TEX Math Here: A Simple, Unified, and Efficient Model for Web-based Math Composition

David Schweitzer, Donald Honeycutt, Amy Lofgren, Daniel Serban
deschweitzer, drhoneycutt, alofgren, dtserban@liberty.edu
Department of Mathematics
Liberty University
24515
USA

Abstract

The TEX Math Here browser add-on for Chrome, Firefox, and (soon) Edge provides a framework for the straightforward composition of mathematical content on a myriad of Web-based creation platforms, including Google Drive's office suite, Microsoft Office Online, Prezi, and Blackboard. Without such a solution, many of these platforms would otherwise present significant obstacles to composing math. Examples include inefficient interfaces, steep learning curves, and even a complete lack of math support. By addressing the problem at the browser level rather than the Web-platform level, a unified solution becomes much easier to develop while also carrying several advantages for the user. Additionally, TEX Math Here has a keen focus on accessibility: the program is free for all; it supports keyboard shortcuts for users who may have mobility challenges; and any expressions created should be fully compatible with screen readers. In short, this project, at its core, searches for a final answer to the questions that math educators too frequently face in an online environment: "Can I write math here? And if so, how?"

1 Introduction

Educators, particularly those involved in some capacity with e-learning, face an ever-changing landscape with regard to electronic platforms. These changes often result in wonderful enhancements to our communication abilities; however, despite technological advances, challenges still exist, and may even directly result from them. For example, simply keeping up with the capabilities, functionality, and interfaces of all of the various platforms for content creation can be daunting to the point of near-impossibility. In the face of so many options (and their associated learning curves), logic dictates educators will, out of necessity, narrow their focus to a manageable subset of platforms that meets their needs.

Of course, the probability this subset can remain static over the long term is small. Sometimes, platforms simply change or cease development entirely. At other times, institutions, in an effort to

remain both current and competitive, may implement various technology initiatives, thus requiring (or prohibiting) the use of specific platforms. Regardless of its cause, change is hard. Regarding the subject at hand, change all too often negatively impacts the time an educator can devote to actively developing content, as he or she must necessarily first learn the new system whenever change occurs.

The challenge of change can be especially acute among educators in the mathematical sciences. As the field's general content remains rather static over time, math educators' methods often tend to adopt a similar rigidity, as alluded to by an old math faculty joke.[1] When considering the challenges in electronic symbolic typesetting that already confront math educators, also requiring frequent adaptation to a dynamic technological environment is burdensome to math faculty, possibly even prohibitively so.

The presentation of mathematical content has evolved significantly over the years. The TEX markup language developed in the 1970's [1] (and enhanced by LATEX [2] in the 1980's) provided a revolutionary and intuitive system for properly typesetting complicated mathematical expressions. MathJax [3] and KaTEX [4] have extended these efficient, yet robust, capabilities to website creators. Math-centric forums, such as Stack Exchange, offer users the ability to have TEX code rendered as mathematical expressions in their forum posts. Similar functionality can also exist on more general purpose platforms, such as Blackboard and Moodle. However, many key platforms still lack native support, and even if a platform does have math support, far too often some issue or challenge accompanies it, including improperly displayed notation, limited (if any) font choice, or a lack of color support. Of particular concern to this project are platforms for content creation, such as Google Drive's office suite, Microsoft Office Online, or Prezi.

The focus of this paper, TEX Math Here [5], is a browser add-on that attempts to seamlessly integrate the composition of mathematical content regardless of what platform one may be using. It is available in both the Chrome Web Store [6] and Firefox Add-Ons [7], and a late 2019 release to Microsoft Edge Extensions is projected. TEX Math Here also allows for font and sizing option selection that is as simple and intuitive as making those same selections for regular text in word processing software.

Accessibility is also a key focus of design, starting with cost for the user: free. Additionally, TEX Math Here utilizes image metadata for a variety of functions, the most significant of which is to facilitate screen reader compatibility for content readers with visual impairments. TEX Math Here also supports keyboard shortcuts for content creators who may have mobility challenges that affect mouse use, or simply prefer the more efficient workflow experience that keyboard shortcuts offer. Figures 1 and 2 briefly demonstrate TEX Math Here in practice.

2 Issues

When working in general-purpose, Web-based applications to create content, one can encounter any one of several potential roadblocks when composing mathematical expressions.

[1] Q: How many math teachers does it take to change a light bulb? A: CHANGE?!?!?

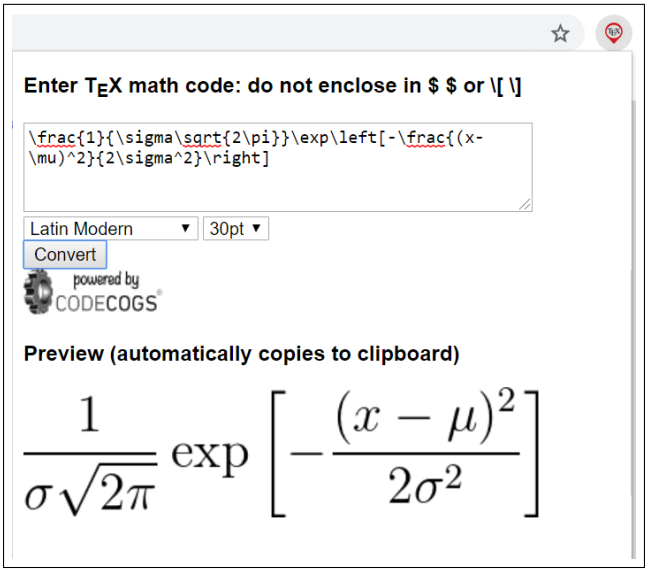

Figure 1: TEX Math Here launches from the menu bar (or via keyboard shortcut) and generates an expression that can be pasted in most Web platforms

