On the Technicalities of Scientific Writing Anno 2012: The Doconce Way

Hans Petter Langtangen

Jan 12, 2013

1 Scope

- *scientific writing* = lecture notes, slides, reports, thesis, books, ...
- (Journal papers typeset by journals are out of scope)
- Scope: documents with much *math* and *computer code*
- Key question: What tools should I use for writing?
- Default answer: LaTeX
- Alternative: MS Word w/math
- Recent popular alternative tools: Sphinx, Markdown, MediaWiki, IPython notebook

2 Scientific writing needs to address many new media

- Old days (1985-2005): mostly black-and-white documents aimed at printing
- Now:
  1. Black-and-white books
  2. Colorful books and PDFs
  3. PDFs with hyperlinks
  4. HTML web pages (plain or fancy design)
  5. Wikis
  6. epub
- LaTeX does not support all of this
- We need to write for multiple formats!

3 Popular tools anno 2012

- **LaTeX**: de facto standard in math-instensive sciences
- **pdfLaTeX**: takes over (figures in png, pdf)
- **Word**: popular, but limited math support and not so good-looking math fonts
- **HTML with MathJax**: "full" LaTeX math
- **Sphinx**: limited LaTeX math support, great support for web design
- **reStructuredText**: no math support, transforms to lots of formats (LaTeX, HTML, XML, Word, OpenOffice, ...)
- **Markdown**: email-style untagged formatting, limited LaTeX math support, transforms to lots of formats (LaTeX, HTML, XML, Word, OpenOffice, ...)
- **MediaWiki**: "full" LaTeX math support (Wikipedia)
- Other **wiki** formats: no math support, great for collaborative editing
- **Epydoc**: old tool for Python code documentation
- **IPython notebooks**: combines Python code, interactivity and Markdown writing
- **Plain text for email** (no tagging)

4 LaTeX is very rich; other tools support only some elements

- LaTeX inline math: works with all (LaTeX, MathJax, Sphinx, Markdown, MediaWiki)
- LaTeX equation math:
  - **LaTeX**: `equation*`, `equation`, `align*`, `align` + `eqnarray`, `split`, `alignat`, ...
  - **MathJax**: `equation*`, `equation`, `align*`, `align`
  - **MediaWiki**: `equation*`, `equation`, `align*`, `align`
  - **Sphinx**: `equation*`, `equation`, `align*`
  - **Markdown**: `equation*`, `equation`

5 LaTeX is very rich; other tools support only some elements

- Figures: all
- Subfigures: LaTeX (`subfigure`)
- Movies: LaTeX (can run separately), just raw embedded HTML in others
- Floating computer code: LaTeX
- Fixed computer code: all
- Floating tables: LaTeX
- Fixed tables: all
- Algorithms: LaTeX
- Margin notes: LaTeX
- Footnotes: LaTeX, Sphinx, reStructuredText, MediaWiki
- Bibliography: LaTeX, Sphinx, reStructuredText, MediaWiki
- Hyperlinks: all (but does not work on paper!)

Conclusion: Highly non-trivial to translate a LaTeX document into something based on HTML and vice versa.

6 Concerns I

- Sphinx refers to figures by the caption (has to be short!) and strips away any math notation (avoid that!).
- Sphinx refers to sections by the title, but removes math in the reference, so avoid math in headlines.
- Algorithms must be written up using basic elements like lists or paragraphs with headings.
- Tables cannot be referred to by numbers and have to appear at fixed positions in the text.
- Computer code has to appear at fixed positions in the text.

7 Concerns II

- Footnotes must appear as part of the running text (e.g., sentences surrounded by parenthesis), since only a few formats support footnotes.
- Sphinx does not handle code blocks where the first line is indented.
- Multiple plots in the same figure: mount the plots to one image file and include this (`montage` for png, gif, jpeg; `pdftk`, `pdfnup`, and `pdfcrop` for PDF).
- Keys for items in the bibliography are made visible by Sphinx so "bibitems" a la BibTeX must look sensible and consistent.
- If you need several equations numbered in an `align` environment, recall that Sphinx and Markdown cannot handle this.

