Development of an Internet-Based Analysis Tool Using
The Boundary Element Method

A thesis submitted to the
Division of Graduate Studies and Research
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

In the Department of Mechanical, Industrial and Nuclear Engineering
of the College of Engineering

August 2000

By

Sachin Chaudhari

B. E., Govt. College of Engineering, Pune, India, 1998

Committee Chair: Dr. Yijun Liu
ABSTRACT

This thesis investigates the usefulness of Boundary Element Method (BEM) and a software tool called UCWaves developed at University of Cincinnati. The Boundary Element Method for structural analysis has emerged as a very promising alternative for the Finite Element Method (FEM). Although BEM is more complicated than FEM, it is faster and it does reduce a lot of storage space over FEM for solving large models. The advantage of BEM becomes apparent for the infinite domain as well as fracture mechanics problems. There are very few CAD tools, which use BEM for analysis. This thesis is a successful attempt to design and develop an Internet based analysis tool using the BEM. The existing standalone BEM software has also been enhanced. This study will lead to more research activities on the Internet-based software development for the BEM.
I would like to express my gratitude to everyone who contributed, in different ways, to my completion of this work. My special thanks go to Dr. Yijun Liu for his advice, support and patience throughout my study at the University of Cincinnati. The experience will certainly become part of the most pleasant memories and most important inspirations of my life journey. I would also like to thank Dr. Edward Berger and Dr. Philip Quo for valuable suggestions and serving on my thesis committee.

I feel thankful to all the past and present members of the UCWaves Project Team. My thanks to Nan Xu and Kumar Puppala for their valuable suggestions and help. Also I would like to thank Kaushik Mitra who helped me to create test cases. Last, but not least, my gratitude goes to my parents, for their love, understanding and encouragement, without which it would have been impossible for me to accomplish this work.
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>1.1 The Boundary Element Method</td>
<td>8</td>
</tr>
<tr>
<td>1.2</td>
<td>1.2 The Advantages of BEM and the Internet Computing</td>
<td>8</td>
</tr>
<tr>
<td>1.3</td>
<td>1.3 Advantages of Java Language</td>
<td>11</td>
</tr>
<tr>
<td>1.4</td>
<td>1.4 UCWaves Software</td>
<td>13</td>
</tr>
<tr>
<td>1.5</td>
<td>1.5 Internet and CAE</td>
<td>13</td>
</tr>
<tr>
<td>1.6</td>
<td>1.6 Structure of This Thesis</td>
<td>14</td>
</tr>
<tr>
<td>2.1</td>
<td>2.1 Introduction</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>2.2 A Case Study-Crack Analysis</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>2.3 Discussions</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>3.1 Introduction</td>
<td>26</td>
</tr>
<tr>
<td>3.2</td>
<td>3.2 Advantages of Web-Based Tool/Application</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>3.3 Author’s Contribution</td>
<td>29</td>
</tr>
</tbody>
</table>
3.4 DESIGN AND ARCHITECTURE ................................................................................... 29
3.5 IMPLEMENTATION OF THE MAIN WEB PAGE .................................................... 30
3.6 IMPLEMENTATION OF APPLET .............................................................................. 32
3.7 UC WAVES JAVA SERVER (UJS) ........................................................................... 34
3.8 INSTALLING AND CONFIGURING WEB SERVER .................................................. 35
3.9 FUTURE WORK ...................................................................................................... 35
3.10 CONCLUSION ....................................................................................................... 37

CHAPTER 4. UC WAVES STANDALONE APPLICATION ........................................ 38

4.1 INTRODUCTION .................................................................................................... 38
4.2 BASIC ARCHITECTURE OF GUI .......................................................................... 38
4.3 SOLVER ............................................................................................................... 40
4.4 ENHANCEMENTS ADDED TO UC WAVES ......................................................... 40
4.5 FUTURE SCOPE FOR DEVELOPMENT ............................................................. 47
4.6 CONCLUSION ...................................................................................................... 49

APPENDIX A ............................................................................................................ 50

THE INPUT AND OUTPUT FILES ........................................................................... 50

APPENDIX B ............................................................................................................ 58

LISTING OF CODE .................................................................................................... 58

REFERENCES .......................................................................................................... 76
List of Figures

FIG.1.1 FEM (LEFT FIG) AND BEM MESH ........................................................................................................ 9
FIG.2.1 MODEL OF INTERFACE CRACK ........................................................................................................ 18
FIG.2.2 SINGULAR ELEMENTS NEAR CRACK TIP (FEM MODEL) ................................................................ 19
FIG 2.3 CRACK OPENING IN SINGLE MATERIAL ........................................................................................... 21
FIG.3.1 PRESENT 2-TIER ARCHITECTURE .................................................................................................... 31
FIG.3.2 IMPROVED 3-TIER ARCHITECTURE .................................................................................................. 36
FIG 4.1 MESH CLASS AND NOTATIONS ........................................................................................................ 39
FIG 4.2 MODEL BEFORE SOLVING .................................................................................................................. 45
FIG 4.3 AFTER SOLVING-DEFORMED SHAPE .............................................................................................. 46
Chapter 1. Introduction

1.1 The Boundary Element Method

The \textit{Boundary Element Method} (BEM) is a numerical method to solve the boundary value problem by the boundary integral equation (BIE) formulation. Boundary Element Method has emerged as a powerful alternative to finite elements particularly in cases where better accuracy is required due to the stress concentration or where the domain extends to infinity.

The name \textit{Boundary Element Method} appeared in the late 70’s in an attempt to make analogy with the Finite Element Method (FEM). Today, the BIE/BEM has gained a great deal of application in many fields of computational mechanics, such as the wave propagation, heat transfer, diffusion and convection, fluid flow, fracture mechanics, electrical problems, geomechanics, plates and shells, inelastic problems, contact problems, design sensitivity and optimization and inverse problems [1].

1.2 The Advantages of BEM and the Internet Computing

The most important feature of BEM is that it only requires discretization of the surface rather than the volume in 3-D problems as shown in Fig.1.1. Therefore mesh generation is easier and faster. This results in lesser CPU usage. The CPU usage is important because the speed of computations is becoming very crucial factor due to the growth and potential of web-based applications.
BEM is a better tool for CAD mainly because it is easy to generate data required to run a problem and carry out modifications needed to achieve an optimum design. For example, while designing a turbine blade section, analysis is much easier with BEM. Turbine blades have a series of cooling ducts whose size, position and number have to be modified during the design process. Such a modification creates difficulties for finite elements as some elements may easily become distorted or have bad aspect ratios. The boundary element mesh instead is easy to modify.

Another important advantage of BEM over FEM is when analyzing problems with stress concentrations. Studies have proved that BEM is more accurate for fracture mechanics problems. A case study has been presented in Chapter 2 to elaborate this point.

Some typical applications like sound-structure interaction, seepage, electrostatics, diffusion, etc., can be solved by BEM where other methods do not give practical results. Corrosion engineers have used the method to design better cathodic protection systems for offshore structures, ships and pipelines [2]. Many of these structures are basically
three-dimensional and the region of interest extends to infinity. Consequently they could not be effectively analyzed before the development of boundary elements. Early attempts to use finite differences or finite elements to solve these problems met with little success. For these cases the computer model has to represent the potential field around the structure, representing the shielding effect of the structural geometry and the effect of the different materials involved. Unlike the structural model the cathodic protection model is concerned with the seawater around the structure and the interface between the seawater and the structure. Hence the use of FEM to analyze the problem would require the subdivision of the seawater surrounding the structures which is an infinite domain problem and thus a difficult task. The advantage of the BEM is that only the structure surfaces needs to be meshed, since the BEM can automatically takes care of the boundary conditions at infinity for the seawater.

The Internet is dramatically changing the engineering practices in design and analysis. More and more engineering software applications are moving to the Internet, prompted by the promising advantages, such as easy access for users everywhere with an Internet connection, easy upgrade of the software, easy control of the license and so on. All these advantages will contribute to faster and cost-effective software development. The successful application of the Internet computing depends largely on the speed of data transfer on the network and the independence on operating systems for the online software. The BEM seems to have the inherent advantages over other domain-based numerical methods, because the size of the BEM data files are always much smaller, leading to faster data transfer on the network. This thesis aims to explore the advantages
and the potential of the BEM on the Internet, since the Internet computing will sooner or later become the dominant model in the engineering design and analysis. The Java language, which renders platform-independent applications, is the choice now in the Internet software development and thus was used for this project to develop the graphical-user interface (GUI) for the BEM.

1.3 Advantages of Java Language

During recent years Java language has emerged as a powerful tool for GUI development as well as network programming. Due to following features, java was chosen for GUI and applet development:

1) Simple, Object Oriented, and Familiar

Primary characteristics of the Java programming language include a simple language that can be programmed without extensive programmer training while being attuned to current software practices. The Java programming language is designed to be object oriented from the ground up. To function within increasingly complex, network-based environments, programming systems must adopt object-oriented concepts. Programmers using the Java programming language can access existing libraries of tested objects that provide functionality ranging from basic data types through I/O and network interfaces to graphical user interface toolkits. These libraries can be extended to provide new behavior.

2) Robust and Secure

The Java programming language is designed for creating highly reliable software. It provides extensive compile-time checking, followed by a second level of run-time
checking. There are no explicit programmer-defined pointer data types. Java provides automatic garbage collection. Automatic garbage collection is the mechanism by which the Java Runtime Environment (JRE) de-allocates the memory once the object is no longer needed by the application. In C/C++, programmer has to explicitly write code to free (de-allocate) memory. In Java it is not needed because of automatic garbage collection.

3) **Architecture Neutral and Portable**

Java technology is designed to support applications that will be deployed into heterogeneous network environments. In such environments, applications must be capable of executing on a variety of hardware architectures. Within this variety of hardware platforms, applications must execute atop a variety of operating systems and interoperate with multiple programming language interfaces. Architecture neutrality is just one part of a truly portable system. Java technology takes portability a stage further by being strict in its definition of the basic language. Programs are the same on every platform--there are no data type incompatibilities across hardware and software architectures.

4) **High Performance**

Performance is always a consideration. The Java platform achieves superior performance by adopting a scheme by which the interpreter can run at full speed without needing to check the run-time environment.
5) Multithreading

Interactive graphical applications have high performance because multiple concurrent threads of activity are supported by the multithreading built into the Java programming language and runtime platform.

1.4 UCWaves Software

UCWaves is 2-D analysis software developed at the Department of Mechanical Engineering, University of Cincinnati. The analysis technique is based on Boundary Element Method. The GUI is developed using Java language and solver is developed using Visual C++ on NT 4.0 platform. UCWaves can be used on Internet or as a standalone application. The input to UCWaves comes in the form of a text input file. UCWaves can display this input model with all its constraints. Once the model is opened in GUI, it can be solved and one can plot the deformed shape of the model. UCWaves is also available on Internet at http://twister.mie.uc.edu/UCWaves.

