Interactive Scientific Visual Data Analysis using Java, PV-Wave, and IMSL

Ronald D. Kriz[1], Randy T. Levensalor[2], and Sanjiv D. Parikh[3]
Virginia Polytechnic Institute and State University

Abstract

A Java framework is described for creating an interface with legacy code through a Web browser. This interface was created in the development of modules for teaching courses on the Mechanical Behavior of Materials. Modules incorporate the results of state of the art simulation techniques. When appropriate, students studied structure property relationships predicted by simulations in an immersive CAVE™ environment. Simulation results span various length scales that start at the atomistic level that use embedded atom method techniques, and continues with simulations at the continuum level that use finite element method techniques. These modules used legacy code written by the researchers teaching these classes. Considerable attention was focused on creating a Web-based interface that allowed researchers to easily construct, and students to easily use, interfaces that access legacy code in an interactive format. Hence when possible commercial software such as JWave was used, but when working with supercomputing simulations a Java based three-tiered architecture called Network Programming Interface Builder (NPIB) was needed to communicate between the clients, the Java server, and remote site supercomputers running the legacy code.

1. Background

Modules were developed and distributed on our SUN-VNI Wave-Java server, [Kriz, 1995]. Early efforts to create a distributed, Web-based, visual computing environment were funded by SUN Microsystems, Visual Numerics Inc. (VNI) and Virginia Tech's Advanced Communications and Information Technology Center (ACITC) 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, [Kriz, 1991]. 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 a Web-based interface.

Interactive Web-based forms were 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 simulation back to server where images of data were generated for viewing and then transferred back to the remote-site client. 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, see [Kriz, 1997]. 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 Web-based Java framework, but the same functionality has been maintained. A more detailed discussion on NPIB and JWave follows.

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 Ultra with 1Gigabyte of memory that could be used to handle larger simulations which could then be used to generate more representative results for analysis by students. The earlier version of NPIB that links students at their personal 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 two SUN Sparc10 Ultra were 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, [Kriz, 1997]. 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 were published in [Kriz, 1999]. Here we describe the development of the Web-based Java framework and explain how the graphical user interface (GUI) was designed and facilitated students in their efforts to parametrically study the relationships modeled and simulated by computer program written by the researchers and instructors.

[1] Associate Professor, Dept. Engineering Science and Mechanics, Norris Hall, Blacksburg, VA 24061, rkriz@vt.edu
[2] Graduate Research Associate, Dept. Computer Science, McBryde Hall, Blacksburg, VA 24061, randyl@vt.edu
[3] Graduate Research Associate, Dept. Engineering Science and Mechanics, Norris Hall, Blacksburg, VA 24061, engineer@vt.edu

2. Development with JWave and NPIB

2.1 JWave

Figure 1 shows the interactive JWave applet GUI. This GUI form allows the student to fill in a few parameters, select a color from the pull down menu, and toggle a switch. Once the form is filled out and the update plot button is pressed, the applet will set all the parameters and make an execute call to the PV-Wave procedure. The procedure will analyze the data and generate a graphical plot corresponding to the input data and then send it back to be displayed in the HTML form. Two files are required to make this happen, the JWave applet embedded in a HTML document and the PV-Wave procedure file located on the server.