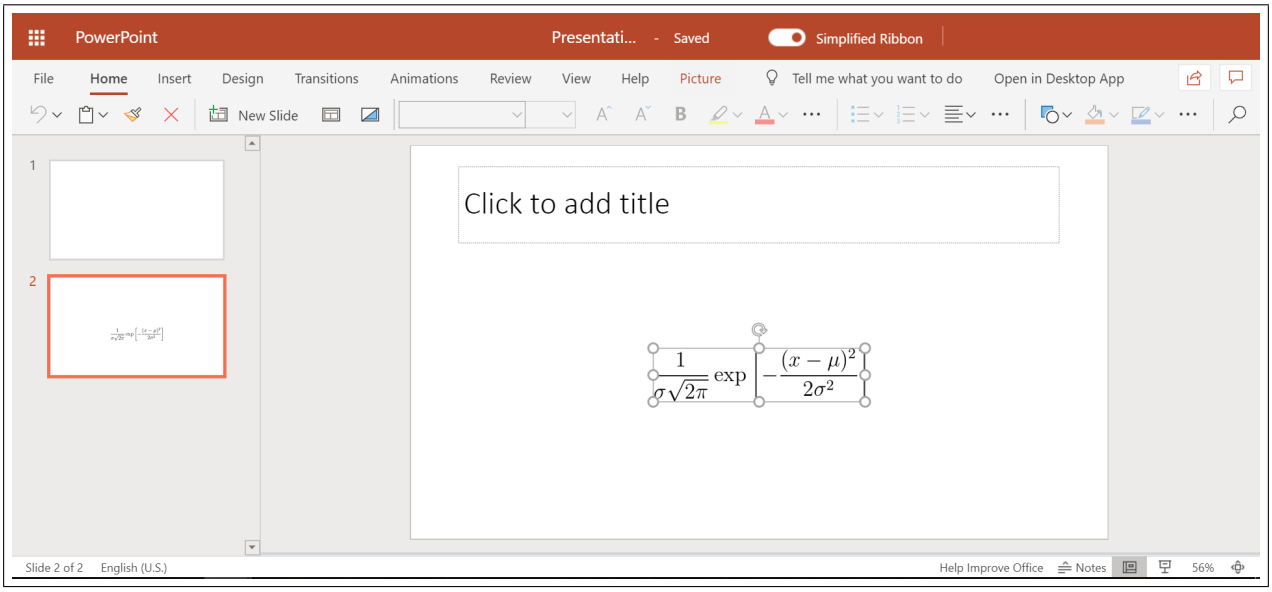

Figure 2: Microsoft PowerPoint Online with a TEX Math Here expression. TEX Math Here is the only current means for composing mathematical expressions in Microsoft Office Online, among other platforms.

2.1 Unavailability

Prior to this project, some platforms, namely Microsoft Office Online, Google Slides, and Prezi, simply had no straightforward solution for the creation of math content. At best, hacks requiring multiple other software tools could be employed, but if frequent math expression inclusion was required, this approach would quickly become completely impractical.

2.2 Third party reliance

Some platforms, for example, GMail and (until 2019) Google Docs, would depend entirely on third party solutions for math content. While support for math, even from a third party, is certainly better than no support, the system structure is intrinsically problematic. Namely, these solutions are typically geared toward just one platform, thus relying upon connecting to a platform's application programming interface (API) specification. These APIs can and do change, often breaking the add-on. These breaks then result in either further time and effort from the add-on developer or complete project abandonment. The potential for complete obsolescence is neither hypothetical nor exaggerated: Google's API changes in 2015 killed LaTeX Lab [8] and all other similar add-on projects for Google Docs.

2.3 Implementation

Even if an online environment has native support for mathematical content creation, the implementation of such can often still challenge the user.

2.3.1 Variety

While certain broad general conventions for user interface do exist and are typically followed, no single standard system for composing math prevails. Therefore, even if a broad convention is followed (such as a palette- or a TeX-based approach), the platform will undoubtedly still have its own unique idiosyncrasies and nuances. These unique characteristics force a learning curve upon the user, thus requiring time and effort just to learn *how* to make the content. When the number of different composition solutions is not significantly less than the number of different platforms supporting math, this does not facilitate educators' aforementioned gravitation toward a manageable subset of platforms.

2.3.2 Inefficiency

One of the more common interfaces for math composition is some form of palette that the user interacts with in a graphical user interface (GUI). While these palettes are often what first comes to mind when non-math educators think of electronic math composition capabilities, GUIs are actually often unpopular solutions among serious math writers. The research sheds clear light on the likely reasons for this unpopularity, as GUIs are inherently inefficient and error-prone [9, 10]. Consequently, they are not, for many, a realistic solution for composing extensive math.

Unfortunately, even attempts to implement support for TeX syntax can still prove quite inefficient. For example, the math composition system on Google Docs supports many TeX symbol and construct names, but in a manner similar to autocorrect. While this approach can have significant benefits in

small doses (as discussed in Section 6.4), this structure too often leads to an unusual hybrid between GUI and TEX, with the user heavily restricted in both the quantity and usage of TEX commands. Specifically, when the interface's behavior is not as one familiar with TEX would expect, this method becomes inefficient and error-prone for many of the same reasons that GUIs are (namely, transitioning systems interrupts workflow).

