To The Graduate School:

The members of the Committee approve the thesis of John W. Regnier presented on March 24, 1998.

Bogdan M. Wilamowski, Chairman

Steven L. Horner

Henry R. Bauer III

APPROVED:

Raymond G. Jacquot, Head, Department of Electrical Engineering

Donald S. Warder, Dean, The Graduate School

Regnier, John W., Computer-Aided Engineering Through Internet Networks,
M.S., Department of Electrical Engineering, May, 1998.

Computer network technology has become a popular and efficient means of computing. Most companies and research institutions use networks to some extent on a regular basis. This thesis demonstrates the capability of computer-aided design (CAD) tools through computer networks using current network programming technologies and Web browsers. There are several benefits that make networked CAD tools desirable. A networked application can be used remotely through any network connection. Any operating system can be used to access a networked application. Much less installation time and configuration time is required because the application is located on one central machine. The "SPICE Internet Package" (SIP) which has been developed is an example of such a networked CAD tool.

COMPUTER-AIDED ENGINEERING THROUGH INTERNET NETWORKS

by
John W. Regnier

A thesis submitted to the Department of Electrical Engineering
and The Graduate School of The University of Wyoming
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE
in
ELECTRICAL ENGINEERING

Laramie, Wyoming
May, 1998

Table of Contents

Table of Figures

Chapter 1

Introduction

Computer networks provide the ability to access a large variety of information. Through the Internet, information is made available from around the world, and Intranet networks provide connectivity for a smaller more isolated domain like a company or a school, for example. Networks are efficient and have become a popular means of computing. "Using the principle of EDA [Electronic Design Automation] tools running on powerful hosts...the World Wide Web could transform design by ushering in an era of pay-per-use tools, explained Richard Newton, a professor at the University of California at Berkeley" [1]. Instead of buying expensive software that is seldom used, it would be more efficient and more convenient to use CAD (Computer-Aided Design) tools through the Internet or Intranet networks. The different uses and capabilities of networks and the World Wide Web are discussed in Chapter 2. The World Wide Web is already used extensively by the engineering community in various areas. Chapter 3 outlines the architecture of the World Wide Web and some of the technologies available for developing remote applications to run on the Internet and Intranet networks. Finally, in Chapter 4 an application called the SPICE Internet Package (SIP) is presented. The

SIP application has been developed for use through the Internet and Intranet networks.

The SIP program was developed in part to help meet the needs of some educational institutions where computing resources may be limited relative to the number of students enrolled. Because the SIP program can be accessed remotely with a Web browser through a network connection, SPICE simulation and analysis can be performed from a computer lab at school, library, or dorm room. Only one copy of the SPICE program is needed on the Web server, so only one computer installation and configuration is required. The SIP program has a user-friendly graphical user interface which helps facilitate the learning of SPICE simulation and analysis. It has a graphical post-processor for generating graphical analysis from simulation data. The SPICE engine used is spice3f5, which was developed at the University of California at Berkeley, and an unlimited number of transistors are available to the designer unlike various "student versions" of SPICE programs that are available.

For the reasons mentioned above, the SPICE Internet Package could be quite useful for educational institutions while industrial software vendors could also develop a similar scheme for their electronic design automation tools.

Chapter 2

Engineering Design Through Networks

2.1 Introduction

In order for networks to be used for engineering and design, they must be stable, reliable, and fast. This has not always been the case with the World Wide Web. It was mainly used for electronic mail and file transfer in its beginning stages because it was slow and the technologies we have today were not yet developed. With new innovations and technologies, the Internet now provides connectivity and communication for millions of people around the world. An emerging use of the World Wide Web in engineering is remote computer-aided engineering (CAE) tools that are accessible through the Internet possibly on a pay-per-use basis.

2.2 Evolution of the World Wide Web

In the early 1970's the ARPAnet, the beginning of the Internet, was initiated by the Defense Department's Advanced Research Projects Agency (ARPA). The ARPAnet was created with four operating nodes connecting UCLA, University of California Santa Barbara, Stanford University, and the University of Utah [6]. If any one link in the

network failed, packets could still be routed via the remaining links, thus providing the needed fault tolerance and reliability that was sought. In the mid-1980's the National Science Foundation (NSF) supplied funding to accelerate the already rapid growth of the Internet. Through the help of the NSF extensive connectivity for campus networks at educational institutions, government agencies, and commercial businesses was obtained by the late 1980's. The growth of the Internet has been phenomenal. Figure 2.1 shows how the number of Internet hosts has grown. The Internet hosts have increased almost

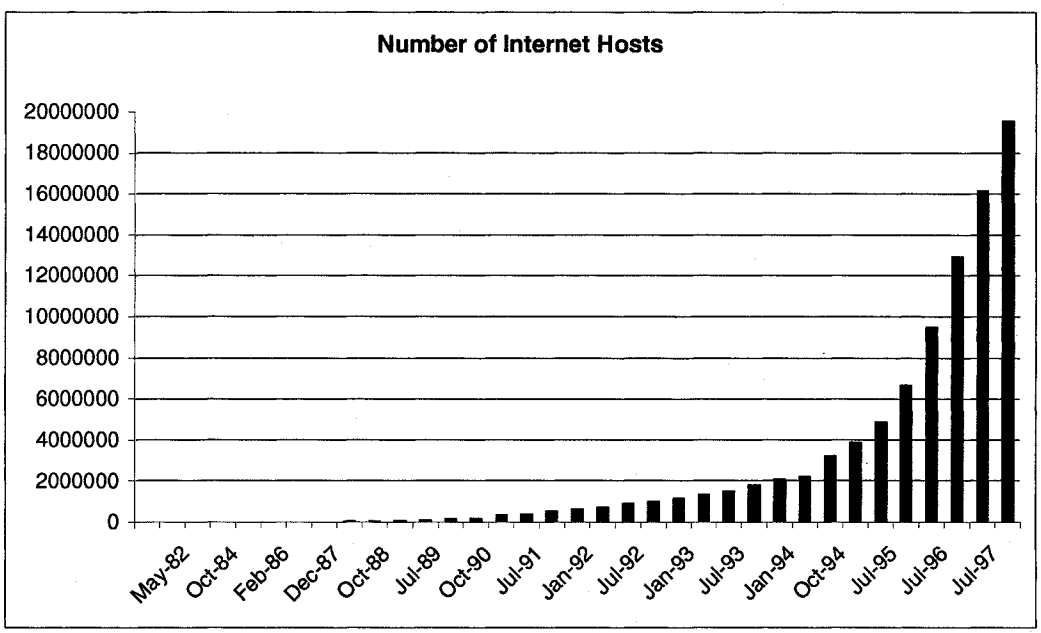

Figure 2.1: Internet Growth

exponentially over the past few years. Many engineers are using the World Wide Web to communicate on a daily basis. It is reasonable to envision how engineers, who are already seasoned in Internet navigation and communication, would easily pick up and

learn new Internet computer-aided design tools and incorporate them into their computing workspace.

2.3 Modern Uses of the Internet

In the beginning the Internet was mainly used for sending electronic mail and transferring files using the File Transfer Protocol (FTP). A limited number of people were able to use these resources because of the limited networking and computing resources. In the recent past the World Wide Web has "transformed the Internet from an exclusive country club frequented by the 'well-connected' and privileged few, to a huge public arena visited daily by people from all walks of life...the World Wide Web has opened the Internet to the masses." [6]. Today, electronic mail and FTP are still frequently used with intuitive graphical user interfaces. Internet navigational tools integrate a graphical user interface with sights and sounds and applications for all imaginable areas. The Internet is used in part for delivery of online electronic catalogs and product information, for distributing software electronically, for browsing multimedia art galleries, for business, and for publishing newspapers and "netcasting" radio and video programs. Educational institutions take advantage of the web by providing course home pages where students can see class policies or grades and other information for a certain class. Businesses also use home pages extensively to post information and provide customer support.

2.4 Computer-Aided Engineering Through the Internet

In these modern times because of the massive popularity of the World Wide Web, a company or business almost has to have a web site to survive. "Virtually every EDA [Electronic Design Automation] software vendor surveyed by *Spectrum* ... has a Web site..." notes Linda Geppert, a *Spectrum* editor. The Web sites are used presently to facilitate the distribution of software, provide corporate profiles, product summaries, post job opportunities, and offer customer support services. All these services are easily accessed with an Internet Web browser. The potential is greater, however. Richard Newton, a professor at the University of California at Berkeley, explains "Using the principle of EDA tools running on powerful hosts and with adequate Internet bandwidth, the World Wide Web could transform design by ushering in an era of pay-per-use tools." [1].

A tool developed for use over the World Wide Web has several advantages. One is that operating system independence is achieved. Any computer that has a standard Web browser is capable of accessing the networked applications seamlessly. Another advantage is that the actual executable program only has to be executed on one machine and that might be at the vendor's place or at a business or an educational institution somewhere. Another advantage is that a pay-per-use scheme could be developed so the user is charged based on how much the program is used. This would greatly benefit those who only need to use an application occasionally. Yet another benefit is in the area of software distribution. For a networked program the software vendor only needs to

distribute passwords to allow customers to access the programs, or if the program is being distributed, only one copy is needed for the server on a network that will be running the application. This would reduce the time spent installing and configuring software on multiple machines which can be quite time consuming each time a new version of a software product is released.

It has been reasoned that Electronic Design Automation could soon include a "remote set of tools as part of a complete design flow over the Internet" [1]. An example of such a remote EDA tool is the SPICE Internet Package that has been developed for use over networks. This program is presented in Chapter 4 to demonstrate the capability of computer-aided engineering through networks.

Chapter 3

Internet Network Programming

3.1 Introduction

Now that the Internet is both reliable and fast, technologies are available and are still being developed, or enhanced, that allow us to take advantage of this vast resource. "The World Wide Web organizes, transmits, and retrieves information of all types by using a combination of hypertext, graphics, and multimedia technologies, unified in a set of naming conventions, network protocols, and document formats, and realized by using a client-server architecture." [6]. The client-server architecture is the key to network programming and is outlined below. Several network programming tools available today include the Common Gateway Interface (CGI), Java, Perl, HTML, JavaScript, ActiveX, and VBScript. The software development strategy for networked applications is different from typical application programming. Network software development requires analysis of the distributed computing requirements at the beginning of the software development cycle. Decisions about what third party software will be used and what software will be developed, as well as the programming languages and technologies that will be utilized to meet the design requirements, are all important issues for the software developer.

3.2 World Wide Web Client-Server Architecture

The architecture of the World Wide Web is illustrated in Figure 3.1. Notice that the client machines are running a variety of operating systems. The client machines can communicate even though they are using different operating systems because of the standardized naming conventions and network protocols. Because of the

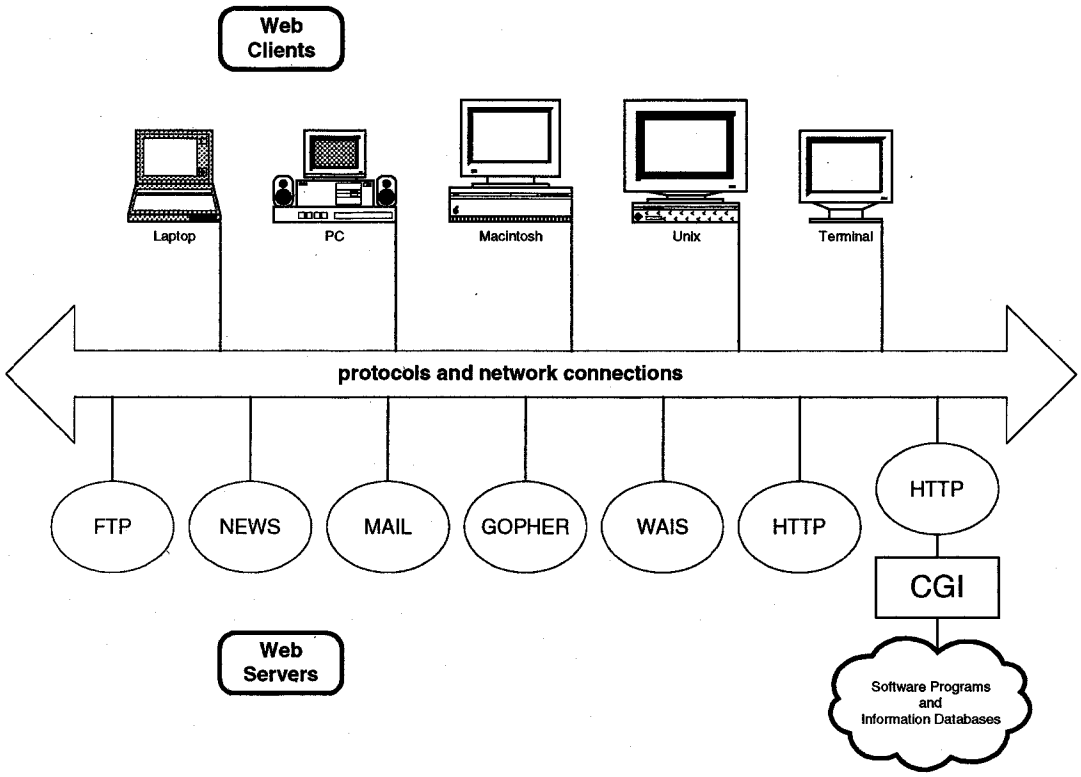

Figure 3.1: World Wide Web Architecture

standardization of the network protocols "clients allow users to navigate the Web or even interact with the server[s] in interesting ways...[and] servers allow Internet sites to

publish information or export data to the world." [6]. The client and server machines perform different tasks by communicating with each other through network connections. Notice the CGI block that connects a Hyper-Text Transfer Protocol (HTTP) server to information databases and software programs. The CGI provides a dynamic communication link among the client machines and the information databases and software programs.

3.3 Web Clients and Web Servers

A client program that runs on a desktop computer can access a multitude of different Web servers distributed across the Internet. This type of program is typically called a Web browser. There are several Web browsers available including Netscape Navigator and MS Internet Explorer. A Web client, or Web browser, can do more than simply access, or browse, information from Web servers. Modern Web browsers can provide forms that users complete in order to send information to servers, using the Common Gateway Interface which allows a bi-directional flow of information. This capability gives Web browsers the interactivity that is now frequently used. Examples of client applications include entering a query string for a search at the Yahoo! home page or submitting a resumé to a company through their Web page. Another feature of a modern Web browser is the ability for dynamic interaction using scripting languages, like JavaScript, which is explained later in section 3.6. Another example is an embedded Java applet that is transmitted with the requested document. A Java applet runs

interactively in a Web browser that has a built in Java interpreter.

A server program typically runs on a multitasking workstation that is powerful enough to handle multiple simultaneous requests from throughout the Internet. Popular operating systems for hosting server programs are Unix and MS Windows/NT. The file system on a server holds documents that are requested by other clients. Also, servers can act as gateways to information databases and software programs through the Common Gateway Interface. Through CGI, a Web server "invokes a program script that takes information provided by the client...processes it according to the instructions in the script, and returns a Web page result to the client." [9]. An example of a Web server is the server that Yahoo! uses to accept a query string that was entered from their home page. This string is passed on to a script that initiates a search of a database. The resulting information is passed to the server and then back to the client with the results of the search displayed in the Web browser window.

3.4 HTTP Internet Protocol

The HTTP is a protocol that provides for the transfer of all kinds of data in an efficient manner across a network. Transferred data might include plain text, hypertext documents, Java applets, images, audio, or video. There are four steps in an HTTP transaction: connect, request, response, and close.

In the connect step, a client tries to establish a connection with a server somewhere on the Internet. This is instigated, for example, by the client program when a

user clicks a hyperlink on a Web page. The request step is where the client tells the server what information is wanted. Actions might be to "GET" information from the server or to "POST" information to the server from a filled-in form. The third step is the response step. Here the server tries to do what the client requested. Usually information is passed back to the client. The close step is the final step that terminates the network connection.

The HTTP protocol is stateless because a network connection is established and broken for each HTTP request. The dynamic capability becomes apparent when a request is made to a server that executes a CGI script and dynamic information is returned to the client based on the original data that was sent to the server.

3.5 The Common Gateway Interface

The key to providing advanced Web interactivity is the Common Gateway Interface. The CGI programming allows dynamic web page generation in a web browser based on user selections in the initial page displayed in the web browser. Communication between a CGI program and the web browser is accomplished through a network connection between the Web browser and a computer running an HTTP server program. A CGI script executes on a server when it receives a request to process information from a Web browser. The server then decides if the request should be granted and if the CGI program actually exists. If the authorization is secured, the server initiates the execution of the CGI program and returns the results to the Web browsers

that requested the processing. This is how Web pages can be dynamic. The CGI programs search databases or run programs on a server and send dynamic information back to the requester. The CGI "scripts" as they are called, can be written in many different languages. Some scripting languages that are commonly used are Perl, Tcl, C Shell, AppleScript, and VBScript. Some programming languages used to write CGI scripts include C, C++, and Visual Basic.

3.6 Programming Languages for Internet Programming

There are several programming languages for network programming like HTML, JavaScript, and Java. Other general purpose programming languages that are used sometimes for network programming include C, C++, Perl, Tcl. Some of these languages and their application in network programming are briefly discussed.

3.6.1 HyperText Markup Language (HTML)

The most widely used language in Internet programming is HTML. This basic language describes the structure and organization of the data in a document. Some example HTML code is shown in Figure 3.2. The tags, which are enclosed in angular brackets (<...>), are used to tell the Web browser how the information should be interpreted and displayed to the user. The information displayed is static because it can not be changed until another page is loaded. This language is plain and simple to write and read. One reason for the plainness of the language is that HTML documents can be

created on the fly by other programming languages like C, C++, Java, and Perl upon request.

The HTML tags are used in part to show headings, paragraphs, lists, character attributes like underlining or bold, and hyperlinks. As an example of an HTML document, consider the source HTML code in Figure 3.2. The source code for the

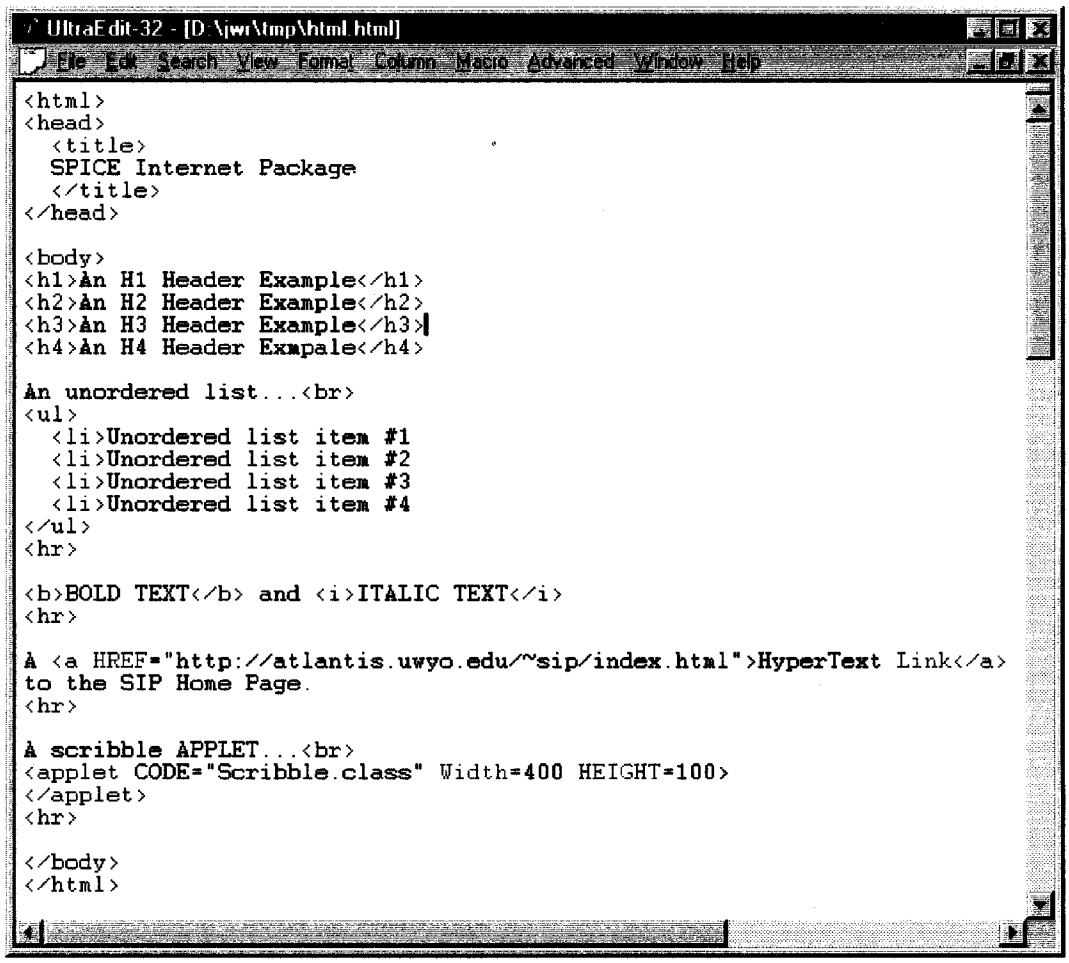

Figure 3.2: HTML Code

HTML document demonstrates only a few of the HTML tags available. The document

viewed with Netscape Navigator is shown in Figure 3.3, and the same document viewed with MS Internet Explorer is shown in Figure 3.4. Notice that both of the Web browsers interpret the HTML document in basically the same manner. Also notice that a Java applet has been included in the HTML source using the <applet>...</applet> tags. Applets are discussed in more detail in Section 3.6.3.

