DEVELOPMENT OF AN APPLIED ELEMENT BASED TOOL FOR PREDICTION OF COLLAPSE BEHAVIOR OF CONCRETE DAM MONOLITHS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HAMIDULLAH HASSAN ZADA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CIVIL ENGINEERING

SEPTEMBER 2020

Approval of the thesis:

DEVELOPMENT OF AN APPLIED ELEMENT BASED TOOL FOR PREDICTION OF COLLAPSE BEHAVIOR OF CONCRETE DAM MONOLITHS

submitted by **HAMIDULLAH HASSAN ZADA** in partial fulfillment of the requirements for the degree of **Master of Science** i**n Civil Engineering, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Ahmet Türer
Head of the Department, **Civil Engineering** _____

Prof. Dr. Yalın Arıcı
Supervisor, **Civil Engineering, METU** _____

Examining Committee Members:

Prof. Dr.Kağan Tuncay
Civil Engineering, METU _____

Prof. Dr. Yalın Arıcı
Civil Engineering, METU _____

Prof. Dr.Özgür Kurç
Civil Engineering, METU _____

Assoc.Prof. Dr. Alper Aldemir
Civil Engineering, Hacettepe University _____

Assist.Prof.Dr. Bekir Özer Ay
Department of Architecture, METU _____

Date: 24.09.2020

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Hamidullah Hassan Zada

Signature:

ABSTRACT

DEVELOPMENT OF AN APPLIED ELEMENT BASED TOOL FOR PREDICTION OF COLLAPSE BEHAVIOR OF CONCRETE DAM MONOLITHS

Hassan Zada, Hamidullah
Master of Science, Civil Engineering
Supervisor: Prof. Dr. Yalın Arıcı

September 2020, 107 pages

In this study, conducted with the support of TUBITAK under the grant 116M524, a software framework with a web-based graphical user interface is developed for the investigation of the seismic behavior of concrete gravity dams. Using a modified discrete element technique, one of the primary goals of the developed tool is to simulate extensive cracking on concrete dam monoliths followed by sliding, rocking and loss of stability. The graphical user interface for using this engine is prepared in HTML, JavaScript and CSS languages. The structure and content of the GUI are controlled with HTML, whereas the style and formatting are conducted in CSS. Interaction within the GUI is handled through the JavaScript language. The GUI is set up to provide a simple, stable, and easy interaction with the numerical problem at hand. The meshing for the solid elements and reservoir elements are handled within the software using ray-tracing algorithms among other tools for controlling the geometry of the discretized numerical system. The work is organized as follows: The structure of the code is presented first, with the explanation of options provided in the GUI as well as the modeling technique. Examples of the use of GUI for the modeling of complicated dam geometry and reservoir systems are shown next. The algoritms used for simulation of large displacement problems of elastic bodies are

shown and the methodology for modeling of contact between objects is presented. Finally, the capability of the solution of very large displacement problems is demonstrated using three benchmark problems, free fall/bouncing, rocking and overturning response of a square block. In summary, this study presents the details of the GUI implementation as well as the demonstration of the capability of the simulation of large displacement problems with the developed tools.

Keywords: Finite Element Method, Applied Element Method, Dam – Reservoir Systems, Frequency Estimation

ÖZ

BETON BARAJI MONOLİTLERİNİN ÇÖKME DAVRANIŞLARININ TAHMİNİ İÇİN MODİFİYE UYGULAMALI ELEMAN YAZILIMI GELİŞTİRİLMESİ

Hassan Zada, Hamidullah
Yüksek Lisans, İnşaat Mühendisliği
Tez Yöneticisi: Prof. Dr. Yalın Arıcı

Eylül 2020, 107 sayfa

Tübitak tarafından 116M524 kodlu proje altında desteklenen bu çalışmada beton ağırlık barajların sismik davranışını inceleyen web tabanlı arayüze sahip bir yazılım platformu geliştirilmiştir. Geliştirilmiş bir ayrık eleman tekniği kullanılan bu yazılım ile baraj gövdelerinde ciddi çatlak ilerlemeleri ile başlayıp kayma, sallanma ve devrilmeye kadar giden davranışların simülasyonu hedeflenmektedir. Yazılımın grafiksel kullanıcı arayüzü (GKA), HTML, JavaScript ve CSS dillerinde hazırlanmıştır. GKA'nın yapısı ve içeriği HTML ile kontrol edilmekte, tasarımı ve düzeni ise CSS ile yönetilmektedir. GKA ile etkileşim JavaScript dili üzerinden yürütülmektedir. GKA karşılaşılan nümerik problemlerle basit, stabil ve kolay etkileşim sağlanması için hazırlanmıştır. Yapı ve rezervuar elemanlarının ağ yapısının oluşturulması yazılım içersindeki ışın izleme algoritması ve ayrıklaştırılmış nümerik sistemin geometrisini kontrol eden diğer araçlarla yapılmıştır. Çalışma dört bölümde sunulmaktadır: Öncelikle arabirim yazılımının yapısı, kullanılan modelleme teknikleri ve seçenekleri ile birlikte sunulmuştur. Arabirimin karmaşık baraj geometrileri ve rezervuar sistemleri için kullanım örnekleri verilmiştir. Daha sonra elastik objelerin yüksek deplasmanlarda

davranışlarını modellemek için kullanılan algoritmalar sunulmuş, bu objelerin çarpışması sırasında kullanılacak kontak algoritmaları verilmiştir. Son olarak, yazılım altyapısının yüksek deplasmanlarda çalıştığının gösterilmesi için düşen/zıplayan, sallanan ve devrilen bir elastik bloğa dair üç denektaşı çözümü sunulmaktadır. Özet olarak bu çalışma yazılım arabiriminin oluşturulması sırasında kullanılan teknikler ve yüksek deplasman davranış simülasyonlarının başarısını gösteren denektaşı problemi çözümlerini içermektedir.

Anahtar Kelimeler: Sonlu Elemanlar Yöntemi, Uygulamalı Eleman Yöntemi, Baraj – Rezervuar Sistemi, Frekans Hesabı

Dedicated to My Beloved Family

ACKNOWLEDGMENTS

I would like to express my sincere and deepest gratitude to my supervisor Prof.Dr. Yalın Arıcı for his continuous support, insight, and profound knowledge. Without his persistent motivation, advice, and patience, this work could not have been conducted successfully, and its goal would not have been realized. I could not have imagined having a better supervisor and mentor for my master's study.

I wish to immensely thank members of the dissertation committee: Prof.Dr.Kağan Tuncay, Prof.Dr. Özgür Kurç, Assoc.Prof. Dr. Alper Aldemir and Assist.Prof.Dr. Bekir Özer Ay for their time, consideration, and guidance throughout the review of this study.

A very special gratitude goes to the professors whose lectures offered me a precious opportunity to expand my knowledge, specially Prof.Dr.Kağan Tuncay, who always supported and encouraged me through the years of the study.

Also, I would like to pay my special regards and dedicate this milestone to my family, my father, Mohammad Zaher Hasanzadeh, my mother Najibeh Hasanzadeh and my brothers Ali Agha Hasanzadeh and Mahdi Hasanzadeh, whose unfailing support, motivation, and encouragement helped me through the years of study, research and writing this thesis. This journey would have been impossible without their inputs.

Some special words of gratitude go to my dear friend Ahmad Shahab Shakibani, for the wonderful times we shared together and for being a source of support and motivation throughout the years of study.

Also, I gratefully acknowledge the funding provided throughout the completion of this study by The Scientific and Technological Research Council of Turkey under grant number TUBİTAK 116M524.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

A novel tool for the prediction of collapse behavior of concrete gravity dams under seismic loading is developed in-house utilizing and combining many different analytical frameworks. The software includes web-based pre- and post-processors along with a computation engine. The web-based application is developed using HTML, CSS, and JavaScript languages, which are responsible for defining the content structure, presenting HTML elements with proper style and format, and making elements interactive and dynamic, respectively. A JavaScript library (JQuery) and various APIs (WebGL, HTML5 < Canvas> element) are also used at numerous stages during the coding of the software with the goal of obtaining an easy to use, streamlined interface that can quickly generate input for use in the engine in addition to post-processing of the output. The computing engine, which is developed in the Matlab environment, gives users the opportunity to solve static and dynamic mechanical problems in a 2D setting, with the capability of simulation of the rigid body motion of bodies with contact and recontact capabilities. The overall structure of the software is intuitive and straightforward: using the GUI, the shape of the structure is drawn by the user, the required data and properties for the formation of the mathematical model and the solution options are provided. Then, the model output from the interface is utilized in the solver engine to create an applied element model to simulate discrete and extensive cracking on concrete monoliths followed by sliding, rocking, loss of stability, and total collapse behavior in extreme loading scenarios.

This study was mainly conducted in two parts:

1. Graphical User Interface: In the first part of this study, many tools with which a complex geometry can be drawn and discretized into an applied element mesh models along with the procedures and the algorithms used during the development of the available tools was studied.

2. Solver Engine: In the second part of this study, a set of algorithms were implemented and developed to simulate discrete and extensive cracking on concrete monoliths followed by sliding, rocking, loss of stability, and total collapse behavior in extreme loading scenarios.

In the following sections, first, the literature is reviewed for the historical development as well as the application areas of the applied element method, and then the main components used during the GUI development is explained.

1.1 Applied Element Method

In the late 1990s, the Applied Element Method (AEM) was developed at the University of Tokyo by Hatem Tagel-Din with the goal of analyzing the behavior of the structures throughout all the loading stages; elastic behavior, crack initiation and propagation, elements separation and recontact, collision with the ground and adjacent structures until perhaps total collapse. The applied element method can be categorized as a discrete element technique for structural analysis in the general category of rigid body spring models ((RBSM) Kawai, 1977). The method has been adopted by many researchers and used for seismic analysis of structures (Pandey and Meguro, 2004, Raparla and Ramancharla, 2012), sensitivity assessment (Karbassi and Nollet, 2008), geomechanical studies (Ramancharla and Meguro, 2006), blast analysis (Tagel-Din and Rahman, 2006), progressive and total collapse of structures (Sasani, 2008, Elkholy et al., 2003). In this section, firstly, the historical development of the Applied Element Method is discussed, and then the literature is reviewed for the application areas of this method.

The Applied Element Method involves discretization of a continuum by rigid elements connected at interfaces by a number of springs constrained with the motion of the rigid body. In the early development stages, the element size and the number of springs connecting the elements were examined, followed by the verification of the large deformation analysis of structures for dynamic loads (Tagel-Din and Meguro, 2000). Material nonlinearity was introduced using brittle tensile springs, which disappeared after reaching the tensile limit (Tagel-Din and Meguro, 2000). Collision and re-contact of the elements were introduced (Tagel-Din and Meguro, 1999). The modeling of reinforcement in concrete was conducted in Meguro and Tagel-Din (2001) comparing crack schemes and load-displacement relations with the experimental results. The authors also addressed the problems associated with the analysis of cracked structures under cyclic load (Meguro and Tagel-Din, 2001). Large deformation and nonlinear material response were combined in (Meguro and Tagel-Din, 2002), comparing well with theoretical formulations showing the promise of the technique. For overcoming discretization problems, improvements have been proposed: different thicknesses can be assigned to the springs in normal and shear directions, and thus nonrectangular elements can be defined enabling modeling with larger element sizes (Elkholy and Meguro, 2004). This method was first verified by nonlinear geometrical and material methods, and then it was concluded that the collapse of a nine-story steel building under the earthquake load was simulated correctly (Elkholy and Meguro, 2005). Using this method, reinforced and plain concrete beams have been analyzed under static load and columns under cyclic load, and it has been observed that the load-displacement relations and collapse modes have been successfully simulated. Elkholy et al. (2012) further improved this method, making it possible to model non-homogeneous sections with multilayer elements. The Voronoi applied element method was recently proposed by Worakanchana and Meguro (2008) based on the creation of element shapes similar to the Voronoi mosaic. The main advantages of the method are that the element node points can be placed anywhere in the element allowing to concentrate to any desired area and can fit any shape with different element sizes. In addition, nodes can be

placed to create weak areas. Elastic verification was made with a cantilever beam and a circular disc sample, and the results were found to be consistent with the theoretical solutions. After this realization, a crack simulation of a reinforced concrete beam was simulated and the obtained crack diagram was close to the test results. Finally, the method has been validated with large deformation analysis.

Applied element method has been utilized in the simulation of many different phenomena. Pandey and Meguro (2004) examined the behavior of clay-brick masonry walls under horizontal loads. The simulation results were compatible with the experiment in terms of crack diagram, tension distribution, and load-displacement relationship. The crash analysis of the Alfred P. Murrah Federal Building due to the bombing was investigated, and the explosion simulations were found to be realistic. (Tagel-Din and Rahman, 2006). Karbassi and Nollet (2008) examined a six-story industrial masonry building with static pushover analysis. The seismic behavior of the building was estimated by developing the damage probability curves. Sasani (2008) investigated the progressive collapse of the San Diego Hotel. Deformation propagation in the building was evaluated, and the mechanism of redistribution of loads was explained. The controlled failure of a reinforced concrete building was modeled by Lupoae and Bucur (2009). The response of the building to the removal of a column was investigated, and the results of the analysis coincided with the actual collapse records of the building. The gradual collapse of reinforced concrete bridges under earthquake loads was studied by Lau and Wibowo (2010) and the progress of collapse was observed with the displacement behavior of the bridge. As a result of the analysis of a five-story reinforced concrete building, Salem et al. (2011) concluded that the applied element method is an effective, sufficient, and accurate tool for progressive collapse analysis. Dongre and Ramancharla (2012) compared the inelastic behavior of brick-filled and unfilled reinforced concrete frames subjected to horizontal loads. Accordingly, the effect of the brick fill on the load-carrying capacity was revealed.

1.2 Software Components

The developed graphical user interface is based on the HTML, CSS, and JavaScript languages, which are the main components of developing a web-based application. In order to display the combined results of the HTML, CSS, and JavaScript codes, any web browser such as Internet Explorer, Mozilla Firefox, and Google Chrome can be used. Short definitions of the utilized languages, HTML, CSS, and JavaScript are provided below.

