Presented at the 2003 American Control
Conference, Denver, Colorado

Interactive Mathematics on the Web: *MathML* for Signals and Systems Demonstrations

Michael J. Ross and Wilson J. Rugh
ECE Department
Johns Hopkins University
Baltimore, MD 21218

Abstract Mathematical Markup Language (MathML) can be used both to render mathematical expressions and to encode the meaning of mathematical expressions in an HTML document. Thus MathML can be used to incorporate dynamic math content in web documents; for example, math expressions that drive and are driven by graphical displays. We will discuss two interactive demonstrations in basic Fourier analysis that illustrate the capabilities of MathML.

1. Introduction

Since the advent of the World Wide Web, content has been produced using the simple *HyperText Markup Language*, or HTML. In HTML, content is enclosed with "tags" that are used to format text, create tables, control interactive content, load images, and so on. The original specification of HTML included a math tag that was intended to mark content as a mathematical expression. Browser support for the math tag has been slow to develop, and in the interim the technical community has displayed equations using cumbersome image files.

The World Wide Web Consortium (W3C), the standards organization for internet technology, has updated the HTML standard in recent years to include a very specific implementation of the math tag called *MathML*, which is short for Mathematical Markup Language. [http://www.w3.org/math] This specification includes two very different formats for creating math. First is *Presentation MathML*, which focuses on cosmetic aspects of rendering an equation in HTML. [http://www.w3.org/TR/MathML2/chapter3.html] This is an alternative to using image files, and the equations appear exactly the same from browser to browser. There are a variety of tools available to simplify creation of an equation using Presentation MathML, though typically these tools do not generate economical code. [http://www.mathtype.com/en/products/]

The second flavor of MathML is called *Content MathML*, and this language provides the capability to encode the mathematical meaning of an equation or expression. [http://www.w3.org/TR/MathML2/chapter4.html] Content MathML is much more concise and structurally elegant, and we have found it to be more suitable for making dynamic or "live" equations because of the efficiency of the code. Unfortunately, Content MathML currently has very limited tools to assist an author in generating the code, and less browser support than Presentation MathML.

Further details concerning Content and Presentation versions of MathML can be found in the Appendix, and in the following references:
[http://www.mathtype.com/en/reference/webmath/status/],
[http://www.mathmlconference.org/2002/],
[http://www.charlesriver.com/titles/mathml.html]

2. Application of Content MathML – Two Examples

This paper focuses on recent efforts to use Content MathML in conjunction with JavaScript and Java applets to create interactive demonstrations for basic signals and systems topics. We will discuss two examples, both of which require the use of Microsoft Internet Explorer 5.5+ with the MathPlayer plugin on a Windows PC. (This lack of portability is discussed in Section 4.) In both examples the presentation portions of the demonstration are rendered in Presentation MathML that was generated from a Microsoft Word document using MathType software. The interactive portions of the demonstrations use Content MathML in conjunction with Java and JavaScript.

The first example, completed in Fall, 2002, is the demonstration *Discrete Time Fourier Series*, URL *http://www.jhu.edu/~signals/dtfs-mathml4/newindex.htm* . The user can input the coefficients in a Fourier series expression for a discrete-time, periodic signal, and then generate the corresponding signal and its magnitude and phase spectra with a mouse-click. Alternatively, the user can input the spectra graphically with the mouse and then generate the corresponding Fourier series expression and the signal, or input the signal graphically with the mouse and then generate the corresponding Fourier series expression and spectra. The heart of the demonstration is a Java applet that accepts user input and generates the graphical displays and/or the MathML character strings that are passed to JavaScript for equation display on the page.

The second example, completed in Spring, 2003, is the demonstration *Continuous-Time Fourier Transform Properties*, URL http://www.jhu.edu/~signals/ctftprops-mathml/index.htm . The user selects from a menu of typical signals, and the corresponding magnitude and phase spectra are shown, along with their mathematical expressions. Then the user selects from a menu of operations on the signal, such as time shift, or differentiation, and the results of this operation on the time signal and its spectra are shown, again with the corresponding mathematical expressions.

