# Missing: FIGURE, MOVIE, environments

TITLE: Doconce: Document Once, Include Anywhere

AUTHOR: Hans Petter Langtangen at Simula Research Laboratory and University of O

slo

DATE: today

- \* When writing a note, report, manual, etc., do you find it difficult to choose the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?
- \* Do you need to write documents in varying formats but find it difficult to remember all the typesetting details of various formats like "LaTeX": "http://refcards.com/docs/silvermanj/amslatex/LaTeXRefCard.v2.0.pdf", "HTML": "http://www.htmlcodetutorial.com/", "reStructuredText": "http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html", "Sphinx": "http://sphinx.pocoo.org/contents.html", and "wiki": "http://code.google.com/p/support/wiki/WikiSyntax"? Would it be convenient

to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?

\* Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.

====== What Does Doconce Look Like? ======

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. Here are som examples.

- \* Bullet lists arise from lines starting with '\*'.
- \* \*Emphasized words\* are surrounded by `\*`.
- \* \_Words in boldface\_ are surrounded by underscores.
- \* Words from computer code are enclosed in back quotes and then typeset 'verbatim (in a monospace font)'.
- \* Section headings are recognied by equality ('=') signs before and after the title, and the number of '=' signs indicates the level of the section: 7 for main section, 5 for subsection, and 3 for subsubsection.
- \* Paragraph headings are recognized by a double underscore before and after the heading.
- \* The abstract of a document starts with \*Abstract\* as paragraph heading, and all text up to the next heading makes up the abstract,

,,

- \* Blocks of computer code can easily be included by placing '!bc' (begin code) and '!ec' (end code) commands at separate lines before and after the code block.
- \* Blocks of computer code can also be imported from source files.
- \* Blocks of LaTeX mathematics can easily be included by placing '!bt' (begin TeX) and '!et' (end TeX) commands at separate lines before and after the math block.
- \* There is support for both LaTeX and text-like inline mathematics.
- \* Figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported.
- \* Invisible comments in the output format can be inserted throughout the text.
- \* Visible comments can be inserted so that authors and readers can comment upon the text (and at any time turn on/off output of such comments).
- \* There is an exercise environment with many advanced features.
- \* With a preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text.
- \* With Mako one can also have Python code embedded in the Doconce document and thereby parameterize the text (e.g., one text can describe programming in two languages).

Here is an example of some simple text written in the Doconce format: !bc

==== A Subsection with Sample Text ===== label{my:first:sec}

Ordinary text looks like ordinary text, and the tags used for \_boldface\_ words, \*emphasized\* words, and 'computer' words look natural in plain text. Lists are typeset as you would do in email,

- \* item 1
- \* item 2
- \* item 3

Lists can also have automatically numbered items instead of bullets,

- o item 1
- o item 2
- o item 3

URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl". If the word is URL, the URL itself becomes the link name, as in "URL": "tutorial.do.txt".

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to Section ref{my:first:sec}.

Doconce also allows inline comments of the form [name: comment] (with a space after 'name:'), e.g., such as [hpl: here I will make some remarks to the text]. Inline comments can be removed from the output by a command-line argument (see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

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
	· 	·

# lines beginning with # are comment lines

The Doconce text above results in the following little document:

==== A Subsection with Sample Text =====
label{my:first:sec}

Ordinary text looks like ordinary text, and the tags used for \_boldface\_ words, \*emphasized\* words, and 'computer' words look natural in plain text. Lists are typeset as you would do in an email,

- \* item 1
- \* item 2
- \* item 3

Lists can also have numbered items instead of bullets, just use an 'o' (for ordered) instead of the asterisk:

- o item 1
- o item 2
- o item 3

URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl". If the word is URL, the URL itself becomes the link name, as in "URL": "tutorial.do.txt".

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to Section ref{my:first:sec}.

Doconce also allows inline comments such as [hpl: here I will make some remarks to the text] for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

time	velocity	acceleration
r	r	r
0.0	1.4186	-5.01
2.0	1.376512	11.919

,

```
tutorial.do.txt
    4.0 | 1.1E+1 | 14.717624
==== Mathematics and Computer Code =====
Inline mathematics, such as \pi = \sin(x)  v = \sin(x),
allows the formula to be specified both as LaTeX and as plain text.
This results in a professional LaTeX typesetting, but in other formats
the text version normally looks better than raw LaTeX mathematics with
backslashes. An inline formula like \ln = \sin(x) is
typeset as
!bc
\ln x = \sin(x)
The pipe symbol acts as a delimiter between LaTeX code and the plain text
version of the formula. If you write a lot of mathematics, only the
output formats 'latex', 'pdflatex', 'html', 'sphinx', and 'pandoc'
are of interest
and all these support inline LaTeX mathematics so then you will naturally
drop the pipe symbol and write just
\ln = \sin(x)
However, if you want more textual formats, like plain text or reStructuredText,
the text after the pipe symbol may help to make the math formula more readable
if there are backslahes or other special LaTeX symbols in the LaTeX code.
Blocks of mathematics are typeset with raw LaTeX, inside
'!bt' and '!et' (begin TeX, end TeX) instructions:
!bc
bt
\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f, label{myeq1}\\
{\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
let
!ec
# Note: |bt and |et (and |bc and |ec below) are used to illustrate
# tex and code blocks in inside verbatim blocks and are replaced
# by !bt, !et, !bc, and !ec after all other formatting is finished. The result looks like this:
\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f, label{myeq1}\\ {\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
!et
Of course, such blocks only looks nice in formats with support
for LaTeX mathematics, and here the align environment in particular
(this includes 'latex', 'pdflatex', 'html', and 'sphinx'). The raw
LaTeX syntax appears in simpler formats, but can still be useful
for those who can read LaTeX syntax.
You can have blocks of computer code, starting and ending with
'!bc' and '!ec' instructions, respectively.
```

```
!bc
|bc pycod
from math import sin, pi
def myfunc(x):
   return sin(pi*x)
import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
lec
!ec
Such blocks are formatted as
!bc pycod
from math import sin, pi
def myfunc(x):
   return sin(pi*x)
import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
A code block must come after some plain sentence (at least for successful
output to 'sphinx', 'rst', and ASCII-close formats),
not directly after a section/paragraph heading or a table.
One can also copy computer code directly from files, either the
complete file or specified parts. Computer code is then never
duplicated in the documentation (important for the principle of
avoiding copying information!).
Another document can be included by writing '# #include "mynote.do.txt"'
at the beginning of a line. Doconce documents have
extension 'do.txt'. The 'do' part stands for doconce, while the
trailing '.txt' denotes a text document so that editors gives you
plain text editing capabilities.
==== Macros (Newcommands), Cross-References, Index, and Bibliography =====
label{newcommands}
Doconce supports a type of macros via a LaTeX-style *newcommand*
construction. The newcommands defined in a file with name
'newcommand_replace.tex' are expanded when Doconce is filtered to
other formats, except for LaTeX (since LaTeX performs the expansion
itself). Newcommands in files with names 'newcommands.tex' and
'newcommands_keep.tex' are kept unaltered when Doconce text is
filtered to other formats, except for the Sphinx format. Since Sphinx
understands LaTeX math, but not newcommands if the Sphinx output is
HTML, it makes most sense to expand all newcommands. Normally, a user
```

elsewhere throughout the text will usually be placed in 'newcommands\_replace.tex' and expanded by Doconce. The definitions of newcommands in the 'newcommands\*.tex' files \*must\* appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

will put all newcommands that appear in math blocks surrounded by '!bt' and '!et' in 'newcommands\_keep.tex' to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used

,

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the 'doc/manual/manual.do.txt' file (see the "demo page": "https://doconce.googlecode.com/hg/doc/demos/manual/index.html" for various formats of this document).

- # Example on including another Doconce file (using preprocess):
- # #include "\_doconce2anything.do.txt"

==== Demos =====

The current text is generated from a Doconce format stored in the file !bc

docs/tutorial/tutorial.do.txt

!ec

The file 'make.sh' in the 'tutorial' directory of the Doconce source code contains a demo of how to produce a variety of formats. The source of this tutorial, 'tutorial.do.txt' is the starting point. Running 'make.sh' and studying the various generated files and comparing them with the original 'tutorial.do.txt' file, gives a quick introduction to how Doconce is used in a real case. "Here": "https://doconce.googlecode.com/hg/doc/demos/tutorial/index.html" is a sample of how this tutorial looks in different formats.

There is another demo in the 'docs/manual' directory which translates the more comprehensive documentation, 'manual.do.txt', to various formats. The 'make.sh' script runs a set of translations.

# #include "../manual/install.do.txt"

# Doconce: Document Once, Include Anywhere

Hans Petter Langtangen<sup>1,2</sup>

<sup>1</sup>Simula Research Laboratory <sup>2</sup>University of Oslo

Jan 16, 2013

- When writing a note, report, manual, etc., do you find it difficult to choose the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LATEX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?
- Do you need to write documents in varying formats but find it difficult to remember all the typesetting details of various formats like LaTeX, HTML, reStructuredText, Sphinx, and wiki? Would it be convenient to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?
- Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.

#### 1 What Does Doconce Look Like?

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. Here are som examples.

- Bullet lists arise from lines starting with \*.
- Emphasized words are surrounded by \*.
- Words in boldface are surrounded by underscores.
- Words from computer code are enclosed in back quotes and then typeset verbatim (in a monospace font).

- Section headings are recognied by equality (=) signs before and after the title, and the number of = signs indicates the level of the section: 7 for main section, 5 for subsection, and 3 for subsubsection.
- Paragraph headings are recognized by a double underscore before and after the heading.
- The abstract of a document starts with Abstract as paragraph heading, and all text up to the next heading makes up the abstract,
- Blocks of computer code can easily be included by placing bc! (begin code) and ec! (end code) commands at separate lines before and after the code block.
- Blocks of computer code can also be imported from source files.
- Blocks of LaTeX mathematics can easily be included by placing bt! (begin TeX) and et! (end TeX) commands at separate lines before and after the math block.
- There is support for both LaTEX and text-like inline mathematics.
- Figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported.
- Invisible comments in the output format can be inserted throughout the text.
- Visible comments can be inserted so that authors and readers can comment upon the text (and at any time turn on/off output of such comments).
- There is an exercise environment with many advanced features.
- With a preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text.
- With Mako one can also have Python code embedded in the Doconce document and thereby parameterize the text (e.g., one text can describe programming in two languages).

Here is an example of some simple text written in the Doconce format:

```
===== A Subsection with Sample Text =====
label{my:first:sec}

Ordinary text looks like ordinary text, and the tags used for
_boldface_ words, *emphasized* words, and 'computer' words look
natural in plain text. Lists are typeset as you would do in email,
    * item 1
    * item 2
    * item 3
```

Lists can also have automatically numbered items instead of bullets,

```
o item 1 o item 2
```

URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl". If the word is URL, the URL itself becomes the link name, as in "URL": "tutorial.do.txt".

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to Section ref{my:first:sec}.

Doconce also allows inline comments of the form [name: comment] (with a space after 'name:'), e.g., such as [hpl: here I will make some remarks to the text]. Inline comments can be removed from the output by a command-line argument (see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

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

# lines beginning with # are comment lines

The Doconce text above results in the following little document:

# 1.1 A Subsection with Sample Text

Ordinary text looks like ordinary text, and the tags used for **boldface** words, *emphasized* words, and computer words look natural in plain text. Lists are typeset as you would do in an email,

- item 1
- item 2
- item 3

Lists can also have numbered items instead of bullets, just use an o (for ordered) instead of the asterisk:

- 1. item 1
- 2. item 2
- 3. item 3

URLs with a link word are possible, as in hpl. If the word is URL, the URL itself becomes the link name, as in tutorial.do.txt.

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to Section 1.1.

Doconce also allows inline comments such as (hpl: here I will make some remarks to the text) for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see Section 2 for an example).

Tables are also supperted, e.g.,

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

#### **Mathematics and Computer Code**

Inline mathematics, such as  $\nu = \sin(x)$ , allows the formula to be specified both as LATEX and as plain text. This results in a professional LATEX typesetting, but in other formats the text version normally looks better than raw LATEX mathematics with backslashes. An inline formula like  $\nu = \sin(x)$  is typeset as

```
\ln = \sin(x)
```

The pipe symbol acts as a delimiter between LATEX code and the plain text version of the formula. If you write a lot of mathematics, only the output formats latex, pdflatex, html, sphinx, and pandoc are of interest and all these support inline LATEX mathematics so then you will naturally drop the pipe symbol and write just

```
nu = \sin(x)
```

However, if you want more textual formats, like plain text or reStructuredText, the text after the pipe symbol may help to make the math formula more readable if there are backslahes or other special LATEX symbols in the LATEX code.

Blocks of mathematics are typeset with raw LATEX, inside bt! and et! (begin TeX, end TeX) instructions:

```
!bt
\begin{align}
{\hat u \neq \hat u \neq 1} 
{\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
!et
```

The result looks like this:

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

$$\frac{\partial u}{\partial t} = \nabla^2 u + f, \qquad (1)$$

$$\frac{\partial v}{\partial t} = \nabla \cdot (q(u)\nabla v) + g \qquad (2)$$

Of course, such blocks only looks nice in formats with support for LaTeX mathematics, and here the align environment in particular (this includes latex, pdflatex, html, and sphinx). The raw LaTeX syntax appears in simpler formats, but can still be useful for those who can read LaTeX syntax.

You can have blocks of computer code, starting and ending with bc! and ec! instructions, respectively.

```
!bc pycod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)
import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec
```

Such blocks are formatted as

```
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
```

A code block must come after some plain sentence (at least for successful output to sphinx, rst, and ASCII-close formats), not directly after a section/paragraph heading or a table.

One can also copy computer code directly from files, either the complete file or specified parts. Computer code is then never duplicated in the documentation (important for the principle of avoiding copying information!).

Another document can be included by writing # #include "mynote.do.txt" at the beginning of a line. Doconce documents have extension do.txt. The do part stands for doconce, while the trailing .txt denotes a text document so that editors gives you plain text editing capabilities.

# 1.3 Macros (Newcommands), Cross-References, Index, and Bibliography

Doconce supports a type of macros via a LaTeX-style <code>newcommand</code> construction. The newcommands defined in a file with name <code>newcommand\_replace.tex</code> are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names <code>newcommands.tex</code> and <code>newcommands\_keep.tex</code> are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by bt! and et! in <code>newcommands\_keep.tex</code> to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used

elsewhere throughout the text will usually be placed in newcommands\_replace.tex and expanded by Doconce. The definitions of newcommands in the newcommands\*.tex files *must* appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LATEX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LATEX, making it easy for Doconce documents to be integrated in LATEX projects (manuals, books). For further details on functionality and syntax we refer to the doc/manual/manual.do.txt file (see the demo page for various formats of this document).

#### 2 From Doconce to Other Formats

Transformation of a Doconce document mydoc.do.txt to various other formats applies the script doconce format:

```
Terminal  
Terminal  

Torninal  

Tornina
```

#### 2.1 Preprocessing

The preprocess and make programs are used to preprocess the file, and options to preprocess and/or make can be added after the filename. For example,

```
Terminal> doconce format latex mydoc -Dextra_sections -DVAR1=5 # preprocess
Terminal> doconce format latex yourdoc extra_sections=True VAR1=5 # mako
```

The variable FORMAT is always defined as the current format when running preprocess or mako. That is, in the last example, FORMAT is defined as latex. Inside the Doconce document one can then perform format specific actions through tests like #if FORMAT == "latex" (for preprocess) or % if FORMAT == "latex": (for mako).

#### 2.2 Removal of inline comments

The command-line arguments --no-preprocess and --no-make turn off running preprocess and make, respectively.

Inline comments in the text are removed from the output by

```
Terminal> doconce format latex mydoc --skip_inline_comments
```

One can also remove all such comments from the original Doconce file by running:

Terminal> doconce remove\_inline\_comments mydoc

This action is convenient when a Doconce document reaches its final form and comments by different authors should be removed.

#### 2.3 **HTML**

Making an HTML version of a Doconce file mydoc.do.txt is performed by

```
Terminal> doconce format html mydoc
```

The resulting file mydoc.html can be loaded into any web browser for viewing. The HTML style can be defined either in the header of the HTML file or in an external CSS file. The latter is enabled by the command-line argument --css=filename. There is a default style with blue headings, and a style with the solarized color palette, specified by the --html-solarized command line argument. If there is no file with name filename in the --css=filename specification, the blue or solarized styles are written to filename and linked from the HTML document. You can provide your own style sheet either by replacing the content inside the style tags or by specifying a CSS file through the --css=filename option.

If the Pygments package (including the pygmentize program) is installed, code blocks are typeset with aid of this package. The command-line argument --no-pygments-html turns off the use of Pygments and makes code blocks appear with plain (pre) HTML tags. The option --pygments-html-linenos turns on line numbers in Pygments-formatted code blocks.

The HTML file can be embedded in a template if the Doconce document does not have a title (because then there will be no header and footer in the HTML file). The template file must contain valid HTML code and can have three "slots": %(title)s for a title, %(date)s for a date, and %(main)s for the main body of text, i.e., the Doconce document translated to HTML. The title becomes the first heading in the Doconce document, and the date is extracted from the DATE: line, if present. With the template feature one can easily embed

the text in the look and feel of a website. The template can be extracted from the source code of a page at the site; just insert %(title)s and %(date)s at appropriate places and replace the main bod of text by %(main)s. Here is an example:

Terminal> doconce format html mydoc --html-template=mytemplate.html

#### 2.4 Blogs

Doconce can be used for writing blogs provided the blog site accepts raw HTML code. Google's Blogger service (blogname.blogspot.com) is particularly well suited since it also allows extensive LaTeX mathematics via MathJax. Write the blog text as a Doconce document without any title, author, and date. Then generate HTML as described above. Copy the text and paste it into the text area in the blog, making sure the input format is HTML. On Google's Blogger service you can use Doconce to generate blogs with LaTeX mathematics and pretty (pygmentized) blocks of computer code. See a blog example for details on blogging.



#### **WARNING**

In the comments after the blog one cannot paste raw HTML code with MathJax scripts so there is no support for mathematics in the comments.

WordPress (wordpress.com) allows raw HTML code in blogs, but has very limited LaTeX support, basically only formulas. The --wordpress option to doconce modifies the HTML code such that all equations are typeset in a way that is acceptable to WordPress. There is a doconce example on blogging with mathematics and code on WordPress.

#### 2.5 Pandoc and Markdown

Output in Pandoc's extended Markdown format results from

Pandoc supports latex, html, odt (OpenOffice), docx (Microsoft Word), rtf, texinfo, to mention some. The -R option makes Pandoc pass raw HTML or Latex to the output format instead of ignoring it, while the --toc option generates a table of contents. See the Pandoc documentation for the many features of the pandoc program.

Pandoc is useful to go from LATEX mathematics to, e.g., HTML or MS Word. There are two ways (experiment to find the best one for your document): doconce format pandoc and then translating using pandoc, or doconce format latex, and then going from LATEX to the desired format using pandoc. Here is an example on the latter strategy:

```
Terminal> doconce format latex mydoc
Terminal> doconce ptex2tex mydoc
Terminal> doconce replace '\Verb!' '\verb!' mydoc.tex
Terminal> pandoc -f latex -t docx -o mydoc.docx mydoc.tex
```

When we go through pandoc, only single equations or align\* environments are well understood.

Note that Doconce applies the Verb macro from the fancyvrb package while pandoc only supports the standard verb construction for inline verbatim text. Moreover, quite some additional doconce replace and doconce subst edits might be needed on the .mkd or .tex files to successfully have mathematics that is well translated to MS Word. Also when going to reStructuredText using Pandoc, it can be advantageous to go via LATEX.

Here is an example where we take a Doconce snippet (without title, author, and date), maybe with some unnumbered equations, and quickly generate HTML with mathematics displayed my MathJax:

```
Terminal> doconce format pandoc mydoc
Terminal> pandoc -t html -o mydoc.html -s --mathjax mydoc.mkd
```

The -s option adds a proper header and footer to the mydoc.html file. This recipe is a quick way of makeing HTML notes with (some) mathematics.

# 2.6 LATEX

Making a LATEX file mydoc.tex from mydoc.do.txt is done in two steps:

**Step 1.** Filter the doconce text to a pre-LaTeX form mydoc.p.tex for the ptex2tex program (or doconce ptex2tex):

```
Terminal> doconce format latex mydoc
```

LaTeX-specific commands ("newcommands") in math formulas and similar can be placed in files newcommands.tex, newcommands\_keep.tex, or newcommands\_replace.tex (see Section 1.3). If these files are present, they are included in the LATEX document so that your commands are defined.

An option --latex-printed makes some adjustments for documents aimed at being printed. For example, links to web resources are associated with a footnote listing the complete web address (URL).

Step 2. Run ptex2tex (if you have it) to make a standard LATEX file,

Terminal
Terminal> ptex2tex mydoc
In case you do not have ptex2tex, you may run a (very) simplified version:
Terminal> doconce ptex2tex mydoc
Note that Doconce generates a .p.tex file with some preprocessor macros that can be used to steer certain properties of the LATEX document. For example, to turn on the Helvetica font instead of the standard Computer Modern font, run
Terminal
reminar accounts point on my account of the transfer of the tr

The title, authors, and date are by default typeset in a non-standard way to enable a nicer treatment of multiple authors having institutions in common. However, the standard LATEX "maketitle" heading is also available through -DLATEX\_HEADING=traditional. A separate titlepage can be generate by -DLATEX\_HEADING=titlepage.

Preprocessor variables to be defined or undefined are

- BOOK for the "book" documentclass rather than the standard "article" class (necessary if you apply chapter headings)
- PALATINO for the Palatino font
- HELVETIA for the Helvetica font
- A4PAPER for A4 paper size
- A6PAPER for A6 paper size (suitable for reading on small devices)
- MOVIE15 for using the movie15 LATEX package to display movies
- PREAMBLE to turn the LATEX preamble on or off (i.e., complete document or document to be included elsewhere)

 MINTED for inclusion of the minted package (which requires latex or pdflatex to be run with the -shell-escape option)

The ptex2tex tool makes it possible to easily switch between many different fancy formattings of computer or verbatim code in LaTeX documents. After any bc! command in the Doconce source you can insert verbatim block styles as defined in your .ptex2tex.cfg file, e.g., bc sys! for a terminal session, where sys is set to a certain environment in .ptex2tex.cfg (e.g., CodeTerminal). There are about 40 styles to choose from, and you can easily add new ones.

Also the doconce ptex2tex command supports preprocessor directives for processing the .p.tex file. The command allows specifications of code environments as well. Here is an example:

```
Terminal doconce ptex2tex mydoc -DLATEX_HEADING=traditional \
-DPALATINO -DA6PAPER \
"sys=\begin{quote}\begin{verbatim}@\end{verbatim}\end{quote}" \
fpro=minted fcod=minted shcod=Verbatim envir=ans:nt
```

Note that @ must be used to separate the begin and end LTEX commands, unless only the environment name is given (such as minted above, which implies \begin{minted}{fortran} and \end{minted} as begin and end for blocks inside bc fpro! and ec!). Specifying envir=ans:nt means that all other environments are typeset with the anslistings.sty package, e.g., bc cppcod! will then result in \begin{c++}. If no environments like sys, fpro, or the common envir are defined on the command line, the plain \begin{verbatim} and \end{verbatim} used.

**Step 2b (optional).** Edit the mydoc.tex file to your needs. For example, you may want to substitute section by section\* to avoid numbering of sections, you may want to insert linebreaks (and perhaps space) in the title, etc. This can be automatically edited with the aid of the doconce replace and doconce subst commands. The former works with substituting text directly, while the latter performs substitutions using regular expressions. Here are two examples:

```
Terminal doconce replace 'section{' 'section*{' mydoc.tex
Terminal> doconce subst 'title\{(.+)Using (.+)\}' \
    'title{\g<1> \\\\ [1.5mm] Using \g<2>' mydoc.tex
```

A lot of tailored fixes to the LATEX document can be done by an appropriate set of text replacements and regular expression substitutions. You are anyway encourged to make a script for generating PDF from the LATEX file.

Step 3. Compile mydoc.tex and create the PDF file:

```
Terminal> latex mydoc
Terminal> latex mydoc
Terminal> makeindex mydoc # if index
Terminal> bibitem mydoc # if bibliography
Terminal> latex mydoc
Terminal> dvipdf mydoc
```

If one wishes to run ptex2tex and use the minted LATEX package for type-setting code blocks (Minted\_Python, Minted\_Cpp, etc., in ptex2tex specified through the \*pro and \*cod variables in .ptex2tex.cfg or \$HOME/.ptex2tex.cfg), the minted LATEX package is needed. This package is included by running ptex2tex with the -DMINTED option:

```
Terminal> ptex2tex -DMINTED mydoc
```

In this case, latex must be run with the -shell-escape option:

```
Terminal> latex -shell-escape mydoc
Terminal> latex -shell-escape mydoc
Terminal> makeindex mydoc  # if index
Terminal> bibitem mydoc  # if bibliography
Terminal> latex -shell-escape mydoc
Terminal> dvipdf mydoc
```

When running doconce ptex2tex mydoc envir=minted (or other minted specifications with doconce ptex2tex), the minted package is automatically included so there is no need for the -DMINTED option.

