Combined Research and Curriculum Development of Web and Java Based Educational Modules with Immersive Virtual Environments

Ronald D. Kriz[1], Randy T.Levensalor[2], and Sanjiv D.Parikh[1]
[1]Department of Engineering Science and Mechanics and [2]Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061 USA

Key words: Computer-simulation, Java, VRML, CAVE

Abstract: A Java framework is described for creating an interface with legacy code on a Web browser. This interface was created in the development of modules for teaching a senior level (I) and first-year graduate level course (II) on the Mechanical Behavior of Materials. Both courses incorporate the results of state-of-the-art simulation techniques. The modules make extensive use of materials available through the Internet. When appropriate, students study structure property relationships predicted by simulations in an immersive environment called a CAVE™. Simulation results span various length scales, starting at the atomistic level that use embedded atom method techniques, and continuing with simulations at the continuum level that use finite element method techniques. Modules for the second course focus on a scale between the atomistic and continuum level where mechanical behavior is predicted by simulations that used a variety of numerical techniques. These modules use legacy code written by the researchers teaching these classes. Considerable attention was given to creating a Web-based interface that allows researchers easily to construct, and students easily to use, interfaces that access legacy code in an interactive format. Hence, researchers and instructors can focus more on content development and students can focus more on experimenting with possible parametric combinations in their solutions.

™ CAVE is a trademark name of the Electronic Visualization Laboratory of the University of Illinois.

1. WEB-BASED JAVA FRAMEWORK

Although modules are still under development for the second course, sufficient progress has been made to report on the origin, current status, and lessons learned in using these modules in the first of the two courses.

1.1 Origin

Modules were developed and distributed on our SUN-VNI Wave-Java server [1]. Early efforts to create a distributed, Web-based, visual computing environment were funded by SUN Microsystems, Visual Numerics, and Virginia Tech's Advanced Communications and Information Technology Center (ACITC), and were made possible by the creation of the Scientific Modeling and Visualization Classroom (SMVC). Modules discussed here were largely motivated by a student project, "Educational Atomic Models Using PV-Wave and Java" by Arturo Falck, in ESM4714: Scientific Visual Data Analysis and Multimedia, spring semester 1996 [2]. The purpose of this project was to create a user-friendly Web-based interface to interact with larger computer simulation models of cracks and dislocations in crystal lattices by using CGI (Common Gateway Interface). With CGI, an interactive Web-based form was created that students used to 1) enter information required by the simulation, 2) compile that information into a data file, 3) submit this file as a batch job to a remote supercomputer, and finally to 4) send raw data of the simulation back to the server where images of data were generated and were returned for viewing to the remote-site student computer. Unique to this project was the level of industrial participation by SUN Microsystems and Visual Numerics in the creation of the Java Web-based interface [3]. Early Java prototypes developed at Virginia Tech have been replaced with JWave interfaces developed by Visual Numerics except for the Network Programming Interface Builder (NPIB), which has replaced the original CGI interface previously described with the We-based Java framework, but the same functionality has been maintained. An example of an NPIB form used to calculate wave surface geometries associated with a fourth order stiffness tensor is shown in Figure 1(a). A more detailed discussion on the current status of NPIB and JWave follows.

1.2 Current Status

1.2.1 Overview

Development of the Java-Web server continued with additional funding from the NSF Combined Research and Curriculum Development (CRCD) Program. With NSF funding the server was upgraded to a SUN Sparc10 with 1Gigabyte of memory that could be used to handle larger simulations, which generated better (i.e., more representative) results for analysis by students. The earlier version of NPIB, which links students at their personnel computers to remote-site supercomputers, was created entirely with Java. Hence this open-source Java-Web server could be implemented at other universities using standard Java-based technology on affordable UNIX, NT, or Linux servers. For this project, a SUN Sparc10 Ultra was selected for development, since it represented an entry-level system that most departments can afford.

The first course was organized on the Web server with hyperlinks to Web modules that were divided into lectures, assignments, and examples [4]. The first course focused on atomistic and continuum mechanics models, and the second course will focus on models that predict mechanical behavior at the scale between the atomistic and continuum. Details on module content development of the first course are available elsewhere [5]. Here we describe the development of the Web-based Java framework and explain how the graphical user interface (GUI) design of these interactive modules facilitated students' efforts to parametrically study the relationships modeled and simulated by computer programs written by the researchers and instructors.