1.5 Internet and CAE

There are various aspects in which the Internet is playing a crucial role in Mechanical Engineering design and analysis field. One of the shortcomings of BEM is its limited user group. Internet will help BEM community to demonstrate advantages of BEM technique to more CAE users.

Some software vendors are putting their software on server and giving users access to it through the Internet. This will increase the use of the specialized tools/applications
because for a particular analysis companies need not to buy the software; they can buy some time and use it. So companies that do not want to buy specialized software will use the web-based applications. In future instead of using generic CAD/CAE product for all problems, mechanical engineers will use specialized web-based tools wherever necessary because specialized tools are usually more sophisticated and accurate. Being on Internet the version controlling of these products will be easier, so the software vendors can keep on updating their software and they need not to ship it to every client for update. This will increase competitiveness and quality of software products. This will lead to better and faster analysis and design. As the use of specialized web-based tools becomes more common, use of the BEM for analysis will increase.

1.6 Structure of This Thesis

The remaining part of this thesis consists of the following chapters:

Chapter 2 called “Advantages of BEM- A Case Study” presents a case study of fracture mechanics problem. It demonstrates the advantages of the BEM in some specific applications.

Chapter 3 called “UCWaves As a Web-based Application” gives details about the developed architecture of the web-based application. It also discusses the possible future enhancements that can be added to this application.

Chapter 4 called “UCWaves Standalone Application” talks about the basic architecture and this author’s contribution towards standalone application. This chapter gives the
detailed description of the low level classes and how they are used by the application. It also gives important suggestions for future enhancements.

Appendix A includes example of the input and output file. Appendix B includes the code of the main classes. References are provided at the end.
Chapter 2. Advantages of BEM- A Case Study

2.1 Introduction

This chapter aims to demonstrate the advantages of the BEM in the analysis of certain types of problems compared with the FEM. It is also served to show that a good graphical-user interface for the BEM is urgently needed, especially one on the Internet which can increase the applications of the BEM software. To prove the advantages of the BEM, a case study of a fracture mechanics problem is provided. This case study compares the accuracy and number of elements required while solving the same crack problem using the BEM and FEM.

2.2 A Case Study-Crack Analysis

Cracks and flaws occur in many structures and components, sometimes leading to disastrous results. Fracture mechanics deals with the study of how a crack in a structure propagates under applied loads. The analytical predictions are made by calculating fracture parameters such as stress intensity factors (SIF) in the crack region, which can be used to estimate crack growth rate.

Stress intensity factors of an interface crack in a finite plate have been determined using the BEM and FEM. The BEM solver developed by Mr. Jianfeng Luo [3] has been used for the BEM analysis, and the commercial software ANSYS has been employed for the FEM analysis.
2.2.1 Problem Definition and Objective

- To analyze a center crack in a finite plate made of single and two materials.
- Calculate the values of stress intensity factors.
- Compare the accuracy and efficiency of the BEM and FEM approaches.

2.2.2 Data Given:

The plate model is shown in Fig. 2.1 with the following parameters.

\[
\frac{a}{w} = 0.2.
\]

Single material problem:

\[
E = 10 \text{ GPa}, \quad \nu = 0.3.
\]

Two material (Si$_3$N$_4$/S45C) problem:

\[
E_1 = 304 \text{ GPa}, \quad \nu_1 = 0.27, \\
E_2 = 206 \text{ GPa}, \quad \nu_2 = 0.30,
\]

where \(E\) is the Young’s modulus, \(\nu\) the Poison’s ratio of the material(s).

2.2.3 Boundary Conditions

Uniform pressure is applied on the upper edge of the plate, while the lower end of the plate is constrained as shown in Fig 2.1. The crack edges (faces) are not loaded (traction free). Due to the applied pressure at the upper end of the plate, crack faces will tear apart after the loading.
Fig. 2.1 Model of Interface Crack
2.2.4 Assumptions

- Material is isotropic.
- Material is homogeneous.
- Plane strain conditions are assumed for calculations.
- Environmental conditions such as temperature do not affect the structure.

2.2.5 Method of Approach

For the BEM analysis 1-D (line) quadratic elements are used on the boundary of the plate and the interface between the two materials (if it is a two-material problem) (cf., Fig. 1.1). For the FEM analysis 6-node triangular elements (called PLANE 2 in ANSYS) are used because they are more suitable for meshing the crack tip area. For the elements around the crack tip, the midpoint nodes (in the radial direction) are placed at the quarter point locations as shown Fig. 2.2. Such elements are called quarter-point (singular) elements which give better results for the stress fields near the crack tip (see, e.g., Ref. [4]).

![Fig.2.2 Singular Elements Near Crack Tip (FEM Model)](image-url)
There are two methods for calculating the stress intensity factors:

- Stress method which uses the calculated stress fields near the crack tip to determine the SIFs;
- COD (Crack Opening Displacement) method which uses the calculated displacement fields near the crack tip to determine the SIFs.

Both methods were tested in this study.

2.2.6 Procedure Using BEM

An input file is created manually which has a format shown in Appendix A. The input file contains the information about the geometry of the model as well as boundary conditions and material constants. The node coordinates are decided so that the elements at the crack tips will be a quarter point element which gives better results for the singular stress fields near the crack tip. The 2-D BEM solver is run and results are stored in an output file. Calculation procedure for determine the SIFs is the same as calculations using ANSYS as described below.

2.2.7 Procedure Using ANSYS

**Single material:** Quarter symmetry is used for the finite element model to save the storage and the CPU time.

- **Pre-processing:**
  - The element type is selected as PLANE 2.
  - To make the first layer of elements near the crack tip singular elements, the command **KSCON** is used. This command automatically generates singular
elements around a specific key point as shown in Fig 2.2. Other field in this command is used to control radius of the first layer of elements, number of elements in the circumferential direction, and so on.

- **Solution:**
  - Boundary conditions as shown in Fig. 2.1 are applied to the FEM model.
  - Symmetry BC is applied on the symmetry lines.
  - The model is solved.

- **Post-processing:**
  - The node number of quarter point of first element is found. The values of Y and X displacement at this node are found out, which will be sued to determine the SIFs
  - The deformation is plotted as shown in Fig.2.3.

  ![Fig 2.3 Crack Opening In Single Material](image)

  **Two materials:** Half symmetry is used.

- **Pre-processing:**
  The procedure is the same as the single material case.
Solution:

- The coincident nodes on the interface from the two materials are merged to ensure the continuity at the interface, except the nodes on the crack faces. At the crack faces there are two layers of elements. The upper layer belongs to domain 1 and lower belongs to domain 2.

Post-processing:

- The procedure is the same as the single material case.

2.2.8 Calculations

The approximate analytical expression [4] of SIF for the single material plate is:

\[ K_1 = \sigma \sqrt{\pi a} \frac{1 - 0.5a/w + 0.326a^2/w^2}{\sqrt{1 - a/w}} \]  \hspace{1cm} (2.1)

with \( a/w = 0.2, \ K_1 = 18093 \) in this particular case.

The expression for calculating the SIF for single as well as two material problems from crack opening displacement is [5]:

\[ K = \frac{2x(2\pi)^{1/2}}{cr^{1/2+i\epsilon}} \cosh(\pi \epsilon) x (1 + 2i \epsilon)(\nabla v + i \nabla u) \]

\[ K = K_1 + iK_2 \]

where

\[ c = \frac{k_1 + 1}{G_1} + \frac{k_2 + 1}{G_2} \]

\[ \epsilon = \frac{1}{2\pi} \ln\left(\frac{k_1}{G_1} + \frac{1}{G_2} / \left(\frac{k_2}{G_2} + \frac{1}{G_1}\right)\right) \]

\[ k_i = 3 - 4v_i \]
\( \nabla u \) and \( \nabla v \) represent the displacements of the nodes near the crack tip in X and Y directions, respectively, and \( r \) represents the distance of the node from crack tip.

The expression [5] for calculating the SIF for single as well as two material problems by using the stress method is:

\[
K_i + iK_2 = (\sigma_y + i\tau_{xy})\sqrt{2\pi r}(r / 2a)^{-i\varepsilon} \\
F_i = \frac{K_i}{T_0 \sqrt{\pi a}} \\
K_i = \sqrt{K_1^2 + K_2^2}
\]

(2.3)

where \( T_0 \) is the traction on the edge, \( a \) is half crack length, and \( r \) represents the distance of the node from crack tip.

According to published results [6], the COD method is found to be more accurate than the stress method. The reason is that the results from stress method are more sensitive to the element type and mesh size near the crack tip due to the singular behaviors of the stresses near the crack tip. The stress in the neighborhood of the crack tip changes rapidly. Theoretically the stress goes to infinity at the crack tip in one homogeneous material. If the node in this region is not close enough to the crack tip, the error of the stress and SIF will be large if the stress method is used.

2.2.9 Results:

The single material plate and two-material plate with the center crack are meshed using the BEM and FEM starting with smaller number of elements. Then the number of
elements are increased in both the BEM and FEM approaches until the results are converged. The converged BEM and FEM results for the SIFs with the final meshes are given below.

**SIF Using COD Method For Single Material Plate:**

<table>
<thead>
<tr>
<th>Method</th>
<th>BEM</th>
<th>FEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIF (N/m$^{3/2}$)</td>
<td>17693.67</td>
<td>18874.61</td>
</tr>
<tr>
<td>Error</td>
<td>2.2%</td>
<td>4.32%</td>
</tr>
<tr>
<td>No. of Elements</td>
<td>150</td>
<td>1230</td>
</tr>
<tr>
<td>Type of Element</td>
<td>Quadratic 1-D</td>
<td>Triangular 6-node</td>
</tr>
</tbody>
</table>

The error is calculated with respect to the analytical solution from Eq. (2.1) (equal to 18093).

**SIF Using COD Method For Two Material Plate:**

<table>
<thead>
<tr>
<th>Method</th>
<th>Using BEM</th>
<th>Using FEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>K$_2$/K$_1$</td>
<td>-0.023015</td>
<td>-0.0233662</td>
</tr>
<tr>
<td>Error</td>
<td>4.6%</td>
<td>6.21%</td>
</tr>
<tr>
<td>No. Of Elements</td>
<td>150</td>
<td>2460</td>
</tr>
<tr>
<td>Type Of Element</td>
<td>Quadratic 1-D</td>
<td>Triangular 6-node</td>
</tr>
</tbody>
</table>

For two material problems the value of K is an imaginary number (K= $K_1$ + i$K_2$). It is usually expressed as the ratio K$_2$/K$_1$ (dimensionless). The error is calculated with respect to published results by Yuuki (1991) [6] (K$_2$/K$_1$= -0.0220 for the material Si$_3$N$_4$/S45C).
2.3 Discussions

From this study, it was found that the meshing near the crack tip is very complicated due to the singularity of the stress field. The very fine mesh required near the crack tip makes the meshing a tedious job. Fortunately, for the BEM only the boundaries and interfaces need to be meshed as compared to the FEM where the whole domain need to be meshed. On the other hand, FEM requires more elements. FEM used 8 times more elements to solve the single material problem than the BEM. In case of the two material problem it took 16 times more elements than the BEM. Therefore, the BEM mesh generation can be a much easier task as compared to the FEM, if a good GUI can be developed for the pre-processing in the BEM.

