IGUANATEX: LATEX POWERPOINT ADD-IN GREATLY SIMPLIFIES CREATING ELEGANT MATHEMATICAL SLIDES

David Schweitzer
Liberty University
Department of Mathematics
1971 University Blvd
Lynchburg, VA 24515 USA
deschweitzer@liberty.edu

Abstract
Ideally, a system for creating mathematical slides should be easy to use, accessible in terms of both cost and required expertise, and robust in its capabilities. The resultant slides should also be consistent in appearance. While various popular platforms have attempted to balance these considerations over the years, no single solution has addressed all simultaneously. This paper will present a methodology that accomplishes all objectives.

Keywords: Mathematical slides, PowerPoint, LaTeX, IguanaTeX

Introduction
Because of the unique symbols and positioning typically required by mathematical typesetting, creating electronic slides containing mathematical content is naturally much more difficult than creating content that is only text-based. As such, this paper will establish a set of reasonable goals that an effective solution for math slide creation and presentation should meet.

When it comes to handling mathematical content, Microsoft PowerPoint (like all other off-the-shelf solutions) certainly has its share of weaknesses. Some of these stem simply from functionality limitations and others are the result of very questionable design decisions. However, with the help of just a few additional downloads, PowerPoint can provide an extremely useful toolset that will effectively accomplish all of our goals.

Goals
Our goals strive for both practicality and functionality.

Goal 1: easy to create. When most individuals first confront the need to create mathematical content in Microsoft Office, they may encounter one of two broad categories of approach: some sort of point, click, and type interface (like Equation Editor) or hand writing recognition. The latter, while promising, has a very long way to go before it is a truly useful tool for math with any degree of complexity. The former, while very intuitive to use, can also be rather tedious and inefficient when creating complicated and/or extensive mathematical structures. We also note that point and click interfaces are rather rigid in their approaches, spacing, and formatting, which we will explore further in Goals 3 and 4.

In contrast, markup languages provide an excellent alternative to a point and click interface. While mathematics is highly symbolic, conventions for how one "speaks math" already exist. Markup languages generally follow these same conventions, as they attempt to allow us to "type math" in a manner similar to how we "speak math." Symbols, structures, and positioning are all determined using various keyboard symbols and keywords. This allows for very efficient entry of a mathematical expression, with minimal clicking around. If properly implemented, this approach also maintains excellent copy/paste functionality for developing step by step derivations or proofs where relatively little is changing between steps.

Goal 2: accessible in both cost and required expertise. While we will ultimately settle on a solution based on Microsoft PowerPoint (a distinctly commercial product), almost all institutions (particularly in academia) provide the Microsoft Office suite for their employees at no individual cost. This makes PowerPoint extremely accessible to almost anyone who would have need of generating mathematical slides. The goal now is to maintain that same level of financial accessibility by using only products and downloads that are 100% free.

In terms of required expertise, this will be somewhat of a challenge simply because of the decision to use a markup language for the input of mathematical expressions. For a novice, there is going to be a learning curve with any markup language; however, the goal will be to keep that learning curve very gradual. For example, knowing the syntax for generating matrices is not required if one's course never uses matrices. This allows one to learn a markup language gradually, only on an "as-needed" basis, thus greatly simplifying the process.

Goal 3: robust capabilities. We wish our slides to be able to handle all manner of mathematical expression, with customization available, if necessary, as well. The TeX computer language was created specifically to typeset mathematics (and more specifically, in an aesthetically pleasing manner) [1]. LaTeX enhances this language by maintaining all of its mathematical capabilities while making those capabilities more user-friendly. As a result of this unique combination, LaTeX has been "the *lingua franca* of the scientific world" for over twenty years [2]. Therefore, a LaTeX-based solution provides an excellent framework for accomplishing this particular goal, provided the other goals remain achievable as well.