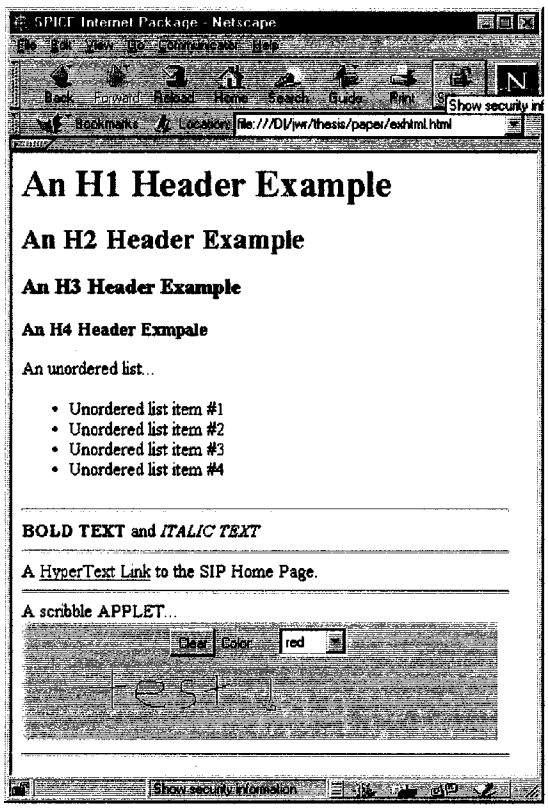

Figure 3.3: Netscape Navigator

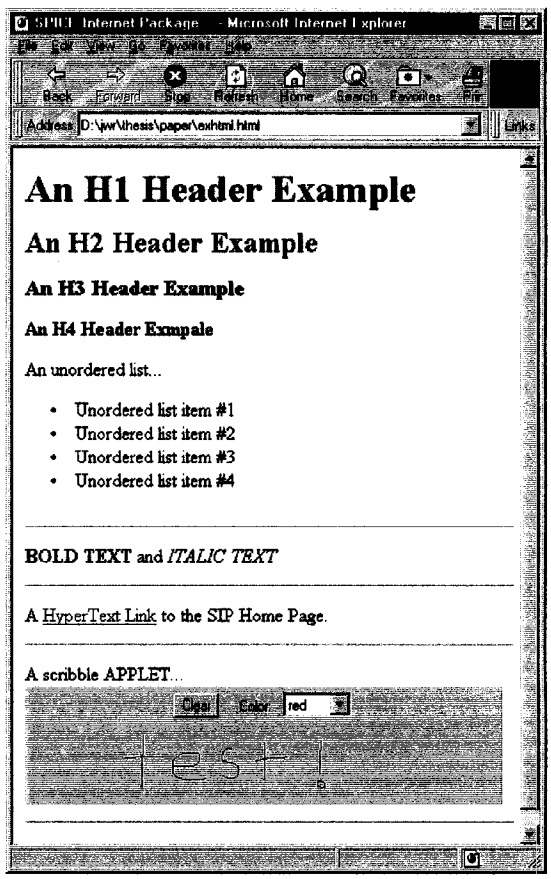

Figure 3.4: Internet Explorer

Forms are also included in the HTML document by the use of the <form>...</form> tags. Forms are important in the dynamic aspect that CGI

programming provides. With forms, the user can interact with the Web browser by entering information as well as submitting it for processing. This is usually done by clicking a button in the Web page.

3.6.2 JavaScript

This language is an object-oriented complement to HTML. The code is integrated into an HTML document. JavaScript is event driven and is derived from the programming language Java. There is no permanent storage using JavaScript; however, it is similar to C++ programming in many other aspects. One popular use of JavaScript is as a preprocessor for submitting a request to an HTTP server. By checking the validity before it is sent, the form can be checked before it ever reaches the server which decreases the work load of the server. In JavaScript events can be monitored so that when a user presses a button, or a page is loaded, some action can be initiated like updating the screen or changing the state of the page. As an example, consider the JavaScript code that has been added to the previous HTML code shown in Figure 3.5. JavaScript functions and other code are included in the <head>...</head> tags of an

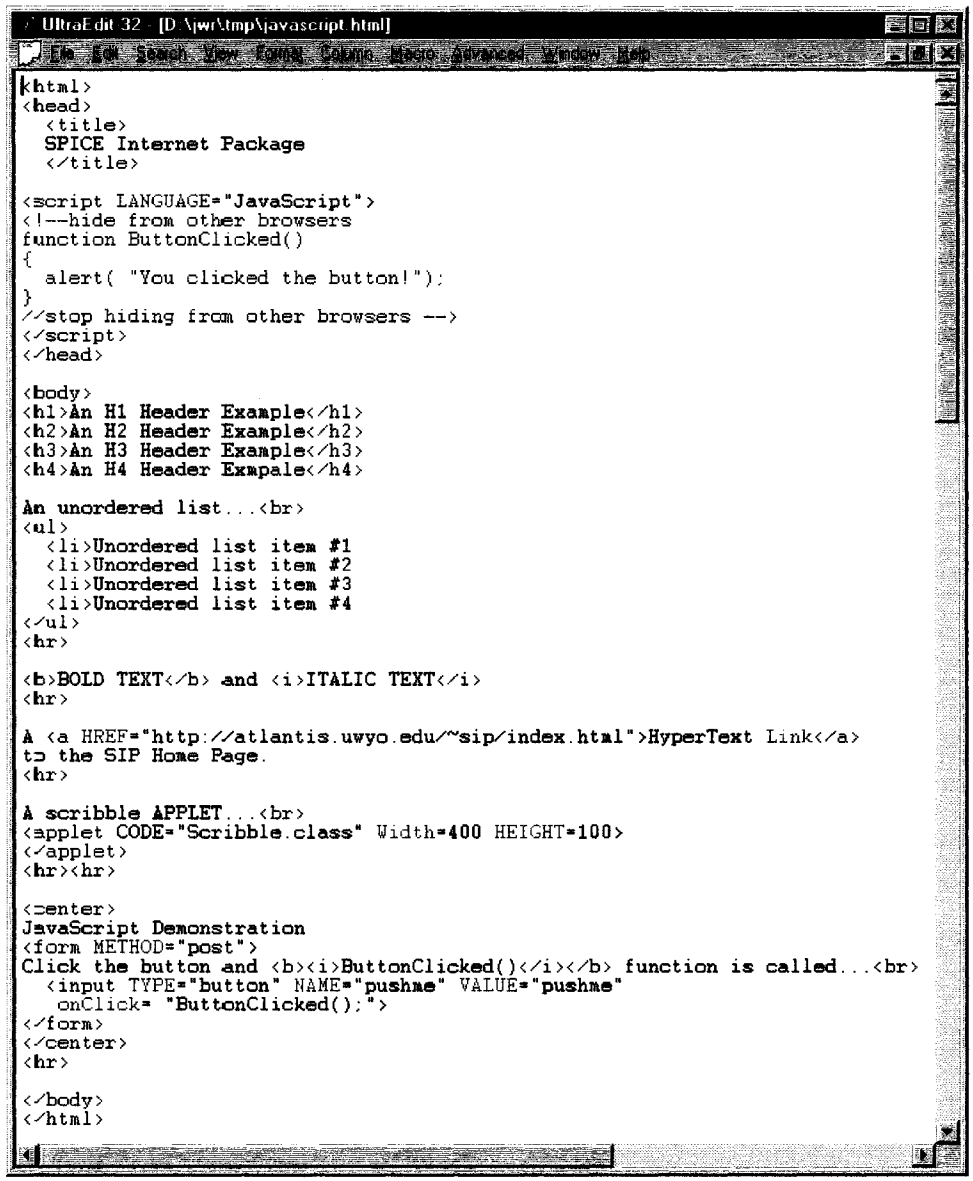

Figure 3.5: JavaScript Code

HTML document. The <script>...</script> tags enclose scripting language code in the head section of the document. Tag parameters in the body of an HTML document send messages based on user interaction. When opened in the Netscape Navigator Web browser, the resulting output is shown in Figure 3.6. When the button labeled "pushme"

shown at the bottom of Figure 3.6 is pushed, the alert dialog box in Figure 3.7 is asserted with the visible text, or value, of "You clicked the button!" This is an example of the event programming capabilities available with the JavaScript language within HTML documents. HTML and JavaScript provide the front-end graphical user interface that allows a user to click a button or enter a circuit filename and start circuit simulations as with the SPICE Internet Package. Events in JavaScript are monitored and processing of information is initiated through features of these two programming technologies.

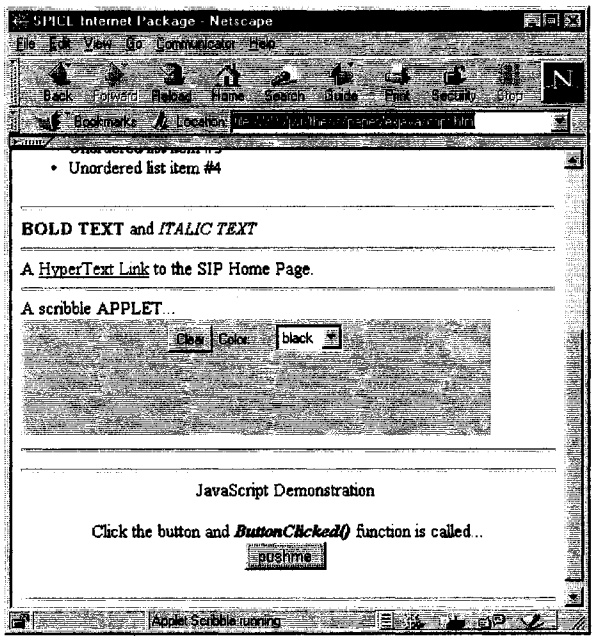

Figure 3.6: JavaScript Example

Figure 3.7: JavaScript Dialog Box

3.6.3 Java

Java has been developed especially for network programming. One text describes the language as "A simple, object-oriented, distributed, robust, secure, architecture neutral, portable, interpreted, high performance, multithreaded, dynamic language." [3]. The Java programming language is similar to C++ in many respects but with much more networking capability. Using Java for Internet applications is accomplished by downloading bytecodes from the Internet and running them on the requesting client machines. The bytecodes are generated using a Java compiler once on any platform, and they can then be interpreted on other machines that have Java-enabled Web browsers. This is how Java is able to be operating system independent. Java applets can use a modern graphical user interface including text boxes, buttons, list boxes, and the like. Applets can also trap events like keystrokes and mouse movements. Using Java applets on the client machine greatly reduces the workload on the server machine and increases the responsiveness of the client machines.

3.6.4 Perl

Perl is a programming language especially suited for CGI network programming and commonly used for text processing and system shell programming. Sometimes a Perl program is called a Perl script because it is an interpreted language and is not compiled like a C program. A Perl script is similar to a Unix shell program. Perl has many features that make it well suited to processing CGI requests and generating HTML

pages. A Perl script function is shown in Figure 3.8. The code is used in the SIP application to return an HTML document, on the fly, to the client if an error is encountered during the execution of the script. The variables *$keyword* and *$message* are passed to this function in the @_ array as shown on line 4 of Figure 3.8. The HTML code that is sent back to the HTTP server and then back to the Web browser is in bold print. Notice that some JavaScript code—the function onClick=window.close()—is embedded in the HTML code as well.

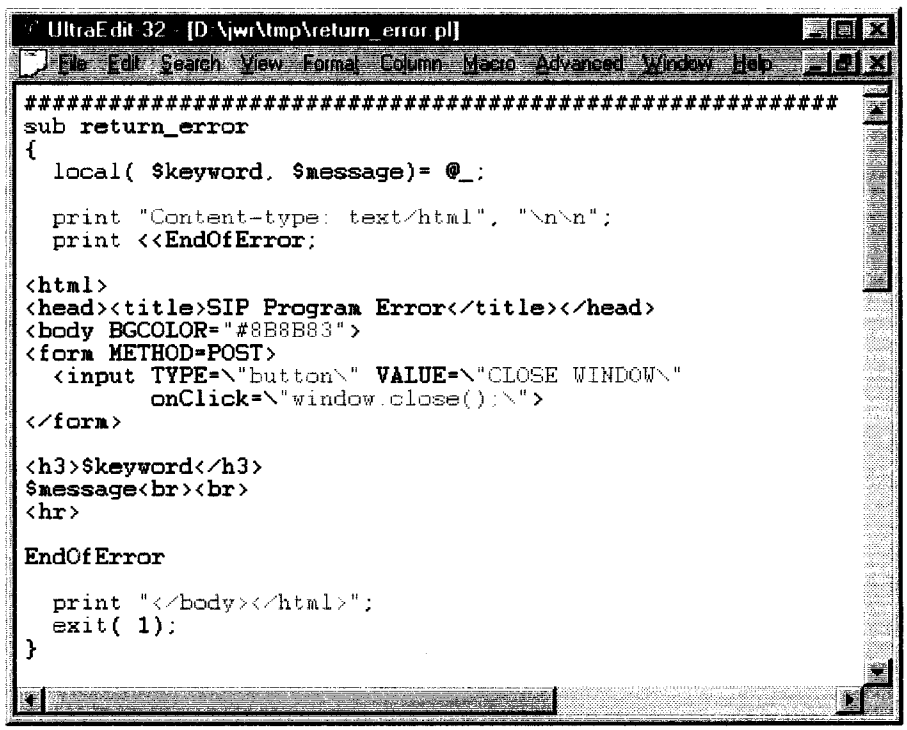

Figure 3.8: Perl Script

It is true that Perl is not one of the more readable languages. Once the Perl syntax and constructs are learned, however, Perl is very nice for CGI programming and text

processing. What may take many lines of C code will only take a few lines of Perl script, and there is much more power in manipulating text including HTML and JavaScript. Perl has good networking capabilities and allows communication between other programs and scripts. Perl is highly portable and readily available. "Perl is by far the most widely used language for CGI programming." [9].

3.7 Network Software Development

During software development it is important to justify which part of the software should run on the client machine and which part should run on the server. Of course this depends on the application. If an extensive program is required, like the SPICE engine, then the SPICE engine has to reside on the server because it would take a long time to download an entire SPICE engine every time it was needed. Also, configuration of a large program, like a SPICE engine, probably would need to be performed on the resident machine because of the complexity of the program.

The graphical analysis generation and the user interface could be implemented on the client machines using a Java applet or JavaScript, for example. If all of the processing was executed on the server, the server would not be able to respond to many multiple requests in a timely manner. On the other hand, if most of the processing is delegated to the client, the client might have to work too hard while the server would have the resources available to respond to multiple simultaneous user requests. These are important issues, and the performance of the server and the client is highly dependent on how much of the workload each has to perform.

Using the CGI for interactivity and using Java applets or JavaScript is quite different functionally. Applets are transferred through a network as bytecodes when requested and execution is performed entirely on the client machine that made the request. In CGI much less information has to be passed to the server. The server executes instructions based on the given information and sends the results back to the local machine that made the request. It is inherently slower to pass information through a network than it is to run a Java interpreter (Web browser) on a client machine and use the client's resources for computing.

Another consideration in the development of Internet network applications is what third party software could be used. There are many Java applets and other programs readily available that enhance the interactivity, capability and appearance of Internet applications and Web pages. For many tasks it is much easier and more efficient to use software that has been developed and just "glue" everything together with the HTML, JavaScript, and the Common Gateway Interface. Of course, someone has to write the software in the first place and by writing applets, the maximum control and requirements can be met for a certain task. If there is some software already developed to meet the requirements for part of a project, it is possible to include the third party software to make the Internet application more functional.

The SPICE Internet Package, for example, uses a software package called *gnuplot* for post processing of the SPICE analysis, and *netpbm* utilities for converting the *gnuplot* file output format to a gif image. Under development is a Java applet that will allow all of the graphical processing to be done on the client machine and interactive controls will

be made available using this Java technology. There are also commercial plotting packages written as Java applets that could be used to generate the graphical analysis as well as provide an interactive user interface for the graph itself.

Chapter 4

SPICE Internet Package (SIP)

4.1 Overview

The SIP software package is a SPICE simulation and analysis program implemented in the network environment. A unique feature of the SIP program is that it is operating system independent. Anyone that has access to the Internet and a Web browser, such as Netscape Navigator or MS Internet Explorer, can run a SPICE simulation and view the results graphically from anywhere in the world using any operating system.

4.2 SIP Implementation

It was determined that the graphical interface should have five windows. The first window a user sees is the login window in Figure 4.1. After the authentication is secured, the main window in Figure 4.2 appears in place of the login window. This is the main graphical user interface window, and it does not change. Based on the selections the user makes, the output is redirected to one of the other three windows—the

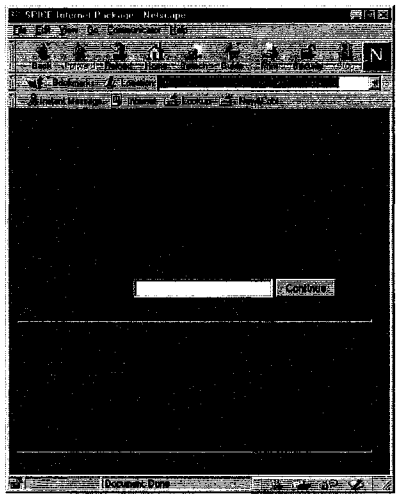

Figure 4.1: SIP Login

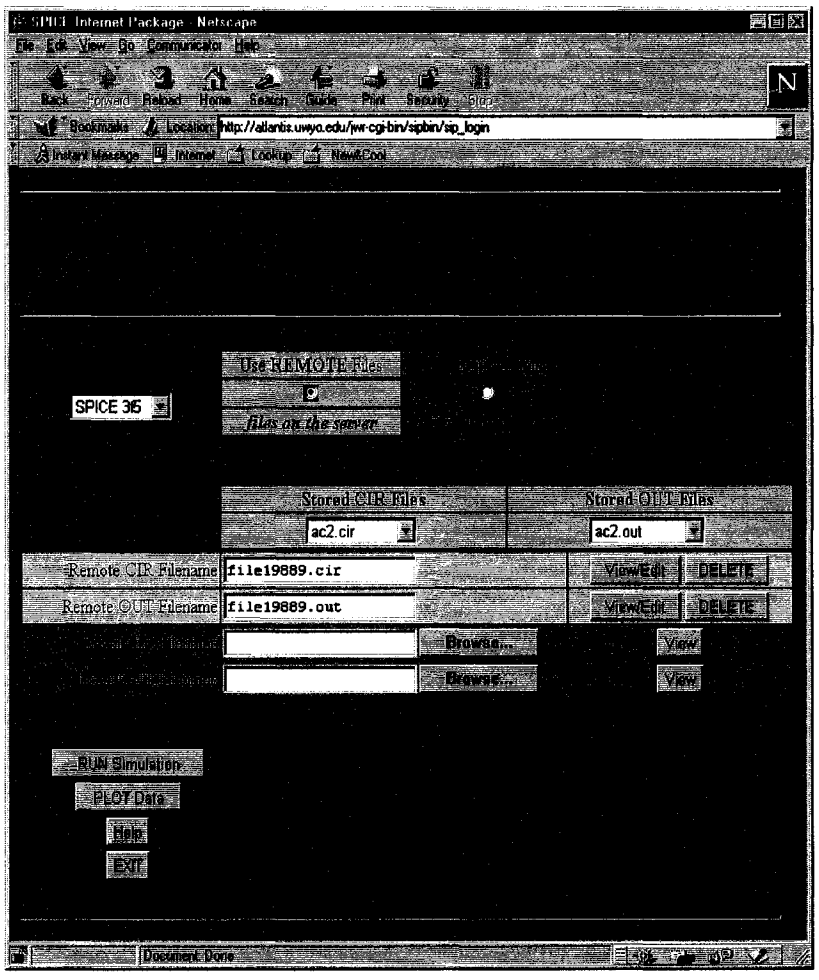

Figure 4.2: SIP Main Window

graphing options window in Figure 4.3, the output window in Figure 4.4, and the editing window shown in Figure 4.5.