During this study, however, the BEM meshes were generated manually, because there were no BEM meshers available for this purpose. Thus, the advantages of the BEM in meshing has not been demonstrated in this project. On the other hand, the BEM solver, developed in [3] and used for this testing, has been demonstrated in many cases to be a very accurate and efficient analysis software for 2-D stress analysis of a variety of structures or materials, including composite materials, thin films and coatings [3]. The lack of a pre- and post-processor (GUI) for this BEM code has limited its applications to real problems. All these limitations or needs prompted the development of a GUI using the Java language for this BEM solver, which will be described in the following two chapters.
Chapter 3. UCWaves as a Web-based Application

3.1 Introduction

Internet is fundamentally changing the way we communicate and interact in personal as well as business world. Users are just getting the feel of the Internet based applications, which are evolving. But after few years most of the applications will be web-based. There will a fundamental change in the architecture of the emerging software. So far major impact was in commerce industry, which is commonly referred as “E-Commerce”. But Internet is changing engineering industry as well. With these thoughts in mind, Dr. Yijun Liu and the author decided to develop a web-based version of UCWaves.

This is a first version of UCWaves on web. This application is entirely designed and implemented by the author.

3.2 Advantages of Web-Based Tool/Application

Following are major reasons for developing a web-based tool:

1) Collaboration With Partners

Users are seeing significant growth in collaboration of technology between different companies on national as well as international levels. So there is increasing demand for tools that make collaborative design and analysis easier. This is one of the aspects where Internet plays a role in mechanical engineering design and analysis.
In future UCWaves will be able to have capability to store the models on server, so that at any time there will be a central place where the engineers can find an updated model. No matter where user is sitting if one of users makes some changes in the model, then next person will be able to see the changes previous user has made. This makes real time design collaboration possible [7]. Also the shipping of drawings and 3-D models is faster and easier with client-server technology. On the UCWaves server some space will be allocated to every user group, which will be shared by all the users in that group.

As an example, companies can have one designer in Japan designing one component and another in Michigan designing another component. Then both the designers can assemble their parts real-time on the web. Suppose engineers at different locations need to run different types of analysis on the assembled part, it is also possible. Depending on the responsibility of the sub-designer, one engineer may be doing stress analysis and other engineer might be doing CFD analysis. Then engineers and managers can transmit the results back and forth.

2) Updating Software

Common problem in software industry is distributing and re-installing updated versions of the software to clients. This is very time-consuming for users as well as the software companies. If the application is web-based then once the software on server is updated, all users can use the updated version. This certainly makes the transition from version to version faster and easier. Also lot of money is saved. The frequency of updating software can be increased. This gives faster feedback to developers regarding their updates, in turn
debugging becomes easier and faster. Internet is also changing the design cycle of the software development. This will result in better analysis tools and in turn, the mechanical design and analysis will be more accurate.

3) Licensing Software

The licensing procedure becomes easier and more flexible. Licensing will be done on the basis of username and password. When users buy the web-based application that means users are buying some time for that particular username. Whenever that username logs into the system, the clock program on the server counts user’s usage time. Once user’s allocated time is over, that means the license has expired for that username. The user needs to buy some more time, to be able to connect to the server which hosts that particular application.

4) User Scope

User can search on the Internet and get access to the software. One of the shortcomings with BEM is that less number of people in industry knows about this technique and its advantages. Internet based software will increase the awareness and use of BEM in specific application.

5) Platform/Location Independence

No matter where the user is or what platform and operating system the user is using, he can still use the software as long as he can access Internet.
6) Thin/Faster Clients

Usually analysis problem take lot of CPU power, which slows down other processes on the user machine. With web-based application, the solver will be running on the server so the user can still use his computer for other application while user is waiting for the results. Usually server stations are faster, more robust and more scalable than PCs or workstations. Also depending on the server load (average number of users connected), more servers can be connected in parallel if needed to give faster access. This way even if many users are connected simultaneously, users will get faster response.

3.3 Author’s Contribution

The author was responsible for investigating the feasibility of the project, designing the architecture, implementing all the code, creating the web site, installation and administration of the server.

3.4 Design and Architecture

The present architecture of UCWaves is depicted in the Fig. 3.1. The design is based on 2-tier architecture where front end is browser and UCWaves Java Server (UJS) as well as Web Server serve as the backend. The applet reads the input file from client’s local disk and displays it on a new frame, which pops out of the browser window. Then the applet sends all the data to the server. When the client gives command like solve, plot, etc., these events starts a connection with UCWaves Java Server. According to the command send by the client, UCWaves Java Server invokes related methods. For example when client clicks on solve button, the UCWaves Java Server invokes the solver on the server
machine and creates the appropriate files. These files are read by the applet using web server and results are displayed on the client screen.

### 3.5 Implementation of the Main Web page

The main web page is not the simple html page, it is constructed using embedded html. Embedded html consists of the simple html tags along with embedded objects. Following lines shows one of the object tags. This tag tells browser to open the object denoted by the classid. In this example the classid belongs to the java Plug-in. When browser sees this tag, it tries to open the java Plug-in. The java Plug-in classid can be obtained from the installation directory of the Plug-in.

```html
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
  WIDTH = 500 HEIGHT = 500
codebase="http://java.sun.com/products/plugin/1.2/jinstall-12-win32.cab#Version=1,2,0,0">
  <PARAM NAME = CODE VALUE = "classes.graphics.UCWaves">
  <PARAM NAME = ARCHIVE VALUE = "ucwaves1.jar">
>
```

Thus the browser calls UCWaves web page from UCWaves web server, this web page calls Plug-in using embedded objects from local hard disk, the Plug-in calls applet which is downloaded from UCWaves web server, then the applet connects to UJS.
Fig. 3.1 Present 2-Tier Architecture

- Web Server
  - Files
  - UCWaves Java Server
  - C++ Solver

- Web Browser (User 1)
  - Applet
  - Local Disk

- Web Browser (User 2)
  - Applet
  - Local Disk

- Web Browser (User 3)
  - Applet
  - Local Disk
3.6 Implementation of Applet

The basic class is called UCWaves.java, which extend applet class. The applet basically initiates a frame, which pops out of the browser window. This frame looks same as the standalone application, but actually it comes from Internet. The files are transferred to and from server using socket streams. Developing applet for UCWaves involved the following tasks:

3.6.1 Security Issues

Ucwaves is designed in such a way that a user should be able to upload a model file on web and view it in GUI. Due to security issues, applets are not allowed to access the local hard-disk [8]. To get around security restrictions imposed by browser as well as JVM (Java Virtual Machine), user needs to install a policy file [9].

3.6.2 What Is Policy File?

One way the Java platform provides protection from attack from a virus, for example, is through the use of a security manager. Currently JDK system code invokes security manager methods to perform resource access control checks.

Most browsers install a security manager, so applets typically run under the scrutiny of a security manager. A applet is not allowed to access resources unless it is explicitly granted permission to do so by the security policy in effect. In Java platforms that are compatible with JDK 1.2, the permission must be granted by an entry in a policy file.
A policy file is an ASCII text file and can be composed via a text editor or the graphical Policy Tool utility (Comes with JDK). Users can install following policy file on their machines.

```java
/* AUTOMATICALLY GENERATED ON Wed Apr 19 18:43:14 EDT 2000*/
/* DO NOT EDIT */
grant codeBase "http://twister.mie.uc.edu/-" {
    permission java.security.AllPermission;
};
```

Basically the policy file is saying that give all permissions to an applet coming from http://twister.mie.uc.edu/.

### 3.6.3 Installing Policy File

Once this policy file is created or copied from UCWaves web site to user’s machine. User need to register it in java.security file in his `${java.home}` directory. This is the directory where user has installed JRE (Java Run-time Environment). Usually this directory has following path :

```
c:\program files\javasoft\jre\1.2\lib\security
```

In this directory locate a file called `java.security`. Find the following two lines in this file:

```
policy.url.1=file:${java.home}/lib/security/java.policy

policy.url.2=file:${user.home}/.java.policy
```

Add the following line after these two lines

```
policy.url.3=file:/c:\DirName\FileName.txt
```

The DirName is the name of the directory where user has stored policy file. FileName.txt is the name of the policy file.
3.6.5 Java Plug-in

Java Plug-in Software enables customers to direct applets to run using Sun's virtual machine instead of the web browser's default virtual machine. Virtual machine is the component needed to run Java class files. So instead of using browser’s virtual machine, java applet will use virtual machine residing inside the Plug-in. This will provide consistency and reliability when running applets. This is important because different browsers have different virtual machine which causes problems while executing Java applets.

3.6.6 Why need Plug-in?

UCWaves applet contains swing component. Swing components are graphical components used to create the GUI. But present versions of Netscape and Internet explorer browser do not support it. Therefore java Plug-in is needed to run the application in the browser.

3.7 UCwaves Java Server (UJS)

UJS is a multi-threaded java program which acts like a server. The server listens on a port for the applet clients. Whenever new client comes on the web site, UJS creates a new thread. According to the commands send by each client, the server invokes solver or creates files for plotting. The solver is invoked in the same manner as the standalone application by creating a new process [explained in section 4.4.2]. The files are transferred to and from server using socket streams.
The UJS is installed on the machine called twister.mie.uc.edu as a NT service. This service is always running on that machine even if nobody is logged on to the machine.

### 3.8 Installing and Configuring Web Server

The author started out installing Netscape FastTrack server, but after extensive testing, the author found out that it cannot handle high load of users. Therefore author decided to use Netscape Enterprise Server. Author also configured the security and other aspects of this web server.

