Writing Guidelines for

Homework with Proofs

Ethan Bloch

August 7, 2015

Contents

1 Introduction

The point of writing mathematics is two-fold: to make sure that the mathematics is correct, and to communicate the mathematics to others. Of course, it is the content that is the most important aspect of mathematical writing, but maintaining proper writing style is important too, because good writing helps make sure that the logical structure of the arguments are correct, and helps the reader understand the arguments. Poor writing can get in the way of the mathematical content.

In contrast to writing in other disciplines, where good style entails artful phrasing and felicitous wording, in mathematics the best writing is simple, clear and plain, to the point of being what in other disciplines would be considered plodding. Mathematical writing should strive for transparency—to draw as little attention to itself as possible in order to allow the often difficult content to be clear and understandable.

This manual includes some basic aspects of good mathematical writing, with a focus on homework assignments for proofs-based mathematics courses. Some of the topics are purely stylistic, whereas others involve the use of LaTeX, which has become an indispensable tool for writing mathematics.

2 Use Proper LaTeX

LaTeX is at times tricky to use, and can be a source of frustration. However, spending a bit of time learning to use LaTeX properly will, in the long run, lead to both better results and less frustration.

The most important thing to keep in mind when using LaTeX is to let LaTeX do the formatting; you should focus on the content. In particular, do not try to override LaTeX formatting commands (for example, for new lines and new paragraphs), and do make use of built-in LaTeX commands for both text and mathematics.

2.1 *Read the Manual for the Bard Macros Style*

The Bard macros style file, called bardmacros.sty, is designed for homework assignments in upper-level mathematics classes, such as Proofs and Fundamentals, Abstract Algebra, Real Analysis and Point Set Topology. It takes care of a number of formatting issues such as exercises, definitions and the like, and has specific macros for various upper level classes.

The manual for the Bard macros style file describes all the features of the style. Please read that manual, in addition to this writing guide, before you start using the Bard macros style file—doing so will save you effort in the long run, and will make your project look nicer.

The manual for the Bard macros style file can be downloaded from the same webpage that has the manual you are now reading, which is

`http://math.bard.edu/bloch/tex.html` .

2.2 *Make and Use Your Own Macros*

One of the most convenient features of LaTeX is that the user can define his or her own mathematical LaTeX commands, called macros, which can then be used similarly to any other LaTeX command. Using macros saves a lot of time when writing a lengthy text, and allows for very easy notational changes in the course of writing. Macros are best suited for mathematical notation, not for text (if you try using macros for text, be careful with spacing issues following such macros).

Defining a macro is via the `\newcommand` command. All such commands are inserted before

```
\begin{document}.
```

For example, suppose we want to write the expression $(v_1, \ldots, v_n)$ often. We could then define a macro for this expression, by writing

```
\newcommand{\vecn}{(v_1, \ldots, v_n)}.
```

We would then type `$\vecn$`, instead of `$(v_1, \ldots, v_n)$`, to obtain $(v_1, \ldots, v_n)$.

Observe that the symbols `$` `$` are not included in macros, so that the commands defined by the macros can be used inside larger formulas.

When you define a macro, you can choose any name you want, as long as it is not the name of an existing LaTeX command, and as long as it is entirely letters (no numbers or other characters).

It is possible to have variables in a macro. For example, suppose that in addition to writing $(v_1, \ldots, v_n)$ often, we also need to change the number n to other numbers in this expression. We could then define a macro with a variable, for example

```
\newcommand{\vecnv}[1]{(v_1, \ldots, v_{#1})}.
```

We would then type `$\vecnv p$` to obtain $(v_1, \ldots, v_p)$, and `$\vecnv {2n + 1}$` to obtain $(v_1, \ldots, v_{2n+1})$.

If more than one variable is needed in a macro, replace `[1]` in the definition with the appropriate number (up to nine variables are allowed).

There is no definitive rule about which mathematical notation should be defined via macros (as opposed to typing the symbols out each time they are used), but, in a lengthy text, the more often you use some notation, the more effort you will save by defining that notation via a macro, and the easier it will be change the notation if you ever need to.