HTML, which stands for Hypertext Markup Language, is a tool for defining the content structure of a web page and ordering the browsers how to display the content of the web pages. HTML consists of several elements such as the text header, paragraph, divider, image and text span which are defined by the tags such as <h1>, <p>, , <img> and respectively. Scripting languages such as JavaScript can be embedded in HTML codes. However, most of the time, HTML and scripting languages are kept separate for the sake of simplicity and coherency. HTML5 version of the HTML is used for developing the software. The overall structure of an HTML5 document is shown in Figure 1.1.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <meta http-equiv="X-UA-Compatible" content="ie=edge">
7      <title>Document</title>
8  </head>
9  <body>
10
11 </body>
12 </html>
```

Figure 1.1. Overall Structure of the HTML5 Document

CSS (Cascading Style Sheets) is used for presenting the HTML elements with proper styles. It enhances the way the content of a page display in terms of color and format. Formatting such as defining the margins, border, position, padding, and background color of the elements and defining the size, color, and weight of the fonts and other

decorative functions can be imposed.CSS scripts can be embedded inside the HTML documents. However, the main idea of using the CSS is that the content (HTML) and formatting (CSS) to be presented separately. The document presentation is enhanced by keeping the formatting of a document separate in an external file. CSS allows HTML elements to share the same formatting avoiding repetitive formatting.

JavaScript is the primary language used during the development of the software, making web-based applications interactive and dynamic. It gives functionality to the HTML elements, allowing the pages to react to events, shows the effect of the events, receives input, and gives feedback to the user using the codes working behind it. JavaScript can be embedded inside the HTML documents. However, most of the time, HTML and scripting languages are kept separate for the sake of simplicity and coherency. The 2018 version of JavaScript named "ECMAScript" was used during the development of this software. A basic code editor such as windows notepad, Visual Studio Code, and Atom code editor can be used for JavaScript coding.

1.3 Novelty and Impact

Understanding of the seismic risks on structures and systems and their outcomes is an essential task enabling engineers to come up with preventive and inhibitory solutions toward the risks. In the case of dams, there exist numerous unknowns regarding their performance under large displacement and extreme loading scenarios. Due to the high importance in terms of economy and cost of these massive structures, it is imperative to prevent any damage in the dam body, as due to the continuous nature of the dam structures, loss at one part may put in danger the whole structure. Determination of the performance behavior of the cracked concrete dams is of high importance, as it considerably helps decision-makers to propose the best solution for the existing problems as well as during the design stages.

Due to the practice of the classic techniques for design and analysis of the dam structures, a performance-based approach to provide information regarding the

behavior of the dams has not been developed yet. Domination of the Finite Element Method (FEM) in today's design and analysis practice is perhaps one of the main impediments to this development. FEM fails to reliably simulate discrete, extensive cracking, sliding, loss of stability, or rocking as the essence of this method is based on continuous displacement field and small deformation assumption. Due to the continuous nature of dams, unlike buildings and bridges, component testing of these structures is not an option. Experimental testing of scaled specimens is challenging and costly; common scaling near 1/100 of the original size does not allow a realistic investigation of the problem. Usually, 1/20 or 1/50 scales of dam structures are needed in order to accurately predict their behavior. However, constructing such large-scale models is not cost-effective and generally not possible in a lab environment. Under these circumstances, a tool capable of simulating large displacements of structures under extreme loading scenarios is significantly valuable. The applied element tools are implemented to this end in this study, coupled with a web-based GUI to improve mesh generation for prediction of the response of dam systems. The discrete nature of cracking in this technique provides the opportunity to reliably keep track of crack propagation and element separation as well as large displacement behavior with element contact and collision. The rigid body response of (any) severed part of the dam can be simulated, enabling the modeling of the eventual total loss of stability.

Web-based software has been gaining traction with the use of cloud services enabling calculation at the back-end of servers instead of user’s machines for large scale computation. Decreasing the resource need for the user computer, this approach also enables the software to completely run on a web browser, removing compatibility constraints as well as the licencing issues. A browser-based graphical user interface has been developed in this project facilitating such use; users can access the program on any browser irrespective of the local machine and can perform analysis on remote computers.

1.4 Organization

This study is organized as follows:

The current chapter includes the literature review on the applied element technique, a discussion on the novelty of the work, and provides information on the general organization of this text.

In the second chapter, the methodologies used during the development of the front-end part of the software (Graphical User Interface), along with the procedures for the use of the available tools, are presented. Many benchmark problems are modeled via the GUI to verify the capabilities of the developed applied element tool.

In the third chapter, the solution of the mathematical models in the back-end part of the software (Solver Engine) will be discussed. A Matlab code focusing on the modeling of the structures using GUI data along with the formulation and the procedures used during the coding of this solver will be presented. At the end of this chapter, the simulation results will be compared to the available experimental results and theory.

In the final chapter, a summary of the work is presented, along with the discussion of the results. The limitations of the study and ideas about future work are presented

CHAPTER 2

GRAPHICAL USER INTERFACE

In this chapter, the overall structure of the Graphical User Interface (GUI) developed for the Modified Applied Element Technique will be presented. In addition, the structure of the code and the use of the GUI for modeling of complicated dam geometry and reservoir systems, the methodologies used behind the software's script for the creation of meshing and discretization of models, boundary assignment process, and load application procedures will be discussed in detail. The chapter will be concluded with some sample problems and benchmark cases to demonstrate the capability of the interface. This chapter can be used as a manual/support document for the developed interface as well.

2.1 The Structure of the Graphical User Interface

The software user interface is comprised of the menu bar, input bar, display window, and the status bar. The material properties, boundary conditions, and mesh options boxes are located on the left side while the solver, results, and delete model boxes are located on the right-hand side of the user interface.The coordinates input bar located in the top portion of the interface receives coordinates of the geometry of the model. The geometry of the model is then discretized into discrete elements and finally sent to the analysis engine. The geometry of the model is presented in the display window after drawing, while the coordinates of the model are recorded and shown in the status bar. Properties related to the material of the model, such as the density, thickness, linear, and nonlinear, can be assigned under the "Material Properties" box. "Boundary Conditions" box is used for applying the boundary

conditions and supports of the model under consideration. Mesh properties such as the size of the applied elements for the dam body, the size of the finite elements for the reservoir, and the number of springs connecting the applied elements are assigned under the “Mesh” box. Finally, modifications to the model, such as changes in the boundary conditions, changes in the meshing size of both the dam body and the reservoir, and the model modifications are performed under the “Delete” model box. The overall structure of the user interface is shown in Figure 2.1.

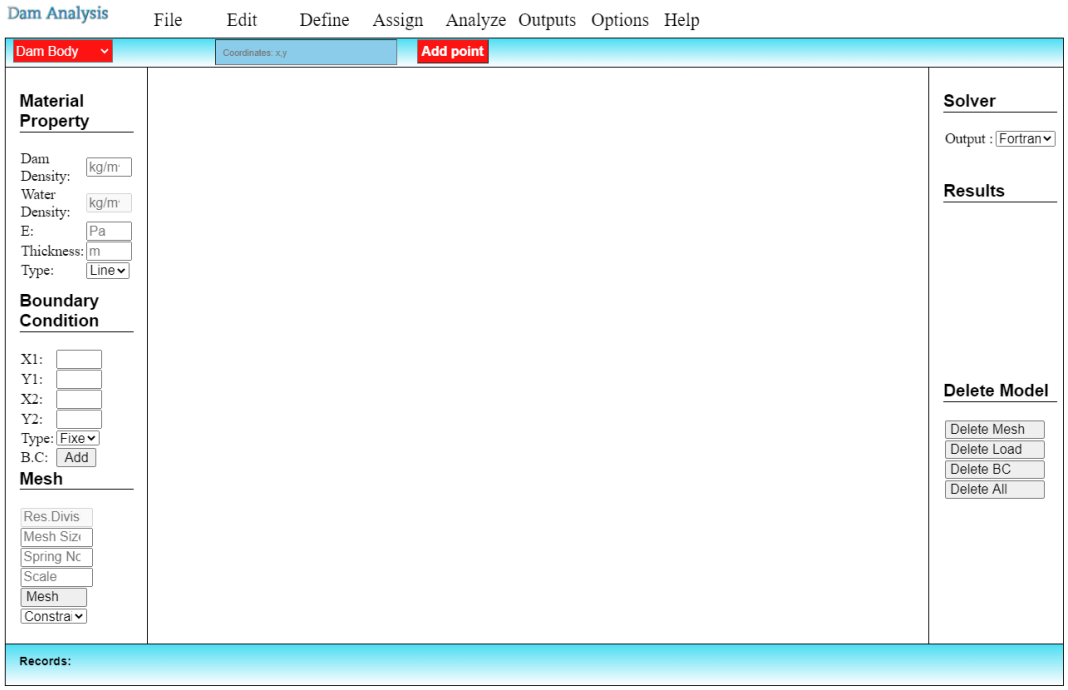


Figure 2.1. The Overall Structure of Software

2.2 Drawing

Many different types of 2D geometries can be drawn quickly via the developed software. Drawings can be created from a wide range of geometries from simple to detailed complex models. In this section, the procedure for creating the geometries of the dam body and reservoir, as well as the methodology used behind the software’s script during the development, are explained. However, before starting, the medium for geometry creation, i.e., the Canvas element, is explained below.

For drawing a shape, much like on physical paper, a digital stage on which the models can be drawn is required. Html 5 canvas element (<canvas>) is the so-called stage utilized for this purpose in this software, which acts as a container for the graphics. This area can be accessed via the JavaScript code enabling the user to create a rendering of 2-D shapes such as line, circle, and polygons. Due to its white color, the canvas element is invisible during the standard display of the window. The canvas element is shown in a green shade for visibility in Figure 2.2.

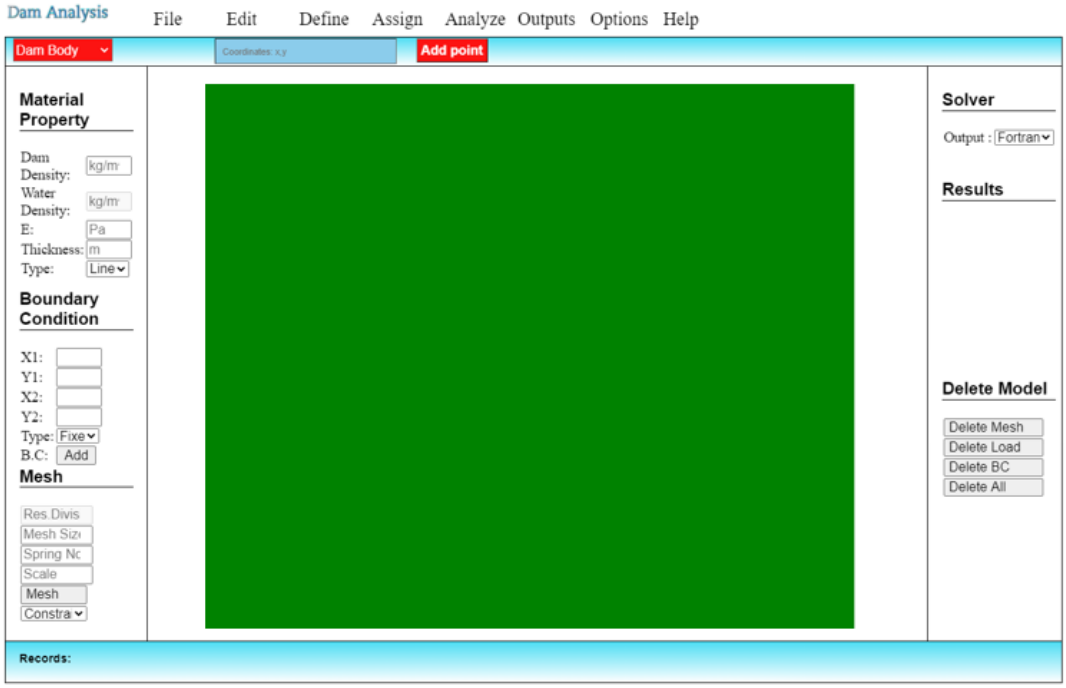


Figure 2.2. Location of Canvas Element in Display Window

2.2.1 Drawing of Solid Geometries

Now that the stage is ready for drawing, the user can create the desired shape. Regardless of the complexity of the model by following the below-mentioned steps, any geometry can be drawn.

1. Set the origin at (0,0) and list all the remaining corners of the geometry either clockwise or counterclockwise in terms of x and y coordinates for each corner, respectively.

2. Insert each corner coordinates as (x,y) format in the coordinates input bar located at the top of the drawing stage.
3. Starting coordinate, which is usually (0,0), should be entered again into the input bar finally to have a closed shape geometry.

Following the three steps as mentioned above, the desired geometry is created. The program recognizes the given corner's coordinates as endpoints of line segments that are connected to each other, making the final geometry. Re-entering the starting coordinate into the input box, the geometry drawn is accepted as a closed shape for further use. No matter how complex the geometry is, it is visualized in terms of connected line segments; any irregular shape can be drawn on the canvas. For simplicity, a 2D square is considered as the geometry of a model to show the drawing methodology using the developed software. The dimensions of the model are given in Figure 2.3.

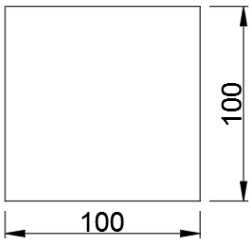


Figure 2.3. Solid Sample Model Dimensions

The corner coordinates of the geometry are considered for drawing the given geometry. In the main software window, all four coordinates are given as input to the box provided on the input bar, as shown in Figure 2.4.

Points	coordinates *(x, y)*
1	*0,0*
2	*100,0*
3	*100,100*
4	*0,100*

Figure 2.4. Coordinates of the Model

Notice that x and y coordinates of each corner are separated with a comma. Therefore, a comma should be used to separate the x and y coordinates while entering the coordinates to the related input box. To complete the drawing of the geometry, the starting coordinate, which is (0,0) in the sample model, should be entered for the second time, so the software considers the drawn model as a closed shape and finishes drawing operation. If the entered coordinates are in the correct format, the coordinates will be shown on the status bar, as shown in Figure 2.5.

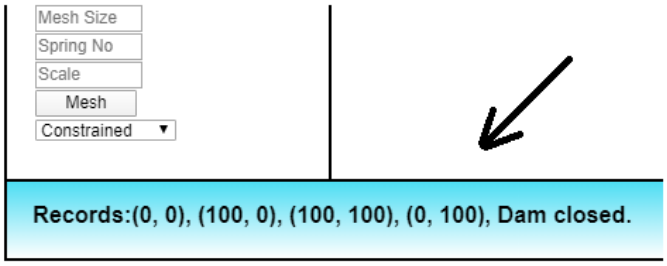


Figure 2.5. Status Bar for the Model

2.2.2 Drawing Geometries with Openings

In the previous section, the methodology for creating solid geometries was explained. In addition to solid shapes, the developed software is capable of creating shapes with openings as well. The methods for forming such features in solid geometries are detailed below.

2.2.2.1 Polygonal Opening

In order to create such an opening to any geometry, first, the desired geometry is drawn following the procedure shown in section 2.2.1. Then the same process is applied for drawing the opening. As long as the provided coordinates for the geometry lie inside the solid geometry, the software considers it as an opening since two solid bodies cannot overlap. For illustration, a square model with an opening at

the center with dimensions shown in Figure 2.6 is assumed. Coordinates of the corners of the opening are shown in Table 2.1.

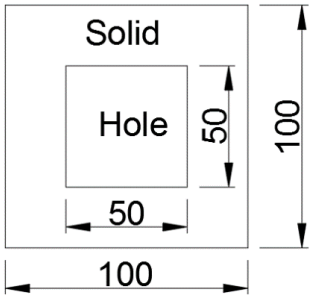


Figure 2.6. Sample Model with Polygonal Opening

Table 2.1 Coordinates of the Opening

Points	*Coordinates*
1	25,25
2	75,25
3	75,75
4	25,75

If the provided coordinates are inside of the solid geometry and are supplied in the correct format (x,y), the software creates the opening and will confirm the operation by updating the status bar, as shown in Figure 2.7.

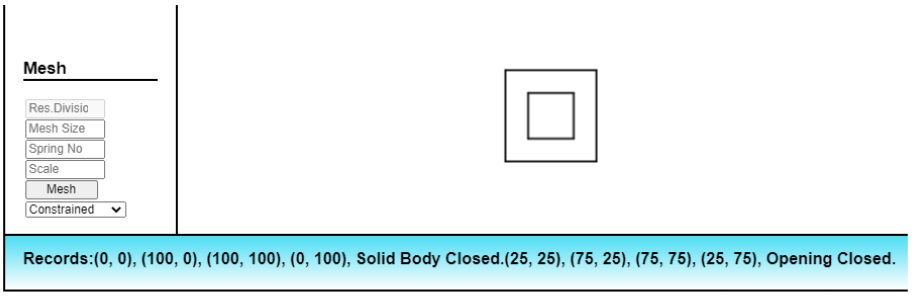
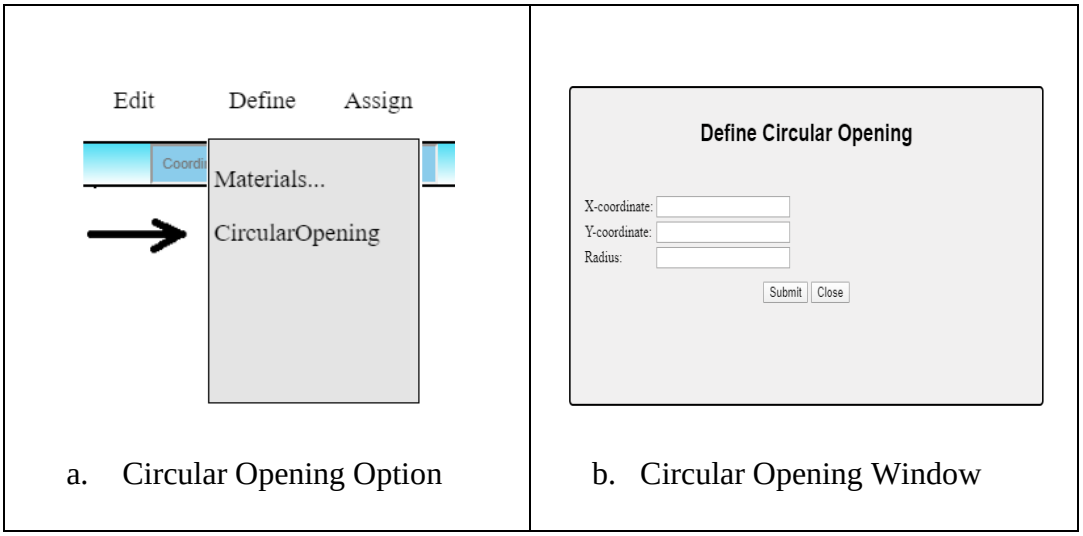


Figure 2.7. Status Bar for the Model with Opening

2.2.2.2 Circular Opening

In this section procedure for creating a circular opening to a solid body is explained. Similar to the polygonal opening, the solid part of the model is drawn. From the "Define" tab in the menu bar, then the "Define Circular Opening" option (Figure 2.8a) is selected, and the related pop-up window will appear (Figure 2.8.b). After providing center coordinates and radius of the circular opening in the pop-up window and clicking the submit button, an opening will be created to the solid body. A sample model with a circular opening shown in Figure 2.9 is created using the interface.



a. Circular Opening Option

b. Circular Opening Window

Figure 2.8. Circular Opening Pop-up Window

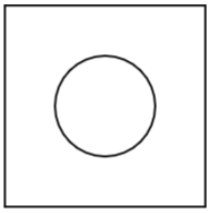


Figure 2.9. A model with Circular Opening

2.2.3 Drawing of the Reservoir

In this section, the methodology for drawing a reservoir next to a solid body is explained. When drawing of the dam part is finished, the value of the click next to the coordinates input box (Figure 2.10) is set to the “Reservoir” option, and then by providing the coordinates of the geometry, the reservoir is constructed.

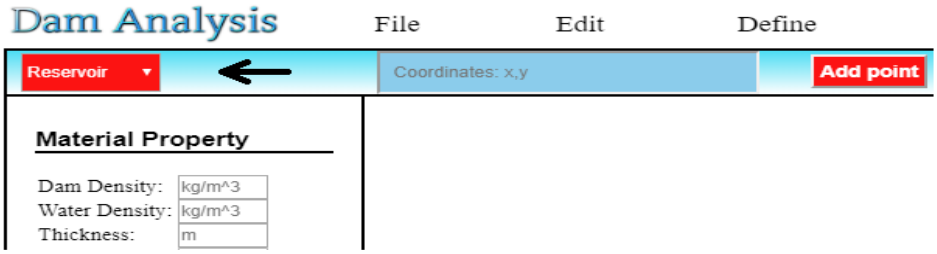


Figure 2.10. Reservoir Selector for Drawing the Reservoir

For illustration purposes, a square with a side length of 100 units as a dam body and a reservoir with a height and width of 80 and 300 units, respectively, shown in Figure 2.11, is considered.

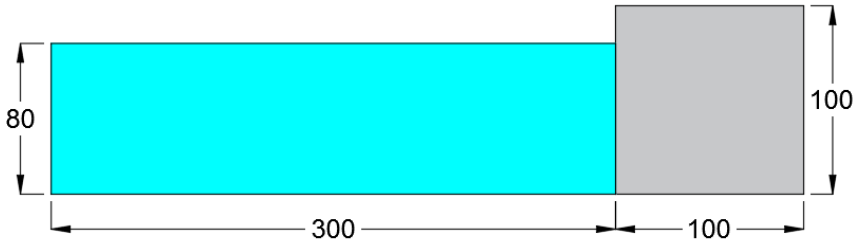


Figure 2.11. Sample Model with Reservoir

First, the value of the click shown in Figure 2.10 is set to “ Dam Body ” and then corner coordinates of the dam body are provided. Then the value is switched to “Reservoir” and reservoir coordinates are inserted accordingly. The coordinates of the model are presented in Table 2.2.

Table 2.2 Coordinates of Dam Body and Reservoir

Dam Body		Reservoir	
Points	*Coordinates*	Points	*Coordinates*
1	0,0	1	0,0
2	100,0	2	-300,0
3	100,100	3	-300,80
4	0,100	4	0,80

Notice that comma should be used to separate the x and y coordinates while entering the coordinates to the coordinates input box. In order to complete the drawing of the geometry, the starting coordinate, which is (0,0) for both the dam body and the reservoir, should be entered as end coordinates as well so that software considers them as closed shapes. The sample model is drawn using the software and shown in Figure 2.12.

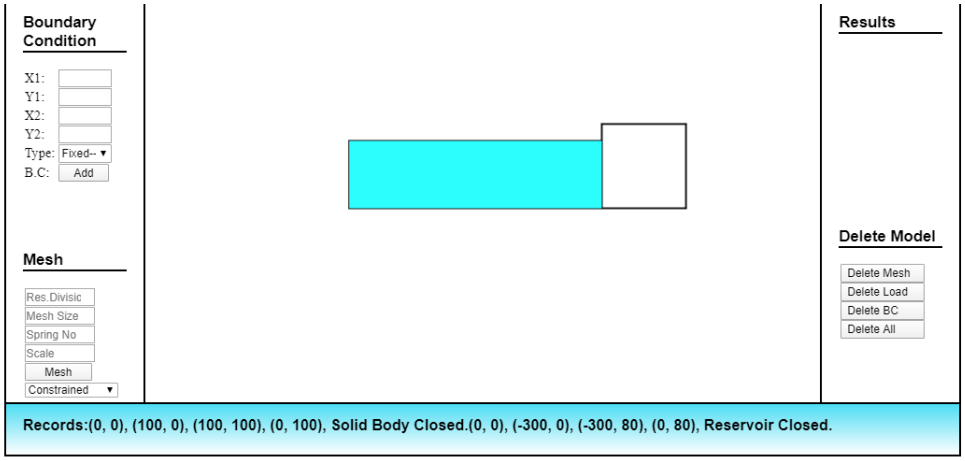


Figure 2.12. Sample Model with Reservoir Drawn by Software

2.3 Meshing

Mesh generation, which is the discretization of a large domain into small elements representing basic deformation and displacement behavior, is the building block for