2.3.3 Poor presentation

Platform designers, likely in attempts to de-clutter the workspace for a general audience, often do not make certain math features or their usage clear. For example, Blackboard (via MathType from Wiris [11]) handles markup of TEX input very well: it recognizes a significant number of constructions, and it is editable. However, its usage is in no way obvious.

To see any mention of TEX/LATEX's availability, one must enter the math input environment by clicking the f_x button traditionally associated with math GUIs. This is a somewhat odd choice, as one wishing to use TEX is unlikely to associate such with the f_x button. Still, once loaded, the math environment does provide a link to the instructions for TEX usage. Unfortunately, what is not clear in these instructions is that one must actually be outside the math input environment in order to have TEX code properly render. This easily leads to confusion for the user.

2.4 General lack of awareness or interest

Whether stemming from reluctance to try new things or frustrations borne out of prior experiences with the many issues already mentioned, it seems many faculty are simply unaware or apathetic to what math composition tools may be available in a given platform, instead trying to find ways to manage without it. While this approach is detrimental to both faculty and students, it is commonplace.

2.5 Cost

While the MathType software by Wiris [11] may ultimately address some of these concerns (it is, as of this writing, in active development), it is not being developed for all platforms, and cost is still a major factor. Even with abilities currently limited mostly to Google Docs, the cost for an academic license starts at $39.95 USD per user, per year. For some, this is a significant cost, especially when considering the many limitations that exist in the software.

3 Design considerations

By continuing to leave responsibility for math composition capabilities in the hands of individual websites to develop, pass off to others, or neglect as they deem fit, the math community is not only at the mercy of the websites but also, by extension, inviting the possibility of innumerable different solutions to the problem. This is far from ideal or practical and leads to the many issues discussed above. As such, this paper presents an alternative solution for math content composition that can theoretically work in almost any Web platform on which one may wish to compose math.

3.1 Brief history, goals, and scope

This project was originally inspired by a desire to easily create good math slides on any platform. Of early concern among the research team was that solutions individualized for each platform would be necessary. However, a single unified solution, once feasibility was established, could not only provide far wider applicability (for example to other online office products and learning management systems) but also facilitate simpler implementation. Thus, developing a unified math composition solution, with target platforms of Google Drive's office suite, Microsoft Office Online, and Blackboard, became the project's primary focus. Of particular benefit is that, by accounting for the differences among just these three platforms, support for many other Web platforms is a natural byproduct. Prezi is such an example.

Accessibility has been a goal of this project from the start. In order to address one of the many facets of "accessibility," specifically as it pertains to educators, making the software completely free has been a longstanding desire of the creators.

While a unified approach may sound appealing, the question of "how" must obviously be addressed. Unsurprisingly, certain design conventions, compromises, and assumptions are necessary along the way.

3.2 Platform

Rather than trying to interact with the plethora of Web platform APIs in existence, addressing the math composition at the browser level consolidates considerably the challenge of accommodating many different websites. A browser-based solution also facilitates composing math, regardless of Web platform, in a uniform manner that should encourage higher usage rates. Under such a system, a user would need to learn just one composition system that could then be applied, theoretically, to almost any content creation platform. Additionally, because of standardization that exists in browser add-on development, one well-programmed solution can be easily extendable to other modern browsers.

Clearly, this approach has some advantages.

- The ability for a platform to handle math would solely depend on that platform's ability to handle the delivery method (see 3.3). If the delivery method has sufficiently widespread support, almost any platform would have math support.
- Having to learn only one system greatly reduces the general learning curve for users desiring to compose math content on the Web, perhaps even for the first time.
- Users would be able to operate at almost all times under the assumption that composing math was possible regardless of online context.

While minor differences do need to exist for different browsers, under this approach, the majority of the codebase can be common to all.

3.3 Delivery method

In order to have a uniform approach to math composition, a widely-supported means of delivery is essential. While, at first glance, rasterized images are unlikely to be considered a glamorous solution,

they actually carry specific benefits that amass to a feature set any other potential solution would be hard-pressed to match.

Most importantly, unlike many other math formatting methods, a wide variety of platforms support the pasting of rasterized images straight from the clipboard in a manner that is largely predictable. In other words, rarely does the pasting process result in any odd formatting changes or an unexpected appearance of the image (unlike, say, HTML, which can behave erratically when pasted). However, almost as important as the widespread, predictable support for images is the metadata that they can contain. This metadata can offer several benefits.

- Embedding a URL in the image's `src` field creates more widespread support by extending it to some platforms that otherwise do not allow for the direct pasting of images. One such example is Blackboard, which will not allow a user to paste an image directly from the clipboard, but will allow the pasting of a remotely hosted image via the information contained in an image's `src` field.
- Embedding the TEX code directly into the `title` field could theoretically allow the pulling of that information at a later time to facilitate the "editing" of existing images.
- Populating the image's `alt` field provides a description of the math expression for users with visual impairments who are using screen readers.

Of course, in terms of both quality and file size, vectorized graphics are far superior to rasterized graphics and support metadata as well. Unfortunately, the issue lies with lack of support for vectorized graphics in the Web platforms themselves. Even when employing various hacks, pasting vectorized images still results in a rasterized format in both Google Drive's office suite and Microsoft Office Online. Consequently, generating vectorized graphics would be a fruitless endeavor for the purposes of this project until more widespread support for them exists.