#### 2.7 PDFLaTeX

Running pdflatex instead of latex follows almost the same steps, but the start is

```
Terminal> doconce format latex mydoc
```

Then ptex2tex is run as explained above, and finally

```
Terminal> pdflatex -shell-escape mydoc
Terminal> makeindex mydoc # if index
Terminal> bibitem mydoc # if bibliography
Terminal> pdflatex -shell-escape mydoc
```

#### 2.8 Plain ASCII Text

We can go from Doconce "back to" plain untagged text suitable for viewing in terminal windows, inclusion in email text, or for insertion in computer source code:

```
Terminal> doconce format plain mydoc.do.txt # results in mydoc.txt
```

#### 2.9 reStructuredText

Going from Doconce to reStructuredText gives a lot of possibilities to go to other formats. First we filter the Doconce text to a reStructuredText file mydoc.rst:

```
Terminal > doconce format rst mydoc.do.txt

We may now produce various other formats:
```

We may now produce various other formats:

```
Terminal> rst2html.py mydoc.rst > mydoc.html # html
Terminal> rst2latex.py mydoc.rst > mydoc.tex # latex
Terminal> rst2xml.py mydoc.rst > mydoc.xml # XML
Terminal> rst2odt.py mydoc.rst > mydoc.odt # OpenOffice
```

The OpenOffice file mydoc.odt can be loaded into OpenOffice and saved in, among other things, the RTF format or the Microsoft Word format. However, it is more convenient to use the program unovonv to convert between the many formats OpenOffice supports on the command line. Run

```
Terminal> unoconv --show
```

to see all the formats that are supported. For example, the following commands take mydoc.odt to Microsoft Office Open XML format, classic MS Word format, and PDF:

```
Terminal> unoconv -f ooxml mydoc.odt
Terminal> unoconv -f doc mydoc.odt
Terminal> unoconv -f pdf mydoc.odt
```

Remark about Mathematical Typesetting. At the time of this writing, there is no easy way to go from Doconce and LaTeX mathematics to reST and further to OpenOffice and the "MS Word world". Mathematics is only fully supported by latex as output and to a wide extent also supported by the sphinx output format. Some links for going from LaTeX to Word are listed below.

- http://ubuntuforums.org/showthread.php?t=1033441
- http://tug.org/utilities/texconv/textopc.html
- http://nileshbansal.blogspot.com/2007/12/latex-to-openofficeword.html

#### 2.10 Sphinx

Sphinx documents demand quite some steps in their creation. We have automated most of the steps through the doconce sphinx\_dir command:

```
Terminal> doconce sphinx_dir author="authors' names" \
    title="some title" version=1.0 dirname=sphinxdir \
    theme=mytheme file1 file2 file3 ...
```

The keywords author, title, and version are used in the headings of the Sphinx document. By default, version is 1.0 and the script will try to deduce authors and title from the doconce files file1, file2, etc. that together represent the whole document. Note that none of the individual Doconce files file1, file2, etc. should include the rest as their union makes up the whole document. The default value of dirname is sphinx-rootdir. The theme keyword is used to set the theme for design of HTML output from Sphinx (the default theme is 'default').

With a single-file document in mydoc.do.txt one often just runs

```
Terminal> doconce sphinx_dir mydoc
```

and then an appropriate Sphinx directory sphinx-rootdir is made with relevant files.

The doconce sphinx\_dir command generates a script automake\_sphinx.py for compiling the Sphinx document into an HTML document. One can either run automake\_sphinx.py or perform the steps in the script manually, possibly with necessary modifications. You should at least read the script prior to executing it to have some idea of what is done.

The doconce sphinx\_dir script copies directories named figs or figures over to the Sphinx directory so that figures are accessible in the Sphinx compilation. If figures or movies are located in other directories, automake\_sphinx.py must be edited accordingly. Files, to which there are local links (not http: or file: URLs), must be placed in the \_static subdirectory of the Sphinx directory. The utility doconce sphinxfix\_localURLs is run to check for local links in the Doconce file: for each such link, say dir1/dir2/myfile.txt it replaces the link by \_static/myfile.txt and copies dir1/dir2/myfile.txt to a local \_static directory (in the same directory as the script is run). However, we recommend instead that the writer of the document places files in \_static or

lets a script do it automatically. The user must copy all \_static/\* files to the \_static subdirectory of the Sphinx directory. It may be wise to always put files, to which there are local links in the Doconce document, in a \_static or \_static-name directory and use these local links. Then links do not need to be modified when creating a Sphinx version of the document.

Doconce comes with a collection of HTML themes for Sphinx documents. These are packed out in the Sphinx directory, the <code>conf.py</code> configuration file for Sphinx is edited accordingly, and a script <code>make-themes.sh</code> can make HTML documents with one or more themes. For example, to realize the themes <code>fenics</code> and <code>pyramid</code>, one writes

```
Terminal> ./make-themes.sh fenics pyramid
```

The resulting directories with HTML documents are \_build/html\_fenics and \_build/html\_pyramid, respectively. Without arguments, make-themes.sh makes all available themes (!).

If it is not desirable to use the autogenerated scripts explained above, here is the complete manual procedure of generating a Sphinx document from a file mydoc.do.txt.

Step 1. Translate Doconce into the Sphinx format:

```
Terminal> doconce format sphinx mydoc
```

**Step 2.** Create a Sphinx root directory either manually or by using the interactive sphinx-quickstart program. Here is a scripted version of the steps with the latter:

```
Terminal
mkdir sphinx-rootdir
sphinx-quickstart <<EOF</pre>
sphinx-rootdir
Name of My Sphinx Document
Author
version
version
.rst
index
n
n
n
n
n
у
```

n n y y y EOF

The autogenerated <code>conf.py</code> file may need some edits if you want to specific layout (Sphinx themes) of HTML pages. The <code>doconce sphinx\_dir</code> generator makes an extended <code>conv.py</code> file where, among other things, several useful Sphinx extensions are included.

**Step 3.** Copy the mydoc.rst file to the Sphinx root directory:

```
Terminal> cp mydoc.rst sphinx-rootdir
```

If you have figures in your document, the relative paths to those will be invalid when you work with mydoc.rst in the sphinx-rootdir directory. Either edit mydoc.rst so that figure file paths are correct, or simply copy your figure directories to sphinx-rootdir. Links to local files in mydoc.rst must be modified to links to files in the \_static directory, see comment above.

**Step 4.** Edit the generated index.rst file so that mydoc.rst is included, i.e., add mydoc to the toctree section so that it becomes

```
.. toctree::
   :maxdepth: 2
   mydoc
```

(The spaces before mydoc are important!)

**Step 5.** Generate, for instance, an HTML version of the Sphinx source:

```
make clean # remove old versions
make html
```

Sphinx can generate a range of different formats: standalone HTML, HTML in separate directories with index.html files, a large single HTML file, JSON files, various help files (the qthelp, HTML, and Devhelp projects), epub, LTEX, PDF (via LTEX), pure text, man pages, and Texinfo files.

#### Step 6. View the result:

```
Terminal> firefox _build/html/index.html
```

Note that verbatim code blocks can be typeset in a variety of ways depending the argument that follows bc!: cod gives Python (code-block:: python in Sphinx syntax) and cppcod gives C++, but all such arguments can be customized both for Sphinx and LaTeX output.

#### 2.11 Wiki Formats

There are many different wiki formats, but Doconce only supports three: Googlecode wiki, MediaWiki, and Creole Wiki. These formats are called <code>gwiki</code>, <code>mwiki</code>, and <code>cwiki</code>, respectively. Transformation from Doconce to these formats is done by

```
Terminal> doconce format gwiki mydoc.do.txt
Terminal> doconce format mwiki mydoc.do.txt
Terminal> doconce format cwiki mydoc.do.txt
```

The produced MediaWiki can be tested in the sandbox of wikibooks.org. The format works well with Wikipedia, Wikibooks, and ShoutWiki, but not always well elsewhere (see this example).

Large MediaWiki documents can be made with the Book creator. From the MediaWiki format one can go to other formats with aid of mwlib. This means that one can easily use Doconce to write Wikibooks and publish these in PDF and MediaWiki format, while at the same time, the book can also be published as a standard LATEX book, a Sphinx web document, or a collection of HTML files.

The Googlecode wiki document, mydoc.gwiki, is most conveniently stored in a directory which is a clone of the wiki part of the Googlecode project. This is far easier than copying and pasting the entire text into the wiki editor in a web browser.

When the Doconce file contains figures, each figure filename must in the .gwiki file be replaced by a URL where the figure is available. There are instructions in the file for doing this. Usually, one performs this substitution automatically (see next section).

#### 2.12 Tweaking the Doconce Output

Occasionally, one would like to tweak the output in a certain format from Doconce. One example is figure filenames when transforming Doconce to re-StructuredText. Since Doconce does not know if the .rst file is going to be filtered to LaTeX or HTML, it cannot know if .eps or .png is the most appropriate image filename. The solution is to use a text substitution command or code

with, e.g., sed, perl, python, or scitools subst, to automatically edit the output file from Doconce. It is then wise to run Doconce and the editing commands from a script to automate all steps in going from Doconce to the final format(s). The make.sh files in docs/manual and docs/tutorial constitute comprehensive examples on how such scripts can be made.

#### **2.13 Demos**

The current text is generated from a Doconce format stored in the file

```
docs/tutorial/tutorial.do.txt
```

The file make.sh in the tutorial directory of the Doconce source code contains a demo of how to produce a variety of formats. The source of this tutorial, tutorial.do.txt is the starting point. Running make.sh and studying the various generated files and comparing them with the original tutorial.do.txt file, gives a quick introduction to how Doconce is used in a real case. Here is a sample of how this tutorial looks in different formats.

There is another demo in the docs/manual directory which translates the more comprehensive documentation, manual.do.txt, to various formats. The make.sh script runs a set of translations.

# 3 Installation of Doconce and its Dependencies

#### 3.1 Doconce

Doconce itself is pure Python code hosted at http://code.google.com/p/doconce. Its installation from the Mercurial (hg) source follows the standard procedure:

```
# Doconce
hg clone https://doconce.googlecode.com/hg/ doconce
cd doconce
sudo python setup.py install
cd ..
```

Since Doconce is frequently updated, it is recommended to use the above procedure and whenever a problem occurs, make sure to update to the most recent version:

```
cd doconce
hg pull
hg update
sudo python setup.py install
```

Debian GNU/Linux users can also run

	Terminal
sudo apt-get install doconce	
This installs the latest release and r On Ubuntu one needs to run	not the most updated and bugfixed version.
sudo add-apt-repository ppa:scito sudo apt-get update sudo apt-get install doconce	Terminalols/ppa
3.2 Dependencies	
<b>Preprocessors.</b> If you make use gram must be installed:	of the Preprocess preprocessor, this pro-
svn checkout http://preprocess.go cd preprocess cd doconce sudo python setup.py install cd	oglecode.com/svn/trunk/ preprocess
A much more advanced alternat most conveniently done by pip,	ive to Preprocess is Mako. Its installation is
pip install Mako	Terminal
This command requires pip to be in Ubuntu, the installation is simply do	nstalled. On Debian Linux systems, such as one by
	Terminal

Alternatively, one can install from the pip source code.

sudo apt-get install python-pip

Make can also be installed directly from source: download the tarball, pack it out, go to the directory and run the usual sude python setup.py install.

**Image file handling.** Different output formats require different formats of image files. For example, PostScript or Encapuslated PostScript is required for latex output, while HTML needs JPEG, GIF, or PNG formats. Doconce calls up programs from the ImageMagick suite for converting image files to a proper format if needed. The ImageMagick suite can be installed on all major platforms. On Debian Linux (including Ubuntu) systems one can simply write



Note that the doconce ptex2tex command, which needs no installation beyond Doconce itself, can be used as a simpler alternative to the ptex2tex program.

The *minted* Lagrange T<sub>E</sub>X style is offered by ptex2tex and doconce ptext2tex is popular among many users. This style requires the package Pygments to be installed. On Debian Linux,

Terminal
sudo apt-get install python-pygments
Alternatively, the package can be installed manually:
ng clone ssh://hg@bitbucket.org/birkenfeld/pygments-main pygments cd pygments sudo python setup.py install

If you use the minted style together with ptex2tex, you have to enable it by the -DMINTED command-line argument to ptex2tex. This is not necessary if you run the alternative doconce ptex2tex program.

All use of the minted style requires the -shell-escape command-line argument when running LATEX, i.e., latex -shell-escape or pdflatex -shell-escape.

reStructuredText (reST) Output. The rst output from Doconce allows further transformation to LaTeX, HTML, XML, OpenOffice, and so on, through the docutils package. The installation of the most recent version can be done by

```
svn checkout http://docutils.svn.sourceforge.net/svnroot/docutils/trunk/docutils
cd docutils
sudo python setup.py install
cd ..

To use the OpenOffice suite you will typically on Debian systems install

sudo apt-get install unovonv libreoffice libreoffice-dmaths
```

There is a possibility to create PDF files from reST documents using ReportLab instead of LaTeX. The enabling software is rst2pdf. Either download the tarball or clone the svn repository, go to the rst2pdf directory and run the usual sudo python setup.py install.

Output to sphinx requires of course the Sphinx software, installed by

```
hg clone https://bitbucket.org/birkenfeld/sphinx
cd sphinx
sudo python setup.py install
cd ..
```

**Markdown and Pandoc Output.** The Doconce format pandoc outputs the document in the Pandoc extended Markdown format, which via the pandoc program can be translated to a range of other formats. Installation of Pandoc, written in Haskell, is most easily done by

```
sudo apt-get install pandoc
```

on Debian (Ubuntu) systems.

**Epydoc Output.** When the output format is epydoc one needs that program too, installed by

**Remark.** Several of the packages above installed from source code are also available in Debian-based system through the apt-get install command. However, we recommend installation directly from the version control system repository as there might be important updates and bug fixes. For svn directories, go to the directory, run svn update, and then sudo python setup.py install. For Mercurial (hg) directories, go to the directory, run hg pull; hg update, and then sudo python setup.py install.

# **Doconce: Document Once, Include Anywhere**

Author: Hans Petter Langtangen

Date: Jan 16, 2013

- When writing a note, report, manual, etc., do you find it difficult to choose the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?
- Do you need to write documents in varying formats but find it difficult to remember all the typesetting details of various formats like LaTeX, HTML, reStructuredText, Sphinx, and wiki? Would it be convenient to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?
- Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.

## What Does Doconce Look Like?

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. Here are som examples.

- Bullet lists arise from lines starting with \*.
- Emphasized words are surrounded by \*.
- Words in boldface are surrounded by underscores.
- Words from computer code are enclosed in back quotes and then typeset verbatim (in a monospace font).
- Section headings are recognied by equality (=) signs before and after the title, and the number of = signs indicates the level of the section: 7 for main section, 5 for subsection, and 3 for subsubsection.
- Paragraph headings are recognized by a double underscore before and after the heading.
- The abstract of a document starts with *Abstract* as paragraph heading, and all text up to the next heading makes up the abstract,
- Blocks of computer code can easily be included by placing !bc (begin code) and !ec (end code) commands at separate lines before and after the code block.
- Blocks of computer code can also be imported from source files.
- Blocks of LaTeX mathematics can easily be included by placing !bt (begin TeX) and !et (end TeX) commands at separate lines before and after the math block.

- There is support for both LaTeX and text-like inline mathematics.
- Figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported.
- Invisible comments in the output format can be inserted throughout the text.
- Visible comments can be inserted so that authors and readers can comment upon the text (and at any time turn on/off output of such comments).
- There is an exercise environment with many advanced features.
- With a preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text.
- With Mako one can also have Python code embedded in the Doconce document and thereby parameterize the text (e.g., one text can describe programming in two languages).

Here is an example of some simple text written in the Doconce format:

```
==== A Subsection with Sample Text =====
label{my:first:sec}
```

Ordinary text looks like ordinary text, and the tags used for \_boldface\_ words, \*emphasized\* words, and 'computer' words look natural in plain text. Lists are typeset as you would do in email,

- \* item 1
- \* item 2
- \* item 3

Lists can also have automatically numbered items instead of bullets,

- o item 1
- o item 2
- o item 3

URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl". If the word is URL, the URL itself becomes the link name, as in "URL": "tutorial.do.txt".

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to Section ref{my:first:sec}.

Doconce also allows inline comments of the form [name: comment] (with a space after 'name: '), e.g., such as [hpl: here I will make some remarks to the text]. Inline comments can be removed from the output by a command-line argument (see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

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

# lines beginning with # are comment lines

The Doconce text above results in the following little document:

# A Subsection with Sample Text

Ordinary text looks like ordinary text, and the tags used for **boldface** words, *emphasized* words, and computer words look natural in plain text. Lists are typeset as you would do in an email,

- item 1
- item 2
- item 3

Lists can also have numbered items instead of bullets, just use an  $\circ$  (for ordered) instead of the asterisk:

- 1. item 1
- 2. item 2
- 3. item 3

URLs with a link word are possible, as in hpl. If the word is URL, the URL itself becomes the link name, as in tutorial.do.txt.

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to the section A Subsection with Sample Text.

Doconce also allows inline comments such as (**hpl**: here I will make some remarks to the text) for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see the section From Doconce to Other Formats for an example).

Tables are also supperted, e.g.,

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

#### **Mathematics and Computer Code**

Inline mathematics, such as  $v = \sin(x)$ , allows the formula to be specified both as LaTeX and as plain text. This results in a professional LaTeX typesetting, but in other formats the text version normally looks better than raw LaTeX mathematics with backslashes. An inline formula like  $v = \sin(x)$  is typeset as:

```
\ln = \sin(x)
```

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula. If you write a lot of mathematics, only the output formats latex, pdflatex, html, sphinx, and pandoc are of interest and all these support inline LaTeX mathematics so then you will naturally drop the pipe symbol and write just:

```
\alpha = \sin(x)
```

However, if you want more textual formats, like plain text or reStructuredText, the text after the pipe symbol may help to make the math formula more readable if there are backslahes or other special LaTeX symbols in the LaTeX code.

Blocks of mathematics are typeset with raw LaTeX, inside !bt and !et (begin TeX, end TeX) instructions:

```
!bt
\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f, label{myeq1}\\
{\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
!et
```

The result looks like this:

```
\begin{align} {\partial u ver partial t} &= \nabla^2 u + f, label{myeq1} \\ {\partial v ver partial t} &= \nabla cdot(q(u) \nabla v) + g \\ end{align}
```

Of course, such blocks only looks nice in formats with support for LaTeX mathematics, and here the align environment in particular (this includes latex, pdflatex, html, and sphinx). The raw LaTeX syntax appears in simpler formats, but can still be useful for those who can read LaTeX syntax.

You can have blocks of computer code, starting and ending with !bc and !ec instructions, respectively:

```
!bc pycod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec
```

Such blocks are formatted as:

```
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
```

A code block must come after some plain sentence (at least for successful output to sphinx, rst, and ASCII-close formats), not directly after a section/paragraph heading or a table.

One can also copy computer code directly from files, either the complete file or specified parts. Computer code is then never duplicated in the documentation (important for the principle of avoiding copying information!).

Another document can be included by writing # #include "mynote.do.txt" at the beginning of a line. Doconce documents have extension do.txt. The do part stands for doconce, while the trailing .txt denotes a text document so that editors gives you plain text editing capabilities.

# Macros (Newcommands), Cross-References, Index, and Bibliography

Doconce supports a type of macros via a LaTeX-style <code>newcommand</code> construction. The newcommands defined in a file with name <code>newcommand\_replace.tex</code> are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names <code>newcommands.tex</code> and <code>newcommands\_keep.tex</code> are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by <code>!bt</code> and <code>!et</code> in <code>newcommands\_keep.tex</code> to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used elsewhere throughout the text will usually be placed in <code>newcommands\_replace.tex</code> and expanded by Doconce. The definitions of newcommands in the <code>newcommands\*.tex</code> files <code>must</code> appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the doc/manual/manual.do.txt file (see the demo page for various formats of this document).

## **From Doconce to Other Formats**

Transformation of a Doconce document mydoc.do.txt to various other formats applies the script doconce format:

```
Terminal> doconce format format mydoc.do.txt

or just:

Terminal> doconce format format mydoc
```

#### **Preprocessing**

The preprocess and make programs are used to preprocess the file, and options to preprocess and/or make can be added after the filename. For example:

```
Terminal> doconce format latex mydoc -Dextra_sections -DVAR1=5  # preprotections doconce format latex yourdoc extra_sections=True VAR1=5  # make
```

The variable FORMAT is always defined as the current format when running preprocess or mako. That is, in the last example, FORMAT is defined as latex. Inside the Doconce document one can then perform format specific actions through tests like #if FORMAT == "latex" (for preprocess) or % if FORMAT == "latex": (for mako).

#### Removal of inline comments

The command-line arguments —no-preprocess and —no-make turn off running preprocess and make, respectively.

Inline comments in the text are removed from the output by:

```
Terminal> doconce format latex mydoc --skip_inline_comments
```

One can also remove all such comments from the original Doconce file by running:

```
Terminal> doconce remove_inline_comments mydoc
```

This action is convenient when a Doconce document reaches its final form and comments by different authors should be removed.

#### **HTML**

Making an HTML version of a Doconce file mydoc.do.txt is performed by:

```
Terminal> doconce format html mydoc
```

The resulting file mydoc.html can be loaded into any web browser for viewing. The HTML style can be defined either in the header of the HTML file or in an external CSS file. The latter is enabled by the command-line argument --css=filename. There is a default style with blue headings, and a style with the solarized color palette, specified by the --html-solarized command line argument. If there is no file with name filename in the --css=filename specification, the blue or solarized styles are written to filename and linked from the HTML document. You can provide your own style sheet either by replacing the content inside the style tags or by specifying a CSS file through the --css=filename option.

If the Pygments package (including the pygmentize program) is installed, code blocks are typeset with aid of this package. The command-line argument --no-pygments-html turns off the use of Pygments and makes code blocks appear with plain (pre) HTML

tags. The option —pygments—html—linenos turns on line numbers in Pygments-formatted code blocks.

The HTML file can be embedded in a template if the Doconce document does not have a title (because then there will be no header and footer in the HTML file). The template file must contain valid HTML code and can have three "slots": %(title)s for a title, %(date)s for a date, and %(main)s for the main body of text, i.e., the Doconce document translated to HTML. The title becomes the first heading in the Doconce document, and the date is extracted from the DATE: line, if present. With the template feature one can easily embed the text in the look and feel of a website. The template can be extracted from the source code of a page at the site; just insert %(title)s and %(date)s at appropriate places and replace the main bod of text by %(main)s. Here is an example:

Terminal> doconce format html mydoc --html-template=mytemplate.html

#### **Blogs**

Doconce can be used for writing blogs provided the blog site accepts raw HTML code. Google's Blogger service (blogname.blogspot.com) is particularly well suited since it also allows extensive LaTeX mathematics via MathJax. Write the blog text as a Doconce document without any title, author, and date. Then generate HTML as described above. Copy the text and paste it into the text area in the blog, making sure the input format is HTML. On Google's Blogger service you can use Doconce to generate blogs with LaTeX mathematics and pretty (pygmentized) blocks of computer code. See a blog example for details on blogging.

#### Warning

In the comments after the blog one cannot paste raw HTML code with Math-Jax scripts so there is no support for mathematics in the comments.

WordPress (wordpress.com) allows raw HTML code in blogs, but has very limited LaTeX support, basically only formulas. The --wordpress option to doconce modifies the HTML code such that all equations are typeset in a way that is acceptable to WordPress. There is a doconce example on blogging with mathematics and code on WordPress.

#### Pandoc and Markdown

Output in Pandoc's extended Markdown format results from:

```
Terminal> doconce format pandoc mydoc
```

The name of the output file is mydoc, mkd. From this format one can go to numerous other formats:

```
Terminal> pandoc -R -t mediawiki -o mydoc.mwk --toc mydoc.mkd
```

Pandoc supports latex, html, odt (OpenOffice), docx (Microsoft Word), rtf, texinfo, to mention some. The -R option makes Pandoc pass raw HTML or LaTeX to the output format instead of ignoring it, while the --toc option generates a table of contents. See the Pandoc documentation for the many features of the pandoc program.

Pandoc is useful to go from LaTeX mathematics to, e.g., HTML or MS Word. There are two ways (experiment to find the best one for your document): doconce format pandoc and then translating using pandoc, or doconce format latex, and then going from LaTeX to the desired format using pandoc. Here is an example on the latter strategy:

```
Terminal> doconce format latex mydoc
Terminal> doconce ptex2tex mydoc
Terminal> doconce replace '\Verb!' '\verb!' mydoc.tex
Terminal> pandoc -f latex -t docx -o mydoc.docx mydoc.tex
```

When we go through pandoc, only single equations or align  $\star$  environments are well understood.

Note that Doconce applies the Verb macro from the fancyvrb package while pandoc only supports the standard verb construction for inline verbatim text. Moreover, quite some additional doconce replace and doconce subst edits might be needed on the .mkd or .tex files to successfully have mathematics that is well translated to MS Word. Also when going to reStructuredText using Pandoc, it can be advantageous to go via LaTeX.

Here is an example where we take a Doconce snippet (without title, author, and date), maybe with some unnumbered equations, and quickly generate HTML with mathematics displayed my MathJax:

```
Terminal> doconce format pandoc mydoc
Terminal> pandoc -t html -o mydoc.html -s --mathjax mydoc.mkd
```

The -s option adds a proper header and footer to the mydoc.html file. This recipe is a quick way of makeing HTML notes with (some) mathematics.