8 Concerns III

- Index words can appear anywhere in LaTeX, but should be outside paragraphs in other tools.
- References to tables, program code and algorithms can only be made in LaTeX.
- Figures are floating in LaTeX, but fixed in other tools, so place figures exactly where they are needed the first time.
- Curve plots with color lines do not work well in black-and-white printing. Make sure plots makes sense in color and BW (e.g., by using colors *and* markers).

9 Solution I: Use a format that translates to many

- Sphinx can do nice HTML, LaTeX, epub, (almost) plain text, man pages, Gnome devhelp files, Qt help files, texinfo, JSON
- Markdown can do LaTeX, HTML, MS Word, OpenOffice, XML, reStructuredText, epub, DocBook, ... but not Sphinx
- IPython notebook: can do LaTeX, reStructuredText, HTML, PDF, Python script
- Sphinx and Markdown has some limited math support

10 Solution II: Use Doconce

Doconce offers minimalistic typing, great flexibility wrt format, especially for scientific writing.

- Can generate LaTeX, HTML, Sphinx, Markdown, MediaWiki and other wiki formats
- Good support for math and code
- Great flexibility for typesetting code
- Made for science books *and* smaller teaching modules
- Support for code snippets from files, embedded movies, warnings/hint/info, generalized links
- Support for HTML5 slides - short way from prose to slides
- Integrates with preprocessors: preprocess and mako
- Handles multiple formats for figures
- Between Mardown and Sphinx wrt tagging (and richness)

11 Doconce demo

`http://hplgit.github.com/teamods/writing_reports/`

- HTML with MathJax
- PDF from LaTeX
- Sphinx (agni theme)
- Sphinx (pyramid theme)
- Sphinx (classy theme)
- Sphinx (fenics theme)
- Sphinx (redcloud theme)
- Doconce source
- Doconce tutorial

12 A tour of Doconce

13 Doconce: title, authors, date, toc

```
TITLE: Some Title
AUTHOR: name1 at institution1, with more info, and institution2
AUTHOR: name2 email:name2@web.com at institution
DATE: today

# A table of contents is optional:
TOC: on
```

NOTICE

Title and authors must have all information *on a single line*!

14 Doconce: abstract

```
__Abstract.__
Here goes the abstract...
```

Or:

```
__Summary.__
Here goes the summary...
```

15 Doconce: section headings

Headings are marked with =:

```
======= This is an H2/section heading =======

===== This is an H3/subsection heading =====

=== This is an H4/paragraph heading ===

__This is a paragraph heading.__
```

16 Doconce: markup and lists

```
 * Bullet list items start with `*`
   and may span several lines
 * Ordered (enumerated) list items start with `o`
 * *Emphasized words* are possible
 * _Boldface words_ are also possible
 * `inline verbatim code` is featured
   * sublists too
   * just indent...
```

This gets rendered as

- Bullet lists start with * and may span several lines
- Ordered (enumerated) list items start with o

- *Emphasized words* are possible
- **Boldface words** are also possible
- `inline verbatim code` is featured
  - sublists too
  - just indent...

17 Doconce: labels, references, index items

```
# Insert index items in the source
idx{key word1} idx{key word2}

# Label
===== Some section =====
label{this:section}

# Make reference
As we saw in Section ref{this:section}, references, index
items and labels follow a syntax similar to LaTeX
but without backslashes.

# Make reference to equations
See (ref{eq1})-(ref{myeq}).

# Make hyperlink
"some link text": "http://code.google.com/p/doconce/"

# Hyperlink with complete URL as link text
URL: "http://code.google.com/p/doconce/"
```

18 Doconce: figures and movies

NOTICE

Figure with HTML and LaTeX info, and caption, *all on one line*:

```
FIGURE: [figdir/myfig, width=300 frac=1.2] My caption. label{fig1}

# This figure will be 300 pixels wide in HTML and span 1.2 times
# the linewidth in LaTeX.
```

Movies are also supported:

```
MOVIE: [http://www.youtube.com/embed/P8VcZzgdfSc, width=420 height=315]
```

and rendered as

: `http://www.youtube.com/watch?v=P8VcZzgdfSc`

19 Doconce: math

Inline math as in LaTeX:

```
...where $a=\int_{\Omega}fdx$ is an integral.
```

gets rendered as ...where $a = \int_\Omega f dx$ is an integral.

