MathTran and TEX as a web service

Jonathan Fine
LTS Strategic, The Open University
J.Fine@open.ac.uk
http://jonathanfine.wordpress.com

Abstract

MathTran, as a public web service, together with suitable client-side JavaScript, the possibility of an easily deployed solution for mathematical content in mainstream web pages.

The MathTran project is now focussed on the authoring of mathematical content. It has produced a prototype instant preview document editor. Funded by the 2008 Google Summer of Code, Christoph Hafemeister is developing JavaScript to provide autocompletion for commands and analysis of TEX errors, all integrated with an online help system embedded in the web page. Separate work is focussed on developing MathTran plugins for WYSIWYG editor web-page components.

1 Introduction

The MathTran project has developed in response to needs, funding opportunities and some underlying goals and principles. Mostly, they have been pulling in the same direction. The influences include:

- Respond to practical needs, particularly those of institutions such as the Open University
- Use TEX (or a variant of TEX) for mathematical typography
- Simplify use by providing a stateless web service
- Use JavaScript to improve the user experience
- Focus on the author and end-user experience
- Provide components for others to use

Web pages are probably both the most difficult and important of all electronic media. The main focus of the MathTran project is to provide both public web services that help with math on web pages and to develop open source software that provides such services.

The MathTran project started in late 2006, with funding from JISC and The Open University. Its main goals were to provide as web services translation of mathematical content from TEX to images, and from TEX to MathML and vice versa, and from MathML to images. In early 2007 the TEX to image web service was running and, in response to requests from JISC, resources were devoted to web browser client software (in JavaScript) instead of MathML translation.

In 2006/7 I developed and set up the public MathTran web service `www.mathtran.org`. This was done with funding from JISC and the Open University. It provides translation of TEX-notation formulas into high-quality bitmaps. In April 2008 it served 48,000 images a day for a growing range of sites. After tuning, the server could provide about 2 million images a day. It takes about 10 milliseconds to process a small formula.

The JISC funding came to an end in August 2007. Since November 2007 development of MathTran has been done as part of the author's COLMSCT project, and in addition Christoph Hafemeister is doing 3 months work as part of the 2008 Google Summer of Code. Other contributions include a 'TEX Anywhere' component to the Enso Windows desktop assistant and a proof-of-concept client for ReportLab.

At present MathTran is serving about 1 million images a month. With some simple tuning, it could be serving 2 million images a day. The largest identifiable group of users are Moodle sites in several countries.

2 Pragmatic principles

In the course of the project, the following principles emerged:

1. Relate to authors, users and developers where they are.
2. No special or excessive server or client requirements for the use of MathTran.
3. Use TEX as the standard for mathematical notation.
4. Use TEX to perform mathematical typography.

5. Be agnostic about the user's preferences and the future range of display options.
6. Ignore semantic and structural aspects of mathematical notation.
7. Sharing code andother resources is important.
8. Favour lightweight solutions.
9. Adopt a neutral attitude to MathML.
10. Develop and provide widely used services.

There is a widespread assumption, that TEX is too difficult for most users to learn. The project does not adopt this assumption, and might prove it to be false, by providing effective and focussed tools for learning TEX.

3 Mathematics on web pages

To put a formula on a web page, the user must specify the formula. This cannot be avoided, and anything is overhead that we seek to reduce. Many people still create web pages by writing raw HTML, and for these users a syntax such as

```
<img alt="tex:x^2+y^2=1" />
```

is about as simple as it can get. This syntax, incidentally, requires the user to meet the accessibility requirement of providing alternative text for the image.

Adding the `mathtran_img.js` script to the page head automatically adds to all such image tags a `src` attribute, which then causes the browser to fetch the corresponding formula ($x^2 + y^2 = 1$) from the MathTran server.

More sophisticated output (such as MathML) will be possible with only minimal changes, namely a change to the included JavaScript file, once additional services such as TEX-to-MathML translation is available.

4 TEX Standards

TEX, as is well-known, allows its users to define their own commands, by using its macro expansion language. LATEX is a very large and widely used collection of TEX macros, that provides a front end to the use of TEX. Without some macros, TEX is not usable. Don Knuth, that author of TEX, provided plain TEX, which is about 1,200 lines. The kernel of LATEX is about 8,000 lines (not counting comments). The popular `amsmath` package is 2,670 lines (again without comments).