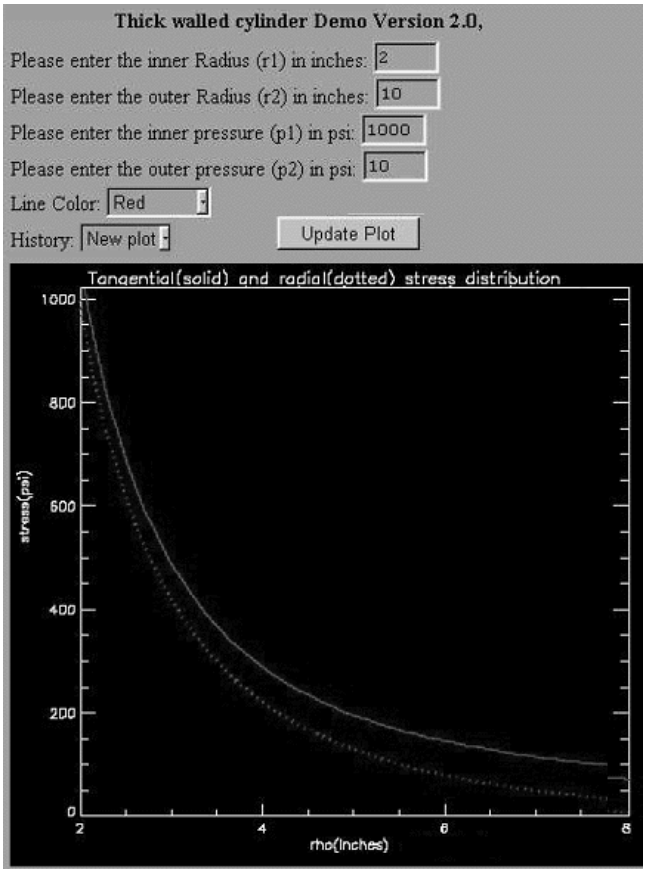


Figure 1: Thick Walled Cylinder: (a) JWave 2.0 form with window showing graphical result

A JWave 2.0 applet was developed following a few simple easy-to-follow steps. These steps were to install the Visual Numerics Inc. (VNI) JWave 2.0 on our SUN-VNI Sparc10 Ultra server (with VNI's PV-Wave preinstalled), run the JWave server manager to configure the paths and directory structure for accessing java archive (jar) files. These files contain all of the classes required by

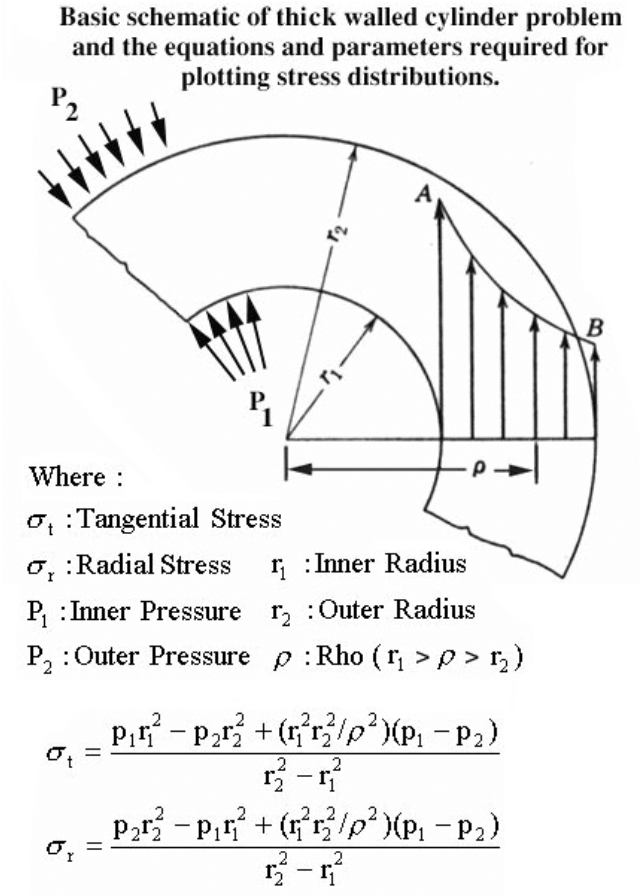


Figure 1: Thick Walled Cylinder: (b) problem definition.

the JWave server and create javascript embedded HTML forms which act as the GUI interface for the data input to and the graphical output from PV-Wave procedures.

