JAVA APPLETS FOR TEACHING FINITE ELEMENT ANALYSIS

Kamal B. Rojiani[1], Suryanarayana Raju S.V.N. [2]

ABSTRACT

An applet for teaching fundamental concepts of finite element analysis is developed. An applet is a web-based computer program written in the object-oriented Java programming language. Applets are well-suited for delivering dynamic content over the Internet since they are platform and operating system independent and are available to students and instructors throughout the world. The applet developed incorporates a wide range of elements, including: bar elements, truss beam and frame elements, triangular and quadrilateral plane stress and plane strain elements, four node and eight node isoparametric elements, and triangular and quadrilateral plate elements. The applet is interactive and makes extensive use of graphics. The associated web pages provide information on the fundamental concepts of finite element analysis, formulation of the various elements, instructions for using the applet, and example problems.

The applet provides a novel approach for teaching basic finite element analysis concepts. It gives students a means of checking their work and reinforces fundamental concepts learned in class. It enhances their learning experience by allowing them to experiment with building and analyzing complex structures, and visualizing the results as changes are made to the finite element model. The applet can be used as supplementary material, complementing classroom and textbook instruction.

KEY WORDS

Finite element analysis, Java applet, web based education.

INTRODUCTION

The finite element method is the most widely used tool for computer-based numerical solution of a wide range of engineering problems. The finite element method is an approximate method where a large number of simultaneous equations are solved and a considerable amount of computational effort is required. With the advances in computing hardware and programming methodologies the finite element method has become a popular method for analyzing structures. This method is used for solving problems in a variety of application areas, such as structural analysis, heat transfer, fluid mechanics, vibrations, seepage, and electric and magnetic fields. With the application of this method, problems that were previously intractable are now solved routinely.

[1] Associate Professor, Charles E. Via Jr. Dept. of Civil and Environmental Engineering, Virginia Tech, Blacksburg, VA, 24061, Phone +1 540/231-7150, FAX 540/231-7532, krojiani@vt.edu.

[2] Graduate Teaching Assistant, Charles E. Via Jr. Dept. of Civil and Environmental Engineering, Virginia Tech, Blacksburg, VA, 24061, sagi@vt.edu

Finite element analysis is generally taught at the graduate level at most universities due to the inherent complexities associated with explaining concepts. However, due to the rapid growth and widespread use of this method, there is now a growing need to teach it at the undergraduate level. Given the importance of finite element analysis, it is essential that engineering students have a thorough understanding of the fundamental concepts of finite element analysis and the computational and programming procedures used in the implementation of this method.

JAVA APPLETS

Java applets are an excellent means for delivering dynamic interactive graphical content over the Internet. An applet is a computer program written in the object-oriented Java programming language and embedded in web-pages. Java applets are platform and operating system independent and can be accessed by students throughout the world. All that is needed is a Java compliant web browser.

Java applets overcome many, if not all, of the limitations of other web-based technologies and have significant advantages. Java is a powerful programming language and can be used to develop complex graphical applications. The advantages of using Java for web-based learning have been discussed by many researchers (e.g., Range and Gramoll 2000; Rojiani et al. 2000). The object-oriented nature of the Java language makes it very easy to develop, maintain and modify applets. Also, since the files and documents that comprise an applet are stored on the server rather than on the client machine, it overcomes many of the difficulties related to the delivery and installation of software. Another significant advantage of Java is that it has extensive security features for protecting client computers.

APPLET DESCRIPTION

The work presented here utilizes Java to create a web-based learning resource for teaching finite element analysis. A set of web-based instructional units to interactively and dynamically illustrate the fundamental concepts of finite element analysis are developed. These instructional units are centered on a Java applet. The applet incorporates a wide range of finite elements, such as bar, truss, beam and frame elements; triangular and quadrilateral plane stress and plane strain elements; four-node and eight-node isoparametric elements; and plate elements. It has features for entering the finite element model of the structure such as joints and joint coordinates, restraints, elements and connectivity information, material and section properties, and loading. The applet displays the finite element model on the screen as it is entered. The interactive nature of the applet provides students with instant feedback which helps to reinforce their understanding of the material.

ELEMENTS

A variety of basic elements are incorporated in the applet. The elements are typical of those studied in a beginning finite element analysis course.