many engineering simulation processes. The accuracy and convergence mainly depend on the meshing operation. Along with the drawing features, a meshing tool is also developed during the coding process of the software, which is capable of converting a large domain into a discretized system with possible different mesh densities. In this section, the methodology used behind the meshing operation, as well as the procedure for meshing of a body, is discussed in detail in the following sections.

Html 5 canvas element (<canvas>) is used as a container for the meshing operation in a similar fashion to geometry drawing. However, all meshed shapes are depicted on an independent layer attached to the main canvas element stage. The origin of the meshing layer is located at the left-bottom corner of the display window. Konva, a 2d canvas library, is activated inside this layer, which enables the user to draw any predefined shape during the meshing operation. This layer can contain squares representing the mesh elements during the discretization process. This layer hosts meshes for the dam body and reservoir simultaneously.

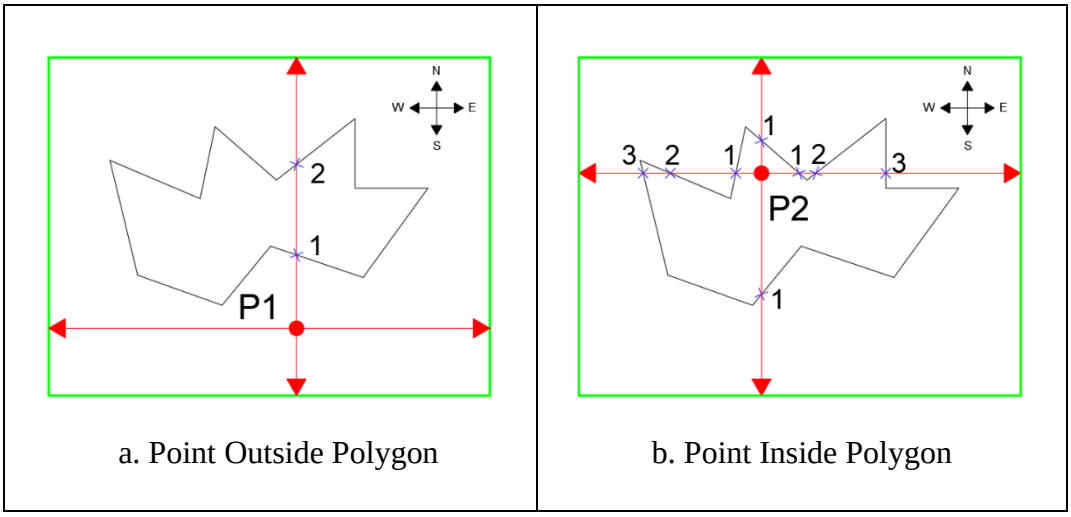
2.3.1 Meshing of the Dam Body

In this section, information regarding the algorithm and methodology used behind meshing operation of the dam body, data input, and output relating to the creation of the mesh is discussed. The associated mesh structure and the procedure for meshing a model via the developed software are explained.

2.3.1.1 Ray-Casting Algorithm

Ray-casting algorithm is one of the most common and popular ways to determine the enclosure of a point within a polygon in the computational geometries field. This algorithm checks the location of a point and determines if the point lies inside, outside, or on the boundary of a polygon. For a given point, four imaginary line segments, which are started from this point and extended up to the boundaries of the

canvas element, are drawn. Then, the number of the intersection of each line segment with the edges of the polygon is calculated. According to the ray-casting algorithm (Galetzka and Glauner, 2017), if the number of the intersection of any line with the polygon boundaries is odd, the point is said to be inside the polygon, and otherwise, if this number is even, the point lies outside the polygon. A schematic view of the ray-casting algorithm is shown for a point outside and inside of a polygon in Figure 2.13. The green boundaries represent the canvas element.



a. Point Outside Polygon

b. Point Inside Polygon

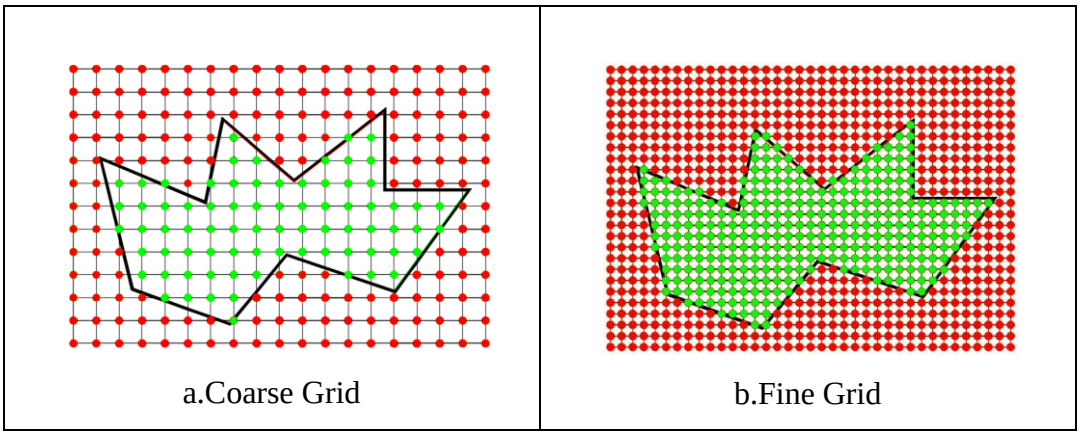
Figure 2.13. Ray-Casing Method for a Point Inside Polygon

It can be seen clearly in Figure 2.13 that regardless of the direction of the ray, as long as it is extended toward the polygon, any line can decide about the position of the investigated point based on the discussed theory.

2.3.1.2 Meshing Methodology for Solid Body

Ray-casting algorithm is used for the meshing operation of the dam body. During the meshing operation, the meshing layer is automatically gridded with grid size equal to the element size provided by the user for discretization. Once the layer is gridded, the location of every intersection of the grid lines, which is basically a point, with respect to the drawn polygon is found via the ray-casting algorithm. In this software, points lie on the models' boundary are considered to be inside the

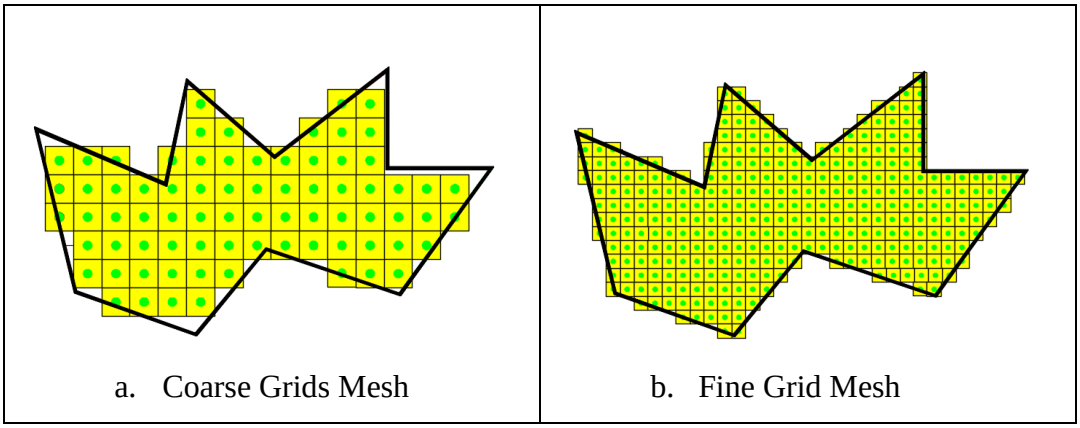
geometry. The schematic views of the ray-casting theory application for the coarse and fine gridded mesh layer are shown in Figure 2.14.



a.Coarse Grid b.Fine Grid

Figure 2.14. Ray-Tracing Algorithm for Meshing

Eventually, having the information about every single point, whether it lies inside or outside the polygon, a square mesh will be created for those lying inside the polygon. Following this procedure, regardless of the complexity of the polygon, a mesh can be created easily for a selected model. The meshed versions of the above model is shown in Figure 2.15. It can be seen in Figure 2.15.a that there is a loss in the area of the model when the mesh size is large while the meshed version with smaller mesh size covers most of the sample model. Therefore, a finer element size should be used for meshing operation to get an accurate discretized model.



a. Coarse Grids Mesh b. Fine Grid Mesh

`Figure 2.15. Meshed Shapes using Ray-Casting Algorithm

2.3.1.3 Meshing Methodology for Body with Opening

In this section, the methodology used during the meshing operation for the models with an opening is discussed. The geometry of the opening plays an essential role in the meshing operation. In this software, the geometry of openings is classified either as polygonal or circular. The methodologies for meshing of the models with circular openings and models with polygonal openings are explained in the following sub-sections.

2.3.1.3.1 Models with Polygonal Opening

For the meshing of models with polygonal opening, the ray-casting algorithm is used in a similar fashion. However, the ray-casting algorithm is used in two stages for these shapes, named as the direct and the indirect ray-casting. For the direct algorithm, which is used for the solid part of the model, mesh elements are created for those points lying inside the solid part. For the indirect algorithm, which is used for the polygonal openings, mesh elements are created for those points lying outside of the opening. Finally, the intersection of the mesh elements will define the region that needs to be discretized. The use of the model is shown in Figure 2.16.

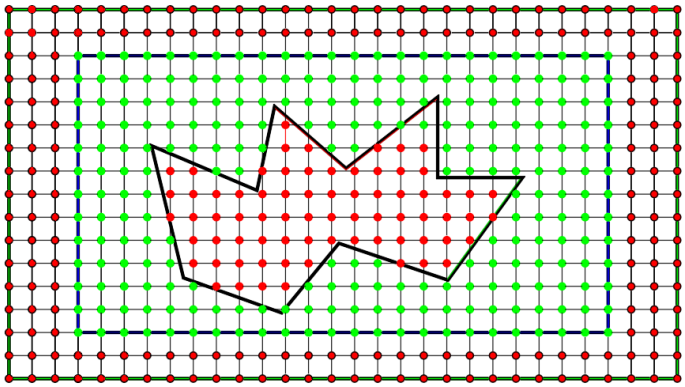


Figure 2.16. Ray-Tracing Algorithm for Models with Polygonal Opening

The outermost boundary shown in green is the boundary of the canvas element containing the solid body with a polygonal opening with blue and black boundaries,

respectively. It can be seen that the intersection of the points shown in green, which lie inside the solid body and outside of the opening, is the region where meshing operation will be done using the ray-tracing algorithm and the remaining points shown in red are discarded. By creating a square mesh element for each point shown in green, the final discretized version of the model is obtained, as shown in Figure 2.17. It should be noted that it is not necessary to use square elements during the meshing in the applied element method. Inclined elements or elements with different shapes can be used in meshing operation. However, the developed interface is only capable of meshing with square elements.

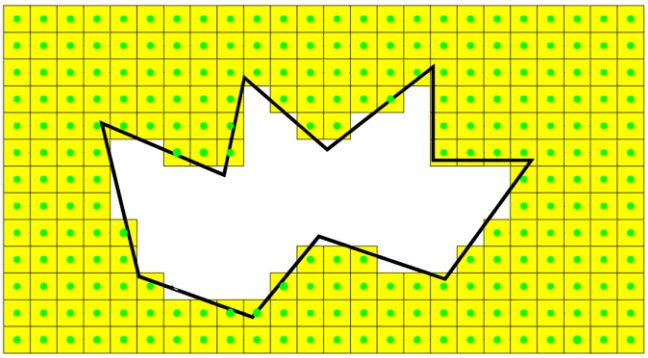


Figure 2.17. The Meshed Version of a Sample Model with Polygonal Opening

2.3.1.3.2 Models with Circular Opening

In the previous section, the interface was able to detect a polygonal opening inside a solid body. In order software to detect circular holes during meshing of the models, a new ray-casting algorithm is developed. In this algorithm, every point of the canvas element is checked for lying inside or outside the opening. Given a point P with coordinates of (X_P, Y_P) and a circle with a radius of R centered at (X_C, Y_C), the distance (d) between every intersection of the canvas grid and the opening center is calculated and compared with the radius of the circular opening as shown in Figure 2.18 for a single grid point.

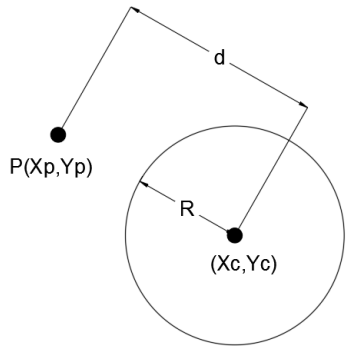


Figure 2.18. Ray-Casting for Circular Opening

If the distance (d) is greater than the radius (R), it is said that point lies outside the opening. Repeating this procedure, the intersection of the points lying outside the opening but within the solid part will be meshed with applied elements. Such a model is shown with the green region in the sample geometry shown in Figure 2.19.

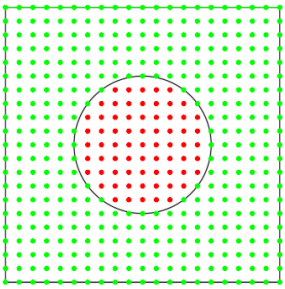


Figure 2.19. The Region for Meshing Operation

2.3.1.4 Mesh Properties

During the discretization operation for the dam part, the model is meshed into many applied elements, each having a node located at the center of the element with 3 degrees of freedom. In this software, only square applied elements are generated, which are connected to each other with springs, the number of which is defined by the user. The square geometry shown in Figure 2.3 is divided into four elements (Figure 2.20) in order to demonstrate the connectivity of the applied elements.

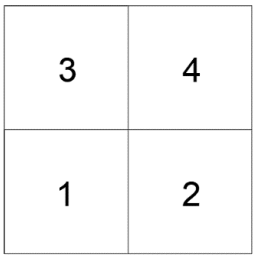


Figure 2.20. Sample Model Divided into 4 Elements

In the analysis of the above geometry, these elements are actually connected to each other horizontally, vertically, and diagonally with the springs. The number of springs connecting elements horizontally and vertically is provided by the user in the mesh box, while elements are always diagonally connected to each other with one spring. For the geometry shown in Figure 2.20, the number of springs connecting the elements horizontally and vertically is assumed as 10. It should be noted this number can be provided by the user through the developed interface, here just for illustration purposes, it is assumed as 10. The actual schematic connections of the elements are shown in Figure 2.21.

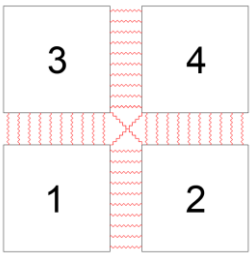


Figure 2.21. Springs Connecting the Meshed Elements

After the meshing operation, an output file containing element connectivity information is created and saved in the related directory by the software automatically for later use. It can be seen in Figure 2.21 clearly that there are four connection types for the meshing of the elements. When elements are connected to each other horizontally and vertically with springs, their connection types are symbolized as 1 and 2 respectively, while connection type 3 refers to the diagonal connection making an angle of 135 degrees, and connection type 4 making an angle of 45 degrees with the horizontal. This distinction is used later in the calculation of

the stiffness matrix and transformation angles. If connections types are 1 and 2, their respective spring type is denoted by one, and if connections type are 3 and 4, their spring type is specified as 2. A summary of connection and material types is provided in Table 2.3, while the output file of a sample containing the connectivity information is presented in Table 2.4.Using the definitions in Table 2.3, the connectivity information in the output file can be interpreted as follows:

- 1st element of the meshing is connected to the 2nd element horizontally (connection type 1) with 10 springs, and the material type is 1.
- 1st element of the meshing is connected to the 3rd element vertically (connection type 2) with 10 springs, and the material type is 1.
- 1st element of the meshing is connected to the 4th element diagonally (connection type 4) with 1 spring, and the material type is 2.
- 2nd element of the meshing is connected to the 3rd element diagonally (connection type 3) with 1 spring, and the material type is 2.
- 2nd element of the meshing is connected to the 4th element vertically (connection type 2) with 10 springs, and the material type is 1.
- 3rd element of the meshing is connected to the 4th element horizontally (connection type 1) with 10 springs, and the material type is 1.

Table 2.3 Summary of the Definitions of Elements Connectivity

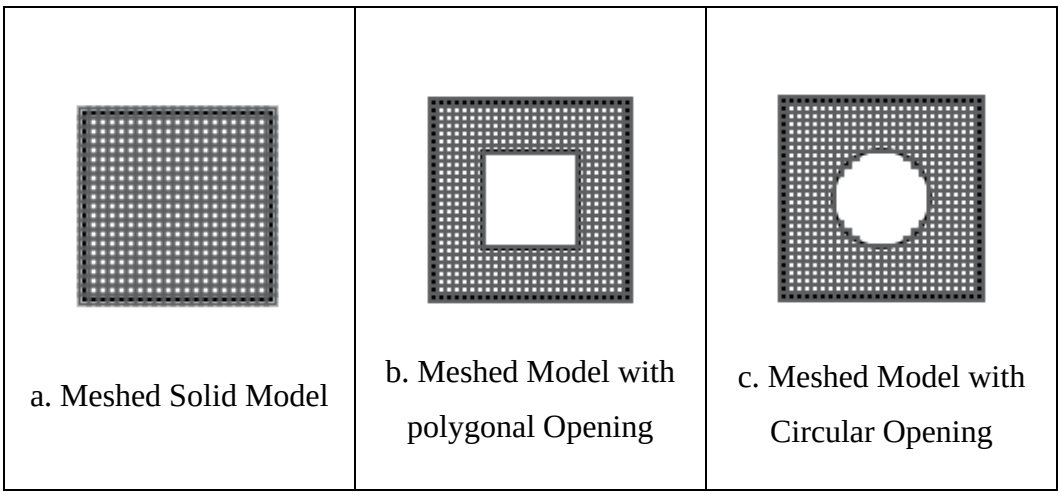
Connection Type	*Definition*
1	Horizontal Connection
2	Vertical Connection
3	Diagonal Connection with 135-degree Angle Counterclockwise from Horizontal
4	Diagonal Connection with 45-degree Angle Counterclockwise from Horizontal
Material Type	***Definition***
1	For Type 1 and 2 Connection
2	For Type 3 and 4 Connection

Table 2.4 Meshing Output Data Created for Figure 2.21

Element ID	Element ID	Connection Type	Spring Number	Material Type
1	2	1	10	1
1	3	2	10	1
1	4	4	1	2
2	3	3	1	2
2	4	2	10	1
3	4	1	10	1

2.3.1.5 Mesh Generation

For the models comprised of a solid dam body or dam body with openings only, input boxes located in the mesh box should be filled properly. Inputs such as meshing size and the number of springs connecting the applied elements together should be provided by the user in the mesh box. For illustration, sample models shown in Figure 2.3, Figure 2.6, and Figure 2.9 are considered for meshing. Mesh size and number of springs connecting the applied elements are chosen as 5 units and 10 springs, respectively. After the inputs in the mesh box are filled with the above data, and the clicking of the mesh button, the mesh of the geometry will be created, as shown in Figure 2.22. The material properties of the model should be provided in the material properties box in advance of the meshing operation.



a. Meshed Solid Model

b. Meshed Model with polygonal Opening

c. Meshed Model with Circular Opening

Figure 2.22. Meshed Versions of the Sample Models

2.3.2 Meshing of the Reservoir

In this section, information regarding the methodology used behind the meshing operation of a dam body having a reservoir is presented. The data input and output relating to the creation of the mesh as well as the associated mesh structure, mesh interaction at the boundary where dam body and reservoir meet, and the procedure

for meshing of a model comprised of dam body and reservoir via the developed software are discussed.

2.3.2.1.1 Dam-Reservoir Meshing Methodology

The methodology used for the meshing operation of a model consisting of a dam body and reservoir is explained in this section. The dam body has meshed with square applied elements having 3-DOFs (2 translations and 1 rotation), while the reservoir is discretized with 9-node structural finite elements and 3 pressure node at the center. Each structural node has 2-DOFs (translation in x and y direction), and each pressure node has one translational DOF, making a total of 21-DOFs per each finite element. 9-node structural finite elements was preferred over the 4-node finite elements as the former is very efficient in representing the dam–reservoir response over a wide frequency range (Bouanani and Ying Lu,2009). A schematic view of the reservoir elements with structural nodes (yellow-colored) and pressure nodes (Blue-colored) is shown in Figure 2.23.

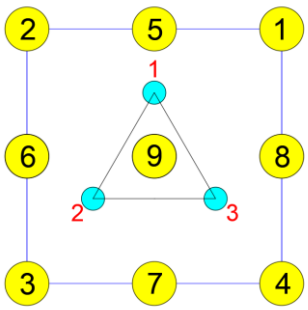


Figure 2.23. Structural and Pressure Nodes of the Reservoir Elements

As the dam body and the reservoir are constituted with very different kinds of elements, special treatment of the intersection of the two elements was required. The approach for discretizing the interaction modeling between the reservoir and dam elements is discussed below.

In order to provide a clear insight regarding the meshing problem, a sample model shown in Figure 2.11 is taken into consideration. Both the dam body and reservoir are divided into 20 applied and finite elements, as shown in Figure 2.24. Connecting the second-order elements to the applied elements at the nodes causes connectivity problems if not done properly. In this case, it can be seen clearly that for every two applied elements vertically, there is one finite element with applied element's nodes coinciding with the reservoir elements at only the gray nodes, as shown in Figure 2.25. Red nodes, i.e., the mid-nodes of finite elements at the boundary, remain free, causing problems in the analysis. Accordingly, the meshing methodology is adjusted for the full coincidence of end nodes of applied and finite elements at the reservoir-dam interface.

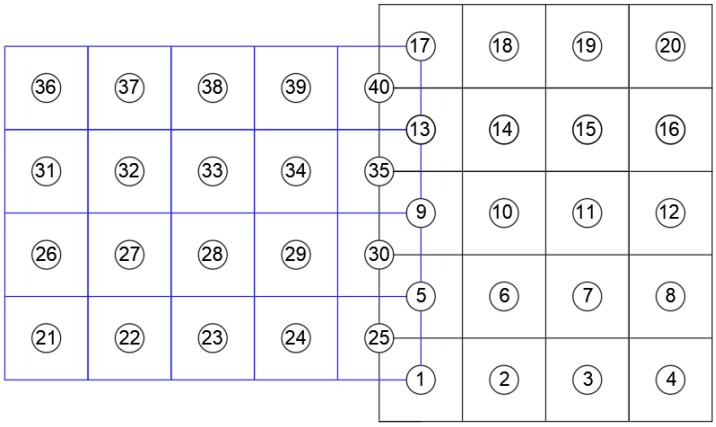


Figure 2.24. Primary Meshing Approach

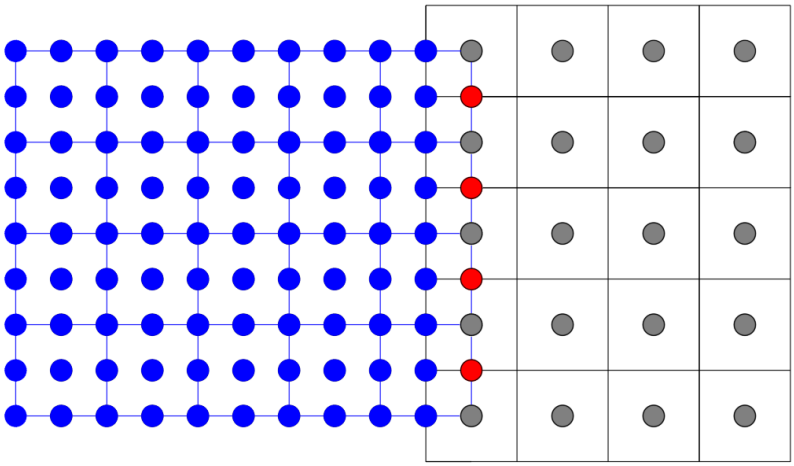


Figure 2.25. Applied and Finite Nodes of the Model

2.3.2.1.1.1 Reservoir Meshing Methodology

Two different options were made available for the meshing of the reservoir in the software. The first option is the direct sharing of common nodes at the reservoir boundary. In this option, horizontal lines from every other two applied elements are drawn up to the end of the reservoir limit prescribed by the user. As the mixed pressure-displacement element is quadratic, a reservoir element is connected to three applied elements on the upstream face of the dam body, which makes the size of the reservoir elements twice as the size of applied elements vertically. For the second option, the same procedure as the first option is followed; however, nodes are not shared at the boundary, but collocated for the dam and the reservoir shapes. Equal displacement constraints connecting the reservoir and the dam body nodes are defined in x-direction.Schematic views of the model sharing common nodes at the boundary and nodes constrained at the boundary are shown in Figure 2.26 and Figure 2.27, respectively.

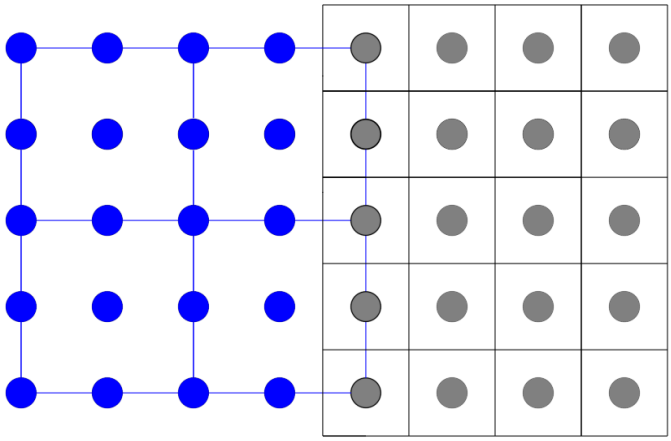


Figure 2.26. Sharing Common Nodes at the Boundary

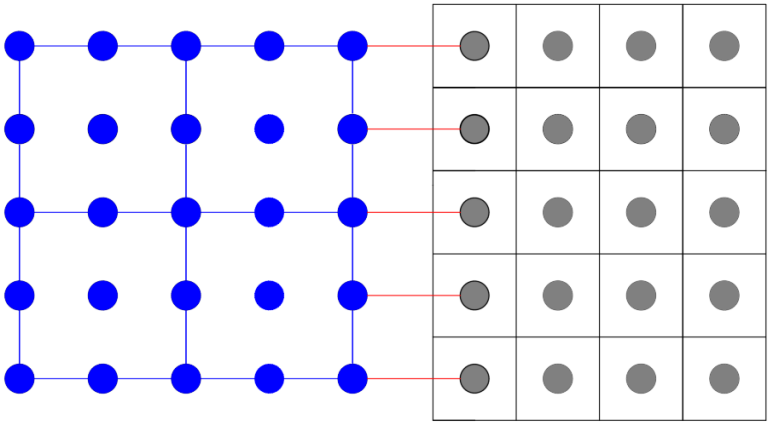