It is also important to note here that mere recognition of LaTeX code by a program is not necessarily sufficient. For example, PowerPoint itself (with some additional configuration) can recognize LaTeX code and generate mathematical expressions from it. However, it is essentially translating the LaTeX code to conform the output to PowerPoint's own system of math presentation, which, as we will see in Goal 4, carries with it some very undesirable characteristics. Additionally, translators generally do not allow one to access and edit the underlying LaTeX code once it has been translated. In other words, if an expression has been coded, then translated, re-editing the code may not be trivial or even possible without

starting over. As accessing the code itself greatly improves efficiency in both error correction and displaying incremental steps, we desire a solution based on native LaTeX.

Goal 4: consistent in appearance. Typically, one does not randomly change fonts when typing something text based. Therefore, it would seem logical that the same rules should apply to mathematical content as well. The appearance of any letter, Greek or Roman, italicized or regular, should remain consistent regardless of whether that character is entered inline or in a math expression.

Unfortunately, Microsoft decided that changing fonts repeatedly within a slide should be the default behavior for PowerPoint. While this questionable design choice is not particularly egregious (or even noticeable) in a text-only slide, when mathematical content is present, this problem is very conspicuous and can be jarring in appearance. Unfortunately, on its own, PowerPoint simply cannot display one uniform font if using math mode. Figure 1 helps demonstrate this weakness.

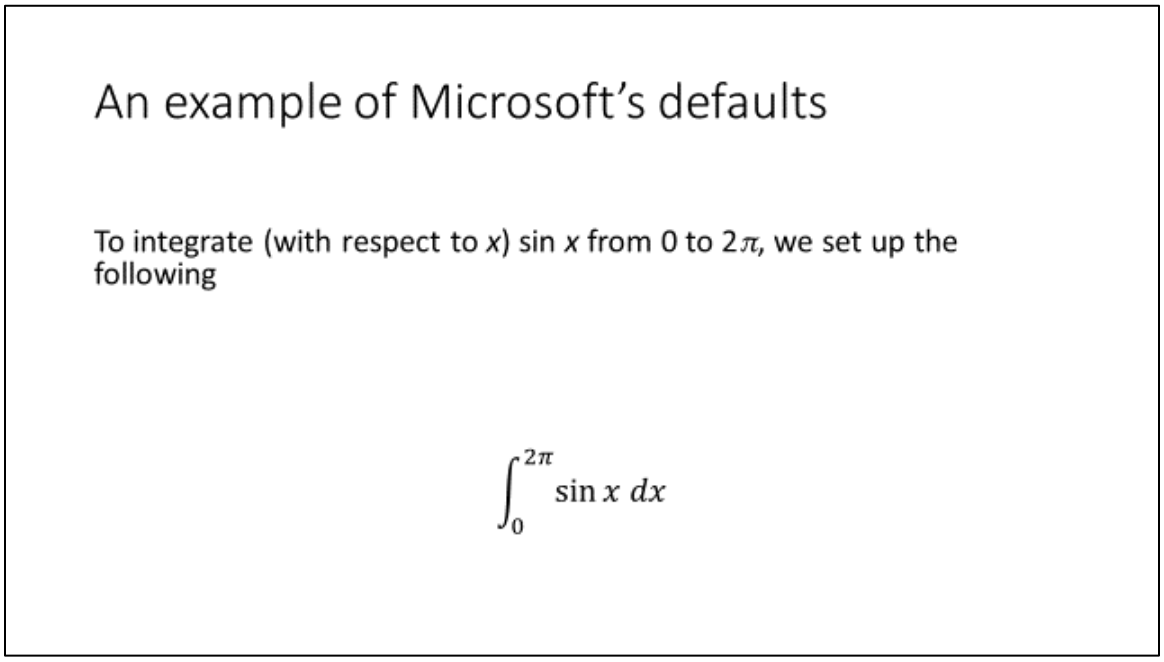

Figure 1: Sample slide using PowerPoint defaults and Microsoft's math tools

In this slide we are actually seeing at least four different fonts:

1. Calibri Light in the header
2. Calibri (including italicized Calibri) for the inline text ("To integrate…")
3. Cambria Math for the 0, 2, and sin in the math expression
4. Some unknown italicized font(s) for the *x*, *dx*, and π terms in the math expression