In the initial stages of development it was necessary to determine what tasks should be delegated to the client machine and which the server should execute. This delegation directly affects the performance of the server and the responsiveness seen by the users on the client machines. Security issues, the implications of simultaneous users, licensing concerns, and the directory structure also had to be taken into consideration and these ideas are discussed.

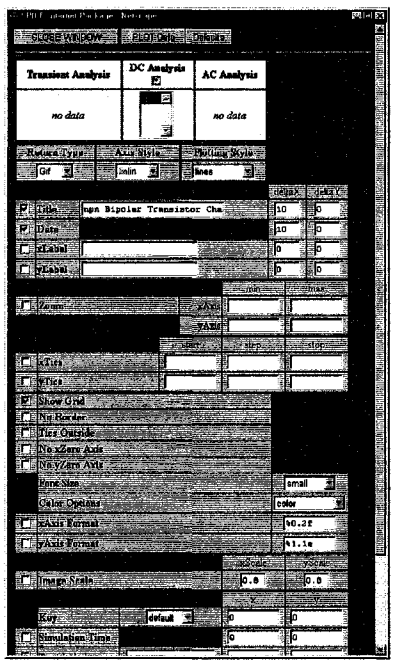

Figure 4.3: SIP Graphing Options

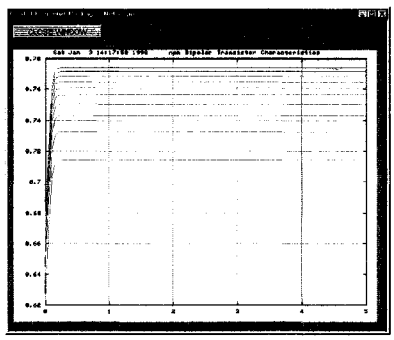

Figure 4.4: SIP Output

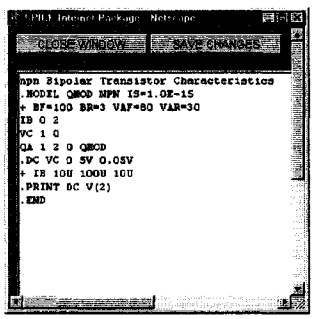

Figure 4.5: SIP Edit

4.2.1 SIP Client-Server Model

A network model of the SIP software package is shown in Figure 4.6. A server is configured to accept requests from web browsers through network connections. The server processes the request for SPICE simulation or analysis and returns the results to the requesting Web browser as an HTML document. In the case of the SPICE Internet Package it only makes sense to use CGI for the SPICE simulation because it would be impossible to send the SPICE engine through the network every time it was requested, and this would be extremely slow.

The SIP program currently incorporates CGI, Perl, HTML, and JavaScript. The graphical analysis is embedded in the HTML document as an image or returned as formatted text. Figure 4.6 shows the flow of information and the distinction from the client and the server.

A Java applet is being developed that will run on the client machine and give the user an interactive interface for manipulating the graphical analysis. This will shift more of the processing to the client machines so the server will not have to do as much work.

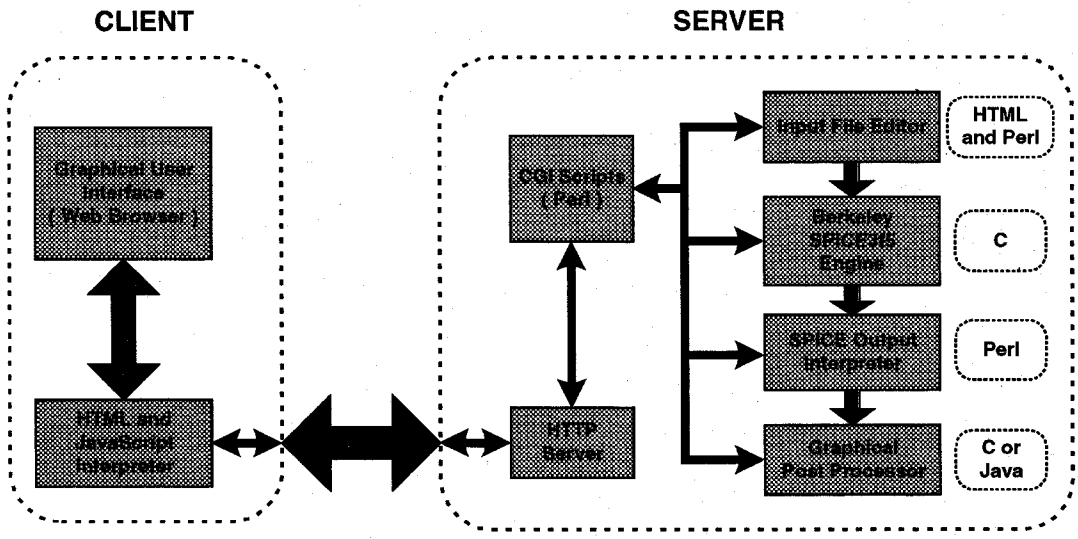

Figure 4.6: SIP Client-Server Architecture

4.2.2 Simultaneous Users

One of the advantages of the SPICE Internet Package is that multiple users can use the program at the same time through a network. This causes some problems when the same files are being modified by different users at the same time. The files that could be potentially modified by different users at the same time include the password file and log files. These files are updated as multiple users access the program. The Perl code in Figure 4.7 illustrates the mechanism used to allow simultaneous users. The idea of atomic instructions is used to lock the files with the *flock()* system function. The code that is responsible for locking and unlocking the file is shown as bold print in Figure 4.7. The argument *$LOCK_EX* given in the statement *flock(PFILE, $LOCK_EX)* is used so a process that encounters a locked file will wait for the file to be unlocked as opposed to returning with an error because the file was locked. This way two or three simultaneous

users will be queued waiting to access a file and it will appear seamless to the users. Each user will be allowed to modify the file in turn so it does not get corrupted.

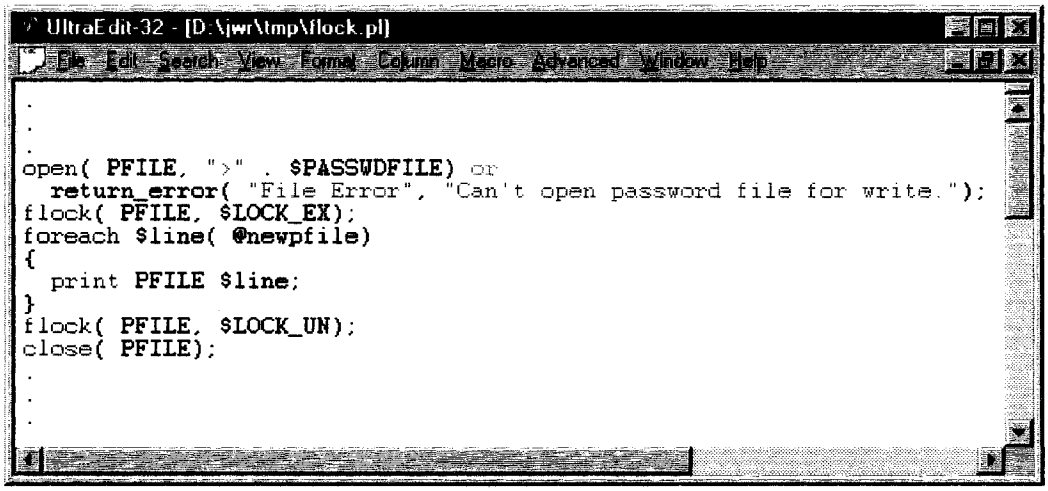

Figure 4.7: File Locking

4.2.3 Security and Logging

Because the SIP is accessible from the Internet, it is available to essentially the entire world. A password scheme is required to ensure only those with proper permission can use the program. Otherwise, the server running the SIP application would be potentially overworked, and no other users could use the system. A password file is maintained with the format shown in Figure 4.8. The fields are comma delimited, spaces around the commas are ignored, and lines that begin with a "#" are ignored as well. The entries are *password*, *username*, *directory*, *accesses*, and *lastlogin*. The *password* is required for a user to access the SIP program. The *username* is the name that appears on the Web page after the login screen at the top of the main window. The *directory* is used

to assign directories where files are saved in the filesystem for different users or groups of users. The *accesses* is the number of times a user has accessed the system, and the

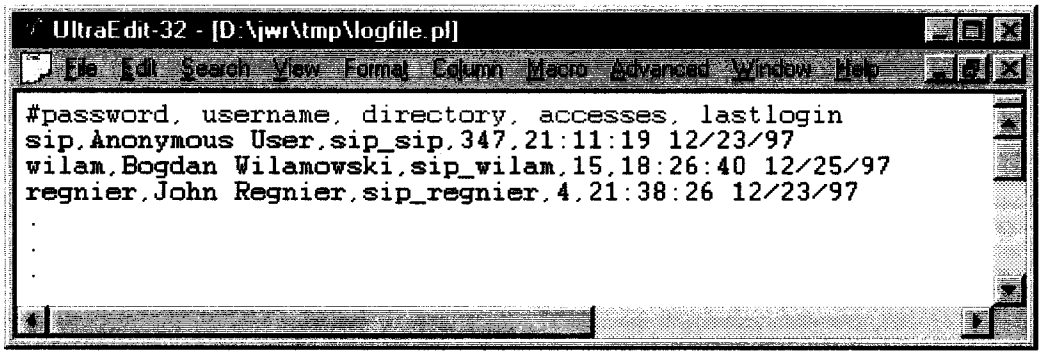

Figure 4.8: SIP Password File

lastlogin is the last time a user accessed the system. As indicated above, it is necessary to lock this file and the log file because simultaneous multiple users could potentially access the files at the same time.

To help prevent system attacks, the input that is solicited from the users is checked so that no harm can be done to the server. File names that are input from the user are checked for characters that would allow the user to get to other directories in the filesystem. For this reason characters like "..", and "/" are not allowed in the filenames. If a user tries to enter a filename with the parent ".." directory as part of the filename, an error message is generated.

A log file is updated every time a user logs into the SIP program or executes one of the commands for simulation or analysis. The format of the log file is shown in Figure 4.9. This file can be searched for statistical data about how much different aspects of the

SIP application are being used, or from where on the Internet the users are coming and the most heavily used time of day.

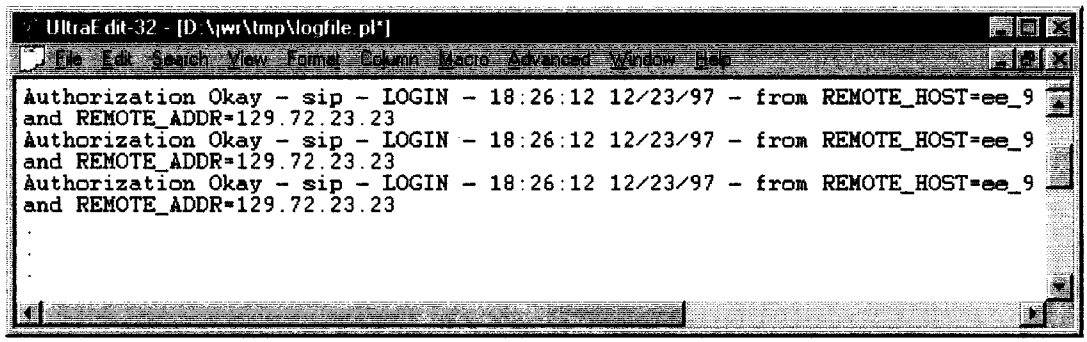

Figure 4.9: SIP Log File

A Perl script has been written to generate a report of the SIP usage. See Appendix C for the source code. The output is shown below. This information could be used for billing purposes or to investigate what aspects of the SIP program are being used most frequently.

```
SPICE Internet Package
Windows/NT Server Statistics
--------------------------------------------------------------------------------
              Total Hits: 1045
      Authorization Okay: 1041
    Authorization Failed:    4
--------------------------------------------------------------------------------
Actions Performed
--------------------------------------------------------------------------------
                   Login  144
           Run Simulation  458
            Edit Cir File  173
            Edit Out File    6
          Delete Cir File    6
          Delete Out File    4
                Plot Data  119
         Really Plot Data  116
                    Help   19
--------------------------------------------------------------------------------
Addresses and Hits
--------------------------------------------------------------------------------
128.196.28.52    cheetah.ece.arizona.edu                                    53
129.101.112.105  dialin095.csrv.uidaho.edu                                   4
129.101.112.91   dialin081.csrv.uidaho.edu                                  17
```

129.101.119.229	blackwidow.csrv.uidaho.edu	15
129.101.53.11	north.mrc.uidaho.edu	16
129.101.53.72	crux.mrc.uidaho.edu	1
129.101.71.148	PC001460.reshall.uidaho.edu	1
129.101.93.103		2
129.101.93.12	remote2.engboi.uidaho.edu	4
129.101.93.40	uipc8.engboi.uidaho.edu	18
129.101.93.84	PC13-212	4
129.186.39.192	adp192.adp.iastate.edu	1
129.72.16.14	ras-e014.uwyo.edu	1
129.72.16.18	ras-e018.uwyo.edu	15
129.72.16.22	ras-e022.uwyo.edu	19
129.72.16.25	ras-e025.uwyo.edu	1
129.72.23.14	olekmali.uwyo.edu	50
129.72.23.195	nn.uwyo.edu	587
129.72.23.197	atlantis.uwyo.edu	15
129.72.23.204	ee-5.uwyo.edu	9
129.72.23.209	EE-9	31
129.72.26.119		7
129.72.26.149		7
129.72.75.198	sci16.uwyo.edu	1
134.197.212.11	ppp11.scs.unr.edu	3
134.197.213.57	ppp57.scs.unr.edu	9
134.197.31.151	pclab1.physics.unr.edu	1
134.49.64.1	magma.aha.com	1
137.132.124.32	synergy.nus.edu.sg	17
143.106.17.210	clarice.demic.fee.unicamp.br	6
153.34.48.237	1Cust109.max2.las-vegas.nv.ms.uu.net	2
156.153.255.186	boirel1.hp.com	4
161.55.88.4	mike.pmel.noaa.gov	2
164.8.17.3	robin3.r.uni-mb.si	2
164.8.8.210	g219pc10.feri.uni-mb.si	10
166.55.5.234	usr12-dialup42.mix1.Sacramento.mci.net	2
166.55.9.227	usr28-dialup35.mix1.Sacramento.mci.net	8
192.114.123.186		4
192.9.25.21	saturn5.Sun.COM	5
194.201.189.41	man09.cogency.co.uk	5
198.3.123.20	matrix.dalsemi.com	2
203.241.133.226		11
203.252.100.130		4
203.65.167.240	host240.20365167.g.gcn.net.tw	2
204.134.209.126	cboi042p06.boi.micron.net	1
204.134.212.40	fwext.micron.com	23
204.134.213.27	cboi082p09.boi.micron.net	4
204.229.102.8		10
205.180.190.2	relay.cxo-x.dec.com	1
206.86.223.237	lame.durables.serpentine.com	3
207.175.199.134	hawaii-134.cchono.com	7
207.228.8.195		5
207.93.216.92		1
208.130.243.11	ts1-08.lar.cyberhighway.net	1
208.136.114.94	tc-1-94-rno.aci.net	1
209.19.190.72	cmos011p07.mos.micron.net	3
38.154.219.6	STAN	6

4.2.4 Licensing Concerns

The idea of using third party software and making it accessible to many users through Internet or Intranet networks is a new idea that is introduced through network programming. The spice3f5 program is distributed for free so allowing multiple users to

access this engine through the Internet is not a problem. For other commercial programs, however, this may be a breech of the software licensing agreement. The software licensing agreement should be read carefully and may need to be negotiated with the software developer if the software used is not your own.

4.3 SIP Features

The SIP program has incorporated the following features that make it user-friendly and productive for the engineer in circuit design with SPICE simulation and analysis.

4.3.1 Circuit and Netlist Entry

The files for simulation and analysis may be saved on the filesystem of the server or on the filesystem of the client machine. Descriptions of circuits are entered as text files. An edit window can be opened and the circuit description entered or viewed in the syntax of SPICE code. The circuit description may be Entered or modified in the case of remote files, but viewed only in the case of using files on a client machine. Files on a client machine can be entered and modified using another text entry program on the client machine, like "Notepad."

4.3.2 Simulation and Analysis

There is an option in the main window of the SIP graphical user interface that can be selected if the files to be used for simulation and graphical analysis are located on

the remote machine. If the files are on the client machine, then a file browser window will popup so the files can be entered into the form with a graphical user interface. When simulation is performed, the output from simulation is echoed to the output window of the SIP program and also saved to the output file specified for the remote simulation option. For the local simulation the output file cannot be saved on the client machine, but the output is still echoed to the output window of the SIP program. This data can be saved by selecting "File..SaveAs.." from the Web browser menu or by using a copy and paste with the system clipboard.

4.3.3 Graphical Analysis

The graphical analysis is performed based on the data that are in the given output file. As with the simulation, the output file may be specified on the remote machine or the local server machine. The data used for graphing is generated with the SPICE "`.PRINT <AC|DC|TRAN> ...`" statement. When the graphical analysis option is selected, the output file is searched, using a Perl script, and the graphical options window is generated dynamically. If DC or AC or Transient analysis is present in the output file, the graphical options window will show the variables that may be plotted for each type of analysis.

4.3.4 Graphical Analysis Customization

There are many options that can be modified using the form in the graphical configuration window. The analysis output can be specified as a gif image, or raw text

containing the data points. This way the data can be imported to some other high-end graphing package to create the plots, like Microsoft Excel. Data can be plotted in several ways using the SIP graphical options. Some options use lines, symbols, both lines and symbols or dots. Some other options include zoom, scale, selecting variables to plot, log scale, tic marks, grid, border, color options for the graph, a key, and text labels for title, data, simulation time, xlabel, ylabel, and text labels. Basically all of the options available with the *gnuplot* package have been added to this graphical configuration form.

4.4 SIP Server Requirements

The current versions of SIP are written for the Unix operating system and the MS Windows/NT operating system. There are few differences between the two versions because of the operating system independence of the programming languages being used. Perl, for example, is available for MS Windows/NT as well as the Unix operating system, and JavaScript and HTML are totally platform independent. The image generation is currently done with the *gnuplot* program and *netpbm* utilities. In a later version, the graphical processing may be moved to the client machine using a Java applet.

4.5 SIP Directory Structure and Files

The directory structure and listing of the files required to make the SIP application operational are shown below in Figure 4.10. The *index.html* file is the home page of the SIP application and brings up the login window. The *graphics* directory is

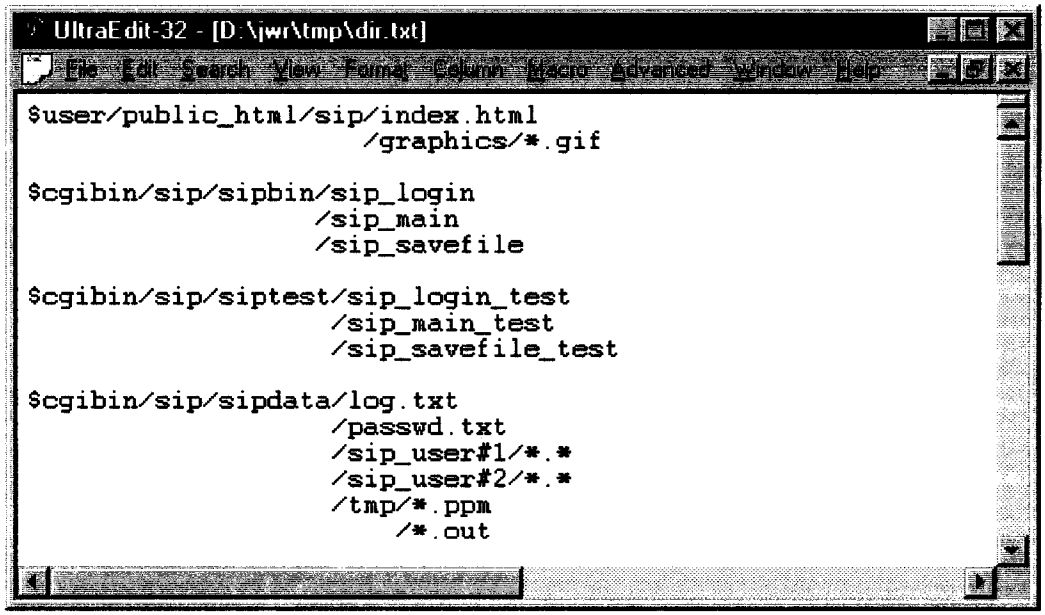

Figure 4.10: SIP Files

where the files generated by the graphical analysis tool are temporarily stored. These files need to be accessible from other Web browsers so they are stored under the *public_html* directory where the permissions are set such that anyone can look at the files. These files are removed from the server filesystem on a regular basis by means of a script or batch file that is executed by a scheduler of the operating system.