1.2.2 NPIB

The first version of NPIB, version 1.0, which was used in the first class, required instructors to learn a simple text-based syntax to create the desired NPIB form layout. Although simple, this process became tedious when the creation of larger forms was required. We were motivated to eliminate this syntax in NPIB version 2.0 and to simplify the process so that instructors could focus on building content. Sun's JDK 1.1 was chosen for the development of both versions of NPIB because of its cross-platform and web browser support. Since the server was also written in Java, it is capable of running on Windows NT or a more robust UNIX platform. Version 1.0 of NPIB will be discussed first, followed by a discussion of version 2.0 of

NPIB. Both versions were designed so that the instructor can avoid doing Web-based programming.

A simple example of a short NPIB version 1.0 form and the corresponding syntax is shown in Figures 1(a) and 1(b). Results are shown in Figure 2. The syntax shown in Figure 1(b) is simple enough so that the reader can deduce the syntax simply by observation. This particular problem illustrates the usefulness of the NPIB form whereby 1) instructors can create a useful GUI for students without learning Java, and 2) students can submit multiple parameter lists, and can view, interpret, and compare results in an easy to understand format. Hence the focus is not on programming but creating the content and process whereby students can explore and experiment with a variety of possible solutions.

The NPIB form in Fig 1(a) shows all the necessary information (fourth order stiffness tensor components and density) required to solve Christoffels' equation [6]. Christoffels' formulation reduces to an eigenvalue problem whose eigenvalues correspond to three wave surfaces, and whose eigenvectors correspond to the particle displacement vibration directions (wave-polarizations) shown here as color gradients mapped onto the wave surfaces. For materials with lower order symmetry, complex wave surface geometries can occur: hence, particular attention is given to viewing the results graphically, either with a VRML† Web-based viewer or in the CAVE when an immersive environment is preferred. At the bottom of the form is a submit button next to an e-mail address. Instructions at the top of the NPIB form, not shown here, explain how students can retrieve results at the specified e-mail address by using either the Web-based built-in e-mail tool or an e-mail tool of their choice.

For this example, graphical interpretation of the numerical solutions is particularly important. Exact solutions only exist in principle material planes [6], and numerical solutions are required when wave surfaces exist outside these planes. Numerical IMSL † subroutines are used to solve for the eigenvalues and eigenvectors, and PV-Wave† is used to generate the polygonal data sets in VRML format, so that the student can first view the results in a Web-based VRML viewer and, if necessary, in the CAVE as an Inventor† file. A special format in VRML 1.0 was created, see [7], to allow the student to view the same file with transparency in both the Web-based VRML viewer and the CAVE. Here transparency is required to observe how

† VRML is Virtual Reality Modeling Language and Inventor is an SGI graphical format (public domain); IMSL and PV-Wave are commercial software sold by Visual Numerics Inc.

the QL, QT, and T wave surfaces connect into a single connected wave surface. For this type of problem, students can explore how small variations in the stiffnesses, shown in Figure 1(a) as parameters in the NPIB form, can lead to significant changes in wave surface geometries, as shown in Figure 2.

Version 2.0 of NPIB does not require the instructor to learn any syntax to create the form layout. The primary difference between versions 2.0 and 1.0 is the addition of a GUI to build the form and to format the output from the form. Building and modifying forms in version 1.0 was a laborious task. The new "what you see is what you get" (WYSIWYG) builder allows the user to create and edit the form through direct manipulation of graphical components. Components are added by selecting the desired component type from the insert menu and then assigning a unique name, as shown in Figure 3(a). This name will be used later in formatting the output of the form. Once a component has been added to the form, dragging it with the mouse can move it. All component properties can be edited through dialog, which appears when the component is selected, as shown in Figure 3(b). The process of editing properties is similar to that in a Java bean box [8].

The format for the output is defined by typing it in the output editor, as shown in Figure 3(c). Upon submission the output will be formatted exactly as it appears in the editor. There are key words for declaring files and linking to components in the form. The file keyword (<file=XXXX.dat>) denotes a new file which is created on the server when the NPIB form is submitted. Links to form components (<link=name>) mark where the value of the named form components is to be inserted into the output file. The "link=name" refers to the name the component was given when it was created.

Another significant design change is the addition of the "Form Interface." This interface is not necessarily noticeable to the end user. It facilitates the use of other components with the NPIB forms. Any Java class that implements the "Form Interface" can be used in the same way as native components such as text boxes. These new components can implement a new means of data input or preprocess data before submission.