The major part of time involved in developing a JWave applet went into understanding how the javascript/JWave applet communicated with PV-Wave. This is done through JWave wrappers. JWave wrappers are VNI's in-house java functions that are archived into the JWave<type>.jar files. These functions are used within the embedded javascript to call and execute specific PV-Wave commands. The following is an example of how data is passed from the applet to a PV-Wave procedure. Here 'setNamedColorSet' would be used in the applet to set a specific color for an object and 'GET_NAMED_COLOR' would be used in the PV-Wave procedure to return the color for that graphical object being generated via the PV-Wave procedure. There are a variety of these wrapper functions within JWave that allow the ease of coding simple yet effective HTML forms which access PV-Wave's massive data analysis and visualization toolkit.

Beyond the JWave wrapper API, one only needs to be able to pick up basic javascript and the PV-Wave programming language. JWave was specifically designed to work with legacy PV-Wave procedure files. To generate web-based forms with text fields, buttons, check boxes,

pull-down menus, etc., a basic understanding of the simple javascript GUI generation is sufficient. The following source code is a sample HTML (javascript) code used to generate a text input field and return its value to the JWave applet's setparameter function (which will for this case sets the value for the variable 'r1' to user's input):

```
Please enter the inner Radius (r1) in inches:
<INPUT TYPE=TEXT NAME=r1 VALUE=2
        SIZE=5
         MAXLENGTH=5
          onChange=”dataNeedsRecalc=true”>
<BR>
…
setFormParam(twcplot3, JWavePlotForm.r1);
…
```

The following is the corresponding code for the PV-Wave procedure, which will get the 'r1' variable's value and incorporate it into the analysis of the procedure.

```
function TWC3, client_data
…
p(3) = getParam(client_data, 'r1', /Value, Default = 2)
…
```

Due to the simplicity of coding these objects in javascript and being a part of the HTML form, the only coding needed is the creation of a single HTML/javascript file. To make learning this task easier, VNI created a few demonstration JWave applets which have the buttons / pull-down / input fields pre-coded into the HTML forms. With a basic understanding of the outline of the applet, it is easy to generate web-based interactive data analysis and visualization toolkits via few modifications to VNI's JWave applet demos and existing PV-Wave procedures.

The applets developed here were JWave version 2.0 applets, however JWave 3.0 is just recently introduced. JWave 3.0 is a newer version of the JWave server, this version is much like JWave 2.0 in its ease of use and carries over all of the functionality with added new features, such as its' two way communication between the applet and the procedure and back to the applet.

To aide in development of future JWave applets, a Web page with links to download the JWave software, download the sample demos, and download the tutorials as well as the JWave user's guide was setup on our website.

2.2 NPI

From the start NPIB has been designed as a simple intermediary between students and complex legacy engineering programs. Use of existing tools, platform independence, and minimal system administration are major themes, which guided the development cycle. Java using Sun's JDK 1.1 was chosen as the primary development platform. JDK 1.1 runs on all major server platforms from PCs and Macintoshes to UNIX servers.

NPIB can be divided into six distinct phases:1) Creation, 2) Form Display, 3)Data Delivery, 4)Local Execution, 5) Remote Execution, and 6) Results Delivery.