The objective in both demonstrations is to tightly couple the mathematical expressions with graphics and visualization tools to facilitate student comprehension of both the intuitive and mathematical aspects of the topic. The extent to which this objective is achieved is an important issue that remains to be addressed. However, it is likely that a substantial suite of material with interactive math capabilities will be needed before a convincing conclusion can be reached.

3. Structure of the CTFT Properties Code

We will briefly describe the code implementing the Fourier transform properties demonstration. In the Java portion, the main applet is MathContTime.java. This applet listens for changes in the HTML drop-down boxes. If the signal *x(t)* has been changed, the display window and corresponding spectra are updated. If the operation type or modifier value have been changed, the *y(t)* display and corresponding spectra are updated.

To handle the signal representations, Signal.java contains the code to generate any of the "ordinary" signals and the code to implement the time-domain operations on the signals. SimpleSignal.java is an empty class of which Impulse and Doublet are subclasses. These handle the special manipulations and displays required for the two generalized signals that occur in the demonstration. MagSpecLibrary.java generates the magnitude spectra for ordinary signals and PhsSpecLibrary.java generates the phase spectra.

Graphical display is handled by OutputGraphPanelWrapper.java. In the main applet, all OutputGraphPanel instances are separate applets. Instances of the wrapper class are held in the main applet and form links to WebOutputGraphPanel applets. Operation relies on calling getAppletContext() and getApplet() through the MathContTime applet. Since there may be differences in the initialization time for each applet on the page, there is a timer that reattempts contact with the designated WebOutputGraphPanel applet if it is not initially found. For more information on this potential problem area, see [http://java.sun.com/docs/books/tutorial/applet/appletsonly/iac.html].

WebOutputGraphPanel.java is a small applet that allows an OutputGraphPanel to function outside of a mother applet, though the applet still relies on another applet to send data via the wrapper class. Finally, OutputGraphPanel.java draws graph labels and calls code from GraphPanel.java to draw everything else.

In the JavaScript interface, the initialization code prepares the left-hand side of equations and executes user-defined functions clearXStrings() and clearYStrings() to set *x(t)*-related signals to zero and *y(t)*-related signals to "undefined." MathML representations for *x(t)*-related signals, based on the nature of the active signal, are produced by setXStrings(). MathML representations for *y(t)*-related signals are produced by setYStrings() based on the nature of the active signal, the active signal operation, and the active modifier value.

In the Web page, applets must have the MAYSCRIPT property set to "true" to permit communication from JavaScript to Java.
[http://developer.netscape.com/docs/manuals/communicator/jsguide4/livecon.htm]

The MathContTime applet appears as a slider and controls the modifier value. When messages are received from the drop-down boxes, setXStrings() and setYStrings() are executed, changing the MathML fields. Also, MathContTime changes the signal to be graphed in the six output panels. The WebOutputGraphPanel applet appears as a graph and accepts commands from MathContTime. The signal type drop-down box executes setActiveSignalType() in MathContTime and resets the active operation type to "No operation." The operation type drop-down box executes setActiveSignalOperation() in MathContTime.

4. Portability Issues

Static mathematical expressions can be displayed with Presentation MathML in either Internet Explorer or Mozilla/Netscape by using W3C stylesheets. These stylesheets determine which browser is running and which method will be used to render MathML. The stylesheets also allow Mozilla/Netscape to display Content MathML – not something for which native functionality is provided. From the user's viewpoint, there is some overhead in that Mozilla/Netscape requires that proper math fonts be installed, and there is not yet a standard set of math fonts. The situation is simpler with Internet Explorer, where the free MathPlayer download includes a comprehensive set of math fonts.

More important to our interests, it appears that Mozilla and Netscape at present do not support the display of dynamic mathematical expressions via MathML because updates to MathML fields are treated as ordinary text. For demonstrations of the type we have prepared, it is crucial to

be able to update MathML fields as program operation continues. Internet Explorer with the MathPlayer plugin can continue to process MathML after initial loading of the page.