An equation environment is surrounded by bt! and et! tags, the rest is plain LaTeX:

```
!bt
\begin{align}
\frac{\partial u}{\partial t} &= \nabla^2 u,
label{a:eq}\\
\nabla\cdot\pmb{v} & = 0
label{b:eq}
\end{align}
!et
```

which is rendered as

$$\frac{\partial u}{\partial t} = \nabla^2 u, \tag{1}$$

$$\nabla \cdot \boldsymbol{v} = 0 \tag{2}$$

Limit math environments to

```
\[ ... \]
```

```
\begin{equation*}
\end{equation*}
```

```
\begin{equation}
\end{equation}
```

```
\begin{align*}
\end{align*}
```

```
\begin{align}
\end{align}
```

20 Doconce: displaying code

Code is enclosed in bc! and ec! tags:

```
!bc pycod
def solver(I, a, T, dt, theta):
    """Solve u'=-a*u, u(0)=I, for t in (0,T] with steps of dt."""
    dt = float(dt)           # avoid integer division
    N = int(round(T/dt))     # no of time intervals
    T = N*dt                 # adjust T to fit time step dt
    u = zeros(N+1)           # array of u[n] values
    t = linspace(0, T, N+1)  # time mesh

    u[0] = I                 # assign initial condition
    for n in range(0, N):    # n=0,1,...,N-1
        u[n+1] = (1 - (1-theta)*a*dt)/(1 + theta*dt*a)*u[n]
    return u, t
!ec
```

This gets rendered as

```
def solver(I, a, T, dt, theta):
    """Solve u'=-a*u, u(0)=I, for t in (0,T] with steps of dt."""
    dt = float(dt)            # avoid integer division
    N = int(round(T/dt))      # no of time intervals
    T = N*dt                  # adjust T to fit time step dt
    u = zeros(N+1)            # array of u[n] values
    t = linspace(0, T, N+1)   # time mesh

    u[0] = I                  # assign initial condition
    for n in range(0, N):     # n=0,1,...,N-1
        u[n+1] = (1 - (1-theta)*a*dt)/(1 + theta*dt*a)*u[n]
    return u, t
```

The bc! command can be followed by a specification of the computer language: `pycod` for Python code snippet, `pypro` for complete Python program, `fcod` for Fortran snippet, `fpro` for Fortran program, and so forth (`c` for C, `cpp` for C++, `sh` for Unix shells, `m` for Matlab).

21 Doconce: copying code from source files

We recommend to copy as much code as possible directly from the source files:

```
@@@CODE path/to/file
@@@CODE path/to/file   fromto: start-regex@end-regex
```

For example, copying a code snippet starting with `def solver(` and ending with (line not included) `def next(x, y,` is specified by start and end regular expressions:

```
@@@CODE src/dc_mod.py  fromto: def solver\(@def next\(x,\s*y,
```

Typesetting of code is implied by the file extension:

- `.py`: `pypro` if complete file, `pycod` if snippet
- `.f`, `.f90`, `f.95`: `fpro` and `fcod`
- `.cpp`, `.cxx`: `cpppro` and `cppcod`
- `.c`: `cpro` and `ccod`
- `.*sh`: `shpro` and `shcod`
- `.m`: `mpro` and `mcod`
- `ptex2tex` can be used to choose between 40+ typesettings of computer code in LaTeX
- `pygments` is used for code typesetting in HTML (about 10 different styles)

22 Doconce: tables

```
|--------------------------------|
|time  | velocity | acceleration |
|---r-------r----------r---------|
| 0.0  | 1.4186   | -5.01        |
| 2.0  | 1.376512 | 11.919       |
| 4.0  | 1.1E+1   | 14.717624    |
|--------------------------------|
```

Gets rendered as

time	velocity	acceleration
0.0	1.4186	-5.01
2.0	1.376512	11.919
4.0	1.1E+1	14.717624

23 Doconce: newcommands for math

- `newcommands*.tex` files contain newcommands
- Used directly in LaTeX
- Substitution made for many other formats