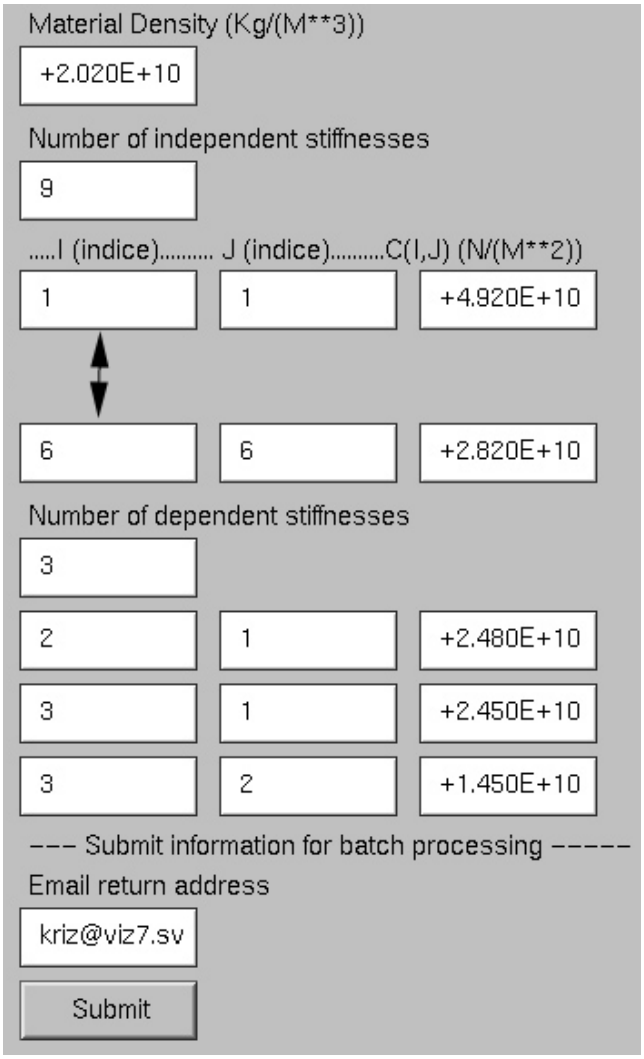


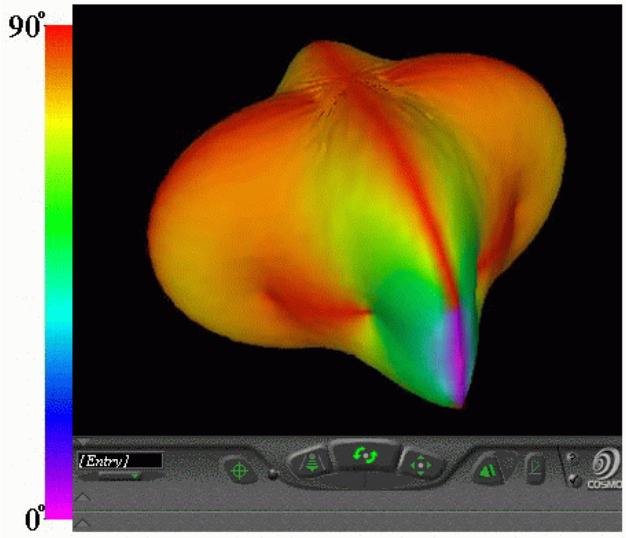
Figure 2: (a) NPIB (Network Programming Interface Builder) version 1.0 of the fourth order stiffness tensor