Figure 2.27. Nodes Constrained at the Boundary

2.3.2.2 Meshing Properties

In order to create an output file containing the element node connectivity information, nodes of the dam body, which coincide with the center of the applied elements, are numbered from left to right row-wise up to the top of the body. Similarly, finite elements, each having 9 nodes are numbered up to the top of the reservoir. Node numbering for node sharing at the boundary and for the case of nodes constrained at the boundary are shown in Figure 2.28 and Figure 2.29, respectively.

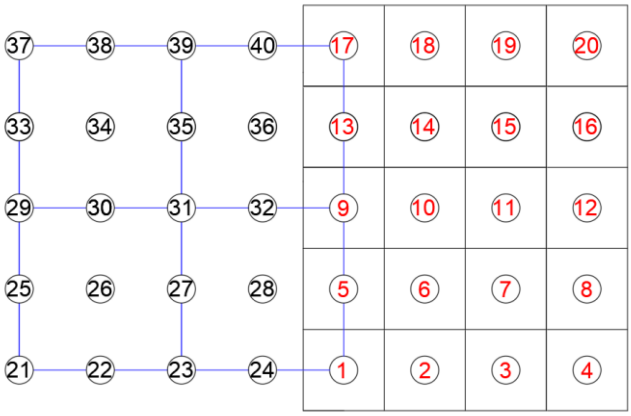


Figure 2.28. Node Numbering for Node Sharing at the Interface

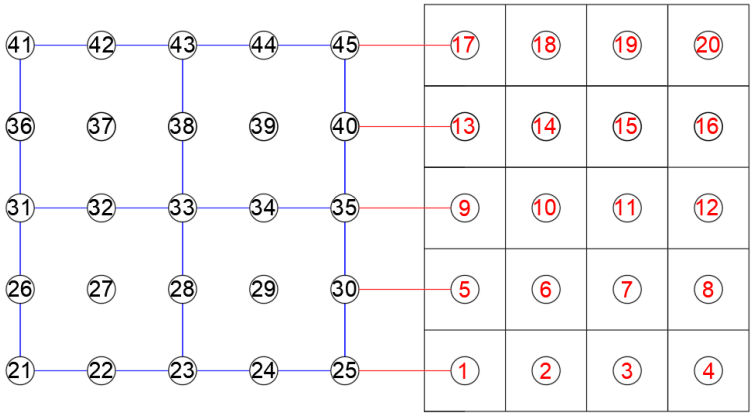


Figure 2.29. Node Numbering for Constrained Nodes at the Interface

In order to complete the meshing of the model comprised of the dam body and reservoir, in addition to the element size and spring number, the horizontal discretization of the reservoir is also required. The discretization of the reservoir in the horizontal direction is conducted with a structured mesh dividing the reservoir into uniformly wide elements in the horizontal direction. The number of the segments for the division of the reservoir is provided to the code. In order to show the meshing process, the reservoir of the sample model shown in Figure 2.11 is assumed to be divided into 15 segments horizontally. Element size for dam body and the number of springs connecting the applied elements are assumed to be 10. Having filled the input boxes accordingly and selecting either constrained or unconstrained

options, the mesh will be generated automatically after clicking the mesh button provided in the mesh box. The generated mesh is shown in Figure 2.30. While the reservoir is divided into 15 segments, it should be noted that each of these segments contain quadratic elements: hence the division is for the size of the elements, not the inter-node distances.

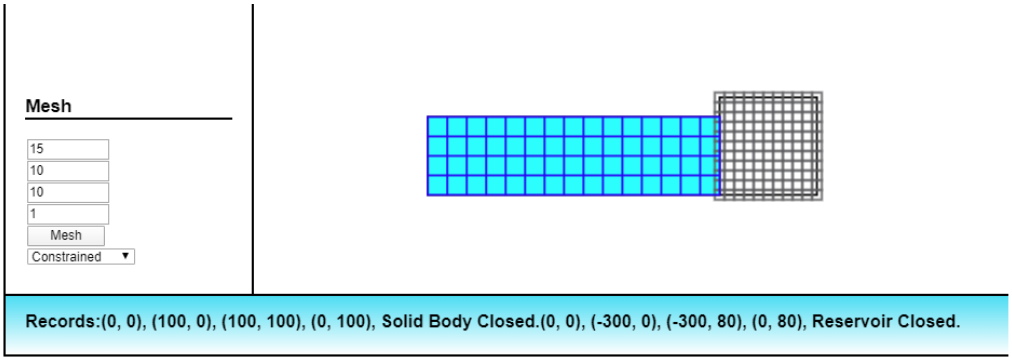


Figure 2.30. Meshed Version of the Model

2.4 Boundary Conditions

In this section, the methodology used during the coding of the software for applying boundary conditions, boundary condition's data input and output creation, as well as the procedure for applying boundary conditions to the different parts of the models are discussed.

2.4.1 Methodology

Different parts of the discretized meshed model can be assigned with different boundary conditions. Fixed, pin, and roller supports can be added to the model easily via the developed software. The assignment of support condition should be done after the meshing operation since support conditions are applied directly to the nodes

of the discretized model. The meshed model shown in Figure 2.31 is used to explain the boundary assignment methodology.

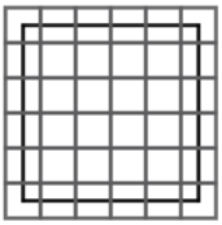


Figure 2.31. Sample Model for Support Assignment

The following steps should be followed to assign support conditions to the base of the sample model shown above. It can be seen clearly that the base of the model consists of many applied elements. An imaginary line passing through centers of base elements is defined, and then the predefined support condition will be applied to the degrees of freedom of each individual element lying along the line's path. A zoom-in view of the procedure applied to the model with a line shown in blue passing through the nodes of the base elements is shown in Figure 2.32.

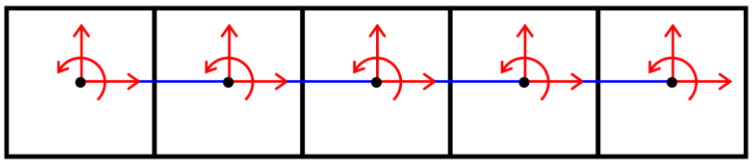


Figure 2.32. Methodology for Assignment of Boundary Conditions

During the boundary assignment operation, the output file containing the information related to the location and type of the support conditions assigned to the model are created via developed software. This information includes the element's node number and the related restraints applied to the degrees of freedom of that node. Eventually, an array (with rows equal to the number of support and 4 columns) is created and exported as an output to a text file for later use where each row will describe a support in the structure. The first column of the created array will be the nodal point ID number of the support, and the remaining three columns will be numerical code, which defines the restraint state for the translation in global X-

direction, translation in global Y-direction and rotation about the global Z-axis. In this software, numerical codes of 0 and 1 are used for free and restrained states, respectively. For illustration, it is assumed the base of the sample model shown in Figure 2.31 is fixed, and data shown in Table 2.5 is obtained.

Table 2.5 Boundary Data for Sample Model with Fixed Base

Nodal ID	*Translation in X-Direction*	*Translation in Y-Direction*	*Rotation about Z-Axis*
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1

2.4.2 Boundary Application

Restraints to different parts of the models can be assigned via the boundary condition section. Since support conditions are applied directly to the nodes of the model, the assignment of support condition should be done after the meshing operation. In order to apply support conditions, the coordinates of start and end of the line along which the support is applied, are inserted to the input boxes provided in the boundary assignment section as X1, Y1, X2, and Y2, respectively. Then the type of support can be selected from the "Type" drop-down list. Finally, by clicking the add button, boundary conditions are assigned on nodes that lie along the defined line. When the application of the supports is acknowledged by printing on the user interface, fixed, pin, and horizontal roller and vertical roller supports are shown in color shading of green, red, black, and orange respectively on the canvas layer. The boundary condition section is shown in Figure 2.33.

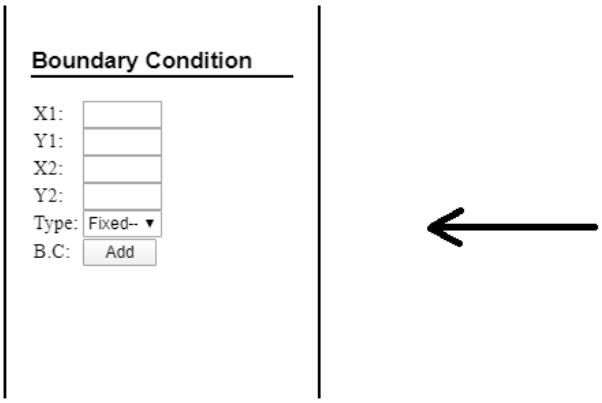


Figure 2.33. Boundary Condition Section

2.4.2.1 Singular Supports

For the sake of illustration, fixed support is going to be assigned at the base of the geometry shown in Figure 2.31 with finer mesh. Coordinates of the imaginary line passing through the base of the sample model are (0,0) and (100,0), which is the side length of the model. After inserting the coordinates properly and selecting the fixed option from the drop-down list, a green shading representing the fixed support is printed by once the add button on the boundary conditions section is clicked. The sample model with a fixed Base is shown in Figure 2.34.

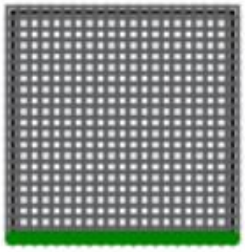


Figure 2.34. Sample Model with Fixed Base

2.4.2.2 Multiple Supports

Multiple boundary conditions can also be added to the desired geometry. For every boundary condition, coordinates of the line along which restraints will be assigned

are applied with the procedure explained before. An example of multiple supports assigned to different parts of a model is shown in Figure 2.35.

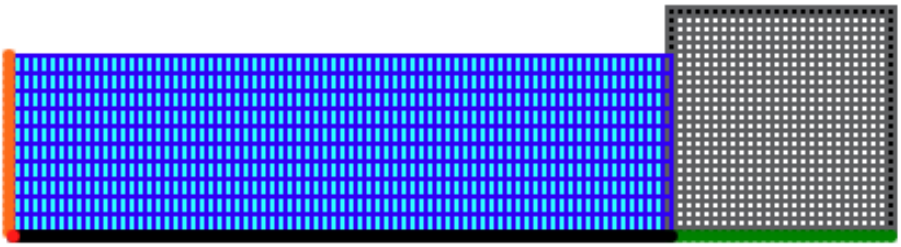


Figure 2.35. Sample with Multiple Supports

2.4.2.3 Inclined Supports

Support conditions can also be defined on inclined surfaces of a structure. A sample model with an inclined edge is shown in Figure 2.36. It is assumed that the base of the model is fixed while the inclined edge is pinned.

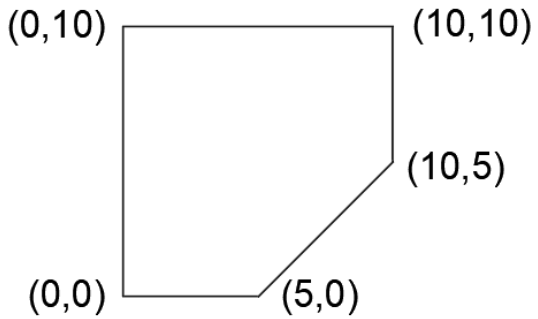


Figure 2.36. Coordinates of the Model

For the base segment, coordinates of (0,0) and (5,0) are inserted in the related input boxes. For the inclined edge coordinates of (5,0) and (10,5) are inserted to the related input boxes, and after selecting the roller and fixed options for each part, respectively, the restrained version of the model is obtained as shown in Figure 2.37. It can be seen that software is capable of assigning both straight and inclined restraints to any part of a model easily.

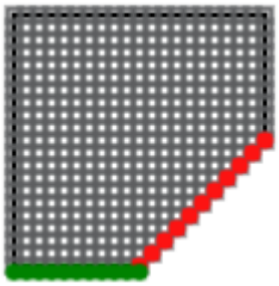


Figure 2.37. Sample Model with Inclined Support

2.5 Loads

The interface is capable of assigning different loads such as point, uniform, triangular, trapezoidal, hydrostatic, uplift, and earthquake loads to different parts of the structure. In this section, information about the methodology used during the coding, input, and output data created during load application, and information about the procedure for applying loads using the interface will be discussed. Load application procedure, requiring meshing information, works after the meshing sequence is completed.

2.5.1 Point Load and Distributed Loads

Point loads can be assigned to various parts of a model via the “Assign” tab located in the menu bar of the interface. For the application of point load, x, and y coordinates, the magnitude and direction of the load should be provided by the user into the related pop-up window. The point load pop-up window is shown in Figure 2.38.

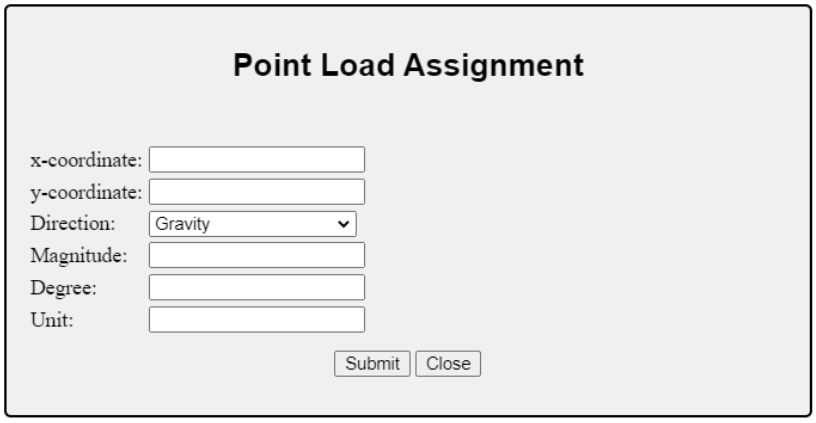


Figure 2.38. Point Load Pop-up Window

A positive and negative value for the magnitude of the load refers to tension and compression force, respectively. Using the direction selector, only horizontal and vertical point loads can be added while the degree option can be used alternatively for applying loads with an angle to the specified point. Point loads with different magnitudes and directions are applied to a sample model shown in Figure 2.39.

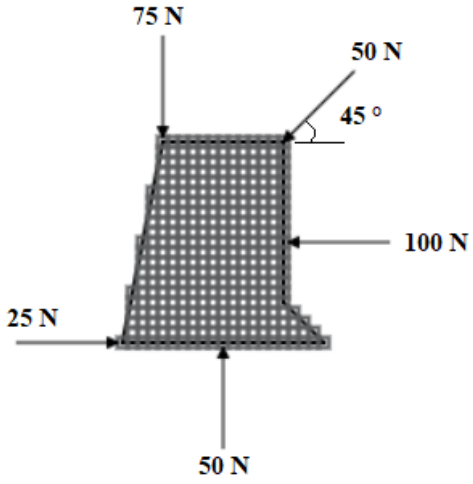


Figure 2.39. Point Loads Applied to a Sample Model

Uniform load application is made through the “Assign” tab located in the menu bar of the interface. In order to apply uniform distributed load along boundaries of a model, magnitude, direction, and coordinates of starting and ending points of a line along which uniform load will be applied should be provided by the user to the input

boxes of the related pop-up window. A sample model with applied uniform loads is shown in Figure 2.40.

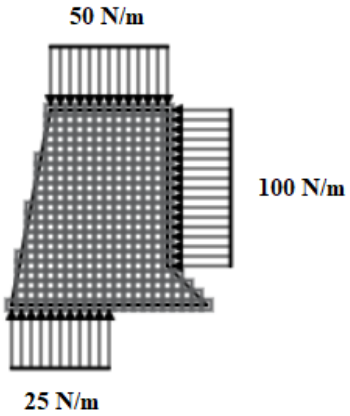


Figure 2.40. Uniform Load Applied to a Sample Mode

The developed interface is capable of applying both trapezoidal and triangular loads to any location of a drawn model via the trapezoidal load option in the "Assign" tab. For the application of trapezoidal distributed load similar to the uniform load, X_1, Y_1, X_2, Y_2, which represent respectively the coordinates of starting and ending points of the line along which trapezoidal will be applied, should be provided by the user. In contrast to the uniform load, which needs a single value for the magnitude of the load, the trapezoidal load application needs two values for the magnitude of the starting and ending, which respectively change linearly along the defined line. Triangular load application is made with the same procedure; the starting or the ending point magnitude of the load is set equal to zero in the related pop-up window. For illustration purposes, trapezoidal and triangular loads are applied to the sample model shown in Figure 2.41.

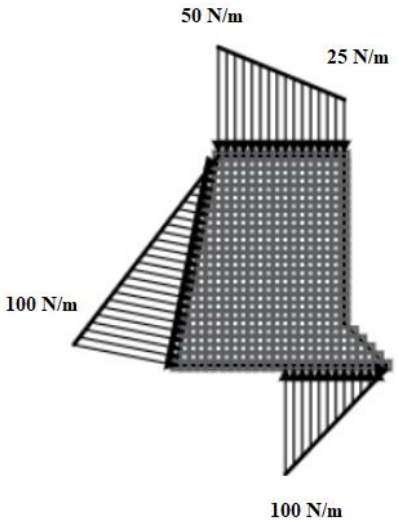


Figure 2.41. Sample Model with Trapezoidal and Triangular Loads

2.5.2 Hydrostatic Force and Uplift Forces

Hydrostatic force on models with the reservoir is added automatically by the software to the dam body elements where the reservoir and dam body meet each other. In fact, the methodology for applying hydrostatic loads to the model is similar to applying a triangular load perpendicular to the surface of the dam, which changes linearly through the depth of the reservoir. For simplicity, this procedure is embedded in the code so that as soon as a reservoir is drawn on the interface, the hydrostatic load is added to the boundary automatically. After the assignment of the hydrostatic load, in contrast to the previous loads, which are visible when added to the model, the hydrostatic load is not visible on the interface. However, data of this type of loading is directly written to the load's output file. In calculating hydrostatic force along the face of the dam, the hydrostatic load ($\rho * g * h$) is calculated and multiplied by element thickness (t) (ρ, g, h, and t are the density of water, gravitational acceleration, reservoir depth, and thickness of the model, respectively). For illustration, the hydrostatic force on two sample models, by default not shown on the main interface window, is shown in Figure 2.42.

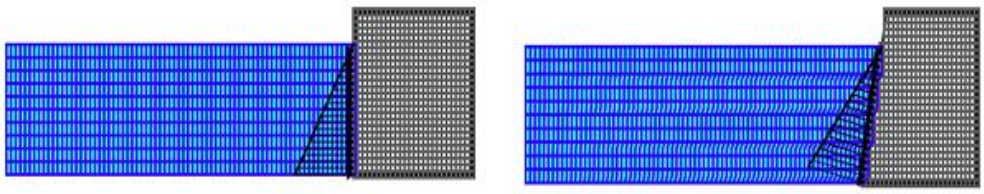


Figure 2.42. Hydrostatic Force Applied to the Models via the Interface

Uplift forces on the base of the dams are calculated according to the Federal Energy Regulatory Commission (FERC 1999) uplift distributions. The interface is able to assign three cases of FERC uplift distributions detailed in the following sections.

In this part, the methodology for defining uplift pressure for a dam cross-section having a drainage gallery (no cracking) is explained. For the calculation of the uplift distribution along the base of a model, the unknown value of H_3 needs to be calculated from the known values of upstream head (H_1), downstream head (H_2), y-coordinate (Y), and x-coordinate (X) of drains measured from the left toe of the dam model. There exist two scenarios for the calculation of H_3 value as follow:

Case-1: $Y > H_2$ $\rightarrow$ $H_3 = K\left[(H_1 - H_2)\,\frac{(L-X)}{L} + H_2 - Y\right] + \mathrm{Y}$

Case-2: $\mathrm{Y} < H_2$ $\rightarrow$ $H_3 = K(H_1 - H_2)\,\frac{(L-X)}{L} + H_2$

In the calculations of H3, L is the base length, and k is defined as $(K = 1 - E)$, where E is a coefficient defining the effectiveness of the drain. The final view of the uplift distribution for this case is shown in Figure 2.43.

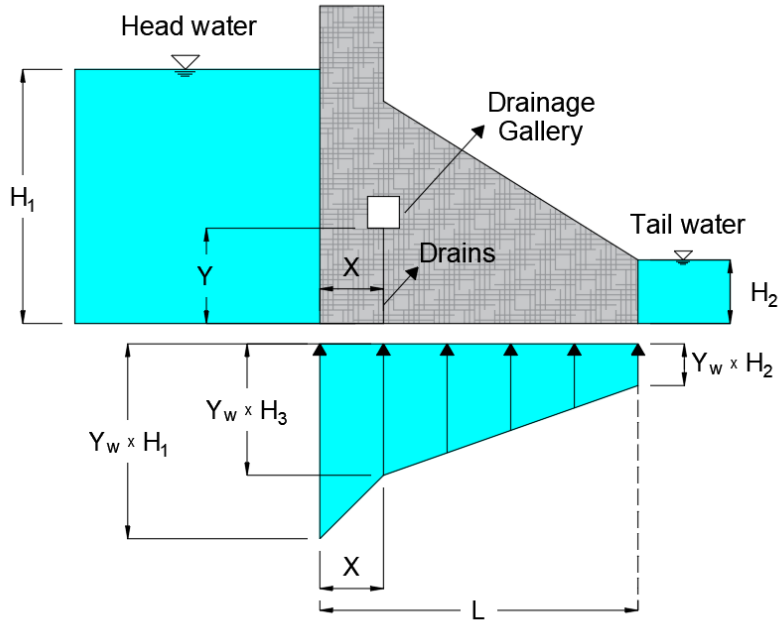


Figure 2.43. FERC Uplift Distribution with Drainage Gallery (No Cracking)

The above formulation is embedded in the code, and it can be applied via the Uplift (No cracking) option in the “Assign” tab. After drawing and meshing of a sample model and filling the related pop-up window’s inputs, uplift distribution similar to the sample model shown in Figure 2.44 is obtained. It should be noted that the obtained distribution is applied to the base elements as a collection of point loads with different magnitudes, and finally, the loading information will be exported to the load's output file.

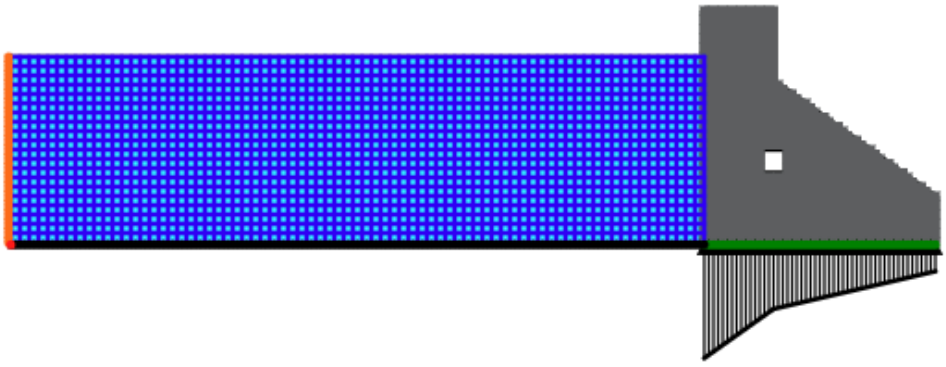


Figure 2.44. Uplift Applied to Model with Drainage Gallery

In this section, the methodology for defining uplift pressure for a dam cross-section with drains near the upstream face (no cracking) is explained. For the calculation of the uplift distribution along the base of a model, similar to the previous case the unknown value of H_3 needs to be calculated from the known values of upstream head (H_1), downstream head (H_2), y-coordinate (Y) and x-coordinate (X) of drains measured from the left toe of the dam model. This case is applied when the drains are near the upstream face and the relation $X \leq 0.05 \times H_1$ is satisfied. In this case, H_3 will be close to H1, therefore pressure distribution can be assumed simply as a trapezoidal distribution, which varies linearly between H_3 and H_2. Based on these values there exist two cases for the calculation of H_3 value as follow:

Case-1: $Y > H_2$ $\rightarrow$ $H_3 = K(H_1 - Y) + \mathrm{Y}$

Case-2: $\mathrm{Y} < H_2$ $\rightarrow$ $H_3 = K(H_1 - H_2) + H_2$

In the calculations of H3, L is the base length, and k is defined as $(K = 1 - E)$, where E is a coefficient defining the effectiveness of the drain. The final view of the uplift distribution for this case is shown in Figure 2.45.

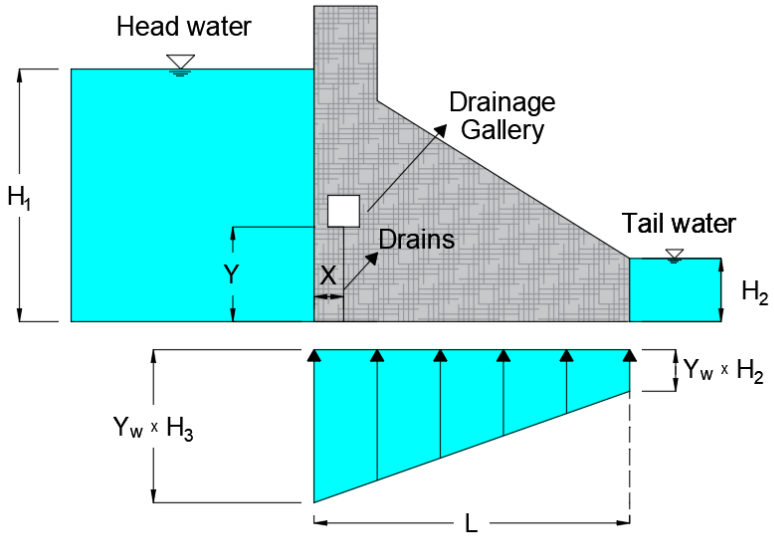


Figure 2.45. FERC Uplift with Drains Near Upstream Face (No Cracking)

A sample model satisfying the above-mentioned criterion is drawn using the interface, and the result shown in Figure 2.46 is obtained.

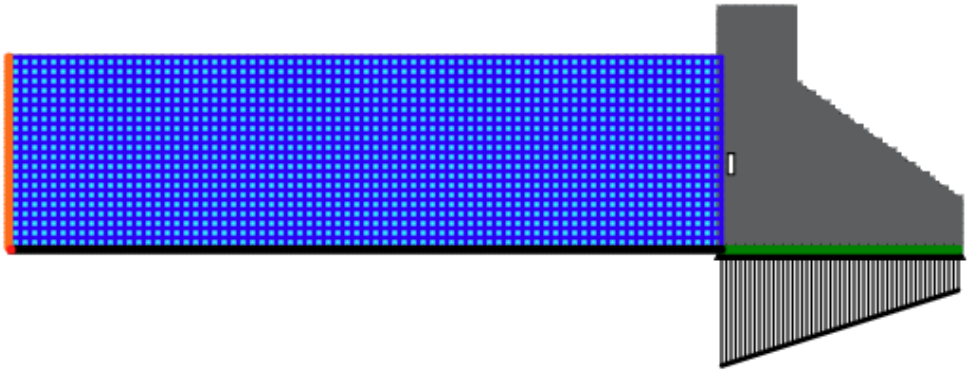


Figure 2.46. Uplift Applied to Model with Drains Near Upstream Face

In this section, the methodology for defining uplift pressure for a dam cross-section having a drainage gallery and cracking not extending beyond drains is explained. For the calculation of the uplift distribution along the base of a model, similar to the previous cases the unknown value of H_3 needs to be calculated from the known values of the upstream head (H_1), downstream head (H_2), y-coordinate (Y) and x-coordinate (X) of drains measured from the left toe of the dam model. In this case, in addition to the mentioned known values, the length of crack (T), with the criteria of $T \leq$ X, must be known. Based on these values there exist two cases for the calculation of H_3 value as follow:

Case-1: $Y > H_2$ → $H_3 = K\left[(H_1 - H_2)\,\frac{(L-X)}{L-T} + H_2 - Y\right] + Y$

Case-2: Y $< H_2$ → $H_3 = K(H_1 - H_2)\,\frac{(L-X)}{L-T} + H_2$

