

***Workflow***

Higham, Nicholas J.

2015

MIMS EPrint: **2015.111**

Manchester Institute for Mathematical Sciences  
School of Mathematics

The University of Manchester

Reports available from: <http://eprints.maths.manchester.ac.uk/>

And by contacting: The MIMS Secretary  
School of Mathematics  
The University of Manchester  
Manchester, M13 9PL, UK

ISSN 1749-9097

---

## Workflow<sup>†</sup>

Nicholas J. Higham

---

Workflow refers to everything involved in producing a mathematical paper other than the actual research. It is about the practicalities of how to do things, including how best to use different kinds of software for different tasks. The characteristics of a good workflow are that it allows the end result to be achieved efficiently, repeatably, and in a way that allows easy recovery from mistakes.

### 1 Typesetting: T<sub>E</sub>X and L<sub>A</sub>T<sub>E</sub>X

In the days before personal computers, articles would be handwritten, then typed on a typewriter by a secretary, and ultimately typeset by a publisher. Nowadays, almost every author prepares the article herself or himself on a computer and the publisher works from the author's files. In many areas of academia it is the custom to use Microsoft Word, or an open source equivalent. In mathematics, computer science, and physics L<sub>A</sub>T<sub>E</sub>X has become the de facto standard.

T<sub>E</sub>X is a typesetting system invented by Donald Knuth in the late 1970s that has a particular strength in handling mathematics. L<sub>A</sub>T<sub>E</sub>X is a macro package, written originally by Leslie Lamport, that sits on top of T<sub>E</sub>X. A T<sub>E</sub>X or L<sub>A</sub>T<sub>E</sub>X file is an ASCII (plain text) file that contains commands that specify how the output is to be formatted, and it must be compiled to produce the final output (nowadays usually a PDF (Portable Document Format) file). This contrasts with a wysiwyg (“what you see is what you get”) word processor, such as Microsoft Word, that displays on the screen a representation of what the output will look like. T<sub>E</sub>X allows finer control than word processors (the latter are sometimes described as “what you see is all you get”) and the ability of L<sub>A</sub>T<sub>E</sub>X to use style files that set various typesetting parameters makes it very easy to adjust the format of an article to match a particular journal. L<sub>A</sub>T<sub>E</sub>X is also well suited to large projects such as books. Indeed this Companion is typeset in L<sub>A</sub>T<sub>E</sub>X and the editors and production editor find it hard to imagine having produced the volume in any other way.

---

<sup>†</sup>. Author's final version, before copy editing and cross-referencing, of: N. J. Higham. Workflow. In N. J. Higham, M. R. Dennis, P. Glendinning, P. A. Martin, F. Santosa, and J. Tanner, editors, *The Princeton Companion to Applied Mathematics*, pages 912–916. Princeton University Press, Princeton, NJ, USA, 2015.

Polynomials are one of the simplest and most familiar classes of functions and they find wide use in applied mathematics.

```
A degree $n$ \py\
$$
p_n(x) = a_0 + a_1 x + \cdots + a_n x^n
$$
is defined by its $n+1$ coefficients
$a_0, \dots, a_n$ \in \C$ (with $a_n \ne 0$).
```

**Figure 1** L<sub>A</sub>T<sub>E</sub>X source. \py and \C are user-defined macros. \cdots and \dots are built-in T<sub>E</sub>X macros. Dollar signs delimit mathematics mode.

Figure 1 shows some L<sub>A</sub>T<sub>E</sub>X source code. Although how the code is formatted makes no difference to the output, it is good practice to make the source as readable as possible, with liberal use of spaces. I like to start new sentences on new lines, which makes it easier to cut and paste them during editing.

T<sub>E</sub>X and L<sub>A</sub>T<sub>E</sub>X are open source software and are available in various distributions. In particular, the T<sub>E</sub>X Live distribution is available for Windows, Linux, and Mac systems (and as the augmented MacT<sub>E</sub>X for the latter).