2.2.1 Creation

Form creation encompassed the majority of the development time. Teachers create the forms, which will collect the data from the students. The look and feel of the forms and the format of the files outputted are decided at this phase. An earlier attempt used a single GUI to view for arranging both the layout of the form and format the output. This approach was cumbersome and inflexible. Another attempt used a language to describe the layout of the forms and the output, see Figure 2. This gave the

```
---------------------- BEGINNING OF FILE ----------------------
* dw2ds.data 0
1 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
1 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
1 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 2: (b) Text file showing syntax that was used to create the NPIB form shown in Figure 1(a).



The second wave surface: COLOR = Displacement Deviation.

Figure 3: Example of results returned as a navigable VRML viewer embedded in browser showing the intermediate QT wave surface corresponding to the density and stiffness tensor tabulated in Figure 2(a) and 2(b), [Ledbetter, 1982]

teachers the most flexibility and control. Due to the strict control of this language, modifying these forms was laborious task. Since a unique language was implemented there was a large learning curve associated with this method.

The current method incorporates the strong points from the previous designs while attempting to eliminate their major drawbacks. A GUI is used to layout the forms through a WYSIWIG approach, see Figure 4a. Teachers were able to directly manipulate the form components and view them, as the students will. Properties for the specific form elements are adjusted by changing values in a dialog

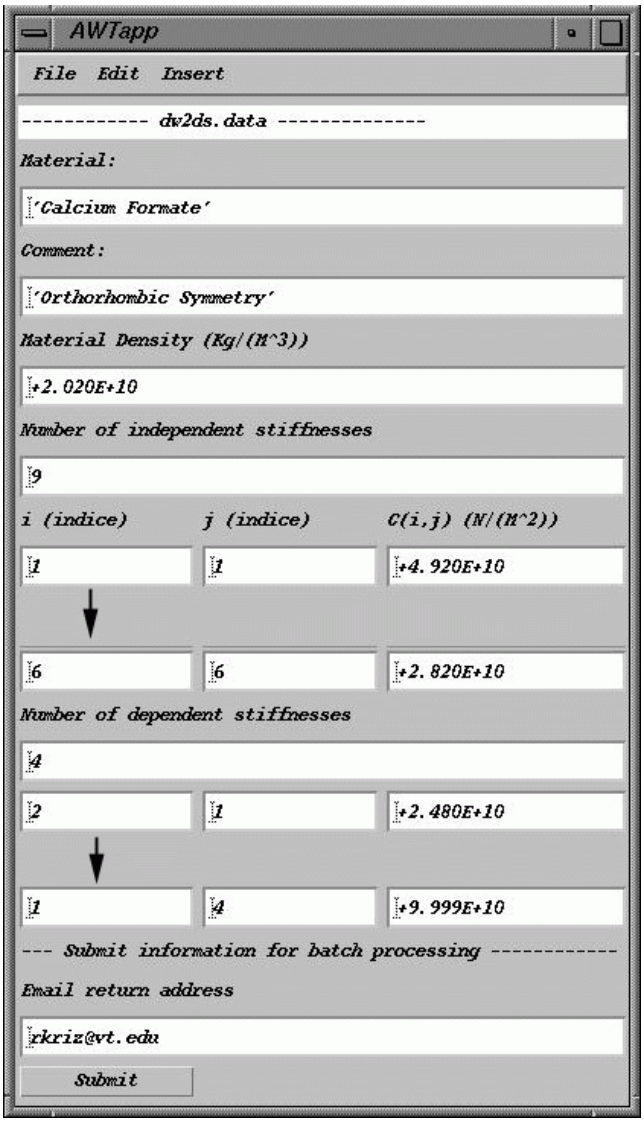


Figure 4: NPIB form version 2.0: (a) "AWTapp" direct manipulation window for creating form by teachers. Same as final Web-based form as viewed by the student, see Figure 5.

similar to Java Beans [Sun], see Figure 4b. The output is formatted using a simpler and more intuitive language, see Figure 4c.

At present no formal usability tests have been conducted to verify these claims. However, direct feedback from all users indicates that the goals laid out for the final approach have been satisfactorily achieved.

2.2.2 Form Display:

This displays the forms created by the teachers, see Figure 5. The viewer is a striped down version of the builder. It is viewed as an applet using a standard web browser. This approach was chosen because of the wide availability of web browsers for students.

