Rumbaugh et al. 1991). In practice the design is implemented in the C++ programming language. The design could also be implemented in other object oriented languages such as Java (Richey 1995).

Several researchers report using object oriented implementations of the discrete element method (Munjiza et al. 1992; Williams et al. 1992; Hustrulid 1993; Oelfke et al. 1994; Oelfke et al. 1995; Oelfke, et al. 1996). This section does not present a detailed discussion on the different aspects of object oriented programming such as inheritance, polymorphism, etc. Instead, the focus is on the object oriented structure and relationships of objects in the discrete element method.

The Object Modeling Technique (OMT) is employed to structure the design and implementation of the DEM. The OMT describes a physical system with three views, (1) the object model, (2) the dynamic model, and (3) the functional model. These views work together
to provide a complete description of the system. Both the DEM and OMT are used to model physical systems. This similarity is one reason the OMT is used with the DEM.

3.5.1 The Object Model

The object model describes the structure of objects in a system—their identity, their relationships to other objects, their attributes, and their operations. The object model provides the essential framework into which the dynamic and functional models can be placed. (Rumbaugh et al. 1991)

The objective of the implementation is to model the behavior of a large system of particles modeled as spheres. These particles interact with each other and with the boundaries through contact. Four base classes are defined in the DEM. These classes are: Discrete Element, Contact, Discrete Element (DE) Container, and Contact Container. From these base classes, new classes are defined which inherit the properties and behaviors of their parents. These classes and their relationships are shown in Figure 54.
Figure 54 Object Model.
**Discrete Element** – Discrete elements are the bodies or particles whose movement and interactions define a discrete element simulation. The discrete element class is intended as an abstract class. An abstract class is defined by a class having at least one pure virtual function and therefore it is not possible to instantiate (create) a discrete element object. Instead the discrete element class defines the properties and behaviors that derived (children) classes must have. The properties defined for a discrete element are the descriptive geometry, velocity, resultant forces and moments, and material properties. Behaviors or methods that are defined for the discrete element class are the ability to store and retrieve defining data to a file, check contact with other discrete elements, maintain a list of contacts, and be stored, located, and retrieved in a data structure. Additional properties and behaviors may need to be added to the discrete element class with future developments.

**Sphere** – Spheres are the only discrete elements in the current implementation that are able to move. A sphere’s geometry is described with the center location and radius. The only material property is the density.

**Boundary** – The boundary class is an abstract class in the current implementation. Boundaries are treated in a special manner in the grid container class used to store the discrete elements and highlight possible contacts between elements. The boundary class is added so that the grid container can identify and handle these special elements. Specifically the method that boundaries “tag” cells in the grid container is different than other discrete elements like spheres as discussed in detail in section 3.1.3.4. In this implementation only a plane boundary is defined as a descendent of the boundary class. Future additions could include arched boundaries, and boundaries that move through space in either a stress controlled or displacement controlled manner.

**Plane Boundary** – The geometry of plane boundaries is defined by the number of vertices and their locations in space. Plane boundaries are assumed to be planar surface whose edges form a convex polygon. Future enhancements could include plane boundaries with edges that form a concave polygon. These plane concave boundaries would be treated as a descendent or child of the plane boundary. The properties included with a plane boundary
include the coefficient of friction, time on, and time off. The time on and time off properties define when a plane boundary is active during a simulation.

**DE Container** – The DE container class is designed to provide an interface to the different methods of contact searching. To provide efficient access to the discrete elements the discrete elements are physically stored in the DE container. The DE container allows external access to the discrete elements for the flexibility required in specialized discrete element applications. For a specified discrete element the DE container returns a list of other discrete elements that are likely to be in contact with the specified element. The DE container is implemented as an abstract class. Alternatively it could be implemented as the all to all container that in most cases would not be used but would provide basic functionality.

**All to All Container** – The all to all container is not implemented. The idea of the all to all container is that all discrete elements would be checked against all other discrete elements for contact. This is obviously not the best method to search for potential contacts between elements in a simulation, however, there may be some benefit in using this method in verifying and evaluating the performance of other contact searching methods.

**Grid Container** – The grid container is the container implemented in this thesis. The contact checking methodology is fully described in section 3.1.3. Plane boundary elements are treated in a special manner by the grid container class.

**Threaded Grid Container** – The threaded grid container is not implemented. It is envisioned that the threaded grid container will eventually encapsulate the research included in section 3.4 of this thesis. The development of the threaded grid container requires a C++ or other object oriented compiler for a parallel machine.

**Sort Container** and **Thread Sort Container** – These two containers are included to illustrate how the work of other researchers can be accommodated by the object model of the discrete element method. The heap sort method presented by O’Connor (1996) is an example of a sort container that could be added to the current implementation.

