Spring 2002

Mathematica Outside the Lab: Transferring Classroom Material onto the World Wide Web

Rebecca Baker

Carroll College, Helena, MT

Follow this and additional works at: https://scholars.carroll.edu/mathengcompsci_theses

Part of the Science and Mathematics Education Commons

Recommended Citation


https://scholars.carroll.edu/mathengcompsci_theses/87

This Thesis is brought to you for free and open access by the Mathematics, Engineering and Computer Science at Carroll Scholars. It has been accepted for inclusion in Mathematics, Engineering and Computer Science Undergraduate Theses by an authorized administrator of Carroll Scholars. For more information, please contact tkratz@carroll.edu.
Mathematica Outside the Lab: Transferring Classroom Material onto the World Wide Web

By

Rebecca Baker

This thesis for honors recognition has been approved for the Department of Mathematics, Engineering, and Computer Science, Carroll College, Helena, Montana.

May 1, 2002
Table of Contents

Table of Figures ................................................................................................................. ii
Acknowledgments ............................................................................................................. iii
Abstract .............................................................................................................................. iv
Background ......................................................................................................................... 1
Mathematical Tools on the Internet .................................................................................... 3
  Formatting ......................................................................................................................... 3
  Calculations ....................................................................................................................... 7
The Decision: webMathematica ......................................................................................... 9
Making webMathematica Work .......................................................................................... 11
  The Web Server ............................................................................................................... 11
  Installing webMathematica .............................................................................................. 12
Web Page Created for Carroll College .............................................................................. 13
How a Live webMathematica Page Works ......................................................................... 16
The Code ............................................................................................................................ 18
  HTML Code ..................................................................................................................... 18
  Mathematica Code .......................................................................................................... 19
  webMathematica Code ................................................................................................. 20
  Explanation and Comparison ......................................................................................... 21
Conclusions and Recommendations ................................................................................... 23
Cited References ............................................................................................................... 24
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1:</td>
<td>Internet growth rate from 1969 to 2001 (logarithmic scale)</td>
<td>1</td>
</tr>
<tr>
<td>Figure 2:</td>
<td>HTML code for displaying mathematics</td>
<td>3</td>
</tr>
<tr>
<td>Figure 3:</td>
<td>Content MathML code</td>
<td>4</td>
</tr>
<tr>
<td>Figure 4:</td>
<td>Presentational MathML code</td>
<td>5</td>
</tr>
<tr>
<td>Figure 5:</td>
<td>Example of an equation</td>
<td>7</td>
</tr>
<tr>
<td>Figure 6:</td>
<td>A portion of the Web page before evaluation</td>
<td>14</td>
</tr>
<tr>
<td>Figure 7:</td>
<td>A portion of the Web page after evaluation</td>
<td>14</td>
</tr>
<tr>
<td>Figure 8:</td>
<td>How <em>webMathematica</em> works</td>
<td>17</td>
</tr>
<tr>
<td>Figure 9:</td>
<td>Example of a default value</td>
<td>18</td>
</tr>
<tr>
<td>Figure 10:</td>
<td>Form action attribute</td>
<td>18</td>
</tr>
<tr>
<td>Figure 11:</td>
<td>Beam analysis <em>Mathematica</em> code</td>
<td>19</td>
</tr>
<tr>
<td>Figure 12:</td>
<td>Beam analysis <em>webMathematica</em> code</td>
<td>20</td>
</tr>
<tr>
<td>Figure 13:</td>
<td>Structure of a Mathlet</td>
<td>21</td>
</tr>
</tbody>
</table>
Acknowledgments

Many people helped make this thesis possible. I’d first like to thank my thesis director, Dr. John Scharf. If not for his support, time, and patience, this project would not have been possible. Thank you to Mike Hawks and Brett Penner for helping with the Web server installation. Without their expertise and assistance, the Web server might never have been working. Thank you to Chris Jones for letting me use his beam analysis Mathematica code. Thank you also to my readers, Joan Stottlemyer and Mark Parker, for their patience, expertise, and assistance.