There are three Perl script files: *sip_login*, *sip_main*, and *sip_savefile*. The first one, *sip_login*, is executed when the user enters a password. This file generates the main login window if authorization is secured. The second script is *sip_main*. This script does most of the work. When the user selects any of the options from the main window, this script is executed. The last script, *sip_savefile*, is used to save the changes made to a file on the server. This script is called when the user selects "Save Changes" from one of the edit windows.

The *log.txt* and *passwd.txt* are used to store the record of users who use the system and to support the security of the application. The *sip_user* directories store the files for each user in a different directory. The *tmp* directory stores temporary files that the SIP application creates intermediately. And finally the *siptest* directory is used for debugging and revision purposes. A copy of the program files are modified in this directory and tested so the files in the *sipbin* directory will always work properly and the SIP application will always be operational. After the modifications to the *siptest* files have been made and the test files are working properly, they are copied to the *sipbin* directory.

4.6 SIP Example

The following example illustrates the capabilities of the SIP program with screen shots illustrating the graphical user interface of the program as it is running.

4.6.1 MOS Inverter

As an example of the use of the SIP program, consider the CMOS Inverter in Figure 4.11 and the SPICE3 model of the circuit in Figure 4.12.

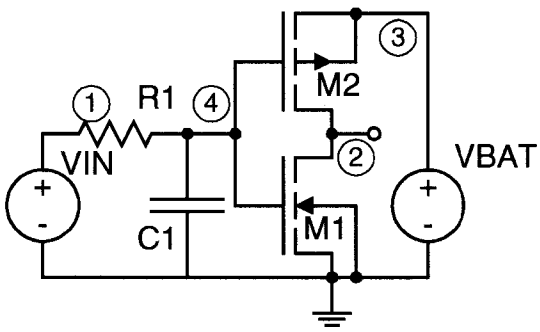

Figure 4.11: CMOS Inverter

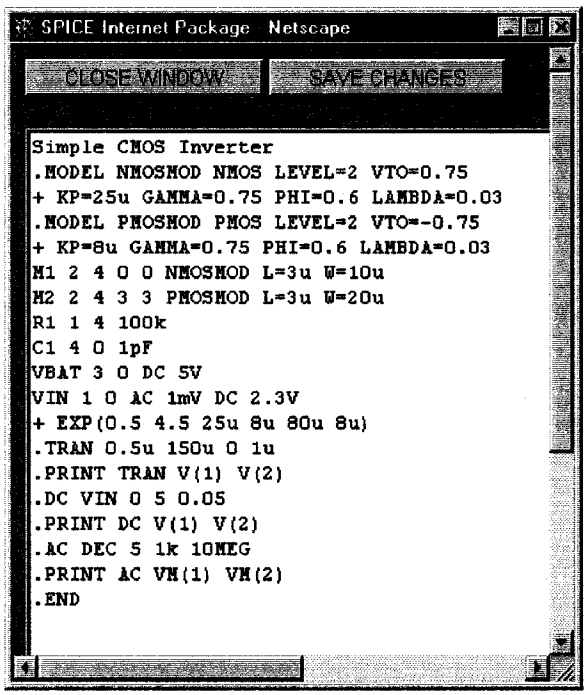

Figure 4.12: SPICE3 Model

Notice in the SPICE code there are commands for the SPICE simulator to analyze the dc sweep, the ac analysis, and the transient response of the circuit and to print the output of V(1) and V(2). The graphical user interface is the same as the one shown in Figure 4.2. It is necessary to select the radio button for ***REMOTE*** or ***LOCAL*** files

depending on which you are using. The button is shown in Figure 4.2. This decision specifies whether the simulation and graphing programs will use the remote or local file names that you entered. If you use local files, you have to save the output from simulation in a file on your local machine and then enter that name in the ***Local OUT Filename*** edit box in order to plot the data.

To enter a SPICE input file, select the filename from the CIR drop down box or type the name into the ***CIR Filename*** edit box and then press the ***VIEW/EDIT*** button. After entering or modifying the input file press ***SAVE CHANGES*** in the edit window to save the changes to the file. After entering the input file and saving it, the simulation is then run by selecting the ***Run Simulation*** button. The output from the simulation is displayed in the output window as shown in Figure 4.13. If remote files were used, the output is also saved to the file in the ***Remote OUT Filename*** edit box automatically.

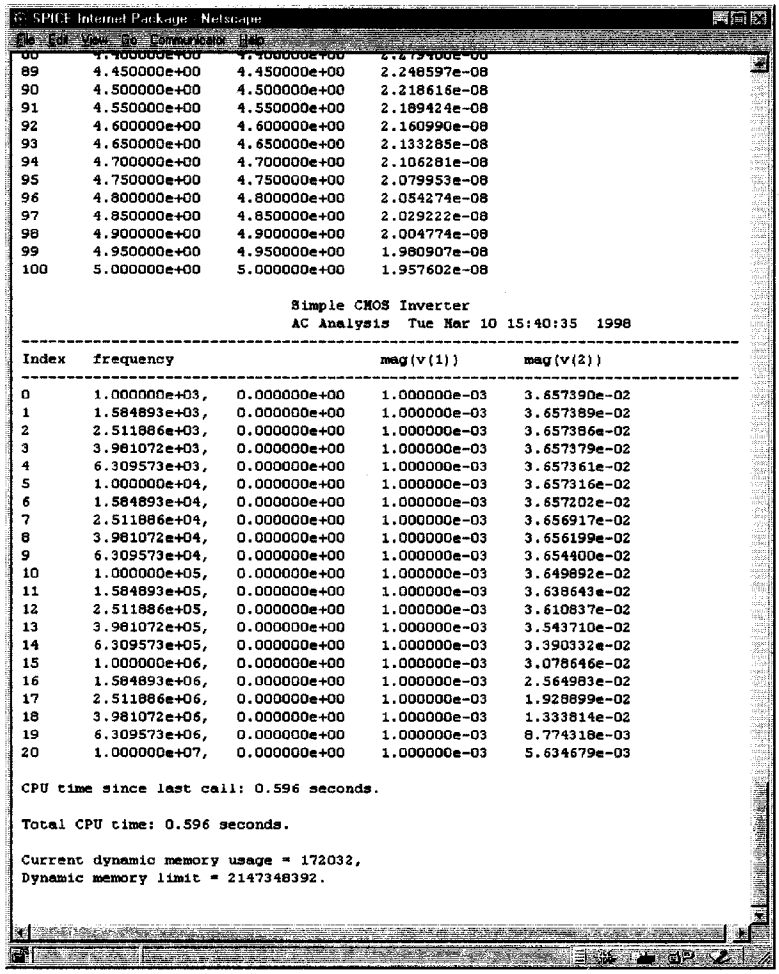

Figure 4.13: Simulation Output

Next a graph of the output data is generated by pressing the ***PLOT Data*** button. A window will open allowing you to select the variables you want to plot and other customization variables as shown in Figure 4.14.

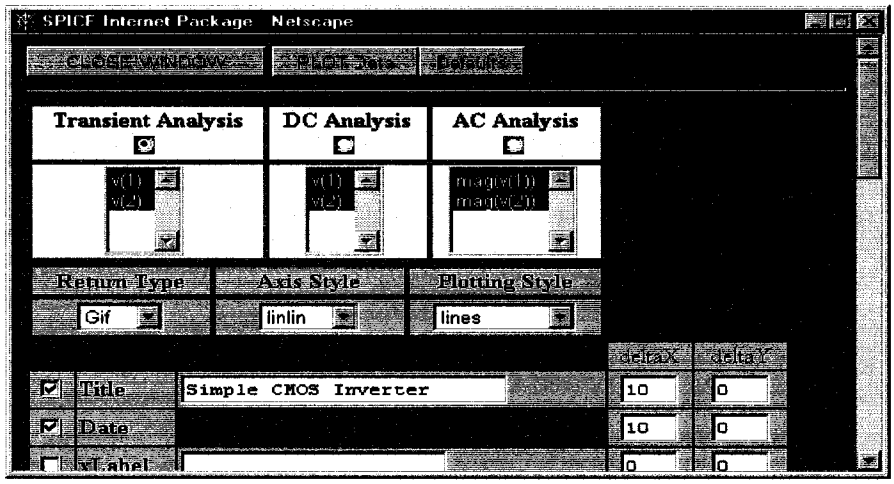

Figure 4.14: Graphing Options

Select ***PLOT Data*** from the graphing options window and the analysis is returned to the output window as shown in Figure 4.15. To plot the dc analysis, select the ***DC Analysis*** button and press ***PLOT Data*** again. The graph in Figure 4.16 is generated. The same procedure creates the ac analysis in Figure 4.17. The AC analysis plot in Figure 4.17 was created using semi-log axes by selecting ***loglin*** from the ***Axis Style*** ddropdown box in the graphing options window. To generate the formatted space delimited data shown in Figure 4.18 the ***Return Type*** dropdown box is changed to ***Text*** instead of ***Gif***.

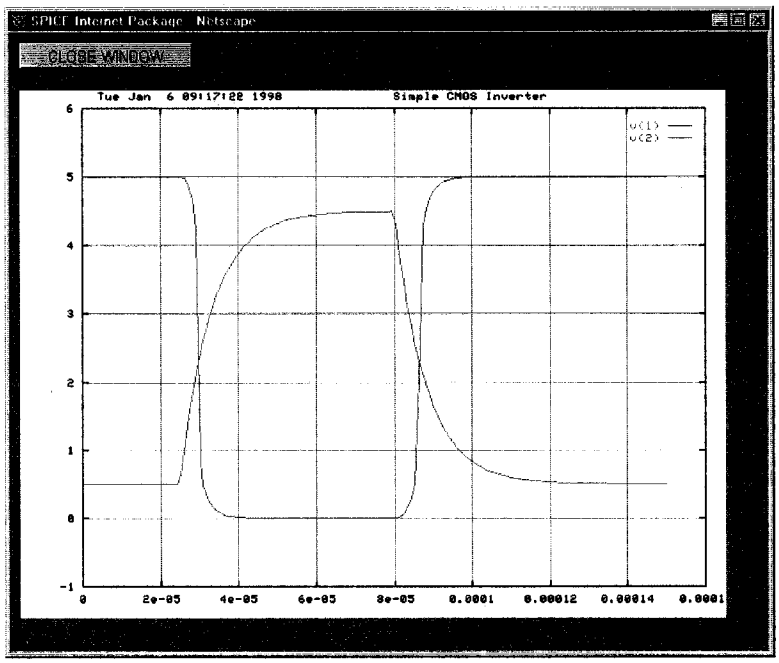

Figure 4.15: Transient Analysis

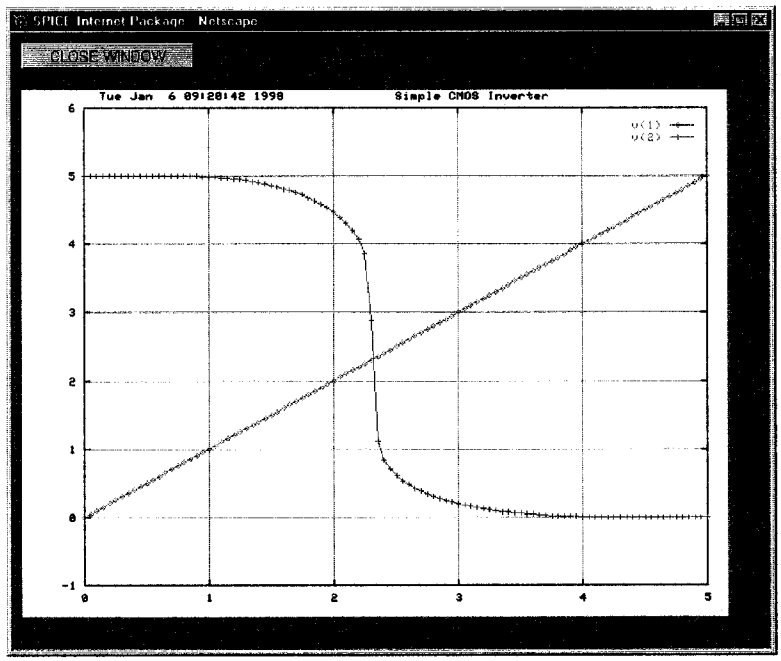

Figure 4.16: DC Analysis

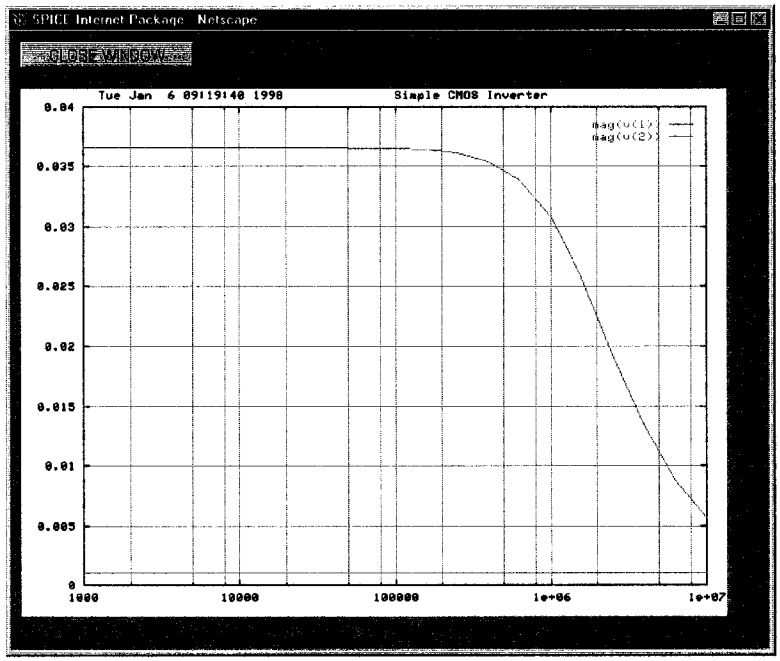

Figure 4.17: AC Analysis

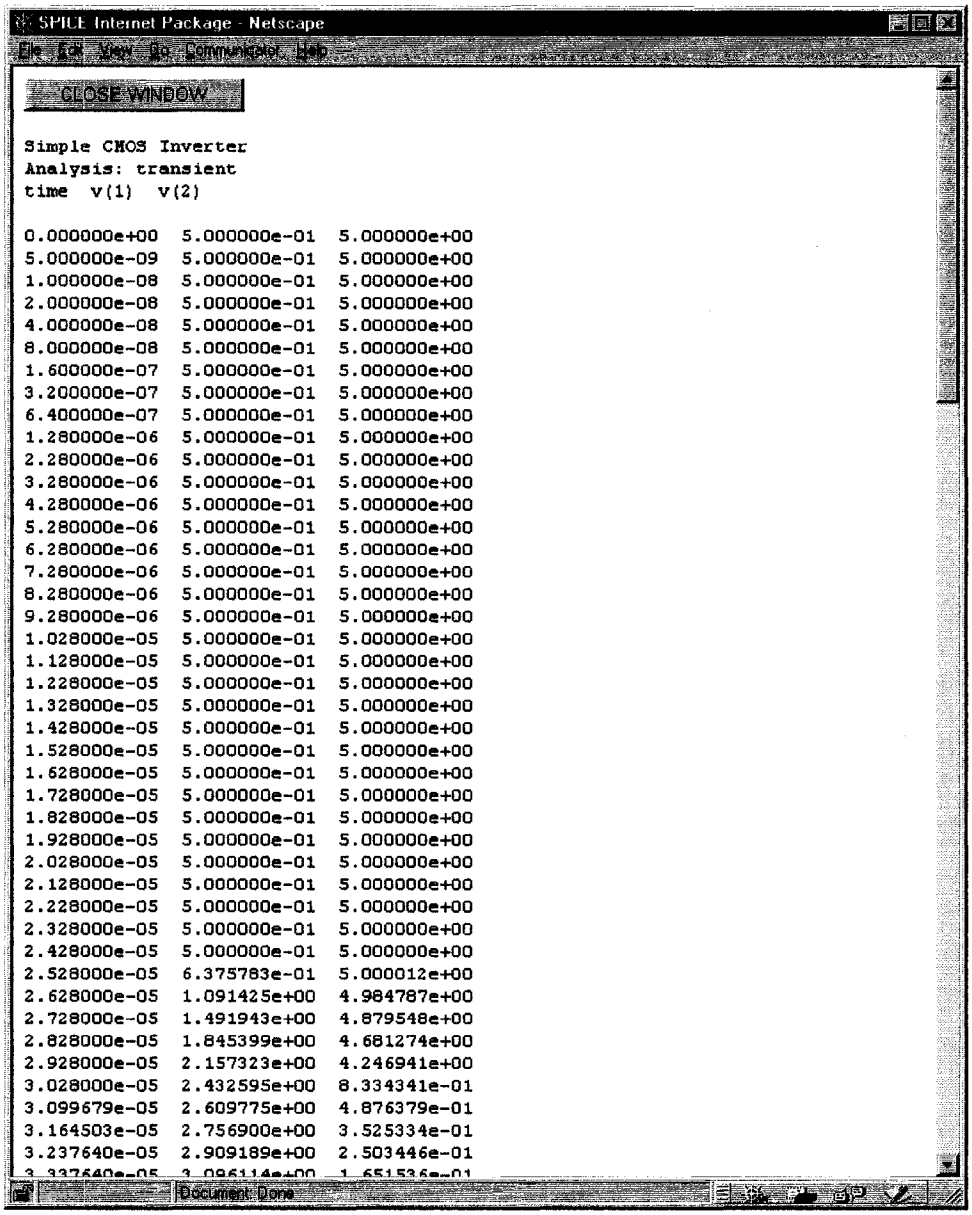

Figure 4.18: Space Delimited Data Analysis

To close all the windows and return to the original web browser window, press the ***EXIT SIP*** button. There is also a ***HELP*** button describing how to use the SIP program. See Appendix D for more details.

4.7 Benefits of a Networked SPICE Program

The SPICE Internet Package runs independently of any particular operating system. The same graphical user interface is seen on a Macintosh or a PC or a Unix machine. Only one copy of the SPICE program is required so less installation and configuration is therefore needed. Remote access allows users to use the program from anywhere that an Internet connection is available, including school or home or business. SPICE files can be saved on the server or on the client machines.

4.8 Alternatives Implementations

An alternative to using the server resources for the graphical post-processing is using the client resources. This would be desirable if the client has the computing power to run the Java interpreter more efficiently than the server. An alternative graphical user interface using the Java programming language would not only shift more of the work to the client machine, but also the GUI might be developed with more functionality with this method.

Chapter 5

Conclusion

5.1 Conclusion

The Internet and Intranet networks are more capable and efficient than ever before. Most companies, businesses, educational institutions, and research institutions use these networks to some extent on a regular basis. The technology is available for developing software applications that run on remote machines independent of the operating systems being used. The purpose of this thesis has been to show how computer-aided engineering tools can be used through networks, and to outline the many advantages that come with these networked applications. A tool developed for use on a network can be used remotely, on a variety of operating systems, on a pay-per-use basis, with little or no installation and configuration.

The SPICE Internet Package that was presented in Chapter 4 is just an example of how SPICE simulation and analysis can be accomplished using the Internet. Several features make the SPICE Internet Package a desirable program for computer-aided engineering and design. Only one copy of the SPICE engine needs to be installed and configured on a server machine. Using a Web browser, multiple users can

simultaneously access the SPICE engine through remote network connections at their convenience. Files can be saved on the server, or local files can be seamlessly uploaded for simulation or analysis. The graphical analysis feature provides customization of graphical output, or the output can be specified as space delimited text for input into other graphing packages. These features make the SPICE Internet Package, and possibly other networked applications, convenient tools for the engineering desktop.