### 3.9 Future Work

It is a better idea to use 3-tier architecture as shown in Fig 3.2. This architecture will give faster results and better user interaction with the solver. This will also take care of the linking of UCWaves with different solvers for example acoustic solver, 3-D BEM solver. The use of database will help users to store their input and result files on server. This will save some time for users in uploading and downloading files. Also there will be a central place where users can store files. So all the engineers working on the same project can access the same updated files simultaneously. This will speed up the mechanical engineering design and analysis process.
Fig. 3.2 Improved 3-Tier Architecture
3.10 Conclusion

Certainly a web-based commercial BEM application will be a breakthrough in BEM analysis. UCWaves can view the model on the web and solve it. At present UCWaves is the only web-based application developed using BEM. It cannot be used for commercial applications right now, because it needs some enhancements. But it can be used as a foundation for 3-D BEM web-based application and as an academic tool.
Chapter 4. UCWaves Standalone Application

4.1 Introduction

“UCWaves” project was originally started in March 1998 by graduate students Mr. Shailesh Karmarkar and Mr. Jianfeng Luo from University of Cincinnati under the guidance of Dr. Yijun Liu. Mr. Shailesh Karmarkar developed a GUI using Java and Mr. Jianfeng Luo developed a 2-D BEM solver using VC++. Later Mr. Kumar Puppala and Ms. Sridevi Kolluri enhanced the GUI. Later the author added new functionality and debugged this application. Creating a solid model and meshing is not the part of this project. The idea is that the model will be created using some sophisticated CAD tools like I-deas, ANSYS, etc., and the input file will be translated to UCWaves input format. The input file is a text file. When the user opens this file in GUI, graphical representation of the model along with the mesh is plotted on the screen. Once the user gives the solve command by clicking on the solve button, the input file is send to the 2-D BEM C++ solver. The solver creates an output file. This file is used to plot the deformation of the structure once user clicks on plot displacement button.

4.2 Basic Architecture of GUI

The basic class to display the geometry and boundary conditions is called Mesh. Fig.4.1 shows the architecture of this class [10]. The figure uses Booch notations [11]. The class Element1D is a derived class of the Element class. Element1D represents 1-D quadratic element that is used to solve 2-D BEM problems. Similarly Traction1D represents derived class of Traction class.
An A is always associated with one B
An A is always associated with one or more B
An A is always associated with zero or more B

Fig 4.1 Mesh Class And Notations

<table>
<thead>
<tr>
<th>Booch</th>
<th>An A is always associated with one B</th>
<th>An A is always associated with one or more B</th>
<th>An A is always associated with zero or more B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mesh</td>
<td>1</td>
<td>+</td>
<td>?</td>
</tr>
<tr>
<td>Element</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traction1D</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.3 Solver

Mr. Jianfeng Luo developed the first version of the 2-D BEM solver using C++ on NT platform. Miss Nan Xu updated the solver to handle composite materials and fracture mechanics problems [12].

4.4 Enhancements Added To UCWaves

Apart from designing and implementing Internet based UCWaves application from scratch, the author also significantly enhanced the latest version of UCWaves software.

4.4.1.1 Reading Scientific notations

The previous version of the GUI was not able to read numbers from input and output files, which are in scientific format for example 123e-12, 23e+2. This was a major bug because GUI while reading input and output files used to treat scientific numbers like 123e-12 as 123 and next number as –12. In previous version the data was read by the GUI using StreamTokenizer class from Java Development Kit (JDK) library. Previous students who worked on UCWaves project thought that the problem is with UCWaves classes and not with SUN’s original java classes. First step the author took is to figure out which class exactly is causing problem. Author found out that StreamTokenizer class has limitation due to which it cannot read scientific notation.

To fix this bug, SUN’s original class has been modified to read scientific notations. This new class called SciStreamTokenizer.java is installed in utils package. The author obtained the original source code of the streamtokenizer.java class from SUN. Being the original SUN class, it was very difficult to understand what algorithm SUN is using to
tokenize the data that comes into compiler through input streams. The task of understanding the original algorithm was difficult because only documentation available was the few comments in the code. It took the author few days to figure out what lines need to be changed in that huge source file to be able to read scientific notations.

### 4.4.1.2 How Does the StreamTokenizer Class Work?

The StreamTokenizer class takes an input stream and parses it into "tokens", allowing the tokens to be read one at a time. The parsing process is controlled by a table and a number of flags that can be set to various states. The stream tokenizer can recognize identifiers, numbers, quoted strings, and various comment styles.

Each byte read from the input stream is regarded as a character in the range ' ' through 'ÿ'. The character value is used to look up five possible attributes of the character: *white space*, *alphabetic*, *numeric*, *string quote*, and *comment character*. Each character can have zero or more of these attributes.

In addition, an instance has four flags. These flags indicate:

- Whether line terminators are to be returned as tokens or treated as white space that merely separates tokens.
- Whether C-style comments are to be recognized and skipped.
- Whether C++-style comments are to be recognized and skipped.
- Whether the characters of identifiers are converted to lowercase.
A typical application first constructs an instance of this class, sets up the syntax tables, and then repeatedly loops calling the nextToken method in each iteration of the loop until it returns the value TT_EOF. TT_EOF is a constant indicating that the end of the stream has been read.

### 4.4.1.3 How Does the SciStreamTokenizer Class Work?

There are 10 fields that are used in this class. The most important one is “ttype” which contains the type of the token just read. Its value is one of the following:

- TT_WORD indicates that the token is a word.
- TT_NUMBER indicates that the token is a number.
- TT_EOL indicates that the end of line has been read.
- TT_EOF indicates that the end of the input stream has been reached.

The e- / e+ part of a construction (entity just read) like e-[^0-9] or e+[^0-9] will be considered TT_NUMBER instead of a new TT_WORD, with value $10^x$ where $x$ is the number next to e or E. Appropriate changes were made in the code to be able to recognize the significance of character “e or E” inside a number.

### 4.4.2.1 Linking GUI and Solver

The previous version of the GUI was not able to properly link the model to solver. This was not very straightforward because the GUI is written in Java and Solver is a C++ executable. The communication between the solver and the GUI is through files. When the GUI opens a model file, that file is copied to Solver directory. The solver executable looks for input.dat file in solver directory to solve and solves it. The solver writes its
output to a file called output.dat. This file is written to the solver directory. While displaying post-processing results the data is read from output.dat file. To invoke the C++ solver Java.lang.Runtime class and its methods are used.

4.4.2.2 What is a Java Runtime?

Every Java application has a single instance of class Runtime that allows the application to interface with the environment in which the application is running. The current runtime can be obtained from the `getRuntime()` Method in `java.lang.Runtime` class. The idea is by obtaining current Runtime, UCWaves can talk to the current environment and ask this environment to start a new process, which will invoke C++ solver.

For starting new process, method `Runtime.exec()` is used. This method creates a new process and returns an instance of a subclass of Process. The process object obtained by UCWaves is used to control the process and obtain information about it. This mean UCWaves can perform input to the process, wait for the process to complete, check the exit status of the process, and destroy (kill) the process. The word process in the context of UCWaves means “Process of Solving BEM problem using C++ Program”. Use of `waitFor()` method from `java.lang.Process` class causes the current thread (Java UCWaves program thread) to wait, until the solver process (C++ program) has terminated.

4.4.3 Handling Multi-Domain Problems
The previous version of the GUI was not able to handle multi-domain problems. The first step to enhance UCWaves to be able to handle multi-domains was to read how many domains the current problem is handling. Once UCWaves find out that current problem handles n domains, next step is to create n different data sets from the data parsed from the input file and output file. Each of these data sets consist of

- Total nodes and their coordinate in that particular domain
- Total elements and their connectivity in that particular domain
- Displacement boundary conditions
- Traction boundary conditions
- Material constants
- Displacement data after solving from output file
- Traction data after solving from output file

The hardest part was to be able to map the raw data obtained from input file to the respective domain and create n data sets. For single domain problems it is easy because all the boundary conditions or other data obtained need not be classified according to domain. Next step was to display these data sets in different colors to be able to differentiate the domains.

To start with this author designed and implemented algorithm to handle 2 domains and then 3. After successfully testing 3 domain problems, author implemented n domain algorithm. In the current version of UCWaves, user can view as many domains as needed. Each domain is drawn in different color, for example first domain is drawn in red color, second in green, etc. Fig 4.2 and Fig 4.3 show the two domains model before and
after solving. The deformed shape is plotted on top of the existing model. To distinguish the original and deformed shape, the deformed shape is plotted in white color.

Fig 4.2 Model Before Solving
Fig 4.3 After Solving-deformed Shape
4.4.4 Representation of Nodes

The position of nodes is represented by small circles. The nodes at the end of the elements are bigger than the nodes at the center as shown in Fig 4.2 and 4.3. This helps to differentiate middle nodes from end nodes. The important part of the implementation of this feature was to be able to use element information to find out the end nodes. Then use it to find the coordinates of end nodes.

4.4.5 Positioning Model

In previous version of UCWaves, depending on the model coordinates; the model could be positioned in any corner of the screen. This author implemented new classes which calculate the transformation matrix to shift the model to the center of the screen. This matrix is added to original coordinate matrix, so that model is moved to the center. Then magnification factor is calculated so that after multiplying it to the coordinate matrix, the model will be resized to fit the entire screen. This helps user to use entire screen area.

4.5 Future Scope for Development

At present UCWaves can be used as an academic tool, but to make it full-fledged software many improvements are needed. The major enhancements are needed in following areas:

4.5.1 Handling 3-D model

At present UCWaves can read only 2-D input file. But by changing the function, which read data to read 3 co-ordinates instead of 2, UCWaves can be extended to 3-D. To plot
and view Java 3-D model, SUN’s Java 3D API can be used. The Java 3D API is a set of classes for writing 3-D graphics applications and 3D applets. It gives developers high level constructs for creating and manipulating 3D geometry and for constructing the structures used in rendering that geometry.

4.5.2 Better Viewing Functionality

After adding 3-D, the complexity of the model will increase. Therefore, the pan, zoom and rotate function should be enhanced to give better representation of model during pre-processing as well as post-processing.

4.5.3 Invoking Solver Methods

At present the GUI is communicating to solver through files. But as solver will become more complicated, it will be necessary to have a mechanism to call the methods in the solver individually instead of executing a new process. This can be done using CORBA or JNI. The Common Object Request Broker Architecture (CORBA) provides a platform-independent, language-independent architecture for writing distributed, object-oriented applications. Java Native Interface (JNI) is the native programming interface for Java that is part of the JDK. JNI allows Java code that runs within a Java Virtual Machine (JVM) to operate with applications and libraries written in other languages, such as C, C++, and assembly. By writing programs using the JNI, it is ensured that the code is completely portable across all platforms.
4.5.4 Developing UNIX version of UCWaves

There are very few changes needed for UNIX version of GUI. One of them is to change the file path separators because for windows file separator is ‘\’ and for UNIX it is ‘/’. Also C++ solver is developed using Visual C++ which creates an windows application. So solver code need to be modified to run on UNIX systems so that UCWaves can call solver on UNIX machines.