Authors who wish to pursue dynamic MathML development in Mozilla/Netscape should explore the Design Science Java applet that processes both Presentation and Content MathML.
[http://www.mathtype.com/en/products/webeq/#viewer]

5. Performance Issues

The overall performance of a given MathML demonstration is largely based on the architecture decisions that trade added functionality with performance. Our experience indicates that the following should be avoided: excessive redraw requests on Java applets and MathML fields; the passing of large variables from Java to JavaScript; excessive numbers of applets or MathML fields on a single web page; and large downloads to the user. For example, in the Fourier transform properties demonstration, MathML generation is handled by JavaScript so that delays in passing data between Java and JavaScript are avoided. Furthermore, Content MathML is used to provide smaller downloads and, in our experience, faster processing.

6. Appendix

We will describe MathML using the expression:

$$\frac{x+\sqrt[3]{x-5}}{y\sin x}$$

Presentation MathML code is written in the same left-to-right order that the resulting expression is read. Definitions of the tags used in this example are in the following table.

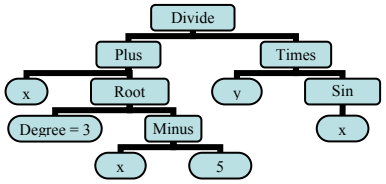
Tags for Presentation MathML

Tag	Usage Notes
<math>	A beginning and ending math tag must enclose the entire expression.
<mrow>	Designates a row of math that the browser will attempt to keep together. In Presentation MathML, mrow is used to enclose multiple tags when only one is expected.
<mfrac>	Denotes a division operation that is written in the numerator-denominator format, instead of inline format.
<mi>	The enclosed data is the identifier of a variable.
<mo>	The enclosed data is an operator.
<mroot>	The first item it encloses is the radicand, and the second is the index.
<mn>	The enclosed data is a number.

Presentation MathML for the expression above is shown below, with comments in footnotes.

```
<math>
   <mrow>
      <mfrac>
         <mrow>[1]
            <mi>x</mi>
            <mo>+</mo>
            <mroot>[2]
               <mrow>
                  <mi>x</mi>
                  <mo>-</mo>
                  <mn>5</mn>
               </mrow>
               <mn>3</mn>
            </mroot>
         </mrow>
         <mrow>[3]
            <mi>y</mi>
            <mo>&InvisibleTimes;</mo>[4]
            <mi>sin</mi>
            <mo>&ApplyFunction;</mo>[5]
            <mi>x</mi>
         </mrow>
      </mfrac>
   </mrow>
</math>
```

Content MathML has a tree structure, with operators as nodes and variables or numbers as the leaves. The highest-order operation is also the root of the tree. The structure and tags for the example are shown below.



Tags for Content MathML

Tag	Usage Notes
<math>	A beginning and ending math tag must enclose the entire expression.
<apply>	Indicates that the next tag is an operator.
<divide/>	Division operator.

[1] This is the row containing the numerator, $x+\sqrt[3]{x-5}$.
[2] The mroot tag requires that there be only two child elements, so the quantity $x-5$ must be a row of its own.
[3] This is the row containing the denominator, $y\sin x$.
[4] "⁢" is a special operator that takes account of the fact that when we write two variable quantities next to each other, separated by a space, we signify that they are being multiplied.
[5] "⁡" is another special operator that connects two adjacent identifiers, such as "sin" and "x" to make "sin x".

<plus/>	Addition operator.
<ci>	The enclosed data is an identifier.
<root/>	A radical expression.
<degree>	A one-purpose tag for specifying the degree of a radical.
<cn>	The enclosed data is a number.
<minus/>	Subtraction operator.
<times/>	Multiplication operator.
<sin/>	Sine operator.

The Content MathML code for the mathematical expressions is:

```
<math>
      <apply><divide/>[6]
            <apply><plus/>
                  <ci>x</ci>
                  <apply><root/>[7]
                        <degree>
                              <cn>3</cn>
                        </degree>
                        <apply><minus/>
                              <ci>x</ci>
                              <cn>5</cn>
                        </apply>
                  </apply>
            </apply>
            <apply><times/>[8]
                  <ci>y</ci>
                  <apply><sin/>
                        <ci>x</ci>
                  </apply>
            </apply>
      </apply

</math>
```

[6] A tag ending with a slash is a self-terminating tag, and thus never has a matching closing tag, unlike other tags in HTML and MathML.

[6] The root tag normally only has one child element (a number, an identifier, or an apply tag, signifying a more complex expression), but the optional degree tag specifies the degree of the radical. The default is square root.