One-Dimensional Bar and Truss Elements: Bar and truss elements are the simplest elements. They are useful for explaining the fundamentals of the finite element method. These elements are used to analyze structures subjected to axial loads (i.e., no bending or shear). The longitudinal axis of the bar element is parallel to the corresponding axis of the

structure, whereas the axis of the truss element can have any orientation in the plane of the structure. For truss elements, transformation matrices are required to convert forces and displacements in the local coordinate system to the global coordinate system. Thus, the truss element provides a good introduction to the concept of transformation between local and global coordinate systems.

Two-Dimensional Beam and Frame Elements: The two-dimensional beam element has two degrees of freedom at each node: a rotation about an axis perpendicular to the plane of beam and a translation perpendicular to the axis of the beam. Axial deformations in the beam are neglected. The frame element has axial deformation at each node in addition to the beam deformations. Since there is no coupling between the axial and flexural deformations, the stiffness matrix of the beam is first derived and it is then extended to the frame element by incorporating the axial stiffness. External loads on beams may be applied on the members as well on the nodes. Member loads are converted into equivalent nodal loads.

Membrane Elements: Three types of membrane elements are developed: (1) constant strain triangle, (2) four-node isoparametric quadrilateral element, and (3) eight-node isoparametric quadrilateral element. The constant strain triangle is the simplest element for analyzing two-dimensional problems. It has three nodes, and each node has two inplane degrees of freedom, for a total of six degrees of freedom. Triangular elements are commonly used for structures with irregular boundary conditions.

The four node isoparametric quadrilateral element has two degrees of freedom (translation in the x and y direction) at each node. A 2x2 Gauss quadrature rule is used to obtain the stiffness matrix of the element. This element is useful for explaining the basic features of isoparametric elements. It is also useful for explaining the application of Gauss quadrature for determining the stiffness matrix.

The eight-node isoparametric quadrilateral element is an extension of the four-node element. Four additional nodes are considered along the mid-points of the element boundaries. The formulation of the stiffness matrix for this element is similar to that of the four-node quadrilateral element. Also, numerical integration and the calculation of the load vector follow the same approach as before.

Plate Elements: Two types of plate bending elements are implemented: (1) the Discrete Kirchhoff Triangle (DKT) element developed by Batoz et al. 1980, and (2) the Discrete Kirchhoff Quadrilateral (DKQ) element developed by Batoz and Tahar 1982. Both elements have three degrees of freedom at each node, two rotations (θx and θy), and a transverse displacement (w).

PROGRAM STRUCTURE

The applet is written in Java using the object-oriented approach. The classes developed can be divided into three categories: (1) structural classes for representing the various elements of the structure, (2) input and output classes for entering the finite element model and displaying analysis results, and (3) interface classes for interacting with the user. A brief description of these classes is presented below.

STRUCTURAL CLASSES

The structural classes include classes for representing the structural model and for analyzing the structure. A finite element model is created from various components, such as nodes, elements, material properties, restraints, element section properties, nodal loads and member loads. Separate classes for representing these components were developed. The structural classes contained in the applet are shown in Figure 1. Examples of these classes are `Joint`, `Material`, `SectProp` and `SystemProp` which represent joints, material properties, section properties and system properties. The classes `JointLoad` and `UniformLoad` represent joint loads and uniformly distributed loads.