2.2.3 Data Delivery:

Deliver the data from the students to be processed on the server. The data is submitted to the server via a socket. Before the data is sent, it is broken into formatted files. The server receives the text and dumps into the appropriate files with the names specified from the form. The server, which receives this data, is written in java for platform independence.

2.2.4 Local Execution:

Since several methods are in place to execute programs no additional programs were created to aid in the actual execution of the programs. Current implementations use basic shell scripts, which mirror commands as they would by typed if being run manually. The location of the data files and parameters passed from the form are passed as command line parameters in a script file. Others methods of execution can be used including c, c++, fortran, perl or any other executable may be used to invoke the programs.

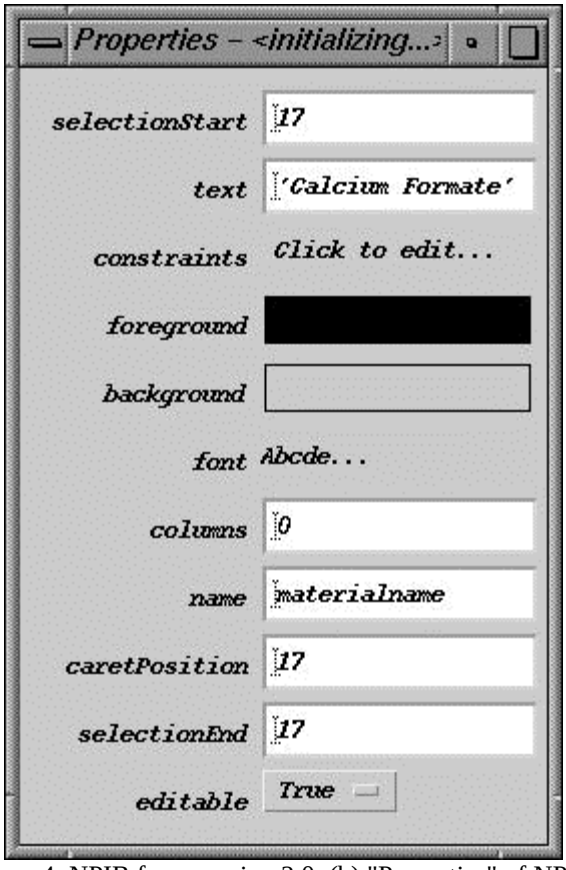


Figure 4: NPIB form version 2.0: (b) "Properties" of NPIB components are assigned here.

2.2.5 Remote execution:

Initially instances of the receiver were run on machines other than the web server to execute programs remotely.

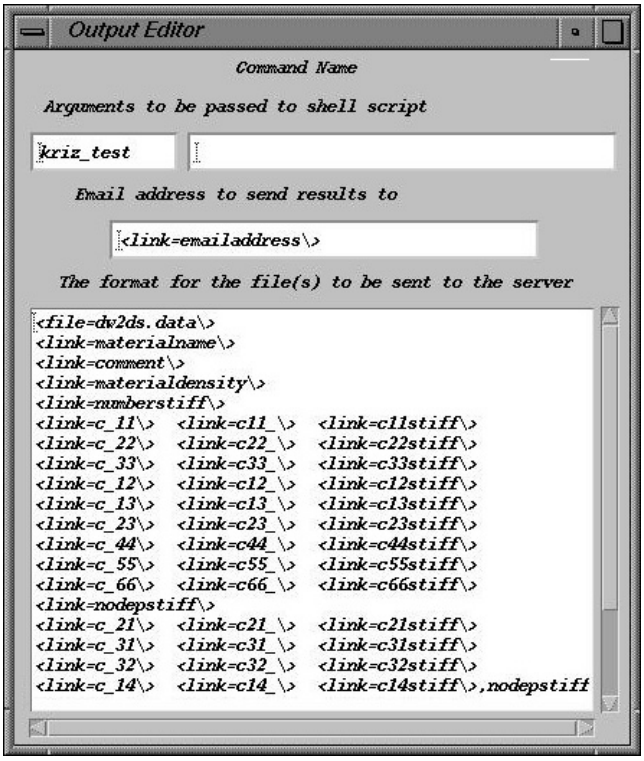


Figure 4: NPIB from version 2.0: (c) "Output Editor" controls from format of the output.

This caused several complications at the implementation level, exercise level, and administrative level. This required the receiver be installed and always running on any machine these programs need to run on. Finally, we looked to existing tools as in the local execution and found several solutions. The preferred method is the use "expect" script and secure shell (ssh) to move files and execute remote programs.

2.2.6 Results Display:

Final results are returned to the students in two forms. First an additional web browser is opened displaying the directory where the results will be located. This provided the students with a means of acquiring the status of the current executions and gives them the results when the simulation is completed, see Figure 6. The second approach is the URL of the results is sent to the students via e-mail. This also provides the students with an archive of the location of the results.

3. Results and Lessons Learned

The first NSF-CRCD course was taught in the Fall semester, 1998 and a second class will be taught this Fall 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 was reserved for lectures and on Fridays students met with instructors in the Scientific Modeling and Visualization Classroom (SMVC) of the ACITC.