4.6 Conclusion

Many enhancements have been added to the present version of UCWaves. At present UCWaves can handle all the cases of 2-D BEM problems including multi-domains. Next major change is to make it handle 3-D problems. Also the invocation of solver can be made dynamic and faster using Corba or JNI technology.
# Appendix A

## The Input and Output Files

Following is an input file, which is a two-domain problem. Followed by the file is the graphical representation of the file using UCWaves.

```
$ A Square Unit Cell Model of a Fiber and Matrix          March 23, 2000
$ProblemType: StressAnalysis
$Dimension: 2D
$ElementType: Quadratic
$NoofDomain: 2
$NoofNodes: 52
$Node
1   -10.655 -10.655 0.0
2   -7.9912 -10.655 0.0
3   -5.3275 -10.655 0.0
4   -2.6638 -10.655 0.0
5     0.0  -10.655 0.0
6    2.6638  -10.655 0.0
7    5.3275  -10.655 0.0
8    7.9912  -10.655 0.0
9   10.655   -10.655 0.0
10  10.655  -10.655 0.0
11  10.655   -7.9912 0.0
12  10.655   -5.3275 0.0
13  10.655   -2.6638 0.0
14  10.655      0.0 0.0
15  10.655    2.6638 0.0
16  10.655    5.3275 0.0
17  10.655    7.9912 0.0
18  10.655   10.655 0.0
19  10.655   10.655 0.0
20   7.9912  10.655 0.0
21   5.3275  10.655 0.0
22   2.6638  10.655 0.0
23     0.0   10.655 0.0
24   -2.6638 10.655 0.0
25   -5.3275 10.655 0.0
26   -7.9912 10.655 0.0
27  -10.655 10.655 0.0
28  -10.655 10.655 0.0
29  -10.655  7.9912 0.0
30  -10.655   5.3275 0.0
31  -10.655  2.6638 0.0
32  -10.655     0.0 0.0
33  -10.655  -2.6638 0.0
34  -10.655  -5.3275 0.0
35  -10.655  -7.9912 0.0
36  -10.655  -10.655 0.0
```
<table>
<thead>
<tr>
<th>Node</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>9.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>38</td>
<td>8.7769</td>
<td>-3.6355</td>
<td>0.0</td>
</tr>
<tr>
<td>39</td>
<td>6.7175</td>
<td>-6.7175</td>
<td>0.0</td>
</tr>
<tr>
<td>40</td>
<td>3.6355</td>
<td>-8.7769</td>
<td>0.0</td>
</tr>
<tr>
<td>41</td>
<td>0.0</td>
<td>-9.5</td>
<td>0.0</td>
</tr>
<tr>
<td>42</td>
<td>-3.6355</td>
<td>-8.7769</td>
<td>0.0</td>
</tr>
<tr>
<td>43</td>
<td>-6.7175</td>
<td>-6.7175</td>
<td>0.0</td>
</tr>
<tr>
<td>44</td>
<td>-8.7769</td>
<td>-3.6355</td>
<td>0.0</td>
</tr>
<tr>
<td>45</td>
<td>-9.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>46</td>
<td>-8.7769</td>
<td>3.6355</td>
<td>0.0</td>
</tr>
<tr>
<td>47</td>
<td>-6.7175</td>
<td>6.7175</td>
<td>0.0</td>
</tr>
<tr>
<td>48</td>
<td>-3.6355</td>
<td>8.7769</td>
<td>0.0</td>
</tr>
<tr>
<td>49</td>
<td>0.0</td>
<td>9.5</td>
<td>0.0</td>
</tr>
<tr>
<td>50</td>
<td>3.6355</td>
<td>8.7769</td>
<td>0.0</td>
</tr>
<tr>
<td>51</td>
<td>6.7175</td>
<td>6.7175</td>
<td>0.0</td>
</tr>
<tr>
<td>52</td>
<td>8.7769</td>
<td>3.6355</td>
<td>0.0</td>
</tr>
</tbody>
</table>

$NoofInterphaseNode: 16
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
$NoofNodesinCurrentDomain: 52
$NoofElements: 24

$Element
1 1 2 3
2 3 4 5
3 5 6 7
4 7 8 9
5 10 11 12
6 12 13 14
7 14 15 16
8 16 17 18
9 19 20 21
10 21 22 23
11 23 24 25
12 25 26 27
13 28 29 30
14 30 31 32
15 32 33 34
16 34 35 36
17 37 38 39
18 39 40 41
19 41 42 43
20 43 44 45
21 45 46 47
22 47 48 49
23 49 50 51
24 51 52 37

$DisplacementBoundaryCondition(NodeId, ComponentId, Value)
28 1 0.0
29 1 0.0
30 1 0.0
31 1 0.0
32 1 0.0 2 0.0
33 1 0.0
34 1 0.0
$\text{TractionBoundary-Condition(ElementId,Direction,Value@nodes)}$
5 1 1.0 1.0
6 1 1.0 1.0
7 1 1.0 1.0
8 1 1.0 1.0
$
$
$\text{NoofNodesinCurrentDomain: 16}$
$\text{NoofElements: 8}$
$\text{Element}$
1 37 52 51
2 51 50 49
3 49 48 47
4 47 46 45
5 45 44 43
6 43 42 41
7 41 40 39
8 39 38 37
$\text{DisplacementBoundary-Condition(NodeId,ComponentId,Value)}$
$\text{TractionBoundary-Condition(ElementId,Direction,Value@nodes)}$
$
$\text{MaterialConstants(E,\mu)}$
0.9467455
0.2307692
0.9467455
0.2307692
Model Plot

The Figure shows the model of the input file.

Output File

$A Square Unit Cell Model of a Fiber and Matrix March 23, 2000

$Node Coordinates:

1  -10.655 -10.655  0
2  -7.9912 -10.655  0
3  -5.3275 -10.655  0
4  -2.6638 -10.655  0
5   0  -10.655  0
6   2.6638 -10.655  0
7   5.3275 -10.655  0
8   7.9912 -10.655  0
9  10.655  -10.655  0
10 10.655  -10.655  0
1  10.655  -7.9912  0
2  10.655  -5.3275  0
3  10.655  -2.6638  0
4  10.655   0   0
5  10.655   2.6638  0
6  10.655   5.3275  0
7  10.655   7.9912  0
8  10.655  10.655  0
9  10.655  10.655  0
10 7.9912  10.655  0
11 5.3275  10.655  0
12 2.6638  10.655  0
13  0  10.655  0
14 -2.6638  10.655  0
15 -5.3275  10.655  0
16 -7.9912  10.655  0
17 -10.655  10.655  0
18 -10.655  10.655  0
19 -10.655   0   0
20 -10.655 -2.6638  0
21 -10.655 -5.3275  0
22 -10.655 -7.9912  0
23 -10.655 -10.655  0
24  0   2.6638  0
25  0   5.3275  0
26  0   7.9912  0
27  0   10.655  0
28  0   10.655  0
29  0    0   0
30  0   -2.6638  0
31  0    -5.3275  0
32  0    -7.9912  0
33  0    -10.655  0
34  0   -10.655  0
35  0    -10.655  0
36  0   -10.655  0
37  0     9.5   0
38  0    -8.7769  0
39  0    -6.7175  0
40  0   -3.6355  0
41  0   -9.5   0
42  0    -8.7769  0
43  0    -6.7175  0
44  0   -3.6355  0
45  0   -9.5   0
46  0    -8.7769  0
47  0    -6.7175  0
48  0   -3.6355  0
49  0     9.5   0
50  0    8.7769  0
51  0    6.7175  0
52  0    3.6355  0

$\text{Displacement:}$

1  -2.12885e-014  3.19643
2    2.6637  3.19623
3    5.32782  3.19639
4    7.99222  3.19692
5    10.6553  3.19682
6    13.3185  3.19689
7    15.9829  3.19632
8    18.6471  3.19617
9    21.311   3.19656
10   21.311   3.19656
11   21.3106  2.39748
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>21.31</td>
<td>1.5983</td>
</tr>
<tr>
<td>13</td>
<td>21.3112</td>
<td>0.798713</td>
</tr>
<tr>
<td>14</td>
<td>21.311</td>
<td>-3.00599e-014</td>
</tr>
<tr>
<td>15</td>
<td>21.3112</td>
<td>-0.798713</td>
</tr>
<tr>
<td>16</td>
<td>21.31</td>
<td>-1.5983</td>
</tr>
<tr>
<td>17</td>
<td>21.3106</td>
<td>-2.39748</td>
</tr>
<tr>
<td>18</td>
<td>21.311</td>
<td>-3.19656</td>
</tr>
<tr>
<td>19</td>
<td>21.311</td>
<td>-3.19656</td>
</tr>
<tr>
<td>20</td>
<td>18.6471</td>
<td>-3.19617</td>
</tr>
<tr>
<td>21</td>
<td>15.9829</td>
<td>-3.19632</td>
</tr>
<tr>
<td>22</td>
<td>13.3185</td>
<td>-3.19689</td>
</tr>
<tr>
<td>23</td>
<td>10.6553</td>
<td>-3.19682</td>
</tr>
<tr>
<td>24</td>
<td>7.99222</td>
<td>-3.19692</td>
</tr>
<tr>
<td>25</td>
<td>5.32782</td>
<td>-3.19639</td>
</tr>
<tr>
<td>26</td>
<td>2.6637</td>
<td>-3.19623</td>
</tr>
<tr>
<td>27</td>
<td>6.21224e-015</td>
<td>-3.19643</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>-3.19643</td>
</tr>
<tr>
<td>29</td>
<td>0</td>
<td>-2.39737</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>-1.59845</td>
</tr>
<tr>
<td>31</td>
<td>0</td>
<td>-0.798881</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>33</td>
<td>0</td>
<td>0.798881</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
<td>1.59845</td>
</tr>
<tr>
<td>35</td>
<td>0</td>
<td>2.39737</td>
</tr>
<tr>
<td>36</td>
<td>0</td>
<td>3.19643</td>
</tr>
<tr>
<td>37</td>
<td>20.1558</td>
<td>-3.23588e-014</td>
</tr>
<tr>
<td>38</td>
<td>19.4324</td>
<td>1.09061</td>
</tr>
<tr>
<td>39</td>
<td>17.3729</td>
<td>2.01512</td>
</tr>
<tr>
<td>40</td>
<td>14.2908</td>
<td>2.63308</td>
</tr>
<tr>
<td>41</td>
<td>10.6553</td>
<td>2.85034</td>
</tr>
<tr>
<td>42</td>
<td>7.01985</td>
<td>2.63312</td>
</tr>
<tr>
<td>43</td>
<td>3.93773</td>
<td>2.01516</td>
</tr>
<tr>
<td>44</td>
<td>1.87844</td>
<td>1.09055</td>
</tr>
<tr>
<td>45</td>
<td>1.15536</td>
<td>-5.72187e-016</td>
</tr>
<tr>
<td>46</td>
<td>1.87844</td>
<td>-1.09055</td>
</tr>
<tr>
<td>47</td>
<td>3.93773</td>
<td>-2.01516</td>
</tr>
<tr>
<td>48</td>
<td>7.01985</td>
<td>-2.63312</td>
</tr>
<tr>
<td>49</td>
<td>10.6553</td>
<td>-2.85034</td>
</tr>
<tr>
<td>50</td>
<td>14.2908</td>
<td>-2.63308</td>
</tr>
<tr>
<td>51</td>
<td>17.3729</td>
<td>-2.01512</td>
</tr>
<tr>
<td>52</td>
<td>19.4324</td>
<td>-1.09061</td>
</tr>
</tbody>
</table>