How does one go about using L<sub>A</sub>T<sub>E</sub>X? There are two approaches. The first is to edit the L<sub>A</sub>T<sub>E</sub>X source in a general purpose text editor such as Emacs, Vim, or a system-specific text editor. Ideally, the editor is customized so that it syntax highlights the L<sub>A</sub>T<sub>E</sub>X source, can directly compile the document, can pinpoint the location of compilation errors in the source, and can invoke a preview of the compiled document with two-way synchronization between the location of the cursor in the source and the page of the preview. I use Emacs together with the AUCT<sub>E</sub>X and RefT<sub>E</sub>X packages, which provides an extremely powerful L<sub>A</sub>T<sub>E</sub>X environment; indeed I use Emacs for all my editing tasks, ranging from programming to writing emails. A popular alternative is to use a program designed specifically for editing L<sub>A</sub>T<sub>E</sub>X documents, which typically comes with an integrated previewer. Such programs tend to be system-specific.

T<sub>E</sub>X compiles to its own DVI (device independent) file format, which can then be translated into PostScript, a file format commonly used for printing. The standard format for distributing documents is now PDF. While PostScript can be converted to PDF, versions of T<sub>E</sub>X and L<sub>A</sub>T<sub>E</sub>X that compile directly to PDF are available (typically invoked as `pdftex` and `pdflatex`). Whether one is using the DVI-based or PDF-based versions of L<sub>A</sub>T<sub>E</sub>X affects how one generates graphics files for inclu-

sion in figures. For DVI, included figures are typically in encapsulated PostScript format, whereas for PDF graphics files are typically in PDF or JPEG format and PostScript files are not allowed. I use a PDF workflow, though the Companion itself uses DVI and PostScript, because many of the figures in the book needed fine-tuning and this is more easily done in PostScript than in PDF. It is important to note that the Adobe Acrobat program is not suitable for use as a PDF previewer in the edit-compile-preview cycle as it does not refresh the view when a PDF file is updated on disk. Various open-source alternatives are available that do not have this limitation.

## 2 Preparing a Bibliography

A potentially time-consuming and error-prone part of writing a paper is preparing the bibliography, which contains the bibliographic details of the articles that are cited. In a  $\LaTeX$  document the bibliography entries are cited with a command of the form `\cite{smit65}`, where `smit65` is a key that uniquely specifies the entry in a `bibliography` environment that contains the item being cited.  $\LaTeX$  has a companion program called `BibTeX` that extracts bibliography entries from a database contained in a `bib` file (an ASCII file of a special structure with a `.bib` extension) and automatically creates the `bibliography` environment. Figure 2 shows an example of a `bib` file entry. Using `BibTeX` is a great time-saver and ensures accurate bibliographies, assuming that the `bib` file is accurate and kept up to date. Most journal web sites allow `BibTeX` entries for papers to be downloaded, so it is easy to build up a personal `bib` file. There exist open source `BibTeX` reference managers (such as `JabRef`) that facilitate creating and maintaining `bib` files.

A digital object identifier (DOI) is a character string that uniquely identifies an electronic document. It can be resolved into a uniform resource locator (URL) by preceding it with the string `http://dx.doi.org/`. Nowadays most papers (and many books) have DOIs and many older papers have been assigned DOIs. A DOI remains valid even if the location of the document changes, provided that the publisher updates the metadata. It is recommended to record DOIs in `BibTeX` databases, and it is then possible with the use of a suitable `BibTeX` style file and the  $\LaTeX$  `hyperref` package to include clickable links in a paper's bibliography (for example, from a paper's title).

```
@book{knut86,
  author = "Donald E. Knuth",
  title = "The  $\TeX$  book",
  publisher = "Addison-Wesley",
  address = "Reading, MA, USA",
  year = 1986,
  pages = "ix+483",
  isbn = "0-201-13448-9"}
```

Figure 2 An example of a `BibTeX` bibliography entry.