1.2.3 Jwave

JWave, unlike NPIB, requires developers to have a working knowledge of Java. Early attempts to create PV-Wave/Java applets ("Wapplets") were made collaboratively between Virginia Tech and Visual Numerics Inc. An early prototype developed at Virginia Tech, called Visualizer [9], had features similar to the current JWave product but was developed only as a proof of concept. The interested reader can consult a complete set of documentation and source code for Visualizer elsewhere [9]. Visualizer no

longer works on our current upgraded Java-Wave server because we chose to discontinue Visualizer upgrades, and to use JWave instead.

Visual Numerics introduced JWave as a means to connect an interactive Web-based GUI with simple sliders, buttons, dials, etc., to a legacy programming language called PV-Wave. Most of the calculations and images generated by PV-Wave legacy code are generated at a server, and images are sent back to the remote-site client, in this case the student's computer. Although JWave developers need to have a working knowledge of Java, the idea here was again to try to minimize the students' efforts outside the area of direct relevance to solving the problem at hand. The student should not need to install any software or learn any particular programming language to understand the various parts and functions of a model that is accessed by the JWave Web-based GUI. Figure 4(a) shows a typical JWave GUI as it would appear on a Web-browser, i.e., Internet Explorer or Netscape Navigator. This particular interface shows all the parameters needed to calculate the radial and tangential stress distributions in a thick-walled cylindrical pressure vessel.

The JWave GUI shown in Figure 4(a) is a necessary but not sufficient interface for the students to learn about stress distributions in a thick-walled cylinder. To assist students in the analysis and interpretation of results, a problem definition with equations and parameters, as shown in Figure 4(b), must be located near the JWave GUI. Instructors need to design the Web-page layout and to program it in Java in a meaningful way, much as they would prepare for a class lecture. Although instructors must have a working knowledge of Java, JWave provides the necessary communication between PV-Wave and the Java applet via the JWave server software. This connection takes the input from the student's Web page, transforms it into the correct variables, and then sends it to the legacy PV-Wave code for execution. Once the code finishes execution, the results, variables, and graphical plots, are returned through the same connection that was established with the student's Web page. JWave is designed to minimize the programming efforts of the instructor and take advantage of any legacy code written in PV-Wave and IMSL.

1.3 Lessons Learned and Future Module Development

The first NSF-CRCD course was taught in the fall semester, 1998, and a second class will be taught this spring semester, 2000. Both classes are three-credit hour classes, which meet for one hour three times a week: Monday, Wednesday, and Friday. Mondays and Wednesdays were reserved for lectures, and on Fridays students met with instructors in the Scientific

Modeling and Visualization Classroom (SMVC), which like the Virginia Tech CAVE is an ACITC facility.

1.3.1 Things that worked well

Except for occasional server downtimes, the NPIB and JWave interfaces worked well. Until the final evaluation is completed, conclusions are speculative. From first impressions, however, it appears that the most productive time spent using these modules occurred when students and instructors met in the SMVC on Fridays. Fridays were more like lab sessions where students could ask questions and try out their ideas with comments from the professors who also helped interpret the simulation results. Instructors also received valuable feedback on how the JWave and NPIB forms were working and what needed to be improved. Friday sessions also built student confidence for successful completion of their homework assignments.

1.3.2 Things that need more work:

Although the NPIB form worked well, the "builder" part of the NPIB was improved with more features but was still not stable enough for instructors to build their own forms. Consequently, the technical support team members built all the NPIB version 1.0 forms using a scripting syntax. When completed, NPIB version 2.0 will allow instructors who are not Java literate more freedom in building interactive NPIB forms. The NPIB form only worked on UNIX workstations with Netscape 4.5. It was not until near the end of the semester that we got the NPIB forms to work on Windows NT. This was largely due to the way Windows NT handles screen refresh.

1.3.3 Lessons Learned

Java interface development is a difficult, if not impossible, task for most professors who do not have backgrounds in computer science. These same professors are also not capable of routine systems administration needed for configuring and maintaining Java-Web servers. Hence, there must be a commitment from the department or college to support a courseware server and to train professors on how to access and use systems such as the NPIB. Because of limited resources and reluctance to accept new technology, building and supporting courseware servers has been the most difficult aspect of this project. The Java-Wave server is now maintained by the Problem Solving Environment (PSE) group in the Department of Computer Science [10].

In this class, we also discovered that programmers and system administrators need to work more closely than in years past, when typically all that was needed was to install a standard language compiler and to have a user service group for answering any questions. With the advent of the network, professors and technical support staff can no longer afford to isolate themselves in the "new world" of computing, where popular Web-based software applications are constantly changing. Successful projects now require that professors devote more time to learning computing skills and to working more closely in teams. We also experienced firsthand how Java needs to be maintained as a standard when early interfaces developed in Netscape's IFC had to be rewritten. Our experience in team-teaching this class was rewarding but difficult. More time was spent solving technical problems than was spent developing course content. We hope the tools already developed and experience gained in the first class will reverse this trend in the next class.