5.2 Further Study

With the Internet growing at such a fast pace the technologies that enable us to tap the resources of the Internet are changing and being enhanced almost as fast. For example, Java and JavaScript are still growing and being developed with added features and functionality. These new changes may provide more efficient and better methods of Internet programming that could further enhance the quality of computer-aided engineering tools designed for use through the Internet and Intranet networks.

References

[1] Linda Geppert. IEEE Spectrum. January 1997.

[2] Bogdan M. Wilamowski, and Richard C. Jaeger. Computerized Circuit Analysis Using SPICE Programs. McGraw-Hill, 1997.

[3] Gary Cornell, and Cay S. Horstmann. Core Java. SunSoft Press, 1997.

[4] David Flanagan. Java in a Nutshell. O'Reilly & Associates, Inc., 1997.

[5] Arman Danesh. Teach Yourself JavaScript 1.1 in a Week. Sams.net Publishing, 1996.

[6] Fah-Chun Cheong. Internet Agents: Spiders, Wanderers, Brokers, and Bots. New Riders Publishing, 1996.

[7] Mark G. Sobell. A Practical Guide To The Unix System. The Benjamin/Cummings Publishing Company, Inc., 1995.

[8] Larry Wall, Tom Christiansen, and Randal L. Schwartz. Programming Perl. O'Reilly & Associates, Inc., 1996.

[9] Shishir Gundavaram. CGI Programming on the World Wide Web. O'Reilly & Associates, Inc., 1996.

[10] Eric F. Johnson. Cross-Platform Perl. M&T Books, 1996.

Appendix

Appendix A: SIP Help
Appendix B: SIP Main Perl Script (sip_main)
Appendix C: SIP Main Perl Script (sip_login)
Appendix D: SIP Main Perl Script (sip_savefile)
Appendix E: SIP Log File Perl Script
Appendix F: SIP Home Page HTML

Appendix A
SIP Help

Assume the "Use REMOTE Files" radio button is selected (the default). If the "Use LOCAL Files" radio button is selected the output is not written to the "Local OUT Filename" but you can do this using copy and paste to a text editor on your machine.

"RUN Simulation"

Simulates the file specified in the "Remote CIR Filename" text box and prints the output in another output window. The output is also written to the "Remote OUT Filename" file.

"Plot Data"

Prepares to plot the data that is located in the "Remote OUT Filename" text box and brings up the Plot Options window. The data in the "Remote OUT Filename" is generated by the "RUN Simulation" command. The CIR file should have the SPICE ".PRINT <AC|DC|TRAN> ..." option specified. This is how the data is generated that will be plotted.

"View/Edit"

The CIR or OUT file is opened in the SIP Edit window. Enter the SPICE circuit description here and save the changes.

"DELETE"

Removes the file specified in the corresponding text box from the filesystem.