## 3 Graphics

Mathematics papers often contain figures that plot functions, depict physical setups, or graph experimental results. These can be produced in many different ways. In a  $\LaTeX$  workflow one can generate a graphic outside  $\LaTeX$  and then include it as an external JPEG, PostScript, or PDF file, or generate it from within  $\LaTeX$ . The most popular  $\LaTeX$  packages for graphics are `TikZ` and `PGFPlots`, which are built on top of the low-level primitives provided by the `PGF` (portable graphics format) package. Most of the figures in part I of this volume were generated using these packages. A major benefit of them is that they can incorporate  $\LaTeX$  commands and fonts, thus providing consistency with the main text. These powerful packages are not easy to use, but one can usually find an example online that provides a starting point for modification.

## 4 Version Control and Backups

Every good workflow contains procedures for making regular backups of files and recording a history of different versions of the files. Backups store one or more copies of the current version of key files on a separate disk or machine, so that a hard disk failure or loss of a complete machine does not result in loss of files. Version control serves a different purpose, which is to record in a repository intermediate states of files so that authors can revert to an earlier version of a file or reinstate part of one. The use of the plural in “authors” refers to the fact that version control systems allow more than one user to contribute to a repository and allow any user to check out the latest versions of files. Although version control originated in software development, it is equally useful for documents. As long as the repository is kept on a different disk or machine, version control also provides a form of backup.

Of course a simple version control system is to regularly copy a file to another directory, renaming it with a

~~Part of every good workflow is~~ Every good workflow contains procedures for making regular backups of files and recording a history of different versions of the files. Backups ~~provide~~ store one or more copies of the current version of key files on a separate disk or machine, so that a hard disk failure or loss of a complete machine does not result in ~~serious~~ loss of files. Version control serves a different purpose, which is to record ~~in a repository~~ intermediate states of files so

Figure 3 Example of output from `latexdiff`.

version number (paper1.tex, paper2.tex, ...). However, this is tedious and error-prone. A proper version control system keeps files in a database and stores only the lines that have changed between one version and the next. Popular version control systems include Git and Subversion (SVN). Although these are command based and can be difficult to learn, graphical user interfaces (GUIs) are available that simplify their usage.

Microsoft Word's "track changes" feature provides annotations of who made what changes to a document (and is a primitive and widely used form of version control). In  $\text{\LaTeX}$  a similar effect can be achieved by using the `latexdiff` command-line program provided with some  $\text{\LaTeX}$  distributions, which takes as input two different version of a  $\text{\LaTeX}$  file and produces a third  $\text{\LaTeX}$  file that marks up the differences between them; see figure 3 for an example.

## 5 Computational Experiments

In papers that involve computational experiments one needs to include figures or tables summarizing the results. It is typical that as a paper is developed the experiments are refined and repeated. One therefore needs an efficient way to regenerate tables and figures. Cutting and pasting the output of a program into the paper is not a good approach. It is much better to make the program output the results in a form that can be directly included in the paper (e.g., via an `\input` command in  $\text{\LaTeX}$ ). Literate programming techniques allow program code to be included within the source code for a paper and they automate the running of the programs and the insertion of the results back into the paper source code.

## 6 Putting it All Together

Here are the things I do when I start to write a paper. I create a directory (folder) with a name that denotes the project in question. In that directory I copy a file

**Fréchet Derivative**

---

**Fréchet derivative of  $f : \mathbb{C}^{n \times n} \rightarrow \mathbb{C}^{n \times n}$  at  $X \in \mathbb{C}^{n \times n}$**

A linear mapping  $L : \mathbb{C}^{n \times n} \rightarrow \mathbb{C}^{n \times n}$  s.t. for all  $E \in \mathbb{C}^{n \times n}$

$$f(X + E) - f(X) - L(X, E) = o(\|E\|).$$

Example For  $f(X) = X^2$  we have

$$f(X + E) - f(X) = XE + EX + E^2,$$

so  $L(X, E) = XE + EX$ .

Nick Higham
Matrix Functions
17 / 28

Figure 4 A Beamer slide.

