Online Mathematics: The Good, the Bad, and the Ugly

http://teachers.sdmesa.sdccd.net/~mreese/webmath.pdf

Michael S. Reese
San Diego Mesa College
mreese@sdccd.net

December 13, 2000

Abstract

Mathematical expressions displayed online usually look awful. Programmers have tried to rectify this situation, but success has been slow in coming. Even sophisticated mathematical typesetting software such as TEX and LATEX, which are extremely good at producing high-quality mathematics on paper, do poorly on screen. With the growing use of web-based instruction, it is imperative that software be designed to produce visually accurate mathematics on the computer screen.

Introduction

People have been concerned about the appearance of text since ancient times, from calligraphers of the eastern and western worlds, to Johannes Gutenberg and his hot metal press in 1448, to modern-day graphic design specialists such as Edward Tufte. In modern times, we rely heavily on computers to produce text for us, both on screen and on paper. In fact, computer software and hardware are getting to the point where their output on paper rivals the output of traditional mechanical presses. Nevertheless, on-screen text is often displayed poorly, due to low resolution, raster scanning displays, and other factors. Fortunately, enough visual cues abound in English text to make most on-screen text comprehensible. Furthermore, software developers have produced products such as Adobe Type Manager that dramatically improve the quality of on-screen text.

Unfortunately, no such products exist for on-screen mathematics, and because of the multitude of typefaces, font sizes, and symbols involved in producing written mathematics, on-screen mathematical expressions often are extremely hard to read. Some attempts have been made at addressing this problem; however, no solution exists which is completely satisfactory. In the next section of this article, we examine this problem in more detail. The section that follows discusses current attempts at solving the problem, and the penultimate section describes one of the more successful of such attempts. In the last section, we describe the future that we would like to see for the online display of mathematics. A bibliography of related works follow the last section.

The Problem

Back to Introduction

Suppose you are teaching a high school algebra class. You are very excited about using technology to enhance your students' learning, so you decide to put your class notes on the Web for your students to download. Since you usually produce math exams and quizzes using Microsoft Word, you create the quadratic formula with Word and paste it into your web document. When you look at it on the Web, it appears something like this.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

It looks a bit like you drew it with a crayon, but at least it is legible. Next you create a series, also using Word. It looks like this.

$$S_\infty = a_{n_1} + a_{n_2} + a_{n_3} + \cdots$$

It is becoming difficult to see those tiny sub-subscripts. Besides that, it is somewhat of a bother to create mathematical expressions in Word, because you have to stop typing, go to a special Equation Editor window, use the mouse to create the expression by choosing items from menus, and then paste the expression back into your document. If you want to edit the expression, you must go back into the Equation Editor to do so; you cannot make any changes in the actual Word document.

Equation Editor has other problems. It has many mathematical symbols, but not all of them you might need. Also, it would be nice to be able to create your own mathematical symbols. It does not appear that Equation Editor allows this. Equation Editor (or Word) seems to have some unresolved bugs, since your computer crashes sometimes when you are using it. Finally, since equations are treated as embedded images in your word documents, they do not always resize properly if you change the size of your document, and you cannot send them to your colleagues through email except as an attachment.

You decide that Equation Editor is not the answer, so you choose not to upgrade to the new version of Microsoft Word for $329. For the present, you use the "traditional" approach for displaying mathematical expressions on a computer screen.

```
  d                    f(x + h) - f(x)
---- f(x) =    lim    -----------------
 dx           h -> 0          h
```

At least this looks as if it were produced with a typewriter rather than a crayon.

This scenario is not uncommon. In fact, there are software packages that produce better looking on-screen mathematics, but none of them produce expressions approaching the quality of professionally typeset mathematical expressions on paper. Additionally, most of these packages require a good deal of effort to learn to use.

Things must change. More and more instructors are putting their notes, exams, solutions, and homework sets on the Web. Publishers are putting mathematical journals online. Mathematicians want the convenience of being able to send mathematical expressions through email and see these expressions displayed as they would see them in a textbook right in their email message, rather having to print them on a piece of paper first.