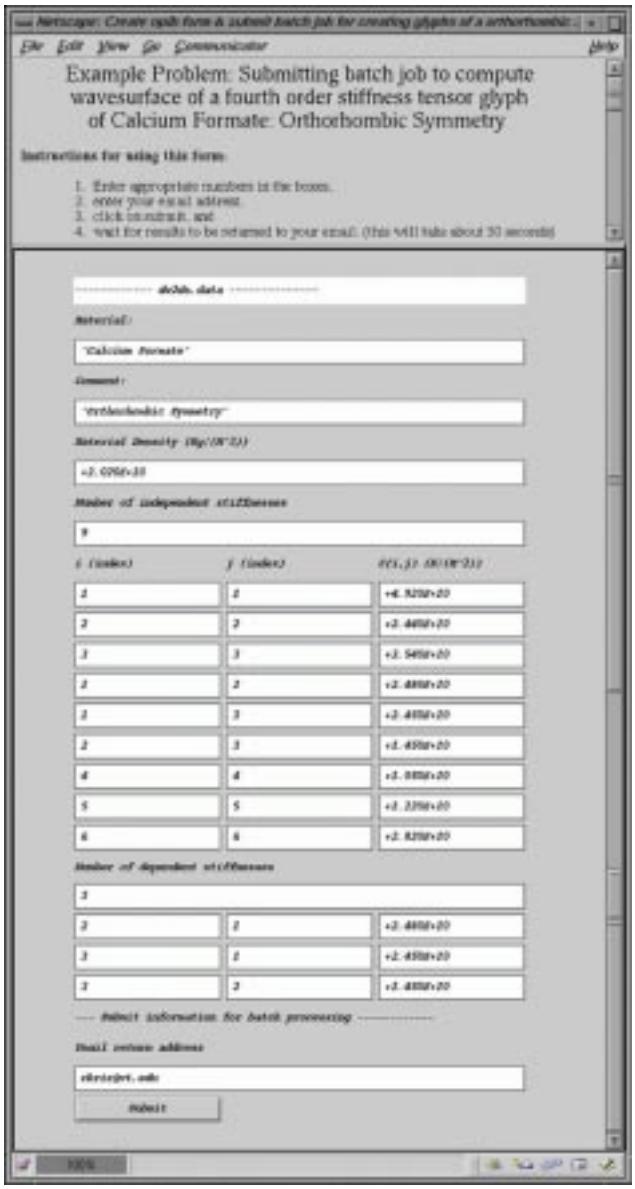


Figure 5. NPIB form used by students to submit batch job.

3.1 Things that worked well

Except for the occasional server downtimes, the NPIB and JWave interfaces worked well. Until the final evaluation is completed, conclusions are speculative. From first impressions however, it appeared that the most productive time spent using these modules was 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

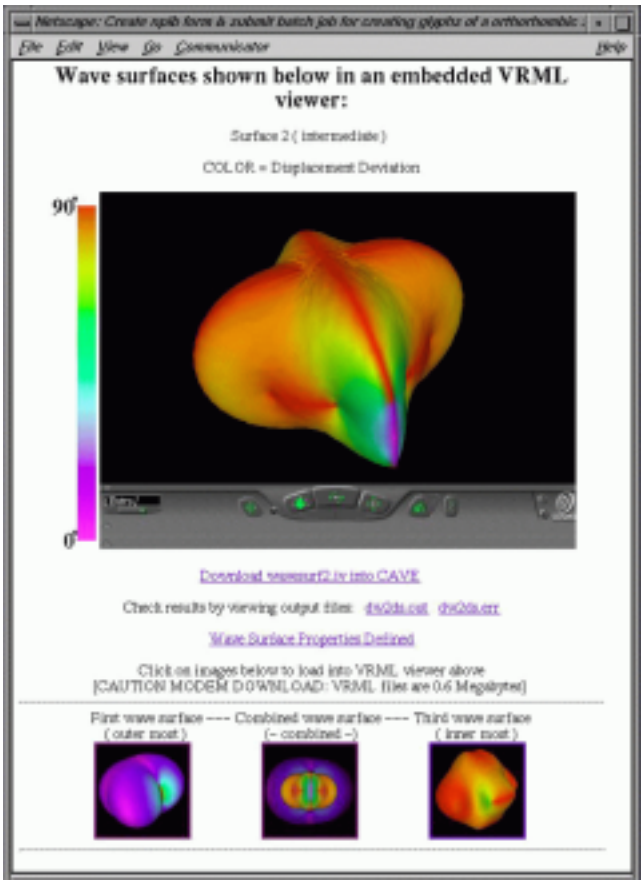


Figure 6: Simulation results of NIPB form shown in Figure 5 returned in embedded VRML Web viewer.

was needed to be improved. Friday sessions also built student confidence for successful completion of their homework assignments.

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 not stable enough for the instructors to build their own forms. Consequently the technical support team members built all the NPIB version 1.0 forms using a scripting syntax. NPIB version 2.0 allows the instructor, who is not Java literate, more freedom in building interactive NPIB forms. At first the NPIB form only worked on the UNIX workstations with Netscape 4.5. Only near the end of the semester did we get the NPIB forms to work on Windows-NT. This was largely due to the way Windows-NT handles screen refresh.

3.3 Lessons Learned

Java interface development is a difficult, if not an impossible task, for most professors who do not have backgrounds in computer science. Even using javascript with HTML is beyond the abilities of most professors. 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 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 presently maintained by the Problem Solving Environment (PSE) group in the Department of Computer Science, [Shaffer, 99].

In this class we also discovered that programmers and system administrators need to work more closely than in years past where typically all that was needed was to install a standard language compiler, with user service group to answer 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 learning computing skills and how to work more closely in teams. We also experienced first hand how Java needs to be maintained as a standard: early interfaces developed in Netscapes' IFC had to be rewritten. Our experience in team teaching this class was a rewarding but difficult task. More time was spent solving technical problems than was spent developing course