Obviously, this approach seems completely unnecessary and certainly not in line with common practice or common sense. What makes the implementation particularly puzzling is the rigidity of Microsoft's font decisions. The font options for math mode are severely restricted. Times New Roman, the standard used for years in the once-ubiquitous

Microsoft Equation Editor and also many textbooks, is surprisingly absent as an option. Additionally, the font used for the variables is not even an italicized version of Cambria Math, or any other common font for that matter. Therefore, matching that font inline is impossible. In other words, there is simply no easy way to get the "*x*" appearing inline to look like the "*x*" in the expression. In contrast, Figure 2 has inline text matching exactly the appearance of the corresponding terms in the math expression.

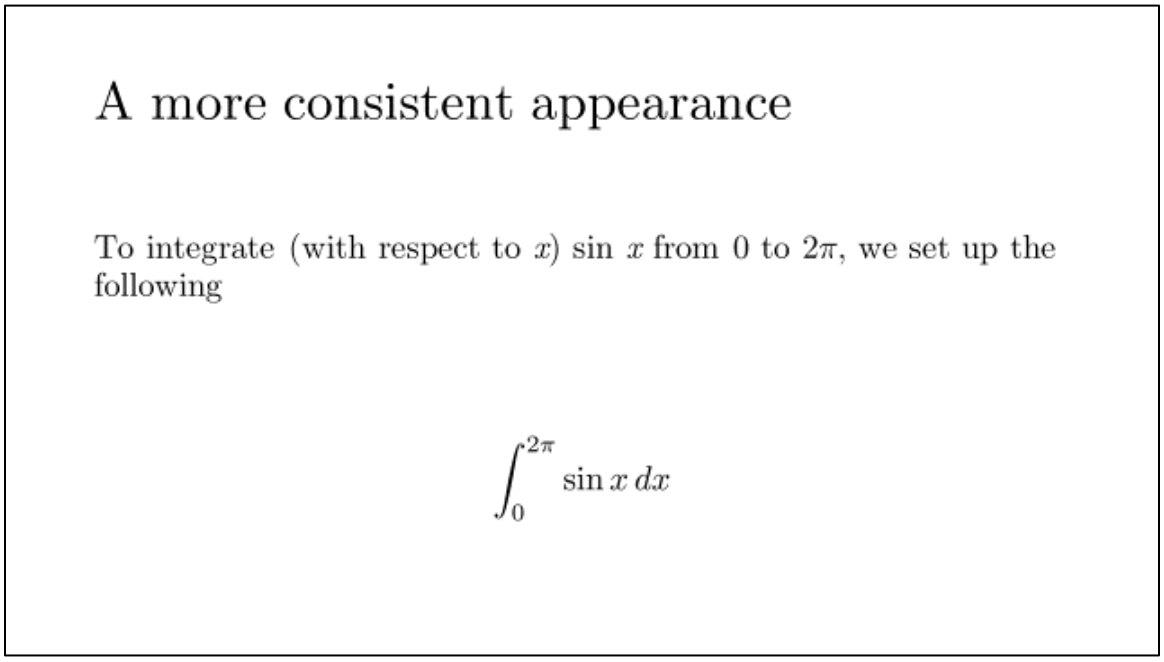

Figure 2: Consistency in appearance both inline and in the math expression

The full explanation for accomplishing all goals (as seen in Figure 2), is in the Solution section.

PowerPoint Alternatives (Namely Beamer)

In light of the weaknesses of PowerPoint's math capabilities, it would be easy to ask the question, why use PowerPoint at all? After all, alternatives certainly exist. There are several answers to this question.

First, PowerPoint is, without a doubt, *the* standard in slide creation software. While other solutions do exist, the market penetration of PowerPoint, whether in industry or in academia, is without peer. Therefore, to address issues such as compatibility with existing structures or collaboration with colleagues, PowerPoint is easily the best solution.