In the calculations of H_3, L is the base length, and k is defined as $(K = 1 - E)$, where E is a coefficient defining the effectiveness of the drain. The final view of the uplift distribution for this case is shown in Figure 2.47.

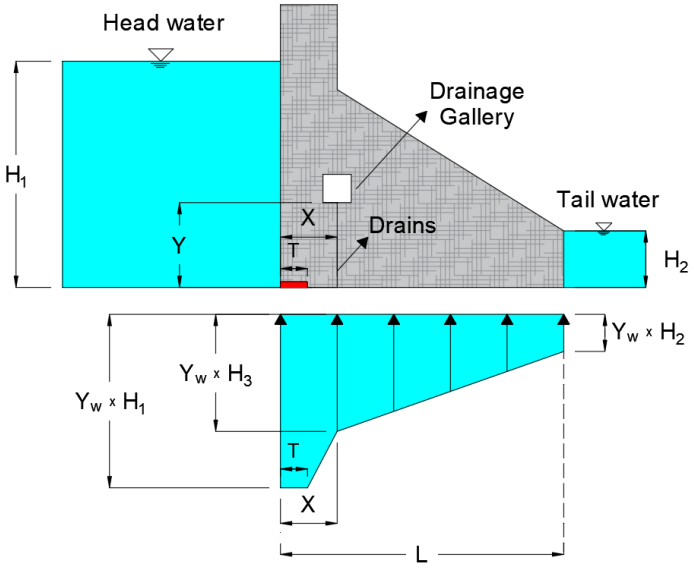


Figure 2.47. FERC Uplift with Cracking not Extending Beyond Drains

A sample model satisfying the above-mentioned criterion is drawn using the interface, as shown in Figure 2.48.

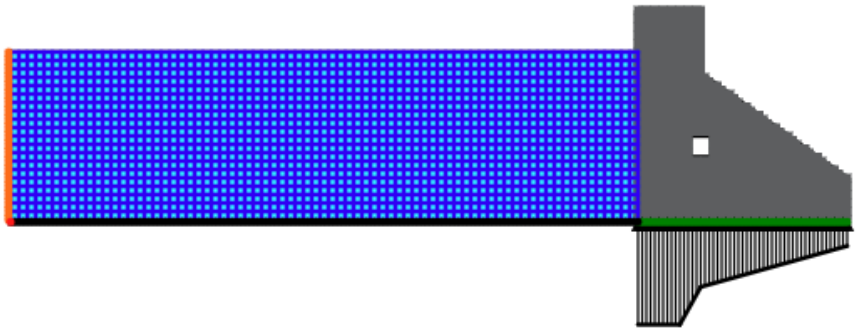


Figure 2.48. Uplift Applied to Model with Cracks not Extending Beyond Drains

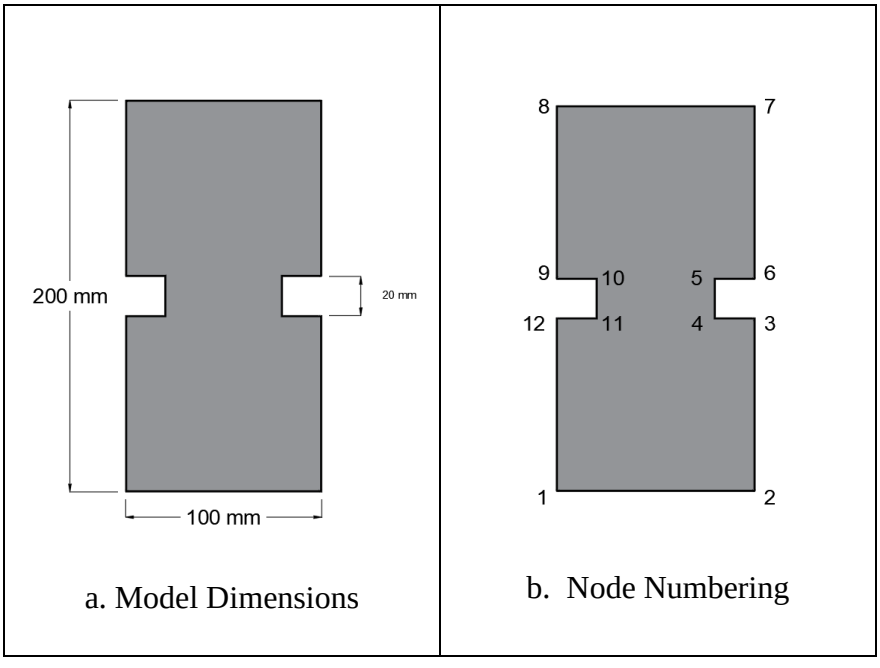
2.6 Benchmark Problems

In this section, the capabilities of the developed interface and the accompanying code are tested via modeling of numerous complex geometries. The reliability and accuracy of the algorithms used during the development the interface, such as ability in drawing complex models, the accuracy of detecting model's boundaries and

openings during the meshing operation with different mesh size, application of various loads and boundary conditions to different parts of the structure as well as the post-processing functionalities are the tasks that need to be verified. Meshing operation capability of the software is the most important and the essential building block of this software since all other operations depend on the flawless operation of this stage. Therefore, more focus will be put on the verification of results obtained from the meshing stage.

2.6.1 Rectangular Prism

The first geometry discretized using the developed software belongs to a test specimen for direct tensile testing. The specimen was modeled as a rectangular prism with a width and height of 100 mm and 200 mm, respectively. Two notches of 20 mm height and width are defined at the mid-section. The geometry of the model and numbering of the model corners are shown in Figure 2.49 and corresponding coordinates for the provided numbers are presented in Table 2.6.



a. Model Dimensions

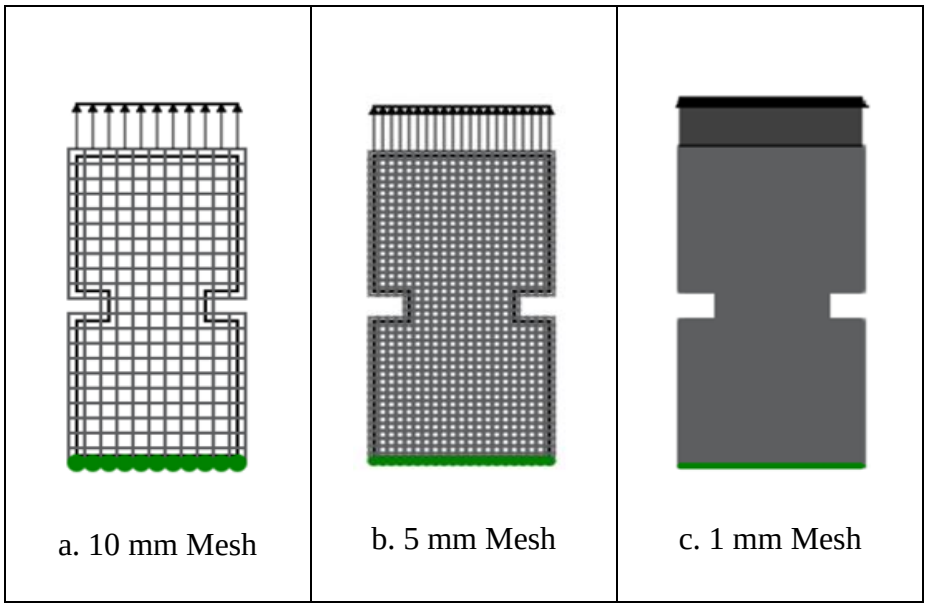
b. Node Numbering

Figure 2.49. Dimensions and Numbering of the Rectangular Prism

Table 2.6 Coordinates of the Rectangular Prism Model

Points	*Coordinates*	Points	*Coordinates*
1	0,0	7	100,200
2	100,0	8	0,200
3	100,90	9	0,110
4	80,90	10	20,110
5	80,110	11	20,90
6	100,110	12	0,90

For the meshing of the model, the number of springs connecting the applied elements is assumed as 10. The meshed version of the model with 10 mm, 5 mm, and 1 mm square elements creating a total of 227, 837, 19541 applies elements, respectively, are shown in Figure 2.50. The run time for the discretization of the model with 10 mm and 5 mm elements is a couple of milliseconds, while it takes around 4 minutes for the case of 1 mm mesh. Support and loading are shown for illustration purposes.

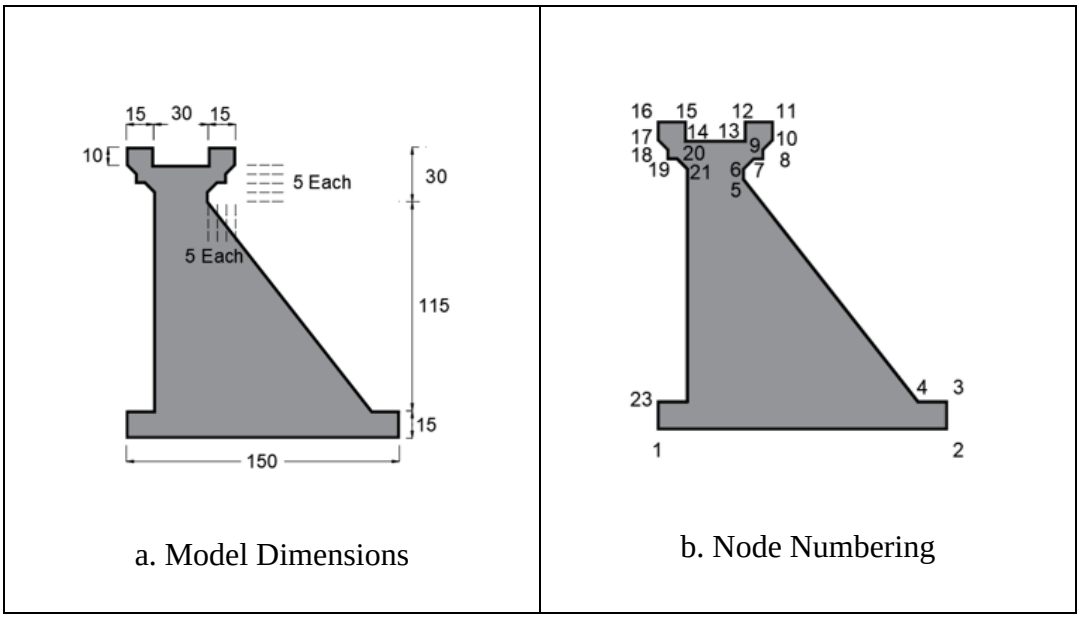


a. 10 mm Mesh b. 5 mm Mesh c. 1 mm Mesh

Figure 2.50. Model Meshed with Three Different Mesh Size

2.6.2 Gravity Dam Cross-Section

The second geometry discretized using the developed software is a complex cross-section of a shake table test specimen with dimensions in centimeter and corner numbering given in Figure 2.51.



a. Model Dimensions

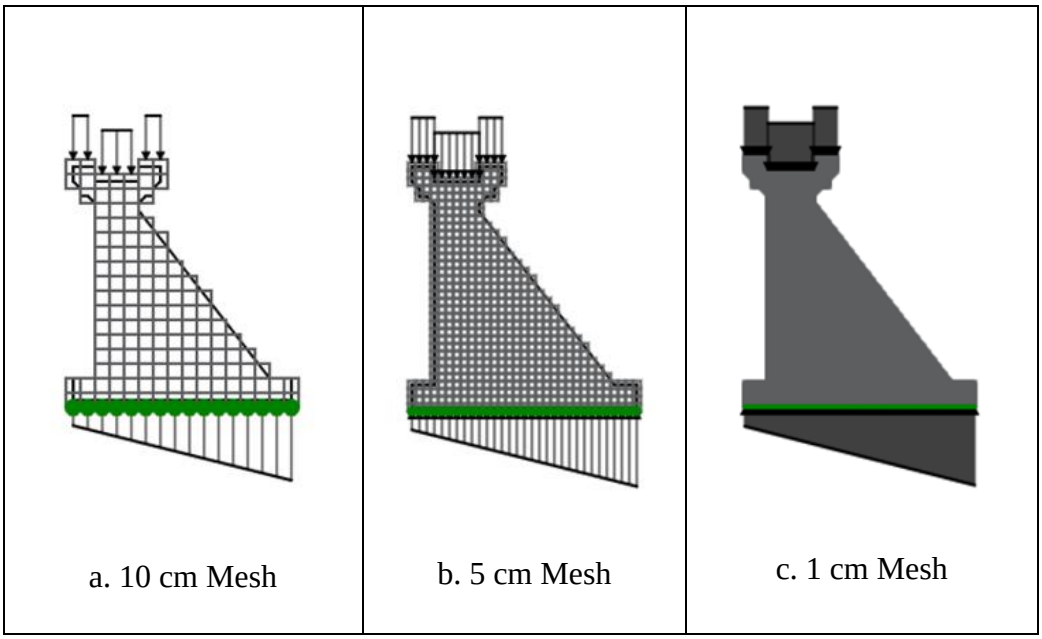
b. Node Numbering

Figure 2.51. The Dimensions of the Sample Gravity Dam Model and Numbering

The corner's coordinates corresponding to the numbers given in Figure 2.51 are presented in Table 2.7. After the definition of the geometry, square elements of 10 cm, 5 cm, and 1cm are used for the meshing of the dam model, making a total of 136, 530, and 12266 elements respectively for each case. A trapezoidal loading is defined at the base, while a uniform load is defined at the top of the geometries. Mesh and the aforementioned loads are shown in Figure 2.52.

Table 2.7 Coordinates of the Gravity Dam Cross-Section

Points	*Coordinates*	Points	*Coordinates*
1	0,0	13	45,150
2	150,0	14	15,150
3	150,15	15	15,160
4	135,15	16	0,160
5	45,130	17	0,150
6	45,135	18	5,145
7	50,140	19	5,140
8	55,140	20	10,140
9	55,145	21	15,135
10	60,150	22	15,15
11	60,160	23	0,15
12	45,160		

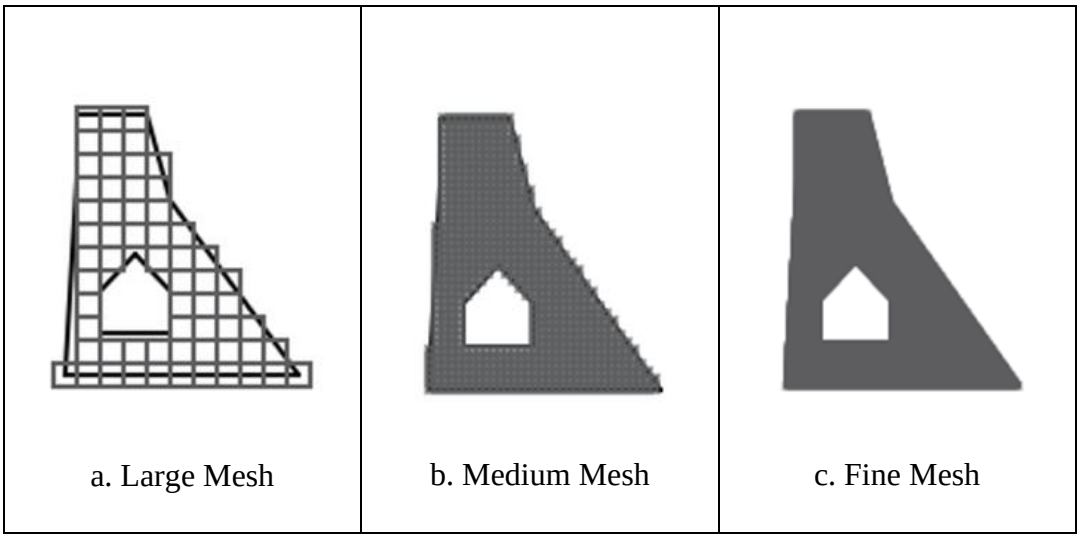


a. 10 cm Mesh b. 5 cm Mesh c. 1 cm Mesh

Figure 2.52. Model Meshed with 10 cm, 5 cm, and 1 cm Square Elements

2.6.3 Irregular Berm Cross-Section

The third geometry is a cross-section of an inclined faced dam with a polygonal opening. Arbitrary dimensions are used for the geometry and the opening in order to examine the meshing capability of the program. This geometry meshed with three different mesh sizes, as shown in Figure 2.53. The models comprised of the large, medium, and small elements are comprised of 61, 650, and 23260 elements, respectively.



a. Large Mesh | b. Medium Mesh | c. Fine Mesh

Figure 2.53. Sample Model with Large, Medium, and Small Mesh Sizes

2.6.4 Body with Reservoir

The developed software is tested for meshing with very small-sized elements in this benchmark. Dimensions of the model and the numbering of corners for the dam part as well as the reservoir are shown in Figure 2.54, and the corresponding coordinates are shown in Table 2.8.

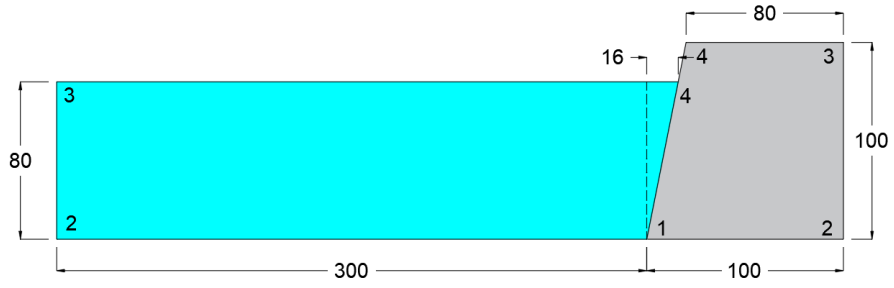
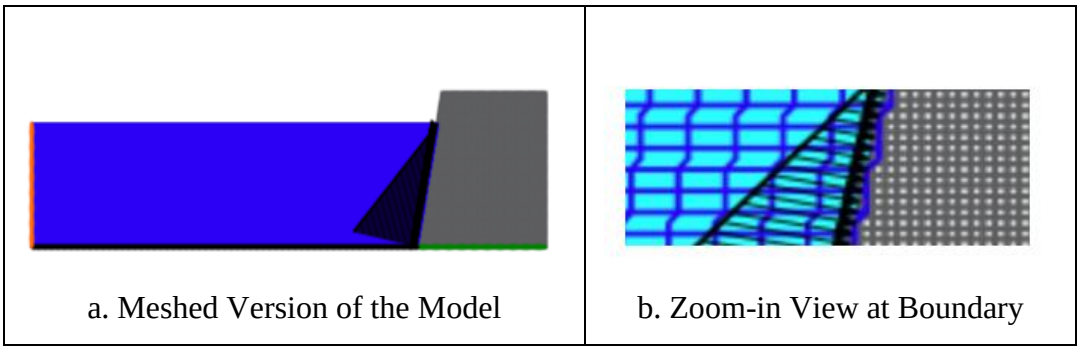


Figure 2.54. The Dimensions of Body with Reservoir and Numbering

Table 2.8 Coordinates of Dam Body with Reservoir

Dam Body		Reservoir	
Points	*Coordinates*	Points	*Coordinates*
1	0,0	1	0,0
2	100,0	2	-300,0
3	100,100	3	-300,80
4	20,100	4	16,80

The body and the reservoir have given in Figure 2.54 have been meshed with 0.5 m square elements creating 36301 applied and 48000 finite elements, corresponding to a total of approximately 500000 degrees of freedom. The meshed version of the model is shown in Figure 2.55. As the mesh elements are very small compared to the size of the model, the meshed version of the model appears as solid sections in Figure 2.55a, with a zoom-in window showing the details in Figure 2.55b. It can be seen that elements with irregular shapes can be formed during the meshing. As the upstream of the dam body is almost always close to the vertical, the irregularity with the reservoir element shapes is limited. During the analysis, irregular shapes are mapped into 9-node rectangular master elements, and then integration over the master element is performed using Gauss quadrature.



a. Meshed Version of the Model | b. Zoom-in View at Boundary

Figure 2.55. Meshed Sample with 84301 Elements

2.7 Post-Processing

In this section, many embedded features for the process of the analysis of results are explained and discussed in detail. When analysis of the model is done through the solver engine in the back-end part of the software, results are also processed with the developed interface, and the corresponding files are created and presented for the user. Gnuplot, which is a portable command-line driven graphing utility, is embedded for creating and plotting of modes shapes, plotting of stress and strain contours of the analyzed models.

Before discussing the procedure for plotting mode shapes and stress-strain contours of a model, Gnuplot should be already installed on the user's computer. During installation, the "add application directory to environment variable" box needs to be checked, as shown in Figure 2.56. If the installation is done properly, after writing the term "Gnuplot" in the command prompt (cmd), the user receives the feedbacks as "Terminal type is now 'wxt'" and " Encoding set to 'cp1252' " which confirms cmd is ready to receive Gnuplot commands and execute them without opening the Gnuplot interface.

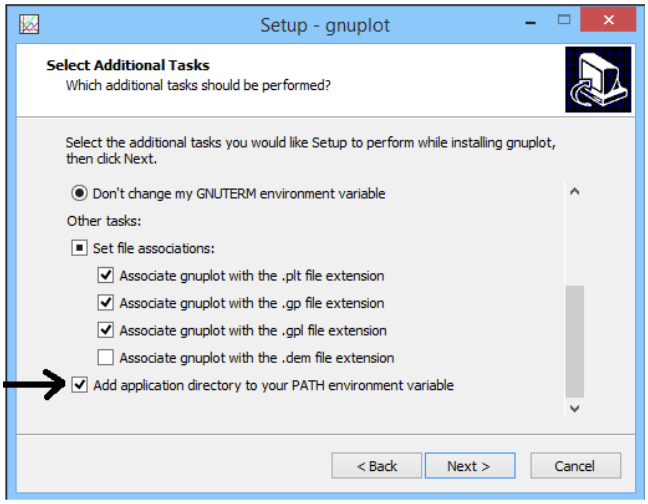


Figure 2.56. Gnuplot Installation Window

2.7.1 Mode Shapes of Models

In this part, how mode shapes of the structure are formed through the software is discussed. When analysis of the model is done through the engine of the software in the back-end part, text files representing mode shapes of the structures are created through the eigenvalue analysis. These text files will be sent to a folder named "Mode-shapes" in the software directory. As a front-end duty of software, the developed interface receives the created text files, and then the desired mode shapes are plotted using the embedded Gnuplot graphing utility. Finally, created plots are converted to JPEG image format and are sent to a folder named "Gnuoplots" in the software directory, so that the user can view the created mode shapes. For illustration, the first ten mode shapes of the model shown in Figure 2.57 is plotted through the software.

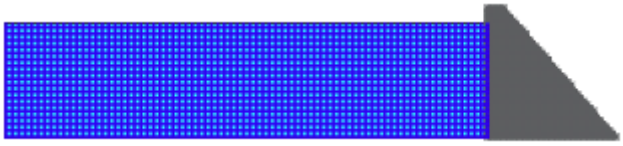


Figure 2.57. Sample Model for Mode Shape Plotting

After drawing the model and providing the related data using the interface, then analyzing the model through the engine, text files containing information regarding mode shapes of the model are received, and when the “Mode Shapes” option in edit tab of the software is clicked, specified mode shapes are created in the software directory, as shown in Figure 2.58. The zoom-in version of the first three modes of the model is shown in Figure 2.59, Figure 2.60, and Figure 2.61, respectively.It should be noted these mode shapes are presented for the user inside the local PC as an output sent to a specific folder and can not be presented in the graphical user interface environment.

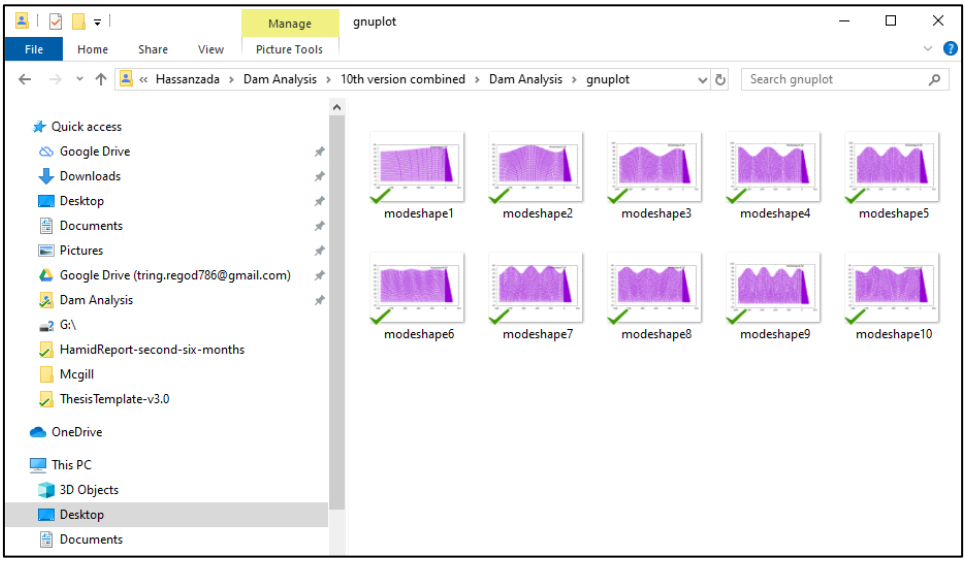


Figure 2.58. Folder Containing Mode Shape of the Model

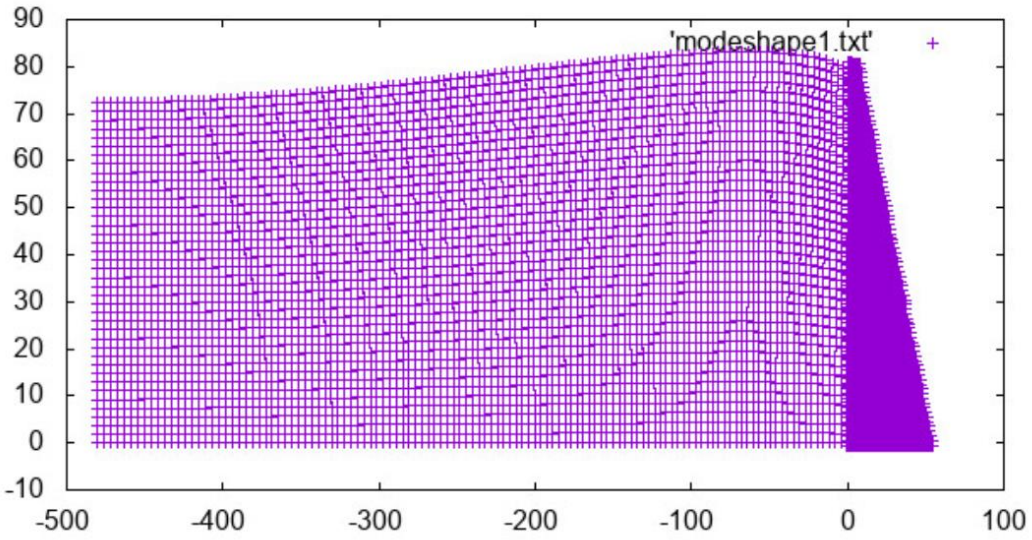


Figure 2.59. 1st Mode Shape of the Sample Model

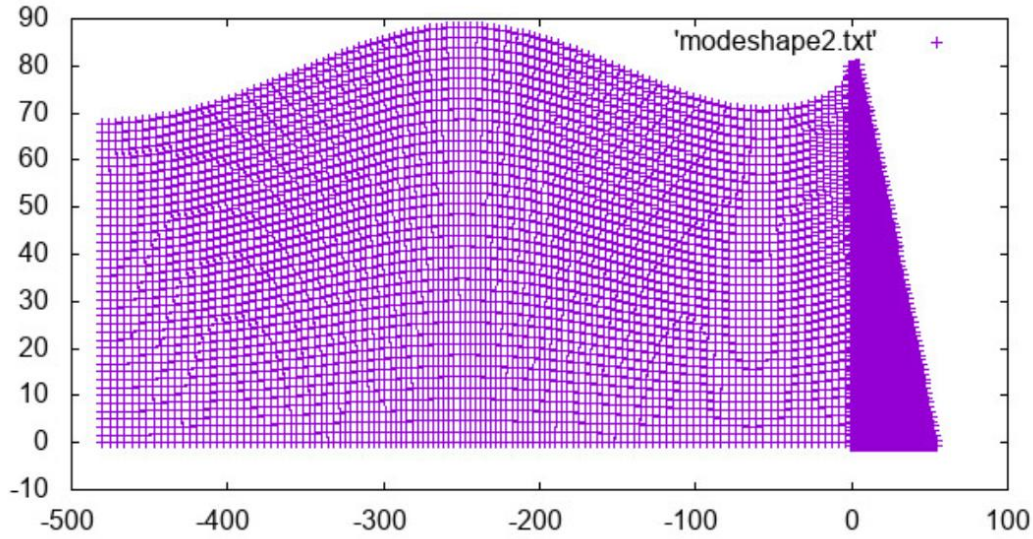


Figure 2.60. 2nd Mode Shape of the Sample Model

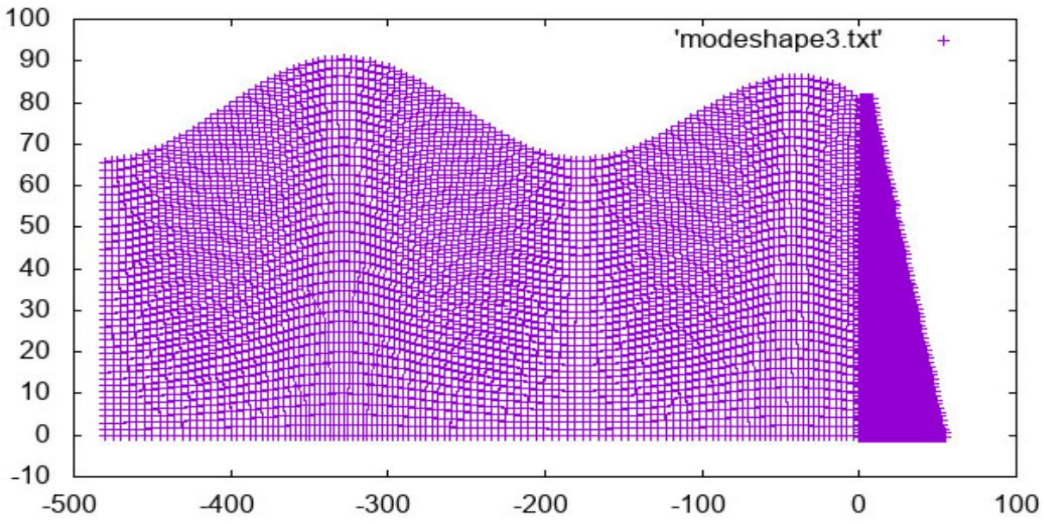


Figure 2.61. 3rd Mode Shape of the Sample Model

2.7.2 Stress and Strain Contours

Stress and strain values of models analyzed and obtained in the back-end part of the program can be plotted and presented as contour plots over the model, so parts of the model under extreme conditions can be visualized easily. As discussed before, applied elements used for meshing of the dam body are actually connected with several springs together vertically and horizontally. These springs undergo various strain values under different load conditions, which directly affect the stresses formed in different parts of the body of the model. For instance, the square element given in Figure 2.62 with a side length of 50 cm is connected to other elements with 4 spring horizontally and vertically. Suppose vertical and horizontal springs have different strain values under an applied load. Data format related to the horizontal and vertical strains of springs connecting the elements is shown in Table 2.9.

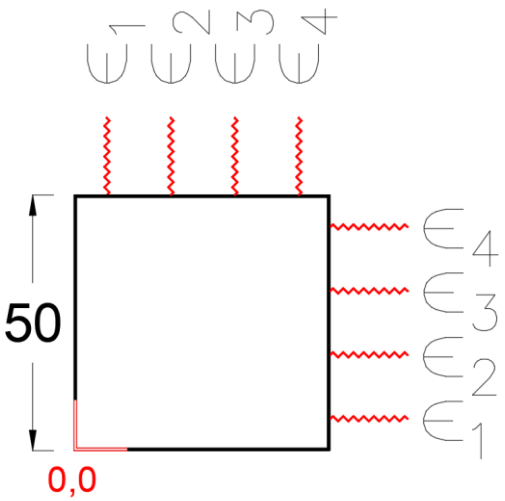


Figure 2.62. Strains of Springs Connecting the Elements

Table 2.9 Sample Vertical and Horizontal Springs Strains Data

		No	*X-Coordinates(cm)*	*Y-Coordinates (cm)*	*Strains*
Horizontal	Springs	1	50	6.25	6.63E-05
		2	50	18.75	5.46E-05
		3	50	31.25	4.29E-05
		4	50	43.75	4.15E-05
Vertical	Springs	1	6.25	50	4.56E-04
		2	18.75	50	3.81E-04
		3	31.25	50	3.06E-04
		4	43.75	50	2.52E-04

After obtaining the strain and corresponding stress values of each individual spring, text files containing this information are created in the format shown above and saved in the software directory. These text files give x, y coordinates of every single spring, and the corresponding strain or stress values. Finally, obtained results are plotted as contours over the model using the embedded Gnuplot. The strains of the normal springs in the horizontal and vertical directions are shown in Figure 2.63 for a pushover analysis. This figure clearly shows the propagation of a crack at the

bottom of the model, propagating from the upstream face to the downstream. The plots for the stress contours also can be obtained in a similar fashion.

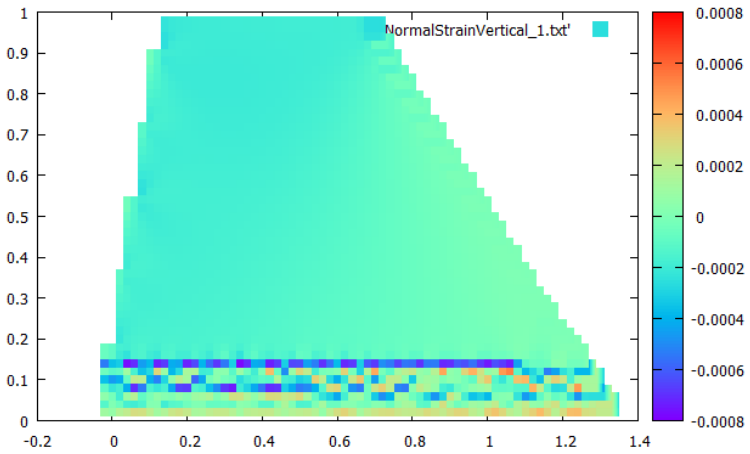


Figure 2.63. Strain in Y-Direction

CHAPTER 3

APPLICATION

In the previous chapter, the capabilities of the user interface were discussed. The models generated with the interface can be used for static and dynamic analysis of 2D nonlinear problems using applied elements. One of the goals of the interface is to create output files in neutral text format, so that the data can be passed to different solver engines for analysis. Within the scope of this study, a Matlab code focusing on the modeling of the rigid body motion of dam monoliths with contact capabilities was developed. The applied element model output from the interface is used to simulate sliding, rocking, and loss of stability of monoliths in extreme loading scenarios. In this chapter, first, the formulation and the procedures used during the coding of this solver are discussed in detail. Two benchmark problems, demonstrating the performance in the simulation of bouncing and rocking of a square block are then solved for validation purposes.

3.1 Applied Element Formulation

3.1.1 Stiffness and Mass Matrix for Applied Elements

As discussed earlier, applied elements are connected via shear and normal springs distributed around their boundaries. Each element has 3 degrees of freedom (translational in x-direction and y-directions, rotation around the z-axis). The number of connecting springs is specified by the user where for each pair of springs at the contact points, a 6x6 stiffness matrix is calculated. In this formulation (Tagel-Din and Meguro, 1997), the stiffness of the element is defined by the location of the contact point specified by distance L, α and θ angles, and the stiffness of normal and shear springs at the interface connected to the next element. The total stiffness

contribution of this connection to the global stiffness is obtained by summing the stiffness matrices of each individual spring pair at this connection. The total stiffness matrix of the structure is obtained by summing up the stiffness contributions of each of these interfaces for the whole structure. The schematic view of two elements connected at one contact point on an element-element interface is shown in Figure 3.1.

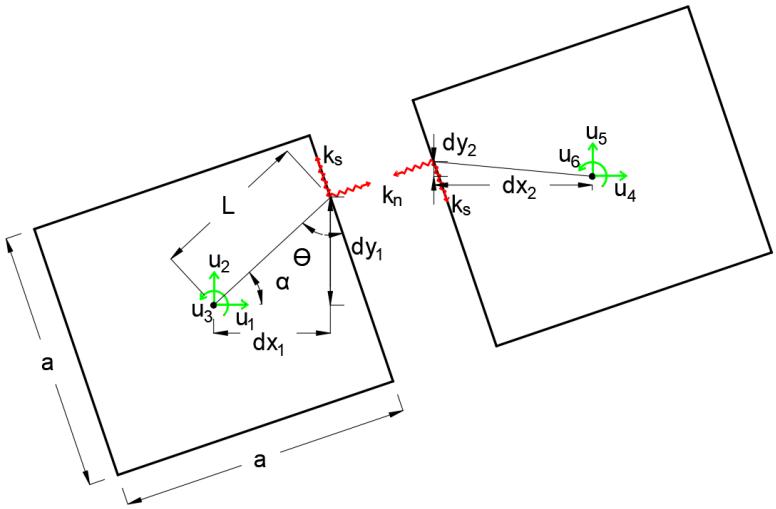


Figure 3.1. Connection of the Applied Elements at the Contact Point

Each pair of springs are responsible for some portion of the elements' contact; the normal and shear stiffness of the springs at the contact point are calculated according to Equations 3.1 and 3.2, respectively (Tagel-Din and Meguro, 1997). In these equations, d, T and a represent the distance between the springs at the interface, the thickness of the element, and the size of the element, respectively. The normal and shear stiffness are calculated using Young's modulus E and shear modulus G of the material for the normal and shear stiffnesses, respectively. Combining all the variables for the Figure 3.1, the upper left quarter of the stiffness matrix is obtained, as shown in Equation 3.3.

$$k_n = \frac{E\ d\ T}{a} \qquad 3.1$$

$$k_s = \frac{G\ d\ T}{a} \qquad 3.2$$

$$\begin{bmatrix} sin^2(\theta+\alpha)k_n & -k_n \sin(\theta+\alpha)\cos(\theta+\alpha) & k_s \sin(\alpha)\cos(\theta+\alpha)\,L \\ +cos^2(\theta+\alpha)k_s & +k_s \sin(\theta+\alpha)\cos(\theta+\alpha) & -k_n \cos(\alpha)\sin(\theta+\alpha)\,L \\ & sin^2(\theta+\alpha)k_s & k_n \cos(\alpha)\cos(\theta+\alpha)\,L \\ & +cos^2(\theta+\alpha)k_n & k_s \sin(\alpha)\sin(\theta+\alpha)\,L \\ & & sin^2(\alpha)k_s\,L^2 \\ & & +cos^2(\alpha)k_n L^2 \end{bmatrix} \qquad 3.3$$

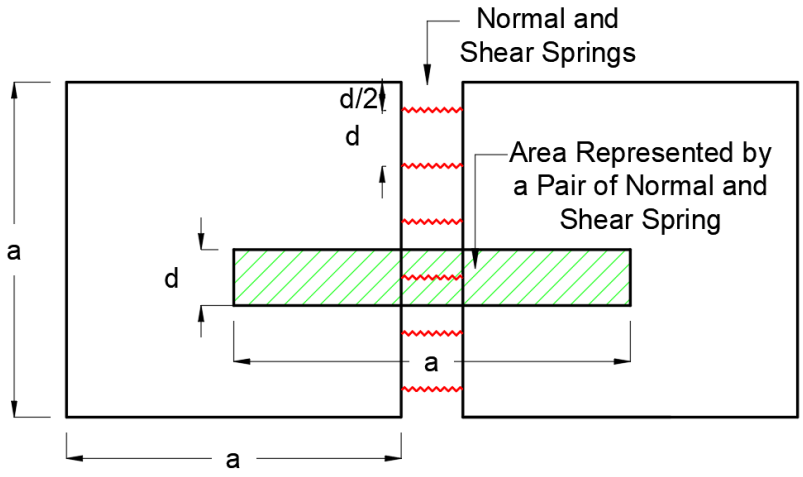


Figure 3.2. Springs on Applied Elements' Interface and their Respective Influence Area

Masses of the discretized elements are lumped at the centers of the elements, as shown in Equation 3.4, where D, t, and ρ represent the size, thickness, and the density of the element, respectively. The element mass is calculated as M_1 and M_2 in x and y directions coupled with the rotational mass M_3 corresponding to the mass moment of inertia around the element's center.

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} D^2 \times t \times \rho \\ D^2 \times t \times \rho \\ D^2 \times t \times \rho/6 \end{bmatrix} \quad 3.4$$

3.1.2 Newmark Implicit Algorithm for Solution of Dynamic Equilibrium

For the solution of the equation of motion, Newmark implicit algorithm was adopted (Chopra,2012). Knowing the initial displacement $\boldsymbol{u_0}$ and initial velocity $\dot{\boldsymbol{u}}_{\boldsymbol{0}}$, the equation of motion (Equation 3.5) at t=0 is solved for the initial acceleration $\ddot{\boldsymbol{u}}_{\boldsymbol{0}}$, where $\boldsymbol{P_0}$, $\boldsymbol{m}$,$\boldsymbol{c}$ and $\boldsymbol{k}$ are initial force vector, mass, damping, and stiffness matrices, respectively. After selecting a proper time step, variables a and b are determined from Equations 3.6 with constant average acceleration method (γ =1/2, β =1/4).