**Contact** – The contact class is implemented as an abstract class. Several methods are defined in the contact class that are common to descendental contact classes. The methods
include the calculation of the normal, shear and friction forces. The contact properties including stiffness and damping are defined in this class.

**Sphere-Sphere Contact** – The contact determination between two spheres is encapsulated in the sphere-sphere contact class. Section 3.1.1 details the calculations.

**Boundary-Sphere Contact** – The contact determination between a sphere and a plane convex boundary is defined in the boundary-sphere contact class. Section 3.1.2 details the calculations.

**Contact Container** – The contact container is added to improve the performance of the implementation. In C++ it is possible to dynamically allocate and de-allocate computer memory. With thousands of contacts possibly created and destroyed each time step, the overhead of memory allocation and de-allocation is significant. Additionally the allocation and de-allocation of such small blocks of memory leads to fragmented memory space further degrading computer performance. The contact container class allocates a large number of contacts when the simulation begins. As the simulations proceeds contacts are checked out and returned from the contact container acting as a type of contact cache.

### 3.5.2 The Dynamic Model

The dynamic model describes those aspects of a system concerned with time and the sequencing of operations—events that mark changes, sequences of events, states that define the context for events, and the organization of events and states. The dynamic model captures control, that aspect of a system that describes the sequences of operations that occur without regard for what the operations do, what they operate on, or how they are implemented. (Rumbaugh et al. 1991)

The intense computational nature of the discrete element method results in an implementation that executes a sequence of operations repeatedly in a nearly endless loop. Once a simulation has begun there is no user interaction. As a result there are no changes of state or external events that the program must contend with, and therefore, only minimal needs for dynamic models.
Two areas that are correctly depicted by dynamic models or state diagrams are the contact logic performed when checking two spheres for contact and contact between a sphere and a boundary.

The detailed implementation of the contact determination between two spheres is presented in section 3.1.1 and depicted in the dynamic model in Figure 55. The determination of whether the contact being checked is a previous or a new contact is also included.

Figure 55 Sphere-Sphere Contact Dynamic Model.

Section 3.1.2 details the contact determination between a sphere and a planar convex boundary. The dynamic model for the contact logic is depicted in Figure 56. The steps required for determining contact between a sphere and a boundary are more complicated than that for the contact determination between two spheres.
Figure 56 Sphere-Boundary Contact Dynamic Model.
3.5.3 The Functional Model

The functional model describes those aspects of a system concerned with transformations of values—functions, mappings, constraints, and functional dependencies. The functional model captures what a system does, without regard for how or when it is done. (Rumbaugh et al. 1991)

The key functions of the discrete element method are (1) the calculation and application of forces acting on and between discrete elements, and (2) the movement of the discrete elements in response to the applied forces. The functional model, shown in Figure 57, illustrates the data transfer between the functions in the discrete element method.
Figure 57 Discrete Element Functional Model.
Data transfers that occur a multitude of times are drawn with thick arrows. In Figure 57, the contact calculation with Sphere A will involve multiple Discrete Element B’s and associated contacts. The specific data transferred for a Discrete Element B, that is a Sphere is illustrated in Figure 57. A similar diagram could be drawn when Discrete Element B is a boundary.

The Contact Container and DE Container are described in the object model, section 3.5.1. The circles represent actual functions that perform calculations on the data. The Calculate Gravity and Add Forces functions are suitably described by their naming. The Update Velocities and Positions function could implement one of several numerical integration techniques, however, the explicit central difference method, detailed in section 3.3, is implemented in this work. The Calculate Contact Force is further defined by drawing a more detailed functional model as shown in Figure 58.
Figure 58 Contact Functional Model.
The data transferred between functions is more refined in the detailed contact functional model. A similar functional model could be drawn for the contact calculation between a sphere and a boundary.

3.6 Computer Technologies

The systems that can be modeled with the discrete element method are limited by the available computer technology. Limiting factors are processor speeds, memory, storage, and visualization and multimedia capabilities. Fortunately these technologies are rapidly developing, increasing the capabilities of the discrete element method. During the time frame of this thesis (1994-1997) incredible technological advances have occurred. These advances include compact disc recordables (CD-R), the World Wide Web, standardized three-dimensional graphics, affordable multiprocessor computers, and color inkjet printers. The technologies becoming available include ISDN access to the Internet, three-dimensional viewing devices, digital video discs (DVDs), desktop video conferencing, and color laser printers. A disadvantage of this rapid growth of computer technology is deciding which technology to use for software development.

In this section the computer equipment used for the discrete element simulations in this thesis is presented. There are also some “predictions” about advances that will be made in the next year. Cundall (1974) includes a discussion of the computer technology at that time. This section provides another “marker in history.” The technology discussed herein, is available and affordable for a small consulting or research group. Application of this upper limit on the technology used for software development and applications insures that the discrete element simulations are also economically viable.