Second, there is an issue of user-friendliness, at which PowerPoint excels. In this author's experience, by far the most suggested alternative for users already familiar with LaTeX is the LaTeX-based slide creation tool, Beamer. Certainly, if one is comfortable and happy with Beamer, it is an effective solution, and this paper likely offers little to such a user. However, Beamer requires a far greater comfort level with LaTeX coding than this paper's

solution requires. This drastically increases the necessary learning curve, which, in turn, decreases accessibility (Goal 2). In fact, Beamer is so code-dependent that many experienced LaTeX users struggle mightily with it, to the point that they do not view it as a viable solution for creating slides. Therefore, treating Beamer as a solution for a wider audience that would include LaTeX novices seems inappropriate.

Third, PowerPoint is exceedingly adept at providing many features that are useful in the typical slide creation process, features that, even with heavy mathematical content, are still used extensively by almost any user. These features would include formatting, automatic resizing, animation, and integration of other media. Since we would still like access to all of these features, PowerPoint remains our best option.

The Solution

The solution proposed will contain three main components: the LaTeX PowerPoint add-in (and its dependencies), LaTeX's default font, and a font for easier access to LaTeX's Greek characters.

IguanaTeX (and its dependencies). This PowerPoint add-in, originally created by Zvika Ben-Haim and now maintained by Jonathan Leroux, offers a LaTeX-based environment anywhere in PowerPoint. A couple features of this product stand out. All instances of math content are saved both as underlying LaTeX code (available for subsequent editing) and as an image (for readability by users lacking IguanaTeX). Also, rather than translating LaTeX code into some other framework, it uses actual LaTeX renderers to ensure proper appearance. Further details, as well as the various downloadable versions of IguanaTeX can be found at the following link.

http://www.jonathanleroux.org/software/iguanatex/

LaTeX is unfortunately only available for Windows. However, LaTeXiT is an open-source alternative that is available for Mac installations. Due to lack of sufficient Mac access, this paper's coverage of LaTeXiT is unfortunately severely limited. However, all indications are that the behavior is largely similar, and the font information presented here should still be relevant. Mac users can find LaTeXiT at the following link.

https://www.chachatelier.fr/latexit/

Security warnings. Any time one deals with macros, security is a concern. Depending on one's installation, PowerPoint settings may require modification just to allow macros to run at all. IguanaTeX is no exception.

Unfortunately, that is not the entirety of the issue. Some malware detection tools flag IguanaTeX as malicious. This is not overly surprising. The developer acknowledges this on the website, discusses the issue at length, claims they are false positives, and makes all source code available for one's own inspection and determination. It is important that any

potential user perform due diligence to ensure they understand and are comfortable with the issue and potential risks.

Installation process. While almost all software follows a relatively straightforward installation process (download file, open file, click through prompts), installing a macro such as IguanaTeX is a bit different. If one simply opens the file, new menu items are visible and IguanaTeX can be used. However, as soon as that session is closed and a new one started, IguanaTeX will no longer appear. Therefore, it is imperative that, in order to get IguanaTeX's changes to persist, the user follow the installation instructions detailed in the Download section of the website.

Dependencies. Because IguanaTeX uses LaTeX to generate all equations, IguanaTeX requires the installation of a LaTeX engine for code conversion purposes. For Windows, MikTeX is the common LaTeX distribution and the one recommended by this author (the IguanaTeX website also mentions TeX Live as a possibility).

https://miktex.org/

When installing MikTeX, it is highly recommended that one select "Yes" when prompted with options for how to "Install missing packages on-the-fly." Selecting "Yes" is much more likely to avoid random crashes when using IguanaTeX (especially for the first time). Even if a crash occurs, there is a high probability waiting a bit and simply trying again will work. This solution often works because the most common type of crash results from needing an uninstalled package. If configured to download on-the-fly, MikTeX will fetch it automatically, but IguanaTeX often times out before that process has sufficient time to complete.

IguanaTeX lists the Ghostscript package as "Recommended" but not "Required." However, its absence can cause problems, as it assists with some of the PDF conversions that take place behind the scenes. Therefore, it is safest to include it in the installation process.
https://www.ghostscript.com/

Using IguanaTeX. Once installed, a new tab (with seven buttons) is added to the ribbon (See Figure 3).