To continue the learning experience, witnessed on Fridays in the SMVC, students need access to the Java-Web server from outside the SMVC. Although convenient to manage, students should not be required to go a single-workstation classroom environment. Some universities, because of security issues and convenience of management, prefer to isolate these resources from remote access. Such policies are counterproductive when all students are also required to own their own personal computers and where professors who are located off-site are expected to create courseware materials for those students.

1.3.4 Future Developments

Another course will be taught spring semester, 2000 at a first-year graduate level. For this graduate class, the same material can be taught at a more comprehensive level, and new simulations of cracks at or near bi-material interfaces will be modeled both at a continuum and atomistic level and comparisons made. The Java-Web server will be upgraded to JWave 2.0, better security measures will be implemented without restricting access to the anonymous ftp site, and the NPIB builder feature will be completed in version 2.0 so that professors can build their own Java forms. The server will be upgraded to 1 Gigabyte of memory with 27 Gigabytes of disk space. Modules in the spring 2000 class will be extensible to other classes taught in the Engineering Science and Mechanics (ESM) Department. We hope that this interest will grow to other ESM classes and that the ESM Department will eventually support their own JWave courseware server.

2. ACKNOWLEDGEMENTS

Authors acknowledge the NSF grant "Combined Research and Curriculum Development: Computer Simulation of Material Behavior – From Atomistic to the Continuum Level" (EEC-9700815) and the foundation grant from SUN Microsystems Inc. and Visual Numerics Inc. to create the Scientific Modeling and Visualization Classroom.

3. REFERENCES

1. SUN-VNI Wave-Java Web Server: http://www.jwave.vt.edu
2. Scientific Visual Data Analysis and Multimedia, ESM4714: http://www.sv.vt.edu/classes/ESM4714/ESM4714.html
3. Kriz, R.D. and Farkas, D. "Using Materials Resources on the World Wide Web for Introductory Materials Science Teaching," J. Materials Education, Vol. 19 No. (1&2), pp. 111-119, (1997).
4. Computer Simulation of Behavior from the Atomistic to the Continuum Level, ESM4984: http://www.jwave.vt.edu/crcd/
5. Kriz, R.D., Farkas, D., and Batra, R.C., "Integrating Simulation Research into Curriculum Modules on Mechanical Behavior for Materials: From the Atomistic to the Continuum", J. Materials Education, Vol. 21, No. (1&2), pp. 43-52, (1999).
6. Ledbetter, H.M. and Kriz, R.D., "Elastic-Wave Surfaces in Solids," Physica Status Solidi, Vol. 114, pp. 475-480, (1982).
7. VRML 1.0 format necessary for viewing in both the CAVE and VRML Web-based viewer: http://www.sv.vt.edu/classes/vrml/exercise3.html
8. Java Beans: http://java.sun.com/beans/
9. Visualizer Web home page: http://www.jwave.vt.edu/javaprj/viscprj/visualizer/html
10. Problem Solving Environment: http://www.cs.vt.edu/~pse/