3.4 Implementation framework

Settling on a means to create the mathematical expressions is paramount. While palette-based (or handwriting recognition) solutions have their advantages, they carry two significant drawbacks with regard to this project (in addition to the efficiency and error propensity issues discussed earlier).

First, with LATEX already "the *lingua franca* of the scientific world" [2], much of the intended audience for this project would likely have a steeper learning curve with a new palette than with TEX-based math syntax. The second issue with a palette is on a practical level: GUIs are far more complicated for programmers to implement, and this complexity would drastically increase the development time necessary to create a solution with a reasonable level of robustness. In other words, developing a palette interface involving a myriad of symbols is far more complicated than just developing an interface whose main component is a text box. Thus, using TEX as a foundation is an easy choice.

3.5 TEX compilation

As the installation and configuration of TEX/LATEX can be an intimidating process, especially for a new user, this project (at least in the present) will focus on remote rendering of the TEX code. At the

time of this writing, a third party online renderer [12] is creating the images; however, development of a first party solution is nearing completion. Local hosting of the TEX renderer (should a user desire to configure such) also remains a possibility for future development.

4 Similar projects

Before proceeding further, it may be worthwhile to briefly address certain projects whose work may seem to overlap. Three in particular warrant comment.

4.1 Math Equations Anywhere

Theoretically, Math Equations Anywhere [13] aligns closely with this project's goals and methods. First released several months after this project had already begun development, its polished interface unfortunately does little for its notable functionality limitations. First, it does not seem to support Blackboard or Prezi at all (likely for the reasons addressed in Sections 3.3 and 5.1.1). Second, even in platforms in which it is supposed to function (for example, Google Docs), Math Equations Anywhere fails to correctly copy any fraction, at least during this team's months-long testing. Somewhat confusingly, the fraction issues are not even consistent across Chrome and Firefox. In short, it is difficult to view a math composition tool as a viable solution if it cannot correctly render something so fundamental in a commonly used platform such as Google Docs.

4.2 MathJax

One may wonder if many of this project's design challenges could have been avoided by simply implementing the existing functionality of MathJax [3]. MathJax is indeed an excellent solution that does much well; however, for this project's target Web platforms, MathJax simply was not a viable solution. The problem lies with the existence of only three possible output formats from MathJax: HTML-with-CSS, scalable vector graphics (SVG) files, and MathML. While any of these provide outstanding options for a content creator to implement mathematics on his or her own website, this project desires to have a user develop content on *others'* platforms. So, for example, if Google Slides does not support HTML-with-CSS, SVG files, or MathML (and in fact, it currently does not), MathJax, at least without significant enhancement to its present form or the Web platforms themselves, would seem ill-suited as a basis for this project.

In possible support of this hypothesis, Math Equations Anywhere's initial foundation was MathJax. For reasons not documented, Math Equations Anywhere abandoned MathJax in favor of the current KaTEX-based solution, which, as mentioned, is still not functioning properly. Of particular interest is that even when MathJax was rendering the expressions for Math Equations Anywhere, multiple file format conversions were necessary; these conversions not only increased the complexity of the project but also effectively removed all of the benefits that MathJax would otherwise bring.

4.3 Markdown Here

Markdown Here [14] is an excellent solution for Web-based email. In fact, while this paper's subject does also support email, if one's *sole* need for math composition is with Web-based email, Markdown Here is likely the best solution out there. Unfortunately, it has no functionality on the other applications in Google's or Microsoft's online office suites, and it can only output relatively small images (which may be fine for email, but not, for example, presentations).

5 The solution: TEX Math Here

A cross-browser add-on utilizing HTML, CSS, and JavaScript, TEX Math Here [5] provides a straightforward interface to convert TEX code to math expressions that can be used on any Web platform supporting image pasting. In addition, it affords the user simple and flexible aesthetic choices, a rarity in math composition. It is freely available for Google Chrome [6] and Mozilla Firefox [7], with Microsoft Edge support forthcoming.

Installation of the browser add-on inserts an additional icon in the browser menu bar. One launches the popup window for math composition either by clicking that icon or by typing the keyboard shortcut (which varies by operating system). In the resulting textbox, one enters the TEX code for the desired math expression (without delimiters) and can choose various options related to the font (which will be remembered on subsequent popup launches). When the Convert button is activated (either via click or Ctrl/Cmd+Enter), the TEX code is rendered as a preview, and the results are automatically copied to the clipboard for pasting (including into desktop applications if so desired). The resultant image is a .png file, as that is the widely supported rasterized file format offering the best balance between lossless image quality and file size.

When generating a math expression in the current version (version 0.6 for Chrome, 0.61 for Firefox), the user may choose various font sizes, as well as the output font. Based on limitations currently imposed by the third party renderer, this font selection is unfortunately rather limited: the sans serif fonts Computer Modern (Bright), Helvetica, Verdana, and Comic Sans, as well as just one serif font, Latin Modern (an extension of Computer Modern's traditional serif font). Despite the seemingly limited selection, this is still a more extensive list of distinct font choices (and easier interface) than just about any other math composition platform currently offers. Furthermore, the upcoming release of version 0.7 will replace the third party renderer with a first party renderer that will greatly enhance the number of supported fonts, including many that are more prominent and relevant for mathematical notation.

Image backgrounds are transparent, which should maintain any background properties that may already exist in the Web platform. This transparency (particularly when combined with future support of 24-bit color in version 0.7) is particularly useful if one is desiring compatibility with some slide presentation theme.