#### LaTeX

Making a LaTeX file mydoc.tex from mydoc.do.txt is done in two steps: .. Note: putting code blocks inside a list is not successful in many

Step 1. Filter the doconce text to a pre-LaTeX form mydoc.p.tex for the ptex2tex program (or doconce ptex2tex):

```
Terminal> doconce format latex mydoc
```

LaTeX-specific commands ("newcommands") in math formulas and similar can be placed in files newcommands.tex, newcommands\_keep.tex, or newcommands\_replace.tex (see the section Macros (Newcommands), Cross-References, Index, and Bibliography). If these files are present, they are included in the LaTeX document so that your commands are defined.

An option --latex-printed makes some adjustments for documents aimed at being printed. For example, links to web resources are associated with a footnote listing the complete web address (URL).

Step 2. Run ptex2tex (if you have it) to make a standard LaTeX file:

```
Terminal> ptex2tex mydoc
```

In case you do not have ptex2tex, you may run a (very) simplified version:

```
Terminal> doconce ptex2tex mydoc
```

Note that Doconce generates a .p.tex file with some preprocessor macros that can be used to steer certain properties of the LaTeX document. For example, to turn on the Helvetica font instead of the standard Computer Modern font, run:

```
Terminal> ptex2tex -DHELVETICA mydoc
Terminal> doconce ptex2tex mydoc -DHELVETICA # alternative
```

The title, authors, and date are by default typeset in a non-standard way to enable a nicer treatment of multiple authors having institutions in common. However, the standard LaTeX "maketitle" heading is also available through <code>-DLATEX\_HEADING=traditional</code>. A separate titlepage can be generate by <code>-DLATEX\_HEADING=titlepage</code>.

Preprocessor variables to be defined or undefined are

- BOOK for the "book" documentclass rather than the standard "article" class (necessary if you apply chapter headings)
- PALATINO for the Palatino font
- HELVETIA for the Helvetica font
- A4PAPER for A4 paper size
- A6PAPER for A6 paper size (suitable for reading on small devices)
- MOVIE15 for using the movie15 LaTeX package to display movies
- PREAMBLE to turn the LaTeX preamble on or off (i.e., complete document or document to be included elsewhere)
- MINTED for inclusion of the minted package (which requires latex or pdflatex to be run with the -shell-escape option)

The ptex2tex tool makes it possible to easily switch between many different fancy formattings of computer or verbatim code in LaTeX documents. After any !bc command in the Doconce source you can insert verbatim block styles as defined in your .ptex2tex.cfg file, e.g., !bc sys for a terminal session, where sys is set to a certain environment in .ptex2tex.cfg (e.g., CodeTerminal). There are about 40 styles to choose from, and you can easily add new ones.

Also the doconce ptex2tex command supports preprocessor directives for processing the .p.tex file. The command allows specifications of code environments as well. Here is an example:

Note that @ must be used to separate the begin and end LaTeX commands, unless only the environment name is given (such as minted above, which implies \begin {minted} {fortran} and \end{minted} as begin and end for blocks inside !bc fpro and !ec). Specifying envir=ans:nt means that all other environments are typeset with the anslistings.sty package, e.g., !bc cppcod will then result in \begin{c++}. If no environments like sys, fpro, or the common envir are defined on the command line, the plain \begin{verbatim} and \end{verbatim} used.

Step 2b (optional). Edit the mydoc.tex file to your needs. For example, you may want to substitute section by section\*\* to avoid numbering of sections, you may

want to insert linebreaks (and perhaps space) in the title, etc. This can be automatically edited with the aid of the doconce replace and doconce subst commands. The former works with substituting text directly, while the latter performs substitutions using regular expressions. Here are two examples:

A lot of tailored fixes to the LaTeX document can be done by an appropriate set of text replacements and regular expression substitutions. You are anyway encourged to make a script for generating PDF from the LaTeX file.

Step 3. Compile mydoc.tex and create the PDF file:

```
Terminal> latex mydoc
Terminal> latex mydoc
Terminal> makeindex mydoc  # if index
Terminal> bibitem mydoc  # if bibliography
Terminal> latex mydoc
Terminal> dvipdf mydoc
```

If one wishes to run ptex2tex and use the minted LaTeX package for typesetting code blocks (Minted\_Python, Minted\_Cpp, etc., in ptex2tex specified through the \*pro and \*cod variables in .ptex2tex.cfg or \$HOME/.ptex2tex.cfg), the minted LaTeX package is needed. This package is included by running ptex2tex with the -DMINTED option:

```
Terminal> ptex2tex -DMINTED mydoc
```

In this case, latex must be run with the -shell-escape option:

```
Terminal> latex -shell-escape mydoc
Terminal> latex -shell-escape mydoc
Terminal> makeindex mydoc # if index
Terminal> bibitem mydoc # if bibliography
Terminal> latex -shell-escape mydoc
Terminal> dvipdf mydoc
```

When running doconce ptex2tex mydoc envir=minted (or other minted specifications with doconce ptex2tex), the minted package is automatically included so there is no need for the -DMINTED option.

#### **PDFLaTeX**

Running pdflatex instead of latex follows almost the same steps, but the start is:

```
Terminal> doconce format latex mydoc
```

Then ptex2tex is run as explained above, and finally:

```
Terminal> pdflatex -shell-escape mydoc
Terminal> makeindex mydoc  # if index
Terminal> bibitem mydoc  # if bibliography
Terminal> pdflatex -shell-escape mydoc
```

#### **Plain ASCII Text**

We can go from Doconce "back to" plain untagged text suitable for viewing in terminal windows, inclusion in email text, or for insertion in computer source code:

```
Terminal> doconce format plain mydoc.do.txt # results in mydoc.txt
```

#### reStructuredText

Going from Doconce to reStructuredText gives a lot of possibilities to go to other formats. First we filter the Doconce text to a reStructuredText file mydoc.rst:

```
Terminal> doconce format rst mydoc.do.txt
```

We may now produce various other formats:

```
Terminal> rst2html.py mydoc.rst > mydoc.html # html
Terminal> rst2latex.py mydoc.rst > mydoc.tex # latex
Terminal> rst2xml.py mydoc.rst > mydoc.xml # XML
Terminal> rst2odt.py mydoc.rst > mydoc.odt # OpenOffice
```

The OpenOffice file mydoc.odt can be loaded into OpenOffice and saved in, among other things, the RTF format or the Microsoft Word format. However, it is more convenient to use the program unovonv to convert between the many formats OpenOffice supports on the command line. Run:

```
Terminal> unoconv --show
```

to see all the formats that are supported. For example, the following commands take mydoc.odt to Microsoft Office Open XML format, classic MS Word format, and PDF:

```
Terminal> unoconv -f ooxml mydoc.odt
Terminal> unoconv -f doc mydoc.odt
Terminal> unoconv -f pdf mydoc.odt
```

Remark about Mathematical Typesetting. At the time of this writing, there is no easy way to go from Doconce and LaTeX mathematics to reST and further to OpenOffice and the "MS Word world". Mathematics is only fully supported by latex as output and to a wide extent also supported by the sphinx output format. Some links for going from LaTeX to Word are listed below.

- http://ubuntuforums.org/showthread.php?t=1033441
- http://tug.org/utilities/texconv/textopc.html
- $\bullet \ http://nileshbansal.blogspot.com/2007/12/latex-to-open of ficeword.html$

#### **Sphinx**

Sphinx documents demand quite some steps in their creation. We have automated most of the steps through the doconce sphinx\_dir command:

```
Terminal> doconce sphinx_dir author="authors' names" \
    title="some title" version=1.0 dirname=sphinxdir \
    theme=mytheme file1 file2 file3 ...
```

The keywords author, title, and version are used in the headings of the Sphinx document. By default, version is 1.0 and the script will try to deduce authors and title from the doconce files file1, file2, etc. that together represent the whole document. Note that none of the individual Doconce files file1, file2, etc. should include the rest as their union makes up the whole document. The default value of dirname is sphinx-rootdir. The theme keyword is used to set the theme for design of HTML output from Sphinx (the default theme is 'default').

With a single-file document in mydoc.do.txt one often just runs:

```
Terminal> doconce sphinx_dir mydoc
```

and then an appropriate Sphinx directory sphinx-rootdir is made with relevant files.

The doconce sphinx\_dir command generates a script automake\_sphinx.py for compiling the Sphinx document into an HTML document. One can either run automake\_sphinx.py or perform the steps in the script manually, possibly with necessary modifications. You should at least read the script prior to executing it to have some idea of what is done.

The doconce sphinx\_dir script copies directories named figs or figures over to the Sphinx directory so that figures are accessible in the Sphinx compilation. If figures or movies are located in other directories, automake\_sphinx.py must be edited accordingly. Files, to which there are local links (not http: or file: URLs), must be placed in the \_static subdirectory of the Sphinx directory. The utility doconce sphinxfix\_localURLs is run to check for local links in the Doconce file: for each such link, say dir1/dir2/myfile.txt it replaces the link by \_static/myfile.txt and copies dir1/dir2/myfile.txt to a local \_static directory (in the same directory as the script is run). However, we recommend instead that the writer of the document places files in \_static or lets a script do it automatically. The user must copy all \_static/\* files to the \_static subdirectory of the Sphinx directory. It may be wise to always put files, to which there are local links in the Doconce document, in a \_static or \_static-name directory and use these local links. Then links do not need to be modified when creating a Sphinx version of the document.

Doconce comes with a collection of HTML themes for Sphinx documents. These are packed out in the Sphinx directory, the <code>conf.py</code> configuration file for Sphinx is edited accordingly, and a script <code>make-themes.sh</code> can make HTML documents with one or more themes. For example, to realize the themes <code>fenics</code> and <code>pyramid</code>, one writes:

```
Terminal> ./make-themes.sh fenics pyramid
```

The resulting directories with HTML documents are \_build/html\_fenics and \_build/html\_pyramid, respectively. Without arguments, make-themes.sh makes all available themes (!).

If it is not desirable to use the autogenerated scripts explained above, here is the complete manual procedure of generating a Sphinx document from a file mydoc.do.txt.

Step 1. Translate Doconce into the Sphinx format:

Terminal> doconce format sphinx mydoc

Step 2. Create a Sphinx root directory either manually or by using the interactive sphinx-quickstart program. Here is a scripted version of the steps with the latter:

```
mkdir sphinx-rootdir
sphinx-quickstart <<EOF</pre>
sphinx-rootdir
Name of My Sphinx Document
Author
version
version
.rst
index
n
У
n
n
n
n
У
n
n
У
У
У
EOF
```

The autogenerated <code>conf.py</code> file may need some edits if you want to specific layout (Sphinx themes) of HTML pages. The <code>doconce sphinx\_dir</code> generator makes an extended <code>conv.py</code> file where, among other things, several useful Sphinx extensions are included.

Step 3. Copy the mydoc.rst file to the Sphinx root directory:

```
Terminal> cp mydoc.rst sphinx-rootdir
```

If you have figures in your document, the relative paths to those will be invalid when you work with mydoc.rst in the sphinx-rootdir directory. Either edit mydoc.rst so that figure file paths are correct, or simply copy your figure directories to sphinx-rootdir. Links to local files in mydoc.rst must be modified to links to files in the \_static directory, see comment above.

Step 4. Edit the generated index.rst file so that mydoc.rst is included, i.e., add mydoc to the tootree section so that it becomes:

```
.. toctree::
    :maxdepth: 2

mydoc
```

(The spaces before mydoc are important!) *Step 5.* Generate, for instance, an HTML version of the Sphinx source:

```
make clean  # remove old versions
make html
```

Sphinx can generate a range of different formats: standalone HTML, HTML in separate directories with index.html files, a large single HTML file, JSON files, various help files (the qthelp, HTML, and Devhelp projects), epub, LaTeX, PDF (via LaTeX), pure text, man pages, and Texinfo files.

Step 6. View the result:

```
Terminal> firefox _build/html/index.html
```

Note that verbatim code blocks can be typeset in a variety of ways depending the argument that follows !bc: cod gives Python (code-block:: python in Sphinx syntax) and cppcod gives C++, but all such arguments can be customized both for Sphinx and LaTeX output.

#### **Wiki Formats**

There are many different wiki formats, but Doconce only supports three: Googlecode wiki, MediaWiki, and Creole Wiki. These formats are called gwiki, mwiki, and cwiki, respectively. Transformation from Doconce to these formats is done by:

```
Terminal> doconce format gwiki mydoc.do.txt
Terminal> doconce format mwiki mydoc.do.txt
Terminal> doconce format cwiki mydoc.do.txt
```

The produced MediaWiki can be tested in the sandbox of wikibooks.org. The format works well with Wikipedia, Wikibooks, and ShoutWiki, but not always well elsewhere (see this example).

Large MediaWiki documents can be made with the Book creator. From the MediaWiki format one can go to other formats with aid of mwlib. This means that one can easily use Doconce to write Wikibooks and publish these in PDF and MediaWiki format, while at the same time, the book can also be published as a standard LaTeX book, a Sphinx web document, or a collection of HTML files.

The Googlecode wiki document, mydoc.gwiki, is most conveniently stored in a directory which is a clone of the wiki part of the Googlecode project. This is far easier than copying and pasting the entire text into the wiki editor in a web browser.

When the Doconce file contains figures, each figure filename must in the .gwiki file be replaced by a URL where the figure is available. There are instructions in the file for doing this. Usually, one performs this substitution automatically (see next section).

#### **Tweaking the Doconce Output**

Occasionally, one would like to tweak the output in a certain format from Doconce. One example is figure filenames when transforming Doconce to reStructuredText. Since Doconce does not know if the .rst file is going to be filtered to LaTeX or HTML, it cannot know if .eps or .png is the most appropriate image filename. The solution is to use a text substitution command or code with, e.g., sed, perl, python, or scitools subst, to automatically edit the output file from Doconce. It is then wise to run Doconce

and the editing commands from a script to automate all steps in going from Doconce to the final format(s). The make.sh files in docs/manual and docs/tutorial constitute comprehensive examples on how such scripts can be made.

#### **Demos**

The current text is generated from a Doconce format stored in the file:

```
docs/tutorial/tutorial.do.txt
```

The file make.sh in the tutorial directory of the Doconce source code contains a demo of how to produce a variety of formats. The source of this tutorial, tutorial.do.txt is the starting point. Running make.sh and studying the various generated files and comparing them with the original tutorial.do.txt file, gives a quick introduction to how Doconce is used in a real case. Here is a sample of how this tutorial looks in different formats.

There is another demo in the docs/manual directory which translates the more comprehensive documentation, manual.do.txt, to various formats. The make.sh script runs a set of translations.

#### **Installation of Doconce and its Dependencies**

#### **Doconce**

Doconce itself is pure Python code hosted at http://code.google.com/p/doconce. Its installation from the Mercurial (hg) source follows the standard procedure:

```
# Doconce
hg clone https://doconce.googlecode.com/hg/ doconce
cd doconce
sudo python setup.py install
cd ..
```

Since Doconce is frequently updated, it is recommended to use the above procedure and whenever a problem occurs, make sure to update to the most recent version:

```
cd doconce
hg pull
hg update
sudo python setup.py install
```

Debian GNU/Linux users can also run:

```
sudo apt-get install doconce
```

This installs the latest release and not the most updated and bugfixed version. On Ubuntu one needs to run:

```
sudo add-apt-repository ppa:scitools/ppa
sudo apt-get update
sudo apt-get install doconce
```

#### **Dependencies**

#### **Preprocessors**

If you make use of the Preprocess preprocessor, this program must be installed:

```
svn checkout http://preprocess.googlecode.com/svn/trunk/ preprocess
cd preprocess
cd doconce
sudo python setup.py install
cd ..
```

A much more advanced alternative to Preprocess is Mako. Its installation is most conveniently done by pip:

```
pip install Mako
```

This command requires pip to be installed. On Debian Linux systems, such as Ubuntu, the installation is simply done by:

```
sudo apt-get install python-pip
```

Alternatively, one can install from the pip source code.

Make can also be installed directly from source: download the tarball, pack it out, go to the directory and run the usual sudo python setup.py install.

#### Image file handling

Different output formats require different formats of image files. For example, PostScript or Encapuslated PostScript is required for latex output, while HTML needs JPEG, GIF, or PNG formats. Doconce calls up programs from the ImageMagick suite for converting image files to a proper format if needed. The ImageMagick suite can be installed on all major platforms. On Debian Linux (including Ubuntu) systems one can simply write:

```
sudo apt-get install imagemagick
```

The convenience program doconce combine\_images, for combining several images into one, will use montage and convert from ImageMagick and the pdftk, pdfnup, and pdfcrop programs from the texlive-extra-utils Debian package. The latter gets installed by:

```
sudo apt-get install texlive-extra-utils
```

#### **Spellcheck**

The utility doconce spellcheck applies the ispell program for spellcheck. On Debian (including Ubuntu) it is installed by:

```
sudo apt-get install ispell
```

#### Ptex2tex for LaTeX Output

To make LaTeX documents with very flexible choice of typesetting of verbatim code blocks you need ptex2tex, which is installed by:

```
svn checkout http://ptex2tex.googlecode.com/svn/trunk/ ptex2tex
cd ptex2tex
sudo python setup.py install
```

It may happen that you need additional style files, you can run a script, cp2texmf.sh:

```
cd latex
sh cp2texmf.sh # copy stylefiles to ~/texmf directory
cd ../..
```

This script copies some special stylefiles that that ptex2tex potentially makes use of. Some more standard stylefiles are also needed. These are installed by:

```
sudo apt-get install texlive-latex-recommended texlive-latex-extra
```

on Debian Linux (including Ubuntu) systems. TeXShop on Mac comes with the necessary stylefiles (if not, they can be found by googling and installed manually in the ~/texmf/tex/latex/misc directory).

Note that the doconce ptex2tex command, which needs no installation beyond Doconce itself, can be used as a simpler alternative to the ptex2tex program.

The *minted* LaTeX style is offered by ptex2tex and doconce ptext2tex is popular among many users. This style requires the package Pygments to be installed. On Debian Linux:

```
sudo apt-get install python-pygments
```

Alternatively, the package can be installed manually:

```
hg clone ssh://hg@bitbucket.org/birkenfeld/pygments-main pygments
cd pygments
sudo python setup.py install
```

If you use the minted style together with ptex2tex, you have to enable it by the -DMINTED command-line argument to ptex2tex. This is not necessary if you run the alternative doconce ptex2tex program.

All use of the minted style requires the -shell-escape command-line argument when running LaTeX, i.e., latex -shell-escape or pdflatex -shell-escape.

#### reStructuredText (reST) Output

The rst output from Doconce allows further transformation to LaTeX, HTML, XML, OpenOffice, and so on, through the docutils package. The installation of the most recent version can be done by:

```
svn checkout http://docutils.svn.sourceforge.net/svnroot/docutils/trunk/doc
cd docutils
sudo python setup.py install
cd ..
```

To use the OpenOffice suite you will typically on Debian systems install:

```
sudo apt-get install unovonv libreoffice libreoffice-dmaths
```

There is a possibility to create PDF files from reST documents using ReportLab instead of LaTeX. The enabling software is rst2pdf. Either download the tarball or clone the svn repository, go to the rst2pdf directory and run the usual sudo python setup.py install.

Output to sphinx requires of course the Sphinx software, installed by:

```
hg clone https://bitbucket.org/birkenfeld/sphinx
cd sphinx
sudo python setup.py install
cd ..
```

#### Markdown and Pandoc Output

The Doconce format pandoc outputs the document in the Pandoc extended Markdown format, which via the pandoc program can be translated to a range of other formats. Installation of Pandoc, written in Haskell, is most easily done by:

```
sudo apt-get install pandoc
on Debian (Ubuntu) systems.
```

#### **Epydoc Output**

When the output format is epydoc one needs that program too, installed by:

```
svn co https://epydoc.svn.sourceforge.net/svnroot/epydoc/trunk/epydoc epydoc
cd epydoc
sudo make install
cd ..
```

Remark. Several of the packages above installed from source code are also available in Debian-based system through the apt-get install command. However, we recommend installation directly from the version control system repository as there might be important updates and bug fixes. For svn directories, go to the directory, run svn update, and then sudo python setup.py install. For Mercurial (hg) directories, go to the directory, run hg pull; hg update, and then sudo python setup.py install.

# Doconce: Document Once, Include Anywhere Documentation

Release 1.0

**Hans Petter Langtangen** 

# **CONTENTS**

1 Doconce: Document Once, Include Anywhere							
2	What	Vhat Does Doconce Look Like?					
	2.1	A Subsection with Sample Text					
	2.2	Mathematics and Computer Code					
	2.3	Macros (Newcommands), Cross-References, Index, and Bibliography					
3	From	Doconce to Other Formats					
	3.1	Preprocessing					
	3.2	Removal of inline comments					
	3.3	HTML					
	3.4	Blogs					
	3.5	Pandoc and Markdown					
	3.6	LaTeX					
	3.7	PDFLaTeX					
	3.8	Plain ASCII Text					
	3.9	reStructuredText					
	3.10	Sphinx					
	3.11	Wiki Formats					
	3.12	Tweaking the Doconce Output					
	3.13	Demos					
1	Incta	llation of Doconce and its Dependencies					
•	4.1	Doconce					
	4.2	Dependencies					
	7.2	Dependencies					
-	Indic	es and tables					

Contents:

CONTENTS 1

2 CONTENTS

# DOCONCE: DOCUMENT ONCE, INCLUDE ANYWHERE

Author Hans Petter Langtangen

Date Jan 16, 2013

- When writing a note, report, manual, etc., do you find it difficult to choose the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?
- Do you need to write documents in varying formats but find it difficult to remember all the typesetting details of various formats like LaTeX, HTML, reStructuredText, Sphinx, and wiki? Would it be convenient to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?
- Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.



# WHAT DOES DOCONCE LOOK LIKE?

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. Here are som examples.

- Bullet lists arise from lines starting with \*.
- Emphasized words are surrounded by \*.
- Words in boldface are surrounded by underscores.
- Words from computer code are enclosed in back quotes and then typeset verbatim (in a monospace font).
- Section headings are recognied by equality (=) signs before and after the title, and the number of = signs indicates the level of the section: 7 for main section, 5 for subsection, and 3 for subsubsection.
- Paragraph headings are recognized by a double underscore before and after the heading.
- The abstract of a document starts with *Abstract* as paragraph heading, and all text up to the next heading makes up the abstract,
- Blocks of computer code can easily be included by placing !bc (begin code) and !ec (end code) commands at separate lines before and after the code block.
- Blocks of computer code can also be imported from source files.
- Blocks of LaTeX mathematics can easily be included by placing !bt (begin TeX) and !et (end TeX) commands at separate lines before and after the math block.
- There is support for both LaTeX and text-like inline mathematics.
- Figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported.
- Invisible comments in the output format can be inserted throughout the text.
- Visible comments can be inserted so that authors and readers can comment upon the text (and at any time turn on/off output of such comments).
- There is an exercise environment with many advanced features.
- With a preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text.
- With Mako one can also have Python code embedded in the Doconce document and thereby parameterize the text (e.g., one text can describe programming in two languages).

Here is an example of some simple text written in the Doconce format:

```
==== A Subsection with Sample Text =====
label{my:first:sec}
Ordinary text looks like ordinary text, and the tags used for
_boldface_ words, *emphasized* words, and 'computer' words look
natural in plain text. Lists are typeset as you would do in email,
 * item 1
  * item 2
  * item 3
Lists can also have automatically numbered items instead of bullets,
 o item 1
 o item 2
 o item 3
URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl".
If the word is URL, the URL itself becomes the link name,
as in "URL": "tutorial.do.txt".
References to sections may use logical names as labels (e.g., a
"label" command right after the section title), as in the reference to
Section ref{my:first:sec}.
Doconce also allows inline comments of the form [name: comment] (with
a space after 'name:'), e.g., such as [hpl: here I will make some
remarks to the text]. Inline comments can be removed from the output
by a command-line argument (see Section ref{doconce2formats} for an
example).
Tables are also supperted, e.g.,
  |time | velocity | acceleration |
  | 0.0 | 1.4186 | -5.01
  | 2.0 | 1.376512 | 11.919
  | 4.0 | 1.1E+1 | 14.717624
```

# lines beginning with # are comment lines

The Doconce text above results in the following little document:

### 2.1 A Subsection with Sample Text

Ordinary text looks like ordinary text, and the tags used for **boldface** words, *emphasized* words, and computer words look natural in plain text. Lists are typeset as you would do in an email,

- item 1
- item 2
- item 3

Lists can also have numbered items instead of bullets, just use an o (for ordered) instead of the asterisk:

- 1. item 1
- 2. item 2
- 3. item 3

URLs with a link word are possible, as in hpl. If the word is URL, the URL itself becomes the link name, as in tutorial.do.txt.

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to the section A Subsection with Sample Text.

Doconce also allows inline comments such as (**hpl**: here I will make some remarks to the text) for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see the section *From Doconce to Other Formats* for an example).