$Traction:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
13  1  0
14  1  0
15  1  0
16  1  0
17  1  0
18  1  0
19  0  0
20  0  0
21  0  0
22  0  0
23  0  0
24  0  0
25  0  0
26  0  0
27  0  0
28  -0.999961  0
29  -0.999968  0
30  -1.00041  0
31  -0.999792  0
32  -1.00017  2.47718e-016
33  -0.999792  0
34  -1.00041  0
35  -0.999968  0
36  -0.999961  0
37  -1.00095  5.41038e-015
38  -0.9239  -1.62658e-005
39  -0.707697  1.8818e-005
40  -0.38271  -7.43545e-006
41  3.57879e-006  -0.000121647
42  0.382706  -7.43545e-006
43  0.707742  1.13007e-006
44  0.923905  1.71261e-005
45  1.00084  1.01462e-015
46  0.923905  -1.71261e-005
47  0.707742  -1.13007e-006
48  0.382706  7.43545e-006
49  3.57879e-006  0.000121647
50  -0.38271  9.5296e-007
51  -0.707697  -1.8818e-005
52  -0.9239  1.62658e-005
Displacement Plot

After solving the input file mentioned above, we get the following displacement plot using UCWaves.
Appendix B

Listing of Code

Ucwaves class is introduced for converting application into applet. This class calls the FrameMain class. FrameMain class extends Frame class and is similar to FrameMain class in standalone application.

/* This class is extension of the Applet class. This class gets the appplet
 * context and creates a new frame in that context by calling FrameMain class .
 * @author Sachin Chaudhari
 */

package classes.graphics;

import java.applet. *
;

public class UCWaves extends Applet {

    public static void main(String args[]) {
        UCWaves app = new UCWaves();
        System.out.println("Loading UCWaves...Please wait ...");
        app.init();
        app.start();
    }

    public void init() {
        System.out.println("Loading UCWaves...Please wait ...");
        AppletContext ac = null;
        try{ac = getAppletContext();}
        catch(NullPointerException npe){}
        new FrameMain(ac);
    }
}

/* This simple extension of the java.awt.Frame class contains all the elements
 * necessary to act as the main window of an application.
 * @author Shailesh Karmarkar and Sachin Chaudhari
 */

package classes.graphics;

import java.applet. *
import java.awt.event. *
import java.awt. *
import javax.swing. *
import java.io. *
import classes.mesh. *
import classes.graphics. *
import classes.geometry. *
import classes.utils. *
import java.util. *
import java.net. *;
public class FrameMain extends Frame{

        //Progress bar to display the progress when the application loads
        public static JLabel progressLabel = null;
        public static JProgressBar progressBar = null;
        public static int currentProgressValue;
        private static boolean flagForpaint = false;

        static AppletContext ac;
        //Screen constants
        private int screenHeight, screenWidth, mnuFrameHeight;
        //public Mesh mesh1;
        public String fileName="";
        //dialog box for selecting the input file
        FileDialog openMyFileDialog,saveMyFileDialog,saveMyFileDialog1;

        //model will be displayed in this window
        DialogModel modelDialog;
        //QuitDialog qd;
        //pan/zoom/rotate window
        DialogPZR dialogPZR;
        //boundary condition window
        DialogBC dialogBCondition;
        //selection mode dialog box
        DialogSelectionMode selectMode;
        //delete boundary conditions dialogbox
        DialogDeleteBC dialogDeleteDisp;
        DialogDeleteTraction dialogDeleteTrac;
        //list dialog box
        ListFileDialog lstEntity;

        MenuBar mainMenuBar;
        Menu fileMenu;
        MenuItem fileOpen;
        MenuItem fileSave;
        MenuItem filePrint;
        MenuItem fileExit;
        Menu listMenu;
        Menu listFiles;
        MenuItem fileInput;
        MenuItem fileRes;
        MenuItem listNodes;
        MenuItem listElements;
        Menu listProp;
        MenuItem propEleType;
        MenuItem propMat;
        Menu list Loads;
        MenuItem loadDof;
        MenuItem loadForce;
        MenuItem loadSurf;
        MenuItem loadBody;
        Menu listOther;
        MenuItem otherSys;
        Menu grafMenu;
        MenuItem grafPZR;
Menu viewMenu;
MenuItem direcView;
MenuItem magView;
MenuItem perspecView;
MenuItem symbolMenu;
Menu styleMenu;
MenuItem hiddenStyle;
MenuItem contourStyle;
MenuItem lightStyle;
Menu scaleMenu;
MenuItem dispScale;
Menu selectMenu;
CheckboxMenuItem selectNode;
CheckboxMenuItem selectElement;
Menu plotMenu;
MenuItem plotRef;
MenuItem plotNodes;
MenuItem plotElements;
Menu preMenu;
MenuItem preMaterial;
MenuItem applyBC;
Menu deleteBCMenu;
MenuItem deleteDisp;
MenuItem deleteTrac;
MenuItem prepQuery;
Menu solveMenu;
MenuItem solveSave;
MenuItem solveStart;
MenuItem saveResult;
Menu postMenu;
Menu listPost;
MenuItem listDisp;
MenuItem listStress;
Menu plotPost;
MenuItem plotDisp;
MenuItem plotStress;
Menu helpMenu;
Menu Examples;
MenuItem CrackFile, HoleInPlate, TeeFile, SquarePlateFile;
MenuItem helpAbout;

/***Toolbar data members***/
JToolBar toolBar;
JButton openFileButton, saveButton, printButton;
JButton symbolButton, pzcButton;

public FrameMain(AppletContext ac) {
    super();
    this.ac = ac;
    Font fBold = new Font("TimesRoman", Font.BOLD, 14);
    Font fPlain = new Font("TimesRoman", Font.BOLD, 12);

    setLayout(new BorderLayout(0,0));
    setVisible(false);
    setResizable(true);
setSize(getInsets().left + getInsets().right + 403, getInsets().top + getInsets().bottom +
327);
//setSize(getInsets().left + getInsets().right + 403, getInsets().top + getInsets().bottom +
150);
setBackground(Color.white);
openMyFileDialog = new java.awt.FileDialog(this);
openMyFileDialog.setMode(FileDialog.LOAD);
openMyFileDialog.setTitle("Open");
saveMyFileDialog = new java.awt.FileDialog(this);
saveMyFileDialog.setMode(FileDialog.SAVE);
saveMyFileDialog.setTitle("Save As");
saveMyFileDialog1 = new java.awt.FileDialog(this);
saveMyFileDialog1.setMode(FileDialog.SAVE);
saveMyFileDialog1.setTitle("Save As");
setTitle("UCWaves ..............");

mainMenuBar = new MenuBar();
mainMenuBar.setFont(new Font("TimesRoman", Font.BOLD, 14));
setMenuBar(mainMenuBar);

fileMenu = new Menu("File");
fileOpen = new MenuItem("Open Model File");
fileOpen.setFont(fPlain);
fileMenu.add(fileOpen);

fileSave = new MenuItem("Save Model File");
fileSave.setFont(fPlain);
fileMenu.add(fileSave);
filePrint = new MenuItem("Print");
filePrint.setFont(fPlain);
fileMenu.add(filePrint);
fileMenu.addSeparator();
fileExit = new MenuItem("Exit");
fileExit.setFont(fPlain);
fileMenu.add(fileExit);
mainMenuBar.add(fileMenu);
listMenu = new Menu("List");
listMenu.setFont(fBold);
listFiles = new Menu("File");
listFiles.setFont(fPlain);
listMenu.add(listFiles);
fileInput = new MenuItem("Input File");
fileInput.setFont(fPlain);
listFiles.add(fileInput);
fileRes = new MenuItem("Result File");
fileRes.setFont(fPlain);
listFiles.add(fileRes);
listMenu.add(listFiles);
listNodes = new MenuItem("Nodes");
listNodes.setFont(fPlain);
listMenu.add(listNodes);
listElements = new MenuItem("Elements");
listElements.setFont(fPlain);
listMenu.add(listElements);
listProp = new MenuItem("Properties");
listProp.setFont(fPlain);
propEleType = new MenuItem("Element Type");
propEleType.setFont(fPlain);
listProp.add(propEleType);
propMat = new MenuItem("Material");
propMat.setFont(fPlain);
listProp.add(propMat);
listMenu.add(listProp);
listLoads = new Menu("Loads");
listLoads.setFont(fPlain);
loadDof = new MenuItem("Dof Constraints");
loadDof.setFont(fPlain);
listLoads.add(loadDof);
loadForce = new MenuItem("Force");
loadForce.setFont(fPlain);
listLoads.add(loadForce);
loadSurf = new MenuItem("Surface Loads");
loadSurf.setFont(fPlain);
listLoads.add(loadSurf);
loadBody = new MenuItem("Body Loads");
loadBody.setFont(fPlain);
listLoads.add(loadBody);
listMenu.add(list Loads);
listOther = new Menu("Other");
listOther.setFont(fPlain);
otherSys = new MenuItem("Local Coordinate System");
otherSys.setFont(fPlain);
listOther.add(otherSys);
listMenu.add(listOther);
mainMenuBar.add(listMenu);
grafMenu = new Menu("Graphics");
grafMenu.setFont(fBold);
grafPZR = new MenuItem("Pan/Zoom/Rotate");
grafPZR.setFont(fPlain);
grafMenu.add(grafPZR);
viewMenu = new Menu("View Settings");
viewMenu.setFont(fPlain);
direcView = new MenuItem("View Direction");
direcView.setFont(fPlain);
viewMenu.add(direcView);
magView = new MenuItem("Magnification");
magView.setFont(fPlain);
viewMenu.add(magView);
perspecView = new MenuItem("Perspective View");
perspecView.setFont(fPlain);
viewMenu.add(perspecView);
grafMenu.add(viewMenu);
symbolMenu = new MenuItem("Symbols");
symbolMenu.setFont(fPlain);
grafMenu.add(symbolMenu);
styleMenu = new Menu("Style");
styleMenu.setFont(fPlain);
hiddenStyle = new MenuItem("Hidden Line");
hiddenStyle.setFont(fPlain);
styleMenu.add(hiddenStyle);
contourStyle = new MenuItem("Contour");
contourStyle.setFont(fPlain);
styleMenu.add(contourStyle);
lighStyle = new MenuItem("Light Source");
lighStyle.setFont(fPlain);
styleMenu.add(lightStyle);
grafMenu.add(styleMenu);
scaleMenu = new Menu("Scale");
scaleMenu.setFont(fPlain);
dispScale = new MenuItem("Displacement");
dispScale.setFont(fPlain);
scaleMenu.add(dispScale);
grafMenu.add(scaleMenu);
mainMenuBar.add(grafMenu);
selectMenu = new Menu("Select");
selectMenu.setFont(fBold);
selectNode = new CheckboxMenuItem("Nodes");
selectNode.setFont(fPlain);
selectMenu.add(selectNode);
selectElement = new CheckboxMenuItem("Elements");
selectElement.setFont(fPlain);
selectMenu.add(selectElement);
mainMenuBar.add(selectMenu);
plotMenu = new Menu("Plot");
plotMenu.setFont(fBold);
plotRef = new MenuItem("Refresh");
plotRef.setFont(fPlain);
plotMenu.add(plotRef);
plotNodes = new MenuItem("Nodes");
plotNodes.setFont(fPlain);
plotMenu.add(plotNodes);
plotElements = new MenuItem("Elements");
plotElements.setFont(fPlain);
plotMenu.add(plotElements);
mainMenuBar.add(plotMenu);
preeMenu = new Menu("Preprocessor");
preeMenu.setFont(fBold);
preeMaterial = new MenuItem("Material");
preeMaterial.setFont(fPlain);
preeMenu.add(preeMaterial);
applyBC = new MenuItem("Apply Constraints");
applyBC.setFont(fPlain);
preeMenu.add(applyBC);
deleteBCMenu = new Menu("Delete Constraints");
deleteBCMenu.setFont(fPlain);
deleteDisp = new MenuItem("Displacements");
deleteDisp.setFont(fPlain);
deleteBCMenu.add(deleteDisp);
deleteTrac = new MenuItem("Tractions");
deleteTrac.setFont(fPlain);
deleteBCMenu.add(deleteTrac);
preeMenu.add(deleteBCMenu);
preeQuery = new MenuItem("Query");
preeQuery.setFont(fPlain);
preeMenu.add(preeQuery);
mainMenuBar.add(preeMenu);
solveMenu = new Menu("Solution");
solveMenu.setFont(fBold);
solveSave = new MenuItem("Save Current Model");
solveSave.setFont(fPlain);
solveMenu.add(solveSave);
solveStart = new MenuItem("Start Solver");
solveStart.setFont(fPlain);
solveMenu.add(solveStart);
saveResult = new MenuItem("Save Result File");
saveResult.setFont(fPlain);
solveMenu.add(saveResult);
mainMenuBar.add(solveMenu);
postMenu = new Menu("Postprocessor");
postMenu.setFont(fBold);
listPost = new Menu("List");
listPost.setFont(fPlain);
listDisp = new MenuItem("Displacement");
listDisp.setFont(fPlain);
listPost.add(listDisp);
listStress = new MenuItem("Stress");
listStress.setFont(fPlain);
listPost.add(listStress);
postMenu.add(listPost);
plotPost = new Menu("Plot");
plotPost.setFont(fPlain);
plotDisp = new MenuItem("Displacement");
plotDisp.setFont(fPlain);
plotPost.add(plotDisp);
plotStress = new MenuItem("Stress");
plotStress.setFont(fPlain);
plotPost.add(plotStress);
postMenu.add(plotPost);
mainMenuBar.add(postMenu);
helpMenu = new Menu("Help");
helpMenu.setFont(fBold);