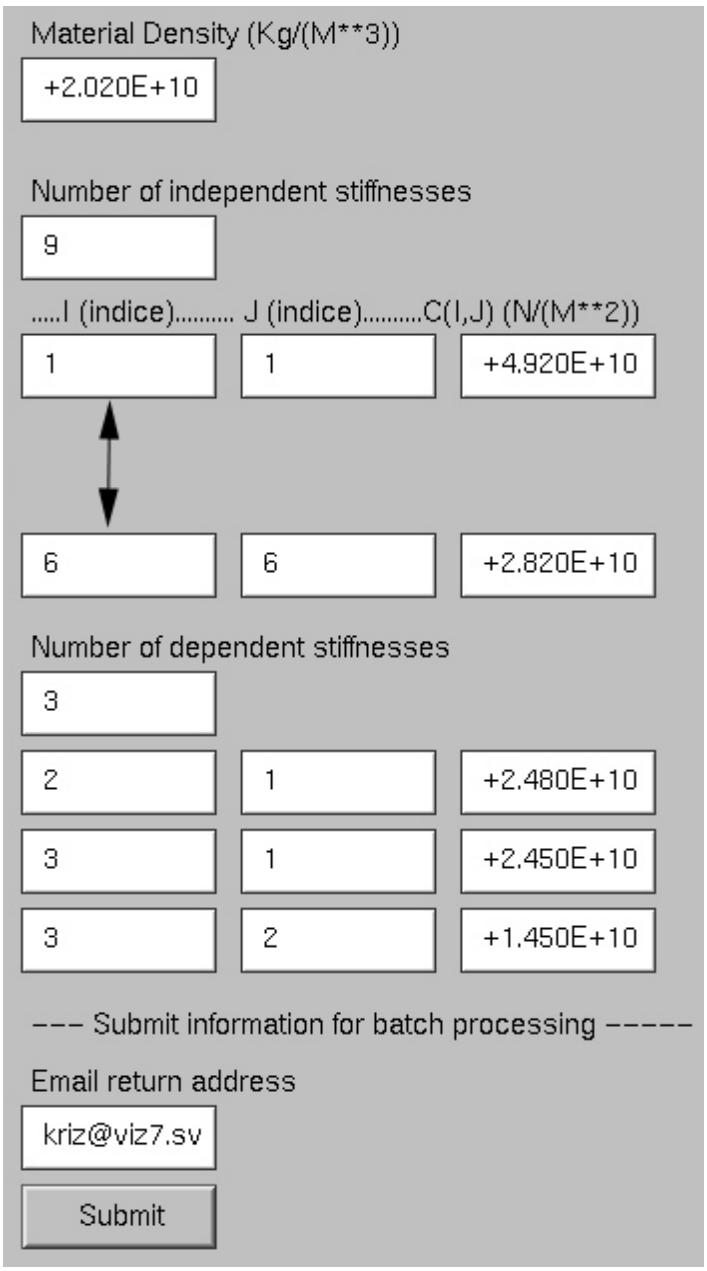


Figure 1(a) NPIB (Network Programming Interface Builder) version 1.0 form used by students to study the geometry of stress waves propagating through an anisotropic continuum: given the density and components of the fourth order stiffness tensor.

---------- BEGINNING OF FILE ----------

```
f dw2ds.data 0
l  20 10 ----- dw2ds.data ------------------------------
tf 20 60  Material: / 'Calcium Formate' / 1 1 0
tf 20 120 Comment: / 'Orthorhombic Symmetry' / 2 1 0
tf 20 180 Material Density (Kg/(M**3)) / +2.020E+10 / 3
1 0
tf 20 260 Number of independent stiffnesses / 9 / 4 1 0
l  20 300 .....I (indice).... J (indice)....C(I,J)
(N/(M**2))
tf 20 320 / 1 / 5 1 0
tf 130 320 / 1 / 5 2 0
tf 240 320 / +4.920E+10 / 5 3 0
.
.
.
tf 20 640 / 6 / 13 1 0
tf 130 640 / 6 / 13 2 0
tf 240 640 / +2.820E+10 / 13 3 0
tf 20 710 Number of dependent stiffnesses / 3 / 14 1 0
tf 20 750 / 2 / 15 1 0
tf 130 750 / 1 / 15 2 0
tf 240 750 / +2.480E+10 / 15 3 0
tf 20 790 / 3 / 16 1 0
tf 130 790 / 1 / 16 2 0
tf 240 790 / +2.450E+10 / 16 3 0
tf 20 830 / 3 / 17 1 0
tf 130 830 / 2 / 17 2 0
tf 240 830 / +1.450E+10 / 17 3 0
l 20 880  --- Submit information for batch processing --
tf 20 930 Email return address / kriz@viz7.sv.vt.edu / -
1 -1 -1
b 20 970 Submit / wavesurf_orth

-------------------------- END OF FILE -----------------
```

Figure 1(b) Text file showing syntax that was used to create the NPIB form shown in Figure 1(a).