Tables are also supperted, e.g.,

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

## 2.2 Mathematics and Computer Code

Inline mathematics, such as  $\nu = \sin(x)$ , allows the formula to be specified both as LaTeX and as plain text. This results in a professional LaTeX typesetting, but in other formats the text version normally looks better than raw LaTeX mathematics with backslashes. An inline formula like  $\nu = \sin(x)$  is typeset as

```
\alpha = \sin(x) = \sin(x)
```

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula. If you write a lot of mathematics, only the output formats latex, pdflatex, html, sphinx, and pandoc are of interest and all these support inline LaTeX mathematics so then you will naturally drop the pipe symbol and write just

```
\alpha = \sin(x)
```

However, if you want more textual formats, like plain text or reStructuredText, the text after the pipe symbol may help to make the math formula more readable if there are backslahes or other special LaTeX symbols in the LaTeX code.

Blocks of mathematics are typeset with raw LaTeX, inside !bt and !et (begin TeX, end TeX) instructions:

```
!bt
\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f, label{myeq1}\\
{\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
!et
```

The result looks like this:

$$\frac{\partial u}{\partial t} = \nabla^2 u + f,\tag{2.1}$$

$$\frac{\partial v}{\partial t} = \nabla \cdot (q(u)\nabla v) + g$$

Of course, such blocks only looks nice in formats with support for LaTeX mathematics, and here the align environment in particular (this includes latex, pdflatex, html, and sphinx). The raw LaTeX syntax appears in simpler formats, but can still be useful for those who can read LaTeX syntax.

You can have blocks of computer code, starting and ending with !bc and !ec instructions, respectively.

```
!bc pycod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec

Such blocks are formatted as
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
```

A code block must come after some plain sentence (at least for successful output to sphinx, rst, and ASCII-close formats), not directly after a section/paragraph heading or a table.

One can also copy computer code directly from files, either the complete file or specified parts. Computer code is then never duplicated in the documentation (important for the principle of avoiding copying information!).

Another document can be included by writing # #include "mynote.do.txt" at the beginning of a line. Doconce documents have extension do.txt. The do part stands for doconce, while the trailing .txt denotes a text document so that editors gives you plain text editing capabilities.

# 2.3 Macros (Newcommands), Cross-References, Index, and Bibliography

Doconce supports a type of macros via a LaTeX-style newcommand construction. The newcommands defined in a file with name newcommand\_replace.tex are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names newcommands.tex and newcommands\_keep.tex are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by !bt and !et in newcommands\_keep.tex to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used elsewhere throughout the text will usually be placed in newcommands\_replace.tex and expanded by Doconce. The definitions of newcommands in the newcommands\*.tex files must appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the doc/manual/manual.do.txt file (see the demo page for various formats of this document).

# FROM DOCONCE TO OTHER FORMATS

Transformation of a Doconce document mydoc.do.txt to various other formats applies the script doconce format:

```
Terminal> doconce format format mydoc.do.txt or just
```

Terminal> doconce format format mydoc

### 3.1 Preprocessing

The preprocess and make programs are used to preprocess the file, and options to preprocess and/or make can be added after the filename. For example,

```
Terminal> doconce format latex mydoc -Dextra_sections -DVAR1=5  # preprocess Terminal> doconce format latex yourdoc extra_sections=True VAR1=5  # mako
```

The variable FORMAT is always defined as the current format when running preprocess or make. That is, in the last example, FORMAT is defined as latex. Inside the Doconce document one can then perform format specific actions through tests like #if FORMAT == "latex" (for preprocess) or % if FORMAT == "latex": (for make).

#### 3.2 Removal of inline comments

The command-line arguments —no-preprocess and —no-make turn off running preprocess and make, respectively.

Inline comments in the text are removed from the output by

```
Terminal> doconce format latex mydoc --skip_inline_comments
```

One can also remove all such comments from the original Doconce file by running:

```
Terminal> doconce remove_inline_comments mydoc
```

This action is convenient when a Doconce document reaches its final form and comments by different authors should be removed.

#### **3.3 HTML**

Making an HTML version of a Doconce file mydoc.do.txt is performed by

Terminal> doconce format html mydoc

The resulting file mydoc.html can be loaded into any web browser for viewing.

The HTML style can be defined either in the header of the HTML file or in an external CSS file. The latter is enabled by the command-line argument --css=filename. There is a default style with blue headings, and a style with the solarized color palette, specified by the --html-solarized command line argument. If there is no file with name filename in the --css=filename specification, the blue or solarized styles are written to filename and linked from the HTML document. You can provide your own style sheet either by replacing the content inside the style tags or by specifying a CSS file through the --css=filename option.

If the Pygments package (including the pygmentize program) is installed, code blocks are typeset with aid of this package. The command-line argument --no-pygments-html turns off the use of Pygments and makes code blocks appear with plain (pre) HTML tags. The option --pygments-html-linenos turns on line numbers in Pygments-formatted code blocks.

The HTML file can be embedded in a template if the Doconce document does not have a title (because then there will be no header and footer in the HTML file). The template file must contain valid HTML code and can have three "slots": % (title) s for a title, % (date) s for a date, and % (main) s for the main body of text, i.e., the Doconce document translated to HTML. The title becomes the first heading in the Doconce document, and the date is extracted from the DATE: line, if present. With the template feature one can easily embed the text in the look and feel of a website. The template can be extracted from the source code of a page at the site; just insert % (title) s and % (date) s at appropriate places and replace the main bod of text by % (main) s. Here is an example:

Terminal> doconce format html mydoc --html-template=mytemplate.html

### 3.4 Blogs

Doconce can be used for writing blogs provided the blog site accepts raw HTML code. Google's Blogger service (blogname.blogspot.com) is particularly well suited since it also allows extensive LaTeX mathematics via MathJax. Write the blog text as a Doconce document without any title, author, and date. Then generate HTML as described above. Copy the text and paste it into the text area in the blog, making sure the input format is HTML. On Google's Blogger service you can use Doconce to generate blogs with LaTeX mathematics and pretty (pygmentized) blocks of computer code. See a blog example for details on blogging.

**Warning:** In the comments after the blog one cannot paste raw HTML code with MathJax scripts so there is no support for mathematics in the comments.

WordPress (wordpress.com) allows raw HTML code in blogs, but has very limited LaTeX support, basically only formulas. The --wordpress option to doconce modifies the HTML code such that all equations are typeset in a way that is acceptable to WordPress. There is a doconce example on blogging with mathematics and code on WordPress.

#### 3.5 Pandoc and Markdown

Output in Pandoc's extended Markdown format results from

```
Terminal> doconce format pandoc mydoc
```

The name of the output file is mydoc.mkd. From this format one can go to numerous other formats:

```
Terminal> pandoc -R -t mediawiki -o mydoc.mwk --toc mydoc.mkd
```

Pandoc supports latex, html, odt (OpenOffice), docx (Microsoft Word), rtf, texinfo, to mention some. The -R option makes Pandoc pass raw HTML or LaTeX to the output format instead of ignoring it, while the --toc option generates a table of contents. See the Pandoc documentation for the many features of the pandoc program.

Pandoc is useful to go from LaTeX mathematics to, e.g., HTML or MS Word. There are two ways (experiment to find the best one for your document): doconce format pandoc and then translating using pandoc, or doconce format latex, and then going from LaTeX to the desired format using pandoc. Here is an example on the latter strategy:

```
Terminal> doconce format latex mydoc
Terminal> doconce ptex2tex mydoc
Terminal> doconce replace '\Verb!' '\verb!' mydoc.tex
Terminal> pandoc -f latex -t docx -o mydoc.docx mydoc.tex
```

When we go through pandoc, only single equations or align\* environments are well understood.

Note that Doconce applies the Verb macro from the fancyvrb package while pandoc only supports the standard verb construction for inline verbatim text. Moreover, quite some additional doconce replace and doconce subst edits might be needed on the .mkd or .tex files to successfully have mathematics that is well translated to MS Word. Also when going to reStructuredText using Pandoc, it can be advantageous to go via LaTeX.

Here is an example where we take a Doconce snippet (without title, author, and date), maybe with some unnumbered equations, and quickly generate HTML with mathematics displayed my MathJax:

```
Terminal> doconce format pandoc mydoc
Terminal> pandoc -t html -o mydoc.html -s --mathjax mydoc.mkd
```

The -s option adds a proper header and footer to the mydoc.html file. This recipe is a quick way of makeing HTML notes with (some) mathematics.

#### 3.6 LaTeX

Making a LaTeX file mydoc.tex from mydoc.do.txt is done in two steps: .. Note: putting code blocks inside a list is not successful in many

Step 1. Filter the doconce text to a pre-LaTeX form mydoc.p.tex for the ptex2tex program (or doconce ptex2tex):

```
Terminal> doconce format latex mydoc
```

LaTeX-specific commands ("newcommands") in math formulas and similar can be placed in files newcommands.tex, newcommands\_keep.tex, or newcommands\_replace.tex (see the section *Macros* (*Newcommands*), *Cross-References*, *Index*, *and Bibliography*). If these files are present, they are included in the LaTeX document so that your commands are defined.

An option ——latex—printed makes some adjustments for documents aimed at being printed. For example, links to web resources are associated with a footnote listing the complete web address (URL).

Step 2. Run ptex2tex (if you have it) to make a standard LaTeX file,

```
Terminal> ptex2tex mydoc
```

3.6. LaTeX 11

In case you do not have ptex2tex, you may run a (very) simplified version:

```
Terminal> doconce ptex2tex mydoc
```

Note that Doconce generates a .p.tex file with some preprocessor macros that can be used to steer certain properties of the LaTeX document. For example, to turn on the Helvetica font instead of the standard Computer Modern font, run

```
Terminal> ptex2tex -DHELVETICA mydoc
Terminal> doconce ptex2tex mydoc -DHELVETICA # alternative
```

The title, authors, and date are by default typeset in a non-standard way to enable a nicer treatment of multiple authors having institutions in common. However, the standard LaTeX "maketitle" heading is also available through <code>-DLATEX\_HEADING=traditional</code>. A separate titlepage can be generate by <code>-DLATEX\_HEADING=titlepage</code>.

Preprocessor variables to be defined or undefined are

- BOOK for the "book" documentclass rather than the standard "article" class (necessary if you apply chapter headings)
- PALATINO for the Palatino font
- HELVETIA for the Helvetica font
- A4PAPER for A4 paper size
- A6PAPER for A6 paper size (suitable for reading on small devices)
- MOVIE15 for using the movie15 LaTeX package to display movies
- PREAMBLE to turn the LaTeX preamble on or off (i.e., complete document or document to be included elsewhere)
- MINTED for inclusion of the minted package (which requires latex or pdflatex to be run with the -shell-escape option)

The ptex2tex tool makes it possible to easily switch between many different fancy formattings of computer or verbatim code in LaTeX documents. After any !bc command in the Doconce source you can insert verbatim block styles as defined in your .ptex2tex.cfg file, e.g., !bc sys for a terminal session, where sys is set to a certain environment in .ptex2tex.cfg (e.g., CodeTerminal). There are about 40 styles to choose from, and you can easily add new ones.

Also the doconce ptex2tex command supports preprocessor directives for processing the .p.tex file. The command allows specifications of code environments as well. Here is an example:

```
Terminal> doconce ptex2tex mydoc -DLATEX_HEADING=traditional \
    -DPALATINO -DA6PAPER \
    "sys=\begin{quote}\begin{verbatim}@\end{verbatim}\end{quote}" \
    fpro=minted fcod=minted shcod=Verbatim envir=ans:nt
```

Note that @ must be used to separate the begin and end LaTeX commands, unless only the environment name is given (such as minted above, which implies \begin{minted} {fortran} and \end{minted} as begin and end for blocks inside !bc fpro and !ec). Specifying envir=ans:nt means that all other environments are typeset with the anslistings.sty package, e.g., !bc cppcod will then result in \begin{c++}. If no environments like sys, fpro, or the common envir are defined on the command line, the plain \begin{verbatim} and \end{verbatim} used.

Step 2b (optional). Edit the mydoc.tex file to your needs. For example, you may want to substitute section by section\* to avoid numbering of sections, you may want to insert linebreaks (and perhaps space) in the title, etc. This can be automatically edited with the aid of the doconce replace and doconce subst commands. The

former works with substituting text directly, while the latter performs substitutions using regular expressions. Here are two examples:

A lot of tailored fixes to the LaTeX document can be done by an appropriate set of text replacements and regular expression substitutions. You are anyway encourged to make a script for generating PDF from the LaTeX file.

Step 3. Compile mydoc.tex and create the PDF file:

```
Terminal> latex mydoc
Terminal> latex mydoc
Terminal> makeindex mydoc  # if index
Terminal> bibitem mydoc  # if bibliography
Terminal> latex mydoc
Terminal> dvipdf mydoc
```

If one wishes to run ptex2tex and use the minted LaTeX package for typesetting code blocks (Minted\_Python, Minted\_Cpp, etc., in ptex2tex specified through the \*pro and \*cod variables in .ptex2tex.cfg or \$HOME/.ptex2tex.cfg), the minted LaTeX package is needed. This package is included by running ptex2tex with the -DMINTED option:

```
Terminal> ptex2tex -DMINTED mydoc
```

In this case, latex must be run with the -shell-escape option:

```
Terminal> latex -shell-escape mydoc
Terminal> latex -shell-escape mydoc
Terminal> makeindex mydoc # if index
Terminal> bibitem mydoc # if bibliography
Terminal> latex -shell-escape mydoc
Terminal> dvipdf mydoc
```

When running doconce ptex2tex mydoc envir=minted (or other minted specifications with doconce ptex2tex), the minted package is automatically included so there is no need for the -DMINTED option.

#### 3.7 PDFLaTeX

Running pdflatex instead of latex follows almost the same steps, but the start is

```
Terminal> doconce format latex mydoc
```

Then ptex2tex is run as explained above, and finally

```
Terminal> pdflatex -shell-escape mydoc
Terminal> makeindex mydoc  # if index
Terminal> bibitem mydoc  # if bibliography
Terminal> pdflatex -shell-escape mydoc
```

#### 3.8 Plain ASCII Text

We can go from Doconce "back to" plain untagged text suitable for viewing in terminal windows, inclusion in email text, or for insertion in computer source code:

3.7. PDFLaTeX

```
Terminal> doconce format plain mydoc.do.txt # results in mydoc.txt
```

#### 3.9 reStructuredText

Going from Doconce to reStructuredText gives a lot of possibilities to go to other formats. First we filter the Doconce text to a reStructuredText file mydoc.rst:

```
Terminal> doconce format rst mydoc.do.txt
```

We may now produce various other formats:

```
Terminal> rst2html.py mydoc.rst > mydoc.html # html
Terminal> rst2latex.py mydoc.rst > mydoc.tex # latex
Terminal> rst2xml.py mydoc.rst > mydoc.xml # XML
Terminal> rst2odt.py mydoc.rst > mydoc.odt # OpenOffice
```

The OpenOffice file mydoc.odt can be loaded into OpenOffice and saved in, among other things, the RTF format or the Microsoft Word format. However, it is more convenient to use the program unovonv to convert between the many formats OpenOffice supports on the command line. Run

```
Terminal> unoconv --show
```

to see all the formats that are supported. For example, the following commands take mydoc.odt to Microsoft Office Open XML format, classic MS Word format, and PDF:

```
Terminal> unoconv -f ooxml mydoc.odt
Terminal> unoconv -f doc mydoc.odt
Terminal> unoconv -f pdf mydoc.odt
```

Remark about Mathematical Typesetting. At the time of this writing, there is no easy way to go from Doconce and LaTeX mathematics to reST and further to OpenOffice and the "MS Word world". Mathematics is only fully supported by latex as output and to a wide extent also supported by the sphinx output format. Some links for going from LaTeX to Word are listed below.

- http://ubuntuforums.org/showthread.php?t=1033441
- http://tug.org/utilities/texconv/textopc.html
- http://nileshbansal.blogspot.com/2007/12/latex-to-openofficeword.html

# 3.10 Sphinx

Sphinx documents demand quite some steps in their creation. We have automated most of the steps through the doconce sphinx dir command:

The keywords author, title, and version are used in the headings of the Sphinx document. By default, version is 1.0 and the script will try to deduce authors and title from the doconce files file1, file2, etc. that together represent the whole document. Note that none of the individual Doconce files file1, file2, etc. should include the rest as their union makes up the whole document. The default value of dirname is sphinx-rootdir. The theme keyword is used to set the theme for design of HTML output from Sphinx (the default theme is 'default').

With a single-file document in mydoc.do.txt one often just runs

```
Terminal> doconce sphinx_dir mydoc
```

and then an appropriate Sphinx directory sphinx-rootdir is made with relevant files.

The doconce sphinx\_dir command generates a script automake\_sphinx.py for compiling the Sphinx document into an HTML document. One can either run automake\_sphinx.py or perform the steps in the script manually, possibly with necessary modifications. You should at least read the script prior to executing it to have some idea of what is done.

The doconce sphinx\_dir script copies directories named figs or figures over to the Sphinx directory so that figures are accessible in the Sphinx compilation. If figures or movies are located in other directories, automake\_sphinx.py must be edited accordingly. Files, to which there are local links (not http: or file: URLs), must be placed in the \_static subdirectory of the Sphinx directory. The utility doconce sphinxfix\_localURLs is run to check for local links in the Doconce file: for each such link, say dir1/dir2/myfile.txt it replaces the link by \_static/myfile.txt and copies dir1/dir2/myfile.txt to a local \_static directory (in the same directory as the script is run). However, we recommend instead that the writer of the document places files in \_static or lets a script do it automatically. The user must copy all \_static/\* files to the \_static subdirectory of the Sphinx directory. It may be wise to always put files, to which there are local links in the Doconce document, in a \_static or \_static-name directory and use these local links. Then links do not need to be modified when creating a Sphinx version of the document.

Doconce comes with a collection of HTML themes for Sphinx documents. These are packed out in the Sphinx directory, the <code>conf.py</code> configuration file for Sphinx is edited accordingly, and a script <code>make-themes.sh</code> can make HTML documents with one or more themes. For example, to realize the themes <code>fenics</code> and <code>pyramid</code>, one writes

```
Terminal> ./make-themes.sh fenics pyramid
```

The resulting directories with HTML documents are \_build/html\_fenics and \_build/html\_pyramid, respectively. Without arguments, make-themes.sh makes all available themes (!).

If it is not desirable to use the autogenerated scripts explained above, here is the complete manual procedure of generating a Sphinx document from a file mydoc.do.txt.

Step 1. Translate Doconce into the Sphinx format:

```
Terminal> doconce format sphinx mydoc
```

Step 2. Create a Sphinx root directory either manually or by using the interactive sphinx-quickstart program. Here is a scripted version of the steps with the latter:

```
mkdir sphinx-rootdir
sphinx-quickstart <<EOF
sphinx-rootdir
n
_
Name of My Sphinx Document
Author
version
version
.rst
index
n
y
n
n
n</pre>
```

3.10. Sphinx 15

```
y
n
n
y
y
y
EOF
```

The autogenerated <code>conf.py</code> file may need some edits if you want to specific layout (Sphinx themes) of HTML pages. The <code>doconce sphinx\_dir</code> generator makes an extended <code>conv.py</code> file where, among other things, several useful Sphinx extensions are included.

*Step 3.* Copy the mydoc.rst file to the Sphinx root directory:

```
Terminal > cp mydoc.rst sphinx-rootdir
```

If you have figures in your document, the relative paths to those will be invalid when you work with mydoc.rst in the sphinx-rootdir directory. Either edit mydoc.rst so that figure file paths are correct, or simply copy your figure directories to sphinx-rootdir. Links to local files in mydoc.rst must be modified to links to files in the \_static directory, see comment above.

Step 4. Edit the generated index.rst file so that mydoc.rst is included, i.e., add mydoc to the toctree section so that it becomes

```
.. toctree::
:maxdepth: 2
mydoc
```

(The spaces before mydoc are important!)

Step 5. Generate, for instance, an HTML version of the Sphinx source:

```
make clean # remove old versions
make html
```

Sphinx can generate a range of different formats: standalone HTML, HTML in separate directories with index.html files, a large single HTML file, JSON files, various help files (the qthelp, HTML, and Devhelp projects), epub, LaTeX, PDF (via LaTeX), pure text, man pages, and Texinfo files.

Step 6. View the result:

```
Terminal> firefox _build/html/index.html
```

Note that verbatim code blocks can be typeset in a variety of ways depending the argument that follows !bc: cod gives Python (code-block:: python in Sphinx syntax) and cppcod gives C++, but all such arguments can be customized both for Sphinx and LaTeX output.

#### 3.11 Wiki Formats

There are many different wiki formats, but Doconce only supports three: Googlecode wiki, MediaWiki, and Creole Wiki. These formats are called gwiki, mwiki, and cwiki, respectively. Transformation from Doconce to these formats is done by

```
Terminal> doconce format gwiki mydoc.do.txt
Terminal> doconce format mwiki mydoc.do.txt
Terminal> doconce format cwiki mydoc.do.txt
```

The produced MediaWiki can be tested in the sandbox of wikibooks.org. The format works well with Wikipedia, Wikibooks, and ShoutWiki, but not always well elsewhere (see this example).

Large MediaWiki documents can be made with the Book creator. From the MediaWiki format one can go to other formats with aid of mwlib. This means that one can easily use Doconce to write Wikibooks and publish these in PDF and MediaWiki format, while at the same time, the book can also be published as a standard LaTeX book, a Sphinx web document, or a collection of HTML files.

The Googlecode wiki document, mydoc.gwiki, is most conveniently stored in a directory which is a clone of the wiki part of the Googlecode project. This is far easier than copying and pasting the entire text into the wiki editor in a web browser.

When the Doconce file contains figures, each figure filename must in the .gwiki file be replaced by a URL where the figure is available. There are instructions in the file for doing this. Usually, one performs this substitution automatically (see next section).

#### 3.12 Tweaking the Doconce Output

Occasionally, one would like to tweak the output in a certain format from Doconce. One example is figure filenames when transforming Doconce to reStructuredText. Since Doconce does not know if the .rst file is going to be filtered to LaTeX or HTML, it cannot know if .eps or .png is the most appropriate image filename. The solution is to use a text substitution command or code with, e.g., sed, perl, python, or scitools subst, to automatically edit the output file from Doconce. It is then wise to run Doconce and the editing commands from a script to automate all steps in going from Doconce to the final format(s). The make.sh files in docs/manual and docs/tutorial constitute comprehensive examples on how such scripts can be made.

#### **3.13 Demos**

The current text is generated from a Doconce format stored in the file

docs/tutorial/tutorial.do.txt

The file make.sh in the tutorial directory of the Doconce source code contains a demo of how to produce a variety of formats. The source of this tutorial, tutorial.do.txt is the starting point. Running make.sh and studying the various generated files and comparing them with the original tutorial.do.txt file, gives a quick introduction to how Doconce is used in a real case. Here is a sample of how this tutorial looks in different formats.

There is another demo in the docs/manual directory which translates the more comprehensive documentation, manual.do.txt, to various formats. The make.sh script runs a set of translations.

Doconce: Document Once, Include Anywhere Documentation, Release 1.0						

**CHAPTER** 

**FOUR** 

# INSTALLATION OF DOCONCE AND ITS DEPENDENCIES

#### 4.1 Doconce

Doconce itself is pure Python code hosted at http://code.google.com/p/doconce. Its installation from the Mercurial (hg) source follows the standard procedure:

```
# Doconce
hg clone https://doconce.googlecode.com/hg/ doconce
cd doconce
sudo python setup.py install
cd ..
```

Since Doconce is frequently updated, it is recommended to use the above procedure and whenever a problem occurs, make sure to update to the most recent version:

```
cd doconce
hg pull
hg update
sudo python setup.py install
```

#### Debian GNU/Linux users can also run

```
sudo apt-get install doconce
```

This installs the latest release and not the most updated and bugfixed version. On Ubuntu one needs to run

```
sudo add-apt-repository ppa:scitools/ppa
sudo apt-get update
sudo apt-get install doconce
```

## 4.2 Dependencies

#### 4.2.1 Preprocessors

If you make use of the Preprocess preprocessor, this program must be installed:

```
svn checkout http://preprocess.googlecode.com/svn/trunk/ preprocess
cd preprocess
cd doconce
```

```
sudo python setup.py install
cd ..
```

A much more advanced alternative to Preprocess is Mako. Its installation is most conveniently done by pip,

```
pip install Mako
```

This command requires pip to be installed. On Debian Linux systems, such as Ubuntu, the installation is simply done by

```
sudo apt-get install python-pip
```

Alternatively, one can install from the pip source code.

Mako can also be installed directly from source: download the tarball, pack it out, go to the directory and run the usual sudo python setup.py install.