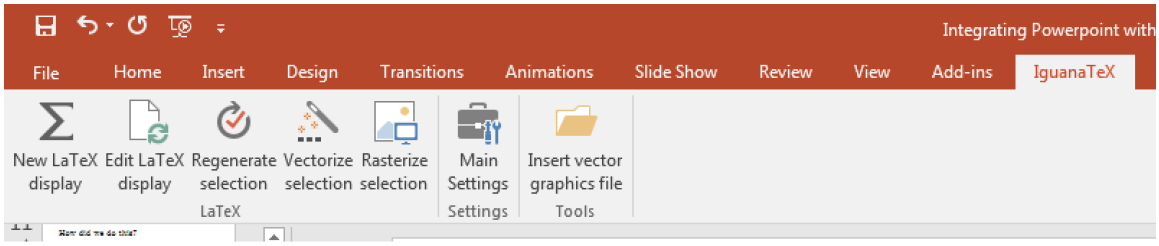

Figure 3: The IguanaTeX tab and its associated Quick Access buttons

The first two buttons are of primary concern. The first button adds a new LaTeX expression; the second is used to edit an existing expression. While one theoretically could add both text and math content using this button, for various reasons (such as sizing), displaying text really is best left to PowerPoint itself.

Clicking either button generates a dialog box (see Figure 4) into which we can enter LaTeX code. The only difference between the buttons is that the Edit button functions on a pre-existing expression, and the window will thus be pre-populated with the expression's current code. Also, note that while Figure 4 displays the default window and code block, one can easily reset the default using the button in the bottom right corner. If using LaTeX for mathematics only (as in this paper), this can be useful for setting the \[and \] LaTeX math delimiters to prepopulate the code block when called. Once desired code is present, the Generate button then creates the math expression.