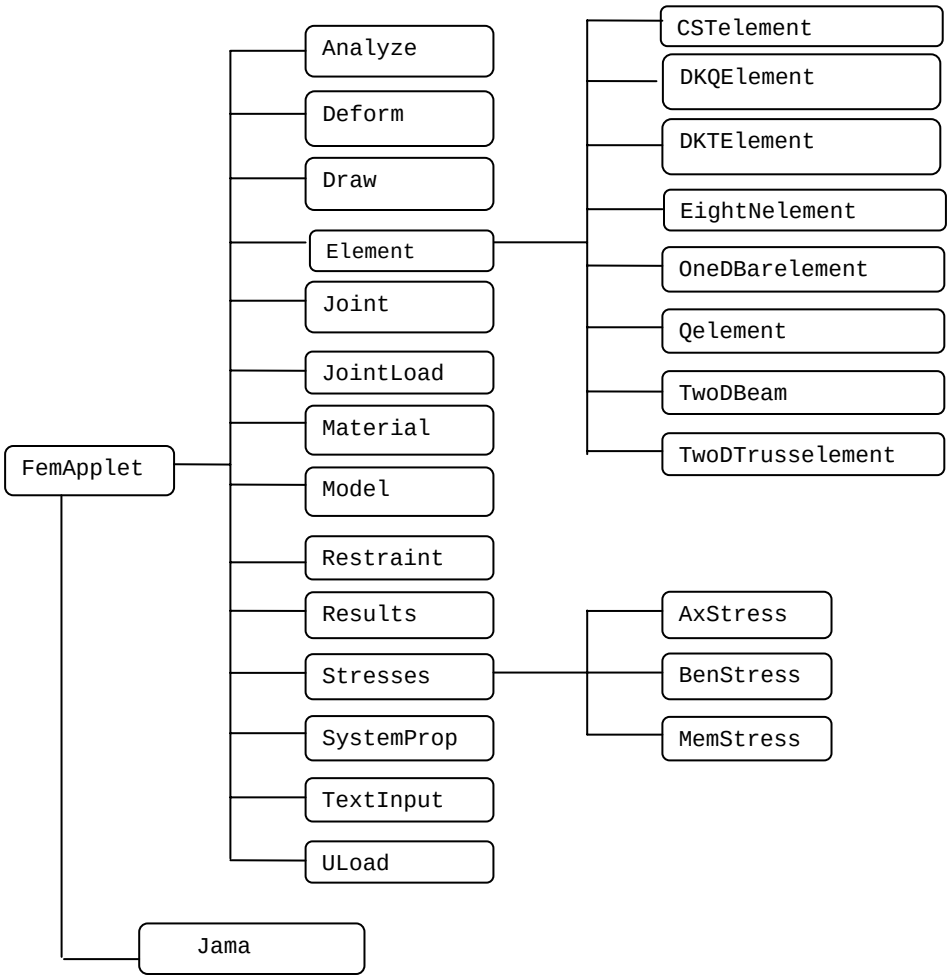

Figure 1: Structural Classes in the Applet.

`FemApplet` is the main class of the applet. This class has methods for reading the input data. The graphical user interface of the applet is mainly provided by the `FemApplet` class. The `FemApplet` class is created by extending the `JApplet` class of the Java class library.

This class contains the required code for controlling the behavior of components contained in main applet window. Various components such as `JTextBox`, `JLabel`, `JList`, `JScrollPane`, `JTextArea`, `JButton`, `JCheckBox` and `JRadioButton` from the Java class library are used for developing the user interface of the applet. Also, several inner classes are provided for event handling operations of the above mentioned components.

The `Model` class is a container class where all components of the finite element model are stored. Java provides a library of collection classes that dynamically allocate space for each object stored. The `Model` class uses the Java API `Vector` class to store components of the finite element model. A `Vector` is a dynamic array that automatically allocates memory for objects stored in this array. A `Vector` has a unique property which allows it to dynamically add or remove objects. This dynamic resizing property of `Vector` makes it possible to create storage containers for objects without limitations on the number of objects stored. Thus, structures with a large number of nodes and elements can be analyzed by storing components of the finite element model as `Vectors`. The only limitation on the size of the structures that can be analyzed is the amount of available memory. The `Model` class creates various instances of the `Vector` class. `Vectors` such as, `elementdata`, `nodedata`, `jointloaddata`, `restraintdata`, `materialprop` and `systemdata` store objects representing components of the finite element model. Methods for manipulating the data stored in `Vectors` are also defined in the `Model` class.

The `Element` class is the base class for all of the element classes. The data members of this class include element number, number of nodes and a nodal connectivity array. The `Element` class also contains methods for determining the type of element and the nodes to which the element is connected. Classes for representing the various elements such as `OneDBarelement`, `TwoDTrusselement`, `TwoDBeam`, and `CSTelement` are derived from the `Element` class by inheritance.

Data from the `Model` class is used by the `Analyze` class to perform the analysis of the structure. The required data for analysis is processed by the `analysisdata()` method. Joint connectivity data is obtained for each element from the `Element` class. Nodal coordinates for each element are obtained by calling methods in the **`Joint`** class. Instances of the element classes are declared in the `stiffnessmatrix()` method. Element stiffness matrices are then computed for each element. Methods in the `Analyze` class assemble the structure stiffness matrix, solve the system of equations and compute displacements and stresses. The Jama matrix package developed by National Institute of Standards and Technology (NIST 2006) is used for matrix operations. Element stresses are stored in instances of the `AxStress`, `BenStress` and `MemStress` classes. These classes represent axial, bending, and membrane stresses, and are inherited from the `Stresses` container class.