Once the image is previewed and automatically copied, the user can either press Esc or click back into the Web platform to automatically close the popup window and paste the math expression. Should one desire another math expression, relaunching the popup will maintain all previous font settings and TEX code. Maintaining font settings facilitates creating cohesive, stylistically consistent content. Maintaining the TEX code is useful for both correcting mistakes and providing a launch point for a similar expression, for example, in a derivation. However, if a completely new expression is desired,

the previous code is automatically highlighted on relaunch so that the user can easily overwrite it by simply starting to type. Figure 3 presents screenshots of the add-on's behavior throughout this process.

5.1 Metadata

In addition to the image itself, several useful pieces of metadata are also injected into the image prior to copying.

5.1.1 The `src` field

As discussed in Section 3.3, the generating URL is placed in the `src` field to allow support for platforms that do not support direct image pasting (likely due to image hosting concerns). Research also revealed that some platforms (for example, Google's products) even reference this field (when populated) instead of the image itself when initially pasting.

5.1.2 The `title` field

The TEX code is injected into the image's `title` field. This was primarily intended to facilitate support for the "editing" of existing TEX Math Here creations. (To be clear, due to interface issues, it is in fact not editing, per se, but rather a pulling of the TEX code to provide a starting point for a new creation, which could then be used to overwrite the old one.) Unfortunately, testing has revealed that developing a singular method for pulling the code that works across all platforms is likely impossible. As platform-specific coding (and maintenance) greatly increases scope and is rather contrary to the overarching methodology of this project, support for this feature is unfortunately going to be somewhat limited. Editing is currently supported only in Google Docs and Google Slides, with GMail support forthcoming. Attempts to support Microsoft Office Online in this fashion were so unsuccessful that this research team concluded it was impossible.

5.1.3 The `alt` field

For the image's `alt` field, useful information for screen readers should be present. Dr. Abraham Nemeth (creator of the Nemeth Braille system for mathematics) has developed the MathSpeak specification [15], which offers what is likely the most sophisticated solution to this problem. Unfortunately, implementation of a sophisticated solution also requires significant effort. While that effort is currently slated to begin in late 2019, an interim solution would still be most helpful. LATEX code, due to its linear and text-based nature, is viewed by the Mathematical Association of America as a perfectly viable solution for mathematical representation in screen readers [16]. Therefore, by inserting the TEX code into the `alt` field, TEX Math Here does currently support creating math content that should meet any accessibility standards, provided the platform on which it is being used properly supports such as well. Figure 4 shows the context menu of a TEX Math Here object already placed in Google Docs. Figure 5 shows the pre-populated accessibility information that that object contains, with improving the contents of that field a primary focus of future work.