$$\ddot{\boldsymbol{u}}_{\boldsymbol{0}} = \frac{\boldsymbol{P_0} - \boldsymbol{c}\dot{\boldsymbol{u}}_{\boldsymbol{0}} - \boldsymbol{k}\boldsymbol{u_0}}{\boldsymbol{m}} \quad 3.5$$

$$a = \frac{1}{\beta \Delta t}\boldsymbol{m} + \frac{\gamma}{\beta}\boldsymbol{c} \qquad b = \frac{1}{2\beta}\boldsymbol{m} + \Delta t\left(\frac{\gamma}{2\beta} - 1\right)\boldsymbol{c} \quad 3.6$$

The tangential stiffness matrix ($\widehat{\boldsymbol{k}}$) at the start of the time step is calculated via Equation 3.7, where $\boldsymbol{k}$,$\boldsymbol{c}$ and $\boldsymbol{m}$ are the static stiffness, damping, and mass matrices, respectively.

$$\widehat{\boldsymbol{k}} = \boldsymbol{k} + \frac{\gamma}{\beta \Delta t}\boldsymbol{c} + \frac{1}{\beta(\Delta t)^2}\boldsymbol{m} \quad 3.7$$

Using Equation 3.8, the geometric residual force vector (F_G) is calculated as the difference of the total external force ($f(t)$) at the current time step minus the inertial, damping, and internal spring forces (F_m) obtained at the center of each element.

$$\boldsymbol{F_G} = \boldsymbol{f(t)} - \boldsymbol{m}\ddot{\boldsymbol{u}} - \boldsymbol{c}\dot{\boldsymbol{u}} - \boldsymbol{F_m} \quad 3.8$$

Forcing vector ($\Delta\hat{P}_i$), which is the summation of the incremental external force, damping, and inertial force at the current time step, and the total residual forces for the whole time is calculated via Equation 3.9.

$$\boldsymbol{\Delta\hat{P}_i = \Delta f(t) + F_G + a\dot{u}_i + b\ddot{u}_i} \qquad 3.9$$

With the $\hat{\boldsymbol{k}}$ and $\boldsymbol{\Delta\hat{P}_i}$, known from the system properties $\boldsymbol{m}$, $\boldsymbol{k}$ and $\boldsymbol{c}$, algorithm parameters γ and β, and the state of the system at the current time step defined by $\boldsymbol{u_i}, \boldsymbol{\dot{u}_i}, \boldsymbol{\ddot{u}_i}$, the change in displacement is calculated via Equation 3.10.

$$\boldsymbol{\Delta u_i = \hat{k}^{-1}\,\Delta\hat{P}_i} \qquad 3.10$$

Once the incremental displacement is known, the change in velocity and acceleration is obtained via Equations 3.11 and 3.12, respectively.

$$\boldsymbol{\Delta\dot{u}_i} = \frac{\gamma}{\beta\Delta t}\boldsymbol{\Delta u_i} - \frac{\gamma}{\beta}\boldsymbol{\dot{u}_i} + \Delta t\left(1 - \frac{\gamma}{2\beta}\right)\boldsymbol{\ddot{u}_i} \qquad 3.11$$

$$\boldsymbol{\Delta\ddot{u}_i} = \frac{1}{\beta(\Delta t)^2}\boldsymbol{\Delta u_i} - \frac{1}{\beta\Delta t}\boldsymbol{\dot{u}_i} - \frac{1}{2\beta}\boldsymbol{\ddot{u}_i} \qquad 3.12$$

The displacement, velocity, and acceleration at the next time step are calculated via Equation 3.13. The procedure is repeated for all time steps.

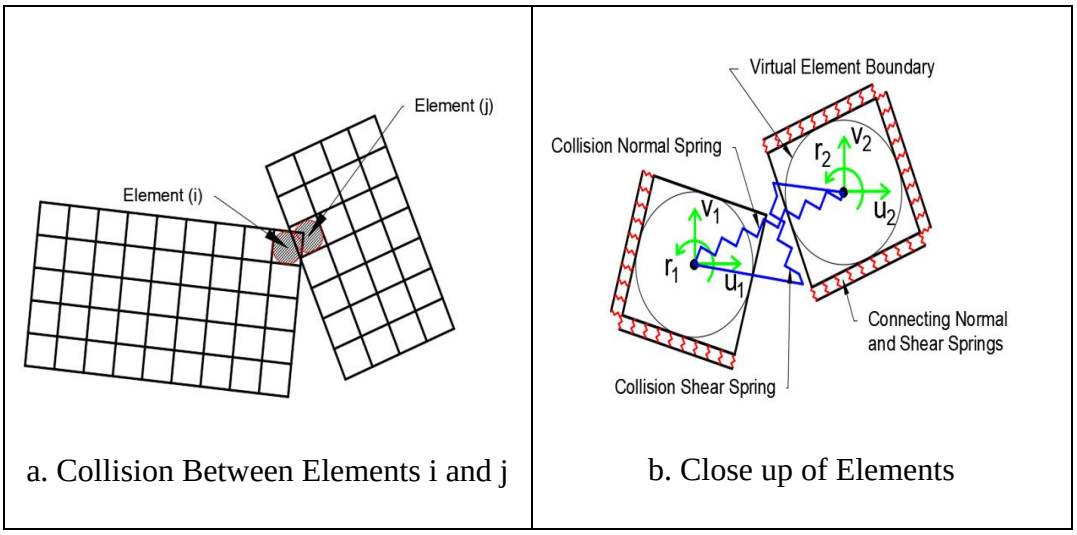
$$\boldsymbol{u_{i+1} = u_i + \Delta u_i} \qquad \boldsymbol{\dot{u}_{i+1} = \dot{u}_i + \Delta\dot{u}_i} \qquad \boldsymbol{\ddot{u}_{i+1} = \ddot{u}_i + \Delta\ddot{u}_i} \qquad 3.13$$

3.2 Modeling of Collision and Re-Contact Process

The numerical approach proposed by Tagel-Din et al. (1999) was implemented to simulate the collision and recontact of bodies. In this model, the contact and separation of individual bodies is treated by the formation of new contact interfaces between colliding elements which simulate the collision behavior in accordance with impact conditions. An important part of the methodology is the contact detection scheme: in order to simulate the collision of separated parts of the model, the distance between the centers of the elements are checked at each time step. In order to

simplify the contact algorithm, the boundaries of the applied elements are assumed as circles at contact boundaries, assuming small element sizes, directly allowing the distance between the centers of the elements to define the contact conditions. This assumption was justified for large elements given the abrasion of the sharp corners of individual objects due to stress concentrations during impact (Tagel-Din and Meguro, 1999).

The collision process of the applied elements is shown in Figure 3.3. In the deformed geometry, the normal spring is formed through the center of the elements defining the contact axis while the shear spring is formed in the perpendicular direction to this axis.



a. Collision Between Elements i and j

b. Close up of Elements

Figure 3.3. Collision Process of Applied Elements

3.2.1 Contact Springs Modeling

For the modeling of contact springs, the following assumptions are used:

1. Normal spring stiffness is obtained using Equation 3.14, where E is the modulus of elasticity, t thickness of the model, D distance between the centers of the collided elements, and d is the contact distance, which is assumed as 10 % of the element size.

$$K_n = \frac{E \times d \times t}{D} \qquad 3.14$$

2. The shear stiffness value is assumed as 1% of the normal spring.
3. Failure of the contact springs is not allowed as the idea of the contact springs is to transmit the stress wave propagation from one element to another. Therefore, there is no nonlinearity in the contact spring, and its stiffness is constant. Nonlinearity of the system occurs due to the contact and loss of contact.
4. Tension force in the contact spring is not allowed, tension force indicating the separation of the elements. Therefore, contact springs should be removed at the moment contact force changes sign from compression to tension.

3.2.2 Energy Dissipation during the Collision Process

During the collision of bodies, some of the kinetic energy is inevitably lost. This energy loss is usually represented in terms of a coefficient of restitution or a rebound factor (r) defined as the ratio of the velocity before and after the collision. However, this approach requires very small time increments to simulate the transmission of stress waves between the colliding elements (Tagel-Din and Meguro, 1999). A different approach is suggested in (Tagel-Din and Meguro, 1999) for modeling the loss of the energy during the collision utilizing different loading and unloading stiffnesses for the contact spring defined with the factor (n), the ratio between the unloading and loading stiffness of the contact spring. This approach was determined to be more time-efficient compared to the former approach as it needs small time increments only during the collision process. When elements are separated, much larger time increments can be used.

The coefficient of restitution for contact between the two elements (the rebound factor) and the unloading stiffness factor is given in Equation 3.15. A schematic view of the application of the n factor for the loading/unloading behavior of the contact stiffness in normal direction is shown in Figure 3.4. For realistic contact

behavior, n value should be greater than 1. The case n=1 corresponds to conservation of energy during collision with the contact spring following the same path during loading-unloading within the impact. A very high n value corresponds to the case for which the force in the contact spring drops very quickly during the impact with the spring consuming the initial load-displacement energy.

$$r = \frac{1}{\sqrt{n}} \qquad 3.15$$

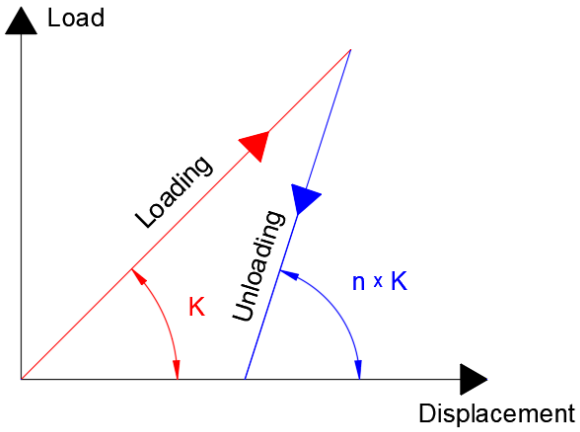


Figure 3.4. Contact Spring in Loading and Unloading Condition

The colliding of two bodies is simulated using the following procedure. If the structure is initially in a static condition, Equation 3.16 is first solved for the static loads where K is the stiffness matrix, ΔF_s the incremental static load vector, and $[\Delta U]$ is the incremental displacement vector.

$$[K]\,[\Delta U] = \Delta F_s \qquad 3.16$$

The dynamic equilibrium is attained through time-stepping given below:

1. The dynamic loading is initialized with a time increment Δt. For this loading, equation of motion given in Equation 3.17 is solved using the Newmark Beta technique, and the corresponding displacements are obtained. In this

equation, M, C and K are the mass, damping and stiffness matrices, respectively, and the incremental acceleration $[\Delta\ddot{u}]$, velocity $\Delta\dot{u}$ and displacement Δu at the current time step is sought for the current time step for the incremental change in loading, $\Delta f(t)$. The dynamic stiffness matrix is represented by $\hat{k}^{-1}$ and is obtained using Eq. 3.7.

$$[M][\Delta\ddot{u}] + [C][\Delta\dot{u}] + [K][\Delta u] = \Delta f(t) \qquad 3.17$$

$$\Delta u = \hat{k}^{-1}\Delta f(t) \qquad 3.18$$

2. Elements coordinates are updated according to the calculated displacements, and the deformed geometry is obtained.
3. According to the new deformed geometry, directions of the spring forces are modified and internal forces are obtained as vector F_m.
4. The updated model is checked for contact during each time step:
   - If a new contact arises, the time increment is reduced to accurately trace the material behavior. Normal and shear collision springs are added between the collided elements, as shown in Figure 3.3.
   - If an old contact expires, when elements separate, collision springs are removed.
5. For each applied element, the resultant of the spring forces, including the collision springs, is obtained at the center of each element and finally added into the vector F_m .
6. A residual force is calculated in order to include the effect of reorientation of forces due to updated geometry and the formation/loss of contact forces. This term, F_G, the geometric residual is calculated as the difference of the total external force ($f(t)$) at the current time step minus the inertial, damping and internal spring forces based on total displacement/velocity quantities.

$$F_G = f(t) - [M][\ddot{U}] - [C]\left[\dot{U}\right] - F_m \qquad 3.19$$

7. A new stiffness matrix (and corresponding dynamic stiffness matrix, $\hat{k}$) is calculated for the updated configuration. Equation 3.20, the updated form of

Equation 3.17 with the residual forces, is solved to obtain the new estimates for the incremental displacement, velocity, and acceleration vectors.

$$[M][\Delta\ddot{u}] + [C][\Delta\dot{u}] + [K][\Delta u] = \Delta f(t) + F_G \qquad 3.20$$

$$\Delta u = \hat{k}^{-1}(\Delta f(t) + F_G) \qquad 3.21$$

8. A new time increment is started for continued time stepping; Apply a new increment and go to step 3.

The diagram showing the overall simulation process is shown in Figure 3.5. Steps 9 and 10 correspond to the solution of nonlinear effects in the process for material and geometric nonlinearities.

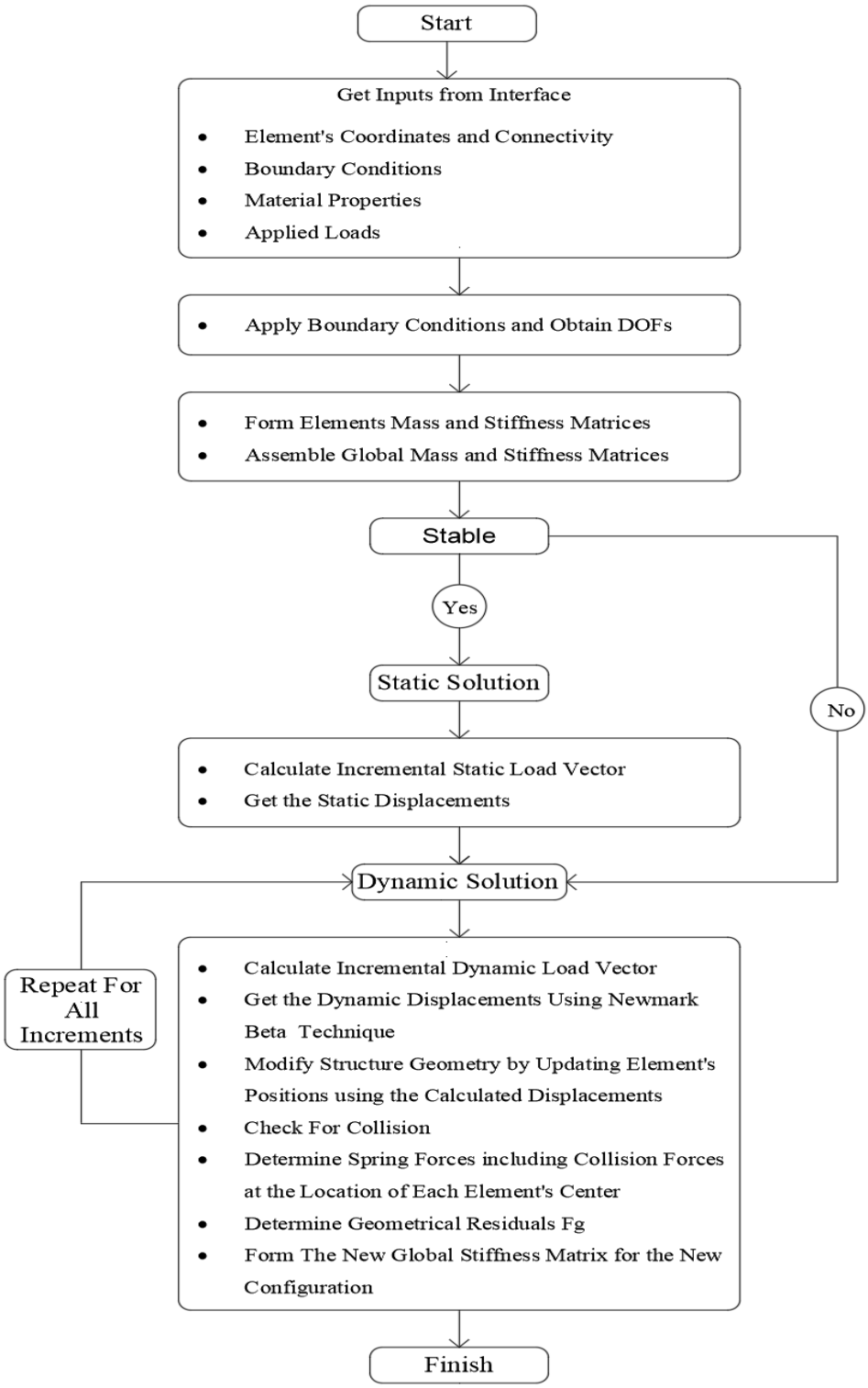


Figure 3.5. Overall Process of Simulation

3.3 Tracing of Boundaries for Contact Checks

Contact check is a numerically rigorous problem. For small deformations, the check of only neighboring elements for contact is reasonable. However, for cases of large deformation and separation of parts, different elements can collide with each other. Checking all possible contact scenarios, a rather daunting task, requires a computational effort proportional to (NxN) where N is the number of elements and increases the CPU time significantly (Tagel-Din and Meguro, 1999). Correspondingly, a prior knowledge of the possible regions of contact is very beneficial if the particular problem allows for the determination of such locations. The current problem of the modeling of the behavior of dam monoliths is one such problem in which a major part of the monolith is expected to be severed from the rest moving independently during an earthquake event. As such, if the contact path of the severed monolith is determined pre-hand based on the cracking of the monolith, the contact problem is simplified to checking two separate bodies at the crack path from the upstream to the downstream surface.

In this section, a new algorithm developed for tracking the individual crack paths inside a 2D closed geometry is presented. On any concrete monolith under extreme loading conditions, there are a range of cracks starting from the upstream and downstream sides, as well as bifurcations of these cracks within the body. A crack path from the upstream to downstream surface splitting the monolith into two separate blocks is sought in this algorithm.

In order to illustrate the crack propagation scenario, a simple rectangular block meshed with several applied elements (Figure 3.6) is considered. When the model is loaded, cracks initiate at the boundaries of the model. As the loading is continued, the initiated cracks start to propagate in different paths. The red lines in the figure represent the interfaces between applied elements on which all the springs failed. At later load increments, new cracks are formed, finally culminating in the last state in which the cracking between the upstream and downstream directions form a continuous but unknown path (Figure 3.7). A continuous path, as such, means the

elements above this path form a body completely separated by a crack from the rest of the monolith.

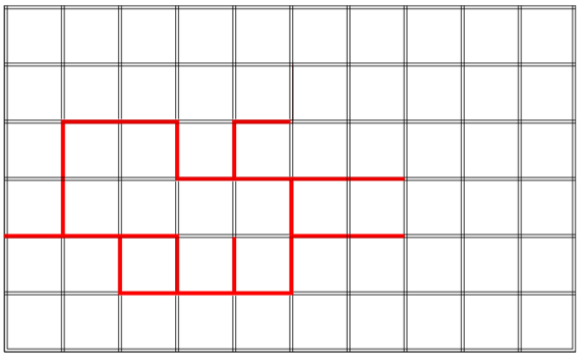


Figure 3.6. Propagation of Crack in a Simple Block

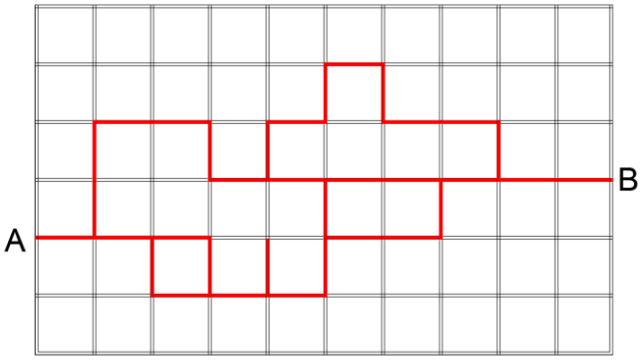


Figure 3.7. Formation of Separate Blocks Due to Crack Propagation

In order to determine the presence of a severed monolith, the available crack paths should be quantified and updated at every load increment and all the available paths need to be controlled. For a severely damaged system, the presence of several different paths connecting points A to B is also evident (Figure 3.7). While these paths form various closed geometries on the geometry, the primary interest is on the topmost severed block. Consequently, the goal of the algorithm is to identify the crack paths from the upstream to downstream surface of the monolith determining the critical path forming a severed block at the top of the monolith.

3.3.1 Discretization & Data Structure

Tracing of cracking requires a further discretization of the model at the interfaces using the applied element model setup. To demonstrate this discretization, a sample model formed of 21 applied elements is formed, as shown in Figure 3.8. As discussed in Chapter 2, during the mesh generation, the elements are numbered row-wise from the bottom up to the top of the model with the origin located at the center of the first element. The coordinates of the centers of the elements are also shown in the figure: for example, the center of element 6 lies at x=50 y=0.

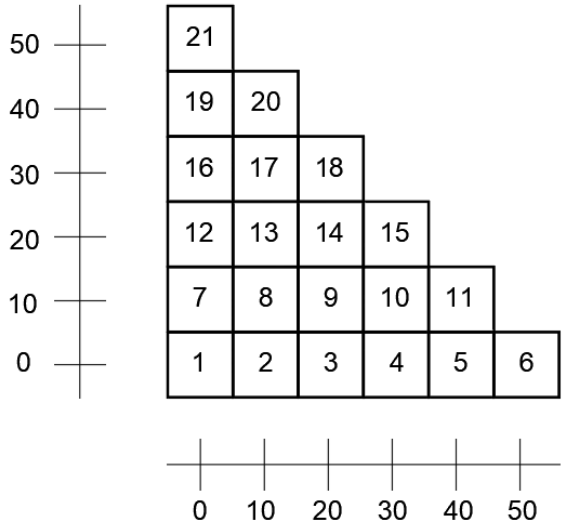


Figure 3.8. Discretization of the Model and Element Numbering

The tracing of crack paths involves searches within the element data in every cycle of iterations. Therefore, management and organization of this discretized data during the coding is crucial for enabling efficient access and modification of the data for later use. In order to enhance the access to the data, elements' information is stored in two different formats. For every element, the element ID, X-coordinate, and the Y-coordinate are available as text files received from the user interface after the discretization process. In the first format, data are structured row-wise or according to their Y-Coordinates (elements 1-6 in Figure 3.8). In the second format, data are structured column-wise or according to their X-Coordinates (elements 1,7,12,16,19

and 21 in Figure 3.8). These structures enable ease of access to the ID of any element knowing its coordinates, avoiding searches through all the data. While this type of data structuring may seem redundant at this stage, for models with small element sizes, it becomes very beneficial reducing the algorithm runtimes. The application of this data structure to the sample model nodal data is given in Table 3.1. Every array in Table 3.1 has the size equal to the number of elements row/column-wise and their respective ID, X-coordinate, and Y-coordinate. The 5x3 array containing the elements with X-coordinate of 10 and the 3x3 array for elements with Y-coordinate of 30 are expanded and shown in Table 3.2.

Table 3.1 Elements Sorted by X and Y Coordinates

Stored By X	*Array*	*Stored By Y*	*Array*
0	6x3	0	6x3
10	5x3	10	5x3
20	4x3	20	4x3
30	3x3	30	3x3
40	2x3	40	2x3
50	1x3	50	1x3

Table 3.2 Expanded Format of the Elements Information

Elements with X-Coordinates of 10			Elements with Y-Coordinates of 30		
ID	*X*	*Y*	*ID*	*X*	*Y*
2	10	0	16	0	30
8	10	10	17	10	30
13	10	20	18	20	30
17	10	30			
20	10	40			

As it was discussed before, applied elements have their nodes located at the center of the elements. However, since the cracking occurs along the boundaries of the elements where they are connected with springs, fictitious nodes are defined at the corners of the elements for use in the crack detection algorithm. These nodes are numbered similarly row-wise, starting from the bottom left to the top-right of the model, as shown in Figure 3.9. The data structure given above is also applied to storing the fictitious nodes' information, i.e., their ID and respective X and Y coordinates.