INPUT AND OUTPUT CLASSES

The required input for the finite element model is specified by entering data in the various input fields provided in the graphical user interface of the program or via a text file. With the text input option the finite element model is generated using data provided in a format that is

similar to the format used by the SAP 2000 (SAP2000b 1997) commercial finite element program. The `TextInput` class reads the input and creates a model of the structure.
The `Result` class creates the Results frame and displays the analysis results. It contains only one method named `AddText()` which adds the displacements and different types of stresses obtained at each node to the `JTextArea` component of the Results frame. The analysis results are tabulated in an orderly fashion before they are displayed. The format of the output is similar to that of SAP 2000. The `Draw` and `Deform` classes plot the structure and the deformed shape of the structure.

INTERFACE CLASSES

The graphical user interface of the applet is provided by the `FemApplet` class. The `Draw` class displays the structure. This class contains methods for displaying the structure, concentrated loads, uniform loads and restraints. Each element is numbered along with the nodes as given in the input. The structure is redrawn as changes to the finite element model are made. The deformed shape of the structure is plotted by the `Deform` class. The deformed shape of the structure helps in visualizing the response of the structure to loading.

APPLET INTERFACE

The fundamental design decision in developing the user interface was to make it easy for users to enter model data, analyze the structure and view results with a minimal amount of effort. The applet developed accomplishes these objectives. All information is contained in one form and the user does not have to navigate between multiple forms. Only the analysis results are displayed on a separate form. The applet has a dynamic and interactive user interface and provides constant and instant feedback to the student. The input fields are arranged logically (in order of input) in a set of tabs which makes it very easy to navigate between the various elements of the model.

The main window of the applet which is derived from the Java `JApplet` class is shown in Figure 2. A menu bar is provided with the File, Model, Analysis and Help menus. The Model menu contains the Input and Draw commands. When Run is selected from the Analysis menu an instance of the `Analyze` class is created for performing the analysis of structure. The Results command instantiates the `Result` class which displays the results on the Results frame once the analysis is completed.

MANUAL INPUT

Two methods for entering data are provided: a) manual input and b) text input. The input option is activated by selecting the Input command from the Model menu. The element type, coordinates of nodes, connectivity information, restraints and loads acting at various nodes are entered in the input frame. The manual input frame consists of five tabs. Each input tab is discussed in detail in following paragraphs.

Element Type Tab: The type of element to be used for modeling the structure, possible degrees of freedom, and method of input (manual or text) are entered in this tab. For membrane elements, the plane stress or plane strain options are activated. The Element Type combo list displays a list of elements which the element type is be selected.