#### 4.2.2 Image file handling

Different output formats require different formats of image files. For example, PostScript or Encapuslated PostScript is required for latex output, while HTML needs JPEG, GIF, or PNG formats. Doconce calls up programs from the ImageMagick suite for converting image files to a proper format if needed. The ImageMagick suite can be installed on all major platforms. On Debian Linux (including Ubuntu) systems one can simply write

```
sudo apt-get install imagemagick
```

The convenience program doconce combine\_images, for combining several images into one, will use montage and convert from ImageMagick and the pdftk, pdfnup, and pdfcrop programs from the texlive-extra-utils Debian package. The latter gets installed by

```
sudo apt-get install texlive-extra-utils
```

#### 4.2.3 Spellcheck

The utility doconce spellcheck applies the ispell program for spellcheck. On Debian (including Ubuntu) it is installed by

```
sudo apt-get install ispell
```

#### 4.2.4 Ptex2tex for LaTeX Output

To make LaTeX documents with very flexible choice of typesetting of verbatim code blocks you need ptex2tex, which is installed by

```
svn checkout http://ptex2tex.googlecode.com/svn/trunk/ ptex2tex
cd ptex2tex
sudo python setup.py install
```

It may happen that you need additional style files, you can run a script, cp2texmf.sh:

```
cd latex
sh cp2texmf.sh # copy stylefiles to ~/texmf directory
cd ../..
```

This script copies some special stylefiles that that ptex2tex potentially makes use of. Some more standard stylefiles are also needed. These are installed by

```
sudo apt-get install texlive-latex-recommended texlive-latex-extra
```

on Debian Linux (including Ubuntu) systems. TeXShop on Mac comes with the necessary stylefiles (if not, they can be found by googling and installed manually in the  $\sim/\text{texmf/tex/latex/misc}$  directory).

Note that the doconce ptex2tex command, which needs no installation beyond Doconce itself, can be used as a simpler alternative to the ptex2tex program.

The *minted* LaTeX style is offered by ptex2tex and doconce ptext2tex is popular among many users. This style requires the package Pygments to be installed. On Debian Linux,

```
sudo apt-get install python-pygments
```

Alternatively, the package can be installed manually:

```
hg clone ssh://hg@bitbucket.org/birkenfeld/pygments-main pygments cd pygments sudo python setup.py install
```

If you use the minted style together with ptex2tex, you have to enable it by the -DMINTED command-line argument to ptex2tex. This is not necessary if you run the alternative doconce ptex2tex program.

All use of the minted style requires the -shell-escape command-line argument when running LaTeX, i.e., latex -shell-escape or pdflatex -shell-escape.

#### 4.2.5 reStructuredText (reST) Output

The rst output from Doconce allows further transformation to LaTeX, HTML, XML, OpenOffice, and so on, through the docutils package. The installation of the most recent version can be done by

```
svn checkout http://docutils.svn.sourceforge.net/svnroot/docutils/trunk/docutils
cd docutils
sudo python setup.py install
cd ..
```

To use the OpenOffice suite you will typically on Debian systems install

```
sudo apt-get install unovonv libreoffice libreoffice-dmaths
```

There is a possibility to create PDF files from reST documents using ReportLab instead of LaTeX. The enabling software is rst2pdf. Either download the tarball or clone the svn repository, go to the rst2pdf directory and run the usual sudo python setup.py install.

Output to sphinx requires of course the Sphinx software, installed by

```
hg clone https://bitbucket.org/birkenfeld/sphinx
cd sphinx
sudo python setup.py install
cd ..
```

#### 4.2.6 Markdown and Pandoc Output

The Doconce format pandoc outputs the document in the Pandoc extended Markdown format, which via the pandoc program can be translated to a range of other formats. Installation of Pandoc, written in Haskell, is most easily done by

4.2. Dependencies 21

```
sudo apt-get install pandoc on Debian (Ubuntu) systems.
```

#### 4.2.7 Epydoc Output

When the output format is epydoc one needs that program too, installed by

```
svn co https://epydoc.svn.sourceforge.net/svnroot/epydoc/trunk/epydoc epydoc
cd epydoc
sudo make install
cd ..
```

Remark. Several of the packages above installed from source code are also available in Debian-based system through the apt-get install command. However, we recommend installation directly from the version control system repository as there might be important updates and bug fixes. For svn directories, go to the directory, run svn update, and then sudo python setup.py install. For Mercurial (hg) directories, go to the directory, run hg pull; hg update, and then sudo python setup.py install.

**CHAPTER** 

**FIVE** 

# **INDICES AND TABLES**

- genindex
- modindex
- search

Doconce: Document Once, Include Anywhere

Hans Petter Langtangen [1, 2]

- [1] Simula Research Laboratory
- [2] University of Oslo

Date: Jan 16, 2013

- \* When writing a note, report, manual, etc., do you find it difficult to choose the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?
- \* Do you need to write documents in varying formats but find it difficult to remember all the typesetting details of various formats like LaTeX (http://refcards.com/docs/silvermanj/amslatex/LaTeXRefCard.v2.0.pdf), HTML (http://www.htmlcodetutorial.com/), reStructuredText (http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html), Sphinx (http://sphinx.pocoo.org/contents.html), and wiki (http://code.google.com/p/support/wiki/WikiSyntax)? Would it be convenient

to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?

\* Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.

What Does Doconce Look Like?

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. Here are som examples.

- \* Bullet lists arise from lines starting with \*.
- \* \*Emphasized words\* are surrounded by \*.
- \* \_Words in boldface\_ are surrounded by underscores.
- \* Words from computer code are enclosed in back quotes and then typeset verbatim (in a monospace font).
- \* Section headings are recognied by equality (=) signs before and after the title, and the number of = signs indicates the level of the section: 7 for main section, 5 for subsection, and 3 for subsubsection.

- \* Paragraph headings are recognized by a double underscore before and after the heading.
- \* The abstract of a document starts with \*Abstract\* as paragraph heading, and all text up to the next heading makes up the abstract,
- \* Blocks of computer code can easily be included by placing !bc (begin code) and !ec (end code) commands at separate lines before and after the code block.
- \* Blocks of computer code can also be imported from source files.
- \* Blocks of LaTeX mathematics can easily be included by placing !bt (begin TeX) and !et (end TeX) commands at separate lines before and after the math block.
- \* There is support for both LaTeX and text-like inline mathematics.
- \* Figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported.
- \* Invisible comments in the output format can be inserted throughout the text.
- \* Visible comments can be inserted so that authors and readers can comment upon the text (and at any time turn on/off output of such comments).
- \* There is an exercise environment with many advanced features.
- \* With a preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text.
- \* With Mako one can also have Python code embedded in the Doconce document and thereby parameterize the text (e.g., one text can describe programming in two languages).

Here is an example of some simple text written in the Doconce format::

```
==== A Subsection with Sample Text =====
label{my:first:sec}
```

Ordinary text looks like ordinary text, and the tags used for \_boldface\_ words, \*emphasized\* words, and 'computer' words look natural in plain text. Lists are typeset as you would do in email,

- \* item 1
- \* item 2
- \* item 3

Lists can also have automatically numbered items instead of bullets,

- o item 1
- o item 2
- o item 3

URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl

" .

If the word is URL, the URL itself becomes the link name, as in "URL": "tutorial.do.txt".

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to Section ref{my:first:sec}.

Doconce also allows inline comments of the form [name: comment] (with a space after 'name:'), e.g., such as [hpl: here I will make some remarks to the text]. Inline comments can be removed from the output by a command-line argument (see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

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
	<u>'</u>	

# lines beginning with # are comment lines

The Doconce text above results in the following little document:

A Subsection with Sample Text

Ordinary text looks like ordinary text, and the tags used for \_boldface\_ words, \*emphasized\* words, and computer words look natural in plain text. Lists are typeset as you would do in an email,

- \* item 1
- \* item 2
- \* item 3

Lists can also have numbered items instead of bullets, just use an o (for ordered) instead of the asterisk:

- 1. item 1
- 2. item 2
- 3. item 3

URLs with a link word are possible, as in hpl (http://folk.uio.no/hpl). If the word is URL, the URL itself becomes the link name, as in tutorial.do.txt.

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to the section "A Subsection with Sample Text".

Doconce also allows inline comments such as [hpl: here I will make

some remarks to the text] for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see the section "From Doconce to Other Formats" for an example).

Tables are also supperted, e.g.,

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

## Mathematics and Computer Code

Inline mathematics, such as  $v = \sin(x)$ , allows the formula to be specified both as LaTeX and as plain text.

This results in a professional LaTeX typesetting, but in other formats the text version normally looks better than raw LaTeX mathematics with backslashes. An inline formula like  $v = \sin(x)$  is typeset as::

$$\ln = \sin(x)$$
 \\ v = \sin(x)\\

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula. If you write a lot of mathematics, only the output formats latex, pdflatex, html, sphinx, and pandoc are of interest

and all these support inline LaTeX mathematics so then you will naturally drop the pipe symbol and write just::

```
\ln = \sin(x)
```

However, if you want more textual formats, like plain text or reStructuredText, the text after the pipe symbol may help to make the math formula more readable if there are backslahes or other special LaTeX symbols in the LaTeX code.

Blocks of mathematics are typeset with raw LaTeX, inside !bt and !et (begin TeX, end TeX) instructions::

```
!bt
\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f, label{myeq1}\\
{\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
!et
```

The result looks like this::

```
\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f, label{myeq1}\\
{\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
```

Of course, such blocks only looks nice in formats with support for LaTeX mathematics, and here the align environment in particular (this includes latex, pdflatex, html, and sphinx). The raw LaTeX syntax appears in simpler formats, but can still be useful for those who can read LaTeX syntax.

You can have blocks of computer code, starting and ending with !bc and !ec instructions, respectively::

```
!bc pycod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)
```

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec

Such blocks are formatted as::

from math import sin, pi
def myfunc(x):
 return sin(pi\*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)

A code block must come after some plain sentence (at least for successful output to sphinx, rst, and ASCII-close formats), not directly after a section/paragraph heading or a table.

One can also copy computer code directly from files, either the complete file or specified parts. Computer code is then never duplicated in the documentation (important for the principle of avoiding copying information!).

Another document can be included by writing # #include "mynote.do.txt" at the beginning of a line. Doconce documents have extension do.txt. The do part stands for doconce, while the trailing .txt denotes a text document so that editors gives you plain text editing capabilities.

Macros (Newcommands), Cross-References, Index, and Bibliography

Doconce supports a type of macros via a LaTeX-style \*newcommand\* construction. The newcommands defined in a file with name newcommand\_replace.tex are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names newcommands.tex and newcommands\_keep.tex are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by

!bt and !et in newcommands\_keep.tex to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used elsewhere throughout the text will usually be placed in newcommands\_replace.tex and expanded by Doconce. The definitions of newcommands in the newcommands\*.tex files \*must\* appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the doc/manual/manual.do.txt file (see the demo page (https://doconce.googlecode.com/hg/doc/demos/manual/index.html) for various formats of this document).

From Doconce to Other Formats

Transformation of a Doconce document mydoc.do.txt to various other formats applies the script doconce format::

Terminal> doconce format format mydoc.do.txt

or just::

Terminal> doconce format format mydoc

#### Preprocessing

\_\_\_\_\_

The preprocess and make programs are used to preprocess the file, and options to preprocess and/or make can be added after the filename. For example::

Terminal> doconce format latex mydoc -Dextra\_sections -DVAR1=5 # pre process

Terminal> doconce format latex yourdoc extra\_sections=True VAR1=5 # mak

The variable FORMAT is always defined as the current format when running preprocess or mako. That is, in the last example, FORMAT is defined as latex. Inside the Doconce document one can then perform format specific actions through tests like #if FORMAT == "latex" (for preprocess) or % if FORMAT == "latex": (for mako).

Removal of inline comments

The command-line arguments --no-preprocess and --no-make turn off running preprocess and make, respectively.

Inline comments in the text are removed from the output by::

Terminal> doconce format latex mydoc --skip\_inline\_comments

One can also remove all such comments from the original Doconce file by running::

Terminal> doconce remove\_inline\_comments mydoc

This action is convenient when a Doconce document reaches its final form and comments by different authors should be removed.

HTML

Making an HTML version of a Doconce file mydoc.do.txt is performed by::

Terminal> doconce format html mydoc

The resulting file mydoc.html can be loaded into any web browser for viewing.

The HTML style can be defined either in the header of the HTML file or in an external CSS file. The latter is enabled by the command-line argument --css=filename. There is a default style with blue headings, and a style with the solarized (http://ethanschoonover.com/solarized) color palette, specified by the --html-solarized command line argument. If there is no file with name filename in the --css=filename specification, the blue or solarized styles are written to filename and linked from the HTML document. You can provide your own style sheet either by replacing the content inside the style tags or by specifying a CSS file through the --css=filename option.

If the Pygments package (including the pygmentize program) is installed, code blocks are typeset with aid of this package. The command-line argument --no-pygments-html turns off the use of Pygments and makes code blocks appear with plain (pre) HTML tags. The option --pygments-html-linenos turns on line numbers in Pygments-formatted code blocks.

The HTML file can be embedded in a template if the Doconce document does not have a title (because then there will be no header and footer in the HTML file). The template file must contain valid HTML code and can have three "slots": %(title)s for a title, %(date)s for a date, and %(main)s for the main body of text, i.e., the Doconce document translated to HTML. The title becomes the first heading in the Doconce document, and the date is extracted from the DATE: line, if present. With the template feature one can easily embed

the text in the look and feel of a website. The template can be extracted from the source code of a page at the site; just insert %(title)s and %(date)s at appropriate places and replace the main bod of text by %(main)s. Here is an example::

Terminal > doconce format html mydoc --html-template=mytemplate.html

# Blogs

Doconce can be used for writing blogs provided the blog site accepts raw HTML code. Google's Blogger service (blogname.blogspot.com) is particularly well suited since it also allows extensive LaTeX mathematics via MathJax.

Write the blog text as a Doconce document without any title, author, and date. Then generate HTML as described above. Copy the text and paste it into the text area in the blog, making sure the input format is HTML. On Google's Blogger service you can use Doconce to generate blogs with LaTeX mathematics and pretty (pygmentized) blocks of computer code. See a blog example (http://doconce.blogspot.no) for details on blogging.

\*Warning.\* In the comments after the blog one cannot paste raw HTML code with Ma thJax scripts so there is no support for mathematics in the comments.

WordPress (wordpress.com) allows raw HTML code in blogs, but has very limited

LaTeX support, basically only formulas. The --wordpress option to doconce modifies the HTML code such that all equations are typeset in a way that is acceptable to WordPress.

There is a doconce example (http://doconce.wordpress.com) on blogging with mathematics and code on WordPress.

# Pandoc and Markdown

Output in Pandoc's extended Markdown format results from::

Terminal> doconce format pandoc mydoc

The name of the output file is mydoc.mkd. From this format one can go to numerous other formats::

Terminal > pandoc -R -t mediawiki -o mydoc.mwk --toc mydoc.mkd

Pandoc supports latex, html, odt (OpenOffice), docx (Microsoft Word), rtf, texinfo, to mention some. The -R option makes Pandoc pass raw HTML or LaTeX to the output format instead of ignoring it, while the --toc option generates a table of contents. See the Pandoc documentation (http://johnmacfarlane.net/pandoc/README.html) for the many features of the pandoc program.

Pandoc is useful to go from LaTeX mathematics to, e.g., HTML or MS Word.

There are two ways (experiment to find the best one for your document): doconce format pandoc and then translating using pandoc, or doconce format latex, and then going from LaTeX to the desired format using pandoc.

Here is an example on the latter strategy::

Terminal> doconce format latex mydoc
Terminal> doconce ptex2tex mydoc
Terminal> doconce replace '\Verb!' '\verb!' mydoc.tex
Terminal> pandoc -f latex -t docx -o mydoc.docx mydoc.tex

When we go through pandoc, only single equations or align\* environments are well understood.

Note that Doconce applies the Verb macro from the fancyvrb package while pandoc only supports the standard verb construction for inline verbatim text. Moreover, quite some additional doconce replace and doconce subst edits might be needed on the .mkd or .tex files to successfully have mathematics that is well translated to MS Word. Also when going to reStructuredText using Pandoc, it can be advantageous to go via LaTeX.

Here is an example where we take a Doconce snippet (without title, author, and date), maybe with some unnumbered equations, and quickly generate HTML with mathematics displayed my MathJax::

Terminal> doconce format pandoc mydoc Terminal> pandoc -t html -o mydoc.html -s --mathjax mydoc.mkd

The -s option adds a proper header and footer to the mydoc.html file. This recipe is a quick way of makeing HTML notes with (some) mathematics.

LaTeX

Making a LaTeX file mydoc.tex from mydoc.do.txt is done in two steps:

\*Step 1.\* Filter the doconce text to a pre-LaTeX form mydoc.p.tex for the ptex2tex program (or doconce ptex2tex)::

Terminal> doconce format latex mydoc

LaTeX-specific commands ("newcommands") in math formulas and similar can be placed in files newcommands.tex, newcommands\_keep.tex, or newcommands\_replace.tex (see the section "Macros (Newcommands), Cross-References, Index, and Bibliography").

If these files are present, they are included in the LaTeX document so that your commands are defined.

An option --latex-printed makes some adjustments for documents aimed at being printed. For example, links to web resources are associated with a footnote listing the complete web address (URL).

\*Step 2.\* Run ptex2tex (if you have it) to make a standard LaTeX file::

Terminal> ptex2tex mydoc

In case you do not have ptex2tex, you may run a (very) simplified version::

Terminal> doconce ptex2tex mydoc

Note that Doconce generates a .p.tex file with some preprocessor macros that can be used to steer certain properties of the LaTeX document. For example, to turn on the Helvetica font instead of the standard Computer Modern font, run::

Terminal> ptex2tex -DHELVETICA mydoc
Terminal> doconce ptex2tex mydoc -DHELVETICA # alternative

The title, authors, and date are by default typeset in a non-standard way to enable a nicer treatment of multiple authors having institutions in common. However, the standard LaTeX "maketitle" heading is also available through -DLATEX\_HEADING=traditional. A separate titlepage can be generate by -DLATEX HEADING=titlepage.

Preprocessor variables to be defined or undefined are

- \* BOOK for the "book" documentclass rather than the standard "article" class (necessary if you apply chapter headings)
- \* PALATINO for the Palatino font
- \* HELVETIA for the Helvetica font
- \* A4PAPER for A4 paper size
- \* A6PAPER for A6 paper size (suitable for reading on small devices)
- \* MOVIE15 for using the movie15 LaTeX package to display movies
- \* PREAMBLE to turn the LaTeX preamble on or off (i.e., complete document or document to be included elsewhere)
- \* MINTED for inclusion of the minted package (which requires latex or pdflatex to be run with the -shell-escape option)

The ptex2tex tool makes it possible to easily switch between many different fancy formattings of computer or verbatim code in LaTeX documents. After any !bc command in the Doconce source you can insert verbatim block styles as defined in your .ptex2tex.cfg file, e.g., !bc sys for a terminal session, where sys is set to a certain environment in .ptex2tex.cfg (e.g., CodeTerminal). There are about 40 styles to choose from, and you can easily add new ones.

Also the doconce ptex2tex command supports preprocessor directives for processing the .p.tex file. The command allows specifications of code environments as well. Here is an example::

Terminal> doconce ptex2tex mydoc -DLATEX\_HEADING=traditional \
 -DPALATINO -DA6PAPER \
 "sys=\begin{quote}\begin{verbatim}@\end{verbatim}\end{quote}"

fpro=minted fcod=minted shcod=Verbatim envir=ans:nt

Note that @ must be used to separate the begin and end LaTeX commands, unless only the environment name is given (such as minted above, which implies \begin{minted} {fortran} and \end{minted} as begin and end for blocks inside !bc fpro and !ec). Specifying envir=ans:nt means that all other environments are typeset with the anslistings.sty package, e.g., !bc cppcod will then result in \begin{c++}. If no environments like sys, fpro, or the common envir are defined on the command line, the plain \begin{verbatim} and \end{verbatim} used.

\*Step 2b (optional).\* Edit the mydoc.tex file to your needs. For example, you may want to substitute section by section\* to avoid numbering of sections, you may want to insert linebreaks (and perhaps space) in the title, etc. This can be automatically edited with the aid of the doconce replace and doconce subst commands. The former works with substituting text directly, while the latter performs substitutions using regular expressions. Here are two examples::

A lot of tailored fixes to the LaTeX document can be done by an appropriate set of text replacements and regular expression substitutions. You are anyway encourged to make a script for generating PDF from the LaTeX file.

\*Step 3.\* Compile mydoc.tex and create the PDF file::

```
Terminal> latex mydoc
Terminal> latex mydoc
Terminal> makeindex mydoc  # if index
Terminal> bibitem mydoc  # if bibliography
Terminal> latex mydoc
Terminal> dvipdf mydoc
```

If one wishes to run ptex2tex and use the minted LaTeX package for typesetting code blocks (Minted\_Python, Minted\_Cpp, etc., in ptex2tex specified through the \*pro and \*cod variables in .ptex2tex.cfg or \$HOME/.ptex2tex.cfg), the minted LaTeX package is needed. This package is included by running ptex2tex with the -DMINTED option::

Terminal> ptex2tex -DMINTED mydoc

```
tutorial.txt
In this case, latex must be run with the
-shell-escape option::
        Terminal> latex -shell-escape mydoc
        Terminal> latex -shell-escape mydoc
        Terminal> makeindex mydoc  # if index
Terminal> bibitem mydoc  # if bibliography
        Terminal> latex -shell-escape mydoc
        Terminal> dvipdf mydoc
When running doconce ptex2tex mydoc envir=minted (or other minted
specifications with doconce ptex2tex), the minted package is automatically included so there is no need for the -DMINTED option.
PDFLaTeX
Running pdflatex instead of latex follows almost the same steps,
but the start is::
         Terminal> doconce format latex mydoc
Then ptex2tex is run as explained above, and finally::
        Terminal> pdflatex -shell-escape mydoc
        Terminal> makeindex mydoc  # if index
Terminal> bibitem mydoc  # if bibliography
        Terminal> pdflatex -shell-escape mydoc
Plain ASCII Text
We can go from Doconce "back to" plain untagged text suitable for viewing
in terminal windows, inclusion in email text, or for insertion in
computer source code::
        Terminal> doconce format plain mydoc.do.txt # results in mydoc.txt
reStructuredText
Going from Doconce to reStructuredText gives a lot of possibilities to
go to other formats. First we filter the Doconce text to a
reStructuredText file mydoc.rst::
        Terminal > doconce format rst mydoc.do.txt
We may now produce various other formats::
        Terminal > rst2html.py mydoc.rst > mydoc.html # html
```

```
Terminal> rst2latex.py mydoc.rst > mydoc.tex # latex
Terminal> rst2xml.py mydoc.rst > mydoc.xml # XML
```

Terminal> rst2odt.py mydoc.rst > mydoc.odt # OpenOffice

The OpenOffice file mydoc.odt can be loaded into OpenOffice and saved in, among other things, the RTF format or the Microsoft Word format. However, it is more convenient to use the program unovonv to convert between the many formats OpenOffice supports \*on the command line\*. Run::

Terminal> unoconv --show

to see all the formats that are supported. For example, the following commands take mydoc.odt to Microsoft Office Open XML format, classic MS Word format, and PDF::

```
Terminal> unoconv -f ooxml mydoc.odt
Terminal> unoconv -f doc mydoc.odt
Terminal> unoconv -f pdf mydoc.odt
```

\*Remark about Mathematical Typesetting.\* At the time of this writing, there is n o easy way to go from Doconce and LaTeX mathematics to reST and further to OpenOffice and the "MS Word world". Mathematics is only fully supported by latex as output and to a wide extent also supported by the sphinx output format. Some links for going from LaTeX to Word are listed below.

- \* http://ubuntuforums.org/showthread.php?t=1033441
- \* http://tug.org/utilities/texconv/textopc.html
- \* http://nileshbansal.blogspot.com/2007/12/latex-to-openofficeword.html

#### Sphinx

----

Sphinx documents demand quite some steps in their creation. We have automated most of the steps through the doconce sphinx\_dir command::

The keywords author, title, and version are used in the headings of the Sphinx document. By default, version is 1.0 and the script will try to deduce authors and title from the doconce files file1, file2, etc. that together represent the whole document. Note that none of the individual Doconce files file1, file2, etc. should include the rest as their union makes up the whole document. The default value of dirname is sphinx-rootdir. The theme keyword is used to set the theme for design of HTML output from Sphinx (the default theme is 'default').

With a single-file document in mydoc.do.txt one often just runs::

Terminal > doconce sphinx\_dir mydoc

and then an appropriate Sphinx directory sphinx-rootdir is made with relevant files.

The doconce sphinx\_dir command generates a script automake\_sphinx.py for compiling the Sphinx document into an HTML document. One can either run automake\_sphinx.py or perform the steps in the script manually, possibly with necessary modifications. You should at least read the script prior to executing it to have some idea of what is done.

The doconce sphinx\_dir script copies directories named figs or figures over to the Sphinx directory so that figures are accessible in the Sphinx compilation. If figures or movies are located in other directories, automake\_sphinx.py must be edited accordingly. Files, to which there are local links (not http: or file: URLs), must be placed in the \_static subdirectory of the Sphinx directory. The utility doconce sphinxfix local URLs is run to check for local links in the Doconce file: for each such link, say dir1/dir2/myfile.txt it replaces the link by \_static/myfile.txt and copies dir1/dir2/myfile.txt to a local \_static directory (in the same directory as the script is run). However, we recommend instead that the writer of the document places files in \_static or lets a script do it automatically. The user must copy all \_static/\* files to the \_static subdirectory of the Sphinx directory. It may be wise to always put files, to which there are local links in the Doconce document, in a \_static or \_static-name directory and use these local links. Then links do not need to be modified when creating a Sphinx version of the document.