Provided users don't define their own macros, and are reasonable in their use of (La)TEX, machine translation of their source document to, say, XML and MathML is possible. Wikipedia uses a special program `texvc` (written in Objective Caml) to validate and check the LATEX notation mathematics it stores. If authors can define their own macros, then machine translation is much harder (and copy-and-paste to another LATEX document can be virtually impossible).

One of the attractions of MathML is that it is a standard in a way that LATEX is not. However, it is scarcely possible to author MathML directly, whereas LATEX can be authored in this way.

MathTran provides a well-defined collection of TEX commands for the author to use, while at the same time preventing the definition of new commands. (In addition to `\def`, it prevents access to many other primitive commands, such as `\csname` and `\input`).

It does this by creating a secure variant of plain TEX, where all the private commands are stored in locations which the ordinary user cannot access. As plain TEX is only 1,200 lines, this only took a couple of days. Creating a secure variant of LATEX would be much more work.

MathTran does not catch every improper use of TEX. For example, the TEX command `\ne` (for not equals) corresponds to a composite of `=` and a cancelling solidus `/`. As a result, `x_\ne` produces $x_{/=}$. To get the expected $x_{\neq}$ one must write `x_{\ne}`.

5 Image standards

MathTran images contain rich metadata, including the TEX source for the image and the `dvi` and `log` outputs due to that source. This makes it straightforward to edit and resize such images, or convert them to another format, such as SVG or PostScript.

6 Other solutions

There are other solutions. Here we describe four of them and indicate how they fit in with the MathTran project.

6.1 MimeTeX

Written by John Forkosh, this is a C-program that generates bitmaps from, roughly speaking, a subset of LATEX formulas. Deployment is simple — just compile the source and place the binary in the cgi-bin directory. MathTran copied from mimeTeX the query string interface. MimeTeX is quick, and is widely used in the Moodle community and elsewhere.

The MathTran server avoids some problems in MimeTeX:

1. MimeTeX does not return error messages, even when the input is wrong, such as `\frac{1}` or `\integral`.

2. MimeTeX does not implement all of LATEX mathematics, and adding new features requires writing C-code.
3. MimeTeX only approximately follows the mathematical typography algorithms used by TeX. Sometimes the output is different (and worse).
4. It is not possible to convert MimeTeX output to vector formats, such as PostScript and SVG.

6.2 JsMath

Written by Davide Cervone, this produces the best looking output, and in particular the output scales. It consists of client-side JavaScript that implements:

- A parser for LATEX syntax formulas
- An approximate implementation of TEX's mathematical typography algorithms
- A renderer for this typeset output, that uses the web-page as a canvas

It produces good output because it uses, when available, the TeX math fonts as outline fonts. (If not available, it substitutes bitmaps.) In many cases the formulas looks just as good in the browser as they do in a PDF viewer.

Its weaknesses are

1. It is slow on large pages: JavaScript is not a suitable language for building such a system
2. Copy-and-paste does not give good results
3. JsMath does not implement all of LATEX mathematics, and adding new features requires writing JavScript.

MathTran would like to use the parser and renderer components. A JavaScript parser of TEX/LATEX could be used as a component in client side translation to MathML. It could also be used to provide a tree-view of the formula, which could help users both understand and navigate the formula they are editing.

The renderer could be used to display, to the same high quality as JsMath, the typeset `dvi` produced by MathTran. This composite would provide, in many circumstances, the best of both systems: the full repetoire of TEX and the excellent display of JsMath.

6.3 AsciiMathML

Written by Peter Jipsen, this is web-browser JavaScript that translates mathematics in a custom syntax into MathML. It requires a MathML enabled browser. The custom syntax is, loosely, based on LATEX. One of its unusual features is that `oo` translates to ∞.

AsciiMathML comes with an instant preview `textarea` editor, which inspired MathTran to do something similar.

Some of the problems with AsciiMathML are:

1. Requires MathML-enabled browser, with suitable fonts.
2. Has its own input syntax.
3. Does not give error message for incorrect input.

AsciiMathML provides, roughly speaking, a client-side TEX-to-MathML translator. Something that did this with TEX syntax and proper error reporting would be very useful. It could be used, for example, as a drop-in alternative to the `mathtran_img.js` script previously referred to, for users that prefer MathML.