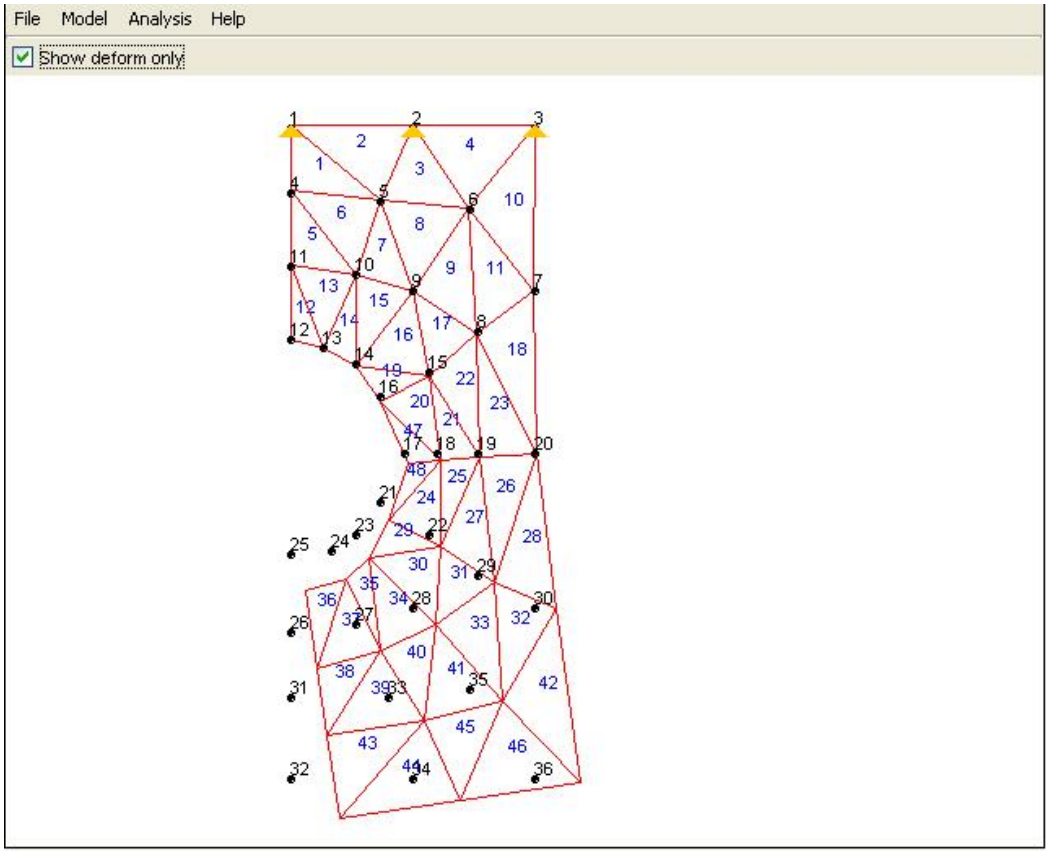

Figure 2: Applet interface showing finite element model and deformed shape.

Coordinates Tab: Nodes in the structure are defined by the node number, and X and Y coordinates. All coordinates are specified in global system. To add a node, the node number and the X and Y coordinates are entered in the text fields and the Add command button is pressed. To remove a node the corresponding node is selected from the list box and the Remove button is pressed. There are no limitations on the number of nodes that can be contained in a structure.

Connectivity Tab: Each element is connected to a set of nodes. The number of nodes in an element depends on the type of element. Connectivity information is required for calculating the stiffness matrix and stresses at each node. Material properties and section properties are also entered in this tab. The Add and Remove buttons function similarly to those mentioned in the Coordinates tab. Figure 3 illustrates the Connectivity tab.

Restraint Tab: The restraints are entered in this tab. The node number of the restraint along with the direction in which it is restrained is entered.

Load Tab: The functioning of *Load Tab* is similar to the Restraint Tab where, instead of restraints in the X and Y directions, loads acting in the X and Y direction are specified for nodal loads.