I would especially like to thank my parents for teaching me determination, perseverance, and that you should never give up, no matter how impossible the task may seem. I’d like to thank the many friends that listened to me whine and complain one day when something was not working, and then cheered with me the next when the problem was solved. Most importantly, I would like to thank my fiancé, Mark Nance, for his love and support. His calm assurances and confidence in me gave me the strength to carry on when I thought I could not.
Abstract

The Internet has greatly expanded over the last decade, yet there are few options for people wishing to create interactive Web pages that can perform computations and neatly format mathematical expressions. Many educators wish to make classroom labs and exercises available online, without hindering students with more code than necessary. In addition to educators, many companies (such as Analytic Cycling) have a need to include interactive calculators on their Web pages. However, current mathematical tools are lacking in one or more areas. Common problems with current mathematical tools include the following:

- Verbose and non-intuitive code
- Lack of supporting software
- Little or no computational ability
- Little or no formatting capabilities

After analyzing a variety of available options, the decision was made to use Wolfram Research’s *webMathematica* to demonstrate the viability of converting a typical classroom lab into an interactive online experience. Once the *webMathematica* software was installed, it was used to create a beam analysis Web page. The Web page takes a variety of inputs from the user and then solves a differential equation, thus analyzing the beam. The output includes graphs and equations for displacement, rotation, moment, and shear. *webMathematica* is not only capable of performing complex computations with a minimal amount of code, it also provides a variety of formatting options. The developer can choose to have the results displayed as images or plain text, depending on the intent of the Web page.

It is concluded that *webMathematica* has a very feasible niche in the Carroll College community, but that resources will need to be allotted before *webMathematica* can be used on a wide-scale basis. A faster Web server is needed to handle the traffic generated when several people attempt to access the server simultaneously. In addition to the funds needed to purchase the new server, staffing time will be required to install, configure, and maintain the new server and the *webMathematica* software.
Background

The first “Internet” was born in 1965 when a computer at MIT’s Lincoln Laboratory in Lexington, Massachusetts made a connection to another computer at the System Development Corporation in Santa Monica, California, via a dedicated phone line [1]. In 1969, scientists created a network spanning three universities in California and one in Utah [2]. The purpose of this network, ARPANet (Advanced Research Projects Agency Network), was to advance the spread of technology during the Cold War. Since its early beginnings in the 1960s, the Internet has grown to enormous proportions, as shown in Figure 1. By January of 2001, over one hundred million computers were registered on the Internet [1]. To be registered on the Internet, a computer must have an IP (internet protocol) address.

![Internet Growth](image)

**Figure 1: Internet growth rate from 1969 to 2001 (logarithmic scale)**

Because of the Internet’s quick advancement, new markets, legal issues, and opportunities have developed. Businesses thrive on the Internet, making it sometimes appear to be one giant billboard of advertisements. Pornographers have a completely new media to spread their wares, allowing explicit material to be seen by everyone, unfortunately even by children. The music industry that once gloried in the Internet’s selling advantages now cringes as the Internet’s anonymity and ability to spread information easily causes music sales to drop.
As the Internet has grown, it is surprising that this medium, created by scientists for scientists, has no interactive protocols for handling mathematical information, such as symbols and calculations. Where do the mathematicians turn when they need to find an equation or a proof? All they have are complicated and difficult-to-read tables or static images. Where do the mathematicians, scientists, and engineers go in order to perform computations on the Internet? What language do they have to make their work dynamic and easily transmittable?
Mathematical Tools on the Internet

There are currently several tools available to implement mathematical ideas on the Internet. These tools have been broken into two categories: tools that format and tools that do calculations.

Formatting

There are currently several tools available to display mathematical symbols on the Internet. These tools include the following: Hypertext Markup Language, Mathematical Markup Language, and $LaTeX$-to-HTML converters.