2.3 *Common LaTeX Errors*

1. When doing display mathematics, which is done with either `$$ ... $$` or `\[ ... \]` (the latter being the better usage), do not leave blank a line before the display mathematics, and do not leave a blank line after it unless you are starting a new paragraph. Putting in unnecessary blank lines leads to unwanted spaces before and after the display mathematics.

2. Do not use `\\` other than inside mathematical commands such as matrices, align and cases. Let LaTeX do all the line breaking, paragraph indenting, etc., by itself. If you are not sure how to get LaTeX to do basic formatting such as starting a new paragraph, consult any introductory manual or website for LaTeX, or ask a more experienced LaTeX user.

3. To write a set such as $\{x \in \mathbb{R} \mid x^2 < \pi\}$, use

   ```
   $\{x \in \mathbb{R} \mid x^2 < \pi\}$.
   ```

   Observe that the curly brackets are written `\{` and `\}`, and that the vertical line is written `\mid`, NOT `|` (which would produce the wrong spacing).

4. To write an inner product such as $\langle x, y \rangle$, use

   ```
   $\langle x, y \rangle$.
   ```

   Observe that the brackets are written `\langle` and `\rangle`, NOT < and > (which would produce the wrong spacing).

5. To write a function such as $f\colon A \to B$, use

```
$f \colon A \to B$.
```

   Observe that the colon is written `\colon`, NOT `:` (which, yet again, would produce the wrong spacing).

   In the Bard macros style file, an abbreviation for defining functions is given by

```
$\func fAB$,
```

   which takes care of the colon and arrow automatically.

2.4 *Lengthy Comments*

To make a comment to yourself in your .tex document that is ignored when the document is compiled, use the % character. Any line in the .tex document that starts with % is ignored; if a % in inserted in the middle of a line, all the text to the right of the % symbol is ignored. Some implementations of LaTeX, for example TeXShop and TeXworks, put comments that start with % in a different color.

Unfortunately, the standard LaTeX program does not give a method for commenting out long sections of the .tex file, other than using lots of % symbols. Fortunately, there is a nice LaTeX package called verbatim.sty that solves this problem. To use this package, you must first insert

```
\usepackage{verbatim}
```

into your file prior to `\begin{document}`.

Then, simply type `\begin{comment}` before the text that you want commented out, and `\end{comment}` after the text. Do not place one `\begin{comment}` and `\end{comment}` inside another. Comments that are made in this way will not be color coded.

2.5 *Typesetting Simulated Computer Code*

The problem with writing simulated computer code in LaTeX is that LaTeX wants to override the spacing and line breaks that you put in the code. The simplest way to write a small amount of simulated computer code in LaTeX is to use the verbatim.sty package. To use this package, you must first insert

```
\usepackage{verbatim}
```

into your file prior to `\begin{document}`.

Then, simply type `\begin{verbatim}` before the simulated computer code, and type `\end{verbatim}` after it. *Everything* within the `verbatim` environment is typeset in typewriter font exactly as it appears in the .tex file—obeying spaces and line breaks exactly as in the .tex file, and not recognizing any special symbols. Be careful, however, if your code has very long lines, because it might run off the page.

For example, to obtain

```
This is     verbatim text, where $ signs are
just $ signs, and commands such as \huge are ignored.
```

simply write

```
\begin{verbatim}
This is     verbatim text, where $ signs are
just $ signs, and commands such as \huge are ingored.
\end{verbatim}
```

There is another package, called moreverb.sty, that contains some very useful variants of the `verbatim` environment; these environments can be used to make typesetting simulated computer code easier. To use this package, you must first insert

```
\usepackage{moreverb}
```

into your file prior to `\begin{document}`.

The `verbatimtab` environment from the moreverb.sty package allows tab characters to be expanded to a given number of spaces (without this package, LaTeX considers tab characters to be single spaces). In the `verbatimtab` environment, tab characters are by default set to equal eight space characters. In the following examples, the ▷ character denotes a tab character.