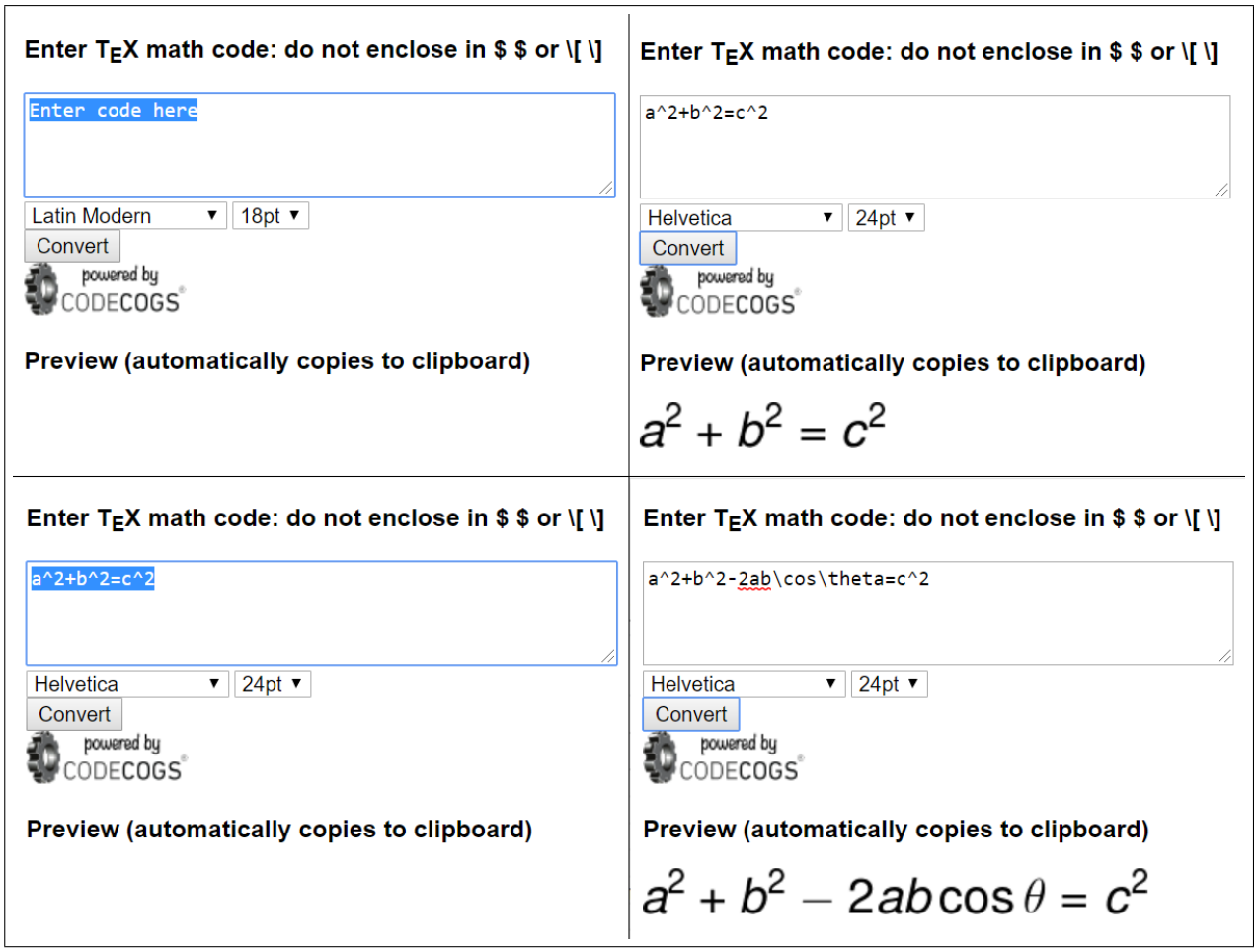

Figure 3: A series of interactions with TEX Math Here
Top left Initial instance of TEX Math Here, with its defaults
Top right An expression generated using non-default font settings
Bottom left Second instance of TEX Math Here, with preserved font settings and TEX code. Code is highlighted for easy erasure if user desires a completely new expression
Bottom right A second expression is generated simply by adding to the pre-existing code

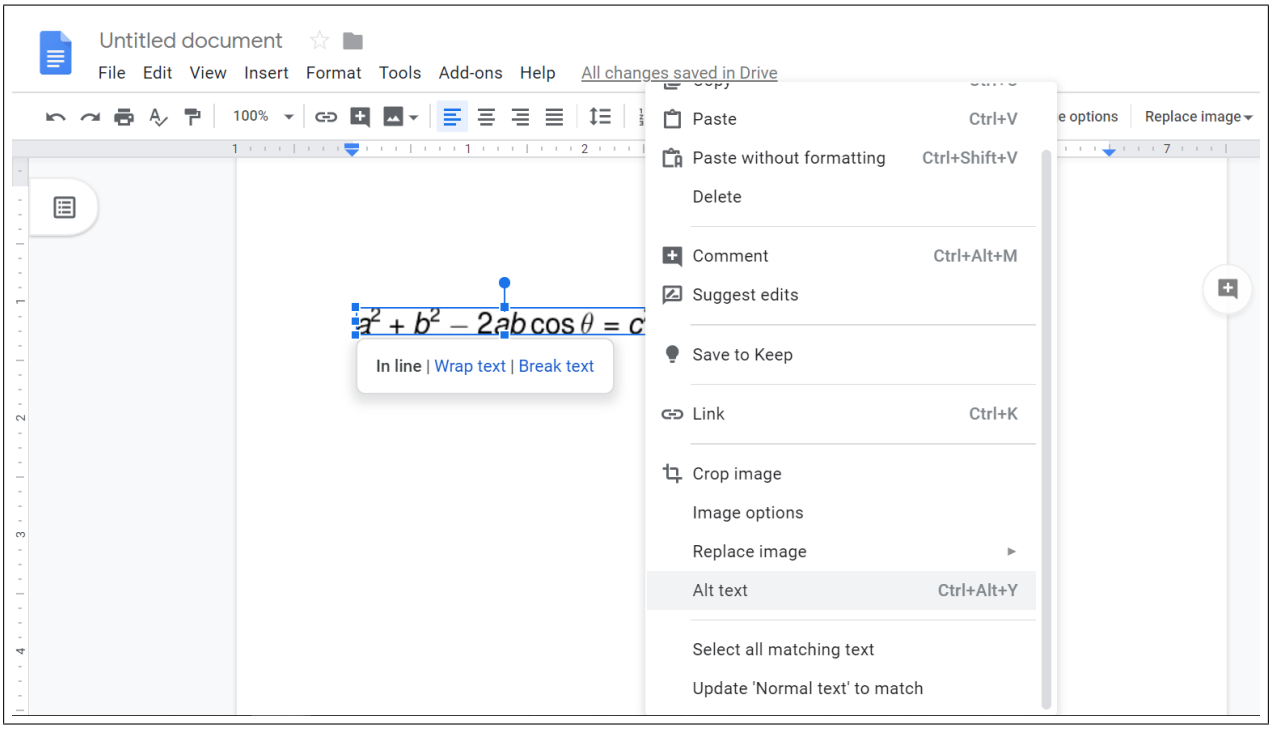

Figure 4: Alt Text information in Google Docs can be confirmed by right clicking a TEX Math Here expression and selecting Alt Text from the context menu

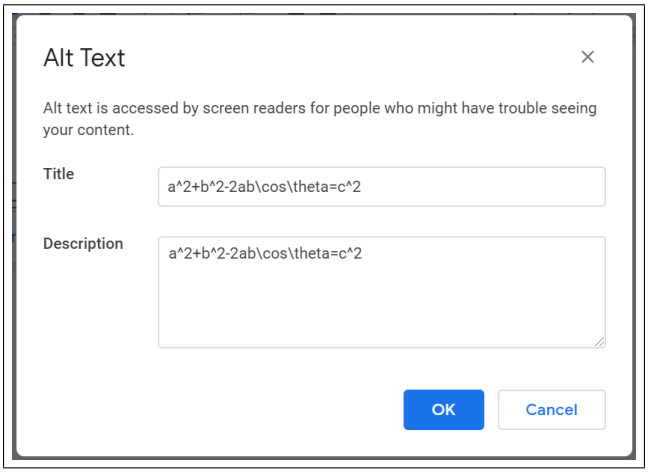

Figure 5: Google's Description field is what is traditionally viewed as the Alt Text. Populating this field with MathSpeak (as opposed to TEX code) is a future goal. Still, the MAA views LATEX as a viable approach to screen reader compatibility.