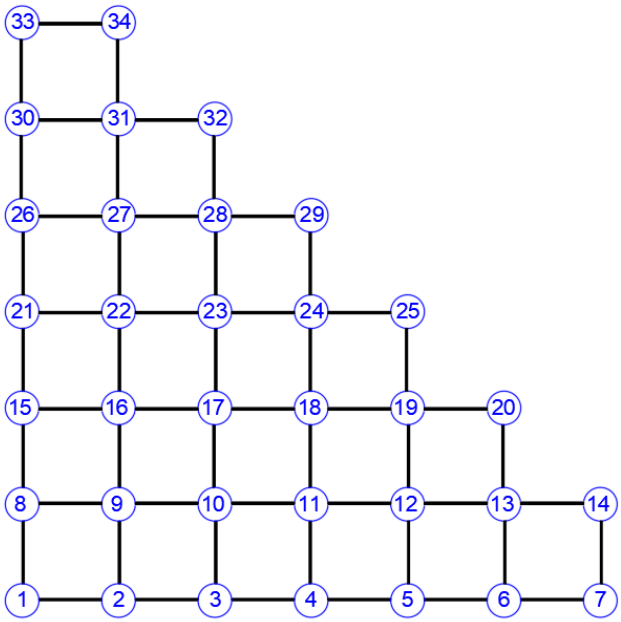


Figure 3.9. The Numbering of the Fictitious Nodes

Applied element surfaces are assigned ID's treating the horizontal and vertical line segments between the elements. Horizontal interfaces are assigned with ID numbers increasing row-wise from bottom left to the top-right of the model, while the vertical surfaces are assigned ID numbers increasing column-wise from bottom left to the top-right. The numbering of the line segments for the sample model in Figure 3.8 is shown in Figure 3.10. Combining all the above data together, a detailed discretization of the model in terms of corners and element surfaces, which can be used for tracing of the cracking, is obtained as given in Figure 3.11.

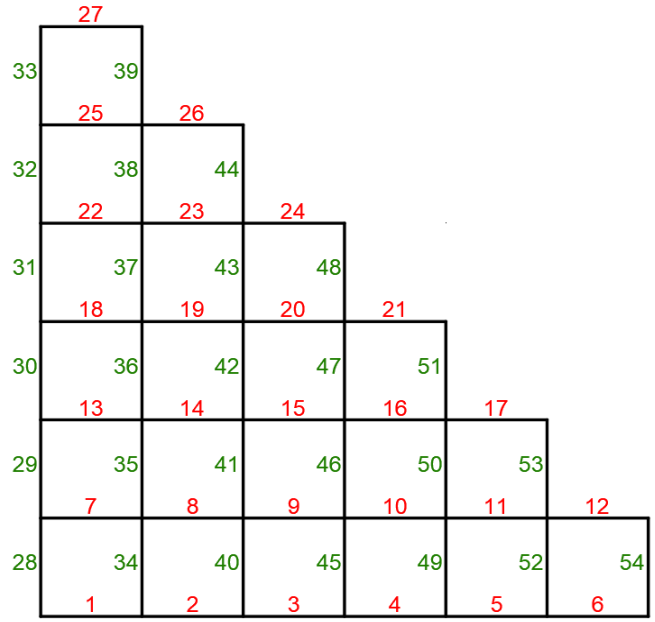


Figure 3.10. The Numbering of Constructing Line Segments

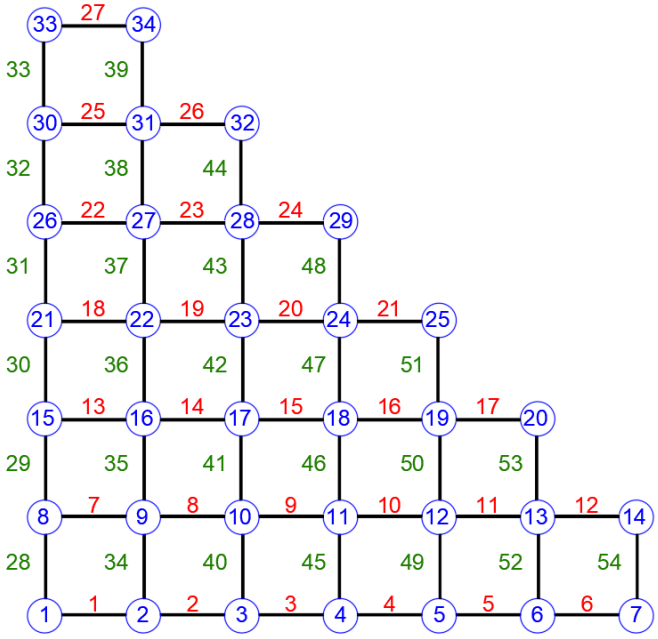


Figure 3.11. All the Discretization Information for Crack Tracing

During loading, some of the springs connecting the applied elements fail. As all boundaries are defined by line segments, the boundaries where all springs fail are

easily tracked. Line segments are assigned a counter value of 1 or -1 representing the condition of the springs along the segment, -1 representing failure of all springs in the interface while 1 represents all other conditions. Initially, all segments start from the intact condition, i.e. counter is assigned as 1 to all the boundaries. During each iteration, the conditions of the springs at the boundaries are checked. If all the springs along a line segment fail, then the condition value of the segment is updated to -1.

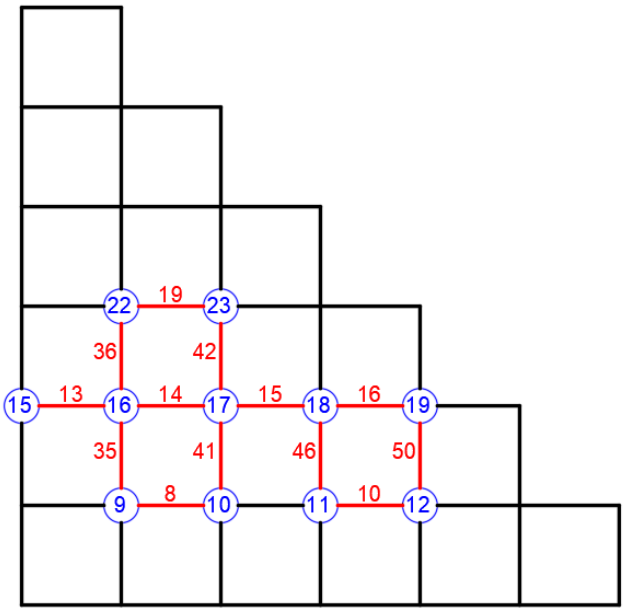


Figure 3.12. Cracked Model

A sample plot of failed interfaces on the demonstration model is shown in Figure 3.12. Using the IDs of a failed line segment, the start and end node number of segments can be traced and stored in an array in the format shown in Table 3.3. Tracing the crack paths requires the information on the start and end nodes of the path: consequently, the nodes located in the boundaries of the model in the upstream and downstream face are also stored in a separate array.

Table 3.3 Array Containing Cracked Interfaces Data

Line ID	Starting Node	Ending Node	Spring Condition
8	9	10	-1
10	11	12	-1
13	15	16	-1
14	16	17	-1
15	17	18	-1
16	18	19	-1
19	22	23	-1
35	9	16	-1
36	16	22	-1
41	10	17	-1
42	17	23	-1
46	11	18	-1
50	12	19	-1

3.3.2 Crack Tracing Algorithm

The procedure devised to extract continuous paths from the downstream to upstream faces of the monoliths is next presented. For this purpose, the model shown in Figure 3.12 with existing cracks is used. Before tracing of the cracks, initial information on the model is prepared in forms of the interface, fictitious node, and boundary node arrays with the aforementioned data structure. As the loading progresses in load or time steps, cracks form on the model: corresponding interfaces with failed springs are stored. The problem then on is to identify the crack paths starting from a (fictitious) node on the upstream or downstream surface and propagates towards the other side. As given in the sample crack scheme in Figure 3.12, the snapshot of the model at any given instant may have a number of failed interfaces, which also corresponds to numerous different paths connecting the upstream to the downstream

face: for example, in this figure, there are six different paths from the node 15 to 19 if backward movement on the path is not allowed. These paths represented in terms of fictitious nodes are summarized in Table 3.4.

Table 3.4 Crack Paths for the Cracked Model

Paths	*Nodes*							
1	15	16	17	18	19			
2	15	16	17	11	12	19		
3	15	16	9	10	17	18	19	
4	15	16	22	23	17	18	19	
5	15	16	9	10	17	11	12	19
6	15	16	22	23	17	11	12	19

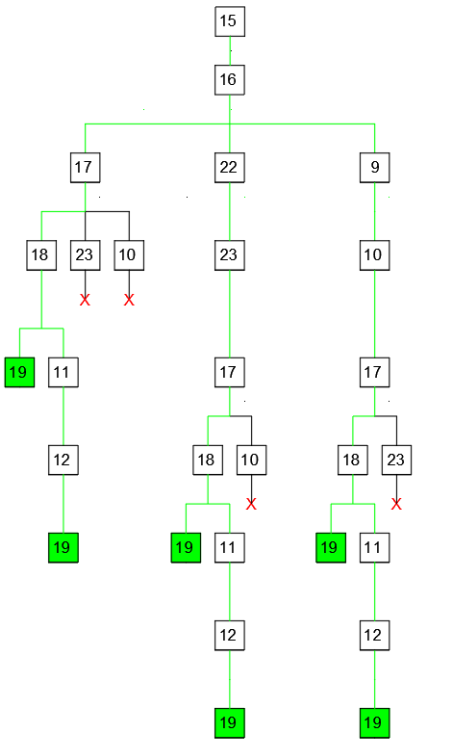
The goal of the algorithm presented here is to find the uppermost crack on the monolith. This corresponds to finding a continuous path between the upstream and downstream faces that is located at the highest point in the monolith, creating a block at the top, separated from the rest of the body. The crack tracing algorithm starts from the upstream side towards the downstream face. A backward movement in the path would lead to infinite crack paths with possible return on the same path and is not allowed.

Each node from the upstream face is checked at the CID data. If one of the start nodes in cracked interfaces data (CID, such as Table 3.3) coincides with the upstream nodes, that node is taken as the start of a crack path. It can be seen clearly that for this instant (Figure 3.12), node 15 is the start node of a crack. The row corresponding to node 15 in CID is then located and the end node (16) is determined. The connections of every node are traced then in a step by step procedure: the number of repetitions for the node as start or end nodes is sought, indicating the number of possible paths to move forward. Node 16 is repeated 4 times in the CID, twice both as a start and end node. For a start node, a lateral forward movement corresponds to node number increasing by one (case 1); if the difference is larger than one (case 2),

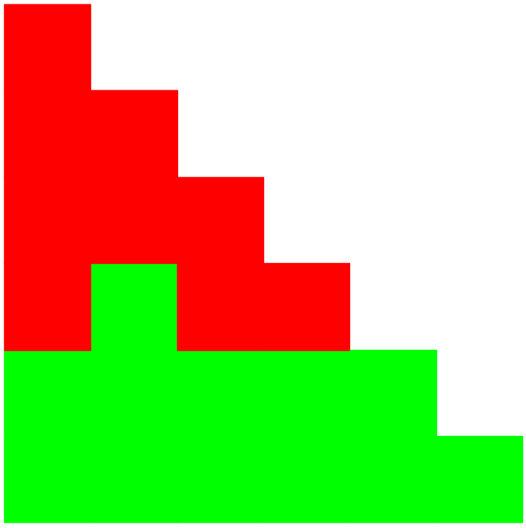
an upward movement is indicated. For an end node, a lateral backward movement corresponds to the start node one less than the end node (case 3), a downward movement, on the other hand, corresponds to a difference larger than one with the end node (case 4). The information on the crack tip from a node is processed, dealing with the first three cases. Starting from node 16, movement is possible in forward, upward and downward directions, reaching nodes, 17,22, and 9, respectively. The possible movement directions for these nodes can also be established in the same fashion with each new segment creating a separate new path. Each new node is checked for boundary conditions; if the node exists on the downstream end, a distinct crack path, joining an upstream node to a downstream node is created.

A diagram showing all crack paths obtained as such by sequential application of this procedure is presented in Figure 3.13a. The continuous downward flow from the start node 16 to the end node marked with green box (19) shows the possibility of six different paths to arrive from 15 to 19 without going backwards on the grid. As the goal is to determine the topmost individual block, the area of the blocks above each individual crack path is calculated, and the crack path corresponding to the minimum area is selected as the final path. Therefore, among the six crack paths determined, the one passing through the nodes 15,16,22,23,17,18 and 19 is selected as the dominant crack path, as shown in Figure 3.13b.

For the case of having multiple crack paths that can divide the model into a separate block, all such cracks are traced and stored first. Among the stored paths, those with the largest starting node is extracted, and then the one with the minimum area is selected as the dominant path. During each time step, new crack interfaces are added to the cracked interface data (CID, such as Table 3.3), and the process is repeated.



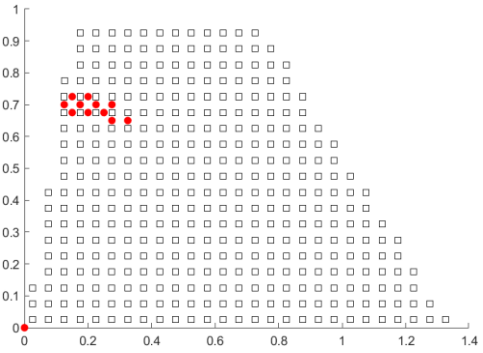
a. Diagram of Crack Paths



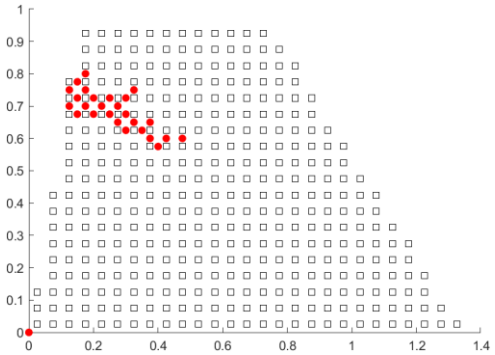
b. The Top Most Block of the Cracked Model

Figure 3.13. Crack Paths Diagram and Top Most Block

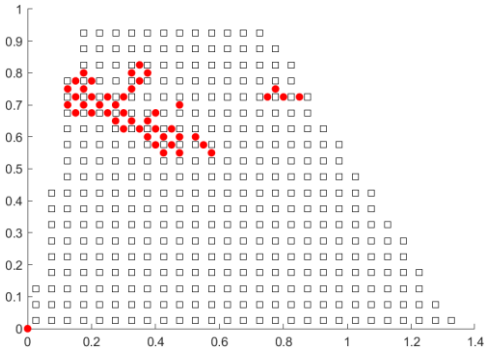
Following the discussed algorithm, the 1/75 scale model of the 120 m high Melen Dam was used. The upstream face of the crest was subjected to a relatively high cyclic load (F = 4e09 t (N/sec)), forcing the cracks to propagate toward the downstream face of the model. The loading information does not represent any experiment set up, and it is applied only for the verification of the proposed algorithm in tracing the complex cracked models. The process of crack tracing for the model is shown in Figure 3.14.



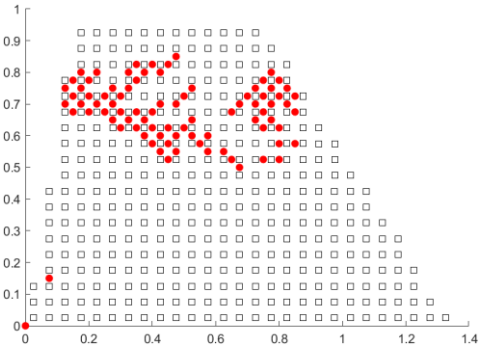
a. Crack Initiation



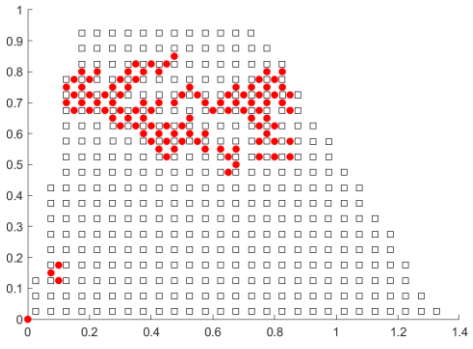
b. Crack Propagation



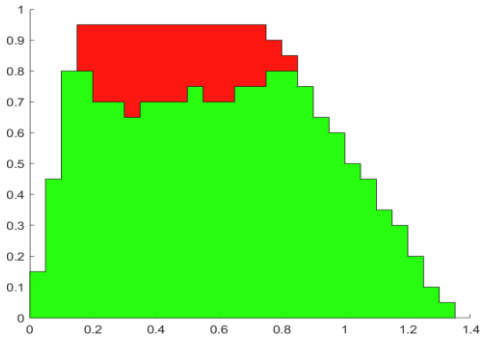
c. Crack Initiation at Downstream



d. Cracks Approaching Each Other



e. Cracks Meet Each Other



f. Dominant Crack Path

Figure 3.14. Finding the Dominant Crack Path

3.4 Benchmark Models

In this section, problems involving large rigid body motions of monoliths are solved as benchmark problems in order to validate the tools developed in the study.The first two validation problems are selected from cases that can be compared to the analytical solutions. The latter validation problems are numerical solutions from the literature for the complex rocking problem. The first problem presented is the free fall of a block under its own weight leading to a direct impact to the ground and the subsequent motion dependent on the coefficient of restitution between the block and the ground. Next, rocking response of a square block under an impulsive load is tackled. Finally, the response of a severed block on top of an intact monolith is investigated under impulsive loads. The results of the analysis were compared to benchmark values if available for verification.

3.4.1 Free-Falling and Bouncing Response of a Square Block

The first group of analyses conducted for verification purposes of the applied element model, and the dynamic solution algorithm was selected as the free-fall and bouncing response of a block under its own weight. In the first stage of this study, the free-fall and the bouncing of a single element off the ground was simulated with own weight of the block as the only external load. A 1m square (single) element with a thickness of 0.25 m, a density of 2500kg/m^3, and modulus of elasticity of 20 GPa was released from a height of 4m above the ground as shown in Figure 3.15. At the instant of the collision with the ground, contact springs are added to the model, which are removed upon the separation. Numerical time stepping was conducted with two different time steps in the analysis: a time step of dt=0.00001 was used during the contact while a much larger time step of dt=0.001 was used elsewhere speeding up the analysis. The damping ratio is zero in this analysis. Instead, the loss of energy is simulated by using different loading/unloading stiffnesses during the collision. The model was analyzed with a series of unloading factors (n) to check the effect of

unloading stiffness factor on the energy dissipation during the motion. The displacements of the block before and after the collision for different unloading stiffness values are shown in Figure 3.16.

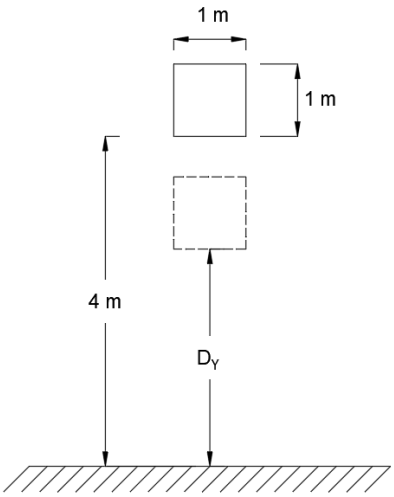


Figure 3.15. Free Falling Element Under its Own Weight

It can be easily noticed that, when the unloading factor is selected as n=1, the block does not dissipate energy and attain the same height after each impact. However, for larger values of n, the height of the bounce is reduced after each rebound depending on the unloading factor: damping of the element's motion is much faster for higher unloading factors corresponding to shorter times for full stop of the element motion.

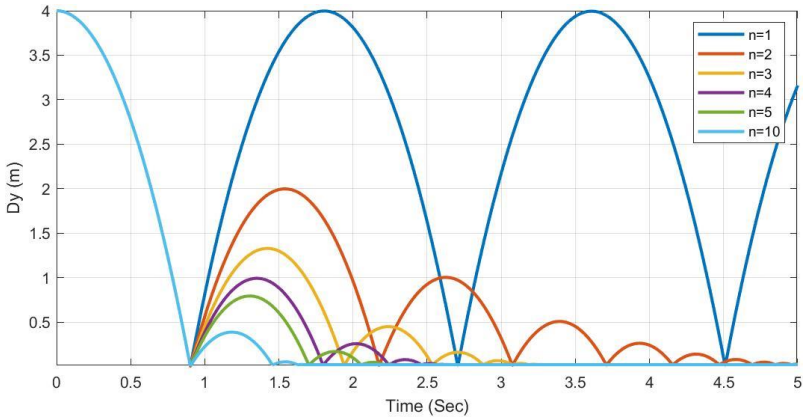


Figure 3.16. Displacement-Time History of a Falling Element Under its Own Weight with Different Unloading Stiffness Factors

The velocity-time history of the element for different unloading stiffness factors is shown in Figure 3.17. In a similar manner, for n=1, the velocity of the element after the rebound was unchanged, implying that all the kinetic energy was recovered after separation of the element from the ground. The reduction in the rebound velocity after each impact is clearly observed for higher n values.

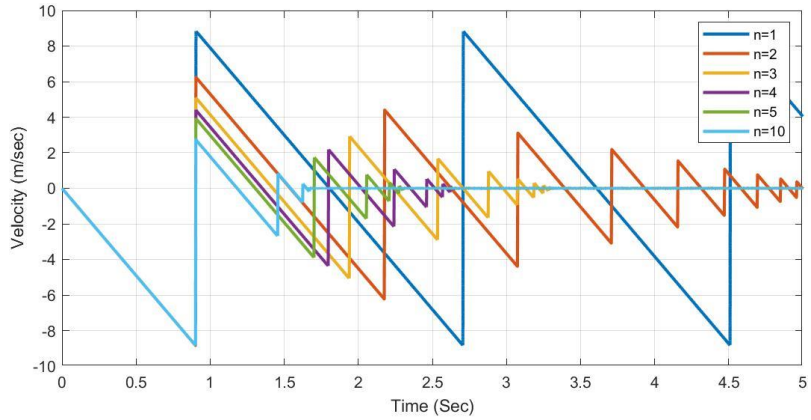


Figure 3.17. Velocity -Time History of a Falling Element Under its Own Weight with Different Unloading Stiffness Factors

The theoretical relation between the rebound factor and the unloading stiffness factor was given in Eq. 3.15. The rebound factor r calculated from the ratio of the element's velocity just before and after the rebound event is compared to this relation in Figure 3.18. The modification of the contact stiffness during impact yields results matching with the theoretical estimate of the coefficient of restitution with the spring consuming the appropriate energy during the impact.

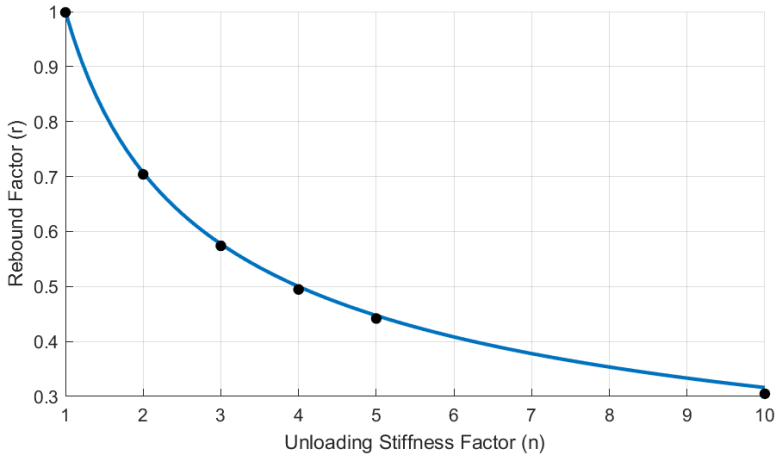


Figure 3.18. Relation between Calculated and Theoretical Rebound Factor "r" for Different Unloading Stiffness Factors

In the second group of analyses, the falling object of Figure 3.15 was modeled with 16 applied elements connected via distributed shear and normal springs. The number of springs along the boundary of the elements was selected as 4, while the model properties were kept identical. A time step of dt=0.001 was used while the block was in the air: during impact, time steps of dt=0.00001 and dt=0.00005 were used for loading coefficients of n=1 and 2, respectively.

The response of the object in terms of the potential, kinetic, and the total energies can be seen in Figure 3.19 for n=1. As expected, for this case, total energy remained constant during the simulation, while potential energy and kinetic energy changed during the motion. For the case of unloading stiffness factor n=2, a similar plot of energy components is presented in Figure 3.20. In each stage of the bounce, the total energy of the block was constant until the next impact. The kinetic and potential energy distributions changed accordingly, consistent with the theory. As the loss of energy led to more stable equations, the time step used could also be increased during this simulation.

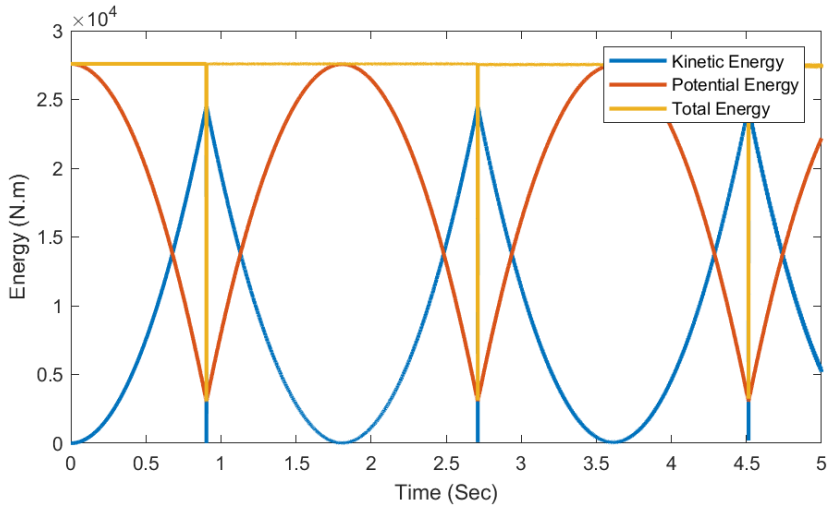


Figure 3.19. Variation of Potential, Kinetic, and Total Energy for a Falling Block Composed of 16 Elements with Unloading Stiffness Factor of n=1

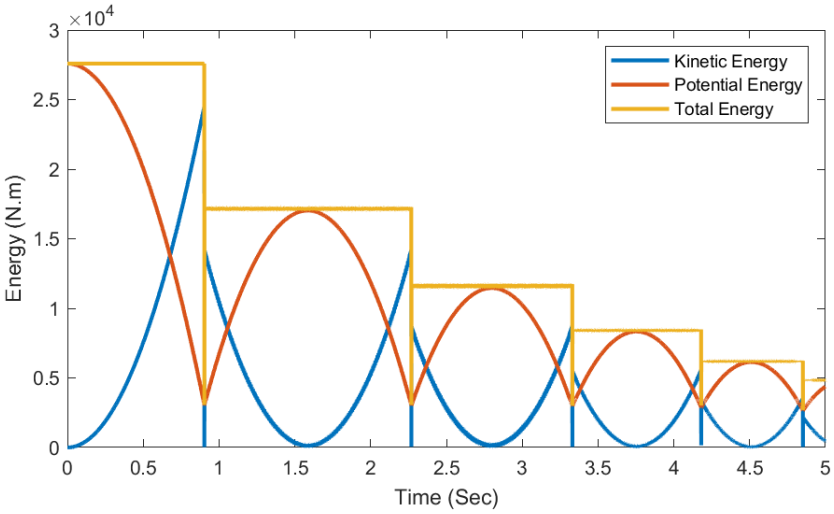
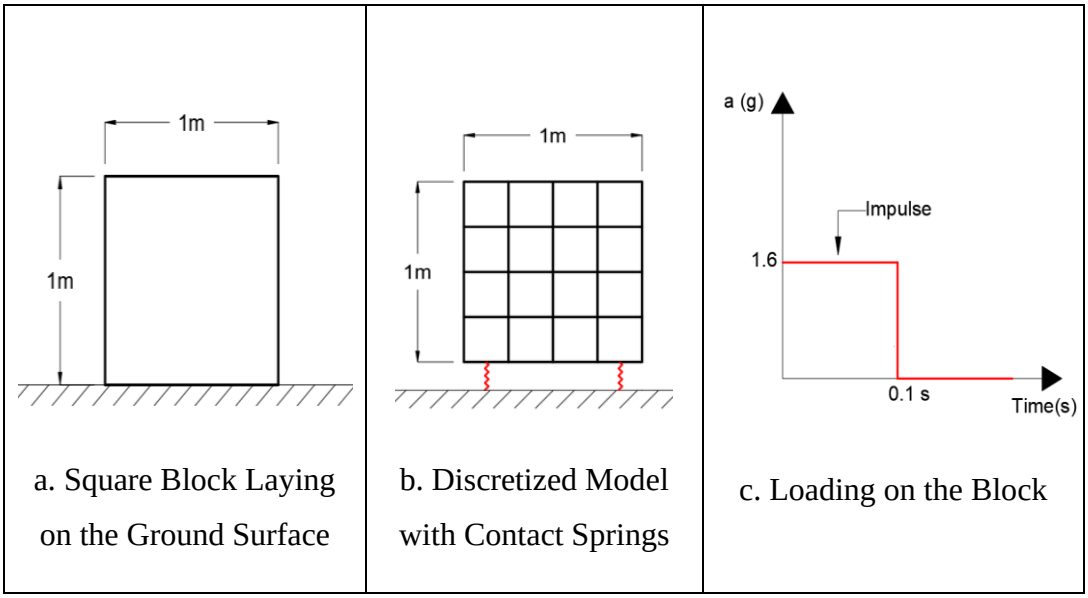


Figure 3.20. Variation of Potential, Kinetic, and Total Energy for a Falling Block Composed of 16 Elements with Unloading Stiffness Factor of n=2

3.4.2 Rocking of a Square Block

In this section, the simulation of the rocking response of a square block laying on the ground surface (Figure 3.21.a) is presented. The block, discretized using 1m thick 4x4 grid of applied elements with 0.25m element size (Figure 3.21.b), was subjected

to a horizontal impulse of 1.6g (Figure 3.21.c) for a duration of 0.1 seconds and was left to vibrate freely afterwards. The Poisson ratio for the model was assumed as υ=0 while the density, modulus of eleasticity and the coefficient of restitution were taken as 2650 kg/m^3, 20 GPa, and r=0.75, respectively. The results of the simulation were compared to the simulations conducted by Pekau and Zhu (2006), who used a 3 degree of freedom rigid body model to solve the equations of motion for the rocking block.



a. Square Block Laying on the Ground Surface

b. Discretized Model with Contact Springs

c. Loading on the Block

Figure 3.21. Square Block Dimensions and its Discretized Version with the Applied Impulse

After the discretization of the block into 16 square elements (Figure 3.21.b), rocking motion and corresponding contact to the ground was enabled by the formation/removal of contact springs at the base of the block on the corner elements. In accordance with the simulations in Pekau and Zhu (2006), the unloading stiffness factor was set as n =1.78, corresponding to a coefficient of restitution of 0.75. A time step of dt=0.00006 when in the air and dt=0.00002 seconds during the contact was used throughout the analysis. The result of the analysis is compared to the result obtained by Pekau in Figure 3.22. The block gains an initial velocity through the impulse, which leads to a rotation of approximately 0.035 radians before it stops and turns back. After the maximum rotation is obtained as such, at every instant, the

block loses some kinetic energy, finally coming to a standstill. The result obtained using the applied element model is generally in a good agreement with the Pekau model, as shown in Figure 3.22.

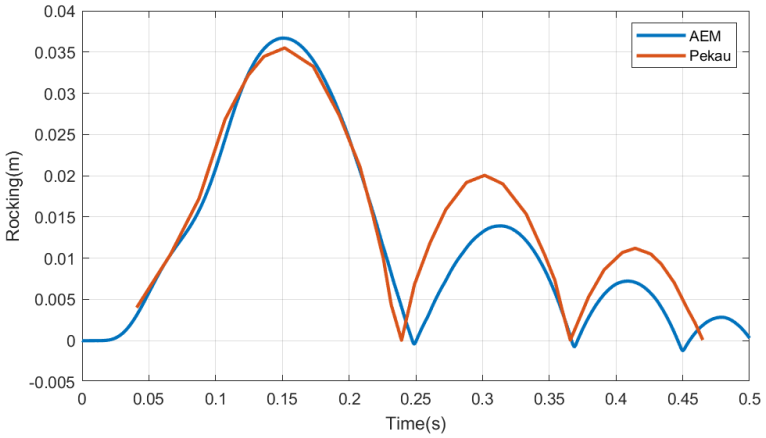
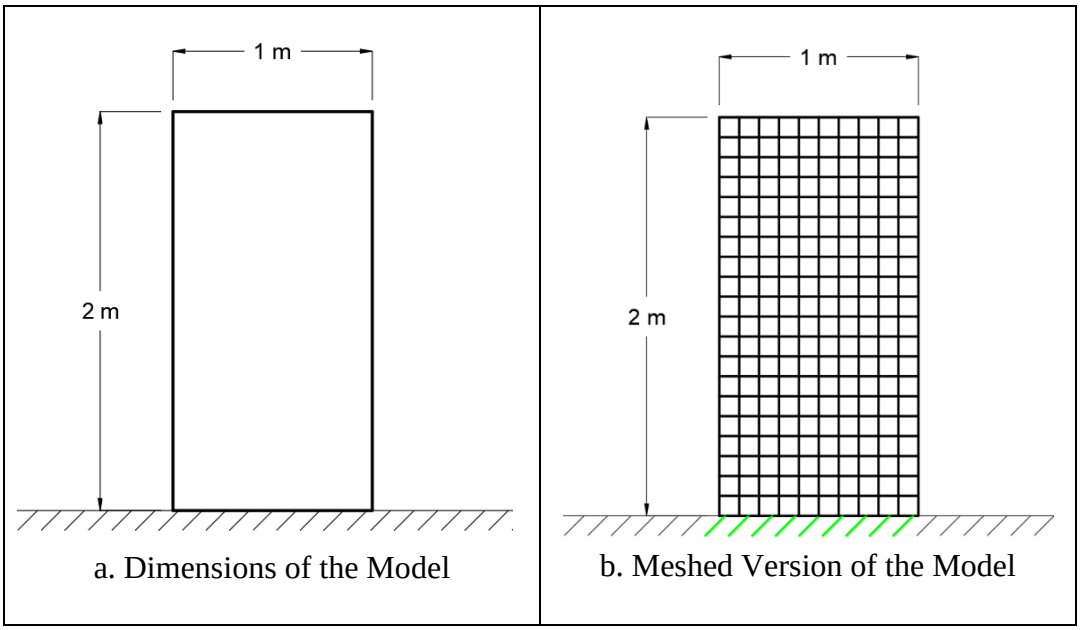


Figure 3.22. Rocking of the Block Due to Sudden Impulse

3.4.3 Loss of Stability for a Severed Monolith

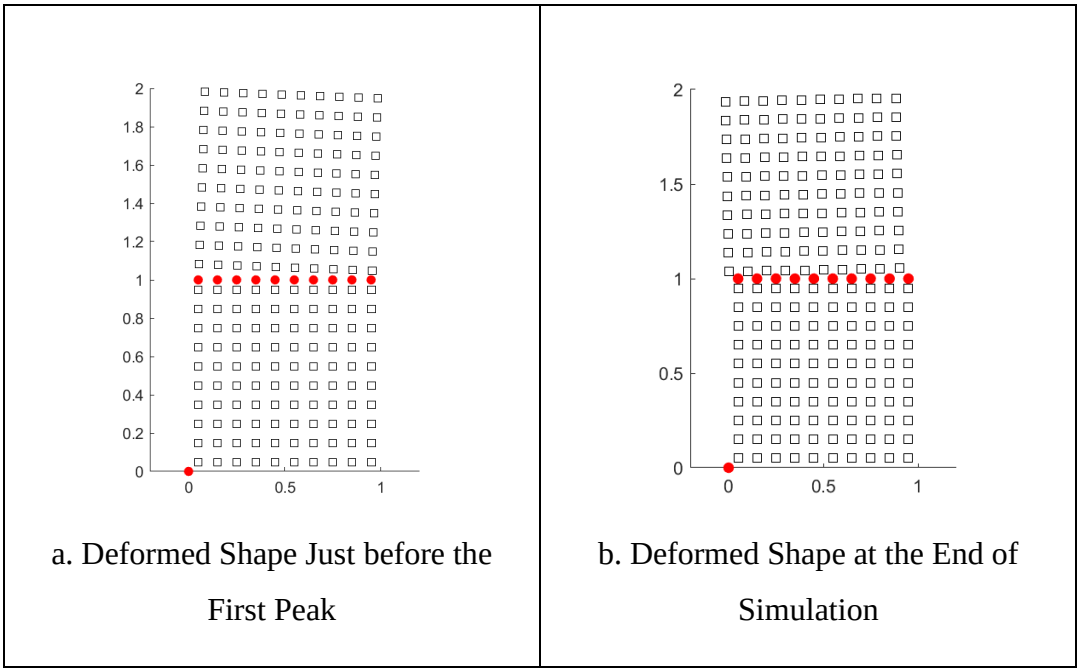