Appendix B
SIP Main Perl Script (sip_main)

```
#!/opt/LWperl/bin/perl

################################################################################
## Program:  SPICE Internet Package perl script
##
## Developed by:
##    John Regnier
##    Bogdan Wilamowski
##    Department of Electrical Engineering
##    University of Wyoming
##
##    Last Revision: 18oct97
##    Contact: wilam@uwyo.edu or regnier@uwyo.edu
################################################################################

use lib '/export/home2/regnier/lib/perl/lib';
use lib '/export/home2/regnier/lib/perl/arch';
use CGI_Lite;

$ENV{LD_LIBRARY_PATH}= "/usr/openwin/lib";
$ENV{PATH}= $ENV{PATH} . ":/export/home2/regnier/spice3/bin:/usr/bin";

#parse the form data
$form= new CGI_Lite;
$form->set_directory( "/export/home2/regnier/cgi-bin/sipdata/tmp") or return_error( "File
Error", "Directory does not exist for CGILite.");
$form->set_buffer_size( 1024);
$form->set_platform( "Unix");
$form->set_file_type( "handle");  #or "handle"
%theform= $form->parse_form_data();
$form->close_all_files;

#set up temporary files
$process_id= $$;
$GNUPLOT= "/export/home2/regnier/gnuplot/bin/gnuplot";
$PPMTOGIF= "/export/home2/regnier/netpbm/ppmtogif";
$OUTPUT_PPM= join( "", "/export/home2/regnier/cgi-bin/sipdata/tmp/", $process_id, ".p");
#was .ppm but filename too long?
$DATAFILE= join( "", "/export/home2/regnier/cgi-bin/sipdata/tmp/", $process_id, ".txt");
$TMPPLOTFILE= join( "", "/export/home2/regnier/public_html/sip/sipdata/", $process_id,
".gif");

$LOGYES= 0;  #send email to john regnier
$WEBMASTER= "regnier\@uwyo\.edu";
$HELPFILE= "/export/home2/regnier/cgi-bin/sipdata/help.txt";
$LOGFILE= "/export/home2/regnier/cgi-bin/sipdata/log.txt";
$SPICE3PATH= "/export/home2/regnier/spice3/bin/spice3";

#input from the /sp/index.html page
$TFpassword= $theform{'password'};
$TFremotelocal= $theform{'remotelocal'};
$TFremotecirfile= $theform{'remotecirfile'};
$TFremoteoutfile= $theform{'remoteoutfile'};
$TFlocalcirfile= $theform{'localcirfile'};
$TFlocaloutfile= $theform{'localoutfile'};
$TFspiceversion= $theform{'spiceversion'};
$TFhiddencommand= $theform{'hiddencommand'};

#from plotoptions window
$TFanalysistype= $theform{'analysistype'};
@TFtransientanalysis= $form->get_multiple_values( $theform{'transientanalysis'});
@TFdcanalysis= $form->get_multiple_values( $theform{'dcanalysis'});
@TFacanalysis= $form->get_multiple_values( $theform{'acanalysis'});

#return type
$TFreturntype= $theform{'returntype'};  #gif|PostScript
#axis style
```

```
$TFaxisstyle= $theform{'axisstyle'};  #linlin|linlog|loglin|loglog
#axis label
$TFxaxislabelbox= $theform{'xaxislabelbox'};  #checked|?
$TFxaxislabeltext= $theform{'xaxislabeltext'};  #text
$TFxaxislabeldeltaxtext= $theform{'xaxislabeldeltax'};  #number
$TFxaxislabeldeltaytext= $theform{'xaxislabeldeltay'};  #number
$TFyaxislabelbox= $theform{'yaxislabelbox'}; #checked|?
$TFyaxislabeltext= $theform{'yaxislabeltext'};  #text
$TFyaxislabeldeltaxtext= $theform{'yaxislabeldeltax'};  #number
$TFyaxislabeldeltaytext= $theform{'yaxislabeldeltay'};  #number
#zero axis
$TFnoxzeroaxisbox= $theform{'noxzeroaxisbox'};  #checkd|?
$TFnoyzeroaxisbox= $theform{'noyzeroaxisbox'};  #checked|?
#title
$TFtitlebox= $theform{'titlebox'};  #checked|?
$TFtitletext= $theform{'titletext'};  #checked|?
$TFtitledeltax= $theform{'titledeltax'};  #number
$TFtitledeltay= $theform{'titledeltay'};  #number
#date
$TFdatebox= $theform{'datebox'};  #checked|?
$TFdatedeltax= $theform{'datedeltax'};  #checked|?
$TFdatedeltay= $theform{'datedeltay'};  #checked|?
#plotting style
$TFplottingstyle= $theform{'plottingstyle'};  #points|lines|linespoints|dots|steps|impulses
#zoom
$TFzoomoptionsbox= $theform{'zoomoptionsbox'};  #checked|?
$TFxmin= $theform{'xmin'};  #number
$TFxmax= $theform{'xmax'};  #number
$TFymin= $theform{'ymin'};  #number
$TFymax= $theform{'ymax'};  #number
#image size
$TFimagesizebox= $theform{'imagesizebox'};  #checked|?
$TFimagesizex= $theform{'imagesizex'};  #number
$TFimagesizey= $theform{'imagesizey'};  #number
#key
$TFkeyselect= $theform{'keyselect'};  #default|hide|moveto
$TFkeyx= $theform{'keyx'};  #number
$TFkeyy= $theform{'keyy'};  #number
#grid and border
$TFgridbox= $theform{'gridbox'};  #checked|?
$TFborderbox= $theform{'borderbox'};  #checked|?
#tics
$TFxticsbox= $theform{'xticsbox'};  #checked|?
$TFxticsstart= $theform{'xticsstart'};  #number
$TFxticsstep= $theform{'xticsstep'};  #number
$TFxticsstop= $theform{'xticsstop'};  #number
$TFyticsbox= $theform{'yticsbox'};  #checked|?
$TFyticsstart= $theform{'yticsstart'};  #number
$TFyticsstep= $theform{'yticsstep'};  #number
$TFyticsstop= $theform{'yticsstop'};  #number
$TFticsoutsidebox= $theform{'ticsoutsidebox'};  #checked|?
#axis labels
$TFformatxlabelsbox= $theform{'formatxlabelsbox'};  #checked|?
$TFformatxlabels= $theform{'formatxlabels'};  #text
$TFformatylabelsbox= $theform{'formatylabelsbox'};  #checked|?
$TFformatylabels= $theform{'formatylabels'};  #text
#font size
$TFfontsizeselect= $theform{'fontsizeselect'};  #small|medium|large
#color options
$TFcolorselect= $theform{'colorselect'};  #color|gray|monochrome
#time stamp
$TFtimestampbox= $theform{'timestampbox'};  #checked|?
$TFtimestampx= $theform{'timestampx'};  #text
$TFtimestampy= $theform{'timestampy'};  #text
#text label
$TFlabel1box= $theform{'label1box'};  #checked|?
$TFlabel1text= $theform{'label1text'};  #text
$TFlabel1x= $theform{'label1x'};  #number
$TFlabel1y= $theform{'label1y'};  #number
$TFlabel2box= $theform{'label2box'};  #checked|?
$TFlabel2text= $theform{'label2text'};  #text
$TFlabel2x= $theform{'label2x'};  #number
```

```
$TFlabel2y= $theform{'label2y'};  #number
#arrow label
$TFarrow1box= $theform{'arrow1box'};  #checked|?
$TFarrow1fromx= $theform{'arrow1fromx'};  #number
$TFarrow1fromy= $theform{'arrow1fromy'};  #number
$TFarrow1tox= $theform{'arrow1tox'};  #number
$TFarrow1toy= $theform{'arrow1toy'};  #number
$TFarrow2box= $theform{'arrow2box'};  #checked|?
$TFarrow2fromx= $theform{'arrow2fromx'};  #number
$TFarrow2fromy= $theform{'arrow2fromy'};  #number
$TFarrow2tox= $theform{'arrow2tox'};  #number
$TFarrow2toy= $theform{'arrow2toy'};  #number


#setup filenames based upon local/remote files given
if( $TFremotelocal eq "remote")
{
  $CIRFILEORIG= $TFremotecirfile;
  $CIRFILEHIDE= "/sip_$TFpassword/" . $TFremotecirfile;
  $CIRFILE= "/export/home2/regnier/cgi-bin/sipdata/sip_$TFpassword/" . $TFremotecirfile;
  $OUTFILEORIG= $TFremoteoutfile;
  $OUTFILEHIDE= "/sip_$TFpassword/" . $TFremoteoutfile;
  $OUTFILE= "/export/home2/regnier/cgi-bin/sipdata/sip_$TFpassword/" . $TFremoteoutfile;
}
else
{
  $CIRFILEORIG= $TFlocalcirfile;
  $CIRFILEHIDE= "/tmp/" . $TFlocalcirfile;
  $CIRFILE= "/export/home2/regnier/cgi-bin/sipdata/tmp/" . $TFlocalcirfile;
  $OUTFILEORIG= $TFlocaloutfile;
  $OUTFILEHIDE= "/tmp/" . $TFlocalcirfile;
  $OUTFILE= "/export/home2/regnier/cgi-bin/sipdata/tmp/" . $TFlocaloutfile;
}

#security measures so files only written in users directory
if( $CIRFILEORIG =~ /\// || $OUTFILEORIG =~ /\//)
{
  return_error( "File Error", "Illegal character \"/\" in filename(s).");
}

logusage();


SWITCH:
{
  #-----------------------------------------------------
  if( $TFhiddencommand eq "runsimulation")
  {
    #check that CIR exists and can be read
    -e $CIRFILE or return_error( "File Error", "The file $CIRFILEORIG does not exist.");
    -r $CIRFILE or return_error( "File Error", "The file $CIRFILEORIG is not readable.");
    -T $CIRFILE or return_error( "File Error", "The file $CIRFILEORIG is not a text file.");

    #check program name and run simulation
    if( $TFspiceversion eq "SPICE 3f5")
    {
      if( $TFremotelocal eq "remote")
      {
        open( OUT, ">" . $OUTFILE) or return_error( "File Error", "The file $OUTFILEORIG can
not be written.");
      }

      open MYPIPE, "spice3 -b -n \"$CIRFILE\" 2>&1 |"
        or return_error( "System Error", "Can't Fork: $!");

      @output= ();
      while( <MYPIPE>)
      {
        if( $TFremotelocal eq "remote")
        {
          print OUT;
        }
```

```
        push( @output, $_);
      }
      close( OUT);
    }
    else
    {
      return_error( "Not Implemented", "The program $TFspiceversion is not available at this
time.");
    }

    #display results
    print "Content-type: text/html\n";
    print "Pragma: no-cache\n\n";
    print htmltop();
    print "<form METHOD=POST>\n";
    print "  <input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\"
onClick=\"window.close();\">\n";
    print "</form>\n";
    print "<b>Simulation Results:</b><br>\n";
    print "<pre>\n";
    foreach $line ( @output)
    {
      print $line;
    }
    print "</pre>\n";
    print htmlbottom();

    last SWITCH;
  }
  #-------------------------------------------------
  if( $TFhiddencommand eq "plotdata")
  {
    local @transientdata= ();  #holds variables and Transient data
    local @dcdata= ();  #holds variables and DC data
    local @acdata= ();  #holds variables and AC data
    local %transientmisc= ();  #holds circuit name and general data
    local %dcmisc= ();  #holds circuit name and general data
    local %acmisc= ();  #holds circuit name and general data

    #check OUT file exists and can be read
    -e $OUTFILE or return_error( "File Error", "The file $OUTFILEORIG does not exist.");
    -r $OUTFILE or return_error( "File Error", "The file $OUTFILEORIG is not readable.");
    -T $OUTFILE or return_error( "File Error", "The file $OUTFILEORIG is not a text file.");

    @wholeoutfile= ();
    readwholeoutfile( \@wholeoutfile);

    #scan file for data using Transient & DC & AC
    getdata( "Transient", \@wholeoutfile, \@transientdata, \%transientmisc);
    getdata( "DC", \@wholeoutfile, \@dcdata, \%dcmisc);
    getdata( "AC", \@wholeoutfile, \@acdata, \%acmisc);

    #now send back the PrintOptions page
    print "Content-type: text/html\n";
    print "Pragma: no-cache\n\n";
    print htmltop();

    print "<form ENCTYPE=\"multipart/form-data\" ACTION=\"/jwr-cgi-bin/sipbin/sip_main\"
METHOD=\"POST\"
          NAME=\"sip\" TARGET=\"outputwindow\">\n";

    print "<input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\"
onClick=\"window.close();\">\n";
    print "<input TYPE=\"button\" VALUE=\"PLOT Data\" onclick=\"this.form.submit();\">";
    print "<input TYPE=\"reset\" VALUE=\"Defaults\">\n";

    print "<input TYPE=\"hidden\" VALUE=\"$TFpassword\" NAME=\"password\">\n";
    print "<input TYPE=\"hidden\" VALUE=\"reallyplotdata\" NAME=\"hiddencommand\">\n";

    print "<input TYPE=\"hidden\" VALUE=\"$TFremotelocal\" NAME=\"remotelocal\">\n";
    print "<input TYPE=\"hidden\" VALUE=\"$TFremoteoutfile\" NAME=\"remoteoutfile\">\n";
    print "<input TYPE=\"hidden\" VALUE=\"$TFlocaloutfile\" NAME=\"localoutfile\">\n";
```

```
    #print out all the HTML for plotoptionswindow
    printhtmloptions();

    #finish the html
    print "</form>\n";
    print htmlbottom();

    last SWITCH;
  }
  #-------------------------------------------------
  if( $TFhiddencommand eq "reallyplotdata")
  {
    local @transientdata= ();  #holds variables and Transient data
    local @dcdata= ();  #holds variables and DC data
    local @acdata= ();  #holds variables and AC data
    local %transientmisc= ();  #holds circuit name and general data
    local %dcmisc= ();  #holds circuit name and general data
    local %acmisc= ();  #holds circuit name and general data

    #check OUT file exists and can be read
    -e $OUTFILE or return_error( "File Error", "The file $OUTFILEORIG does not exist.");
    -r $OUTFILE or return_error( "File Error", "The file $OUTFILEORIG is not readable.");
    -T $OUTFILE or return_error( "File Error", "The file $OUTFILEORIG is not a text file.");

    @wholeoutfile= ();
    readwholeoutfile( \@wholeoutfile);

    #scan file for data using Transient & DC & AC
    if( $TFanalysistype eq "transient")
    {
      getdata( "Transient", \@wholeoutfile, \@transientdata, \%transientmisc);
      gnuplotdata( \@transientdata, \%transientmisc);
    }
    elsif( $TFanalysistype eq "dc")
    {
      getdata( "DC", \@wholeoutfile, \@dcdata, \%dcmisc);
      gnuplotdata( \@dcdata, \%dcmisc);
    }
    elsif( $TFanalysistype eq "ac")
    {
      getdata( "AC", \@wholeoutfile, \@acdata, \%acmisc);
      gnuplotdata( \@acdata, \%acmisc);
    }
    else
    {
      return_error( "No Data", "Select analysis type button and use SPICE '.print' statement
to generate data.");
    }

    last SWITCH;
  }
  #-------------------------------------------------
  if( $TFhiddencommand eq "remotecirfile")
  {
    unless( $TFremotelocal eq "remote")
    {
      return_error( "File Error", "Select the REMOTE FILES radio button.");
    }
    #check CIRFILE exists and can be read or create one
    ( -T $CIRFILE && -r $CIRFILE) or system( "touch", "$CIRFILE");
    ( -T $CIRFILE && -r $CIRFILE) or return_error( "File Error", "Can not open $CIRFILEORIG
for reading.");

    @wholecirfile= ();
    readwholecirfile( \@wholecirfile);

    #put wholecirfile in text element
    print "Content-type: text/html\n";
    print "Pragma: no-cache\n\n";
    print htmltop();
    print "<form METHOD=POST ACTION=\"/jwr-cgi-bin/sipbin/sip_savefile\">\n";
```

```
    print "  <input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\"
onClick=\"window.close();\">\n";
    print "  <input TYPE=\"submit\" VALUE=\"SAVE CHANGES\"> Remote CIR File:
<b>$CIRFILEORIG</b><br>\n";
    print "  <input TYPE=\"hidden\" NAME=\"filename\" VALUE=\"$CIRFILEHIDE\">\n";
    print "  <textarea NAME=\"filedata\" ROWS=30 COLS=80>\n";
    print @wholecirfile;
    print "  </textarea>\n";
    print "</form>\n";
    print htmlbottom();

    last SWITCH;
  }
  #-------------------------------------------------
  if( $TFhiddencommand eq "remotecirfiledelete")
  {
    unless( $TFremotelocal eq "remote")
    {
      return_error( "File Error", "Select the REMOTE FILES radio button.");
    }
    #check CIRFILE exists
    ( -e $CIRFILE) or return_error( "File Error", "Can not find $CIRFILEORIG to delete.");

    #delete file
    @args= ( "/usr/bin/rm", $CIRFILE);
    system( @args) == 0 or
      return_error( "System Error", "Could not execute rm command.");

    print "Content-type: text/html\n";
    print "Pragma: no-cache\n\n";
    print htmltop();
    print "<form METHOD=POST>\n";
    print "  <input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\"
onClick=\"window.close();\">\n";
    print "</form>\n";
    print "File <b>$CIRFILEORIG</b> has been deleted.\n";
    print htmlbottom();

    last SWITCH;
  }
  #-------------------------------------------------
  if( $TFhiddencommand eq "remoteoutfile")
  {
    unless( $TFremotelocal eq "remote")
    {
      return_error( "File Error", "Select the REMOTE FILES radio button.");
    }
    #check OUTFILE exists and can be read or create one
    ( -T $OUTFILE && -r $OUTFILE) or system( "touch", "$OUTFILE");
    ( -T $OUTFILE && -r $OUTFILE) or return_error( "File Error", "Can not open $OUTFILEORIG
for reading.");

    @wholeoutfile= ();
    readwholeoutfile( \@wholeoutfile);

    #put wholeoutfile in text element
    print "Content-type: text/html\n";
    print "Pragma: no-cache\n\n";
    print htmltop();
    print "<form METHOD=POST ACTION=\"/jwr-cgi-bin/sipbin/sip_savefile\">\n";
    print "  <input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\"
onClick=\"window.close();\">\n";
    print "  <input TYPE=\"submit\" VALUE=\"SAVE CHANGES\"> Remote OUT File:
<b>$OUTFILEORIG</b><br>\n";
    print "  <input TYPE=\"hidden\" NAME=\"filename\" VALUE=\"$OUTFILE\">\n";
    print "  <textarea NAME=\"filedata\" ROWS=30 COLS=80>\n";
    print @wholeoutfile;
    print "  </textarea>\n";
    print "</form>\n";
    print htmlbottom();

    last SWITCH;
```

```
  }
  #-------------------------------------------------
  if( $TFhiddencommand eq "remoteoutfiledelete")
  {
    unless( $TFremotelocal eq "remote")
    {
      return_error( "File Error", "Select the REMOTE FILES radio button.");
    }
    #check CIRFILE exists
    ( -e $OUTFILE) or return_error( "File Error", "Can not find $OUTFILEORIG to delete.");

    #delete file
    @args= ( "/usr/bin/rm", $OUTFILE);
    system( @args) == 0 or
      return_error( "System Error", "Could not execute rm command.");

    print "Content-type: text/html\n";
    print "Pragma: no-cache\n\n";
    print htmltop();
    print "<form METHOD=POST>\n";
    print "  <input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\"
onClick=\"window.close();\">\n";
    print "</form>\n";
    print "File <b>$OUTFILEORIG</b> has been deleted.\n";
    print htmlbottom();

    last SWITCH;
  }
  #-------------------------------------------------
  if( $TFhiddencommand eq "localcirfile")
  {
    unless( $TFremotelocal eq "local")
    {
      return_error( "File Error", "Select the LOCAL FILES radio button.");
    }
    #check CIRFILE exists and can be read
    ( -T $CIRFILE && -r $CIRFILE) or return_error( "File Error", "Can not open $CIRFILEORIG
for reading.");

    @wholecirfile= ();
    readwholecirfile( \@wholecirfile);

    #put wholeoutfile in window
    print "Content-type: text/html\n";
    print "Pragma: no-cache\n\n";
    print htmltop();
    print "<form METHOD=POST>\n";
    print "  <input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\"
onClick=\"window.close();\">\n";
    print "</form>\n";
    print "<pre>\n";
    print "-------------------------------------------------------------------------------
\n";
    print "Local File:  <b>$CIRFILEORIG</b>\n";
    print "-------------------------------------------------------------------------------
\n";
    print @wholecirfile;
    print "</pre>\n";
    print htmlbottom();

    last SWITCH;
  }
  #-------------------------------------------------
  if( $TFhiddencommand eq "localoutfile")
  {
    unless( $TFremotelocal eq "local")
    {
      return_error( "File Error", "Select the LOCAL FILES radio button.");
    }
    #check OUTFILE exists and can be read
    ( -T $OUTFILE && -r $OUTFILE) or return_error( "File Error", "Can not open $OUTFILEORIG
for reading.");
```

```
    @wholeoutfile= ();
    readwholeoutfile( \@wholeoutfile);

    #put wholeoutfile in text element
    print "Content-type: text/html\n";
    print "Pragma: no-cache\n\n";
    print htmltop();
    print "<form METHOD=POST>\n";
    print "  <input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\"
onClick=\"window.close();\">\n";
    print "</form>\n";
    print "<pre>\n";
    print "--------------------------------------------------------------------------------
\n";
    print "Local OUT File:  <b>$OUTFILEORIG</b>\n";
    print "--------------------------------------------------------------------------------
\n";
    print @wholeoutfile;
    print htmlbottom();

    last SWITCH;
  }
  #-------------------------------------------------
  if( $TFhiddencommand eq "help")
  {
    #check HELPFILE exists and can be read
    open( FILE, $HELPFILE) or return_error( "File Error", "Can not open Help file for
reading.");

    @wholehelpfile= ();
    readwholehelpfile( \@wholehelpfile);

    #return help file
    print "Content-type: text/html\n";
    print "Pragma: no-cache\n\n";
    print htmltop();
    print "<form METHOD=POST>\n";
    print "  <input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\"
onClick=\"window.close();\">\n";
    print "</form>\n";
    print @wholehelpfile;
    print htmlbottom();

    last SWITCH;
  }
  return_error( "Server Error", "Unknown command.");
}
exit( 0);
#the end.###################################################################################
###########################################################################################


#############################################################################
sub gnuplotdata
{
  local( $data, $misc)= @_;  #references

  #create the output file
  createoutputfile();

  #return type either GIF or TEXT
  if( $TFreturntype eq "Gif")
  {
    senddatatognuplot();
    printgiffileandcleanup();
  }
  elsif( $TFreturntype eq "Text")
  {
    returntextdata();
  }
  else
```

```
  {
    return_error( "Program Error", "Return type error.");
  }
}


################################################################################
sub returntextdata
{
  #output top of output page
  print "Content-type: text/html\n";
  print "Pragma: no-cache\n\n";
  print htmltop();
  print "<form METHOD=POST>\n";
  print "  <input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\"
onClick=\"window.close();\">\n";
  print "</form>\n";
  print "<pre>\n";

  print $TFtitletext . "\n";
  print "Analysis: " . $TFanalysistype . "\n";
  foreach $index ( @variableindex)
  {
    print ${$$data[0]}[$index] . "  ";
  }
  print "\n\n";

  open( FILE, $DATAFILE) or return_error( "SIP Error", "Cannot open data file.");
  while( <FILE>)
  {
    print;
  }
  print "</pre>\n";
  close( FILE);
  print htmlbottom();
}


################################################################################
sub createoutputfile
{
  open( FILE, ">" . $DATAFILE) or return_error( "SIP Error", "Cannot write to data file for
GNUPlot.");

  #only print selected variables
  @selectedvariables= eval( "\@TF" . $TFanalysistype . "analysis");
#  return_error( "@selectedvariables", "selected variables");

  @variableindex= ( 0);
  for( $i= 1; $i < @{$$data[0]}; $i++)
  {
    foreach $variable ( @selectedvariables)
    {
      if( ${$$data[0]}[$i] eq $variable)
      {
        push( @variableindex, $i);
        last;
      }
    }
  }

  #add selected data to the file
  for( $i= 1; $i < @{$data}; $i++)
  {
    #if the analysis starts over again add blank line to data file
    if( $i > 1 && ${$$data[$i-1]}[0] > ${$$data[$i]}[0])
    {
      print FILE "\n";
    }

    #print the selected data to the file
    foreach $j ( @variableindex)
```

```
    {
      print FILE "${$$data[$i]}[$j]  ";
    }

    print FILE "\n";
  }

  close( FILE);
}


#############################################################################
sub senddatatognuplot()
{
  open( GNUPLOT, "|$GNUPLOT");

  print GNUPLOT "set output \"$OUTPUT_PPM\"\n";
#  print GNUPLOT "set term pbm color small\n";
#  print GNUPLOT "set term pbm small monochrome\n";

#set term postscript default
#set term enhpost landscape color "Times-Roman" 14


  #axis style
  if( $TFaxisstyle eq "linlog")  #default is linlin
  {
    print GNUPLOT "set logscale x\n";
  }
  elsif( $TFaxisstyle eq "loglin")
  {
    print GNUPLOT "set logscale y\n";
  }
  elsif( $TFaxisstyle eq "loglog")
  {
    print GNUPLOT "set logscale xy\n";
  }

  #xaxis label
  if( $TFxaxislabelbox eq "ON")
  {
    print GNUPLOT "set xlabel \"$TFxaxislabeltext\" $TFxaxislabeldeltax,
$TFxaxislabeldeltay\n";
  }
  if( $TFyaxislabelbox eq "ON")
  {
    print GNUPLOT "set ylabel \"$TFyaxislabeltext\" $TFyaxislabeldeltax,
$TFyaxislabeldeltay\n";
  }

  #zero axis
  if( $TFnoxzeroaxisbox eq "ON")
  {
    print GNUPLOT "set noxzeroaxis\n";
  }
  if( $TFnoyzeroaxisbox eq "ON")
  {
    print GNUPLOT "set noyzeroaxis\n";
  }

  #title
  if( $TFtitlebox eq "ON")
  {
    print GNUPLOT "set title \"$TFtitletext\" $TFtitledeltax, $TFtitledeltay\n";
  }

  #date
  if( $TFdatebox eq "ON")
  {
    print GNUPLOT "set time $TFdatedeltax, $TFdatedeltay\n";
  }
```

```
#plottingstyle
print GNUPLOT "set data style $TFplottingstyle\n";

#zoom
if($TFzoomoptionsbox eq "ON")
{
  print GNUPLOT "set xrange [$TFxmin:$TFxmax]\n";
  print GNUPLOT "set yrange [$TFymin:$TFymax]\n";
}

#image size
if($TFimagesizebox eq "ON")
{
  print GNUPLOT "set size $TFimagesizex, $TFimagesizey\n";
}

#key
if( $TFkeyselect eq "hide")
{
  print GNUPLOT "set nokey\n";
}
elsif( $TFkeyselect eq "moveto")
{
  print GNUPLOT "set key $TFkeyx, $TFkeyy\n";
}

#grid and border
if( $TFgridbox eq "ON")
{
  print GNUPLOT "set grid\n";
}
if( $TFborderbox eq "ON")
{
  print GNUPLOT "set noborder\n";
}

#tics
if( $TFxticsbox eq "ON")
{
  print GNUPLOT "set xtics $TFxticsstart, $TFxticsstep, $TFxticsstop\n";
}
if( $TFyticsbox eq "ON")
{
  print GNUPLOT "set ytics $TFyticsstart, $TFyticsstep, $TFyticsstop\n";
}
if( $TFticsoutsidebox eq "ON")
{
  print GNUPLOT "set tics out\n";
}

#axis labels
if( $TFformatxlabelsbox eq "ON")
{
  print GNUPLOT "set format x \"$TFformatxlabels\"\n";
}
if( $TFformatylabelsbox eq "ON")
{
  print GNUPLOT "set format y \"$TFformatylabels\"\n";
}

#font size and color
print GNUPLOT "set term pbm $TFcolorselect $TFfontsizeselect\n";

#time stamp
if( $TFtimestampbox eq "ON")
{
  print GNUPLOT "set label \"$transientmisc{'date'}\" at $TFtimestampx, $TFtimestampy\n";
}

#text label
if( $TFlabel1box eq "ON")
{
```

```
    print GNUPLOT "set label \"$TFlabel1text\" at $TFlabel1x, $TFlabel1y\n";
  }
  if( $TFlabel2box eq "ON")
  {
    print GNUPLOT "set label \"$TFlabel2text\" at $TFlabel2x, $TFlabel2y\n";
  }

  #arrow label
  if( $TFarrow1box eq "ON")
  {
    print GNUPLOT "set arrow from $TFarrow1fromx, $TFarrow1fromy to $TFarrow1tox,
$TFarrow1toy\n";
  }
  if( $TFarrow2box eq "ON")
  {
    print GNUPLOT "set arrow from $TFarrow2fromx, $TFarrow2fromy to $TFarrow2tox,
$TFarrow2toy\n";
  }



  #send the data
  print GNUPLOT "plot ";
  for( $i= 1; $i < @variableindex; $i++)
  {
    print GNUPLOT "\"$DATAFILE\" using 1:$i+1 title \"@{$$data[0]}[$variableindex[$i]]\" ";
    unless( $i == @variableindex-1)
    {
      print GNUPLOT ", ";
    }
  }

  close( GNUPLOT);

#  return_error( "Error with GnuPlot","i am trying to fix this...sorry.");
}


################################################################################
sub printgiffileandcleanup()
{
  $|= 1;  #standard output unbuffered

  system( "$PPMTOGIF $OUTPUT_PPM 2> /dev/null > $TMPPLOTFILE");

  #output top of output page
  print "Content-type: text/html\n";
  print "Pragma: no-cache\n\n";
  print htmltop();
  print "<form METHOD=POST>\n";
  print "  <input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\"
onClick=\"window.close();\">\n";
  print "</form>\n";
  print "<img SRC=\"/~regnier/sip/sipdata/$$.gif\">";  #relative to HTTPd
  print htmlbottom();

  unlink( $OUTPUT_PPM, $DATAFILE);
}


################################################################################
sub calculatebounds
{
  #xmin,xmax,ymin,ymax
  @bounds= ( $$data[1][0],$$data[1][0],  #initialize @bounds
             $$data[1][1],$$data[1][1]);

  #find xmin and xmax
  for $j ( 1 .. $#$data)
  {
    if( $$data[$j][0] > $bounds[1])
    {
```

```
      $bounds[1]= $$data[$j][0];
    }
    elsif( $$data[$j][0] < $bounds[0])
    {
      $bounds[0]= $$data[$j][0];
    }
  }

  #find ymin and ymax
  for $i ( 1 .. $#$data)
  {
    for $j ( 1 .. $#{$$data[0]})
    {
      if( $$data[$i][$j] > $bounds[3])
      {
        $bounds[3]= $$data[$i][$j];
      }
      elsif( $$data[$i][$j] < $bounds[2])
      {
        $bounds[2]= $$data[$i][$j];
      }
    }
  }
#  return_error( "calculate bounds", "$bounds[0], $bounds[1], $bounds[2], $bounds[3]");
}


################################################################################
sub getdata
{
  my( $searchtype, $wholeoutfile, $data, $misc)= @_;  #references

  if( $searchtype eq "DC")
  {
    $searchstring= "DC transfer characteristic";
  }
  elsif( $searchtype eq "AC")
  {
    $searchstring= "AC Analysis";
  }
  elsif( $searchtype eq "Transient")
  {
    $searchstring= "Transient Analysis";
  }
  else
  {
    return_error( "No Data", "Neither DC nor AC nor Transient search string.");
  }

  my $firstsetofdata= 0;
  my $newsetofdata= 0;

  foreach $_ ( @$wholeoutfile)
  {
    if( /^\s*Circuit:\s*(.*)\s*$/)
    {
      $$misc{'circuit'}= $1;
    }
    elsif( /^\s*Date:\s*(.*)\s*$/)
    {
       $$misc{'date'}= $1;
    }
    elsif( /^\s*($searchstring)/)
    {
      $firstsetofdata++;
      $newsetofdata= 1;
    }
    elsif( /^\s*Index\s+(\S+)\s+(.*)\s*/ && $firstsetofdata)
    {
      if( $newsetofdata == 1)
      {
        if( $firstsetofdata == 1)  #include sweep variable
```

```
        {
          push( @{$$data[0]}, $1, split( /\s+/, $2));  #store more data variable names
        }
        else  #don't include sweep variable
        {
          push( @{$$data[0]}, split( /\s+/, $2));
        }
        $newsetofdata= 0;
      }
    }
    elsif( /^\s*(\d+)\s+([\d\.\-\+e]+)\s+([\d\.\-\+e\s]+)\s*/ && $firstsetofdata)  #data
    {
      if( $firstsetofdata == 1)  #include sweep value
      {
        push( @{$$data[$1+1]}, $2, split( /\s+/, $3));  #store more data values
      }
      else  #don't include sweep value
      {
        push( @{$$data[$1+1]}, split( /\s+/, $3));
      }
    }
    elsif( /^\s*(\d+)\s+([\d\.\-\+e]+)\s*,\s*[\d\.\+\-e]+\s+([\d\.\-\+e\s]+)\s*/ &&
$firstsetofdata)  #ac data
    {
      if( $firstsetofdata == 1)  #include sweep value
      {
        push( @{$$data[$1+1]}, $2, split( /\s+/, $3));  #store more data values
      }
      else  #don't include sweep value
      {
        push( @{$$data[$1+1]}, split( /\s+/, $3));
      }
    }
  }
}


##############################################################################
sub readwholeoutfile
{
  local( $wholefile)= @_;  #reference to data

  open( FILE, $OUTFILE) or return_error( "File Error", "Can not open file $OUTFILEORIG.");
  @$wholefile= <FILE>;

  close( FILE);
}


##############################################################################
sub readwholecirfile
{
  local( $wholefile)= @_;  #reference to data

  open( FILE, $CIRFILE) or return_error( "File Error", "Can not open file $CIRFILEORIG.");
  @$wholefile= <FILE>;

  close( FILE);
}


##############################################################################
sub readwholehelpfile
{
  local( $wholefile)= @_;  #reference to data
  open( FILE, $HELPFILE) or return_error( "File Error", "Can not open Help file.");

  @$wholefile= <FILE>;

  close( FILE);
}
```

```
##############################################################################
sub logusage
{
  ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)= localtime( time);

  #write to log file
  open( TMP, ">>" . $LOGFILE)
       or return_error( "No File", "The log file could not be opened.");
  flock( TMP, $LOCK_EX);

  print( TMP "Authorization Okay -- $TFpassword -- $TFhiddencommand -- " .
             sprintf( "%02d:%02d:%02d %02d/%02d/%02d -- ", $hour, $min, $sec, $mon+1, $mday,
$year) .
             "from REMOTE_HOST=$ENV{'REMOTE_HOST'} and REMOTE_ADDR=$ENV{'REMOTE_ADDR'}\n");
  flock( TMP, $LOCK_UN);
  close( TMP);

  if( $LOGYES == 1)
  {
    #email regnier@uwyo.edu
    open( MailPipe, "| /usr/ucb/mail -s \"SPICE Internet Package\" $WEBMASTER");
    print( MailPipe "REMOTE_HOST: $ENV{'REMOTE_HOST'}\n" .
                    "REMOTE_ADDR: $ENV{'REMOTE_ADDR'}\n" .
                    "       USER: $PASSWORD\n" .
                    "\n\n");
    close( MailPipe);
  }
}


##############################################################################
sub checkauthorization
{
  #log who is using the application
  logusage();

  #for now password is anything with sip in it
  unless ($TFpassword =~ /sip/i)
  {
    return( 0);
  }
  return( 1);
}


##############################################################################
sub htmltop
{
  $x= "<html>\n" .
      "<head><title>SPICE Internet Package</title></head>\n" .
      "<body BGCOLOR=\"\#8B8B83\" onLoad=\"window.focus();\">\n";

  return $x;
}


##############################################################################
sub htmlbottom
{
  $x= "\n</body>\n" .
      "</html>\n";

  return $x;
}


##############################################################################
sub return_error
{
  local( $keyword, $message)= @_;
```

```
  print "Content-type: text/html", "\n\n";

  print <<End_of_Error;

<html>
<head><title>SIP Program Error</title></head>
<body BGCOLOR="#8B8B83" onLoad=\"window.focus();\">
<form METHOD=POST>
  <input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\" onClick=\"window.close();\">
</form>

<h3>$keyword</h3>
$message<br><br>
<hr>

End_of_Error

  print "</body></html>";

  exit( 1);
}


##############################################################################
sub parse_form_data
{
  local( *FORM_DATA)= @_;
  local( $request_method, $query_string, @key_value_pairs,
         $key_value, $key, $value);

  $request_method= $ENV{'REQUEST_METHOD'};
  if( $request_method eq "GET")
  {
    $query_string= $ENV{'QUERY_STRING'};
  }
  elsif( $request_method eq "POST")
  {
    read( STDIN, $query_string, $ENV{'CONTENT_LENGTH'});
  }
  else
  {
    return_error( "Server Error",
                  "Server uses unsupported method");
  }

  @key_value_pairs= split( /&/, $query_string);
  foreach $key_value( @key_value_pairs)
  {
    ($key, $value)= split( /=/, $key_value);
    $value =~ tr/+/ /;
    $value =~ s/%([\dA-Fa-f][\dA-Fa-f])/pack ("C", hex( $1))/eg;

    if( defined( $FORM_DATA{$key}))
    {
      $FORM_DATA{$key}= join( "\0", $FORM_DATA{$key}, $value);
    }
    else
    {
      $FORM_DATA{$key}= $value;
    }
  }
}


##############################################################################
sub printhtmloptions
{
  my( $transientchecked, $dcchecked, $acchecked);
  $transientchecked= $dcchecked= $acchecked= "";

  #determine default selected analysis type
  if( @transientdata > 3)
```

```
  {
    $transientchecked= "CHECKED";
  }
  elsif( @dcdata > 3)
  {
    $dcchecked= "CHECKED";
  }
  elsif( @acdata > 3)
  {
    $acchecked= "CHECKED";
  }

  print "<hr>\n";
  print "<table WIDTH=\"380\" BGCOLOR=\"#EDEDED\">\n";
  print "<tr>\n";

  print "  <td ALIGN=\"center\"><b>Transient Analysis</b><br>\n";
  if( @transientdata > 3)
  {
    print "  <input TYPE=\"radio\" NAME=\"analysistype\" VALUE=\"transient\"
$transientchecked>\n";
  }
  print "  </td>\n";

  print "  <td ALIGN=\"center\"><b>DC Analysis</b><br>\n";
  if( @dcdata > 3)
  {
    print "  <input TYPE=\"radio\" NAME=\"analysistype\" VALUE=\"dc\" $dcchecked>\n";
  }
  print "  </td>\n";

  print "  <td ALIGN=\"center\"><b>AC Analysis</b><br>\n";
  if( @acdata > 3)
  {
    print "  <input TYPE=\"radio\" NAME=\"analysistype\" VALUE=\"ac\" $acchecked>\n";
  }
  print "  </td>\n";

  print "</tr>\n";
  print "<tr>\n";

####################
  if( @transientdata > 3)
  {
    print "  <td ALIGN=\"center\"><select NAME=\"transientanalysis\" SIZE=4 MULTIPLE>\n";
    for( $i= 1; $i < @{$transientdata[0]}; $i++)
    {
      print "  <option SELECTED>${$transientdata[0]}[$i]\n";
    }
    print "  </select>\n";
    print "  </td>\n";
  }
  else
  {
    print "  <td ALIGN=\"center\"><i>no data</i></td>\n";
  }

####################
  if( @dcdata > 3)
  {
    print "  <td ALIGN=\"center\"><select NAME=\"dcanalysis\" SIZE=4 MULTIPLE>\n";
    for( $i= 1; $i < @{$dcdata[0]}; $i++)
    {
        print "  <option SELECTED>${$dcdata[0]}[$i]\n";
    }
      print "  </select>\n";
      print "  </td>\n";
  }
  else
  {
    print "  <td ALIGN=\"center\"><i>no data</i></td>\n";
  }
```

```
####################
  if( @acdata > 3)
  {
    print "  <td ALIGN=\"center\"><select NAME=\"acanalysis\" SIZE=4 MULTIPLE>\n";
    for( $i= 1; $i < @{$acdata[0]}; $i++)
    {
      print "  <option SELECTED>${$acdata[0]}[$i]\n";
    }
    print "  </select>\n";
    print "  </td>\n";
  }
  else
  {
    print "  <td ALIGN=\"center\"><i>no data</i></td>\n";
  }

  print "</tr>\n";
  print "</table>\n";

################################################################################
# HOT METAL output html follows...
################################################################################
  print <<EndPlotHTML;

    <TABLE WIDTH="380" BGCOLOR="#C0C0C0">
      <TR>
        <TD WIDTH="123" ALIGN="CENTER"><FONT COLOR="#FF0000"><B><FONT COLOR="#0000FF">Return
        Type</FONT></B></FONT></TD>
        <TD WIDTH="123" ALIGN="CENTER"><FONT COLOR="#0000FF"><B>Axis Style</B></FONT></TD>
        <TD WIDTH="123" ALIGN="CENTER"><FONT COLOR="#8000FF"><B><FONT
COLOR="#0000FF">Plotting
        Style</FONT></B></FONT></TD>
      </TR>
      <TR>
        <TD WIDTH="123" ALIGN="CENTER"><SELECT NAME="returntype" SIZE="1">
        <OPTION SELECTED="SELECTED">Gif </OPTION>
        <OPTION>Text </OPTION>
        </SELECT></TD>
        <TD WIDTH="123" ALIGN="CENTER"><SELECT NAME="axisstyle" SIZE="1">
        <OPTION SELECTED="SELECTED">linlin </OPTION>
        <OPTION>linlog </OPTION>
        <OPTION>loglin </OPTION>
        <OPTION>loglog </OPTION></SELECT></TD>
        <TD WIDTH="123" ALIGN="CENTER"><SELECT NAME="plottingstyle" SIZE="1">
        <OPTION>points </OPTION>
        <OPTION SELECTED="SELECTED">lines </OPTION>
        <OPTION>linespoints </OPTION>
        <OPTION>dots </OPTION>
        <OPTION>steps </OPTION>
        <OPTION>impulses </OPTION></SELECT></TD>
      </TR>
    </TABLE>
    <TABLE WIDTH="500" BORDER="0" BGCOLOR="#C0C0C0">
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"></TD>
        <TD WIDTH="70"></TD>
        <TD WIDTH="280"></TD>
        <TD WIDTH="60" ALIGN="CENTER"><FONT COLOR="#FF0000">deltaX</FONT></TD>
        <TD WIDTH="60" ALIGN="CENTER"><FONT COLOR="#FF0000">deltaY</FONT></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="titlebox" VALUE="ON"
CHECKED="CHECKED"></TD>
        <TD WIDTH="70"><FONT COLOR="#0000FF"><B>Title</B></FONT></TD>
        <TD WIDTH="280"><INPUT TYPE="TEXT" NAME="titletext"
VALUE="$transientmisc{'circuit'}" SIZE="25"></TD>
        <TD WIDTH="60" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="titledeltax" VALUE="10"
SIZE="3"></TD>
        <TD WIDTH="60" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="titledeltay" VALUE="0"
SIZE="3"></TD>
      </TR>
```

```
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="datebox" VALUE="ON"
CHECKED="CHECKED"></TD>
        <TD WIDTH="70"><FONT COLOR="#0000FF"><B>Date</B></FONT></TD>
        <TD WIDTH="280"></TD>
        <TD WIDTH="60" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="datedeltax" VALUE="10"
SIZE="3"></TD>
        <TD WIDTH="60" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="datedeltay" VALUE="0"
SIZE="3"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="xaxislabelbox"
VALUE="ON"></TD>
        <TD WIDTH="70"><FONT COLOR="#0000FF"><B>xLabel</B></FONT></TD>
        <TD WIDTH="280"><INPUT TYPE="TEXT" NAME="xaxislabeltext" SIZE="20"></TD>
        <TD WIDTH="60" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="xaxislabeldeltax" VALUE="0"
SIZE="3"></TD>
        <TD WIDTH="60" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="xaxislabeldeltay" VALUE="0"
SIZE="3"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="yaxislabelbox"
VALUE="ON"></TD>
        <TD WIDTH="70"><FONT COLOR="#0000FF"><B>yLabel</B></FONT></TD>
        <TD WIDTH="280"><INPUT TYPE="TEXT" NAME="yaxislabeltext" SIZE="20"></TD>
        <TD WIDTH="60" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="yaxislabeldeltax" VALUE="0"
SIZE="3"></TD>
        <TD WIDTH="60" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="yaxislabeldeltay" VALUE="0"
SIZE="3"></TD>
      </TR>
    </TABLE>
    <TABLE WIDTH="500" BORDER="0" BGCOLOR="#C0C0C0">
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"></TD>
        <TD WIDTH="230"></TD>
        <TD WIDTH="60" ALIGN="RIGHT"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">min</FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">max</FONT></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="zoomoptionsbox"
VALUE="ON"></TD>
        <TD WIDTH="230"><FONT COLOR="#0000FF"><B>Zoom</B></FONT></TD>
        <TD WIDTH="60" ALIGN="RIGHT"><FONT COLOR="#004000">xAxis</FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="xmin" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="xmax" SIZE="8"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"></TD>
        <TD WIDTH="230"></TD>
        <TD WIDTH="60" ALIGN="RIGHT"><FONT COLOR="#004000">yAxis</FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="ymin" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="ymax" SIZE="8"></TD>
      </TR>
    </TABLE>
    <TABLE WIDTH="500" BORDER="0" BGCOLOR="#C0C0C0">
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"></TD>
        <TD WIDTH="200"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">start</FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">step</FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">stop</FONT></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="xticsbox"
VALUE="ON"></TD>
        <TD WIDTH="200"><FONT COLOR="#0000FF"><B>xTics</B></FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="xticsstart" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="xticsstep" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="xticsstop" SIZE="8"></TD>
      </TR>
      <TR>
```

```
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="yticsbox"
VALUE="ON"></TD>
        <TD WIDTH="200"><FONT COLOR="#0000FF"><B>yTics</B></FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="yticsstart" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="yticsstep" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="yticsstop" SIZE="8"></TD>
      </TR>
    </TABLE>
    <TABLE WIDTH="500" BORDER="0" BGCOLOR="#C0C0C0">
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="gridbox" VALUE="ON"
CHECKED="CHECKED"></TD>
        <TD WIDTH="360"><FONT COLOR="#0000FF"><B>Show Grid</B></FONT></TD>
        <TD WIDTH="110" ALIGN="CENTER"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="borderbox"
VALUE="ON"></TD>
        <TD WIDTH="360"><FONT COLOR="#0000FF"><B>No Border</B></FONT></TD>
        <TD WIDTH="110" ALIGN="CENTER"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="ticsoutsidebox"
VALUE="ON"></TD>
        <TD WIDTH="360"><FONT COLOR="#0000FF"><B>Tics Outside</B></FONT></TD>
        <TD WIDTH="110" ALIGN="CENTER"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="noxzeroaxisbox"
VALUE="ON"></TD>
        <TD WIDTH="360"><FONT COLOR="#0000FF"><B>No xZero Axis</B></FONT></TD>
        <TD WIDTH="110" ALIGN="CENTER"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="noyzeroaxisbox"
VALUE="ON"></TD>
        <TD WIDTH="360"><FONT COLOR="#0000FF"><B>No yZero Axis</B></FONT></TD>
        <TD WIDTH="110" ALIGN="CENTER"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"></TD>
        <TD WIDTH="360"><FONT COLOR="#0000FF"><B>Font Size</B></FONT></TD>
        <TD WIDTH="110" ALIGN="CENTER"><SELECT NAME="fontsizeselect" SIZE="1">
        <OPTION SELECTED="SELECTED">small </OPTION>
        <OPTION>medium </OPTION>
        <OPTION>large </OPTION></SELECT></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"></TD>
        <TD WIDTH="360"><FONT COLOR="#0000FF"><B>Color Options</B></FONT></TD>
        <TD WIDTH="110" ALIGN="CENTER"><SELECT NAME="colorselect" SIZE="1">
        <OPTION SELECTED="SELECTED">color </OPTION>
        <OPTION>gray </OPTION>
        <OPTION>monochrome </OPTION></SELECT></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="formatxlabelsbox"
VALUE="ON"></TD>
        <TD WIDTH="360"><FONT COLOR="#0000FF"><B>xAxis Format</B></FONT></TD>
        <TD WIDTH="110" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="formatxlabels" VALUE="%0.2f"
SIZE="8"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="formatylabelsbox"
VALUE="ON"></TD>
        <TD WIDTH="360"><FONT COLOR="#0000FF"><B>yAxis Format</B></FONT></TD>
        <TD WIDTH="110" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="formatylabels" VALUE="%1.1e"
SIZE="8"></TD>
      </TR>
    </TABLE>
    <TABLE WIDTH="500" BORDER="0" BGCOLOR="#C0C0C0">
      <TR>
```

```
        <TD WIDTH="30" ALIGN="CENTER"></TD>
        <TD WIDTH="290"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">xScale</FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">yScale</FONT></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="imagesizebox"
VALUE="ON"></TD>
        <TD WIDTH="290"><FONT COLOR="#0000FF"><B>Image Scale</B></FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="imagesizex" VALUE="0.8"
SIZE="3"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="imagesizey" VALUE="0.8"
SIZE="3"></TD>
      </TR>
    </TABLE>
    <TABLE WIDTH="500" BORDER="0" BGCOLOR="#C0C0C0">
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"></TD>
        <TD WIDTH="140"></TD>
        <TD WIDTH="150" ALIGN="CENTER"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">x</FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">y</FONT></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"></TD>
        <TD WIDTH="140"><FONT COLOR="#0000FF"><B>Key</B></FONT></TD>
        <TD WIDTH="150" ALIGN="CENTER"><SELECT NAME="keyselect" SIZE="1">
        <OPTION SELECTED="SELECTED">default </OPTION>
        <OPTION>hide </OPTION>
        <OPTION>moveto </OPTION></SELECT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="keyx" VALUE="0"
SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="keyy" VALUE="0"
SIZE="8"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="timestampbox"
VALUE="ON"></TD>
        <TD WIDTH="140"><FONT COLOR="#0000FF"><B>Simulation Time</B></FONT></TD>
        <TD WIDTH="150" ALIGN="CENTER"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="timestampx" VALUE="0"
SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="timestampy" VALUE="0"
SIZE="8"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="label1box"
VALUE="ON"></TD>
        <TD WIDTH="140"><FONT COLOR="#0000FF"><B>Text Label</B></FONT></TD>
        <TD WIDTH="150" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="label1text" SIZE="15"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="label1x" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="label1y" SIZE="8"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="label2box"
VALUE="ON"></TD>
        <TD WIDTH="140"><FONT COLOR="#0000FF"><B>Text Label</B></FONT></TD>
        <TD WIDTH="150" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="label2text" SIZE="15"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="label2x" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="label2y" SIZE="8"></TD>
      </TR>
    </TABLE>
    <TABLE WIDTH="500" BGCOLOR="#C0C0C0" BORDER="0">
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"></TD>
        <TD WIDTH="110"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">fromX</FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">fromY</FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">toX</FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><FONT COLOR="#FF0000">toY</FONT></TD>
      </TR>
      <TR>
```

```
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="arrow1box"
VALUE="ON"></TD>
        <TD WIDTH="110"><FONT COLOR="#0000FF"><B>Arrow</B></FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="arrow1fromx" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="arrow1fromy" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="arrow1tox" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="arrow1toy" SIZE="8"></TD>
      </TR>
      <TR>
        <TD WIDTH="30" ALIGN="CENTER"><INPUT TYPE="CHECKBOX" NAME="arrow2box"
VALUE="ON"></TD>
        <TD WIDTH="110"><FONT COLOR="#0000FF"><B>Arrow</B></FONT></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="arrow2fromx" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="arrow2fromy" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="arrow2tox" SIZE="8"></TD>
        <TD WIDTH="90" ALIGN="CENTER"><INPUT TYPE="TEXT" NAME="arrow2toy" SIZE="8"></TD>
      </TR>
    </TABLE>

EndPlotHTML
}

################################################################################
################################################################################
################################################################################
################################################################################
```