Current Solutions

Back to Introduction

Current software packages for displaying mathematics on screen fall into two general categories: What-You-See-Is-What-You-Get (WYSIWYG) editors and markup languages. The former includes packages such as Microsoft Word/Equation Editor, MathType (the "professional" version of Equation Editor), and Corel WordPerfect. WYSIWYG software is sometimes referred to What-You-See-Is-All-You-Get (WYSIAYG) since mathematical expressions produced with this type of software are stored as images and do not respond well to resizing or other visual manipulations. Furthermore, it is difficult to change these expressions, requiring approximately the same effort as was required to create the expressions in the first place. Beyond mathematical expressions, WYSIWYG editors are very useful for producing small, visual documents such as newsletters or brochures, especially when these contain multiple fonts; however, they do not perform well with larger documents such as journal articles, books, or multivolume works.

Markup languages, on the other hand, are excellent for producing large written works. This is because these languages are logically oriented, rather than visually oriented. That is, they require that the user input information describing the relative appearance, logical structure, and meaning of the document, such as emphasized text, sections and subsections, beginning and end of documents, header information, mathematical expressions, figures, and tables. A source file for a markup language document contains not only the text to be output, but also notations, or "tags," that tell the computer how to display the text, somewhat like "marks" a human editor might make on a piece of news copy describing how to include the piece in the next issue of a newspaper. Markup language source files are written in ASCII text, therefore they can be sent via email quite easily.

Markup tags are kept as general as possible. For example, positioning tags perform their action on bodies of text in a relative way, rather than an absolute way. Thus, a tag that centers a headline would ensure that the headline is centered for any width of page being displayed. This is quite easy to see. Simply visit a web page that has a centered title (e.g., the author's home page) and change the size of your browser window. You will see that the title remains centered, no matter what the width of the window.

> The visual representation of an HTML document depends entirely on the browser that is processing the HTML code. Because no HTML editor can possibly anticipate all of the variations in HTML viewing environments, the idea of WYSIWYG HTML editing is fundamentally flawed. (http://www.barebones.com/products/bbedit.html)

Many markup languages exist, but they are all based on TEX. They include SGML, HTML, XML, MathML, Medium-Independent Notation for Structured Expressions (MINSE), Scientific Workplace, and LATEX. While these languages tend to display text on screen quite well, and some of them produce printed mathematics of professional quality, they all have problems when it comes to displaying mathematics on screen. In fact, MathML is an extension of XML (which is an extension of HTML, which is an extension of SGML) which was created precisely to deal with the issues of poorly displayed mathematics on screen.

> MathML is a well-deserved tribute to Mathematics, the "Queen of the Sciences." However, the crowning ceremony is yet to take place. Convergence and collaboration of MathML with other technologies will be the key for a significant mathematical presence on the Web. According to the MathML 1.01 Specification, "the goal of MathML is to enable mathematics to be served, received, and processed on the Web, just as HTML has enabled this functionality for text." This goal is yet to be realized. (http://tech.irt.org/articles/js208/)

Markup languages are called "languages" for a reason: essentially they are computer languages. When you write a file using a markup language, you are programming. Also, you have no immediate visual feedback the way you do with a WYSIWYG editor. Rather, after writing your source file, you must "compile" it, either by processing it with a special program or by opening it with a web browser, such as Netscape. Software packages such as MathType and Scientific Workplace address this problem by incorporating graphical front ends so the user has immediate visual feedback, yet they output markup language code.

In general, markup languages such as MathML produce fairly high quality mathematical expressions on screen, certainly better than Equation Editor or the "traditional" approach. Unfortunately, most browsers do not support MathML or other markup languages. This means that users must convert their fairly high quality mathematical expressions to images in order to display them on the Web, and suddenly all the benefits of using markup languages disappear.

TEX and LATEX

Back to Introduction

> *Don Knuth's Tau Epsilon Chi (TEX) is potentially the most significant invention in typesetting in this century. It introduces a standard language in computer typography and in terms of importance could rank near the introduction of the Gutenberg press.*
>
> —Gordon Bell

In the mid 1970s, Donald Knuth, a highly respected computer scientist at Stanford University, received the galley proofs from the second volume of his multivolume opus entitled "The Art of Computer Programming." As he did with the first volume several years earlier, he had sent his manuscripts to Germany to be printed using the high-quality hot lead presses available at the time. When he looked at the galleys from the second volume, he was shocked. They looked horrible—the typesetting of the mathematics was all wrong, the spacing was off, the mathematical symbols were misshapen. Upon inquiring, Knuth discovered that the experts in hot lead type were slowly disappearing through retirement and death, and computer software was replacing them. Being a stickler for precision in mathematical typesetting, Knuth took it upon himself to create a software package that would replace the dwindling hot lead typesetters.