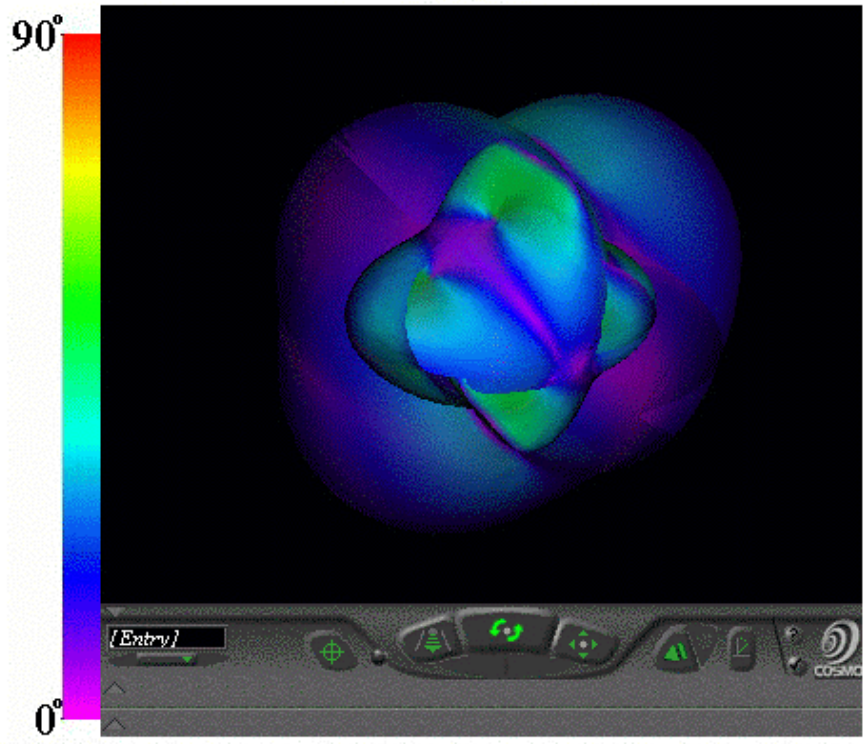


Figure 2. Example of results returned as a navigable VRML viewer embedded in browser showing a connected T, QT, and QL wave surface corresponding to the density and stiffness tensor shown in Figure 1(a).

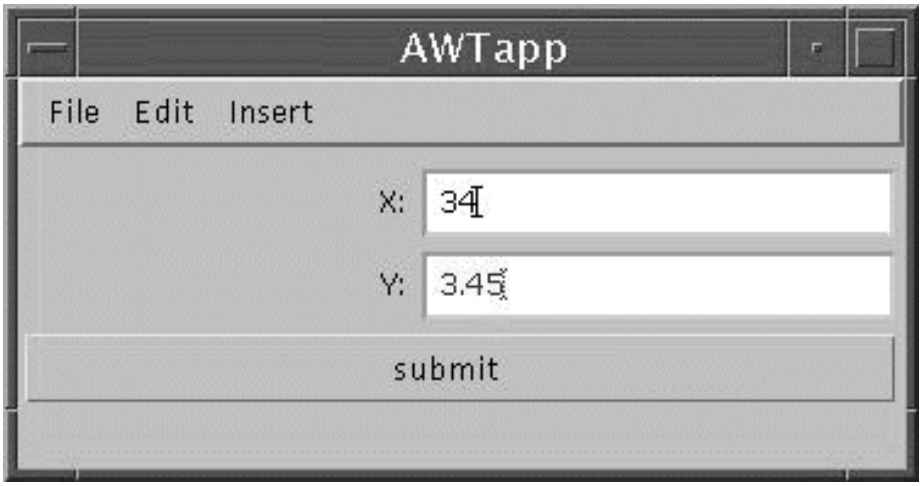


Figure 3. NPIB form version 2.0: (a) “AWTapp” final Web-based form as viewed by the student.

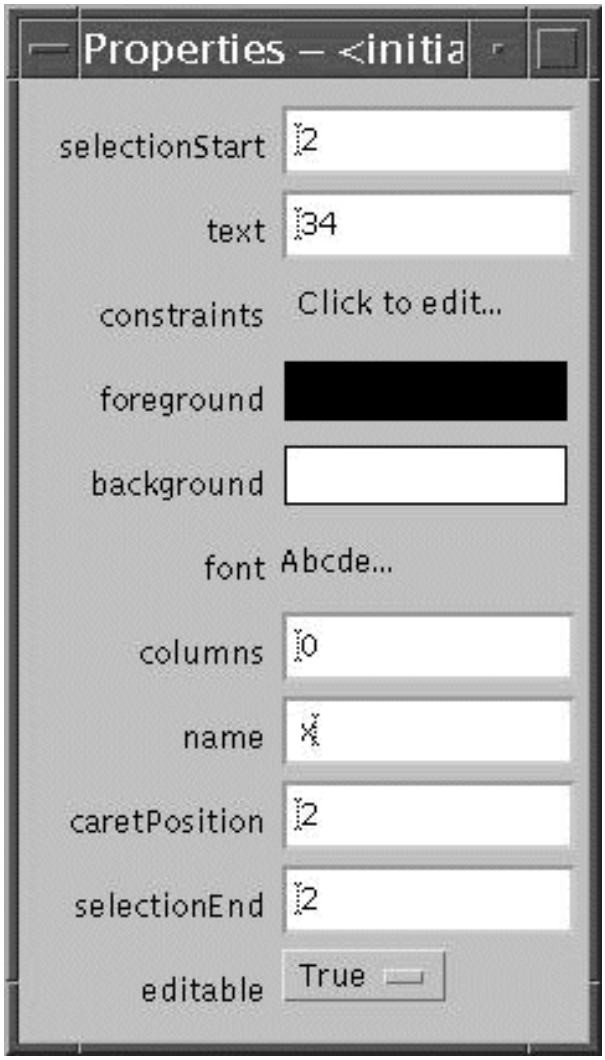


Figure 3. NPIB form version 2.0: (b) “Properties” of NPIB components are assigned here.