Hypertext Markup Language

Hypertext Markup Language (HTML) is the language of the Internet [3]. This language creates Web pages that are viewed by computers connected to the Internet. HTML uses a series of tags to instruct the Web browser how to format text and images. Because HTML was created to display text and graphics, it does not lend itself very easily to the formatting of mathematical equations. Below is an example of how to code the simple equation $y = \frac{x(x^2 - 1)}{x}$ in HTML.

```html
<table>
  <tr align="middle">
    <td>&nbsp;</td>
    <td>x(x<sup>2</sup> - 1)</td>
  </tr>
  <tr align="middle">
    <td>y =</td>
    <td>-x</td>
  </tr>
</table>
```

Figure 2: HTML code for displaying mathematics
Mathematical Markup Language

The World Wide Web Consortium (W3C) has worked with several individuals and groups of people to develop a language called MathML (Mathematical Mark-up Language) [4]. MathML is solely a formatting language, meaning that it does not do any computations. MathML consists of two independent sub-languages: content MathML and presentational MathML.

Content MathML

Content MathML describes mathematical entities according to what they represent. Tags that describe the context in mathematical terms surround each part of the equation. This concept gives mathematical meaning to the information inside the tags, providing potential for future Web browsers that can evaluate equations. The code for
\[ y = \frac{x(x^2 - 1)}{x} \]
in content MathML would be as follows:

```xml
<math>
  <ci>y</ci>
  <eq/>
  <apply>
    <divide/>
    <apply>
      <times/>
      <ci>x</ci>
      <apply>
        <minus/>
        <apply>
          <power/>
          <ci>x</ci>
          <cn>2</cn>
        </apply>
        <cn>-1</cn>
      </apply>
    </apply>
  </apply>
</math>
```

Figure 3: Content MathML code
Presentational MathML

Presentational MathML describes mathematical entities according to how they should look on the screen. Much like HTML, this language does not “give meaning” to the content; it merely formats the information for display on a Web page. For example, the same expression above, \( y = \frac{x(x^2 - 1)}{x} \), would look like the following in presentational MathML:

```xml
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mi>y</mi>
  <mo>=</mo>
  <mfrac>
    <mrow>
      <mi>x</mi>
      <msup>
        <mi>x</mi>
        <mn>2</mn>
      </msup>
      <mo>-</mo>
      <mn>1</mn>
    </mrow>
    <mi>x</mi>
  </mfrac>
</math>
```

Figure 4: Presentational MathML code

A major problem with content and presentational MathML is its limited support. The number of Web browsers that support presentational MathML is limited, and it is expanding very slowly. Currently, the only Web browsers to effectively display presentational MathML are Mozilla [5], Netscape 6.2 [6] (which incorporates Mozilla technology), and the W3C’s Amaya [7].

The computer algebra system, *Mathematica*, by Wolfram Research Inc. [8] allows a user to paste presentational MathML code into a *Mathematica* notebook. *Mathematica* then recognizes the code as MathML and converts the expression into the corresponding *Mathematica* expression. A user can also write code in *Mathematica*, and then use the “Copy As” option to copy the text as MathML; when the user “Pastes” the copied code in
a word editor, the pasted code is in presentational MathML. As previously mentioned, another disadvantage of MathML is that it does not evaluate mathematical expressions; it is merely a way to format and display mathematics.

At this writing, no applications could handle content MathML. Despite the current lack of technology, however, content MathML has great potential. By describing mathematical expressions in mathematical terms and context, content MathML opens the door for computational use in the future. Perhaps Web browsers will be able to perform computations, or maybe an interface can be created between the Web browser and another application that can perform the computations. The potential is there, but unfortunately the technology is not.

LaTeX to HTML Converters

LaTeX (pronounced “Lay-Teck”) is a free piece of software that helps lay out technical documents by interpreting commands given by the user and formatting the document for publication. LaTeX is able to handle multiple equations better than programs such as Microsoft’s Word, and is much less likely to crash the computer while doing it. LaTeX is the standard for many scientific and mathematical magazines and journals.