The discrete element sublevel caving simulations presented in this thesis take hours to days of CPU run time, depending on the length of the simulation and the number of discrete elements used. Because of these long run times a simulation is either run from the command line or with a simple user interface. Information concerning the progress of a simulation is
output to the user interface or the command line. The discrete element method is implemented in C++ and runs on Unix and Microsoft Windows machines.

During the duration of this research discrete element simulations have been performed on the following machines:

1. Silicon Graphics Inc. (SGI) Indigo Extreme with 64 Mbytes of memory
2. 100 MHz Intel Pentium with 32 Mbytes of memory running Windows NT 3.51
3. 64 Processor Inmos Transputer
4. 133 MHz Intel Pentium with 32 Mbytes of EDO memory running Windows 95
5. Dual 200 MHz Pentium Pro with 64 Mbytes of memory running Windows NT 4.0

The discrete element implementation and limitations on the Inmos Transputer are discussed in section 3.4. The switch from the Unix (SGI) to Windows (Intel) machines was driven by cost, computer access, and development tools. The Engineering Division at Colorado School of Mines now has an 8 processor SGI Power Challenge. The discrete element method has not been implemented on this machine because of the lack of a C++ compiler for the multiple processors. The dual 200 MHz Pentium Pro machine is running Microsoft Windows NT Workstation 4.0 which supports 1 or 2 processors and is optimized for single users. Microsoft Windows NT Server 4.0 supports up to 32 processors and is optimized for multiple users. Both processors of the machine are not yet being fully utilized. Once the discrete element implementation is fully multithreaded it will take advantage of all of the available processing power. The cost of the dual 200 MHz Pentium Pro is broken down in Table 11.
<table>
<thead>
<tr>
<th>Item</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 x 200 MHz Intel Pentium Pro</td>
<td>$1,323</td>
</tr>
<tr>
<td>Mother Board</td>
<td>$499</td>
</tr>
<tr>
<td>64 Mbyte memory</td>
<td>$466</td>
</tr>
<tr>
<td>1.6 Gbyte Hard Drive</td>
<td>$256</td>
</tr>
<tr>
<td>Diamond FireGL Graphics Card 8MB VRAM + 8MB DRAM</td>
<td>$859</td>
</tr>
<tr>
<td>8X CD-ROM</td>
<td>$111</td>
</tr>
<tr>
<td>16 bit sound card</td>
<td>$80</td>
</tr>
<tr>
<td>Keyboard</td>
<td>$21</td>
</tr>
<tr>
<td>Mouse</td>
<td>$26</td>
</tr>
<tr>
<td>Case</td>
<td>$78</td>
</tr>
<tr>
<td>CPU Fans</td>
<td>$38</td>
</tr>
<tr>
<td>1.44MB Floppy Disk</td>
<td>$24</td>
</tr>
<tr>
<td>17” Monitor</td>
<td>$701</td>
</tr>
<tr>
<td>Microsoft Windows NT 4.0 Workstation</td>
<td>$155</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$4,637</strong></td>
</tr>
</tbody>
</table>

As this technology improves the cost of multiprocessor computers will continue to decrease. This market segment is being driven by computers being used as network servers.

The amount of data generated in a discrete element simulation is immense. Snapshots of the particle configuration is output several times per second. For the simulations in this thesis, data is output 15 times per second. The data output for each discrete element includes location, velocity, mean stress, radius, and density. The location and velocity of the discrete elements are vectors. Values are saved to a data file in binary format. Using binary file format requires significantly less space than an ASCII file. The extra precision of the doubles used during the simulation is not needed for the post processing, therefore, the data is output as floats. For a twenty second simulation with 20,000 discrete elements the amount of data saved is calculated as:
\[
\frac{9 \text{ floats}}{\text{element}} \times \frac{4 \text{ bytes}}{\text{float}} \times 20,000 \text{ elements} \times \frac{15 \text{ frames}}{\text{second}} \times 20 \text{ seconds} = 216 M\text{Bytes (3.123)}
\]

This amount of data can easily be saved on a hard drive. Typical sizes of hard drives today are 1.0 - 2.0 GBytes. However, if several different simulations are saved, hard drive space quickly runs out and it is necessary to back-up the data.

There are some different options to backing up the data. The least expensive is to backup onto a tape drive. There are several major disadvantages to this including: (1) the data on the tapes must be first restored to the hard drive before it can be accessed, (2) tapes are relatively slow, and (3) tapes are not easily transferred between different types of machines. The preferred alternative is to create CD-ROM's of the data. Compact Disc Recordable (CD-R) drives are available today for around $800. Each CD can hold 650 MB of data and cost approximately $7.50 a piece. The data on the CD can be read by any computer with a standard CD-ROM player and does not need to be restored to the hard drive. The speed at which data on the CD can be accessed is also fairly fast. The disadvantage of the recordable CD-ROM is that it is a "write once read many" media. CD-ROM's that overcome this disadvantage will be available in the near future. Another major technical advancement that will be available later this year is Digital Video Discs (DVDs) that hold a full length feature film or 4 GBytes of data. The initial success and demand for the DVD technology should come from the movie industry.