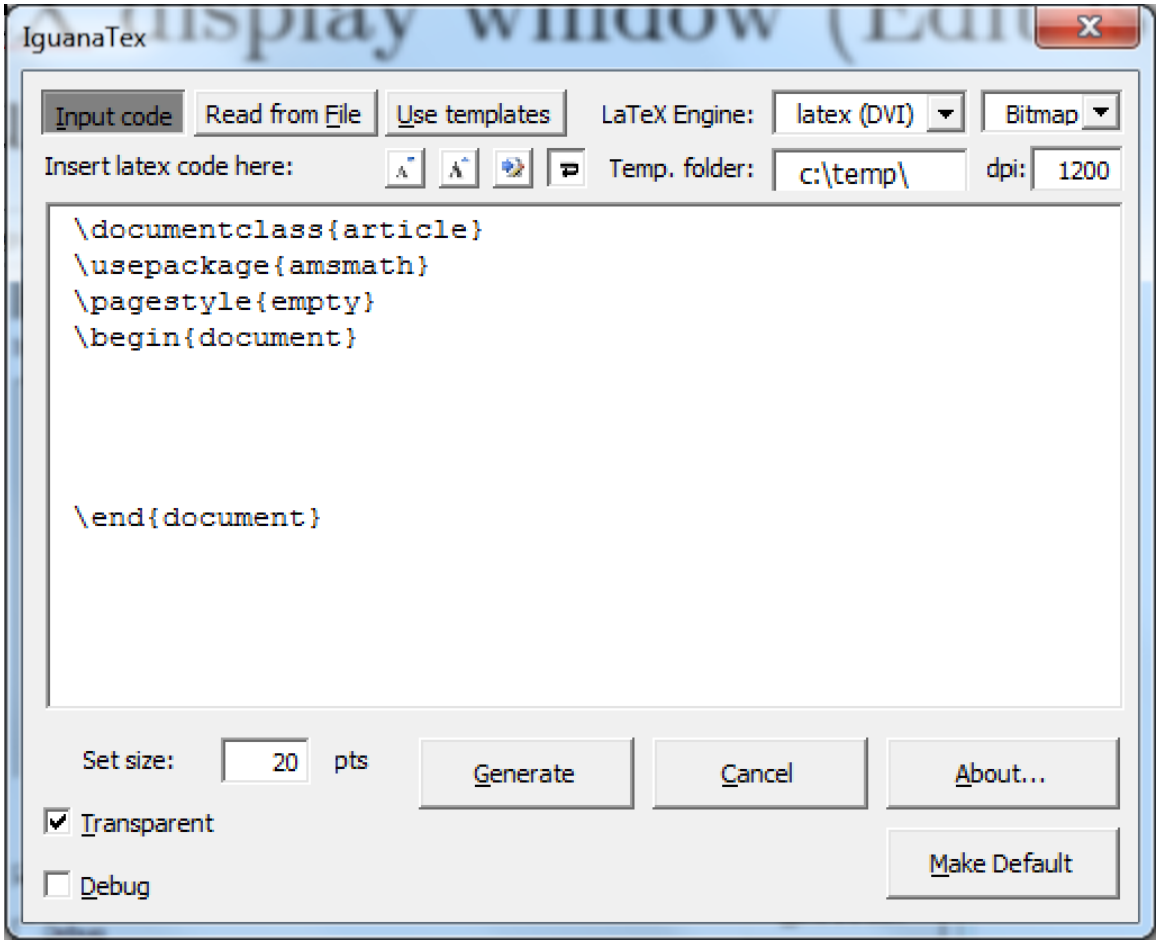

Figure 4: IguanaTeX input box

There are many possible actions now that we have access to all of LaTeX's capabilities. Essentially, any mathematical expression one can imagine is within reach. Of course, knowledge of LaTeX is required. For those completely unfamiliar with this markup language many great resources exist for learning. Unfortunately, many of these make the

system seem *far* more daunting than it actually is, especially for our purposes. As such, only basic reference resources are recommended for novices. Two examples would be

https://en.wikibooks.org/wiki/LaTeX/Mathematics

and

https://www.sharelatex.com/learn/Mathematical_expressions

Certainly as users become more comfortable with the system, they can increase the complexity of what they can generate.

Once a math expression is generated, it can then be treated as any other object within PowerPoint: it can be resized, moved, animated, etc. In particular, resizing from the rather small default size is helpful, and expanding size will not introduce any blockiness or blur, as the math expression is a vectorized graphic.

The Computer Modern font family. Now that we can generate mathematical expressions in PowerPoint using LaTeX code, we desire to match its output with the inline font used in PowerPoint. One possible solution is to download (and tell IguanaTeX to use) a Times New Roman equivalent in LaTeX. Unfortunately, with this approach, a problem still exists in matching the Greek letters. Therefore, instead of having IguanaTeX's output strive to match (one of) PowerPoint's conventions, we will instead have PowerPoint match LaTeX's. This is actually quite easy and also gives presentations a distinct look, using a font that was designed specifically for mathematical output [1].

Computer Modern Serif Roman is LaTeX's default font. It was created by Donald Knuth alongside the aforementioned TeX language upon which LaTeX is based. He subsequently donated it to the American Mathematical Society, and it is freely available through SourceForge as the Computer Modern Unicode family.

https://sourceforge.net/projects/cm-unicode/

To install this font family, one will first likely need to decompress the file (and also possibly the compressed file within the compressed file) using 7-Zip.

https://www.7-zip.org/

Once the individual fonts are extracted, they will need to be installed. One's particular version of Windows determines the exact process necessary to accomplish this. Specific font installation details are left to the reader.

After downloading and installing this font family, all members are easily accessible in PowerPoint using typical Font selection methods. The specific font we wish to use inline is CMU Serif, which, in its original form, will match the appearance of LaTeX's numbers

and functions such as sin and ln. In its italicized form, it matches all (Roman letter) variables.

The Euclid font family. Just as there are many lookalikes for Times New Roman, or really any standard font, Computer Modern is no exception. One such font family is the Euclid font by Design Science. Design Science makes this family of fonts freely available in order to facilitate the reading of Microsoft Office documents containing math content created using Design Science's MathType equation editor. MathType may sound familiar to some readers, as pop up ads for it were commonplace in Microsoft Office prior to Office 2007. This is because a stripped down version of MathType was rebadged and presented as Microsoft Equation Editor. For many, this point and click product served as the primary method of math input for some time.