Mathematical Equations

In addition to formatting text, LaTeX provides a stable environment for technical documents that contain many mathematical elements such as equations and tables. Common word processors do not handle the formatting needed to insert multiple equations into a document very well. MathType’s Equation Editor (used by Microsoft Word [9]) is a common source of frustration for users who wish to insert equations into their Word documents. These users must suffer through the nearly inevitable crashing of their computer that accompanies multiple equations within a document [10]. Because LaTeX was created for technical papers, it does not have the limitations of MS Word. Below is just one example of an equation that LaTeX can handle with ease [10]:

\[ 6 \]
Because of its professional typesetting capabilities and software stability, *LaTeX* is very popular among scientific journals and publishers who typeset books. To download the full, free version of *LaTeX* or to view the program’s specifications and more examples of its abilities, go to http://www.latex-project.org/LaTeX-home.html.

**LaTeX-HTML Converters**

*LaTeX*-HTML converters, also known as *LaTeX2HTML*, translate documents written in *LaTeX* into HTML code [11]. Both languages format documents by using typesetting tags to identify individual parts of a document. Like HTML, *LaTeX* is a formatting language, not a language that evaluates mathematical functions. When converted, *LaTeX* equations become images, reducing flexibility. To modify an equation on a Web page, there are two options. The first is to retrieve the original *LaTeX* document, modify the document, and then reconvert the document to HTML. The second option is to remove the image from the Web page and retype the equation using HTML code. Both of these methods increase the risk of mistakes in the modifying process, especially if numerous changes are required.

**Calculations**

Two methods of performing computations on the Internet are applets and *webMathematica*. These methods will be discussed in detail below.

**Applets**

Applets are programs written with scripting languages such as *Visual Basic* scripting and Java script. Applets are small programs designed to run within other
applications, such as Web browsers. The INT Media Group, the publisher of an online dictionary of computer terms, states:

Web browsers, which are often equipped with Java virtual machines, can interpret applets from Web servers. Because applets are small in files [sic] size, cross-platform compatible, and highly secure (can't be used to access users' hard drives), they are ideal for small Internet applications accessible from a browser.

Despite these definite advantages, applets have several disadvantages. The code needed to create applets is non-intuitive and verbose. There are Web sites on the Internet that supply applets available for download and for use in Web pages; however, these applets are usually created by an anonymous source with little or no testing. Applets are also very inflexible. The user viewing the applet over a Web browser usually has few options to choose what variables in the mathematical equation can be changed. In addition, applets contain very few capabilities to format mathematical expressions.

**webMathematica**

*webMathematica* is a software system that allows the creation of interactive Web pages by combining with *Mathematica*, traditional Web languages such as HTML, JavaScript and many more. MSP scripting is the name of the language comprised of *Mathematica* commands and a few special *webMathematica* commands. Therefore, the MSP scripting language is a super-set of *Mathematica* commands. The greatest attraction to *webMathematica* is that anything that can be done in *Mathematica* can also be done in *webMathematica*.
The Decision: webMathematica

After all the methods discussed above were researched and analyzed, it became apparent that webMathematica was the best choice for use in the Carroll College Department of Mathematics, Engineering, and Computer Science. For this project the software would be acquired, installed, implemented, and critiqued as to its efficiency in order to judge the feasibility of using webMathematica in the department. The decision to use webMathematica over other available mathematical tools was based on several factors, listed here, and discussed in detail below:

- Carroll’s extensive use of Mathematica
- Computational work done server side
- Additional code minimal and simple
- Clean and flexible formatting capabilities
- Ability to perform complex computations

The fact that webMathematica uses the same commands as Mathematica is a great advantage. In the Carroll College Department of Mathematics, Engineering, and Computer Science, Mathematica is used in nearly all classes. Labs, homework assignments, and projects all utilize the Mathematica software. Therefore, any person who might be interested in learning and using webMathematica will already know a large part of the language.

Another great advantage to webMathematica is that all the computations are done server side. This means that the end-user (the person accessing the Web site) does not need any special software in order to use the Web page. Having the work done server side means that the creator of the Web page does not need to be concerned with the computer of the end-user. Any browser, on any computer, with any type of connection can use a webMathematica page (although slower computers and connections may see a decrease in speed).

As mentioned above, webMathematica contains all of the commands found in Mathematica. However, MSP scripting (the name of the webMathematica scripts used in Web pages) also contains a few additional commands that are specific to Web page
creation. Most of these additional commands are needed when formatting is required to display the Mathematica results on the Web page. These additional commands are intuitive and easy to use. Specific examples of MSP scripting code will be discussed in an additional section.