5.1.4 Unpredictable behavior

It should be noted that, on occasion, platforms will interact with the image, especially its metadata, in an unpredictable manner. Three main behaviors have been noticed thus far.

- The most common of the issues is that not all metadata fields are supported in all platforms. Typically, when this occurs, the Web platform simply does not maintain that data (for whatever reason), and it is lost.
- In some platforms, namely Google's (Docs, Slides, and GMail), the image from the clipboard is not pasted, but rather the URL in the `src` field is used to generate another (hopefully identical) image. In other words, an image without metadata will paste the image itself, but an image with metadata will pull the image using the contents of the `src` field. This can result in a pasting process that experiences a delay (while the image is retrieved) or results in an unexpected expression. For example, an early version of TEX Math Here displayed a bug (since fixed) where any spaces in the math expression (which TEX typically ignores) were converted to + signs in GMail due to unusual parsing of the `.src` field. Interestingly, this behavior was only visible *after* the email was sent and did not affect Docs or Slides.
- When using Ctrl/Cmd+V in Prezi, it pastes metadata from the title field instead of the image itself from TEX Math Here's autocopy. Thankfully, a workaround exists. In order for the math image to render correctly in Prezi, one must simply right click on the TEX Math Here preview image and manually copy it (as opposed to relying on auto-copy) prior to pasting.

6 Future work

6.1 Version 0.7

The next major release of TEX Math Here, version 0.7, should take place by the time of publication, as programming development is largely complete. As alluded to in Section 3.5, the primary objective of the new version is to switch reliance of this project to its own renderer, rather than that of a third party. Release of 0.7 will occur once server hosting is finalized. While one obvious benefit of this setup will be full control of the software and its dependencies, very practical features will also result.

While the switch in renderers will require at least temporary loss of two fonts (Comic Sans and Verdana), the overall number of font options will more than double, with far greater support for serif fonts in particular. Newly supported fonts (sometimes through the use of a clone) will include many standards of the publishing industry (mathematics or otherwise): Times New Roman, Palatino, Garamond, Bookman, New Century Schoolbook, Utopia, and Euler. A field below the drop down menu will also populate with any additional font information, such as an alternate name, specific version, or clone identification.

The new renderer will also bring support for full 24-bit color. Similar to other font settings, the color's RGB values will persist from previous instances of the popup. The popup will also include a dynamically changing box to indicate the selected color. A screenshot of the new interface, including color support and font descriptors, appears in Figure 6.

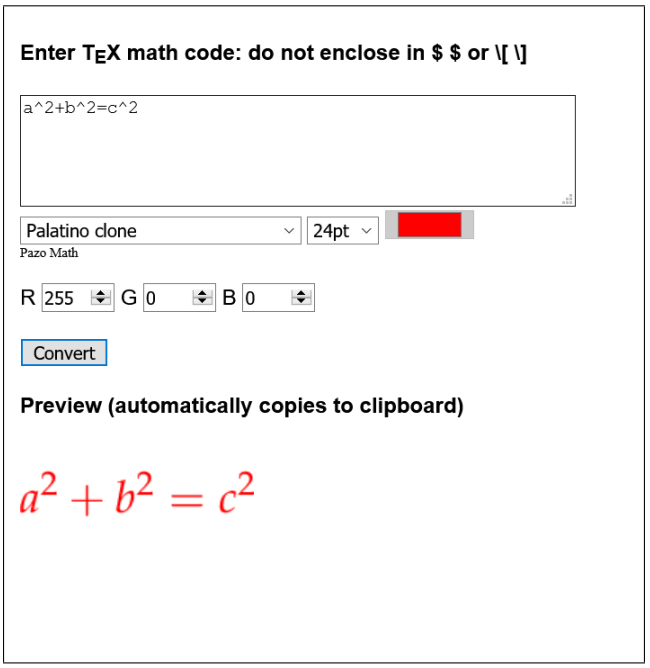

Figure 6: The interface for version 0.7, slated for release before February 2020

Unrelated to the renderer will be added support for pulling the code from TEX Math Here creations in GMail, as well as support for Microsoft Edge.

6.2 Accessibility improvements

After the release of version 0.7, focus will shift to attempting to implement Nemeth's MathSpeak protocol in an automated fashion. If successful, this information could then populate the image's `alt` field in order to improve screen reader support over the existing TEX-based solution.

6.3 Desktop version

Developing a desktop version of this software, complete with offline rendering performed by a local TEX installation is another potential direction for future development. This could be desirable as companion software to any application requiring math composition (such as a local Microsoft Office installation) and not just a browser. Additionally, by using a locally installed renderer, one would not need an Internet connection for functionality. Also, as font licensing for personal use is far easier than licensing for a Web server, the number of fonts supported would be virtually limitless (although behavior in math mode could be more unpredictable).

6.4 Inline math improvements

Obviously, if entering smaller mathematical expressions as part of a sentence, an image-based solution will generally cause problems, as subsequent edits to the surrounding text affect image placement without actually moving it. While very simple expressions (for example, $2+2=4$) can be adequately

represented using typical keyboard commands, many common mathematical symbols, such as ÷ and π do not have keys on the keyboard. While most common fonts actually contain many of these non-keyboard symbols, the challenge lies in simply accessing them. Historically, one could do so by either typing some four digit (or more) numeric code on the numpad (along with a special key, such as Alt) or using some form of GUI to select it. From a practical perspective, neither of these solutions are particularly useful even if needed only occasionally.