The main Euclid font is very similar to Computer Modern and can be used in its place if desired. However, more useful to us is the Euclid Symbol font that is included. Interestingly, because Windows contains a Symbol font already, Euclid Symbol is only included for Mac and its compatability, as a Mac has no comparable font for Greek letters. However, in our Windows setting, Euclid Symbol is of great use. The entire font family (and other fonts) can be downloaded at the following link.

https://www.dessci.com/en/dl/fonts/

For most Microsoft Office users, when a Greek letter is needed, the Symbol font included with Windows is used. One significant advantage of this font is that each of the letters is easily accessible by simply typing our corresponding (Roman) letters using the Symbol font. For example, "a," "b," and "c" yield "α," "β," and "χ" respectively. Unsurprisingly, Microsoft's Symbol font unfortunately does not match the Greek font LaTeX uses; however, Euclid Symbol does. Just as important, Euclid Symbol is every bit as easy to use as Microsoft's Symbol: just type the corresponding Roman letter in the Euclid Symbol font. See Figure 5 for a comparison of appearances (note all fonts are italicized with the same point size).

LaTeX: $\alpha\beta\chi\delta\varepsilon\phi\gamma\eta\iota\varphi\kappa\lambda\mu\nu o\pi\theta\rho\sigma\tau\upsilon\varpi\omega\xi\psi\zeta$

Euclid Symbol: $\alpha\beta\chi\delta\varepsilon\phi\gamma\eta\iota\varphi\kappa\lambda\mu\nu o\pi\theta\rho\sigma\tau\upsilon\varpi\omega\xi\psi\zeta$

Symbol: $\alpha\beta\chi\delta\varepsilon\phi\gamma\eta\iota\varphi\kappa\lambda\mu\nu o\pi\theta\rho\sigma\tau\upsilon\varpi\omega\xi\psi\zeta$

Figure 5: LaTeX's Greek letters along with those from Euclid Symbol and (MS) Symbol

One could validly question why this paper presents Computer Modern at all if the Euclid family contains both the Greek letters we need *and* a Computer Modern clone. It is not necessary, but the main benefit to having both is that PowerPoint's autocomplete functionality in the Font dialog box works much more efficiently if regularly using fonts

with two very different names (CMU Serif and Euclid Symbol) rather than using fonts with two similar names (Euclid and Euclid Symbol).

A Key Enhancement

Once IguanaTeX, the Computer Modern family of fonts, and the Euclid family of fonts are installed, the goals of this paper are met. However, it does not take long for a PowerPoint user to realize that, if trying to use a font other than the default font, PowerPoint constantly gravitates back to that default. This can be frustrating, as font selections must be modified almost constantly. In other words, even if all of one's slides are currently in CMU Serif, the creation of a new slide will still start out in, say, Calibri.

Thankfully, this is fixable, though the solution is not obvious. By changing the configuration of the Slide Master, one provides PowerPoint with the default to which content will go. To accomplish this, select the "View" tab on the ribbon. Click on the "Slide Master" button, and access the "Fonts" drop down menu. Select the bottom option, "Customize Fonts…," and choose CMU Serif for both the Heading font and the Body font. Click the "Save" button, then click the "Close Master View" button. At this point, PowerPoint will use the CMU Serif font by default for that file. Unless set as a new default for PowerPoint as a whole (which is beyond the scope of this paper), new files will require a similar process. However, the CMU Serif custom configuration will be available directly from the "Fonts" drop down on subsequent PowerPoint sessions, rather than needing to click "Customize Fonts…" again.

Final Words

Having successfully accomplished the four goals of this paper, one can now experience a significant improvement in the visual consistency of any created slides. Equally important, this drastic improvement in visual quality dovetails nicely with a dramatic improvement in the efficiency of the slide creation process.

---

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

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