Examples = new Menu("Examples");
Examples.setFont(fPlain);
helpMenu.add(Examples);

helpAbout = new MenuItem("About");
helpAbout.setFont(fPlain);
helpMenu.add(helpAbout);

CrackFile = new MenuItem("Crack_In_Plate");
CrackFile.setFont(fPlain);
Examples.add(CrackFile);

HoleInPlate = new MenuItem("Two_Domain_Plate");
HoleInPlate.setFont(fPlain);
Examples.add(HoleInPlate);

TeeFile = new MenuItem("Tee_Shape_Plate");
TeeFile.setFont(fPlain);
Examples.add(TeeFile);

SquarePlateFile = new MenuItem("Square_Plate");
SquarePlateFile.setFont(fPlain);
Examples.add(SquarePlateFile);

mainMenuBar.add(helpMenu);

//resize the frame to the screen width
screenHeight = getToolkit().getScreenSize().height;
screenWidth = getToolkit().getScreenSize().width;
mnuFrameHeight = (int)(screenHeight * 0.15);

/** Add Toolbar **/
toolBar = new JToolBar();
toolBar.setFloatable(false);
toolBar.setBounds(getInsets().left, getInsets().top + mnuFrameHeight - 5, screenWidth, 25);
openFileButton = makeButton("open.jpg","open");
toolBar.add(openFileButton);
saveButton = makeButton("save.jpg","save");
toolBar.add(saveButton);
printButton = makeButton("print.jpg","print");
toolBar.add(printButton);
toolBar.addSeparator();
pzrButton = makeButton("pzr.gif","pan/zoom/rotate");
toolBar.add(pzrButton);
toolBar.addSeparator();
symbolButton = makeButton("hand.gif","node/element numbers");
toolBar.add(symbolButton);

toolBar.putClientProperty("JToolBar.isRollover", Boolean.FALSE);
add(toolBar, BorderLayout.NORTH);

pack();

//open the model window
modelDialog = new DialogModel(this,false);
modelDialog.setBounds(0,70,screenWidth-135,screenHeight-100);

//instantiate the pan/zoom/rotate window and boundary condition window
dialogPZR = new DialogPZR(this,false);
dialogBCondition = new DialogBC(this,false);
dialogDeleteDisp = new DialogDeleteBC(this,false);
dialogDeleteTrac = new DialogDeleteTraction(this,false);

//instantiate selection mode dialog box
selectMode = new DialogSelectionMode(this,false);
setBounds(0,0,screenWidth,screenHeight-25);
setVisible(true);
modelDialog.setVisible(true);

//Register Listeners
SymWindow aSymWindow = new SymWindow();
SymItem aSymItem = new SymItem();
this.addWindowListener(aSymWindow);
SymAction lSymAction = new SymAction();
fileOpen.addActionListener(lSymAction);
saveButton.addActionListener(lSymAction);
solveSave.addActionListener(lSymAction);
fileSave.addActionListener(lSymAction);
helpAbout.addActionListener(lSymAction);
fileExit.addActionListener(lSymAction);
listNodes.addActionListener(lSymAction);
listElements.addActionListener(lSymAction);
grafPZR.addActionListener(lSymAction);
pzrButton.addActionListener(lSymAction);
applyBC.addActionListener(lSymAction);
deleteDisp.addActionListener(lSymAction);
deleteTrac.addActionListener(lSymAction);
solveSave.addActionListener(lSymAction);
solveStart.addActionListener(lSymAction);
saveResult.addActionListener(lSymAction);
listDisp.addActionListener(lSymAction);
listStress.addActionListener(lSymAction);
plotDisp.addActionListener(lSymAction);
plotStress.addActionListener(lSymAction);
openFileButton.addActionListener(lSymAction);
symbolButton.addActionListener(lSymAction);
symbolMenu.addActionListener(lSymAction);
selectNode.addItemListener(aSymItem);
selectElement.addItemListener(aSymItem);
plotRef.addActionListener(lSymAction);
CrackFile.addActionListener(lSymAction);
HoleInPlate.addActionListener(lSymAction);
TeeFile.addActionListener(lSymAction);
SquarePlateFile.addActionListener(lSymAction);
}

public JButton makeButton(String image, String name) {
    // Changes for applet security consideration
    String ButtonFilename = "icons/" + image;
    URL ButtonURL = getURL(ButtonFilename);
    ImageIcon ButtonIcon = new ImageIcon(ButtonURL);
    JButton b = new JButton(ButtonIcon);
    b.setToolTipText(name);
    b.setMargin(new Insets(0,0,0,0));
    b.getAccessibleContext().setAccessibleName(name);
    return b;
}

public static URL getURL(String filename) {
    URL url = null;
    Applet thisApplet = ac.getApplet("UCWaves");
    try {
        URL codeBase = thisApplet.getCodeBase();
        // URL codeBase = new URL("file:c:/temp/work");
        url = new URL(codeBase, filename);
    } catch (java.net.MalformedURLException e) {
        System.err.println("Couldn't create image: "+
            "badly specified URL");
        return null;
    }
}
public FrameMain(String title) {
    this(ac);
    setTitle(title);
}
/* public void setVisible(boolean b) {
    setLocation(0, 0);
    super.setVisible(b);
} */

/*******************Window adapter class implementation*******************/
class SymWindow extends java.awt.event.WindowAdapter {
    public void windowClosing(WindowEvent event) {
        Object object = event.getSource();
        if (object == FrameMain.this)
            FrameMain_WindowClosing(event);
    }
}

void FrameMain_WindowClosing(WindowEvent event) {
    setVisible(false);     // hide the Frame
    dispose();
    // free the system resources
    if (FrameMain.this.ac == null)
        System.exit(0);     // close the application
}

/********************Action Listener Interface Implementation Using Inner Classes******/
class SymAction implements java.awt.event.ActionListener {
    public void actionPerformed(ActionEvent event) {
        Object object = event.getSource();
        if (object == fileOpen || object == openFileButton)
            try{
                open_Action(event);
                flagForpaint = false;
            }catch(IOException e){;}
        else if (object == CrackFile || object == HoleInPlate || object == TeeFile || object == SquarePlateFile)
            try{
                System.out.println("i heard hole in plate");
                Test_File_open_Action(event);
                flagForpaint = false;
            }catch(IOException e){;}
        else if(object == fileSave || object == saveButton )
            try{
                save_Action(event);}
            catch(IOException e){;}
        else if (object == helpAbout)
            about_Action(event);
        return url;
    }
}
else if (object == fileExit)
    exit_Action(event);
else if (object == listNodes)
    listNodes_Action(event);
else if (object == listElements)
    listElements_Action(event);
else if (object == grafPZR || object == pzrButton)
    grafPZR_Action(event);
else if (object == applyBC)
    applyBC_Action(event);
else if (object == listDisp)
    listDisp_Action(event);
else if (object == listStress)
    listStress_Action(event);
else if (object == grafPZR || object == pzrButton)
    listDisp_Action(event);
else if (object == plotDisp)
    try {
        plotDisp_Action(event);
        flagForPaint = true;
        } catch (IOException e) {} 
else if (object == plotStress)
    plotStress_Action(event);
else if (object == deleteDisp)
    deleteDisp_Action(event);
else if (object == deleteTrac)
    deleteTrac_Action(event);
else if (object == symbolMenu || object == symbolButton)
    symbolMenu_Action(event);
else if (object == solveSave)
    solveS_Action(event);
else if (object == solveStart)
    solve_Action(event);
else if (object == plotRef)
    try {
        plotRef_Action(event);
        } catch (IOException e) {} 
else if (object == saveResult)
    try {
        save_Result(event);
        } catch (IOException e) {} 
}

void plotRef_Action(ActionEvent event) throws IOException{

//*********************************************
public void readDisp1() throws IOException{
    modelDialog.setFilename1();
    if (modelDialog.getMeshRef() != null) {
        modelDialog.Scale();
        modelDialog.Sample();
        (modelDialog.getMeshRef()).test1();
    }
    //modelDialog.setFilename("temp.dat");
public void readTraction()
{
//this needs to be worked on to show the stress results
}

void listNodes_Action(ActionEvent event) {
    lstEntity = new ListFileDialog(this,true);
    lstEntity.setModelReference(modelDialog);
    lstEntity.list(ListFileDialog.listTheNodes);
    lstEntity.setVisible(true);
}

void listElements_Action(ActionEvent event) {
    lstEntity = new ListFileDialog(this,true);
    lstEntity.setModelReference(modelDialog);
    lstEntity.list(ListFileDialog.listTheElements);
    lstEntity.setVisible(true);
}

void listDisp_Action(ActionEvent event) {
    lstEntity = new ListFileDialog(this,true);
    lstEntity.setModelReference(modelDialog);
    lstEntity.listTheDis();
    lstEntity.setVisible(true);
}

void listStress_Action(ActionEvent event) {
    lstEntity = new ListFileDialog(this,true);
    lstEntity.setModelReference(modelDialog);
    lstEntity.listTheStres();
    lstEntity.setVisible(true);
}