To visualize the discrete element simulations the OpenGL graphics library is used. (Neider et al. 1993; Rogelberg 1992) OpenGL is the successor to gl from Silicon Graphics Inc. which is used by Mathews (1994) and O'Connor (1996) to present discrete element results. OpenGL is available on most Unix environments and ships with Microsoft Windows NT and 95. Using OpenGL is straightforward. The location and properties of the discrete elements are specified, the lighting conditions are input, and finally the viewpoint and viewing direction are given. OpenGL handles the three-dimensional graphic calculations and renders the scene. A FireGL graphics card from Diamond Multimedia is used in this research. The card has a dedicated chip for doing OpenGL graphics which greatly improves graphics performance.
Relating to OpenGL is the Virtual Reality Modeling Language (VRML) that is being used on the World Wide Web to depict three-dimensional worlds. There may be some application of VRML to discrete element modeling.

Another three-dimensional library called Direct 3D is being offered by Microsoft. Direct 3D is targeted at the computer game market and provides efficient performance by directly communicating with the graphics hardware. Direct 3D is available for Windows 95 but is not yet available for Windows NT. How Direct 3D will impact the popularity of OpenGL is not yet clear.

There are numerous medium range two-dimensional and three-dimensional graphics cards available on the market today. Demand for these cards is being driven by computer games, World Wide Web surfing, and computer videos. Intel has also recently released a new technology with their Pentium chips called MMX that is supposed to improve a computer systems multimedia performance. Clearly these technological advances will also improve the performance of discrete element simulations.

Even with these accelerated three-dimensional graphics cards each snapshot of a discrete element simulation can take several seconds to render. This slow display rate makes it difficult to fully observe and appreciate the flow of material in a simulation. To overcome this technological limitation, computer videos are created. Computer videos are created by capturing a sequence of snapshots and playing them back in rapid succession. Several digital video formats are available including; MPEG, Audio-Video Interleave (AVI), and QuickTime. AVI videos of the large scale sublevel caving simulations in this thesis have been made. The AVI format is chosen because of the programming tools included with Microsoft Visual C++ 4.0. The digital format eventually used for feature films recorded on DVDs should define which video format is the most “standard” and should also be used for recording discrete element simulations.

An exciting technology now available, that has not yet been used to view discrete element results is stereo vision. A visor that has small video screen for each eye is available for $600 from Virtual IO. The visor is shown in Figure 59.
Figure 59 Virtual IO Visor.

The stereo vision technology will become more prevalent as it develops in the computer game market.
Chapter 4

MODELING SUBLEVEL CAVING WITH THE DISCRETE ELEMENT METHOD

A numerical model based on the Discrete Element Method (DEM) has been developed to aid in the study of sublevel caving. By using this new tool it is hoped that a better understanding of what helps and what hinders the performance of the ore recovery can be gained. This section includes a description of the input and boundary conditions of the model and outputs from the sublevel caving simulations.

In the sublevel caving DEM model, pieces of ore and waste are modeled with a system of spherical shaped bodies. The bodies can interact with other bodies and with the boundaries representing the surrounding solid rock.

4.1 Defining the Boundary Conditions, Geometry, and Contact Parameters

The boundary conditions in a discrete element model are analogous to the actual geometry. The geometry used is for a height of 27 m between sublevels, 25 m between drifts, a 3 m burden, 80° ring inclination, and the outside ring holes drilled at 60°. The drifts are 7 m wide and 5 m high. The actual geometry is input into the discrete element model with three-dimensional coordinates using a data file. The sublevel caving geometries for each step are included in the following section. Only half of the ring is modeled -- assuming symmetric flow.

The coefficient of friction between all of the particles and the boundaries is set at 0.3. This value is chosen as an initial estimate of an appropriate friction value. The coefficient of friction with the smooth symmetry boundary in the model is set to 0. During the model creation steps (1-3) the friction has been turned off.
A maximum particle overlap of 10% is used in the sublevel caving simulations. The derivation of the resulting contact stiffness is presented in section 3.2. A time step of 10% of the critical time step is used to ensure numerical stability. The coefficient of restitution is set to 20% in the sublevel caving simulations. The damping between discrete elements and friction are the only mechanisms for energy dissipation in the system.

The particles are prevented from rotating in simulation steps 4-6. This condition was required to ensure the system settled in a "reasonable" length of time. In the recommendations of further research it is suggested that the influence of restraining various percentages of the particles from rotating be studied.