To obtain

```
This        is        verbatimtab        environment.
```

write

```
\begin{verbatimtab}
This▷is▷verbatimtab▷environment.
\end{verbatimtab}
```

An optional argument after the defintion of `verbatimtab` environment can be used to set tab characters to any desired number of spaces.

To obtain

```
This    is    verbatimtab    environment.
```

write

```
\begin{verbatimtab}[4]
This▷is▷verbatimtab▷environment.
\end{verbatimtab}
```

The `verbatimcmd` environment from the moreverb.sty package is generally like the `verbatim` environment, except that in the `verbatimcmd` environment any LaTeX command that starts with a `\` is obeyed, and and `{ }` is respected. Hence, styles such as bold and italics can be used in the `verbatimcmd` environment. Mathematics symbols can also be used in the `verbatimcmd` environment, though only those that start with the `\` symbol; therefore `$a^{x+y}$` would not be recognized (because neither `$` nor `^` starts with the `\` symbol), but it could be replaced with the less commonly used `\(a\sp{x+y}\)`, where `\(  \)` are another way of starting and ending in-line math mode, and `\sp` means superscript (the commands `\[  \]` are another way of starting and ending display math mode, and `\sb` means subscript).

To obtain

```
This        is a^{x+y}        in verbatimcmd.
```

write

```
\begin{verbatimcmd}
This        is \(a\sp{x+y}\)        in {\bf verbatimcmd}.
\end{verbatimcmd}
```

The `listing` environment from the moreverb.sty package is like the `verbatim` environment, except that it numbers lines. The `listing` environment has two parameters, the first (in square brackets) specifies the step between numbered lines (a value of 1 makes every line numbered), and the second parameter (in curly brackets) specifies the number of the first line. The `listing` environment handles tab characters similarly to the `verbatimtab` environment, though tab characters are unalterably set to equal eight space characters.

To obtain

```
     This is the environment that
 8   numbers lines.  In this example,
     every other line is numbered,
10   and the first line is set at line 7.
```

write

```
\begin{listing}[2]{7}
This is the environment that
numbers lines.  In this example,
every other line is numbered,
and the first line is set at line 7.
\end{listing}
```

The `listingcont` environment from the moreverb.sty package is like the `listing` environment, except that it does not have any parameters, and it continues numbering lines where the previous `listing` or `listingcont` environment left off. The `listing*`

and `listingcont*` environments from the moreverb.sty package are very much like the `listing` and `listingcont` environments, respectively, except for two differences: they write ␣ for each blank space in the original text, and they do not handle tab characters.

3 Figures

There are a number of issues concerning the use of figures in LaTeX: what format the figures need to be in, how to create figures, where to put the figure files on your computer, how to insert figures into a LaTeX document and how to refer to figures elsewhere in the text.

3.1 *Format for Figures for Use in LaTeX*

The format of the figures you use in LaTeX depends upon which implementation you use. For TeXShop (recommended for Mac), TeXworks (recommended for Windows and Linux), and any other implementation that uses PDF files instead of DVI files, the correct format for graphics is PDF. (It is also possible to use PNG and JPEG files in TeXShop, and perhaps in the other programs referred to above.) For implementations of LaTeX that do not use PDF files, the standard format to use is EPS (encapsulated postscript, ending in .eps).

The method of inserting these different types of files is the same in all implementations; the format of the graphics file simply has to work with your implementation of LaTeX.

3.2 *Creating Mathematical Figures*

In principle any figures file in the correct format can be inserted into LaTeX, no matter how it was created.

There are three common ways to create mathematical graphics: via mathematical software such as Mathematica, Maple or Sage; via vector drawing programs such as Adobe Illustrator, Inkscape or GeoGebra; or via LaTeX packages such as TikZ. Do not use bitmap graphics programs such as Adobe Photoshop or Gimp (their images do not scale properly, and can appear pixelated)—vector drawing programs are much better suited to mathematical drawing.