It took him over five years, but in 1982, Knuth released his software, TEX, to the world. In fact, he donated TEX to the public so that everyone could take advantage of its profound ability to produce professional quality documents from a desktop computer. Today there is no doubt that TEX is the standard for producing typeset mathematics. Mathematicians and scientists have written millions of documents with TEX. By the way,

to pronounce TEX, say *teck*; it comes from the Greek $\tau\epsilon\chi\nu\eta$ *(technē)*, which is the root of such English words as "technical" and "technique." Also notice the special way that TEX is written. On ASCII systems, the form `TeX` is used.

In 1986, Leslie Lamport of Digital Equipment Corporation finished LATEX and gave it to the public as Knuth had done with TEX a few years earlier. LATEXis essentially a set of macros running on top of TEX, but it is much easier to learn and program in LATEX. Notice the specical way that LATEX is written; the ASCII version is `LaTeX`. Acceptable pronunciations are *lay*-teck, *lah*-teck, or lah-*teck*.

The Good

TEX (and LATEX) is a **typesetting program**, not a text processor. In other words, it is a markup language, not a WYSIWYG editor. As in other markup languages (TEX is the original, by the way), you worry about the content of your document and you let TEX handle the look of the document. When Knuth created TEX, he wanted to produce something that would be robust for hundreds of years, so he made the smallest unit of measurement in TEX be less than one-hundredth the wavelength of visible light! It takes 65,536 of these units, called Ridiculously Small Units (RSUs), to equal a single typographical point, and there are 72.27 of the latter per inch. TEX produces output of such precision that microfiche can be typeset directly from TEX, with no photoreduction.

Another feature of TEX is its excellence in typesetting, both for standard text and for mathematical expressions. TEX has excellent kerning and justification capabilities, it automatically provides ligatures (for example, compare "saffron" with "saffron"), and it automatically hyphenates based on the language it is processing.

TEX automatically formats and numbers logical units in your document, e.g., sections, parts, figures, tables, etc. It does this according to an associated style file (e.g., `apa.sty`, `mla.sty`, `ieee.sty`, etc.), hundreds of which have been contributed by TEX users. This means that you can write your document without worrying about the overall style for layout, numbering, citations, and references. After writing the file, if you process it with the APA style file, your document will be produced according to guidelines established by the American Psychological Association. If you select the IEEE style file, your document will be produced according to guidelines established by the Institute of Electrical and Electronics Engineers. Thus, it is extremely easy to change the look of TEX documents. For example, suppose your LATEX source file contains the following code.

```
\documentclass{article}        % Your input file must contain these two lines
\begin{document}               % plus the \end{document} command at the end.

Words are separated by one or more spaces.  Paragraphs are separated by one
or more  blank  lines.  The output is not affected by adding extra spaces or
extra blank lines to the input file.

Double quotes are typed like this: ``quoted text''.  Single quotes are typed
like this: `single-quoted text'.

Long dashes are typed as three dash characters---like this.

Emphasized text is typed like this: \emph{this \emph{is} emphasized}.
Bold       text is typed like this: \textbf{this is bold}.
```

```
If you get too much space after a mid-sentence period---abbreviations
like etc.\ are the common culprits)---then type a backslash followed by
a space after the period, as in this sentence.

Remember, don't type the 10 special characters (such as dollar sign and
backslash) except as directed!  The following seven are printed by
typing a backslash in front of them:  \$  \&  \#  \%  \_  \{  and  \}.
```

Then LaTeX produces

> Words are separated by one or more spaces. Paragraphs are separated by one or more blank lines. The output is not affected by adding extra spaces or extra blank lines to the input file.
>
> Double quotes are typed like this: “quoted text”. Single quotes are typed like this: ‘single-quoted text’.
>
> Long dashes are typed as three dash characters—like this.
>
> Emphasized text is typed like this: *this* is *emphasized*. Bold text is typed like this: **this is bold**.
>
> If you get too much space after a mid-sentence period—abbreviations like etc. are the common culprits)—then type a backslash followed by a space after the period, as in this sentence.
>
> Remember, don't type the 10 special characters (such as dollar sign and backslash) except as directed! The following seven are printed by typing a backslash in front of them: $ & # % _ { and }.

Now if you change the first line of the LaTeX source code from

```
\documentclass{article}        % Your input file must contain these two lines
```

to

```
\documentclass[twocolumn]{article}        \% Your input file must contain these two lines
```

then LaTeX will produce the following.