Doconce comes with a collection of HTML themes for Sphinx documents. These are packed out in the Sphinx directory, the conf.py configuration file for Sphinx is edited accordingly, and a script make-themes.sh can make HTML documents with one or more themes. For example,

to realize the themes fenics and pyramid, one writes::

Terminal> ./make-themes.sh fenics pyramid

The resulting directories with HTML documents are \_build/html\_fenics and \_build/html\_pyramid, respectively. Without arguments, make-themes.sh makes all available themes (!).

If it is not desirable to use the autogenerated scripts explained above, here is the complete manual procedure of generating a Sphinx document from a file mydoc.do.txt.

\*Step 1.\* Translate Doconce into the Sphinx format::

Terminal> doconce format sphinx mydoc

\*Step 2.\* Create a Sphinx root directory

```
tutorial.txt
*Step 5.* Generate, for instance, an HTML version of the Sphinx source::
        make clean
                     # remove old versions
        make html
Sphinx can generate a range of different formats:
standalone HTML, HTML in separate directories with index.html files,
a large single HTML file, JSON files, various help files (the qthelp, HTML,
and Devhelp projects), epub, LaTeX, PDF (via LaTeX), pure text, man pages,
and Texinfo files.
*Step 6.* View the result::
        Terminal> firefox _build/html/index.html
Note that verbatim code blocks can be typeset in a variety of ways
depending the argument that follows !bc: cod gives Python
(code-block:: python in Sphinx syntax) and cppcod gives C++, but
all such arguments can be customized both for Sphinx and LaTeX output.
Wiki Formats
There are many different wiki formats, but Doconce only supports three:
Googlecode wiki (http://code.google.com/p/support/wiki/WikiSyntax),
MediaWiki (http://www.mediawiki.org/wiki/Help:Formatting), and
Creole Wiki (http://www.wikicreole.org/wiki/Creole1.0).
These formats are called
gwiki, mwiki, and cwiki, respectively.
Transformation from Doconce to these formats is done by::
        Terminal> doconce format gwiki mydoc.do.txt
        Terminal> doconce format mwiki mydoc.do.txt
        Terminal > doconce format cwiki mydoc.do.txt
The produced MediaWiki can be tested in the sandbox of
wikibooks.org (http://en.wikibooks.org/wiki/Sandbox). The format
works well with Wikipedia, Wikibooks, and
ShoutWiki (http://doconce.shoutwiki.com/wiki/Doconce_demo),
but not always well elsewhere
(see this example (http://doconcedemo.jumpwiki.com/wiki/First demo)).
Large MediaWiki documents can be made with the
Book creator (http://en.wikipedia.org/w/index.php?title=Special:Book&bookcmd=boo
k creator).
From the MediaWiki format one can go to other formats with aid
of mwlib (http://pediapress.com/code/). This means that one can
easily use Doconce to write Wikibooks (http://en.wikibooks.org)
and publish these in PDF and MediaWiki format, while
at the same time, the book can also be published as a
standard LaTeX book, a Sphinx web document, or a collection of HTML files.
```

The Googlecode wiki document, mydoc.gwiki, is most conveniently stored in a directory which is a clone of the wiki part of the Googlecode project. This is far easier than copying and pasting the entire text into the wiki editor in a web browser.

When the Doconce file contains figures, each figure filename must in the .gwiki file be replaced by a URL where the figure is available. There are instructions in the file for doing this. Usually, one performs this substitution automatically (see next section).

Tweaking the Doconce Output

Occasionally, one would like to tweak the output in a certain format from Doconce. One example is figure filenames when transforming Doconce to reStructuredText. Since Doconce does not know if the .rst file is going to be filtered to LaTeX or HTML, it cannot know if .eps or .png is the most appropriate image filename.

The solution is to use a text substitution command or code with, e.g., sed, perl, python, or scitools subst, to automatically edit the output file from Doconce. It is then wise to run Doconce and the editing commands from a script to automate all steps in going from Doconce to the final format(s). The make.sh files in docs/manual and docs/tutorial constitute comprehensive examples on how such scripts can be made.

#### Demos

\_\_\_\_

The current text is generated from a Doconce format stored in the file::

docs/tutorial/tutorial.do.txt

The file make.sh in the tutorial directory of the Doconce source code contains a demo of how to produce a variety of formats. The source of this tutorial, tutorial.do.txt is the starting point. Running make.sh and studying the various generated files and comparing them with the original tutorial.do.txt file, gives a quick introduction to how Doconce is used in a real case. Here (https://doconce.googlecode.com/hg/doc/demos/tutorial/index.html) is a sample of how this tutorial looks in different formats.

There is another demo in the docs/manual directory which translates the more comprehensive documentation, manual.do.txt, to various formats. The make.sh script runs a set of translations.

Installation of Doconce and its Dependencies

#### Doconce

\_\_\_\_\_

Doconce itself is pure Python code hosted at http://code.google.com/p/doconce. Its installation from the Mercurial (hg) source follows the standard procedure::

```
tutorial.txt
        # Doconce
        hg clone https://doconce.googlecode.com/hg/ doconce
        cd doconce
        sudo python setup.py install
        cd ..
Since Doconce is frequently updated, it is recommended to use the
above procedure and whenever a problem occurs, make sure to
update to the most recent version::
        cd doconce
        hg pull
        hg update
        sudo python setup.py install
Debian GNU/Linux users can also run::
        sudo apt-get install doconce
This installs the latest release and not the most updated and bugfixed
version.
On Ubuntu one needs to run::
        sudo add-apt-repository ppa:scitools/ppa
        sudo apt-get update
        sudo apt-get install doconce
Dependencies
Preprocessors
If you make use of the Preprocess (http://code.google.com/p/preprocess)
preprocessor, this program must be installed::
        svn checkout http://preprocess.googlecode.com/svn/trunk/ preprocess
        cd preprocess
        cd doconce
        sudo python setup.py install
        cd ..
A much more advanced alternative to Preprocess is
Mako (http://www.makotemplates.org). Its installation is most
conveniently done by pip::
        pip install Mako
This command requires pip to be installed. On Debian Linux systems,
such as Ubuntu, the installation is simply done by::
```

sudo apt-get install python-pip

Alternatively, one can install from the pip source code (http://pypi.python.org/pypi/pip).

Mako can also be installed directly from source (http://www.makotemplates.org/download.html): download the tarball, pack it out, go to the directory and run the usual sudo python setup.py install.

Image file handling

Different output formats require different formats of image files. For example, PostScript or Encapuslated PostScript is required for latex output, while HTML needs JPEG, GIF, or PNG formats. Doconce calls up programs from the ImageMagick suite for converting image files to a proper format if needed. The ImageMagick suite (http://www.imagemagick.org/script/index.php) can be installed on all major platforms. On Debian Linux (including Ubuntu) systems one can simply write:

sudo apt-get install imagemagick

The convenience program doconce combine\_images, for combining several images into one, will use montage and convert from ImageMagick and the pdftk, pdfnup, and pdfcrop programs from the texlive-extra-utils Debian package. The latter gets installed by::

sudo apt-get install texlive-extra-utils

Spellcheck

~~~~~~

The utility doconce spellcheck applies the ispell program for spellcheck. On Debian (including Ubuntu) it is installed by::

sudo apt-get install ispell

Ptex2tex for LaTeX Output

To make LaTeX documents with very flexible choice of typesetting of verbatim code blocks you need ptex2tex (http://code.google.com/p/ptex2tex), which is installed by::

svn checkout http://ptex2tex.googlecode.com/svn/trunk/ ptex2tex
cd ptex2tex
sudo python setup.py install

It may happen that you need additional style files, you can run

a script, cp2texmf.sh::

cd latex

sh cp2texmf.sh # copy stylefiles to ~/texmf directory cd ../..

This script copies some special stylefiles that that ptex2tex potentially makes use of. Some more standard stylefiles are also needed. These are installed by::

sudo apt-get install texlive-latex-recommended texlive-latex-extra

on Debian Linux (including Ubuntu) systems. TeXShop on Mac comes with the necessary stylefiles (if not, they can be found by googling and installed manually in the ~/texmf/tex/latex/misc directory).

Note that the doconce ptex2tex command, which needs no installation beyond Doconce itself, can be used as a simpler alternative to the ptex2tex program.

The \*minted\* LaTeX style is offered by ptex2tex and doconce ptext2tex is popular among many users. This style requires the package Pygments (http://pygments.org) to be installed. On Debian Linux::

sudo apt-get install python-pygments

Alternatively, the package can be installed manually::

hg clone ssh://hg@bitbucket.org/birkenfeld/pygments-main pygments cd pygments sudo python setup.py install

If you use the minted style together with ptex2tex, you have to enable it by the -DMINTED command-line argument to ptex2tex.

This is not necessary if you run the alternative doconce ptex2tex program.

#### All

use of the minted style requires the -shell-escape command-line argument when running LaTeX, i.e., latex -shell-escape or pdflatex -shell-escape.

reStructuredText (reST) Output

The rst output from Doconce allows further transformation to LaTeX, HTML, XML, OpenOffice, and so on, through the docutils (http://docutils.sourceforge.net) package. The installation of the most recent version can be done by::

svn checkout http://docutils.svn.sourceforge.net/svnroot/docutils/trunk/
docutils

```
tutorial.txt
       cd docutils
        sudo python setup.py install
To use the OpenOffice suite you will typically on Debian systems install::
        sudo apt-get install unovonv libreoffice libreoffice-dmaths
There is a possibility to create PDF files from reST documents
using ReportLab instead of LaTeX. The enabling software is
rst2pdf (http://code.google.com/p/rst2pdf). Either download the tarball
or clone the svn repository, go to the rst2pdf directory and
run the usual sudo python setup.py install.
Output to sphinx requires of course the
Sphinx software (http://sphinx.pocoo.org),
installed by::
       hg clone https://bitbucket.org/birkenfeld/sphinx
       cd sphinx
       sudo python setup.py install
       cd ..
Markdown and Pandoc Output
The Doconce format pandoc outputs the document in the Pandoc
extended Markdown format, which via the pandoc program can be
translated to a range of other formats. Installation of Pandoc (http://johnmacfa
rlane.net/pandoc/), written in Haskell, is most
easily done by::
        sudo apt-get install pandoc
on Debian (Ubuntu) systems.
Epydoc Output
When the output format is epydoc one needs that program too, installed
by::
        svn co https://epydoc.svn.sourceforge.net/svnroot/epydoc/trunk/epydoc ep
ydoc
       cd epydoc
        sudo make install
       cd ..
*Remark.* Several of the packages above installed from source code
are also available in Debian-based system through the
apt-get install command. However, we recommend installation directly
```

|                                                                                                                                                                                                                                                                                                                    | Printed by hp |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| " tutorial.txt                                                                                                                                                                                                                                                                                                     | ,,            |
| from the version control system repository as there might be important updates and bug fixes. For svn directories, go to the directory, run svn update, and then sudo python setup.py install. For Mercurial (hg) directories, go to the directory, run hg pull; hg update, and then sudo python setup.py install. | t             |
|                                                                                                                                                                                                                                                                                                                    |               |
|                                                                                                                                                                                                                                                                                                                    |               |
|                                                                                                                                                                                                                                                                                                                    |               |
|                                                                                                                                                                                                                                                                                                                    |               |
|                                                                                                                                                                                                                                                                                                                    |               |
|                                                                                                                                                                                                                                                                                                                    |               |
|                                                                                                                                                                                                                                                                                                                    |               |
|                                                                                                                                                                                                                                                                                                                    |               |
|                                                                                                                                                                                                                                                                                                                    |               |

TITLE: Doconce: Document Once, Include Anywhere

BY: Hans Petter Langtangen (Simula Research Laboratory, and University of Oslo)D ATE: today

- When writing a note, report, manual, etc., do you find it difficult to choose the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?

- Do you need to write documents in varying formats but find it
 difficult to remember all the typesetting details of various
 formats like U{LaTeX<http://refcards.com/docs/silvermanj/amslatex/LaTeXRefCar
d.v2.0.pdf>}, U{HTML<http://www.htmlcodetutorial.com/>}, U{reStructuredText<http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html>}, U{Sphinx<http://sphinx.pocoo.org/contents.html>}, and U{wiki<http://code.google.com/p/support/wiki/WikiSyntax>}? Would it be convenient

to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?

- Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.

What Does Doconce Look Like?

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. Here are som examples.

- Bullet lists arise from lines starting with C{\*}.
- I{Emphasized words} are surrounded by C{\*}.
- B{Words in boldface} are surrounded by underscores.
- Words from computer code are enclosed in back quotes and then typeset C{verbatim (in a monospace font)}.
- Section headings are recognied by equality  $(C\{=\})$  signs before and after the title, and the number of  $C\{=\}$  signs indicates the level of the section: 7 for main section, 5 for subsection, and 3 for subsubsection.
- Paragraph headings are recognized by a double underscore before and after the heading.
- The abstract of a document starts with I{Abstract} as paragraph heading, and all text up to the next heading makes up the abstract,
- Blocks of computer code can easily be included by placing
  C{!bc} (begin code) and C{!ec} (end code) commands at separate lines
  before and after the code block.
- Blocks of computer code can also be imported from source files.
- Blocks of LaTeX mathematics can easily be included by placing C{!bt} (begin TeX) and C{!et} (end TeX) commands at separate lines before and after the math block.
- There is support for both LaTeX and text-like inline mathematics.
- Figures and movies with captions, simple tables,

- URLs with links, index list, labels and references are supported.
- Invisible comments in the output format can be inserted throughout the text.
- Visible comments can be inserted so that authors and readers can comment upon the text (and at any time turn on/off output of such comments).
- There is an exercise environment with many advanced features.
- With a preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text.
- With Mako one can also have Python code embedded in the Doconce document and thereby parameterize the text (e.g., one text can describe programming in two languages).

Here is an example of some simple text written in the Doconce format::

```
==== A Subsection with Sample Text =====
label{my:first:sec}
```

Ordinary text looks like ordinary text, and the tags used for \_boldface\_ words, \*emphasized\* words, and 'computer' words look natural in plain text. Lists are typeset as you would do in email,

- \* item 1
- \* item 2
- \* item 3

Lists can also have automatically numbered items instead of bullets,

- o item 1
- o item 2
- o item 3

URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl

If the word is URL, the URL itself becomes the link name, as in "URL": "tutorial.do.txt".

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to Section ref{my:first:sec}.

Doconce also allows inline comments of the form [name: comment] (with a space after 'name:'), e.g., such as [hpl: here I will make some remarks to the text]. Inline comments can be removed from the output by a command-line argument (see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

| 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    |
|      | '<br>    | ·<br>        |

# lines beginning with # are comment lines

The Doconce text above results in the following little document:

A Subsection with Sample Text

\_\_\_\_\_

Ordinary text looks like ordinary text, and the tags used for  $B\{boldface\}$  words,  $I\{emphasized\}$  words, and  $C\{computer\}$  words look natural in plain text. Lists are typeset as you would do in an email,

- item 1
- item 2
- item 3

Lists can also have numbered items instead of bullets, just use an  $C\{o\}$  (for ordered) instead of the asterisk:

- 1. item 1
- 2. item 2
- 3. item 3

URLs with a link word are possible, as in U{hpl<http://folk.uio.no/hpl>}. If the word is URL, the URL itself becomes the link name, as in U{tutorial.do.txt<tutorial.do.txt>}.

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to the section "A Subsection with Sample Text".

Doconce also allows inline comments such as [hpl: here I will make some remarks to the text] for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see the section "From Doconce to Other Formats" for an example).

Tables are also supperted, e.g.,

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

Mathematics and Computer Code

\_\_\_\_\_

Inline mathematics, such as  $M\{v=\sin(x)\}$ , allows the formula to be specified both as LaTeX and as plain text. This results in a professional LaTeX typesetting, but in other formats the text version normally looks better than raw LaTeX mathematics with backslashes. An inline formula like  $M\{v=\sin(x)\}$  is typeset as:

NOTE: A verbatim block has been removed because it causes problems for Epytext.

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula. If you write a lot of mathematics, only the output formats  $C\{\text{latex}\}$ ,  $C\{\text{pdflatex}\}$ ,  $C\{\text{html}\}$ ,  $C\{\text{sphinx}\}$ , and  $C\{\text{pandoc}\}$  are of interest

and all these support inline LaTeX mathematics so then you will naturally drop the pipe symbol and write just::

NOTE: A verbatim block has been removed because it causes problems for Epytext.

However, if you want more textual formats, like plain text or reStructuredText, the text after the pipe symbol may help to make the math formula more readable if there are backslahes or other special LaTeX symbols in the LaTeX code.

Blocks of mathematics are typeset with raw LaTeX, inside C{!bt} and C{!et} (begin TeX, end TeX) instructions::

NOTE: A verbatim block has been removed because it causes problems for Epytext.

The result looks like this::

NOTE: A verbatim block has been removed because it causes problems for Epytext.

Of course, such blocks only looks nice in formats with support for LaTeX mathematics, and here the align environment in particular (this includes C{latex}, C{pdflatex}, C{html}, and C{sphinx}). The raw LaTeX syntax appears in simpler formats, but can still be useful for those who can read LaTeX syntax.

You can have blocks of computer code, starting and ending with  $C\{!bc\}$  and  $C\{!ec\}$  instructions, respectively::

!bc pycod
from math import sin, pi
def myfunc(x):
 return sin(pi\*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec

Such blocks are formatted as::

from math import sin, pi
def myfunc(x):

,,

..

return sin(pi\*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)

A code block must come after some plain sentence (at least for successful output to C{sphinx}, C{rst}, and ASCII-close formats), not directly after a section/paragraph heading or a table.

One can also copy computer code directly from files, either the complete file or specified parts. Computer code is then never duplicated in the documentation (important for the principle of avoiding copying information!).

Another document can be included by writing  $C\{\# \#include \#mynote.do.txt"\}$  at the beginning of a line. Doconce documents have extension  $C\{do.txt\}$ . The  $C\{do\}$  part stands for doconce, while the trailing  $C\{.txt\}$  denotes a text document so that editors gives you plain text editing capabilities.

Macros (Newcommands), Cross-References, Index, and Bibliography

Doconce supports a type of macros via a LaTeX-style I{newcommand} construction. The newcommands defined in a file with name C{newcommand\_replace.tex} are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names C{newcommands.tex} and C{newcommands\_keep.tex} are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by C{!bt} and C{!et} in C{newcommands\_keep.tex} to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used elsewhere throughout the text will usually be placed in C{newcommands\_replace.tex} and expanded by Doconce. The definitions of newcommands in the C{newcommands\*.tex} files I{must} appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the C{doc/manual/manual.do.txt} file (see the U{demo page<https://doconce.googlecode.com/hg/doc/demos/manual/index.html>} for various formats of this document).

From Doconce to Other Formats 

Transformation of a Doconce document C{mydoc.do.txt} to various other formats applies the script C{doconce format}::

Terminal> doconce format format mydoc.do.txt

or just::

Terminal > doconce format format mydoc

# Preprocessing

The C{preprocess} and C{mako} programs are used to preprocess the file, and options to C{preprocess} and/or C{mako} can be added after the filename. For example::

Terminal > doconce format latex mydoc -Dextra\_sections -DVAR1=5 # pre process Terminal > doconce format latex yourdoc extra\_sections=True VAR1=5 # mak

The variable C{FORMAT} is always defined as the current format when running C{preprocess} or C{mako}. That is, in the last example, C{FORMAT} is defined as C{latex}. Inside the Doconce document one can then perform format specific actions through tests like C{#if FORMAT == "latex"} (for C{preprocess}) or C{% if FORMAT == "latex":} (for C{mako}).

Removal of inline comments

The command-line arguments C{--no-preprocess} and C{--no-mako} turn off running C{preprocess} and C{mako}, respectively.

Inline comments in the text are removed from the output by::

Terminal> doconce format latex mydoc --skip\_inline\_comments

One can also remove all such comments from the original Doconce file by running::

Terminal> doconce remove\_inline\_comments mydoc

This action is convenient when a Doconce document reaches its final form and comments by different authors should be removed.

HTML

Making an HTML version of a Doconce file C{mydoc.do.txt} is performed by:

Terminal> doconce format html mydoc

The resulting file C{mydoc.html} can be loaded into any web browser for viewing.

The HTML style can be defined either in the header of the HTML file or in an external CSS file. The latter is enabled by the command-line argument C{--css=filename}. There is a default style with blue headings, and a style with the U{solarized<http://ethanschoonover.com/solarized>} color palette, specified by the C{--html-solarized} command line argument. If there is no file with name C{filename} in the C{--css=filename} specification, the blue or solarized styles are written to C{filename} and linked from the HTML document. You can provide your own style sheet either by replacing the content inside the C{style} tags or by specifying a CSS file through the C{--css=filename} option.

If the Pygments package (including the C{pygmentize} program) is installed, code blocks are typeset with aid of this package. The command-line argument C{--no-pygments-html} turns off the use of Pygments and makes code blocks appear with plain (C{pre}) HTML tags. The option C{--pygments-html-linenos} turns on line numbers in Pygments-formatted code blocks.

The HTML file can be embedded in a template if the Doconce document does not have a title (because then there will be no header and footer in the HTML file). The template file must contain valid HTML code and can have three "slots":  $C\{\%(\text{title})s\}$  for a title,  $C\{\%(\text{date})s\}$  for a date, and  $C\{\%(\text{main})s\}$  for the main body of text, i.e., the Doconce document translated to HTML. The title becomes the first heading in the Doconce document, and the date is extracted from the  $C\{\text{DATE}:\}$  line, if present. With the template feature one can easily embed the text in the look and feel of a website. The template can be extracted from the source code of a page at the site; just insert  $C\{\%(\text{title})s\}$  and  $C\{\%(\text{date})s\}$  at appropriate places and replace the main bod of text by  $C\{\%(\text{main})s\}$ . Here is an example:

Terminal > doconce format html mydoc --html-template=mytemplate.html

# Blogs

Doconce can be used for writing blogs provided the blog site accepts raw HTML code. Google's Blogger service (C{blogname.blogspot.com}) is particularly well suited since it also allows extensive LaTeX mathematics via MathJax.

Write the blog text as a Doconce document without any title, author, and date. Then generate HTML as described above. Copy the text and paste it into the text area in the blog, making sure the input format is HTML. On Google's Blogger service you can use Doconce to generate blogs with LaTeX mathematics and pretty (pygmentized) blocks of computer code. See a U{blog example<http://doconce.blogspot.no>} for details on blogging.

I{Warning.} In the comments after the blog one cannot paste raw HTML code with M athJax  ${\bf M}$ 

scripts so there is no support for mathematics in the comments.

WordPress (C{wordpress.com}) allows raw HTML code in blogs, but has very limited

LaTeX support, basically only formulas. The  $C\{--wordpress\}$  option to  $C\{doconce\}$  modifies the HTML code such that all equations are typeset in a way that is acceptable to WordPress.

There is a U{doconce example<a href="http://doconce.wordpress.com">http://doconce.wordpress.com</a>} on blogging with mathematics and code on WordPress.

#### Pandoc and Markdown

\_\_\_\_\_

Output in Pandoc's extended Markdown format results from::

Terminal> doconce format pandoc mydoc

The name of the output file is C{mydoc.mkd}. From this format one can go to numerous other formats::

Terminal> pandoc -R -t mediawiki -o mydoc.mwk --toc mydoc.mkd

Pandoc supports C{latex}, C{html}, C{odt} (OpenOffice), C{docx} (Microsoft Word), C{rtf}, C{texinfo}, to mention some. The C{-R} option makes Pandoc pass raw HTML or LaTeX to the output format instead of ignoring it, while the C{--toc} option generates a table of contents. See the U{Pandoc documentation<http://johnmacfarlane.net/pandoc/README.html>} for the many features of the C{pandoc} program.

Pandoc is useful to go from LaTeX mathematics to, e.g., HTML or MS Word. There are two ways (experiment to find the best one for your document):  $C\{doconce\ format\ pandoc\}\ and\ then\ translating\ using\ C\{pandoc\}\ ,$  or  $C\{doconce\ format\ latex\}\ ,$  and then going from LaTeX to the desired format using  $C\{pandoc\}\ .$ 

Here is an example on the latter strategy::

Terminal> doconce format latex mydoc

Terminal > doconce ptex2tex mydoc

Terminal> doconce replace '\Verb!' '\verb!' mydoc.tex

Terminal> pandoc -f latex -t docx -o mydoc.docx mydoc.tex

When we go through  $C\{pandoc\}$ , only single equations or  $C\{align*\}$  environments are well understood.

Note that Doconce applies the  $C\{Verb\}$  macro from the  $C\{fancyvrb\}$  package while  $C\{pandoc\}$  only supports the standard  $C\{verb\}$  construction for inline verbatim text. Moreover, quite some additional  $C\{doconce\ replace\}$  and  $C\{doconce\ subst\}$  edits might be needed on the  $C\{.mkd\}$  or  $C\{.tex\}$  files to successfully have mathematics that is well translated to MS Word. Also when going to reStructuredText using Pandoc, it can be advantageous to go via LaTeX.

Here is an example where we take a Doconce snippet (without title, author,

and date), maybe with some unnumbered equations, and quickly generate HTML with mathematics displayed my MathJax::

> Terminal > doconce format pandoc mydoc Terminal > pandoc -t html -o mydoc.html -s --mathjax mydoc.mkd

The  $C\{-s\}$  option adds a proper header and footer to the  $C\{mydoc.html\}$  file. This recipe is a quick way of makeing HTML notes with (some) mathematics.

LaTeX

Making a LaTeX file C{mydoc.tex} from C{mydoc.do.txt} is done in two steps:

I{Step 1.} Filter the doconce text to a pre-LaTeX form C{mydoc.p.tex} for the C{ptex2tex} program (or C{doconce ptex2tex})::

Terminal> doconce format latex mydoc

LaTeX-specific commands ("newcommands") in math formulas and similar can be placed in files C{newcommands.tex}, C{newcommands\_keep.tex}, or C{newcommands\_replace.tex} (see the section "Macros (Newcommands), Cross-Referen ces, Index, and Bibliography").

If these files are present, they are included in the LaTeX document so that your commands are defined.

An option C{--latex-printed} makes some adjustments for documents aimed at being printed. For example, links to web resources are associated with a footnote listing the complete web address (URL).

I{Step 2.} Run C{ptex2tex} (if you have it) to make a standard LaTeX file::

Terminal> ptex2tex mydoc

In case you do not have C{ptex2tex}, you may run a (very) simplified version::

Terminal> doconce ptex2tex mydoc

Note that Doconce generates a C{.p.tex} file with some preprocessor macros that can be used to steer certain properties of the LaTeX document. For example, to turn on the Helvetica font instead of the standard Computer Modern font, run::

> Terminal> ptex2tex -DHELVETICA mydoc Terminal > doconce ptex2tex mydoc -DHELVETICA # alternative

The title, authors, and date are by default typeset in a non-standard way to enable a nicer treatment of multiple authors having institutions in common. However, the standard LaTeX "maketitle" heading is also available through C{-DLATEX\_HEADING=traditional}. A separate titlepage can be generate by C{-DLATEX\_HEADING=titlepage}.

Preprocessor variables to be defined or undefined are

- C{BOOK} for the "book" documentclass rather than the standard "article" class (necessary if you apply chapter headings)
- C{PALATINO} for the Palatino font
- C{HELVETIA} for the Helvetica font
- C{A4PAPER} for A4 paper size
   C{A6PAPER} for A6 paper size (suitable for reading on small devices)
- C{MOVIE15} for using the movie15 LaTeX package to display movies
- C{PREAMBLE} to turn the LaTeX preamble on or off (i.e., complete document or document to be included elsewhere)
- C{MINTED} for inclusion of the minted package (which requires C{latex}
   or C{pdflatex} to be run with the C{-shell-escape} option)

The C{ptex2tex} tool makes it possible to easily switch between many different fancy formattings of computer or verbatim code in LaTeX documents. After any C{!bc} command in the Doconce source you can insert verbatim block styles as defined in your C{.ptex2tex.cfg} file, e.g., C{!bc sys} for a terminal session, where C{sys} is set to a certain environment in C{.ptex2tex.cfg} (e.g., C{CodeTerminal}). There are about 40 styles to choose from, and you can easily add new ones.

Also the  $C\{doconce\ ptex2tex\}$  command supports preprocessor directives for processing the  $C\{.p.tex\}$  file. The command allows specifications of code environments as well. Here is an example::