`paper.tex` from a recent paper that I have written and use it as a template. I delete most of the content of `paper.tex` but keep the macros and some of the basic structural commands. I set up a repository in my version control system and commit `paper.tex` to it. I create a subdirectory for the computer programs I will write and a subdirectory named `figs` into which the PDF figures will be placed.

## 7 Presentations

As well as writing a paper about a piece of research one may want to give a presentation about it in a seminar or at a conference. This will normally involve preparing slides or a poster, although it is still sometimes possible to give a blackboard talk.  $\text{\LaTeX}$  has excellent tools for preparing slides and posters.

The Beamer class is the most widely used way to prepare slides in  $\text{\LaTeX}$ . It can create overlays, allowing a slide to change dynamically (perhaps as an equation is built-up, a piece at a time). Slide color and background, and elements such as a header (which may contain a mini table of contents) and a footer, are all readily customized. Figure 4 shows an example slide.

Various  $\text{\LaTeX}$  packages are available for producing posters, at up to A0 paper size. A popular one is the `beamerposter` package built on Beamer.

## 8 Collaboration

In the early days of the internet the most common way for authors to collaborate was to email documents back and forth. A regularly encountered problem was that Unix mailers would insert a greater than sign in front of any word "from" that appeared at the start of a line of a plain text message, so  $\text{\LaTeX}$  files would often have stray `>` characters. For many people, email still

serves as a useful mechanism for collaborative writing, but more sophisticated approaches are available. A file hosting service such as Dropbox enables a group of users to share and synchronize a folder on their disks via the cloud. Version control based on a shared repository hosted on the internet is the most powerful approach; it is widely used by programmers (for example on sites such as GitHub and SourceForge) and increasingly popular with authors of papers.

## 9 Workflow for This Book

I wrote my articles using Emacs and  $\text{\TeX}$ Live, with all files under version control with Git. I edited some of my figures in Adobe Photoshop. The production editor/typesetter, Sam Clark, used WinEdit and  $\text{\TeX}$ Live with a PostScript-based workflow, editing PostScript figures in Adobe Illustrator.

I produced a draft index for the articles that I authored using  $\text{\LaTeX}$  indexing commands and the MakeIndex program. A professional indexer then expanded the index to cover the whole book.

The font used for this book is Lucida Bright, which has a full set of mathematical symbols that work well in  $\text{\TeX}$ . It is from the same family as the Lucida Grande sans serif font that was used throughout the Mac OS X user interface up until version 10.9.

## 10 Further Reading

Of the many good references on  $\text{\LaTeX}$  I recommend Griffiths and D. J. Higham (1997) for a brief introduction and Kopka and Daly (2004) for a more comprehensive treatment. Knuth (1986) continues to be worth reading, even for those who use only  $\text{\LaTeX}$ . Various aspects of workflow are covered in Higham (1998). Version control is best explored with the many freely available web resources.

A good place to start looking for information about  $\text{\TeX}$  and  $\text{\LaTeX}$  is the website of the  $\text{\TeX}$  Users Group, <http://tug.org>. A large collection of  $\text{\LaTeX}$  packages is available at the Comprehensive TeX Archive Network (CTAN), <http://www.ctan.org>.

- [1] David F. Griffiths and Desmond J. Higham. *Learning  $\text{\LaTeX}$* . Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997. x+84 pp. ISBN 0-89871-383-8.
- [2] Nicholas J. Higham. *Handbook of Writing for the Mathematical Sciences*. Second edition, Society for

Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998. xvi+302 pp. ISBN 0-89871-420-6.

- [3] Donald E. Knuth. *The  $\text{\TeX}$ book*. Addison-Wesley, Reading, MA, USA, 1986. ix+483 pp. ISBN 0-201-13448-9.
- [4] Helmut Kopka and Patrick W. Daly. *Guide to  $\text{\LaTeX}$* . Fourth edition, Addison-Wesley, Boston, MA, USA, 2004. xii+597 pp. ISBN 0-321-17385-6.