4.2. Modeling Steps of the DEM Sublevel Caving Simulation

The setup procedure required to create a discrete element simulation is a complex multi-step process. Specific steps and custom subroutines must be written for each new application. A large amount of time has been devoted to developing and refining these steps for the sublevel caving simulation. This setup procedure is significantly different than for continuum methods like the Finite Element Method. In discrete elements, the initial conditions must be constructed in a similar manner to how they are created in nature. These conditions are required because there is no unique way to place all the discrete elements in the model. In the sublevel caving simulation, the ore and waste discrete elements are randomly located and allowed to settle to a steady state equilibrium condition before they are blasted and subsequently mined out. It is obvious that the initial configuration of the particle system will influence the simulation results. In the current investigation it is assumed that the influence of variations in the initial particle configuration on the overall material flow results are small.

Six basic steps are used to setup the simulation and extract a single ring of ore. These steps can be expanded to model several rings. The steps are as follows:

1. Generate the ore in the ring.
2. Reduce the pressures on the ore in the ring by removing some of the ore particles.
3. Generate the waste.
4. Allow the waste and the ore to settle to a steady state; remove the retaining wall in the drift allowing the waste to flow into the drift; and remove the boundaries separating the ore and waste allowing them to settle together.

5. Blast the ore.

6. Muck out the ore and the waste.

Each of these steps are described in detail in the following sections. It is conceded there are other ways of performing each of these steps.

4.2.1 Step 1 - Generating the Ore in the Ring

In step 1 the ore bodies are generated in one ring. The ring geometry is shown in Figure 60. The slanted boundary on the mid level drift is used to prevent ore particles from remaining on this ledge while settling.

Because it is difficult to fill this complex ore geometry explicitly with ore particles, a box enclosing the ring is filled with randomly located ore bodies. The bodies inserted have a radius of 0.381 meters and a density of 4600 kg/m$^3$. The box is filled until room for an additional particle cannot be found after approximately 20,000 attempts. The model is filled with 7189 bodies as shown in Figure 61.

The radius of all the ore particles is increased to 0.48 m to fill void space. With zero friction between the bodies the model is run for 5.0 seconds to equilibrium. Bodies which are not in the ring geometry fall out of the modeling space and are deleted. The model now contains 1173 bodies.

At the end of step 1 the ore particles are physically squeezed into the ring. The pressures acting on the ore particles at the end of step 1 are shown in Figure 62. The color scale limits represent the maximum and minimum mean stresses acting on the particles at this time step.
Figure 60 Ring Geometry for Step 1.
Figure 61 Density of Randomly Generated Ore.
Figure 62 Number of Particles and Pressures at the End of Step 1.
4.2.2. Step 2 - Reducing the Pressures on the Ore Particles

The overall objective of this step is to relieve the high stress generated in step 1 that were required to tightly pack the ore bodies together. Particles are removed from the ore ring until the maximum hydrostatic stress in the ore ring is essentially unchanged. The resulting geometry is a closely packed particle system in a relatively low state of stress.

The geometry used in step 2 is the same as step 1 except the angled shelf on the upper drift is no longer needed and is removed. The geometry for step 2 is shown in Figure 63. There is zero friction during this step and the particles are free to rotate.

At the end of step 1 there are 1173 particles in the ring and the maximum pressure on a particle is 24527 kPa. The high pressure at the end of step 1 results from the particles being squeezed into the available volume. Step 2 reduces the pressures acting on the ore particles in the ring by removing some of the ore particles.

If one particle is removed, the maximum pressure in the system decreases. Removing another particle continues to decrease the pressure. Pressure will continue to decrease with the removal of additional particles until a point where the particles are no longer squeezed into the volume. After that point, removing additional particles does not significantly affect the maximum particle pressure. When this state of stress is achieved step 2 is complete. It is desired that the number of ore particles in the model is the same number where the particles are not squeezed in the volume.

Three methods have been tried to determine how many particles should be left in the ore ring. The first method tried was to remove a number of particles at the start of the simulation, allow the particles to redistribute themselves and settle, then calculate and find the maximum pressure acting on a particle. Several simulations were run with different numbers of particles removed. The results were unclear and varied depending on where in the ring the particles were removed from.

The second and third methods are similar. One particle is removed, the system is allowed to settle, the maximum pressure acting on a particle is recorded, and the process is repeated with another particle being removed. The system is defined as being settled when the
total kinetic energy drops below 10 Joules. The difference between the second and third method is the selection criteria for particle removal. In the second method the last particle in the data structure is removed, and in the third method the particle with the maximum pressure is found and removed. The results from these two methods run for 120 seconds of simulation time are shown in Figure 64.
Time: 0.000 sec

Figure 63 Geometry for Step 2.
Figure 64 Maximum Particle Pressure during Last Particle and Maximum Pressure Removal.