//**************************************************************************

void plotDisp_Action(ActionEvent event) throws IOException {
    System.out.println("into the plotDisp procedure");
    flagForpaint = true;
    URL tempURL = null;
    if ( modelDialog.getMeshRef() != null ) {
        modelDialog.setFilename1();
        modelDialog.Scale();
        modelDialog.Sample();
        (modelDialog.getMeshRef()).test1();
        try{
            File Filetemp = new File("temp.dat");
            tempURL = Filetemp.toURL();
        }
        catch(MalformedURLException e) { }
        modelDialog.plotFilename(tempURL);
    }
void plotStress_Action(ActionEvent event) {
    System.out.println("into the plotStress procedure");
}

//*******************************************************************
void grafPZR_Action(ActionEvent event) {
    dialogPZR.setVisible(true);
    dialogPZR.setModelReference(modelDialog);
}

void open_Action(ActionEvent event) throws IOException {
    String fileForSolve;
    String IntermediateFile = "/intermediate/currentProblem.res";
    File currentplot = new File(IntermediateFile);
    //Set the directory for the dialog box
    //openMyFileDialog.setDirectory("datFiles");
    openMyFileDialog.setDirectory("c:\temp");
    openMyFileDialog.setFile("**");
    openMyFileDialog.show();
    //get the selected file name
    fileName = openMyFileDialog.getFile();
    //set this fileName as title for dialog
    modelDialog.setTitle(fileName);
    String dirName = openMyFileDialog.getDirectory();
    fileName = dirName + fileName;
    URL fileContentURL = null;
    try {
        File inputFile = new File(fileName);
        fileContentURL = inputFile.toURL();
    } catch (MalformedURLException e) {
    }
    System.out.println("the file passed to mesh11 by ucwaves "+ fileName);
    //File CurrentFile = new File(fileName);
    //String absfileName = CurrentFile.getAbsolutePath();
    fileForSolve = ".\datFiles\"+fileName;
    //Read the mesh geometry from the file
    if (fileName != null)
        //currentplot.delete();
        modelDialog.setFilename(fileContentURL);
        //CopyFile.doCopy(fileForSolve, ".\solver\Input.dat");
        //applet
        //modelDialog.setFilename("input.dat");
void Test_File_open_Action(ActionEvent event) throws IOException {
    URL fileContentURL = null;
    String fileName = null;

    String ModelFileName = event.getActionCommand();
    System.out.println("the command by test file " + ModelFileName);
    if (ModelFileName.equals("Crack_In_Plate")) {
        fileName = "Crack_In_Plate.dat";
    } else if (ModelFileName.equals("Two_Domain_Plate")) {
        fileName = "Two_Domain_Plate.dat";
    } else if (ModelFileName.equals("Tee_Shape_Plate")) {
        fileName = "Tee_Shape_Plate.dat";
    } else if (ModelFileName.equals("Square_Plate")) {
        fileName = "Square_Plate.dat";
    }
    fileName = "datFiles/" + fileName;
    fileContentURL = getURL(fileName);

    //set this fileName as title for dialog
    modelDialog.setTitle(fileName);
    System.out.println("the file passed to mesh by ucwaves " + fileName);
    String fileForSolve = ".\datFiles\" + fileName;

    //Read the mesh geometry from the file
    if (fileName != null)
        //currentplot.delete();
        modelDialog.setFilename(fileContentURL);
        //CopyFile.doCopy(fileForSolve, ".\solver\Input.dat");
        //aplet
        //modelDialog.setFilename("input.dat");
}

void save_Result(ActionEvent event) throws IOException {
    saveMyFileDialog.setDirectory("datFiles");
    saveMyFileDialog.setFile("*.res");
    saveMyFileDialog.show();
    String fileName1 = saveMyFileDialog.getFile();
    if (fileName1 != null)
        try {
            
        
    }
File f1=new File("intermediate","currentProblem.res");
FileInputStream fis=new FileInputStream(f1);
BufferedReader is=new BufferedReader(new FileReader(fis.getFD()));
String s=is.readLine();
File f2=new File("datFiles",fileName1);
FileOutputStream fos=new FileOutputStream(f2);
PrintWriter os=new PrintWriter(fos);
while(s!=null)
{
    StringTokenizer t=new StringTokenizer(s,"\n");
    while(t.hasMoreTokens())
    {
        os.write(t.nextToken());
    }
    os.println();
    s=is.readLine();
}
is.close();
os.close();
}
catch(Exception ec){System.out.println("the exception is over here" + ec);}
}

void save_Action(ActionEvent event)throws IOException{
    saveMyFileDialog1.setDirectory("datFiles");
    saveMyFileDialog1.setFile("*.dat");
    saveMyFileDialog1.show();
    String fileName2=saveMyFileDialog1.getFile();
    if(fileName2!=null)
    {
        try
        {
            File f1=new File("intermediate","currentProblem.dat");
            FileInputStream fis=new FileInputStream(f1);
            BufferedReader is=new BufferedReader(new FileReader(fis.getFD()));
            String s=is.readLine();
            File f2=new File("datFiles",fileName2);
            FileOutputStream fos=new FileOutputStream(f2);
            PrintWriter os=new PrintWriter(fos);
            while(s!=null)
            {
                StringTokenizer t=new StringTokenizer(s,"\n");
                while(t.hasMoreTokens())
                {
                    os.write(t.nextToken());
                }
                os.println();
                s=is.readLine();
            }
            is.close();
            os.close();
        }catch(Exception ec){System.out.println("the exception is over here" + ec);}
    }
}

void save_Action(ActionEvent event)throws IOException{
    try
    {
        File f1=new File("intermediate","currentProblem.dat");
        FileInputStream fis=new FileInputStream(f1);
        BufferedReader is=new BufferedReader(new FileReader(fis.getFD()));
        String s=is.readLine();
        File f2=new File("datFiles",fileName2);
        FileOutputStream fos=new FileOutputStream(f2);
        PrintWriter os=new PrintWriter(fos);
        while(s!=null)
        {
            StringTokenizer t=new StringTokenizer(s,"\n");
            while(t.hasMoreTokens())
            {
                os.write(t.nextToken());
            }
            os.println();
            s=is.readLine();
        }
        is.close();
        os.close();
    }catch(Exception ec){}
}
void applyBC_Action(ActionEvent event) {
    dialogBCondition.setVisible(true);
    dialogBCondition.setModelReference(modelDialog);
}

void deleteDisp_Action(ActionEvent event) {
    if (modelDialog.getMeshRef() != null) {
        if (!dialogDeleteDisp.isShowing()) {
            dialogDeleteDisp.setModelReference(modelDialog);
            dialogDeleteDisp.setVisible(true);
        }
    }
}

void deleteTrac_Action(ActionEvent event) {
    if (modelDialog.getMeshRef() != null) {
        if (!dialogDeleteTrac.isShowing()) {
            dialogDeleteTrac.setModelReference(modelDialog);
            dialogDeleteTrac.setVisible(true);
        }
    }
}

void symbolMenu_Action(ActionEvent event) {
    ShowDialog setting = new ShowDialog(this,true);
    /* Show the current settings */
    setting.setSettings(modelDialog.getDisplaySettings());
    setting.setVisible(true);
    if (setting.isChanged())
        modelDialog.repaint();
    setting.dispose();
}

void selectNode_Action() {
    selectElement.setState(false);
    if (selectNode.getState()) {
        modelDialog.getDisplaySettings().setSelectionMode(1);
        modelDialog.unselect(2);
        if (!selectMode.isShowing()) {
            selectMode.setDisplaySettings(modelDialog.getDisplaySettings());
            selectMode.setVisible(true);
        }
    } else {
        modelDialog.getDisplaySettings().setSelectionMode(0);
        modelDialog.unselect(1);
        selectMode.setVisible(false);
    }
}

void selectElement_Action() {
    selectNode.setState(false);
}
if (selectElement.getState()) {
    modelDialog.getDisplaySettings().setSelectionMode(2);
    modelDialog.unselect(1);
    if (!selectMode.isShowing()) {
        selectMode.setDisplaySettings(modelDialog.getDisplaySettings());
        selectMode.setVisible(true);
    }
} else {
    modelDialog.getDisplaySettings().setSelectionMode(0);
    modelDialog.unselect(2);
    selectMode.setVisible(false);
}

void solveS_Action(ActionEvent event) {
    if (modelDialog.getMeshRef() != null) {
        try {
            (modelDialog.getMeshRef()).test();
        } catch (Exception e) {}}
}

void solve_Action(ActionEvent event) {
    System.out.println("in the begining of the solve action");
    if (modelDialog.getMeshRef() != null) {
        try {
            (modelDialog.getMeshRef()).solve();
        } catch (Exception e) {}}
}

void about_Action(ActionEvent event) {
    // Action from About Create and show as modal
    (new AboutDialog(this, true)).show();
}

void exit_Action(ActionEvent event) {
    dispose();
    if (FrameMain.this.ac == null)
        System.exit(0);
}

/*******************Item Listener Implementation*******************/
class SymItem implements java.awt.event.ItemListener {
    public void itemStateChanged(ItemEvent e) {
        Object object = e.getSource();
        if (object == selectNode)
            selectNode_Action();
        else if (object == selectElement)
            selectElement_Action();
    }
}

public static boolean getflagForpaint(){
return flagForpaint;
}

/***************************************************************************/
*/
References