Unlike applets, webMathematica has the capability to format mathematical expressions easily and neatly. webMathematica has a specific command called “MSPFormat.” With the “MSPFormat” command, results can be displayed in a variety of formats, including presentational MathML. Equations and results can be displayed as text or images. This flexibility provides the Web page designer with much needed formatting flexibility. If the designer believes the results will often be copied and entered into a word processing document, he or she may choose to display the results as text. However, if it is only necessary for the code to look “pretty,” an image will be sufficient.

The main reason why webMathematica was chosen for Carroll College is its amazing ability to compute complex mathematical expressions. webMathematica takes the computational power of Mathematica and puts it on a Web page. This concept enables teachers to put labs on Web pages and students to create projects. Unlike the situation with applets, there is enormous flexibility for both the user and the designer with very little overhead required to learn and implement the language.
Making \textit{webMathematica} Work

There are three main steps necessary to make \textit{webMathematica} work. The first is to install an appropriate Web server. The second is to install the \textit{webMathematica} software on the Web server. The third is to create a Web page using MSP (\textit{Mathematica} Server Pages) scripting.

\textbf{The Web Server}

Wolfram Research has tested and guaranteed that their software will work on a variety of Web servers and operating systems. Software that has been tested and found to be compatible with \textit{webMathematica} is said to be “supported by” Wolfram Research. If a problem should arise, Wolfram Research will be able to provide technical support, but for only supported systems.

A Web server supported by Wolfram Research must first be installed on a supported operating system. Information on the supported software is found on the Wolfram Research \textit{webMathematica} Web site \cite{13}. The operating systems currently supported are Red Hat Linux versions 6.1, 6.2, 7.0, and 7.1, Windows version 95, 98, NT 4.0, 2000, and XP, and Solaris. The servers that currently have tested compatibility are Microsoft’s PWS (Personal Web Server) and the Apache Web server 1.3 or later.

After the Web server is installed, a current version of Java must be installed. The minimum version that is compatible with \textit{webMathematica} is Java 1.3; however, the newest version is recommended. Java can be downloaded free of charge from http://java.sun.com/products/archive/.

After Java is installed, a servlet container (sometimes called an “engine”) must be installed. Supported servlet containers include Apache \textit{Tomcat} 3.2.3, \textit{JRun} 3.0, and \textit{JBoss} 2.2.2 and 2.4.1. When all three components (server, Java and servlet container) are working together, the server is ready for \textit{webMathematica} installation. The Web server configuration used for Carroll College is an Apache Web server running on Red Hat Linux, using Java 1.4, and an Apache \textit{Tomcat} servlet container.
Installing *webMathematica*

There are four steps to installing *webMathematica*. The order of installation is as follows: *Mathematica, J/Link, MSP Application, webMathematica* software. *Mathematica* is responsible for computations, and therefore must be installed and working correctly before any of the other software is installed. *J/Link* is software that connects the *Mathematica* software with the Java software on the computer. The *J/Link* software enables the *Mathematica* kernel to be manipulated from a Java application (the MSP scripts). The MSP application provides support for the launching and closing of the *Mathematica* sessions that are opened when the Web page is accessed. The MSP application “cleans up” the MSP script, passes the *Mathematica* code to the kernel, retrieves the results from the kernel, and then converts them to a format suitable for display on the Web page. The *webMathematica* software runs in the servlet container and is the driver that makes all the other software work together.
There are five steps to creating a webMathematica Web page. The first step is to create a Web page using HTML. If input is desired from the user, a form must be made. An HTML editor such as Microsoft’s FrontPage or Macromedia’s Dreamweaver can be used to generate the HTML code. The second step is to create the Mathematica code. The developer of the Web page must have a licensed version of Mathematica (one is included with the webMathematica software). The Mathematica code should be tested to ensure it is in proper working order before implementing it on the Web page. The third step is to create the webMathematica code. This code will aid in interpreting the Mathematica code and in displaying the webMathematica results. The fourth step is to integrate the previous steps together. The webMathematica code is combined with the Mathematica code, creating an MSP script. The MSP script is then placed in the appropriate place in the HTML code. The fifth, and final, step is to place the finished Web page in the appropriate directory on the Web server. This directory will vary from system to system (see the webMathematica user’s guide for more information).