7 MathML

MathML is a W3C recommendation for mathematics on web pages. Despite that, it is not widely supported. FireFox 2 supports it natively, but requires an additional download of option fonts. Internet Explorer supports it only with the third-party MathPlayer plugin from DesignScience. (Not all W3C recommendations are successful. PNG is very successful, and SVG moderately so. Almost no-one has heard of or uses the InkML language for handwriting. MathML is more successful than that, but not as successful as SVG.)

One of the big advantages of MathML is that screen-reader software and other accessibility tools exist for it, whereas they do not for TEX-notation mathematics.

8 JavaScript programming

It is JavaScript that allows sophisticated display and editing of content on web pages. The present MathTran JavaScript amounts to perhaps 1,000 lines of JavaScript code, together with use of the jQuery JavaScript library. This is quite modest. For example, `jsmath.js` is almost 6,400 lines (which implements an approximation of LATEX to TEX-typeset mathematics). AsciMathML is 3,500 lines of code. MathDox uses over 26,000 lines of JavaScript (to provide a GUI programming environment for mathematics in JavaScript).

This author has found JavaScript to be a difficult programming environment, not least because of major differences in how the various browsers implement JavaScript and provide an interface to the browser's Document Object Model.

It is not easy to write reliable cross-browser JavaScript, and yet this ability is crucial to the success of any sophisticated use of the rich capabilities provided by today's browsers.

The MathTran project intends to provide such code, for use as components in other systems, and will use such code when available. It is, for example, making heavy use of the jQuery JavaScript framework.

9 Editing math on web pages

The two most widespread ways of authoring content on a web page are to use an `textarea` form, or to use a WYSIWYG HTML editor such as TinyMCE, HtmlArea or FCKEditor. Very little if any content is authored in any other way.

As MathTran considers mathematics formulas to be text (in the special TEX notation), it is important to provide support for these two methods.

9.1 Mathematics in `textarea` elements

For `textarea` MathTran has developed an instant preview editor. The user enters a mix of text and mathematics, using the same `$` and `$$` as does TEX. As she types, so a preview of the current paragraph is displayed on the same web page as the form. This is the instant preview.

9.2 Mathematics in WYSIWYG HTML editors

We have also developed an experimental plugin for the WYM WYSIWYG HTML editor. It works by placing custom elements in the HTML being edited, and keeping these items up to date as the content changes. These elements contain both the formula in TEX notation and the image as rendered by MathTran.

9.3 Autocomplete for TEX commands

As part of his Google Summer of Code project, Christoph Hafemeister has developed an autocomplete device that can be used with either of these types of editors. In its present state, when the user types a backslash '\', autocomplete is offered. This is a list of all TEX mathematics commands known to the system, that begin with the current string. As the user types letters, so the list reduces. For example, if the user has typed `\ta` then the commands `\tan`, `\tanh` and `\tau` are offered.

10 Appropriate online help

Learning TEX notation is not exactly the same as learning mathematics, nor is it completely different. There is an overlap between the two realms. Part of engineering, for example, is to know that μ is commonly used to denote the coefficient of friction. Also, the ability to create a μ, both on paper and in electronic media is a useful skill for engineers. Similar remarks go for the other subject specialities.

When writing mathematics in the TEX notation, it helps to have domain-specific help. A traditional source for this information is the TEX source for another paper in the field, possibly written by the user himself.

The autocomplete device can offer the user help information on the commands, if such is availabe. Creating such help is an important task, that is best done with the active involvement of subject specialists. This might amount to 50 words on perhaps as many as 300 TEX commands.

Some subjects will, of course, use the same commands in similar ways. But for autocomplete it is important that the offered list of commands be appropriate to the users needs. For example, some who is learning trigonometry should be offered `\tan`, but to also offer `\tanh` would be unhelpful.

All systems for entering mathematics will benefit from the creation and use of domain-specific help. This author hopes that in the next year such help files will be developed for at least some areas, and will receive widespread approval. This would be a valuable shared resource.

11 Future plans

Although there are many interesting things that could be done by developing the MathTran TEX-server further, the main focus for the coming year is on making better use of the service that alread exists. This has four aspects:

- Improved documentation for the system — for example installation instructions for the MathTran server.
- Help files for the TEX-notation for mathematics.
- JavaScript programming for the web-browser, particularly for editing mathematical content.
- Integration of MathTran into existing systems, such as MediaWiki, MoinMoin, Drupal and WordPress.

This may lead to an online 'Learn TEX' course delivered on the web browser, and with components that can be readily incorporated on other web pages.