24 Doconce: exercises

Doconce offers a special format for *exercises*, *problems* and *projects*:

```
===== Problem: Flip a Coin =====
label{demo:ex:1}

files = flip_coin.py, flip_coin.pdf
solutions = mysol.txt, mysol_flip_coin.py
keywords = random numbers; Monte Carlo simulation

!bsubex
Make a program that simulates flipping a coin $N$ times.

!bhint
Use ‘r = random.random()‘ and define head as ‘r <= 0.5‘.
!ehint
!esubex

!bsubex
Compute the probability of getting heads.

!bans
A short answer: 0.5.
!eans

!bsol
A full solution to this subexercise can go here.
!esol
!esubex

!bsubex
Make another program that computes the probability
of getting at least three heads out of 5 throws.
!esubex
```

25 Doconce: exercises

Last page gets rendered as follows:

Problem 1: Flip a Coin

a) Make a program that simulates flipping a coin N times.

Hint. Use `r = random.random()` and define head as `r <= 0.5`.

b) Compute the probability of getting heads.

Answer. A short answer: 0.5.

Solution. A full solution to this subexercise can go here.

c) Make another program that computes the probability of getting at least three heads out of 5 throws.

Filenames: `flip_coin.py`, `flip_coin.pdf`.

26 Doconce: exercises

All *exercises*, *problems*, and *projects* in a document are parsed and available in a data structure (list of dicts) for further processing.

```
[{'answer': '',
  'closing_remarks': '',
  'file': ['flip_coin.py', 'flip_coin.pdf'],
  'heading': '=====',
  'hints': [],
  'keywords': ['random numbers', 'Monte Carlo simulation'],
  'label': 'demo:ex:1',
  'no': 1,
  'solution': '',
  'solution_file': ['mysol.txt', 'mysol_flip_coin.py'],
  'subex': [{'answer': '',
             'file': None,
             'hints': ['Use `r = random.random()` ...'],
             'solution': '',
             'text': 'Make a program that simulates ...'},
            {'answer': 'A short answer: 0.5.',
             'file': None,
             'hints': [],
             'solution': 'A full solution to this ...',
             'text': 'Compute the probability of ...'},
            {'answer': '',
             'file': None,
             'hints': [],
             'solution': '',
             'text': 'Make another program that computes ...'}],
  'text': '',
  'title': 'Flip a Coin',
  'type': 'Problem'}]
```

27 Doconce: use of preprocessors

- Simple if-else tests a la C preprocessor
- `FORMAT` variable can be used to test on format
  - if latex/pdflatex do one sort of code (raw LaTeX)
  - if html, do another type of code (raw HTML)
- Easy to comment out large portions of text
- Easy to make different versions of the document
- The mako preprocessor is really powerful - gives a complete programming language inside the document!

28 Doconce: slides

Very effective way to generate slides from running text:

- Take a copy of your Doconce prose
- Strip off as much text as possible
- Emphasize key points in bullet items
- Focus on figures and movies
- Focus on key equations
- Focus on key code snippets
- Insert split! wherever you want a new slide to begin
- Insert bpop! and epop! around elements to pop up in sequence
- Use 7 = or 5 = in headings (H2 or H3)
- Slides are made with HTML5 tools such as reveal.js, deck.js, csss, or dzslides

29 Doconce: example on slide code

```
!split
======= Headline =======

 * Key point 1
 * Key point 2

FIGURE: [fig/teacher1, width=150]

Key equation:

\[ -\nabla^2 u = f \quad\hbox{in }\Omega \]

And maybe a final comment?

!split
======= Next slide... =======
```

30 Doconce: example on slide code

Last page gets rendered to

31 Headline

- Key point 1
- Key point 2

Key equation:

$$-\nabla^2 u = f \quad \text{in } \Omega$$

And maybe a final comment?