Several observations are made from the results in Figure 64:

- When the last particle in the data structure is selected for removal several particles can be removed before the pressure drops significantly.

- When the particle with maximum pressure is removed the maximum pressure acting on a particle in the simulation occasionally increases. The pressure increase results from particles adjacent to the one removed taking up the redistributed load.

- The maximum particle pressure appears to flatten out just below 5 MPa.

- Removing the particle with maximum pressure results in lower pressures than when removing the last particle in the data structure for the same number of particles.

- In the same amount of simulation time (120 seconds), more particles are removed with the last particle removal method than the maximum pressure particle removal method.
This indicates that it takes longer for the particles to settle and reach equilibrium by removing the particle with the highest pressure.

For step 2 of the sublevel caving simulation the method of removing the particle with the highest pressure is used in determining how many particles to remove. Nineteen particles are removed during a simulation time of 45 seconds. The maximum pressure acting on the remaining 1154 ore particles is 4839 kPa as shown in Figure 65. This pressure is consistent with an approximate calculation of the vertical stress due to 54 m of overburden in the ore model geometry. This is 19.7% of the initial maximum pressure.
Time: 44.933 sec

Ore: 1154  Waste:  0

Figure 65 Number of Particles and Pressures at the End of Step 2.
4.2.3 Step 3 - Generating the Caved Waste Material

In step 3 the caved material is added. This is done in a similar matter as the ore in step 1. The geometry used in this step is shown in Figure 66. All of the surfaces in this step have zero friction and there is zero friction between the particles.

A box surrounding the model is randomly filled with 20000 waste particles with a diameter of 0.5 meters and a density of 2700 kg/m³ as seen in Figure 67. The total number of particles in the system is 21154.

The model is then run for 5.0 seconds to allow the waste particles to settle around the ore particles. Waste particles outside the bounds of the caving geometry fall out of the modeling space and are deleted. At the end of step 3 the model contains 1154 ore particles and 16945 waste particles as shown in Figure 68.

The pressures on the particles at the end of this step are shown in Figure 69. The particles have not completely settled into static equilibrium at the end of step 3. The speeds of the particles are shown in Figure 70.
Figure 67 Density of Ore and Randomly Located Waste.
Figure 68 System at the End of Step 3.
Figure 69 Particle Pressures at the End of Step 3.
Figure 70 Particle Speeds at the End of Step 3.
4.2.4 Step 4 - Settling and Preparing the Ore and Waste Material

The objective of step 4 is to remove some of the kinetic energy in the system, allow the waste material to flow into the drift, and to allow the ore and waste to come into closer contact by removing the boundaries separating the ore and the waste. Allowing the ore and waste to settle into closer contact prior to the blasting minimizes the influence of how tightly the ore particles are squeezed into the ring. The procedure, of allowing the ore and waste to come into contact, complements the methodology of step 2, Reducing the Pressures on the Ore Particles.

Friction is present between the particles and between the particles and the boundaries. This is the first step where friction is used. A friction coefficient of 0.3 is used between all of the materials. The friction coefficient on the smooth symmetry plane in the model is kept at 0.0. The particles are restrained from rotating in this step.

The geometry for step 4 is the similar to the geometry of step 3. The model height is decreased and the boundaries defining the drift are added. The intent of the first five seconds of simulation time is to remove some kinetic energy from the simulation. Figure 71 shows the total system kinetic energy during the step 4. For the first 5 seconds of simulation time the plug in the drift is present as seen in Figure 72.
A plug is placed in the drift to keep material from flowing into the drift when the ore and waste are created. The plug is removed in step 4 after 5 seconds of simulation time allowing the waste material to flow into the drift. The model geometry with the drift plug removed is shown in Figure 73.

A boundary is used during the creation of the ore and waste to separate the two materials. The boundary between the ore and the waste creates a slight separation of the ore and the waste that is not present in the mine. To allow the ore and waste particles to settle together the boundaries separating the ore and the waste are removed in step 4 after 10 seconds of simulation time. The model geometry with the boundaries separating the ore and waste removed is shown in Figure 74. Removing the boundary separating the ore and the waste also ensures that the ore particles are not squeezed into the ring.

The model is run for an additional 10 seconds after the boundaries separating the ore and waste are removed. Figure 75 shows the speeds of particles after 20 seconds of simulation time. The pressures acting on the particles at the end of step 4 are shown in Figure 76.
Time: 0.000 sec

Figure 72 Geometry at the Beginning of Step 4.
Time: 5,000 sec

Figure 73 Step 4 Geometry with Drift Plug Removed.
Figure 74 Step 4 Geometry with Boundaries Separating the Ore and Waste Removed.
Figure 75 Particle Speeds at the End of Step 4.
Figure 76 Particle Pressures at the End of Step 4.
Time: 19.933 sec

Ore: 1154  Waste: 16945

Figure 77 Ore and Waste Particles at the End of Step 4.
4.2.5 Step 5 - Blasting the Ring

The ore in the ring is blasted in step 5. The geometry of the step is similar to the geometry of the last portion of step 4, except that the boundary at the bottom of the ring of ore is removed as shown in Figure 78.