It was decided that a beam analysis Web page would be created for the Carroll College Department of Mathematics, Engineering and Computer Science. The user would fill in a set of required fields, providing information about a specific beam. The Mathematica code would then find the displacement, rotation, moment, and shear functions for the beam. Below are pictures of the Web page before and after evaluation:
Figure 6: A portion of the Web page before evaluation

Figure 7: A portion of the Web page after evaluation
As can be seen in Figure 3, the user is presented with an aesthetically pleasing Web page with a series of input fields. The Web page was designed so that when the Web page is first opened, each field contains a default value. The default values give the user an idea of what type of information he or she should input.

After the user presses the “Evaluate” button that is directly under the last input field, the process that is described in the following section occurs. The results are displayed as a series of equations and graphs. The rotation, moment, and shear functions and graphs are displayed similarly to the displacement results shown in Figure 4. The Web page was created so that both the equations and the graphs are displayed as images (.gif format). Displaying the equations and graphics as images was done for formatting purposes and to make the Web page “look pretty.”
How a Live *Mathematica* Page Works

Once the Web server is running and the *webMathematica* software is properly installed (including the *Mathematica* software used for computations), *webMathematica* Web pages may be processed. There are six steps that occur when a Web page is accessed. They are described below:

**Step One: The User Accesses the Web Page**

A user accesses the Web page. The user’s Web browser sends a request to the Web server asking the server to send the specified page.

**Step Two: The Web Server Forwards the Request**

The Web server processes the Web page, recognizes the .msp extension (see Step One), and forwards the request to the MSP server.

**Step Three: The MSP Server Opens *Mathematica***

The MSP server connects with the *Mathematica* software and opens a session, which will handle the computations. The session receives any variables or values entered by the user.

**Step Four: *Mathematica* Processes the Web Page**

The *Mathematica* session loads the Web page and processes any information contained within the special Mathlet tags (see Step One). The results are returned to the MSP server.

**Step Five: The MSP Server Passes the Results to the Web Server**

The MSP server receives the results and creates a new Web page, using only HTML, to display the results as specified by the MSP script. Any temporary settings are cleared from the *Mathematica* session, and the session is closed. The MSP server passes the dynamically created Web page to the Web server.
Step Six: The User Receives the Page

The Web server performs any processing needed and returns the Web page to the user. The user receives the Web page, fills in the input boxes used for computation, and after the evaluation button is pressed, the process begins again.

The following flow chart outlines the process described above (the primary source of information comes from http://www.wolfram.com/products/webmathematica/technology/):

![Diagram](image)

Figure 8: How webMathematica works
The Code

As mentioned above, there are three sets of code needed to create a `webMathematica` Web page: HTML, `Mathematica`, and `webMathematica`. The HTML code will not be covered here, except for a small example of form code. The `Mathematica` code needed to analyze the beam is given below, followed by the corresponding MSP script code.

**HTML Code**

The only difference between HTML code for a regular Web page and the HTML code necessary for a `webMathematica` Web page becomes clear when creating a default value for a form. As seen below, a small mathlet using the command “MSPValue” must be added to the value attribute. If the mathlet is not added, the server will not know that the default value is a number rather than a string of characters.

```
Enter the redundant interior support:  <input type="text" name="rsups" align="left" size="50" value="<%Mathlet MSPValue[rsups,\{80,160\}]%>\" />
```

*Figure 9: Example of a default value*

`webMathematica` Web pages have the extension “.msp,” which indicates a file using `Mathematica` Server Page technology. As with any HTML form, it is necessary to use the “action” attribute to specify what the form should do when activated. When using the form with an MSP script, the action attribute of the form is set to the URL that contains the MSP script that is desired. In the case below, the URL is called “math.” Because the MSP script is in the same file as the form, this is a relative URL. The action attribute is then the name of the file, minus the extension.

```
<\form action="math" method="post">\n```

*Figure 10: Form action attribute*
Mathematica Code

The *Mathematica* code in Figure 11 is courtesy of Chris Jones. The code creates a beam analysis function that accepts a variety of input, and displays several functions and graphs related to the beam.