32 Doconce: slide styles

- Supported HTML5 packages:

  - reveal.js
  - deck.js
  - dzslides
  - csss
  - html5slides (experimental)

- **Problem**: each package has its (similar) syntax
  - **Solution**: slide code is autogenerated from compact Doconce syntax
- **Problem**: reveal and deck have numerous styles
  - **Solution**: easy to autogenerate all styles for a talk
- **Problem**: HTML5 slides need many style files
  - **Solution**: autocopy all files to talk directory
- **Problem**: original versions of the styles have too large fonts, centering, and other features not so suitable for lectures
  - **Solution**: Doconce contains adjusted css files

33 Doconce: output in HTML

Run in terminal window:

```
doconce format html doconcefile

# Solarized HTML style
doconce format html doconcefile --html-solarized

# Control pygments typesetting of code
doconce format html doconcefile --pygments-html-style=native

# Or use plain <pre> tag
doconce format html doconcefile --no-pygments-html

# Further making of slides
doconce slides_html doconcefile reveal --html-slide-theme=darkgray
```

34 Doconce: output in pdfLaTeX

```
doconce format pdflatex doconcefile

# Result: doconcefile.p.tex (ptex2tex file)
# Run either
ptex2tex doconcefile
# or
doconce ptex2tex doconcefile -DHELVETICA envir=minted

pdflatex doconcefile
```

```
bibtex doconcefile
pdflatex doconcefile

# More control of how code is typeset
doconce format pdflatex doconcefile --minted-latex-style=trac
doconce ptex2tex doconcefile envir=minted

doconce format pdflatex doconcefile
doconce ptex2tex doconcefile envir=ans:nt
```

35 Doconce: output in Sphinx

```
doconce format sphinx doconcefile

# Autocreate sphinx directory
doconce sphinx_dir theme=pyramid doconcefile
# Copy files and build HTML document
python automake-sphinx.py

google-chrome sphinx-rootdir/_build/html/index.html
```

Much easier than running the Sphinx tools manually...

36 Doconce: output in other formats

```
doconce format pandoc doconcefile  # (Pandoc extended) Markdown
doconce format mwiki  doconcefile  # MediaWiki
doconce format gwiki  doconcefile  # Googlecode wiki
doconce format cwiki  doconcefile  # Creole wiki (Bitbucket)
doconce format rst    doconcefile  # reStructuredText
doconce format plain  doconcefile  # plain, untagged text for email
```

37 Doconce: installation

- Ubuntu: `sudo apt-get install python-doconce` (old version)
- Source at Googlecode (recommended!)
- Check out source, `sudo python setyp.py install`
- Many dependencies...
- Must have `preprocess` and `mako`
- Need `latex`, `sphinx`, `pandoc`, etc. (see Installation in manual)
- For slides: only `preprocess` is needed :-)

38 Writing tips

- See the previous *Concerns I, II and III* slides for issues when writing for multiple formats. However: Doconce makes a fix so that Sphinx works with labels in align environments :-)
- Prepare figures in the right format: EPS for `latex`, PDF for `pdflatex`, PNG, GIF or JPEG for HTML formats (`html`, and HTML output from `sphinx`, `rst`, `pandoc`). One can omit the figure file extension and `doconce` will pick the most appropriate file for the given output format.
- Let plotting programs produce both PDF/EPS and PNG files. (Recall that PDF and EPS are vector graphics formats that can scale to any size with much higher quality than PNG or other bitmap formats.)
- Use `doconce combine_images` to combine several images into one.
- Use `doconce spellcheck *.do.txt` to automatically spellcheck files.

39 Writing tips for sphinx and other formats

For output formats different from `latex`, `pdflatex`, and `html`, the following points are important:

- Do not use math in section headings or figure captions if output in `sphinx` is wanted (such math are removed in references).
- Let all running text start in column 1 (`sphinx` is annoyed by intended lines).
- Use progressive section headings: after chapter (`=========`) comes section (`=======`), and then subsection (`=====`) before paragraph heading (`===`). "Jumps", say `===` after `======` works fine for `latex`, `pdflatex`, and `html`, but not for `rst` and `sphinx`.
- Place index entries (`\index{keyword}`) before the paragraph where they are introduced (not inside the text). Also place index entries before `===` headings.
- Be careful with labels in `align` math environments: `pandoc` and `mwiki` cannot refer to them.
- Always have a line of running text before a code snippet or program.
- Do not insert comments before intented text, such as computer code and lists.