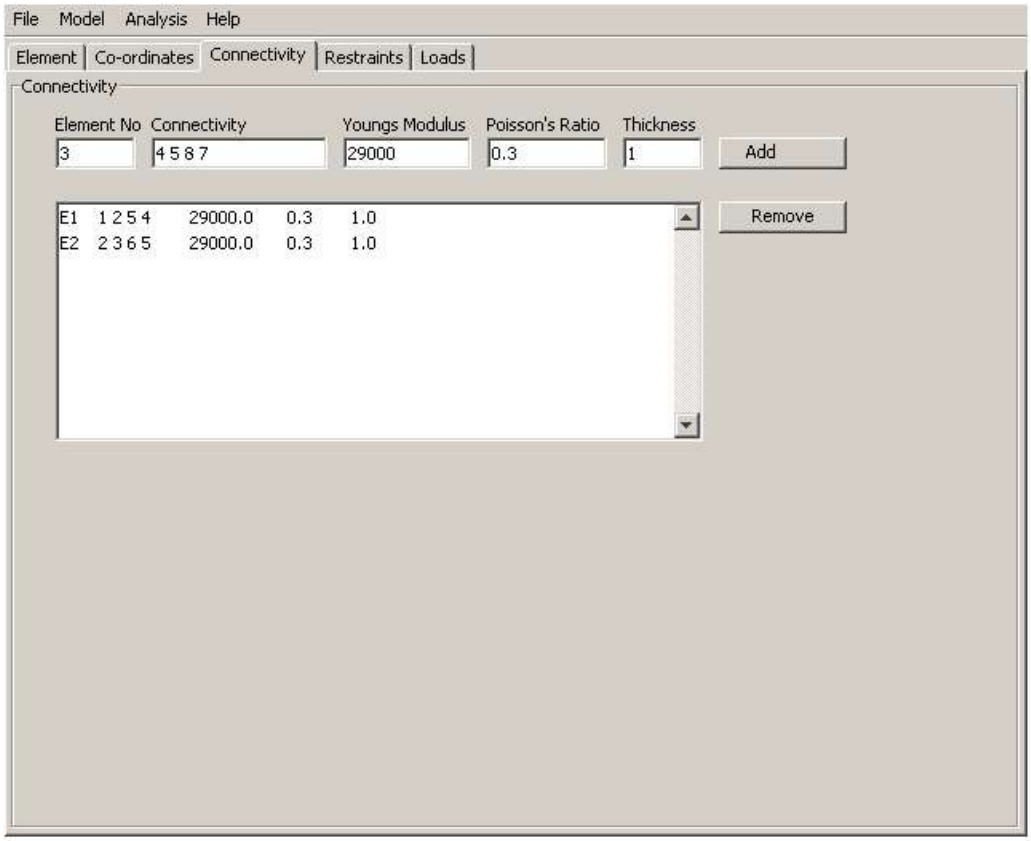

Figure 3: Connectivity tab.

TEXT INPUT

When a structure with a large number of nodes is analyzed it is more convenient to provide input in text format. Generally, an input text file is created and is read by the program. However a Java applet cannot read input from a file due to security reasons. To overcome this limitation the input file is copied to the clipboard and the contents of the clipboard are imported to a text input field in the applet. The text input field is shown in Figure 4. The information required for generating a finite element model is extracted from the text input field and stored as an instance of the `Model` class. The `TextInput` class has all the necessary code required for interpreting the input and storing it in the `Model` class. The format of the input is similar to that used by SAP 2000 (SAP 2000a).

VERIFICATION

To verify the accuracy of the program, results obtained from the applet are compared with those obtained from SAP 2000 (SAP-2000b 1997) a widely used commercial finite element analysis program. A series of test problems were analyzed using the both applet and SAP

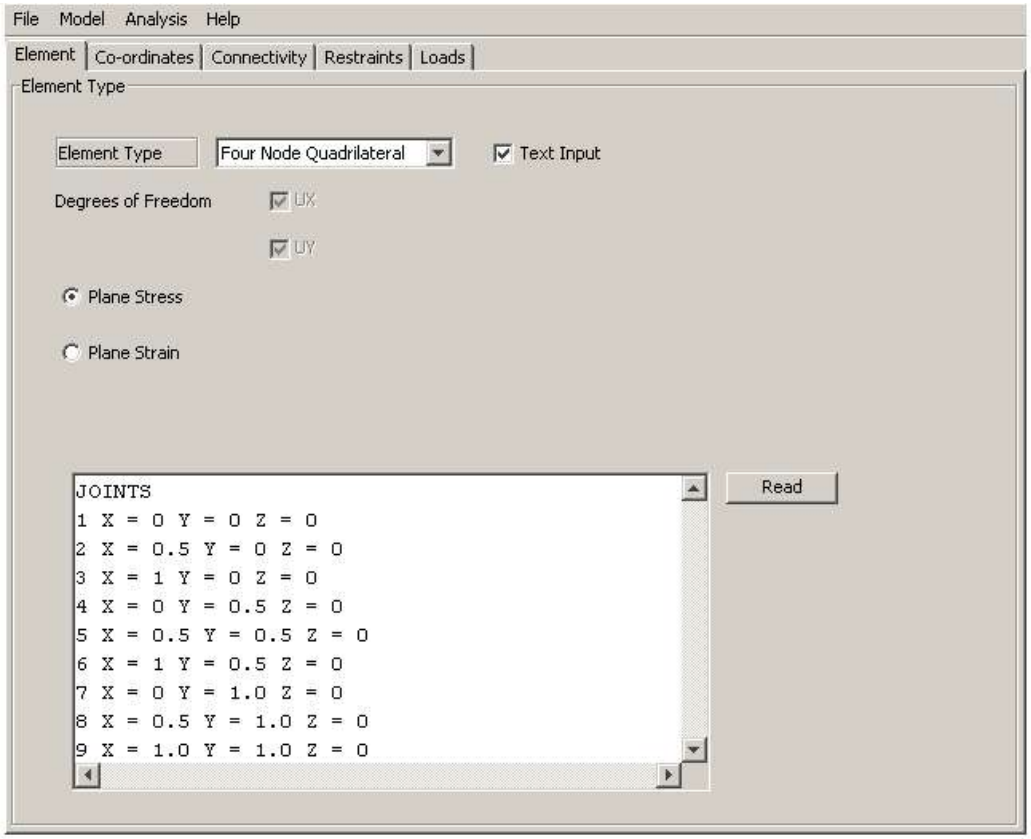

Figure 4: Text input field.