```mathematica
Clear[ic, ei, y, m];
Clear[beamAnalysis, ics, rload, soln, displ];

beamAnalysis[L_, ei_, load_, ics_, rsups_] :=
  Block[{},
    Clear[y, phi, m, v];
    nrs = Length[rsups];
    rload = load + If[nrs > 0, Sum[r[i]*DiracDelta[x - rsups[[i]]], {i, 1, nrs}], 0];
    soln = DSolve[Flatten[{y'[x] == phi[x], m[x] == ei*phi'[x], m[x] == v[x], v'[x] == rload, ics}], {y[x], phi[x], m[x], v[x]}, x];
    displtemp[x_] = soln[[1, 1, 2]];
    If[nrs > 0, eqns = {}, ukns = {};
      Do[AppendTo[eqns, displtemp[rsups[[i]]] == 0];
        AppendTo[ukns, r[i]], {i, 1, nrs}];
      soln2 = Flatten[Solve[eqns, ukns]], soln2 = {}];
    displ[x_] = soln[[1, 1, 2]] /. soln2;
    Print["Displacement -> ", displ[x]];
    Plot[displ[x], {x, 0, L}];
    rot[x_] = soln[[1, 2, 2]] /. soln2;
    Print["Rotation -> ", rot[x]];
    Plot[rot[x], {x, 0, L}];
    shear[x_] = soln[[1, 4, 2]] /. soln2;
    Print["Shear -> ", shear[x]];
    Plot[shear[x], {x, 0, L}];
    moment[x_] = soln[[1, 3, 2]] /. soln2;
    Print["Moment -> ", moment[x]];
    Plot[moment[x], {x, 0, L}] ];

beamAnalysis[240, 29*10^-9, 100*(UnitStep[x - 0] - UnitStep[x - 240]), {y[0] == 0, y[240] == 0, m[0] == 0, m[240] == 0}, {80, 160}]
```

Figure 11: Beam analysis Mathematica code
**webMathematica Code**

The code shown below in Figure 12 is the *webMathematica* version of the *Mathematica* code shown in Figure 11. The highlighted code will be discussed in the following section.

```mathematica
<%Mathlet
 MSPBlock[{\$L,\$ei,\$load,\$ics,\$rsups},
 Clear[y,phi,m,v];
nrs=Length[\$rsups];
 rload=\$load+If[nrs>0,Sum[r[i]*DiracDelta[x-\$rsups[[i]]],{i,1,nrs}],0];
 soln=DSolve[Flatten[{y'[x]==phi[x],m[x]==\$ei*phi'[x],m'[x]==v[x],v'[x]==-
 rload,\$ics}],{y[x],phi[x],m[x],v[x]},x];
 displtemp[x_]=soln[[1,1,2]]; If[nrs>0,eqns={},uksns={}; Do[AppendTo[eqns,displtemp[\$rsups[[i]]]==0];
 AppendTo[uksns,r[i]],{i,1,nrs}];
 soln2=Flatten[Solve[eqns,uksns]],soln2={};
 displ[x_]=soln[[1,1,2]]/soln2;
 MSPFormat[displ[x],TraditionalForm]] %>

<%Mathlet
 MSPBlock[{\$L,\$ei,\$load,\$ics,\$rsups},
 MSPShow[Plot[displ[x],{x,0,\$L}]]]%>

<%Mathlet
 MSPBlock[{\$L,\$ei,\$load,\$ics,\$rsups},
 rot[x_]=soln[[1,2,2]]/soln2;
 MSPFormat[rot[x],TraditionalForm]]%>

<%Mathlet
 MSPBlock[{\$L,\$ei,\$load,\$ics,\$rsups},
 MSPShow[Plot[rot[x],{x,0,\$L}]]]%>