Whatever method you use, work in the proprietary format of the program while you are making the graphic, and only when you are done, export the graphic to PDF or EPS format as needed.

One tricky matter when creating graphics is to make sure that when a graphic is exported to PDF or EPS format there is no extra white space surrounding the graphic (having extra white space creates problems with spacing). EPS files have a "bounding box" that determines how much white space is inserted; the bounding box is usually created automatically with as little white space as possible. For PDF files, which are often created with the graphic located on an entire blank page, the white space can be cropped using an appropriate program (such as Preview on the Mac).

3.3 *Where to Put the Figure Files on Your Computer*

Once you have created your figure files, put them in the same folder on your computer as your .tex file. If the figure files are anywhere else, LaTeX will not be able to find them,

unless, when you insert the figures into your .tex file, you give the full path+filename for the figures rather than just the filename of the figures.

3.4 *Inserting Figures in LaTeX*

The standard way to insert graphics into LaTeX documents is to use either the graphics.sty package or the graphicx.sty package. The two packages are essentially the same, but use slightly different formats for a few commands; use whichever package you prefer (though use only one of them). The manual for these two packages, called grfguide.pdf, can be found on the web; it is a very useful guide to inserting graphics into LaTeX documents, and to using color.

To use either of these packages, you must first insert

```
\usepackage{graphics}
```

or

```
\usepackage{graphicx}
```

into your file prior to `\begin{document}`. The command `\usepackage{graphics}` has already been inserted in the Bard macros template.

To insert a PDF file (or an EPS file if needed), use

```
\begin{figure}[ht]
\centering
\includegraphics{[filename]}
\caption{[caption]}
\label{[figure Label]}
\end{figure}.
```

Make sure to put in your own filename, caption, and figure label, and remove the `[ ]` symbols (which are meant to imply that your own text needs to be inserted); the symbols `[ht]` should not be modified (including the `[ ]` symbols).

The caption can be left blank if you do not wish to insert one (though you still need to write `\caption{}` if you leave the caption blank). You should not write the figure number in the caption, since LaTeX automatically calculates the figure numbers.

3.5 *Referring to Figures*

To refer to a figure that you have inserted, use

```
Figure~\ref{[figure label]}.
```

Make sure to put in your own figure label, and remove the `[ ]` symbols (which are meant to imply that your own text needs to be inserted). Note the use of `~`, which is a

non-breaking space, and prevents the figure number from being placed by itself at the start of a new line.

Here are some important points to note when you refer to figures in a mathematics text.

1. *Every figure* must be referred to by its figure number somewhere in the text. If a figure is not referred to, it will not necessarily be read.

2. Every figure should be inserted *after* it is first referred to.

3. Captions should be left blank, or should be minimal. The main explanation of a figure should be done in the text, not in the caption.

4 Good Mathematical Writing

Some aspects of good mathematical writing, for example the first two rules stated below, apply to all forms of writing, not only mathematical; other aspects are specific to mathematics.

4.1 *The Written Text Should Stand on Its Own*

A good written text should be understandable to the reader without any need for clarification by the writer. Err on the side of too much explanation; do not assume the reader is a mind reader.

4.2 *Revise, Revise, Revise*

The single most important thing to do to write well is to revise your writing repeatedly.

Be merciless with your own writing—do not be afraid to add, delete and move text. Do not become too attached to anything you have written; if it does not work, it needs revision.

4.3 *Write Precisely and Carefully*

There is no room in mathematics for ambiguity. The most minute matters of phraseology in mathematics may make a difference. Something as seemingly insignificant as the change of the location of a comma can change the meaning of a statement. Make sure that what is written is what you meant.

Use your imagination when exploring the mathematical content of your project, but write your ideas in simple, straightforward, unimaginative prose. Serious mathematics is hard enough as it is, without having unnecessary verbiage or convoluted sentences making it even less clear.

Mathematics is often read by skipping back and forth, and so a particular need for precision is in the statements of theorems, lemmas, propositions and the like, which must contain all their hypotheses, rather than having the hypotheses in some earlier paragraphs. Better a bit of redundancy than a confused reader.