2000. The test problems included one-dimensional bar systems, two-dimensional trusses, beams and frames, several plane stress and plane strain problems, solid plates and plates with semi-circular and rectangular holes. Displacements and stresses at various points were compared. It was found that displacements computed by the applet were within 0.1 percent of those obtained from SAP 2000. Also, stresses obtained from the applet were typically within 3 percent of those obtained from SAP 2000. In a few cases the difference in stresses was approximately 5 percent. The reason for these differences is that SAP 2000 uses a stress averaging procedure for computing stress. The differences in stresses for the plate problems I due to the different plate element used in SAP 2000.

WEB SITE

An important aspect of the work was to develop a web site where the applet is hosted. The web site provides instruction on the fundamental concepts of finite element analysis. Detailed information regarding the formulation of each element is provided. This includes shape functions used, the derivation of the element stiffness matrix, and the process of converting member loads and pressures into nodal loads. Details on the application of Gauss quadrature for computing the element stiffness matrix, and for converting member loads in nodal loads are also presented.

An important feature of the web site is a description of the implementation of the applet in the Java programming language. Details of the various classes develop including the data members and methods in each class are provided. This information is useful for teaching the programming aspects of the finite element method. There is also information on the assembly process for obtaining the stiffness matrix of the structure and the solution of the system of equations. Since these tasks are accomplished in the `Analyze` class the important features of this class are also given. The web site also has step by step instructions for using the applets. To allow students to gain valuable experience with using the applet a series of example problems addressing different types of structures are provide along with answers. There is also a survey form for obtaining comments from students so that the applet may be modified and improved.

CONCLUSIONS

The applet and the associated web site provide a novel approach for teaching finite element analysis. The applet has a number of features that makes it very attractive in teaching. The graphical user interface combined with the various input forms make it easy to enter data. The ability to plot the finite element model provides interactive and immediate feedback and makes it easy to detect errors in the finite element model. It provides students with a means for checking their work, reinforces fundamental concepts learned in class, and enhances their learning experience. Students can experiment by building and analyzing complex structures and visualizing analysis results. It is easy to maintain and modify the applet since it is written in the object-oriented Java programming language. Since the applet is self-contained and is available over the web, it can easily be incorporated into existing courses. It is anticipated that the applet will be widely used primarily as supplementary material, reinforcing and complementing classroom and textbook instruction.

REFERENCES

Batoz J.-L., Bathe K. J. and Ho L. W. (1980). "A Study of Three-Noded Triangular Plate Bending Elements," *International Journal for Numerical Methods in Engineering*, Vol.15, 1771-1812.

Batoz J.-L. and Tahar M. B.(1982). "Evaluation of a New Quadrilateral Thin Plate Bending Element," *International Journal for Numerical Methods in Engineering*, Vol. 18, 1655-1677.

National Institute of Standards and Technology (NTIST) (2006). "JAMA: A Java Matrix Package." http://math.nist.gov/javanumerics/jama/. Last accessed: February 18, 2006.

Ranga, K. and Kurt Gramoll, K. (2000). "3-D Finite Element Analysis on the Internet using Java and VRML." *Proceedings, ASEE Annual Conference*, June, St. Louis, MO.

Rojiani, K. B., Kim, Y.Y., and Kapania, R.K. (2000). "Web-Based Java Applets for Teaching Engineering Mechanics." *Proceedings, ASEE Annual Conference*, June, St. Louis, MO.

SAP-2000a (1997). *Integrated Finite Element Analysis and Design of Structures, Input File Format*, Computers and Structures, Inc., Berkeley, California.

SAP-2000b (1997). *Integrated Finite Element Analysis and Design of Structures, Analysis Reference*, Computers and Structures, Inc., Berkeley, California.