content. We hope this trend reverses in the next class because of the tools already developed and experience gained in the first 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 to any particular 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 every student is also required to own his/her own personal computer and where professors who are located off-site, are expected to create courseware materials but cannot do so remotely.

3.4 Future Developments

Another course will be taught in the Fall semester, 2000 at a first year graduate level. The Java-Web server will be upgraded to JWave 3.0, better security measures will be implemented without restricting access to the anonymous ftp site, and the NPIB builder feature will be completed in version 3.0 so that professors can build their own Java forms. We hope to deploy NPIB3.0 and JWave3.0 on an NT server in the near future. Other ideas for future development are: 1) validate and verify data before they are submitted to server to check for data types: int, floats, etc., and 2) auto-generate NPIB forms given a specific input file (work with existing legacy data files). Although our current Sparc10 Ultra server has been upgraded to 1GB of memory with 27GB of disk space, we hope to access a Sun Enterprise 6500 computer with seventeen 400MHz (8MB cache) processors, 18GB memory, and 144GB of RAID disk as a remote-site computer for the larger simulations. Modules in the Fall 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 eventually the ESM Department will support their own JWave courseware server.

4. Acknowledgments

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 of the ACITC.

5. References

Kriz, R.D., SUN-VNI Wave-Java Server: http://www.jwave.vt.edu, 1995.

Kriz, R.D., Scientific Visual Data Analysis and Multimedia: http://www.sv.vt.edu/classes/ESM4714/ESM4714.html, 1991

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.

Kriz, R.D., Farkas, D., and Batra, R.C., Computer Simulation of Behavior from the Atomistic to the Continuum Level: http://www.jwave.vt.edu/crcd/, 1997

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.

Ledbetter, H.M. and Kriz, R.D., "Elastic-Wave Surfaces in Solids," Physica Status Solidi, Vol. 114, pp. 475-480, 1982.

Parikh, S.D., 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, 1997

SUN Microsystems Java Beans: http://java.sun.com/beans

Coupleux, F., Visualizer: http://www.jwave.vt.edu/javaprj/viscprj/visualizer.html, 1996.

Shaffer, C., Problem Solving Environment: http://www.cs.vt.edu/~pse, 1999.