The third problem solved in this section is a rectangular monolith, which is severed in the middle with the top block behaving as a rigid body after this stage. The 2m(H)x1m(W) rectangular block (Figure 3.23.a) with a thickness of 1m, density of 2650 kg/m^3, Poisson ratio of υ=0 and a modulus of elasticity of E=20 GPa is subjected to a horizontal ground motion impulse of 1.6g (Figure 3.21.c) as before. A time step of dt=0.00005 was used throughout the analysis. The model was discretized with square elements (0.1 m x 0.1 m). The base of the model was fixed, as shown in Figure 3.23. A weak section, by defining a negligible tensile strain limit, was introduced at mid-height of the model (h=1m), enabling the model to crack horizontally and split into separate blocks at this height. The upper block obtained henceforth was similar to the block shown in the previous section; the rocking takes place on top of the flexible bottom in this section after the cracking. A snapshot of the deformed shape before the first peak at t=0.136 seconds and at the end of the simulation is shown in Figure 3.24.a, and Figure 3.24.b, respectively.



a. Dimensions of the Model

b. Meshed Version of the Model

Figure 3.23. High Block Subjected to a Sudden Impulse



a. Deformed Shape Just before the First Peak

b. Deformed Shape at the End of Simulation

Figure 3.24. Separation and Recontact of the Cracked Model

After the application of the horizontal ground motion impulse, the lower block vibrates as it is fixed at the base while the upper block is rocking. The lateral motion of the top of the lower block and the vertical movement of the upper block measured at the left and right corners are shown in Figure 3.25 and Figure 3.26, respectively.

In addition to the rocking, the upper block appears to have gone through some horizontal movement of about 48.7 mm at the end of the simulation as a consequence of the drifting behavior. As shown in Figure 3.26, during rocking the body loses contact with the bottom block completely leading to rigid body motion in the air without restraints, i.e. the phenomenon called drifting (Pekau and Zhu, 2006). Any residual velocity in the horizontal direction leads to permanent deformations during the drifting motion. The increase in drift times usually leads to more permanent deformations. The lateral deformations during contact periods are very small; increasing the shear stiffness of the contact springs therefore does not change this behavior. In fact, increasing the shear springs spring constants contradicts the proposed method (Tagel-Din and Meguro, 1999) changing the nature of the contact. High shear stiffnesses at contact led to erratic results and usually increased drift times.

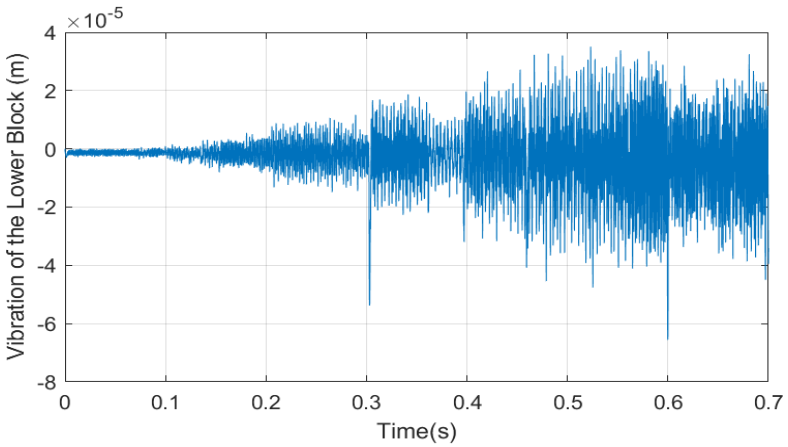


Figure 3.25. Vibration of Top of the Lower Block Due to the Sudden Impulse

A particular advantage of the applied element models is the ability to simulate very large rigid body displacements using classical implicit Newmark integration. In order to show that the model can simulate such behavior, the same model is subjected to 2.5x the original impulse at 4g. A time step of dt=0.00005 was used throughout the analysis. The results of the corresponding analysis shown in Figure 3.28 is obtained. The top block separated from the bottom block first in a rocking motion, later moving towards the right with the attained velocity.

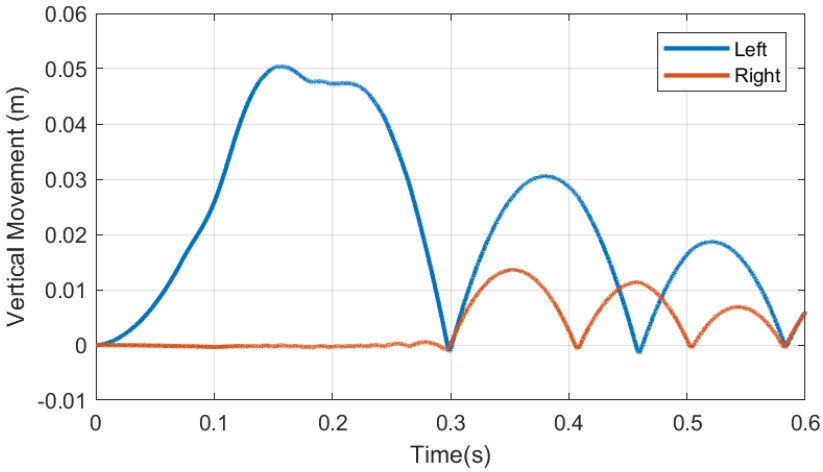


Figure 3.26. Vertical Movement of Left and Right Corners of the Upper Block

Unlike the previous case, however, this time the attained velocity is so high that the rotation cannot be reversed due to gravity. The block keeps rotating in clockwise direction and completely detaches itself from the bottom block flying towards the right-hand side. During this motion, it also hits the top right corner of the bottom block at approximately t=0.17 sec and t=0.4 sec and bounces upon the contacts. The free vibration of the bottom block during the simulation and the projectile motion of the top block after complete severing from the bottom can be simulated, as shown in Figure 3.27 and Figure 3.28, respectively.

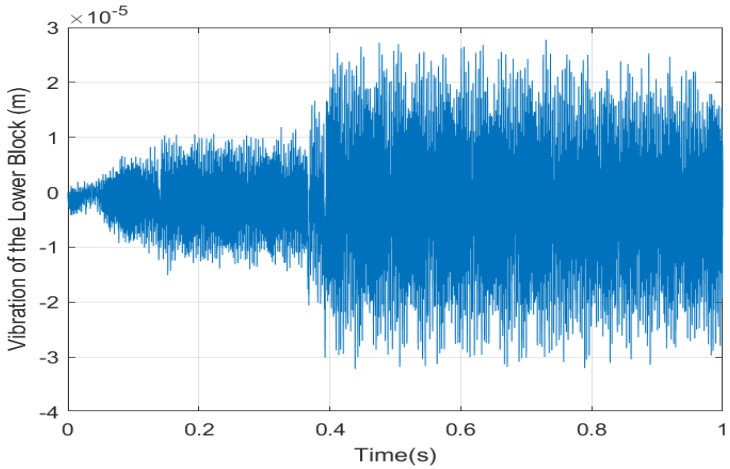
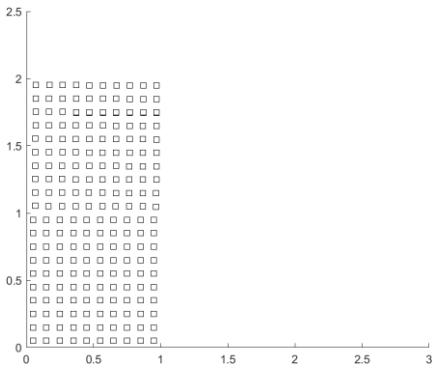
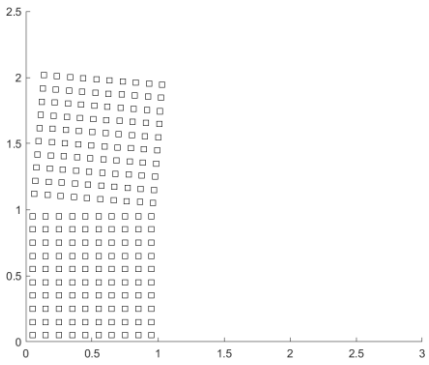


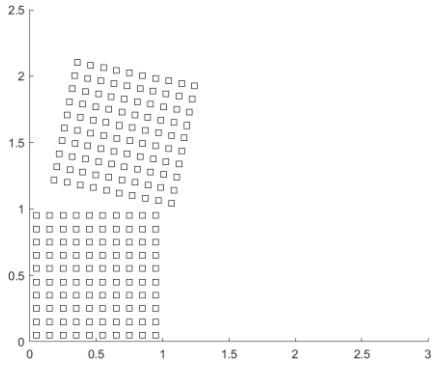
Figure 3.27. Time History for Lateral Motion of the Bottom Block at the Middle



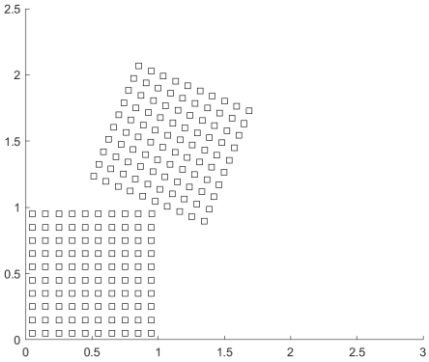
a. T=0.0365 sec



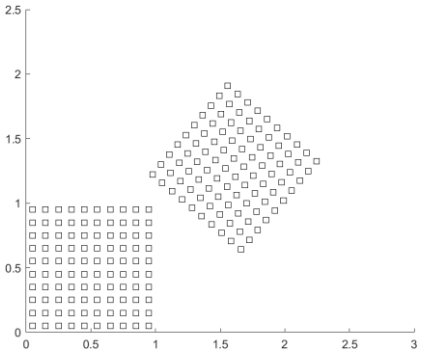
b. T=0.0841 sec



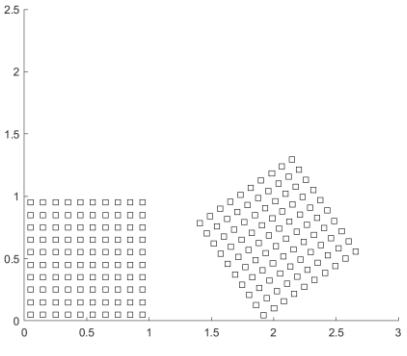
c. T=0.17 sec



d. T= 0.4 sec



e. T= 0.64 sec



f. T= 0.82 sec

Figure 3.28. Motion of the Separated Block Subjected to an Impulse of 4g

CHAPTER 4

SUMMARY AND CONCLUSIONS

4.1 Summary

A set of tools for predicting the large displacement behavior of concrete gravity dams was developed in this study carried under grant no 116M524 with TUBITAK support. The study is presented in two parts; first, the development of a graphical user interface for the creation of the discrete element models was presented. The software interface, which is web-based, utilizes HTML, JavaScript, and CSS languages to give the user a simple utility with which a complex geometry can be drawn and discretized into an applied element-finite element hybrid mesh. Complex shapes can be formed and meshed in the interface, including any type of opening. Meshing conducted using ray-tracing allows these shapes, with or without openings, to be discretized easily into mathematical models. Examples with different geometries were shown to demonstrate the capabilities of the developed software. Using the GUI, different types of boundary conditions can be added to the various parts of the models. In a similar fashion, the interface is able to apply both static and dynamic loads to any part of the structures. These loads include point loads, distributed uniform, triangular and trapezoidal loads, and uplift pressures at the base of the model with a reservoir. The data output of boundary conditions and loads are in text format and can be easily used for any solver engine. For the post-processing of the results, a portable command-line driven graphing utility (Gnuplot) was used as a tool. This embedded graphing utility can be used for presenting the analysis results to the user after corresponding output files are created with the help of the solver engine. Creating and plotting of modes shapes, plotting of stress and strain contours of the analyzed models are the type of the post process applications embedded inside the user interface.

In the second part of the study, a set of algorithms were implemented and developed for enabling the simulation of large displacement behavior for the 2D models created using the interface. Matlab was used to create a solver module for the back-end part of the software. Several functions for discretizing the GUI output to a mathematical model were developed. These functions included routines for obtaining the stiffness and mass matrices, internal forces, and contact control. In the case of collision and recontact process, functions for modeling of the contact springs and updating elements connectivity after separation of the elements were also developed. For post-processing, a new algorithm for tracking the individual crack paths within a 2D closed geometry was developed. The outputs generated via the interface were tested by using them in the solver engine for the solution of the static and dynamic problems. The solver engine was tested with several models, including the bouncing of free-falling objects composed of both single and multiple elements and for rocking and sliding of the bodies subjected to ground motions. The module was able to simulate the separation of blocks and the resulting rocking and sliding of the separated bodies consistent with the theory and experiments.

4.2 Conclusions

A simple and easy to use web-based applied element tool which can be used for analyzing of dam monoliths was developed first in this study. The following conclusions were derived from the first part of the study:

1. Applied element allows the use of simple and fast algorithms to create mesh structures in contrast to regular finite elements with often used complex meshing algorithms.
2. HTML, Javascript and Canvas provide tools for browser based generation of structural models, very fast upto 10000 elements without using complex data structures.

Next, a Matlab based solver was prepared for the solution of large displacement and contact problems involving monoliths. Abilities and functionalities of the GUI, as well as the solver engine, were tested with simple and complex benchmark cases. During this work, the following conclusions were obtained:

1. The normal and shear stiffness calculations suggested by Tagel-Din and Meguro (1999) for the contact springs work properly. Reduction of the shear contact stiffness, especially when the separated elements are moving horizontally due to the lateral applied loads and make contact in shear mode, leads to the sliding of elements in the contact interface while increasing the shear stiffness, force the collided elements to stick upon contact. A decrease in stiffness of normal contact springs results in the stop of the motion after the collision, while extra energy is added to the system if the normal spring stiffness is increased. This effect can be seen easily in the case of falling objects colliding with the ground.
2. The selection of the time step before and during the contact significantly affects the collision and recontact behavior. The selection of a very small time increment in the order of 10-4 and 10-5 allows following material behavior properly during the contact, and thus energy is perfectly transmitted upon contact. The algorithm becomes very sensitive to the contact time step when elements have velocity in x and y directions simultaneously.
3. The main advantage of the applied element method compared to the finite elements or EDEM is that the time increment can be enlarged before the collision starts. Using AEM, implicit integration, and larger time increments in the order of 10-1 and 10-2 can be used before the collision while small time increment is needed only during the collision process.

4.3 Limitations and Future Work

There are some limitations to the capabilities of the front-end (GUI) and back-end (Solver Engine) part of the software.

- The current version of the GUI is only capable of creating 2D geometries and continuum models.
- The orientation of the applied elements is fixed in the developed framework; i.e. elements are always horizontal. Inclined elements or elements with different shapes were not considered.
- The discretization capability of the GUI is limited to around 100000 elements. If the total number of discretized elements exceed this limit,the browser become unresponsive due to memory issues, which leads to the failure of the discretization process.

Future work for the improvement of the GUI can include;

- The addition of features that enable users to create frame elements like columns and beams from stick geometry,
- Improvement of the post-processing for direct output of important cracks and their properties,
- Inclusion of reinforcement or steel shell elements for enabling the simulation of reinforced concrete structures.

The solver engine used in this study can be used as a building block tool in later research projects in the future. As it is developed in a modular form using input from the GUI, additional features that advance the capabilities, in terms of new material models and solvers, can be easily implemented.

REFERENCES

A Bhattacharjee SS, Leger P. 1994. Application of NLFM models to predict cracking in concrete gravity dams. Journal of Structural Engineering 120(4): 1255-1271.

Calayir Y, Karaton M. 2005. Seismic fracture analysis of concrete gravity dams including dam-reservoir interaction. Computers and Structures 83: 1595-1606.

CSS.2020. Retrieved from https://developer.mozilla.org/en-US/docs/Learn/CSS/Introdution_to_CS

Cundall PA. 1971. A computer model for simulating progressive large scale movements in blocky rocky systems. International Symposium on Rock Fracture, October 4-6, Nancy, France.

Elkholy, S., Meguro, K. 2004. Numerical Simulation of High-Rise Steel Buildings Using Improved Applied Element Method, 13th World Conference on Earthquake Engineering, Vancouver, B.C., Canada.

Elkholy, S., Meguro K. 2005. Simulation of Seismic Damage to Steel Buildings, Bulletin of Earthquake Resistant Structure, 38, 145-153.

Elkholy, S., Tagel-Din, H., Meguro, K. 2003. Structural Failure Simulation due to Fire by Applied Element Method, The 5th Japan Conference on Structural Safety and Reliability, JCOSSAR 2003.

Elkholy, S.A., Gomaa, M.S., Akl, A.Y. 2012. Improved Applied Element Simulation of RC and Composite Structures Under Extreme Loading Conditions, Arabian Journal for Science and Engineering, 37,4, 921-933.

FERC (Federal Energy Regulatory Commission), 1991. Engineering guidelines for evaluation of hydropower projects - Chapter III Gravity Dams. Federal Energy Regulatory Commission, Office of Hydropower Licensing, Report No. FERC 0119-2,Washington D.C.,USA.

FERC,1999. Engineering Guidelines for the Evaluation of Hydroelectric Projects. Chapter 3. Gravity dams.

HTML5 Tutorial.2020. Retrieved from https://www.w3schools.com/html/

Karbassi, A., Nollet, M.J. 2008. Application of the Applied Element Method to the Seismic Vulnerability Evaluation of Existing Buildings, CSCE 2008 Annual Conference, Quebec, QC.

Lau, D.T., Wibowo, H. 2010. Seismic Progressive Collapse Analysis of Reinforced Concrete Bridges by Applied Element Method, Earth and Space 2010: Engineering, Science, Construction and Operations in Challenging Environments ASCE.

Meguro, K., and Tagel-Din, H. 1997. A New Efficient Technique for Fracture Analysis of Structures. Bulletin of Earthquake Resistant Structure Research Center, Institute of Industrial Science, The University of Tokyo, No. 30, 103-116 p.

Meguro, K., and Tagel-Din, H. 2000. Applied element method for structural analysis: Theory and application for linear materials. Structural Eng. /Earthquake Eng., Vol. 17. No. 1, pp. 21s-35s, Japan Society of Civil Engineers

Meguro, K., and Tagel-Din, H. 2001. Applied Element Simulation of RC Structures under Cyclic Loading. Journal of Structural Engineering ASCE, 127 (11), pp. 1295-1305.

Meguro, K., and Tagel-Din, H. 2002. Applied Element Method Used for Large Displacement Structural Analysis. Journal of Natural Disaster Science, 24 (1), pp. 25-34.

Michael Galetzka, Patrick Glauner (2017). A Simple and Correct Even-Odd Algorithm for the Point-in-Polygon Problem for Complex Polygons. Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017), Volume 1: GRAPP.

Pandey, B.H., Meguro, K. 2004. Simulation of Brick Masonry Wall Behavior Under In-Plane Lateral Loading Using Applied Element Method, Paper No: 1664, 13th World Conference on Earthquake Engineering, Vancouver, B.C., Canada.

Ramancharla, P.K., Meguro, K. 2006. "3D Numerical Modelling of Faults for Study of Ground Surface Deformation Using Applied Element Method", Current Science, 91, 8, 1026-1037.

Raparla, H.B., Ramancharla, P.K. 2012. Nonlinear Large Deformation Analysis of RC Bare Frames Subjected to Lateral Loads. Indian Concrete Institute. Report No: IIIT/TR/2012/-1.

Sasani, M. 2008. Response of a Reinforced Concrete Infilled-Frame Structure to Removal of Two Adjacent Columns, Engineering Structures, 30, 9, 2478-2491.

Tagel-Din, H., and Meguro, K., 1999. Applied Element Simulation for Collapse Analysis of Structures. Bulletin of Earthquake Resistant Structure Research Center, Institute of Industrial Science, The University of Tokyo, No. 32, 113-123 p.

Tagel-Din, H., Rahman, N.A. 2006. Simulation of the Alfred P. Murrah Federal Building Collapse due to Blast Loads, Building Integration Solutions: Proceedings of the Architectural Engineering Conference (AEI), Omaha, Nebraska, USA.

JavaScript .2020. Retrieved from https://www.w3schools.com/js

APPENDICES

A. Post Processing of Mode Shapes Plots

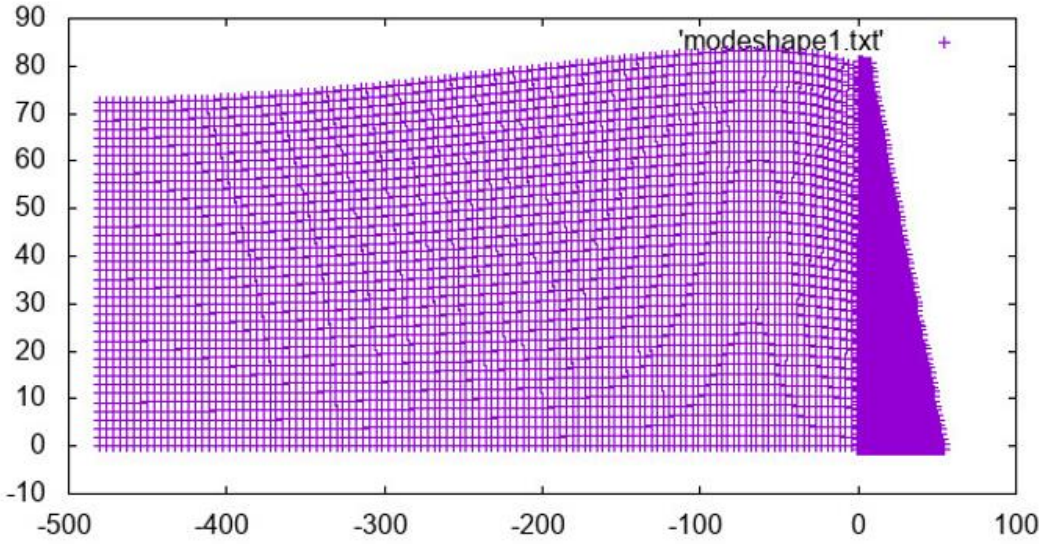


Figure A.1. 1st Mode Shape of the Sample Model

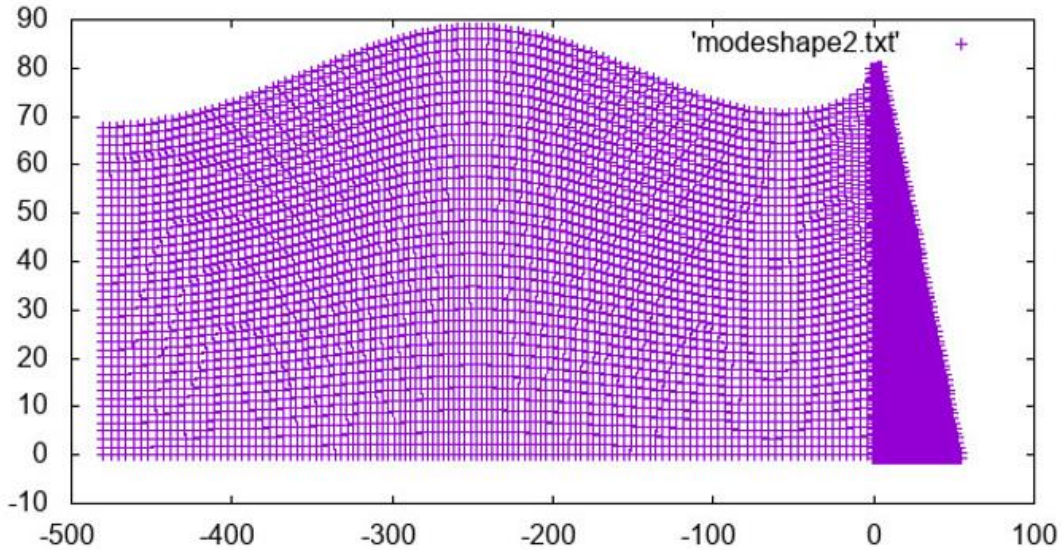


Figure A.2. 2nd Mode Shape of the Sample Model

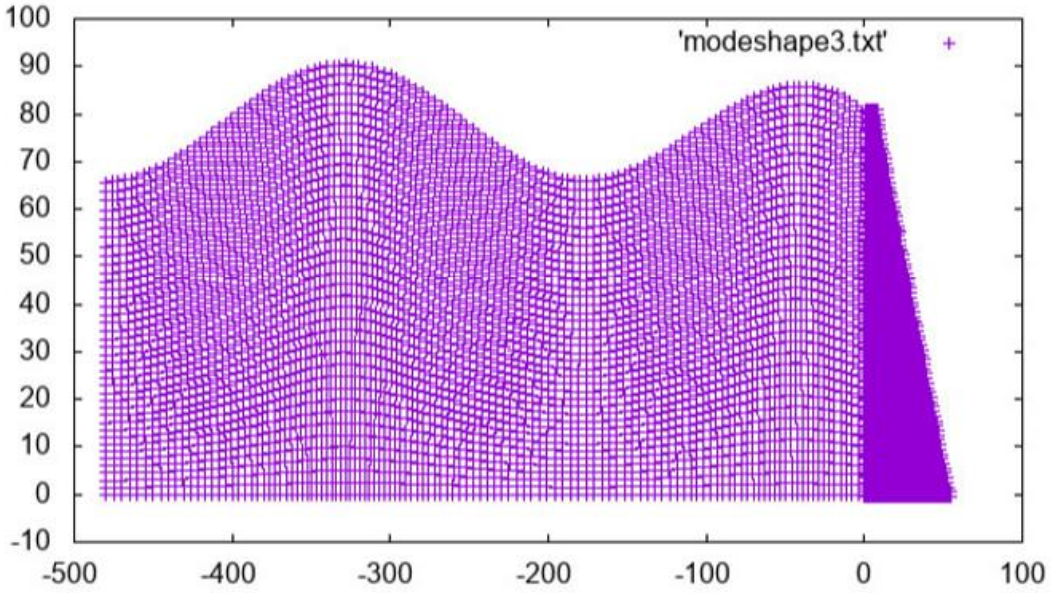


Figure A.3.3rd Mode Shape of the Sample Model

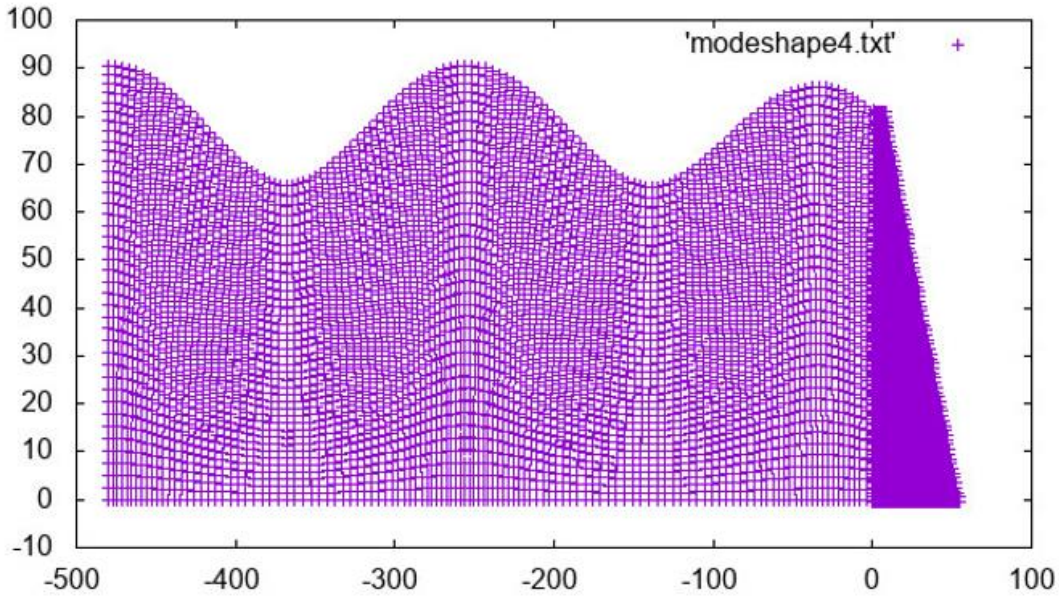


Figure A.4. 4th Mode Shape of the Sample Model

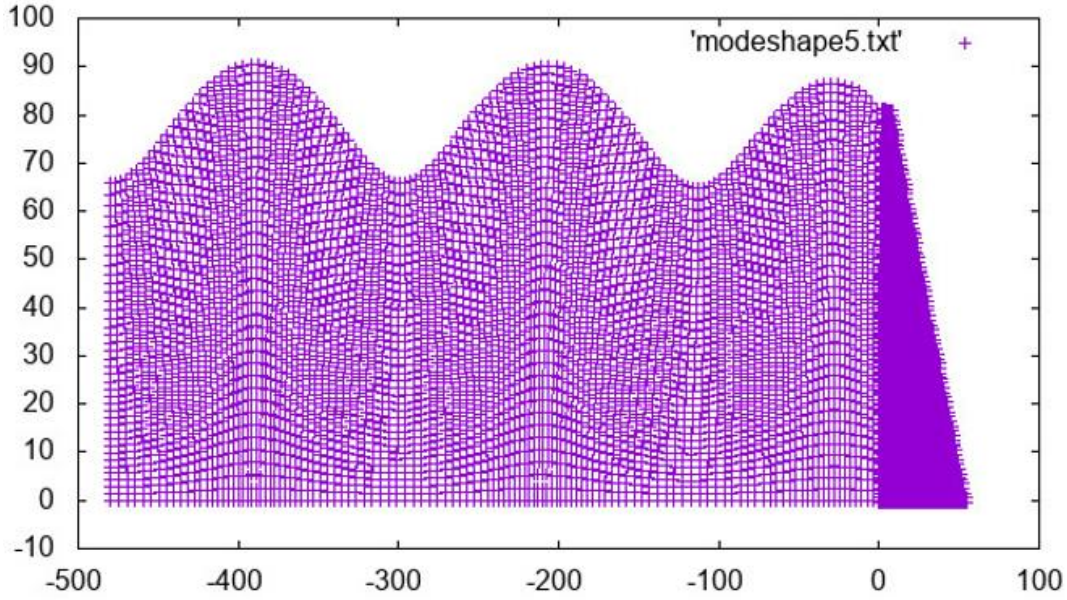


Figure A.5. 5th Mode Shape of the Sample Model

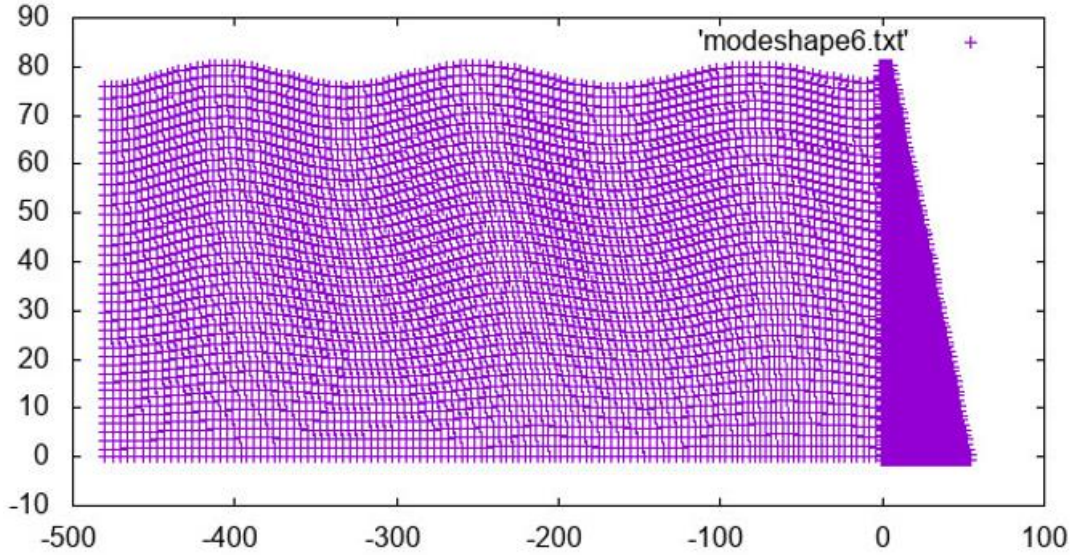


Figure A.6. 6th Mode Shape of the Sample Model

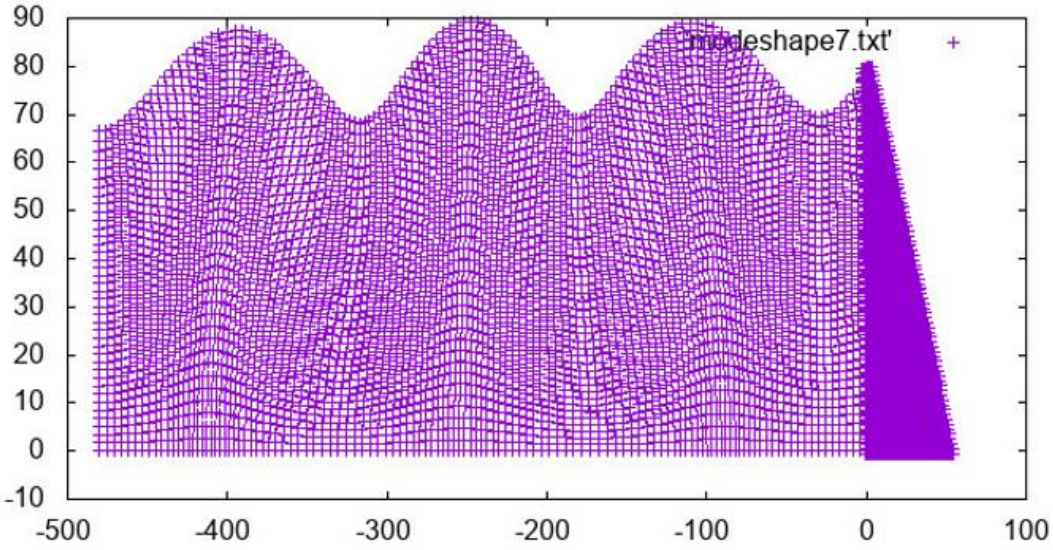


Figure A.7. 7th Mode Shape of the Sample Model

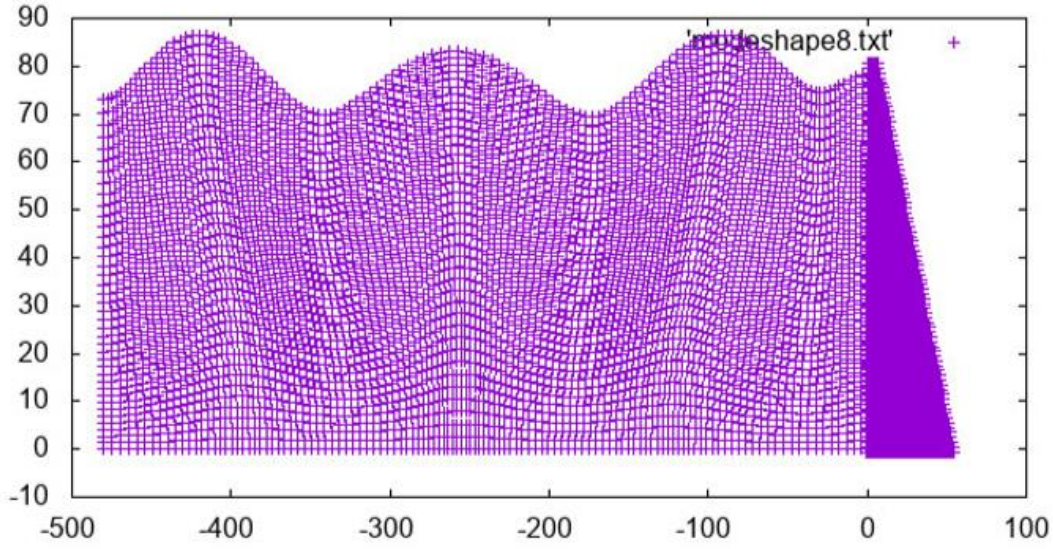


Figure A.8. 8th Mode Shape of the Sample Model

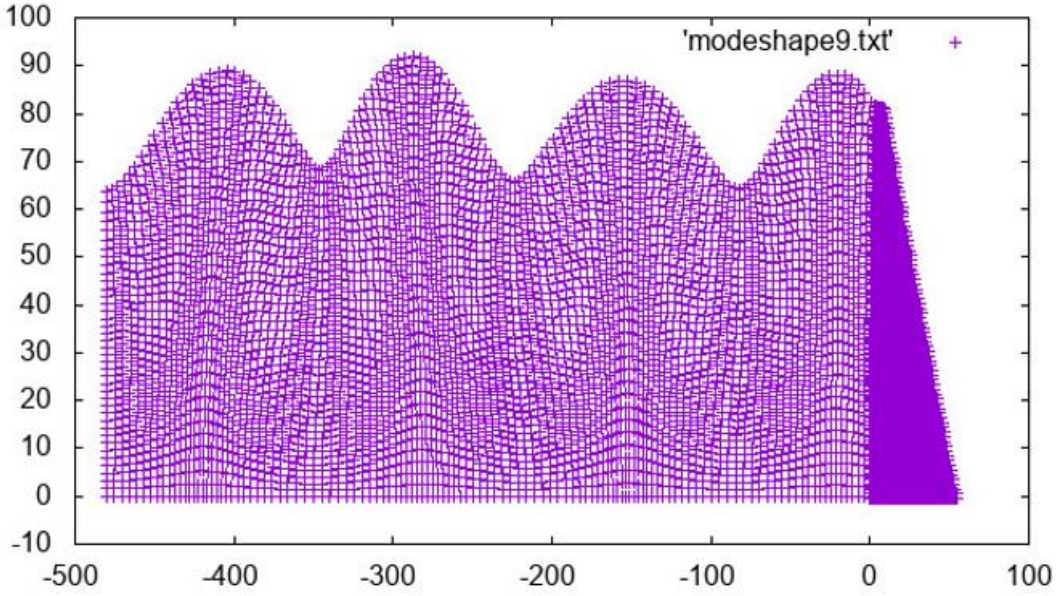


Figure A.9. 9th Mode Shape of the Sample Model

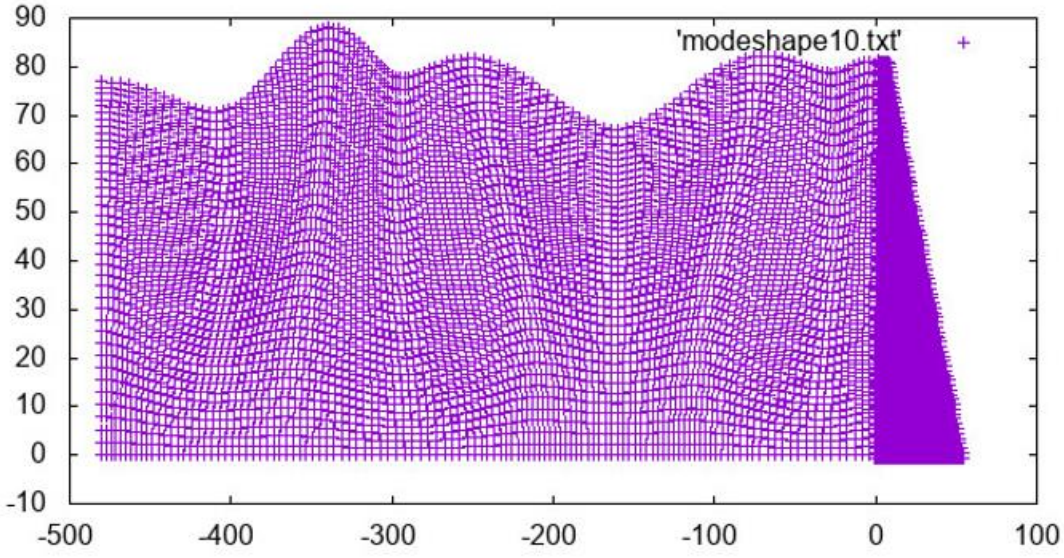


Figure A.10. 10th Mode Shape of the Sample Model