> Words are separated by one or more spaces. Paragraphs are separated by one or more blank lines. The output is not affected by adding extra spaces or extra blank lines to the input file.
>
> Double quotes are typed like this: “quoted text”. Single quotes are typed like this: ‘single-quoted text’.
>
> Long dashes are typed as three dash characters—like this.
>
> Emphasized text is typed like this: *this* is *emphasized*. Bold text is typed like this: **this is bold**.
>
> If you get too much space after a mid-sentence period—abbreviations like etc. are the common culprits)—then type a backslash followed by a space after the period, as in this sentence.
>
> Remember, don't type the 10 special characters (such as dollar sign and backslash) except as directed! The following seven are printed by typing a backslash in front of them: $ & # % _ { and }.

Because TeX source code is ASCII text, and it runs on virtually every computer platform that exists, TeX code is extremely portable. That is, you can write your TeX

document on any computer and it will print the same, no matter which computer you print it from. Knuth presented TEX to the public in the early 1980s. Having spent so much time in the public domain, TEX is quite bug free, which is like a breath of fresh air to someone who has been using Microsoft products. Since TEX is essentially a programming language, it is extensible; that is, it can be programmed to do exactly what you want it to do. Indeed, a Pascal compiler has been written entirely in TEX. Many macros have been contributed to the public domain by users, including the extremely popular and useful BibTEX, which has many of the same features as the bibliography database software EndNote.

Of course, the best feature of TEX is that it is free. Implementations of TEX are available for the Macintosh and for Windows platforms, and TEX comes preinstalled in all UNIX systems. The American Mathematical Society lists several public domain TEX implementations.

The Bad

Unfortunately, TEX has a notoriously steep learning curve. This was the main reason that Leslie Lamport developed LATEX, but even the latter is nowhere near as simple to learn as most WYSIWYG editors, since it is not interactive and does not provide immediate feedback. To improve the situation, some commercial software manufacturers have come up with graphical front ends, including Blue Sky Research's Textures and Pete Keleher's Alpha for the Macintosh, and Aleksander Simonic's WinEdt for Windows. Several online manuals for TEX and LATEX exist, such as The Not So Short Introduction to LATEX2$_\varepsilon$ and A Gentle Introduction to TEX.

While writing TEX documents requires nothing more than the simplest text editor, processing TEX files is quite RAM intensive and can be slow on older desktop machines. The files required to run TEX eat up a bit of hard disk space as well; OzTEX on the author's Macintosh G4 requires over 56 megabytes of storage space (on the other hand, Microsoft Office 98 requires 180 megabytes on the same machine).

Even for TEX-educated users with RAM-packed, ultra-fast computers, one problem remains: it is not easy to display mathematics on screen with TEX. The author is using LATEX(Blue Sky's Textures) to produce the document you are reading, yet he will have to convert it to an Adobe Portable Document Format (PDF) file before it can be displayed on screen in a form that is accessible to a wide audience. Utilities for conversion from LATEX to HTML do exist, but they perform quite poorly for mathematics of any real level of sophistication.

The Beautiful

TEX is great for typesetting mathematics:

> It is quite counterintuitive that the sets $\mathbb{N}$, $\mathcal{Z}$, and $\mathbb{Q}$ would be equivalent, since it seems that there should be more integers than natural numbers, for example. Such counterintuition often appears when we deal with infinite sets (see Table 1). For example, the sets $\mathbb{R}$, $\mathbb{R}^2$, $\mathbb{R}^3$, and $\mathbb{C}$ are equivalent, but they are much larger than the infinite sets $\mathbb{N}$, $\mathcal{Z}$, and $\mathbb{Q}$. We denote the cardinality of the sets $\mathbb{R}$, $\mathbb{R}^2$, $\mathbb{R}^3$, and $\mathbb{C}$ by the symbol $\aleph_1$. Note that $\aleph_1 > \aleph_0$. For now, ignore the symbols (r, s) and (r, s, t) in the table. They will be defined later.