<%Mathlet
 MSPBlock[{\$L,\$ei,\$load,\$ics,\$rsups},
 moment[x_]=soln[[1,3,2]]/soln2;
 MSPFormat[moment[x],TraditionalForm]]%>

<%Mathlet
 MSPBlock[{\$L,\$ei,\$load,\$ics,\$rsups},
 MSPShow[Plot[moment[x],{x,0,\$L}]]]%>

<%Mathlet
 MSPBlock[{\$L,\$ei,\$load,\$ics,\$rsups},
 shear[x_]=soln[[1,4,2]]/soln2;
 MSPFormat[shear[x],TraditionalForm]]%>

<%Mathlet
 MSPBlock[{\$L,\$ei,\$load,\$ics,\$rsups},
 MSPShow[Plot[shear[x],{x,0,\$L}]]]%>
```

*Figure 12: Beam analysis webMathematica code*
Explanation and Comparison

The code that is highlighted in Figure 12 illustrates the differences between the original Mathematica code, Figure 11, and the new webMathematica code, Figure 12. The webMathematica commands separate Mathematica commands from HTML and allow the Web browser to display Mathematica output. The MSP scripting code must be enclosed within special Mathlet tags. The structure of an MSP script is shown below:

```mathematica
<% Mathlet
   Code for computation by Mathematica
%>
```

Figure 13: Structure of a Mathlet

These beginning and end tags notify the Web server where the HTML ends and where script begins. The next line of webMathematica code begins “MSPBlock.” The “MSPBlock” command inspects the following arguments and ensures their validity. If any of the values are null, the “MSPBlock” returns a null, and either an error message is displayed or nothing is displayed. Immediately following the “MSPBlock” command are the variables that will be passed into the function. These variables will be passed to the Mathematica session when the “Evaluate” button is pressed. The variables taken from the form must begin with $$$. The double dollar signs signify that the symbols (variables) are Mathematica symbols and not other symbols from the server (from an image map for example).

The block of code in Figure 12 that begins with “Clear[y,phi,m,v];” and ends with “displ[x_] = soln[[1,1,2]]/.soln2;” is the same as the Mathematica code necessary to analyze the beam. The next difference between the Mathematica and webMathematica codes occurs when the displacement equation is displayed. The Mathematica “Print” command is replaced with the webMathematica “MSPFormat” command. The “MSPFormat” command controls the method by which the equation is displayed. There are several different options other than the TraditionalForm used here. Other options include OutputForm, InputForm, StandardForm, and MathMLForm. The TraditionalForm neatly formats the equation and then turns it into an image. There is an
additional third option that can be used in the “MSPFormat” command and that is the “Type” command. This option allows the equation to be converted into either a .gif image or a .jpeg image.

A quirk of webMathematica is that it dislikes displaying more than one object in a single MSP script. Therefore, in order to display an equation and a graph, it is necessary to have two different MSP scripts. The fact that webMathematica retains the values of the variables makes it possible to display many results and place them in an organized fashion on the Web page.

The final difference between the Mathematica code and the MSP script is the “MSPShow” command. This command is used in conjunction with the Mathematica command “Plot.” The “MSPShow” command formats graphical results for display on the Web page.
Conclusions and Recommendations

Through research and analysis, it was decided that webMathematica would fulfill the desire to easily create neatly formatted, interactive, mathematical Web pages. Through experimentation with the webMathematica software, it was discovered that webMathematica does indeed pick up where other current methods have left off. The code needed to create Web pages is straightforward and minimal. The complexity of computations is restricted only by the limitations of Wolfram Research's Mathematica software. The mathematical formatting is superior to other tools on the Internet and has the additional flexibility of displaying mathematical expressions as images or text, depending on the developers' preference.

It is very feasible for webMathematica to be used at Carroll College, provided that a new Web server is used. The current Web server used to house webMathematica pages, the webMathematica software, and Mathematica is lacking in speed and storage. A new Web server will be able to handle the demands required when an entire class accesses the server simultaneously. Monetary and staff resources will need to be allotted to make this a possibility. Once the new server is purchased, an administrator will need to handle all the installations, configurations, and maintenance required.