The blasting is simulated by increasing the radius of all of the ore particles from 0.48 m to 0.49 m. The size of this increase is arbitrarily chosen in this analysis. There is no change to the size of the waste particles.

Figure 79 shows the velocity wave traveling through the material at 0.067 seconds and Figure 80 shows the pressures acting on the particles during the blast. The simulation is run for 5 seconds. The speeds and pressures of the particles at the end of the simulation are shown in Figure 81 and Figure 82 respectively.

The ore in the bottom of the ring flows into the drift. The ore and waste particles are shown in Figure 83.
Figure 78 Part 5 Model Geometry.
Figure 79 Particle Speeds During Blasting.
Figure 80 Particle Pressures During Blasting.
Figure 81 Particle Speeds at the End of Step 5.
Figure 82 Particle Pressures at the End of Step 5.
Figure 83 Ore and Waste Particles at the End of Step 5.
4.2.6 Step 6 - Mining the Material

The ore and waste are mined out in step 6. To simulate the mining process a number of particles in the drift are removed, the system of particles are allowed to re-settle and the process repeats. To accurately simulate the mining behavior the particles must be allowed to settle before taking the next scoop. The total kinetic energy of the system is monitored to determine when to take a scoop. When the total system kinetic energy drops below 100 Joules the system is considered settled and a scoop is taken. The total system kinetic energy for the 60 second simulation performed in step 6 is shown in Figure 84.

![Graph showing total kinetic energy during mining in Step 6.]

Figure 84 Total Kinetic Energy During Mining in Step 6.

A scoop of material is taken by removing a number of particles. In the simulation presented in this section 20 particles are removed each scoop. The particles that are closest to the end of the drift are selected to be removed. The selection process is illustrated in Figure 85.
The model geometry for step 6 is the same as the geometry from step 5 and is shown in Figure 86. The ore and waste particles just before and after the first scoop taken at 42.35 seconds are shown in Figure 87 and Figure 88 respectively. The particle pressures just prior to the scoop are shown in Figure 89 and 0.5 seconds after the scoop is taken are shown in Figure 90. The particle speeds just prior to and 0.5 seconds after the first scoop are shown in Figure 91 and Figure 92 respectively.

Resulting from the time required for the particles in the system to return to steady state, only one scoop is mined in the 60 second simulation. The ore and waste particles at the end of the simulation are shown in Figure 93. The particle pressures are shown in Figure 94.
Time: 0.000 sec

Figure 86 Geometry for Step 6.
Figure 87 Ore and Waste Particles Prior to Scoop.
Figure 88 Ore and Waste Particles Immediately Following Scoop.
Figure 89 Particle Pressures Before Scoop Taken.
Figure 90 Particle Pressures 0.5 Seconds After Scoop is Taken.
Figure 91 Particle Speeds Before Scoop Taken.
Figure 92 Particle Speeds 0.5 Seconds After Scoop is Taken.
Time: 59.933 sec

Density (kg/m^3)

4600

2700

Ore: 1137  Waste: 16942

Figure 93 Ore and Waste Particles at the End of Step 6.
Time: 59.933 sec

Figure 94 Particle Pressures at the End of Step 6.
Chapter 5

CONCLUDING REMARKS AND FURTHER WORK

This chapter describes the conclusions drawn from this research and outlines suggestions for further work.

5.1 Conclusions

An advanced DEM based computational methodology has been developed for the analysis of the large scale sublevel caving mining method used by LKAB in their iron ore mines. This methodology has been successfully implemented and applied to a specific mining simulation. Initial quantitative comparisons between the simulation results obtained from the three-dimensional DEM model and the in-situ test data gathered by LKAB at the Kiruna mine illustrate the potential of this new methodology to accurately model the physical sublevel caving mining process. Note, the authors participation in the in-situ testing during the summer of 1995 was intended to insure the accurate incorporation of the mining process and mechanics into the discrete element numerical simulation. This research shows that the discrete element method can be used to model sublevel caving with a sophisticated three-dimensional numerical simulation.

The present discrete element sublevel caving model is a complex multi-step modeling method that attempts to incorporate the mechanical and material behaviors found in the large scale sublevel caving mining process. These features include:

1. The preparation of the ore/waste geometry and the initial mining conditions.
2. Blasting of the ore ring simulated by instantaneously increasing the size of the elements in the ore region.
3. The material is mined by allowing the system of particles to settle to near equilibrium and then removing a number of particles. The following scoop is taken only after the system has resettled.