Table 1: Special Sets in Mathematics

Symbol	Definition	Name	Cardinality
$\emptyset$	$\{\}$	Empty Set	0
$\mathbb{N}$	$\{1, 2, 3, \ldots\}$	Naturals	$\aleph_0$
$\mathcal{Z}$	$\{n : n \in \mathbb{N} \text{ or } -n \in \mathbb{N} \text{ or } n = 0\}$	Integers	$\aleph_0$
$\mathbb{Q}$	$\{q = a/b : a, b \in \mathcal{Z}, b \neq 0\}$	Rationals	$\aleph_0$
$\mathbb{R}$	$\{r : r = \lim_{k\to\infty} r_k, r_k \in \mathbb{Q}\}$	Real Line	$\aleph_1$
$\mathbb{R}^2$	$\{(r, s) : r, s \in \mathbb{R}\}$	Real Plane	$\aleph_1$
$\mathbb{R}^3$	$\{(r, s, t) : r, s, t \in \mathbb{R}\}$	Real 3-Space	$\aleph_1$
$\mathbb{C}$	$\{z = a + bi : a, b \in \mathbb{R}, i^2 = -1\}$	Complex Numbers	$\aleph_1$

TEX can do many wonderful things besides mathematics. For instance, it is well suited for foreign languages and other situations requiring special symbols or accents. The following is a discussion on the quadratic formula in Spanish.

La forma común de la ecuación cuadrática en una variable es

$$ax^2 + bx + c = 0 \tag{1}$$

donde a, b, y c son constantes y x es la variable. Los valores de x que la hacen verídica a Ecuación (1), llamados las *raíces* o las *soluciones* de la ecuación, se pueden obtener por medio de la fórmula cuadrática

$$x_{\pm} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Mañana mostraremos ejemplos del uso de esta fórmula maravillosa.

TEX can be programmed in 16 lines to calculate and output prime numbers:

The first 113 primes are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, and 617.

TEX can organize text into any shape that can be described mathematically.

Textures, the typesetting software for Macintosh computers, is an integrated document preparation package that combines the precision of the TEX typesetting language with the responsive features of the Macintosh—especially its screen display and ability to manage pictures. Hence the name *Textures*—TEX plus pictures. The TEX system—an integral part of *Textures*—is a document formatting language designed by Stanford University's Donald Knuth (author of the five-volume *Computers and Typesetting*) to produce typeset documents such as reports and memos or large, complex books that contain math and scientific notation or ruled forms. It is TEX that transforms a text file into typeset pages, and *Textures* allows you to put this precise page makeup system to use on the Macintosh. *Textures* allows you to perform simple typesetting tasks, such as setting galleys in the manner of traditional typesetting machines, but it is a sophisticated tool better suited for designing a complete book format. With *Textures*, you can create programs that control the arrangement and appearance of the text, the graphic divisions of that text into larger units such as chapters, sections, paragraphs and lists, as well as the placement of tables, charts and other illustrations that are part of your document. At its simplest level, *Textures* allows you to use TEX as a code-based typesetting system that requires no more training than Microsoft Word, for example, in which TEX will only kern, hyphenate and justify your text with optimum line-breaking. But *Textures* is really designed for more complete structures, and it enables you to employ TEX as a programming language that defines formats for your entire text, thus freeing you from point-of-use commands to concentrate on writing and design. The programmed layouts that you create with *Textures* are fully reusable. Once you have decided how you want your document to look, that format can be used again and again—or employed as a base for more elaborate designs. Now that computer-based document production systems encourage format design *prior* to writing the text, the logical design of a document style that *Textures* makes possible works especially well in production settings that include multiple authors, editors, graphic artists, programmers and designers.

The Ugly

Remember Ecuación (1)? To get it and the surrounding Spanish text, we must type the following code into our LATEX source file.

```
La forma com{\'u}n de la ecuaci{\'o}n cuadr{\'a}tica  en una variable es
\begin{equation}\label{eqn}
ax^2+bx+c=0
\end{equation}
donde $a$, $b$, y $c$ son constantes y $x$ es la variable.   Los valores
de $x$ que la hacen ver{\'\i}dica  a Ecuaci{\'o}n~(\ref{eqn}), llamados
las \emph{ra{\'\i}ces} o las \emph{soluciones} de la ecuaci{\'o}n, se
pueden obtener por medio de la f{\'o}rmula cuadr{\'a}tica
$$x_\pm=\frac{-b\pm\sqrt{b^2-4ac}}{2a}.$$
Ma{\~n}ana mostraremos ejemplos del uso de esta f{\'o}rmula maravillosa.
```

Not fun, unless you are a veteran TEXnical writer. Fortunately, help is available to those who wish to learn TEX or LATEX. Online resources such as the Comprehensive TEX Archive Network (CTAN) offer a wealth of free information to the inquiring user, and many cities have TEX Users Groups (TUGs) that are full of people willing to help the new user learn TEX. The bibliography lists several good books on TEX.