Appendix C
SIP Main Perl Script (sip_login)

```
#!/opt/LWperl/bin/perl

##############################################################################
## Program:  SPICE Internet Package perl script
##
## Developed by:
##    John Regnier
##    Bogdan Wilamowski
##    Department of Electrical Engineering
##    University of Wyoming
##
##  Last Revision: 18oct97
##        Contact: wilam@uwyo.edu or regnier@uwyo.edu
##############################################################################
use lib '/export/home2/regnier/lib/perl/lib';
use lib '/export/home2/regnier/lib/perl/arch';
use CGI_Lite;

$LOGYES= 1;  #sends $WEBMASTER email
$WEBMASTER= "regnier\@uwyo.edu";

#file locking
$LOCK_EX= 2;
$LOCK_NB= 4;

$ENV{LD_LIBRARY_PATH}= "/usr/openwin/lib";
$ENV{PATH}= $ENV{PATH} . ":/export/home2/regnier/spice3/bin";

#parse the form data
$form= new CGI_Lite;
$form->set_directory( "/tmp") or return_error( "File Error", "Directory does not exist for
CGILite.");
$form->set_buffer_size( 1024);
$form->set_platform( "Unix");
$form->set_file_type( "handle");  #or "file"
%theform= $form->parse_form_data();
$form->close_all_files;

$LOGFILE= "/export/home2/regnier/cgi-bin/sipdata/log.txt";
$PASSWDFILE= "/export/home2/regnier/cgi-bin/sipdata/passwd.txt";

#input from the /sip/index.html page
$TFpassword= $theform{'password'};

#check password file and get information from it
open( PFILE, $PASSWDFILE) or return_error( "File Error", "Can't open the password file.");

undef( $ok);
while( <PFILE>)
{
  if( /^\s*(.+)\s*,\s*(.+)\s*,\s*(.+)\s*,\s*(.+)\s*,\s*(.+)\s*$/)
  {
    $PASSWORD= $1;
    $USERNAME= $2;
    $DIRECTORY= $3;
    $ACCESSES= $4;
    $LASTLOGIN= $5;
    if( $TFpassword =~ /^$PASSWORD$/i)
    {
      $ok= 1;
      #make sure lower case password
      $TFpassword =~ tr/A-Z/a-z/;
      last;
    }
  }
```

```
}
close( PFILE);

if( !defined( $ok))
{
  logusage( "Authorization Failure", $TFpassword);
  return_error( "Authorization Error", "Invalid password was entered.");
}

#authorization okay - update passwd file
logusage( "Authorization Okay", $TFpassword);
updatepasswd();  #uses local variables

#get list of users remote files
@remotefiles= ();
getremotefiles();


#send the sipwindow.html data
print "Content-type: text/html\n\n";

print <<ENDOFFILE;

<!DOCTYPE HTML PUBLIC "-//SoftQuad//DTD HoTMetaL PRO 4.0::19970916::extensions to HTML
4.0//EN"
 "hmpro4.dtd">

<HTML>

  <HEAD>
    <TITLE>SPICE Internet Package</TITLE>

<SCRIPT LANGUAGE="JavaScript">
<!-- HIDE FROM OLD BROWSERS
holdRemoteCirFile= "file17006.cir";
holdRemoteOutFile= "file17006.out";
holdLocalCirFile= "";
holdLocalOutFile= "";
var editwindow= 0;
var outputwindow= 0;
var plotoptionswindow= 0;
////////////////////////////////////////////////////////////////////////////
function exitSip()
{
  if( editwindow != 0 && !editwindow.closed )
  {
    editwindow.close();
  }
  if( outputwindow != 0 && !outputwindow.closed )
  {
    outputwindow.close();
  }
  if( plotoptionswindow != 0 && !plotoptionswindow.closed)
  {
    plotoptionswindow.close();
  }
}
////////////////////////////////////////////////////////////////////////////
function submitForm( form)
{
  if( form.hiddencommand.value == "remotecirfile" || form.hiddencommand.value ==
"remoteoutfile" ||
      form.hiddencommand.value == "localcirfile" || form.hiddencommand.value ==
"localoutfile")
  {
    form.target= "editwindow";
    editwindow= window.open( "", "editwindow", "resizable=1,scrollbars=1,width=400");
  }
  else if( form.hiddencommand.value == "plotdata")
  {
    form.target= "plotoptionswindow";
```

```
    plotoptionswindow= window.open( "", "plotoptionswindow",
"resizable=1,scrollbars=1,width=500");
  }
  else
  {
    form.target= "outputwindow";
    outputwindow= window.open( "", "outputwindow", "resizable=1,scrollbars=1,width=650");
  }
  form.submit();
}
////////////////////////////////////////////////////////////////////////////
function updateRemoteChecked( form)
{
  //restore old values
  form.remotecirfile.value= holdRemoteCirFile;
  form.remoteoutfile.value= holdRemoteOutFile;
  //save values
  //holdLocalCirFile= form.localcirfile.value;
  //holdLocalOutFile= form.localoutfile.value;
  //disable local
  //form.localcirfile.value= "";
  //form.localoutfile.value= "";
}
////////////////////////////////////////////////////////////////////////////
function updateLocalChecked( form)
{
  //restore old values
  //form.localcirfile.value= holdLocalCirFile;
  //form.localoutfile.value= holdLocalOutFile;
  //save values
  holdRemoteCirFile= form.remotecirfile.value;
  holdRemoteOutFile= form.remoteoutfile.value;
  //disable remote
  form.remotecirfile.value= "";
  form.remoteoutfile.value= "";
}
////////////////////////////////////////////////////////////////////////////
function changeCIRuserfiles( form)
{
  var nTmp;
  nTmp= form.ciruserfiles.selectedIndex;
  form.remotecirfile.value= form.ciruserfiles.options[nTmp].text;

  var nLen;
  nTmp= form.remotecirfile.value;
  nDot= nTmp.lastIndexOf( ".", nTmp.length);
  if( nDot == -1)
  {
      nTmp= nTmp + ".out";
  }
  else
  {
    nTmp= nTmp.substring( 0, nDot) + ".out";
  }
  form.remoteoutfile.value= nTmp;
}
////////////////////////////////////////////////////////////////////////////
function changeOUTuserfiles( form)
{
  var nTmp;
  nTmp= form.outuserfiles.selectedIndex;
  form.remoteoutfile.value= form.outuserfiles.options[nTmp].text;
}
// STOP HIDING FROM OLD BROWSERS -->
</SCRIPT>
  </HEAD>

  <BODY BGCOLOR="#8B8B83" ONLOAD="">

    <hr>
    <pre>
  Authorized User: <b>$USERNAME</b>
```

```
        Accesses: $ACCESSES
      Last Login: $LASTLOGIN</pre><hr>


    <FORM ENCTYPE="multipart/form-data"
    ACTION="/jwr-cgi-bin/sipbin/sip_main" METHOD="POST" NAME="sip"
    TARGET="sipwindow" ONSUBMIT="submitForm( this);">
    <TABLE WIDTH="500" BORDER="0">
      <TR>
        <TD WIDTH="250" BGCOLOR="#408080" ALIGN="CENTER"><B>SPICE Engine</B></TD>
        <TD WIDTH="250" BGCOLOR="#C0C0C0" ALIGN="CENTER">Use
        <FONT COLOR="#FF8000"><FONT COLOR="#800000"><B>REMOTE</B> Files</FONT></FONT></TD>
        <TD WIDTH="250" ALIGN="CENTER" BGCOLOR="#999999">Use
        <FONT COLOR="#0000FF"><B>LOCAL</B> Files</FONT></TD>
      </TR>
      <TR>
        <TD WIDTH="250" ALIGN="CENTER" ROWSPAN="2"><SELECT NAME="spiceversion" SIZE="1">
        <OPTION SELECTED="SELECTED">SPICE 3f5</OPTION>
        <OPTION>SPICE 2</OPTION></SELECT></TD>
        <TD WIDTH="250" BGCOLOR="#C0C0C0" ALIGN="CENTER"><INPUT TYPE="radio"
NAME="remotelocal" VALUE="remote" CHECKED="CHECKED" ONCLICK="updateRemoteChecked(
this.form);"></TD>
        <TD WIDTH="250" ALIGN="CENTER" BGCOLOR="#999999"><INPUT TYPE="radio"
NAME="remotelocal" VALUE="local" ONCLICK="updateLocalChecked( this.form);"></TD>
      </TR>
      <TR>
        <TD WIDTH="250" BGCOLOR="#C0C0C0" ALIGN="CENTER"><FONT COLOR="#004000"><B><I>files
        on the server</I></B></FONT></TD>
        <TD WIDTH="250" ALIGN="CENTER" BGCOLOR="#999999"><FONT COLOR="#004000"><B><I>files
        on your machine</I></B></FONT></TD>
      </TR>
    </TABLE>

    <P><BR>
      </P>
    <TABLE WIDTH="700" BORDER="0">
      <TR>
        <TD ALIGN="CENTER" WIDTH="190"></TD>
        <TD ALIGN="CENTER" WIDTH="255" BGCOLOR="#C0C0C0"><B><FONT COLOR="#800000">Stored
        <FONT COLOR="#FF0000">CIR</FONT> Files</FONT></B></TD>
        <TD ALIGN="CENTER" WIDTH="255" BGCOLOR="#C0C0C0"><B><FONT COLOR="#800000">Stored
        <FONT COLOR="#FF0000">OUT</FONT> Files</FONT></B></TD>
      </TR>
      <TR>
        <TD ALIGN="CENTER" WIDTH="190"></TD>
        <TD ALIGN="CENTER" WIDTH="255" BGCOLOR="#C0C0C0"><SELECT NAME="ciruserfiles"
SIZE="1" ONCHANGE="changeCIRuserfiles( this.form);">
ENDOFFILE

if( @remotefiles == 0)
{
  push( @remotefiles, "no stored files");
}

foreach $file ( @remotefiles)
{
  if( $file =~ /.*cir$|.*ckt$|.*spi$|^\w+$/i)
  {
    print "<option>$file</option>\n";
  }
}

print <<ENDOFFILE;
            </SELECT>
            </TD>

        <TD ALIGN="CENTER" WIDTH="255" BGCOLOR="#C0C0C0"><SELECT NAME="outuserfiles"
SIZE="1" ONCHANGE="changeOUTuserfiles( this.form);">
ENDOFFILE

if( @remotefiles == 0)
{
```

```
  push( @remotefiles, "no stored files");
}

foreach $file ( @remotefiles)
{
  if( $file =~ /.*out$/i)
  {
    print "<option>$file</option>\n";
  }
}

print <<ENDOFFILE;
            </SELECT>
            </TD>

      </TR>
    </TABLE>
    <TABLE WIDTH="700" BORDER="0">
      <TR BGCOLOR="#CCCCCC">
        <TD WIDTH="190" ALIGN="RIGHT"><FONT COLOR="#800000">Remote
        <FONT COLOR="#FF0000">CIR</FONT> Filename</FONT></TD>
        <TD WIDTH="310"><INPUT TYPE="text" NAME="remotecirfile" VALUE="file$$.cir"
SIZE="20"></TD>
        <TD WIDTH="200" ALIGN="CENTER"><INPUT TYPE="button" VALUE="View/Edit"
ONCLICK="this.form.hiddencommand.value= 'remotecirfile'; submitForm( this.form);">
        <INPUT TYPE="button" VALUE="DELETE" ONCLICK="this.form.hiddencommand.value=
'remotecirfiledelete'; submitForm( this.form);"></TD>
      </TR>
      <TR BGCOLOR="#CCCCCC">
        <TD WIDTH="190" ALIGN="RIGHT"><FONT COLOR="#800000">Remote
        <FONT COLOR="#FF0000">OUT</FONT> Filename</FONT></TD>
        <TD WIDTH="310"><INPUT TYPE="text" NAME="remoteoutfile" VALUE="file$$.out"
SIZE="20"></TD>
        <TD WIDTH="200" ALIGN="CENTER"><INPUT TYPE="button" VALUE="View/Edit"
ONCLICK="this.form.hiddencommand.value= 'remoteoutfile'; submitForm( this.form);">
        <INPUT TYPE="button" VALUE="DELETE" ONCLICK="this.form.hiddencommand.value=
'remoteoutfiledelete'; submitForm( this.form);"></TD>
      </TR>
      <TR BGCOLOR="#999999">
        <TD WIDTH="190" ALIGN="RIGHT"><FONT COLOR="#0000FF">Local CIR Filename</FONT></TD>
        <TD WIDTH="310"><INPUT TYPE="file" NAME="localcirfile" SIZE="20"></TD>
        <TD WIDTH="200" ALIGN="CENTER"><INPUT TYPE="button" VALUE="View"
ONCLICK="this.form.hiddencommand.value= 'localcirfile'; submitForm( this.form);"></TD>
      </TR>
      <TR BGCOLOR="#999999">
        <TD WIDTH="190" ALIGN="RIGHT"><FONT COLOR="#0000FF">Local OUT Filename</FONT></TD>
        <TD WIDTH="310"><INPUT TYPE="file" NAME="localoutfile" SIZE="20"></TD>
        <TD WIDTH="200" ALIGN="CENTER"><INPUT TYPE="button" VALUE="View"
ONCLICK="this.form.hiddencommand.value= 'localoutfile'; submitForm( this.form);"></TD>
      </TR>
    </TABLE><INPUT TYPE="hidden" VALUE="help" NAME="hiddencommand">
    <INPUT TYPE="hidden" VALUE="$TFpassword" NAME="password">

    <P><BR>
      </P>
    <TABLE WIDTH="700" BORDER="0">
      <TR>
        <TD WIDTH="190" ALIGN="CENTER"><INPUT TYPE="button" VALUE="RUN Simulation"
ONCLICK="this.form.hiddencommand.value= 'runsimulation'; submitForm( this.form);"></TD>
        <TD WIDTH="510" ALIGN="LEFT"><I>simulate CIR file and store output to
        OUT file</I> [use SPICE <FONT COLOR="#0000FF"><B>.print</B> </FONT>option]</TD>
      </TR>
      <TR>
        <TD WIDTH="190" ALIGN="CENTER" COLSPAN="1"><INPUT TYPE="button" VALUE="PLOT Data"
ONCLICK="this.form.hiddencommand.value= 'plotdata'; submitForm( this.form);"></TD>
        <TD WIDTH="510" ALIGN="LEFT" COLSPAN="1"><I>generate plot using data
        in OUT file</I></TD>
      </TR>
      <TR>
        <TD WIDTH="190" ALIGN="CENTER" COLSPAN="1"><INPUT TYPE="button" VALUE="Help"
ONCLICK="this.form.hiddencommand.value= 'help'; submitForm( this.form);"></TD>
        <TD WIDTH="510" ALIGN="LEFT" COLSPAN="1"><I>help with the SIP program</I></TD>
```

```
      </TR>
      <TR>
        <TD WIDTH="190" ALIGN="CENTER" COLSPAN="1"><INPUT TYPE="button" VALUE="EXIT"
ONCLICK="exitSip( this.form);"></TD>
        <TD WIDTH="510" ALIGN="LEFT" COLSPAN="1"><I>close other open windows</I></TD>
      </TR>
    </TABLE></FORM>
    <HR>
  </BODY>
</HTML>

ENDOFFILE

exit( 0);
################################################################################
################################################################################
################################################################################
################################################################################


################################################################################
sub getremotefiles
{
  $dir= "/export/home2/regnier/cgi-bin/sipdata/sip_" . $PASSWORD;
  unless( -d $dir)
  {
    return_error( "File Error", "Your remote directory does not exist.");
  }
  @remotefiles= `ls $dir`;
}


################################################################################
sub updatepasswd
{
  ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)= localtime( time);
  $THISLOGIN= sprintf( "%02d:%02d:%02d %02d/%02d/%02d", $hour, $min, $sec, $mon+1, $mday,
$year);
  $THISACCESSES= $ACCESSES+1;

  open( PFILE, $PASSWDFILE) or die( "File Error", "Can't open password file for read.");
  @newpfile= ();

  while( <PFILE>)
  {
    if(
/^\s*$PASSWORD\s*,\s*$USERNAME\s*,\s*$DIRECTORY\s*,\s*$ACCESSES\s*,\s*$LASTLOGIN\s*$/)
    {
      push( @newpfile, "$PASSWORD,$USERNAME,$DIRECTORY,$THISACCESSES,$THISLOGIN\n");
    }
    else
    {
      push( @newpfile, $_);
    }
  }
  close( PFILE);

  open( PFILE, ">" . $PASSWDFILE) or return_error( "File Error", "Can't open password file
for write.");
  flock( PFILE, $LOCK_EX);
  foreach $line( @newpfile)
  {
    print PFILE $line;
  }
  flock( PFILE, $LOCK_UN);
  close( PFILE);
}



################################################################################
```

```
sub logusage
{
  ( $message, $user)= @_;

  ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)= localtime( time);

  #write to log file
  open( TMP, ">>" . $LOGFILE)
       or return_error( "No File", "The log file could not be opened.");
  flock( TMP, $LOCK_EX);
  print( TMP "$message -- $user -- LOGIN -- " .
            sprintf( "%02d:%02d:%02d %02d/%02d/%02d -- ", $hour, $min, $sec, $mon+1, $mday,
$year) .
            "from REMOTE_HOST=$ENV{'REMOTE_HOST'} and REMOTE_ADDR=$ENV{'REMOTE_ADDR'}\n");
  flock( TMP, $LOCK_UN);
  close( TMP);

  if( $LOGYES == 1)
  {
    #email regnier@uwyo.edu
    open( MailPipe, "| /usr/ucb/mail -s \"SPICE Internet Package\" $WEBMASTER");
    print( MailPipe "REMOTE_HOST: $ENV{'REMOTE_HOST'}\n" .
                    "REMOTE_ADDR: $ENV{'REMOTE_ADDR'}\n" .
                    "    MESSAGE: $message\n" .
                    "       USER: $user\n" .
                    "\n\n");
    close( MailPipe);
  }
}


################################################################################
sub return_error
{
  local( $keyword, $message)= @_;

  print "Content-type: text/html", "\n\n";

  print <<End_of_Error;

<html>
<head><title>SIP Program Error</title></head>
<body BGCOLOR="#8B8B83">

<h3>$keyword</h3>
$message<br><br>
<hr>

End_of_Error

  print "</body></html>";
  exit( 1);
}
```