4.4 *Use Full Sentences and Correct Grammar*

The use of correct grammar, such as complete sentences and correct punctuation, is crucial if the reader is to follow what is written. Mathematical writing should be no less grammatically correct than literary prose.

It is very important to recognize that mathematical symbols are nothing but shorthand for expressions that could just as well have been written out in words. For example, the phrase "$x = z^2$" could be written as "the variable x equals the square of the variable

z." All mathematical symbols, even those displayed between lines, must be embedded in sentences and paragraphs, and are subject to the rules of grammar just as words are.

Proper grammar helps the reader follow the logical flow of an argument. Connective words such as "therefore," "hence" and "it follows that" help guide the logical flow, and should be used liberally.

4.5 *Define All Symbols and Terms You Make Up*

All mathematical symbols used as "variables," even simple ones such as x or n, need to be defined before they are used. Such a definition might be as simple as "let x be a real number" or "let $x \in \mathbb{R}$."

Just because a letter such as n is often used to denote an integer, or the letter f is often used to denote a function, one cannot rely upon such conventions, because these same letters can be used to mean other things as well. If you want to use n to denote an integer, you must say so explicitly, and similarly for f denoting a function.

For the sake of readability, avoid as much as possible the temptation to make up new words and symbols and to use exotic alphabets and other complications (such as subscripts of subscripts). Try to stick to standard notation. If you do make up some notation, make sure you define it explicitly.

4.6 *Break Up Long Proofs into Steps*

If a proof is long and difficult to follow, it is often wise to break it up into steps, or to isolate those parts of the proof that are self-contained technical steps as lemmas (which are simply smaller theorems used to prove bigger theorems).

If you use lemmas, be sure to state them precisely, with all the needed hypotheses. All lemmas and their proofs should be placed before they are used in the main theorem. Do not put a lemma inside the proof of the main theorem—doing so can be very confusing to the reader.

When a long proof cannot be broken up into pieces, it is helpful to the reader if, prior to going into the details of the proof, a sentence or two outlining the strategy of the proof are provided.

4.7 *Miscellaneous Mathematical Writing Tips*

1. Do not put a mathematical symbol directly following punctuation. As a corollary, do not start a sentence with a symbol. The only exception to this rule is when the punctuation is part of the mathematical notation, for example "(x, y)." It is important to avoid ambiguities that might arise from using punctuation without proper care. For example, does the expression "$0 < x, y < 1$" mean that both x and y are between 0 and 1, or does it mean that $0 < x$ and $y < 1$?

2. Whereas it is fine to use logical symbols, such as $\wedge$, $\vee$, $\exists$, $\forall$, $\therefore$ and $\Rightarrow$, as abbreviations for words in your scratch work, such symbols should not be used in the actual written text of the project. Unless the subject of the project is logic, where logical symbols are necessary as parts of formulas, the use of logical symbols makes the exposition harder for others to follow.

3. Use consistent notation throughout your writing. For example, if you initially use uppercase letters for matrices and lowercase letters for numbers, stick with that notation for the duration of the project. Do not use the same notation to mean two different things, except when it is unavoidable due to standard mathematical usage—for example, there are multiple standard uses of the notation "(a, b)."

4. Display long formulas, as well as short ones that are important, on their own lines. Recall, however, that such displayed formulas are still parts of sentences, and require normal punctuation. In particular, if a sentence ends with a displayed formula, there needs to be the period at the end of the formula.

5. Colons are very rarely needed in mathematical writing. They are usually either unnecessary, or meant as substitutes for words in situations where words would be much more clear. The basic rule is not to use a colon in mathematical writing in a place where you would not use one in non-mathematical writing. Restrict the use of colons in mathematical writing to headings or at the starts of lists, and in certain mathematical symbols (such as the definition of functions).

6. Capitalize names such as "Theorem 2.3" and "Lemma 17." No capitalization is used in phrases such as "by the previous theorem."