The size and complexity of the large scale sublevel caving DEM simulation are demonstrated clearly by the scarcity of similar engineering applications cited in current technical publications. To model such a large system of discrete elements in three-dimensions an advanced special purpose three-dimensional discrete element simulation technique has been designed. This new discrete element implementation can be used to continue the modeling of large scale sublevel caving and other material flow problems.

The current implementation of the discrete element based computational methodology is used on a high end personal computer workstation that is comprised of widely available affordable computer technology. This implementation was made because it represents the constraints that many engineering analysts in industry usually have to work within.

To computationally simulate the large scale sublevel caving mining in three dimensions in a reasonable length of CPU time several advances to the discrete element method have been developed in this thesis. The specific advances include:

1. The three-dimensional implementation of the discrete element method.
2. A new, more efficient, contact search algorithm based on the traditional grid method.
3. Object oriented programming techniques to elegantly document and implement the discrete element method.
4. A parallel implementation of the discrete element method.
5. A detailed derivation of the contact determination for a system of spherical particles contained within a general shaped region consisting of multiple planar surfaces.
6. Three-dimensional visualization of the discrete element model results to view the material flow.

The three-dimensional discrete element model has not been implemented in parallel at present. The parallel implementation was originally done for the two-dimensional implementation on a 32 processor Transputer. The parallel technology was not included in the
three-dimensional implementation because of the lack of an Object Oriented C++ compiler for the available parallel machines and the marginal performance gain of the Transputer over high end personal workstations available to the author.

5.2 Future Research

The discrete element method for three-dimensional analysis is very computationally intensive and only now with the increasing computer speeds becoming feasible. Simulating 5 seconds of sublevel caving with 18000 particles, requires approximately 24 hours of CPU time and generates 200 Mbytes of data. One suggested course of further research would involve a more focused investigation of some sub-problems in the sublevel caving mining process that require shorter CPU simulation times.

An example sub-problem of the sublevel caving mining process is the refinement of the blasting step. The blasting process occurs over a relatively short period of time that can be reasonably handled with available computer technology. When faster computers are available the experience and further developments from the blasting simulations could be applied to the steps that take long periods of simulation time such as the ore extraction step. The suggested refinements of the blasting step are:

1. Compare the total energy rise in the DEM model with the energy of the explosive.
2. Plot stress contours in the ring before and after the blast.
3. Vary the time that each hole detonates.
4. Vary the blasting of each hole with the velocity of detonation of the explosive.
5. Investigate the effects of changing the blast design.
6. Effects on the stress contours of slashing the drifts wider.

A second example sub-problem is the modeling of the LHD taking scoops of material from the muck pile. The specific questions that should be investigated include:

1. How many particles are removed before large failure of the muck surface resulting in major material flow?
2. Is the flow dependent on the number of particles that are removed each scoop? For example: If slope failure occurs after 40 particles are removed, does removing 2 scoops of 20 particles or 4 scoops of 5 particles produce different overall results?

The CPU time required to model the scoops taken by the LHD could be significantly improved by modeling only the lower half or third of the ring thereby reducing the number of discrete elements in the simulation.

A future application of this type of discrete element methodology could be as a computational experimental laboratory for improving the general understanding of material flow. A series of simulations where material parameters can be varied in a controlled setting would provide insight into the flow behavior of granular media. Example material parameters that can be easily controlled are:

1. Particle size distribution – slight size distribution and major size distribution.
2. Influence of particle rotation – varying the percentage of particles in the system free to rotate from 0 to 100%.
3. Use of a randomized normal on a particle surface to simulate a form of particle roughness.
4. Use the calculated normal from a superquadric on a sphere to give the sphere pseudo edges to inhibit the unrealistically high rotation of spherical particles.
5. Bind spheres together to form irregular shaped objects.

General improvements to the current three-dimensional discrete element model could include:

1. Inclusion of the heap sort contact searching method described by O'Connor (1996).
2. Multi-threaded three-dimensional grid and heap sort contact searching algorithms.
3. Spheres bound together to form irregular shapes.
4. Incorporate general shaped elements such as ellipsoids and superquadrics.
5. Couple the discrete element method with a finite difference porosity based fluid flow model for the simulation of multiphase particle-liquid and particle-gas systems.
Chapter 6

REFERENCES CITED


Cundall, P. A. Accurate simulation of plastic flow without introducing interfaces for singular points, Itasca Consulting Group, Inc.


Cundall, P. A. (1974). Rational design of tunnel supports: A computer model for rock mass behavior using interactive graphics for the input and output of geometric data. Omaha, Nebraska, Missouri River Division, Corps of Engineers.