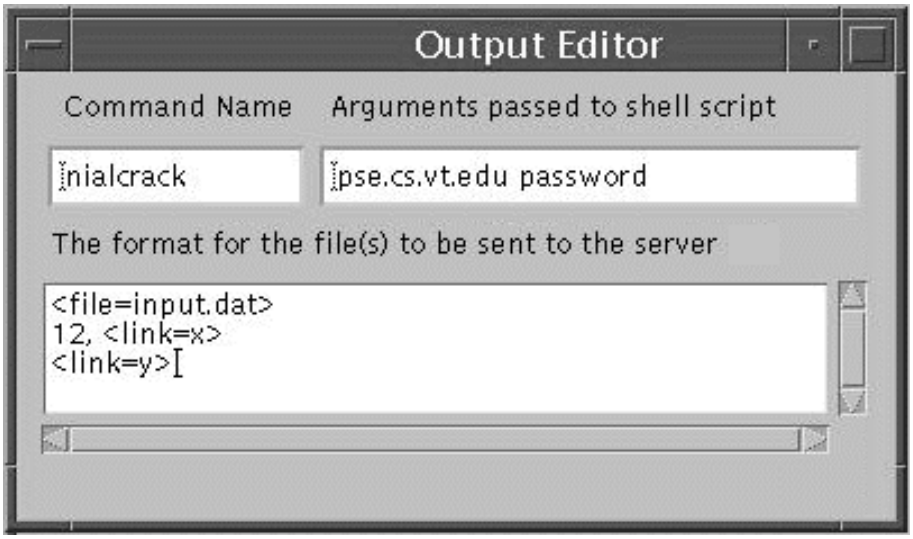


Figure 3. NPIB from version 2.0: (c) "Output Editor" controls from format of the output.