One popular solution for creating TEX documents, modeled after a similar procedure in computer programming, is to copy someone else's TEX code and modify it to do what you want it to do.

A Desirable Future

Back to Introduction

Web-based teaching and learning are growing daily, creating a rising demand for the ability to display mathematical expressions accurately on computer screens. Even beyond the proper display of mathematical content on the Web, computers contain an untapped potential for making mathematics come alive for students, as Miroslaw Majewski explains.

> HTML documents on the web are quite static. We can read them like a paper book. We can improve them by adding some Java applets and other active elements that will display animations of graphs and visualization of some concepts, however the major part will still be just a read-only document.
>
> At the same time mathematics is very dynamic. While teaching mathematics we transform formulae, solve equations, plot graphs, etc. Imagine that we downloaded a mathematical web page. In order to change any formula on this page we have to develop it in a computer graphics program, save as a computer file and then embed it into our document. We would have to do the same with graphs, diagrams and other elements.
>
> Now imagine quite a different situation. After downloading a mathematical web page, the student can recalculate each part of this document in the web browser. He can change coefficients of equations and explore alternative solutions. He can plot graphs of modified functions or animate them to see the shape of curves and surfaces better. Finally, he can take the whole document, add the solutions of the enclosed problems to it and post the final document to his instructor for grading.
> (http://www.iium.edu.mo/mirek/articles/beirut/beirut1.htm)

Majewski identifies the three most important features of online mathematics for teaching as follows.

1. The ability to produce dynamic content for the Web.
2. The ability to use formulæ for the coding of graphs of functions and relations, rather than using images in GIF or JPG format.
3. The ability to produce online exams, mark them, and submit scores to the database on the web server.

Although we are not quite there, things are getting close. Chariot Software Group's MicroGrade has the ability to post grades on the Web. The LATEX class exam.cls can produce multiple versions of exams, randomizing question order or which questions are posed. Through a combination approach utilizing TEX and MathML, Design Science's WebEQ has the ability to create "dynamic web pages" with the following features.

- Equations can display messages, or highlighting part of an equation when the mouse moves over a specific part of an equation.
- Equations can have links to other HTML pages from specific areas, like an image map.
- Equations can toggle between two different displays when the reader click the mouse, e.g. to display an answer, or reveal additional terms in a sequence.
- Editable equations can let users modify or enter math expressions.
- Equation can update when readers change values in a form, or click on buttons in a page.
- Equations can be updated in real time by other active components in a page, such as an applet running a simulation, or an input control connecting to a computation engine on a server.

Because of the Web, teaching and learning no doubt will look quite different in five years. For the sake of all those struggling math learners and exacting mathematicians out there, let's hope displayed mathematical content does as well.

Bibliography

Back to Introduction

Abraham, P. W. (1990). *TeX for the impatient.* Reading, MA: Addison-Wesley.

Borde, A. (1992). *TeX by example: A beginner's guide.* Orlando, FL: Academic Press.

Buerger, D. J. (1990). *LaTeX for engineers and scientists.* New York: McGraw-Hill.

Doob, M. (Ed.). (1993). *TeX: Starting from 1.* New York: Springer-Verlag.

Eijkhout, V. (1992). *TeX by topic.* Reading, MA: Addison-Wesley.

Goosens, M., Mittelbach, F., & Samarin, A. (1994). *The LaTeX companion.* Reading, MA: Addison-Wesley.

Grätzer, G. (2000). *Math into LaTeX* (3rd ed.). Boston: Birkhauser.

Knuth, D. E. (1986). *The TeX book.* Reading, MA: Addison-Wesley.

Kopka, H., & Daly, P. (1993). *A guide to LaTeX.* Reading, MA: Addison-Wesley.

Lamport, L. (1994). *LaTeX, a document preparation system* (2nd ed.). Reading, MA: Addison-Wesley.

Majewski, M. (1998). HTML approach in publishing mathematics on the WWW. *Papua New Guinea Journal of Mathematics Computing and Education*, *4*, 53–63.

Salomon, D. (1995). *The advanced TeX book.* New York: Springer-Verlag.

Seroul, R., & Levy, S. (1991). *A beginner's book of TeX.* New York: Springer-Verlag.

Snow, W. (1992). *TeX for the beginner.* Reading, MA: Addison-Wesley.

Spivak, M. (1990). *The joy of TeX* (2nd ed.). Providence, RI: American Mathematical Society.