By incorporating functionality similar to autocorrect, TEX Math Here can currently replace the text used to generate many of the symbols available in TEX with the corresponding symbols in the user's font. For example, typing \alpha autocorrects to α. This solution, though hardly complicated in concept, greatly reduces interruptions to workflow and therefore increases efficiency. It works on all Web pages utilizing the standard document object model (DOM).

Unfortunately, development of this feature also revealed that all of this project's major target platforms use a proprietary interface that deviates from the DOM. As such, while this functionality can be seen to work on, say, google.com, it does not work on Google Docs, where it would be far more useful.

Two potential workarounds to this issue would seem to exist, though both carry drawbacks. In the first, if dealing with a single character (like π), TEX Math Here could theoretically generate the HTML or plain text equivalent instead of its typical image. The HTML or text could then be pasted directly into the Web platform. The drawback is the drastic increase in interfacing required just to type a simple character. The second possible workaround would be to develop, when possible, unique code that offers the same functionality as the solution for the DOM, for each of the different platforms. While this would be a far less obtrusive implementation than the popup method, it raises the same sorts of issues that exist with pulling code from TEX Math Here creations; namely, development would have to be individualized platform by platform, and platforms would need to remain relatively static over time. Despite these limitations, future development of at least one of these solutions is possible.

Another issue with inline math is the sizing and spacing of certain special characters, namely subscripts, superscripts, integrals, and summations. Testing has revealed that, though often not perfectly formatted, copying and pasting formatted HTML directly into the Web platform is possible, provided expressions involving these conventions are relatively uncomplicated. As such, it may be possible using either MathJax or, more likely, TEX4ht [17], both of which are certainly worthy of further exploration. In fact, if such a solution does prove feasible, an added benefit would be simultaneous implementation of the first autocorrect workaround discussed above.

7 Final Words

While much development remains, this project's cross-browser add-on allows one to perform mathematical composition relatively efficiently in Web-based environments, all while aiming to provide a user-friendly means for selecting from a relatively wide range of relevant fonts, sizes, and (in the near future) colors. TEX symbol codes can also be used to type mathematical symbols inline, albeit not on the many websites that stray from the standard Document Object Model.

With the ubiquity of Web-based learning management systems and e-learning in education, the issues this paper addresses are very relevant to the academic community. Particularly in online courses where an educator's only avenue of communication with students may be through these Web-based

platforms, it becomes clear that a method for easily and concisely inserting mathematical content is necessary. Not only would the use of TEX Math Here benefit the educator, but it could also facilitate communication and ease frustration on the part of the student as well, particularly in online courses. TEX Math Here can be used across practically all of these platforms to effectively convey mathematical content, facilitating both development of online resources as well as mathematical discussion.

Acknowledgements

The authors would like to thank the reviewers for their many thoughtful comments and suggestions, which led to improvements in not just this paper, but also the associated software. This research was made possible in part by support from the Center for Research and Scholarship, Liberty University.

References

[1] D. Knuth, "Mathematical Typography," *Bulletin (New Series) of the American Mathematical Society*, vol. 1, no. 2, March, pp. 337–372, 1979.

[2] L. Lamport, *LATEX: A Document Preparation System*, 2nd ed. Boston: Addison-Wesley, 1994.

[3] MathJax: Beautiful Math in All Browsers. [Online]. Available: http://www.mathjax.org

[4] KaTEX: The Fastest Math Typesetting Library for the Web. [Online]. Available: https://katex.org/

[5] TEX Math Here. [Online]. Available: http://www.mathaddons.com/ourtools.html

[6] TEX Math Here for Chrome. [Online]. Available: https://chrome.google.com/webstore/detail/tex-math-here/gopfokpflndblbooehdbffnnjmnegeph

[7] TEX Math Here for Firefox. [Online]. Available: https://addons.mozilla.org/en-US/firefox/addon/tex-math-here/

[8] LATEX Lab. [Online]. Available: https://code.google.com/archive/p/latex-lab/

[9] D. M. Lane, H. A. Napier, S. C. Peres, and A. Sandor "Hidden Costs of Graphical User Interfaces: Failure To Make the Transition from Menus and Icon Toolbars to Keyboard Shortcuts," *International Journal of Human-Computer Interaction*, vol. 18, no. 2, pp. 133–144, 2005.

[10] E. M. Altmann, J. G. Trafton, and D. Z. Hambrick, "Momentary Interruptions Can Derail the Train of Thought." *Journal of Experimental Psychology: General*, vol. 143, no. 1, p. 215, 2014.

[11] Wiris, *MathType*. [Online]. Available: http://www.wiris.com/en/mathtype

[12] CodeCogs. [Online]. Available: https://www.codecogs.com/latex/eqneditor.php

[13] Math Equations Anywhere. [Online]. Available: https://github.com/brendena/MathEquations-Extension

[14] Markdown Here. [Online]. Available: https://markdown-here.com/

[15] gh, LLC, *2004 MathSpeak Initiative*. [Online]. http://www.gh-mathspeak.com/

[16] A. Maneki, "Guidelines for Collegiate Faculty to Teach Mathematics to Blind or Visually Impaired Students," *2015 CUPM Curriculum Guide*. [Online]. Available: https://www.maa.org/sites/default/files/cupm/FacultyGuidelinesForTeachingVisuallyImpairedStudents.pdf

[17] TEX4ht. [Online]. Available: https://www.tug.org/tex4ht/