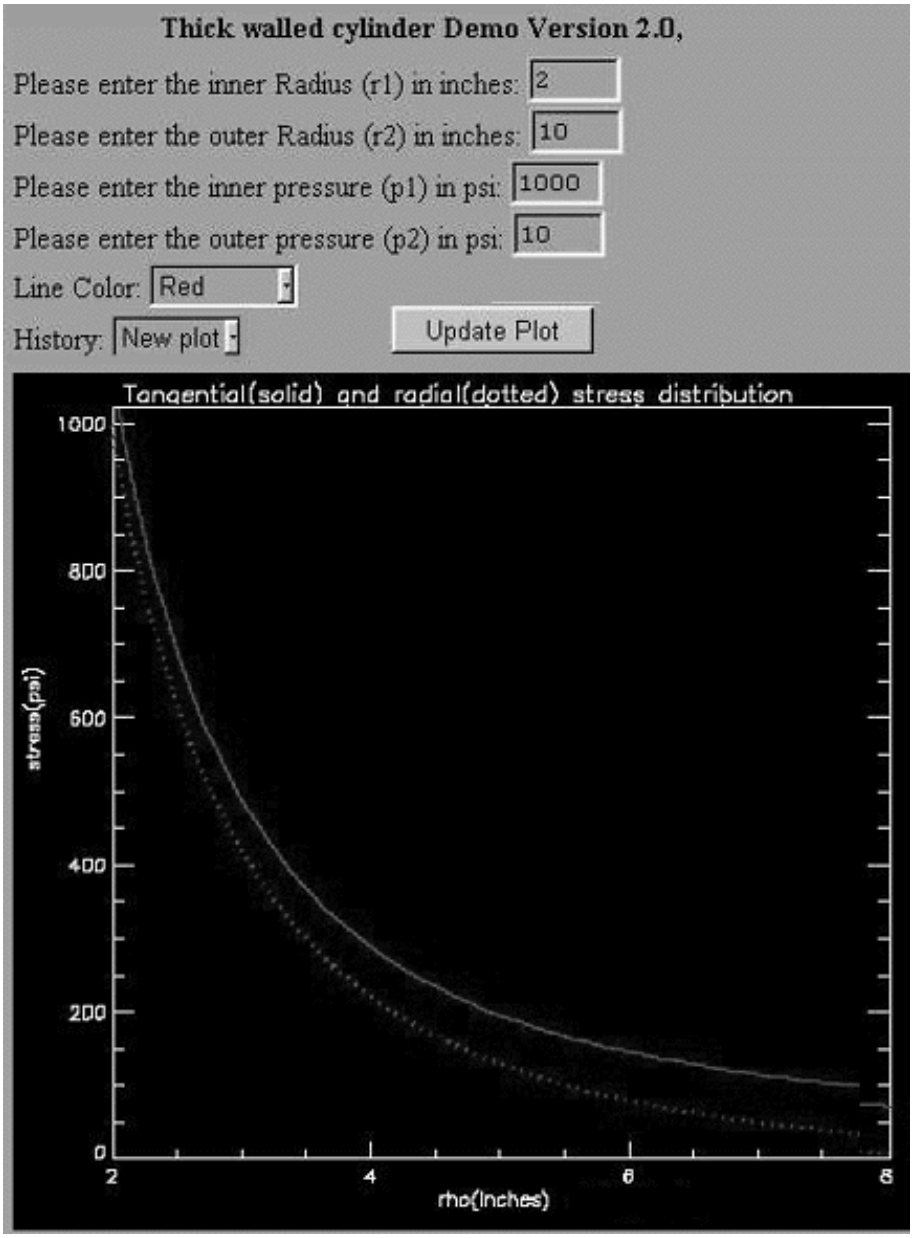
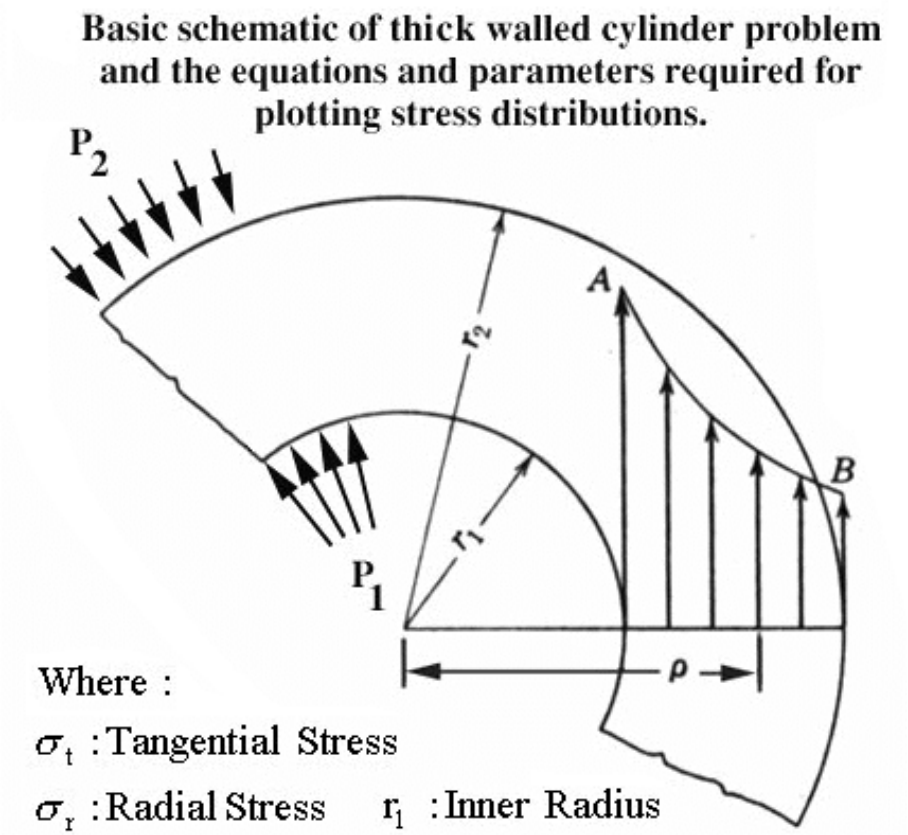


Figure 4. Thick Walled Cylinder: (a) JWave 2.0 form with window showing graphical results.



$$\sigma_t = \frac{p_1 r_1^2 - p_2 r_2^2 + (r_1^2 r_2^2/\rho^2)(p_1 - p_2)}{r_2^2 - r_1^2}$$

$$\sigma_r = \frac{p_2 r_2^2 - p_1 r_1^2 + (r_1^2 r_2^2/\rho^2)(p_1 - p_2)}{r_2^2 - r_1^2}$$

Figure 4. Thick Walled Cylinder: (b) problem definition.