Appendix D
SIP Log File Perl Script

```
#!/opt/LWperl/bin/perl

$webmaster= "regnier\@uwyo\.edu";
&parse_form_data( *theform);

$filename= $theform{'filename'};
$filedata= $theform{'filedata'};

#for security do not show the absolute path
$FILENAMEORIG= $filename;
$FILENAMEORIG =~ s/.*\/(.*)/\1/g;

$filename= "/export/home2/regnier/cgi-bin/sipdata" . $filename;
$filenametemp= $filename . "_temp";

#check file can be written
open( FILE, ">".$filename) or return_error( "File Error", "The file $FILENAMEORIG could not
be opened for writing.");

#write data to file
print( FILE $filedata);
close( FILE);

#run dos2unix utility
@args= ( "/usr/bin/dos2unix", $filename, $filenametemp);
system( @args) == 0 or
  return_error( "System Error", "Could not execute dos2unix command.");

#remove temporary file and replace original
@args= ( "/usr/bin/mv", $filenametemp , $filename);
system( @args) == 0 or
  return_error( "System Error", "Could not move file.");

print "Content-type: text/html\n";
print "Pragma: no-cache\n\n";
print htmltop();
print "<form METHOD=POST>\n";
print "  <input TYPE=\"button\" NAME=\"\" VALUE=\"CLOSE WINDOW\"
onClick=\"window.close();\">\n";
print "</form>\n";
print "File <b>$FILENAMEORIG</b> has been saved.\n";
print htmlbottom();

exit( 0);


################################################################################
sub htmltop
{
  $x= "<html>\n" .
      "<head><title>SPICE Internet Package</title></head>\n" .
      "<body BGCOLOR=\"\#8B8B83\">\n";
  return $x;
}


################################################################################
sub htmlbottom
{
  $x= "\n</body>\n" .
      "</html>\n";

  return $x;
}


################################################################################
```

```
sub return_error
{
  local( $keyword, $message)= @_;

  print "Content-type: text/html", "\n\n";

  print <<End_of_Error;

<html>
<head><title>CGI Program Error</title></head>
<body BGCOLOR="#BEBEBE">
<hr>
<h1>CGI Program Error</h1>
<hr>
<h3>$keyword</h3>
$message<br><br>
Please contact $webmaster for more information.
<hr>
</body>
</html>

End_of_Error

  exit( 1);
}


##############################################################################
sub parse_form_data
{
  local( *FORM_DATA)= @_;
  local( $request_method, $query_string, @key_value_pairs,
         $key_value, $key, $value);

  $request_method= $ENV{'REQUEST_METHOD'};
  if( $request_method eq "GET")
  {
    $query_string= $ENV{'QUERY_STRING'};
  }
  elsif( $request_method eq "POST")
  {
    read( STDIN, $query_string, $ENV{'CONTENT_LENGTH'});
  }
  else
  {
    return_error( "Server Error",
                  "Server uses unsupported method");
  }

  @key_value_pairs= split( /&/, $query_string);
  foreach $key_value( @key_value_pairs)
  {
    ($key, $value)= split( /=/, $key_value);
    $value =~ tr/+/ /;
    $value =~ s/%([\dA-Fa-f][\dA-Fa-f])/pack ("C", hex( $1))/eg;

    if( defined( $FORM_DATA{$key}))
    {
      $FORM_DATA{$key}= join( "\0", $FORM_DATA{$key}, $value);
    }
    else
    {
      $FORM_DATA{$key}= $value;
    }
  }
}
```

Appendix E
SIP Log File Perl Script

```
open LOGFILE, "e:\\sipnt\\sipdata\\log.txt" or return_error( "Couldn't Open File");

$totalhits= 0;
$authyes= 0;
$autno= 0;
%addr= ();
%action= ();

while( <LOGFILE>)
{
  #       auth          user        action      timedate     host
  if( /^\s*(.*)\s*--\s*(\w+)\s*--\s*(\w*)\s*--\s*(.*)\s*--.*REMOTE_ADDR=([a-zA-Z0-9\.]*)/)
  {
    ( $auth, $user, $act, $timedate, $host)= ( $1, $2, $3, $4, $5);

    #total number of hits on the server
    $totalhits++;

    #authorization okay and failed
    if( $auth =~ /Authorization Okay/)
    {
      $authyes++;
    }
    else
    {
      $authno++;
    }

    #addresses of sip users
    $addr{$host}++;

    #actions the server executed
    if( $act =~ /^LOGIN$/)
    {
      $action{'Login'}++;
    }
    elsif( $act =~ /^runsimulation$/)
    {
      $action{'Run Simulation'}++;
    }
    elsif( $act =~ /^remotecirfile$/)
    {
      $action{'Edit Cir File'}++;
    }
    elsif( $act =~ /^remoteoutfile$/)
    {
      $action{'Edit Out File'}++;
    }
    elsif( $act =~ /^remotecirfiledelete$/)
    {
      $action{'Delete Cir File'}++;
    }
    elsif( $act =~ /^remoteoutfiledelete$/)
    {
      $action{'Delete Out File'}++;
    }
    elsif( $act =~ /^plotdata$/)
    {
      $action{'Plot Data'}++;
    }
    elsif( $act =~ /^reallyplotdata$/)
    {
      $action{'Really Plot Data'}++;
    }
    elsif( $act =~ /^help$/)
    {
      $action{'Help'}++;
```

```
    }
  }
}

print "Content-type: text/html\n\n";
print "<html><head><title>SPICE Internet Package LOG</title><head>\n";
print "<body BGCOLOR=\"#8FBC8F\" TEXT=\"#000000\">\n";
print "<h2><center>SPICE Internet Package</center></h1>\n";
print "<h3><center><i>Windows/NT</i> Server Statistics</center></h3>\n";
print "<pre>\n";
print "-------------------------------------------------------------------------------\n";
printf "%25s %4d\n", "Total Hits:", $totalhits;
printf "%25s %4d\n", "Authorization Okay:", $authyes;
printf "%25s %4d\n", "Authorization Failed:", $authno;

print "\n-------------------------------------------------------------------------------\n";
print "Actions Performed\n";
print "-------------------------------------------------------------------------------\n";
@tmp= ( "Login", "Run Simulation", "Edit Cir File", "Edit Out File",
        "Delete Cir File", "Delete Out File", "Plot Data",
        "Really Plot Data", "Help");
foreach $key ( @tmp)
{
  printf( "%25s %4d\n", $key, $action{$key});
}

print "\n-------------------------------------------------------------------------------\n";
print "Addresses and Hits\n";
print "-------------------------------------------------------------------------------\n";
@tmp= sort( keys( %addr));
foreach $key ( @tmp)
{
  @a= split( /\./, $key);
  $address= pack( 'C4', @a);
  $name= gethostbyaddr( $address, AF_INET);
  printf "%-15s  %-52s %5d\n", $key, $name, $addr{$key};
}
print "-------------------------------------------------------------------------------\n";
print "</pre></body></html>\n";
exit( 0);



############################################################
sub return_error
{
  local( $message)= @_;
  print <<TheEnd
Content-type: text/html


$message

TheEnd
}
```

Appendix F
SIP Home Page HTML

```
<html>
<head>
  <title>SPICE Internet Package</title>
</head>

<body BGCOLOR="#8B8B83" onLoad= "document.forms[0].password.focus();">

<center>
<img SRC="/~regnier/graphics/sip.gif">
<form ENCTYPE="multipart/form-data" ACTION="/jwr-cgi-bin/sipbin/sip_login"
      METHOD="POST" NAME="sip" TARGET="_top">
  <b>Enter Password</b> <input TYPE="password" NAME="password" VALUE="" SIZE="20">
  <input TYPE="submit" VALUE="Continue">
</form>
</center>
<hr>

<pre>
  Developed by:
    <b><a HREF="http://nn.uwyo.edu/index.html">Bogdan Wilamowski</a></b>
    <b><a HREF="http://w3.uwyo.edu/~regnier/index.html">John Regnier</a></b>
    Department of Electrical Engineering
    University of Wyoming

  Last Revision: <b>23dec97</b>
</pre>
<hr>

</body>
</html>
```