```
Terminal > doconce ptex2tex mydoc -DLATEX_HEADING=traditional \
          -DPALATINO -DA6PAPER \
          "sys=\begin{quote}\begin{verbatim}@\end{verbatim}\end{quote}"
```

fpro=minted fcod=minted shcod=Verbatim envir=ans:nt

Note that  $C\{@\}$  must be used to separate the begin and end LaTeX commands, unless only the environment name is given (such as C{minted}) above, which implies C{\begin{minted} {fortran} } and C{\end{minted}} as begin and end for blocks inside C{!bc fpro} and C{!ec}). Specifying C{envir=ans:nt} means that all other environments are typeset with the C{anslistings.sty} package, e.g., C{!bc cppcod} will then result in C{\begin{c++}}. If no environments like C{sys}, C{fpro}, or the common C{envir} are defined on the command line, the plain C{\begin{verbatim}} and C{\end{verbatim}} used.

I{Step 2b (optional).} Edit the C{mydoc.tex} file to your needs. For example, you may want to substitute C{section} by C{section\*} to avoid numbering of sections, you may want to insert linebreaks (and perhaps space) in the title, etc. This can be automatically edited with the aid of the C{doconce replace} and C{doconce subst} commands. The former works with substituting text directly, while the latter performs substitutions using regular expressions. Here are two examples::

```
Terminal> doconce replace 'section{' 'section*{' mydoc.tex
Terminal> doconce subst 'title\\{(.+)Using (.+)\\}' \
          'title\{\g<1> \\\\ [1.5mm] Using \g<2>' mydoc.tex
```

# tutorial.epytext A lot of tailored fixes to the LaTeX document can be done by an appropriate set of text replacements and regular expression substitutions. You are anyway encourged to make a script for generating PDF from the LaTeX file. I{Step 3.} Compile C{mydoc.tex} and create the PDF file:: Terminal> latex mydoc Terminal> latex mydoc Terminal> makeindex mydoc # if index Terminal> bibitem mydoc # if bibliography Terminal> latex mydoc Terminal> dvipdf mydoc If one wishes to run C{ptex2tex} and use the minted LaTeX package for typesetting code blocks (C{Minted\_Python}, C{Minted\_Cpp}, etc., in C{ptex2tex} specified through the C{\*pro} and C{\*cod} variables in C{.ptex2tex.cfg} or C{\$HOME/.ptex2tex.cfg}), the minted LaTeX package is needed. This package is included by running C{ptex2tex} with the C{-DMINTED} option:: Terminal> ptex2tex -DMINTED mydoc In this case, C{latex} must be run with the C{-shell-escape} option:: Terminal> latex -shell-escape mydoc Terminal> latex -shell-escape mydoc Terminal> makeindex mydoc # if index Terminal> bibitem mydoc # if bibliography Terminal> latex -shell-escape mydoc Terminal> dvipdf mydoc When running C{doconce ptex2tex mydoc envir=minted} (or other minted specifications with C{doconce ptex2tex}), the minted package is automatically included so there is no need for the C{-DMINTED} option. PDFLaTeX Running C{pdflatex} instead of C{latex} follows almost the same steps, but the start is::

Terminal> pdflatex -shell-escape mydoc Terminal> makeindex mydoc # if index

Then C{ptex2tex} is run as explained above, and finally::

Terminal> doconce format latex mydoc

Terminal> bibitem mydoc # if bibliography

Terminal> pdflatex -shell-escape mydoc

Plain ASCII Text

\_\_\_\_\_

We can go from Doconce "back to" plain untagged text suitable for viewing in terminal windows, inclusion in email text, or for insertion in computer source code::

Terminal> doconce format plain mydoc.do.txt # results in mydoc.txt

#### reStructuredText

\_\_\_\_\_

Going from Doconce to reStructuredText gives a lot of possibilities to go to other formats. First we filter the Doconce text to a reStructuredText file C{mydoc.rst}::

Terminal> doconce format rst mydoc.do.txt

We may now produce various other formats::

```
Terminal> rst2html.py mydoc.rst > mydoc.html # html
Terminal> rst2latex.py mydoc.rst > mydoc.tex # latex
Terminal> rst2xml.py mydoc.rst > mydoc.xml # XML
Terminal> rst2odt.py mydoc.rst > mydoc.odt # OpenOffice
```

The OpenOffice file C{mydoc.odt} can be loaded into OpenOffice and saved in, among other things, the RTF format or the Microsoft Word format. However, it is more convenient to use the program C{unovonv} to convert between the many formats OpenOffice supports I{on the command line}. Run::

Terminal> unoconv --show

to see all the formats that are supported. For example, the following commands take  $C\{mydoc.odt\}$  to Microsoft Office Open XML format, classic MS Word format, and PDF::

```
Terminal> unoconv -f ooxml mydoc.odt
Terminal> unoconv -f doc mydoc.odt
Terminal> unoconv -f pdf mydoc.odt
```

I{Remark about Mathematical Typesetting.} At the time of this writing, there is no easy way to go from Doconce and LaTeX mathematics to reST and further to OpenOffice and the "MS Word world". Mathematics is only fully supported by C{latex} as output and to a wide extent also supported by the C{sphinx} output format. Some links for going from LaTeX to Word are listed below.

-  $U\{http://ubuntuforums.org/showthread.php?t=1033441<http://ubuntuforums.org/showthread.php?t=1033441>\}$ 

- U{http://tug.org/utilities/texconv/textopc.html<http://tug.org/utilities/texc onv/textopc.html>}

- U{http://nileshbansal.blogspot.com/2007/12/latex-to-openofficeword.html<http: //nileshbansal.blogspot.com/2007/12/latex-to-openofficeword.html>}

#### Sphinx

\_\_\_\_\_

Sphinx documents demand quite some steps in their creation. We have automated most of the steps through the C{doconce sphinx\_dir} command::

The keywords  $C\{author\}$ ,  $C\{title\}$ , and  $C\{version\}$  are used in the headings of the Sphinx document. By default,  $C\{version\}$  is 1.0 and the script will try to deduce authors and title from the doconce files  $C\{file1\}$ ,  $C\{file2\}$ , etc. that together represent the whole document. Note that none of the individual Doconce files  $C\{file1\}$ ,  $C\{file2\}$ , etc. should include the rest as their union makes up the whole document. The default value of  $C\{dirname\}$  is  $C\{sphinx-rootdir\}$ . The  $C\{theme\}$  keyword is used to set the theme for design of HTML output from Sphinx (the default theme is  $C\{'default'\}$ ).

With a single-file document in C{mydoc.do.txt} one often just runs::

Terminal> doconce sphinx\_dir mydoc

and then an appropriate Sphinx directory C{sphinx-rootdir} is made with relevant files.

The C{doconce sphinx\_dir} command generates a script C{automake\_sphinx.py} for compiling the Sphinx document into an HTML document. One can either run C{automake\_sphinx.py} or perform the steps in the script manually, possibly with necessary modifications. You should at least read the script prior to executing it to have some idea of what is done.

The C{doconce sphinx\_dir} script copies directories named C{figs} or C{figures} over to the Sphinx directory so that figures are accessible in the Sphinx compilation. If figures or movies are located in other directories, C{automake\_sphinx.py} must be edited accordingly. Files, to which there are local links (not C{http:} or C{file:} URLs), must be placed in the C{\_static} subdirectory of the Sphinx directory. The utility C{doconce sphinxfix\_localURLs} is run to check for local links in the Doconce file: for each such link, say C{dir1/dir2/myfile.txt} it replaces the link by C{\_static/myfile.txt} and copies C{dir1/dir2/myfile.txt} to a local C{\_static} directory (in the same directory as the script is run). However, we recommend instead that the writer of the document places files in C{\_static} or lets a script do it automatically. The user must copy all C{\_static/\*} files to the C{\_static} subdirectory of the Sphinx directory. It may be wise to always put files, to which there are local links in the Doconce

#### tutorial.epytext

may need some edits if you want to specific layout (Sphinx themes) of HTML pages. The C{doconce sphinx\_dir} generator makes an extended C{conv.py} file where, among other things, several useful Sphinx extensions are included.

I{Step 3.} Copy the C{mydoc.rst} file to the Sphinx root directory::

Terminal> cp mydoc.rst sphinx-rootdir

If you have figures in your document, the relative paths to those will be invalid when you work with C{mydoc.rst} in the C{sphinx-rootdir} directory. Either edit C{mydoc.rst} so that figure file paths are correct, or simply copy your figure directories to C{sphinx-rootdir}. Links to local files in C{mydoc.rst} must be modified to links to files in the C{\_static} directory, see comment above.

I{Step 4.} Edit the generated C{index.rst} file so that C{mydoc.rst} is included, i.e., add C{mydoc} to the C{toctree} section so that it becomes::

.. toctree::
 :maxdepth: 2

mydoc

(The spaces before C{mydoc} are important!)

I{Step 5.} Generate, for instance, an HTML version of the Sphinx source::

make clean # remove old versions
make html

Sphinx can generate a range of different formats: standalone HTML, HTML in separate directories with C{index.html} files, a large single HTML file, JSON files, various help files (the qthelp, HTML, and Devhelp projects), epub, LaTeX, PDF (via LaTeX), pure text, man pages, and Texinfo files.

I{Step 6.} View the result::

Terminal> firefox \_build/html/index.html

Note that verbatim code blocks can be typeset in a variety of ways depending the argument that follows  $C\{!bc\}$ :  $C\{cod\}$  gives Python ( $C\{code-block:: python\}$  in Sphinx syntax) and  $C\{cppcod\}$  gives C++, but all such arguments can be customized both for Sphinx and LaTeX output.

Wiki Formats

There are many different wiki formats, but Doconce only supports three: U{Googlecode wiki<http://code.google.com/p/support/wiki/WikiSyntax>},

#### tutorial.epytext U{MediaWiki<http://www.mediawiki.org/wiki/Help:Formatting>}, and U{Creole Wiki<http://www.wikicreole.org/wiki/Creole1.0>}. These formats are called C{gwiki}, C{mwiki}, and C{cwiki}, respectively. Transformation from Doconce to these formats is done by:: Terminal> doconce format gwiki mydoc.do.txt Terminal> doconce format mwiki mydoc.do.txt Terminal> doconce format cwiki mydoc.do.txt The produced MediaWiki can be tested in the U{sandbox of wikibooks.org<ahref="http://en.wikibooks.org/wiki/Sandbox">http://en.wikibooks.org/wiki/Sandbox</a>>}. The format works well with Wikipedia, Wikibooks, and U{ShoutWiki<http://doconce.shoutwiki.com/wiki/Doconce\_demo>}, but not always well elsewhere (see U{this example<http://doconcedemo.jumpwiki.com/wiki/First\_demo>}). Large MediaWiki documents can be made with the U{Book creator<http://en.wikipedia.org/w/index.php?title=Special:Book&bookcmd=bo ok creator> \}. From the MediaWiki format one can go to other formats with aid of U{mwlib<http://pediapress.com/code/>}. This means that one can easily use Doconce to write U{Wikibooks<a href="http://en.wikibooks.org">http://en.wikibooks.org</a>} and publish these in PDF and MediaWiki format, while at the same time, the book can also be published as a standard LaTeX book, a Sphinx web document, or a collection of HTML files. The Googlecode wiki document, C{mydoc.gwiki}, is most conveniently stored in a directory which is a clone of the wiki part of the Googlecode project. This is far easier than copying and pasting the entire text into the wiki editor in a web browser. When the Doconce file contains figures, each figure filename must in the C{.gwiki} file be replaced by a URL where the figure is available. There are instructions in the file for doing this. Usually,

Tweaking the Doconce Output

Occasionally, one would like to tweak the output in a certain format from Doconce. One example is figure filenames when transforming Doconce to reStructuredText. Since Doconce does not know if the C{.rst} file is going to be filtered to LaTeX or HTML, it cannot know if C{.eps} or C{.png} is the most appropriate image filename. The solution is to use a text substitution command or code with, e.g., sed, perl, python, or scitools subst, to automatically edit the output file from Doconce. It is then wise to run Doconce and the editing commands from a script to automate all steps in going from Doconce to the final format(s). The C{make.sh} files in C{docs/manual} and C{docs/tutorial} constitute comprehensive examples on how such scripts can be made.

one performs this substitution automatically (see next section).

Demos

#### tutorial.epytext

The current text is generated from a Doconce format stored in the file::

docs/tutorial/tutorial.do.txt

The file C{make.sh} in the C{tutorial} directory of the Doconce source code contains a demo of how to produce a variety of formats. The source of this tutorial, C{tutorial.do.txt} is the starting point. Running C{make.sh} and studying the various generated files and comparing them with the original C{tutorial.do.txt} file, gives a quick introduction to how Doconce is used in a real case. U{Here<https://doconce.googlecode.com/hg/doc/demos/tutorial/index.html>} is a sample of how this tutorial looks in different formats.

There is another demo in the  $C\{docs/manual\}$  directory which translates the more comprehensive documentation,  $C\{manual.do.txt\}$ , to various formats. The  $C\{make.sh\}$  script runs a set of translations.

Installation of Doconce and its Dependencies

#### Doconce

\_\_\_\_\_

Doconce itself is pure Python code hosted at  $U\{\text{http://code.google.com/p/doconce} \land \text{http://code.google.com/p/doconce} \}$ . Its installation from the Mercurial ( $C\{\text{hg}\}$ ) source follows the standard procedure::

# Doconce
hg clone https://doconce.googlecode.com/hg/ doconce
cd doconce
sudo python setup.py install
cd ..

Since Doconce is frequently updated, it is recommended to use the above procedure and whenever a problem occurs, make sure to update to the most recent version::

cd doconce
hg pull
hg update
sudo python setup.py install

Debian GNU/Linux users can also run::

sudo apt-get install doconce

This installs the latest release and not the most updated and bugfixed version.

On Ubuntu one needs to run::

sudo add-apt-repository ppa:scitools/ppa
sudo apt-get update

## tutorial.epytext sudo apt-get install doconce Dependencies Preprocessors If you make use of the U{Preprocess<http://code.google.com/p/preprocess>} preprocessor, this program must be installed:: svn checkout http://preprocess.googlecode.com/svn/trunk/ preprocess cd preprocess cd doconce sudo python setup.py install A much more advanced alternative to Preprocess is U{Mako<http://www.makotemplates.org>}. Its installation is most conveniently done by C{pip}:: pip install Mako This command requires C{pip} to be installed. On Debian Linux systems, such as Ubuntu, the installation is simply done by:: sudo apt-get install python-pip Alternatively, one can install from the C{pip} U{source code<a href="http://pypi.python">http://pypi.python</a>. org/pypi/pip>}. Mako can also be installed directly from U{source<http://www.makotemplates.org/download.html>}: download the tarball, pack it out, go to the directory and run the usual C{sudo python setup.py install}. Image file handling ~~~~~~~~~~~~~~~ Different output formats require different formats of image files. For example, PostScript or Encapuslated PostScript is required for C{latex} output, while HTML needs JPEG, GIF, or PNG formats. Doconce calls up programs from the ImageMagick suite for converting image files to a proper format if needed. The U{ImageMagick suite<http://www.imagemagick.org/script/index.php>} can be installed on all major platforms. On Debian Linux (including Ubuntu) systems one can simply write:: sudo apt-get install imagemagick

the C{pdftk}, C{pdfnup}, and C{pdfcrop} programs from the C{texlive-extra-utils}

The convenience program C{doconce combine\_images}, for combining several images into one, will use C{montage} and C{convert} from ImageMagick and

```
tutorial.epytext
Debian package. The latter gets installed by::
        sudo apt-get install texlive-extra-utils
Spellcheck
The utility C{doconce spellcheck} applies the C{ispell} program for
spellcheck. On Debian (including Ubuntu) it is installed by::
        sudo apt-get install ispell
Ptex2tex for LaTeX Output
To make LaTeX documents with very flexible choice of typesetting of
verbatim code blocks you need U{ptex2tex<http://code.google.com/p/ptex2tex>},
which is installed by::
        svn checkout http://ptex2tex.googlecode.com/svn/trunk/ ptex2tex
        cd ptex2tex
        sudo python setup.py install
It may happen that you need additional style files, you can run
a script, C{cp2texmf.sh}::
        cd latex
        sh cp2texmf.sh # copy stylefiles to ~/texmf directory
        cd ../..
This script copies some special stylefiles that
that C{ptex2tex} potentially makes use of. Some more standard stylefiles
are also needed. These are installed by::
        sudo apt-get install texlive-latex-recommended texlive-latex-extra
on Debian Linux (including Ubuntu) systems. TeXShop on Mac comes with
the necessary stylefiles (if not, they can be found by googling and installed
manually in the C{~/texmf/tex/latex/misc} directory).
Note that the C{doconce ptex2tex} command, which needs no installation
beyond Doconce itself, can be used as a simpler alternative to the C{ptex2tex}
program.
The I{minted} LaTeX style is offered by C{ptex2tex} and C{doconce ptext2tex}
is popular among many
users. This style requires the package U{Pygments<a href="http://pygments.org">http://pygments.org</a>}
to be installed. On Debian Linux::
```

sudo apt-get install python-pygments

```
tutorial.epytext
Alternatively, the package can be installed manually::
       hg clone ssh://hg@bitbucket.org/birkenfeld/pygments-main pygments
       cd pygments
       sudo python setup.py install
If you use the minted style together with C{ptex2tex}, you have to
enable it by the C{-DMINTED} command-line argument to C{ptex2tex}.
This is not necessary if you run the alternative C{doconce ptex2tex} program.
use of the minted style requires the C{-shell-escape} command-line
argument when running LaTeX, i.e., C{latex -shell-escape} or C{pdflatex
-shell-escape \}.
reStructuredText (reST) Output
The C{rst} output from Doconce allows further transformation to LaTeX,
HTML, XML, OpenOffice, and so on, through the U{docutils<http://docutils.sourcef
orge.net>} package. The installation of the
most recent version can be done by::
        svn checkout http://docutils.svn.sourceforge.net/svnroot/docutils/trunk/
docutils
        cd docutils
        sudo python setup.py install
To use the OpenOffice suite you will typically on Debian systems install::
        sudo apt-get install unovonv libreoffice libreoffice-dmaths
There is a possibility to create PDF files from reST documents
using ReportLab instead of LaTeX. The enabling software is
U{rst2pdf<http://code.google.com/p/rst2pdf>}. Either download the tarball
or clone the svn repository, go to the C{rst2pdf} directory and
run the usual C{sudo python setup.py install}.
Output to C{sphinx} requires of course the
U{Sphinx software<http://sphinx.pocoo.org>},
installed by::
       hg clone https://bitbucket.org/birkenfeld/sphinx
       cd sphinx
       sudo python setup.py install
       cd ..
Markdown and Pandoc Output
```

# tutorial.epytext

The Doconce format C{pandoc} outputs the document in the Pandoc extended Markdown format, which via the C{pandoc} program can be translated to a range of other formats. Installation of U{Pandoc<a href="http://johnmacfarlane.net/pandoc/">http://johnmacfarlane.net/pandoc/</a>, written in Haskell, is most easily done by::

sudo apt-get install pandoc

on Debian (Ubuntu) systems.

Epydoc Output

When the output format is  $C\{epydoc\}$  one needs that program too, installed by::

svn co https://epydoc.svn.sourceforge.net/svnroot/epydoc/trunk/epydoc ep
ydoc

cd epydoc
sudo make install
cd ..

I{Remark.} Several of the packages above installed from source code are also available in Debian-based system through the C{apt-get install} command. However, we recommend installation directly from the version control system repository as there might be important updates and bug fixes. For C{svn} directories, go to the directory, run C{svn update}, and then C{sudo python setup.py install}. For Mercurial (C{hg}) directories, go to the directory, run C{hg pull; hg update}, and then C{sudo python setup.py install}.

"

<wiki:comment> Missing: FIGURE, MOVIE, environments </wiki:comment>

#summary Doconce: Document Once, Include Anywhere

By \*Hans Petter Langtangen\*

==== Jan 16, 2013 ====

- \* When writing a note, report, manual, etc., do you find it difficult to choo se the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?
- \* Do you need to write documents in varying formats but find it difficult to remember all the typesetting details of various formats like [http://refcards.com/docs/silvermanj/amslatex/LaTeXRefCard.v2.0.pdf LaTeX], [http://www.htmlcodet utorial.com/ HTML], [http://docutils.sourceforge.net/docs/ref/rst/restructuredte xt.html reStructuredText], [http://sphinx.pocoo.org/contents.html Sphinx], and [http://code.google.com/p/support/wiki/WikiSyntax wiki]? Would it be convenient to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?
- \* Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.

== What Does Doconce Look Like? ==

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. Here are som examples.

- \* Bullet lists arise from lines starting with '\*'.
- \* \*Emphasized words\* are surrounded by `\*`.
- \* \*Words in boldface\* are surrounded by underscores.
- \* Words from computer code are enclosed in back quotes and then typeset 've rbatim (in a monospace font)'.
- \* Section headings are recognied by equality ('=') signs before and after the title, and the number of '=' signs indicates the level of the section: 7 for main section, 5 for subsection, and 3 for subsubsection.
  - \* Paragraph headings are recognized by a double underscore before and after the heading.
- \* The abstract of a document starts with \*Abstract\* as paragraph heading, a nd all text up to the next heading makes up the abstract,
- \* Blocks of computer code can easily be included by placing '!bc' (begin co de) and '!ec' (end code) commands at separate lines before and after the code block.
  - \* Blocks of computer code can also be imported from source files.
- \* Blocks of LaTeX mathematics can easily be included by placing '!bt' (begin TeX) and '!et' (end TeX) commands at separate lines before and after the math block.
  - \* There is support for both LaTeX and text-like inline mathematics.
- \* Figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported.

# tutorial.gwiki \* Invisible comments in the output format can be inserted throughout the te xt. \* Visible comments can be inserted so that authors and readers can comment upon the text (and at any time turn on/off output of such comments).

- \* There is an exercise environment with many advanced features.
- \* With a preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text.
- \* With Mako one can also have Python code embedded in the Doconce document and thereby parameterize the text (e.g., one text can describe programming in two languages).

Ordinary text looks like ordinary text, and the tags used for \_boldface\_ words, \*emphasized\* words, and 'computer' words look natural in plain text. Lists are typeset as you would do in email,

- \* item 1
- \* item 2
- \* item 3

Lists can also have automatically numbered items instead of bullets,

- o item 1
- o item 2
- o item 3

URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl". If the word is URL, the URL itself becomes the link name, as in "URL": "tutorial.do.txt".

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to Section ref{my:first:sec}.

Doconce also allows inline comments of the form [name: comment] (with a space after 'name:'), e.g., such as [hpl: here I will make some remarks to the text]. Inline comments can be removed from the output by a command-line argument (see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

# lines beginning with # are comment lines
}}}

The Doconce text above results in the following little document:

==== A Subsection with Sample Text ====

Ordinary text looks like ordinary text, and the tags used for \*boldface\* words, \*emphasized\* words, and 'computer' words look natural in plain text. Lists are typeset as you would do in an email,

- \* item 1
- \* item 2
- \* item 3

Lists can also have numbered items instead of bullets, just use an 'o' (for ordered) instead of the asterisk:

```
# item 1
# item 2
# item 3
```

URLs with a link word are possible, as in [http://folk.uio.no/hpl hpl]. If the word is URL, the URL itself becomes the link name, as in tutorial.do.txt.

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to the section [#A\_Subsection\_with\_Sample\_Text].

Doconce also allows inline comments such as [hpl: here I will make some remarks to the text] for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see the section [#From\_Doconce\_to\_Other\_Formats] for an example).

Tables are also supperted, e.g.,

==== Mathematics and Computer Code ====

Inline mathematics, such as 'v =  $\sin(x)$ ', allows the formula to be specified both as LaTeX and as plain text. This results in a professional LaTeX typesetting, but in other formats the text version normally looks better than raw LaTeX mathematics with backslashes. An inline formula like 'v =  $\sin(x)$ ' is typeset as

```
{{{
$\nu = \sin(x)$|$v = sin(x)$
}}}
```

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula. If you write a lot of mathematics, only the output formats 'latex', 'pdflatex', 'html', 'sphinx', and 'pandoc' are of interest

and all these support inline LaTeX mathematics so then you will naturally drop the pipe symbol and write just

{{{

```
tutorial.gwiki
\ln = \sin(x)
} } }
However, if you want more textual formats, like plain text or reStructuredText,
the text after the pipe symbol may help to make the math formula more readable
if there are backslahes or other special LaTeX symbols in the LaTeX code.
Blocks of mathematics are typeset with raw LaTeX, inside
'!bt' and '!et' (begin TeX, end TeX) instructions:
{{{
!bt
\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f, label{myeq1}\\ {\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
!et
}}}
<wiki:comment> Note: !bt and !et (and !bc and !ec below) are used to illustrate
</wiki:comment>
<wiki:comment> tex and code blocks in inside verbatim blocks and are replaced /
wiki:comment>
<wiki:comment> by !bt, !et, !bc, and !ec after all other formatting is finished.
 </wiki:comment>
The result looks like this:
{{{
\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f, label{myeq1}\\
{\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
} } }
Of course, such blocks only looks nice in formats with support
for LaTeX mathematics, and here the align environment in particular
(this includes 'latex', 'pdflatex', 'html', and 'sphinx'). The raw
LaTeX syntax appears in simpler formats, but can still be useful
for those who can read LaTeX syntax.
You can have blocks of computer code, starting and ending with
'!bc' and '!ec' instructions, respectively.
{ { {
!bc pycod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)
import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec
} } }
Such blocks are formatted as
{ { {
from math import sin, pi
def myfunc(x):
    return sin(pi*x)
import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
```

} } }

A code block must come after some plain sentence (at least for successful output to 'sphinx', 'rst', and ASCII-close formats), not directly after a section/paragraph heading or a table.

One can also copy computer code directly from files, either the complete file or specified parts. Computer code is then never duplicated in the documentation (important for the principle of avoiding copying information!).

Another document can be included by writing '# #include "mynote.do.txt"' at the beginning of a line. Doconce documents have extension 'do.txt'. The 'do' part stands for doconce, while the trailing '.txt' denotes a text document so that editors gives you plain text editing capabilities.

==== Macros (Newcommands), Cross-References, Index, and Bibliography ====

Doconce supports a type of macros via a LaTeX-style \*newcommand\* construction. The newcommands defined in a file with name 'newcommand replace.tex' are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names 'newcommands.tex' and 'newcommands\_keep.tex' are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by '!bt' and '!et' in 'newcommands\_keep.tex' to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used elsewhere throughout the text will usually be placed in 'newcommands\_replace.tex' and expanded by Doconce. The definitions of newcommands in the 'newcommands\*.tex' files \*must\* appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the 'doc/manual/manual.do.txt' file (see the [https://doconce.googlecode.com/hg/doc/demos/manual/index.html demo page] for various formats of this document).

<wiki:comment> Example on including another Doconce file (using preprocess): </wr>

== From Doconce to Other Formats ==

```
tutorial.gwiki
Transformation of a Doconce document 'mydoc.do.txt' to various other
formats applies the script 'doconce format':
Terminal > doconce format format mydoc.do.txt
} } }
or just {{{
Terminal > doconce format format mydoc
} } }
==== Preprocessing ====
The 'preprocess' and 'mako' programs are used to preprocess the
file, and options to 'preprocess' and/or 'mako' can be added after the
filename. For example,
{ { {
Terminal> doconce format latex mydoc -Dextra_sections -DVAR1=5
                                                                    # preprocess
Terminal> doconce format latex yourdoc extra_sections=True VAR1=5 # make
The variable 'FORMAT' is always defined as the current format when
running 'preprocess' or 'mako'. That is, in the last example, 'FORMAT' is
defined as 'latex'. Inside the Doconce document one can then perform
format specific actions through tests like '#if FORMAT == "latex"'
(for 'preprocess') or '% if FORMAT == "latex": ' (for 'mako').
==== Removal of inline comments ====
<wiki:comment> mention notes also </wiki:comment>
The command-line arguments '--no-preprocess' and '--no-mako' turn off
running 'preprocess' and 'mako', respectively.
Inline comments in the text are removed from the output by
Terminal> doconce format latex mydoc --skip_inline_comments
One can also remove all such comments from the original Doconce
file by running:
{{{
Terminal > doconce remove inline comments mydoc
}}}
This action is convenient when a Doconce document reaches its final form
and comments by different authors should be removed.
==== HTML ====
Making an HTML version of a Doconce file 'mydoc.do.txt'
is performed by
{{{
Terminal > doconce format html mydoc
} } }
The resulting file 'mydoc.html' can be loaded into any web browser for viewing.
The HTML style can be defined either in the header of the HTML file or
in an external CSS file. The latter is enabled by the command-line
argument '--css=filename'. There is a default style with blue headings,
and a style with the [http://ethanschoonover.com/solarized solarized]
color palette, specified by the '--html-solarized' command line
argument. If there is no file with name 'filename' in the '--css=filename'
```

specification, the blue or solarized styles are written to 'filename' and linked from the HTML document. You can provide your own style sheet either by replacing the content inside the 'style' tags or by specifying a CSS file through the '--css=filename' option.

If the Pygments package (including the 'pygmentize' program) is installed, code blocks are typeset with aid of this package. The command-line argument '--no-pygments-html' turns off the use of Pygments and makes code blocks appear with plain ('pre') HTML tags. The option '--pygments-html-linenos' turns on line numbers in Pygments-formatted code blocks.

The HTML file can be embedded in a template if the Doconce document does not have a title (because then there will be no header and footer in the HTML file). The template file must contain valid HTML code and can have three "slots": '%(title)s' for a title, '%(date)s' for a date, and '%(main)s' for the main body of text, i.e., the Doconce document translated to HTML. The title becomes the first heading in the Doconce document, and the date is extracted from the 'DATE:' line, if present. With the template feature one can easily embed the text in the look and feel of a website. The template can be extracted from the source code of a page at the site; just insert '%(title)s' and '%(date)s' at appropriate places and replace the main bod of text by '%(main)s'. Here is an example: {{

==== Blogs ====

} } }

Doconce can be used for writing blogs provided the blog site accepts raw HTML code. Google's Blogger service ('blogname.blogspot.com') is particularly well suited since it also allows extensive LaTeX mathematics via MathJax.

Write the blog text as a Doconce document without any title, author, and date. Then generate HTML as described above. Copy the text and paste it into the text area in the blog, making sure the input format is HTML. On Google's Blogger service you can use Doconce to generate blogs with LaTeX mathematics and pretty (pygmentized) blocks of computer code. See a [http://doconce.blogspot.no blog example] for details on blogging.

\*Warning.\* In the comments after the blog one cannot paste raw HTML code with Ma thJax scripts so there is no support for mathematics in the comments.

WordPress ('wordpress.com') allows raw HTML code in blogs, but has very limited LaTeX support, basically only formulas. The '--wordpress' option to 'doconce' modifies the HTML code such that all equations are typeset in a way that is acceptable to WordPress.

There is a [http://doconce.wordpress.com doconce example] on blogging with mathematics and code on WordPress.

==== Pandoc and Markdown ====

Output in Pandoc's extended Markdown format results from  $\{\{\{$ 

```
tutorial.gwiki
Terminal> doconce format pandoc mydoc
The name of the output file is 'mydoc.mkd'.
From this format one can go to numerous other formats:
Terminal > pandoc -R -t mediawiki -o mydoc.mwk --toc mydoc.mkd
} } }
Pandoc supports 'latex', 'html', 'odt' (OpenOffice), 'docx' (Microsoft
Word), 'rtf', 'texinfo', to mention some. The '-R' option makes
Pandoc pass raw HTML or LaTeX to the output format instead of ignoring it,
while the '--toc' option generates a table of contents.
See the [http://johnmacfarlane.net/pandoc/README.html Pandoc documentation]
for the many features of the 'pandoc' program.
Pandoc is useful to go from LaTeX mathematics to, e.g., HTML or MS Word.
There are two ways (experiment to find the best one for your document):
'doconce format pandoc' and then translating using 'pandoc', or 'doconce format latex', and then going from LaTeX to the desired format
using 'pandoc'.
Here is an example on the latter strategy:
{{{
Terminal > doconce format latex mydoc
Terminal > doconce ptex2tex mydoc
Terminal> doconce replace '\Verb!' '\verb!' mydoc.tex
Terminal> pandoc -f latex -t docx -o mydoc.docx mydoc.tex
} } }
When we go through 'pandoc', only single equations or 'align*'
environments are well understood.
Note that Doconce applies the 'Verb' macro from the 'fancyvrb' package
while 'pandoc' only supports the standard 'verb' construction for
inline verbatim text. Moreover, quite some additional 'doconce
replace' and 'doconce subst' edits might be needed on the '.mkd' or
`.tex` files to successfully have mathematics that is well translated
to MS Word. Also when going to reStructuredText using Pandoc, it can
be advantageous to go via LaTeX.
Here is an example where we take a Doconce snippet (without title, author,
and date), maybe with some unnumbered equations, and quickly generate
HTML with mathematics displayed my MathJax:
Terminal > doconce format pandoc mydoc
Terminal > pandoc -t html -o mydoc.html -s --mathjax mydoc.mkd
The '-s' option adds a proper header and footer to the 'mydoc.html' file.
This recipe is a quick way of makeing HTML notes with (some) mathematics.
==== LaTeX ====
Making a LaTeX file 'mydoc.tex' from 'mydoc.do.txt' is done in two steps:
<wiki:comment> Note: putting code blocks inside a list is not successful in many
 </wiki:comment>
<wiki:comment> formats - the text may be messed up. A better choice is a paragra
ph </wiki:comment>
<wiki:comment> environment, as used here. </wiki:comment>
*Step 1.* Filter the doconce text to a pre-LaTeX form 'mydoc.p.tex' for
the 'ptex2tex' program (or 'doconce ptex2tex'):
{ { {
```

```
tutorial.gwiki
Terminal> doconce format latex mydoc
LaTeX-specific commands ("newcommands") in math formulas and similar
can be placed in files 'newcommands.tex', 'newcommands keep.tex', or
'newcommands_replace.tex' (see the section [#Macros_(Newcommands),_Cross-Referen
ces, Index, and Bibliography]).
If these files are present, they are included in the LaTeX document
so that your commands are defined.
An option '--latex-printed' makes some adjustments for documents
aimed at being printed. For example, links to web resources are
associated with a footnote listing the complete web address (URL).
*Step 2.* Run 'ptex2tex' (if you have it) to make a standard LaTeX file,
Terminal> ptex2tex mydoc
}}}
In case you do not have 'ptex2tex', you may run a (very) simplified version:
Terminal> doconce ptex2tex mydoc
}}}
Note that Doconce generates a '.p.tex' file with some preprocessor macros
that can be used to steer certain properties of the LaTeX document.
For example, to turn on the Helvetica font instead of the standard
Computer Modern font, run
Terminal> ptex2tex -DHELVETICA mydoc
Terminal > doconce ptex2tex mydoc -DHELVETICA # alternative
The title, authors, and date are by default typeset in a non-standard
way to enable a nicer treatment of multiple authors having
institutions in common. However, the standard LaTeX "maketitle" heading
is also available through `-DLATEX_HEADING=traditional`.
A separate titlepage can be generate by
'-DLATEX_HEADING=titlepage'.
Preprocessor variables to be defined or undefined are
 * 'BOOK' for the "book" documentclass rather than the standard "article" clas
s (necessary if you apply chapter headings)
 * 'PALATINO' for the Palatino font
* 'HELVETIA' for the Helvetica font
 * 'A4PAPER' for A4 paper size
* 'A6PAPER' for A6 paper size (suitable for reading on small devices)
 * 'MOVIE15' for using the movie15 LaTeX package to display movies
 * 'PREAMBLE' to turn the LaTeX preamble on or off (i.e., complete document
                                                                              or
 document to be included elsewhere)
 * 'MINTED' for inclusion of the minted package (which requires 'latex'
                                                                          or 'pd
flatex' to be run with the '-shell-escape' option)
The 'ptex2tex' tool makes it possible to easily switch between many
different fancy formattings of computer or verbatim code in LaTeX
documents. After any '!bc' command in the Doconce source you can
insert verbatim block styles as defined in your '.ptex2tex.cfg'
file, e.g., '!bc sys' for a terminal session, where 'sys' is set to
```

,,

a certain environment in `.ptex2tex.cfg` (e.g., `CodeTerminal`). There are about 40 styles to choose from, and you can easily add

```
tutorial.gwiki
new ones.
Also the 'doconce ptex2tex' command supports preprocessor directives
for processing the '.p.tex' file. The command allows specifications
of code environments as well. Here is an example:
{ { {
Terminal > doconce ptex2tex mydoc -DLATEX_HEADING=traditional \
           -DPALATINO -DA6PAPER \
           "sys=\begin{quote}\begin{verbatim}@\end{verbatim}\end{quote}" \
           fpro=minted fcod=minted shcod=Verbatim envir=ans:nt
} } }
Note that '@' must be used to separate the begin and end LaTeX
commands, unless only the environment name is given (such as 'minted' above, which implies '\begin{minted}{fortran}' and '\end{minted}' as begin and end for blocks inside '!bc fpro' and '!ec'). Specifying
'envir=ans:nt' means that all other environments are typeset with the
'anslistings.sty' package, e.g., '!bc cppcod' will then result in '\begin\{c++\}'. If no environments like 'sys', 'fpro', or the common
'envir' are defined on the command line, the plain '\begin{verbatim}'
and '\end{verbatim}' used.
*Step 2b (optional).* Edit the 'mydoc.tex' file to your needs.
For example, you may want to substitute 'section' by 'section*' to
avoid numbering of sections, you may want to insert linebreaks
(and perhaps space) in the title, etc. This can be automatically
edited with the aid of the 'doconce replace' and 'doconce subst' commands. The former works with substituting text directly, while the
latter performs substitutions using regular expressions.
Here are two examples:
{ { {
Terminal> doconce replace 'section{' 'section*{' mydoc.tex
Terminal> doconce subst 'title\{(.+)Using (.+)\}' \
           }}}
A lot of tailored fixes to the LaTeX document can be done by
an appropriate set of text replacements and regular expression
substitutions. You are anyway encourged to make a script for
generating PDF from the LaTeX file.
*Step 3.* Compile 'mydoc.tex'
and create the PDF file:
{ { {
Terminal > latex mydoc
Terminal> latex mydoc
Terminal> makeindex mydoc
                              # if index
                             # if bibliography
Terminal> bibitem mydoc
Terminal> latex mydoc
Terminal > dvipdf mydoc
} } }
If one wishes to run 'ptex2tex' and use the minted LaTeX package for
typesetting code blocks ('Minted_Python', 'Minted_Cpp', etc., in
'ptex2tex' specified through the '*pro' and '*cod' variables in
'.ptex2tex.cfg' or '$HOME/.ptex2tex.cfg'), the minted LaTeX package is
needed. This package is included by running 'ptex2tex' with the
'-DMINTED' option:
Terminal > ptex2tex -DMINTED mydoc
```

```
tutorial.gwiki
} } }
In this case, 'latex' must be run with the
'-shell-escape' option:
{ { {
Terminal> latex -shell-escape mydoc
Terminal> latex -shell-escape mydoc
Terminal> makeindex mydoc
                            # if index
Terminal> bibitem mydoc # if bibliography
Terminal> latex -shell-escape mydoc
Terminal> dvipdf mydoc
When running 'doconce ptex2tex mydoc envir=minted' (or other minted
specifications with 'doconce ptex2tex'), the minted package is automatically included so there is no need for the '-DMINTED' option.
==== PDFLaTeX ====
Running 'pdflatex' instead of 'latex' follows almost the same steps,
but the start is
{ { {
Terminal> doconce format latex mydoc
Then 'ptex2tex' is run as explained above, and finally
Terminal> pdflatex -shell-escape mydoc
Terminal> makeindex mydoc # if index
Terminal> bibitem mydoc # if bibliography
Terminal> pdflatex -shell-escape mydoc
} } }
==== Plain ASCII Text ====
We can go from Doconce "back to" plain untagged text suitable for viewing
in terminal windows, inclusion in email text, or for insertion in
computer source code:
{ { {
Terminal> doconce format plain mydoc.do.txt # results in mydoc.txt
}}}
==== reStructuredText ====
Going from Doconce to reStructuredText gives a lot of possibilities to
go to other formats. First we filter the Doconce text to a
reStructuredText file 'mydoc.rst':
Terminal> doconce format rst mydoc.do.txt
}}}
We may now produce various other formats:
{ { {
Terminal> rst2html.py mydoc.rst > mydoc.html # html
Terminal> rst2latex.py mydoc.rst > mydoc.tex # latex
Terminal> rst2xml.py mydoc.rst > mydoc.xml # XML
Terminal> rst2odt.py mydoc.rst > mydoc.odt # Open
                        mydoc.rst > mydoc.odt # OpenOffice
} } }
The OpenOffice file 'mydoc.odt' can be loaded into OpenOffice and
saved in, among other things, the RTF format or the Microsoft Word format.
However, it is more convenient to use the program 'unovonv'
to convert between the many formats OpenOffice supports *on the command line*.
```

```
tutorial.gwiki
Run
{ { {
Terminal> unoconv --show
}}}
to see all the formats that are supported.
For example, the following commands take
'mydoc.odt' to Microsoft Office Open XML format,
classic MS Word format, and PDF:
Terminal> unoconv -f ooxml mydoc.odt
Terminal> unoconv -f doc mydoc.odt
Terminal > unoconv -f pdf mydoc.odt
} } }
*Remark about Mathematical Typesetting.* At the time of this writing, there is n
o easy way to go from Doconce
and LaTeX mathematics to reST and further to OpenOffice and the
"MS Word world". Mathematics is only fully supported by 'latex' as
output and to a wide extent also supported by the 'sphinx' output format.
Some links for going from LaTeX to Word are listed below.
 * http://ubuntuforums.org/showthread.php?t=1033441
 * http://tug.org/utilities/texconv/textopc.html
 * http://nileshbansal.blogspot.com/2007/12/latex-to-openofficeword.html
==== Sphinx ====
Sphinx documents demand quite some steps in their creation. We have automated
most of the steps through the 'doconce sphinx_dir' command:
{ { {
Terminal> doconce sphinx_dir author="authors' names" \
          title="some title" version=1.0 dirname=sphinxdir \
          theme=mytheme file1 file2 file3 ...
} } }
The keywords 'author', 'title', and 'version' are used in the headings
of the Sphinx document. By default, 'version' is 1.0 and the script
will try to deduce authors and title from the doconce files 'file1',
'file2', etc. that together represent the whole document. Note that
none of the individual Doconce files 'file1', 'file2', etc. should
include the rest as their union makes up the whole document.
The default value of 'dirname' is 'sphinx-rootdir'. The 'theme'
keyword is used to set the theme for design of HTML output from
Sphinx (the default theme is ''default'').
With a single-file document in 'mydoc.do.txt' one often just runs
Terminal > doconce sphinx_dir mydoc
} } }
and then an appropriate Sphinx directory 'sphinx-rootdir' is made with
relevant files.
The 'doconce sphinx_dir' command generates a script
'automake_sphinx.py' for compiling the Sphinx document into an HTML
document. One can either run 'automake_sphinx.py' or perform the
steps in the script manually, possibly with necessary modifications.
You should at least read the script prior to executing it to have
some idea of what is done.
```

"

The 'doconce sphinx\_dir' script copies directories named 'figs' or 'figures' over to the Sphinx directory so that figures are accessible in the Sphinx compilation. If figures or movies are located in other directories, 'automake\_sphinx.py' must be edited accordingly. Files, to which there are local links (not 'http:' or 'file:' URLs), must be placed in the '\_static' subdirectory of the Sphinx directory. The utility 'doconce sphinxfix\_localURLs' is run to check for local links in the Doconce file: for each such link, say 'dirl/dir2/myfile.txt' it replaces the link by '\_static/myfile.txt' and copies 'dirl/dir2/myfile.txt' to a local '\_static' directory (in the same directory as the script is run). However, we recommend instead that the writer of the document places files in '\_static' or lets a script do it automatically. The user must copy all '\_static/\*' files to the '\_static' subdirectory of the Sphinx directory. It may be wise to always put files, to which there are local links in the Doconce document, in a '\_static' or '\_static-name' directory and use these local links. Then links do not need to be modified when creating a Sphinx version of the document.

```
Sphinx version of the document.
Doconce comes with a collection of HTML themes for Sphinx documents.
These are packed out in the Sphinx directory, the 'conf.py'
configuration file for Sphinx is edited accordingly, and a script
'make-themes.sh' can make HTML documents with one or more themes.
For example,
to realize the themes 'fenics' and 'pyramid', one writes
Terminal> ./make-themes.sh fenics pyramid
} } }
The resulting directories with HTML documents are '_build/html_fenics'
and '_build/html_pyramid', respectively. Without arguments, 'make-themes.sh' makes all available themes (!).
If it is not desirable to use the autogenerated scripts explained
above, here is the complete manual procedure of generating a
Sphinx document from a file 'mydoc.do.txt'.
*Step 1.* Translate Doconce into the Sphinx format:
Terminal > doconce format sphinx mydoc
} } }
*Step 2.* Create a Sphinx root directory
either manually or by using the interactive 'sphinx-quickstart'
program. Here is a scripted version of the steps with the latter:
{{{
mkdir sphinx-rootdir
sphinx-quickstart <<EOF
sphinx-rootdir
n
Name of My Sphinx Document
Author
version
version
.rst
index
```

,,

n y n

```
tutorial.gwiki
n
n
n
У
n
n
У
У
У
EOF
} } }
The autogenerated 'conf.py' file
may need some edits if you want to specific layout (Sphinx themes)
of HTML pages. The 'doconce sphinx_dir' generator makes an extended 'conv.py' file where, among other things, several useful Sphinx extensions
are included.
*Step 3.* Copy the 'mydoc.rst' file to the Sphinx root directory:
{ { {
Terminal> cp mydoc.rst sphinx-rootdir
} } }
If you have figures in your document, the relative paths to those will
be invalid when you work with 'mydoc.rst' in the 'sphinx-rootdir'
directory. Either edit 'mydoc.rst' so that figure file paths are correct,
or simply copy your figure directories to 'sphinx-rootdir'.
Links to local files in 'mydoc.rst' must be modified to links to
files in the '_static' directory, see comment above.
*Step 4.* Edit the generated 'index.rst' file so that 'mydoc.rst'
is included, i.e., add 'mydoc' to the 'toctree' section so that it becomes
{ { {
.. toctree::
   :maxdepth: 2
   mydoc
} } }
(The spaces before 'mydoc' are important!)
*Step 5.* Generate, for instance, an HTML version of the Sphinx source:
{ { {
              # remove old versions
make clean
make html
}}}
Sphinx can generate a range of different formats:
standalone HTML, HTML in separate directories with 'index.html' files,
a large single HTML file, JSON files, various help files (the qthelp, HTML,
and Devhelp projects), epub, LaTeX, PDF (via LaTeX), pure text, man pages,
and Texinfo files.
*Step 6.* View the result:
Terminal > firefox _build/html/index.html
} } }
Note that verbatim code blocks can be typeset in a variety of ways
depending the argument that follows '!bc': 'cod' gives Python
('code-block:: python' in Sphinx syntax) and 'cppcod' gives C++, but
```

```
tutorial.gwiki
all such arguments can be customized both for Sphinx and LaTeX output.
==== Wiki Formats ====
There are many different wiki formats, but Doconce only supports three:
[http://code.google.com/p/support/wiki/WikiSyntax Googlecode wiki],
[http://www.mediawiki.org/wiki/Help:Formatting MediaWiki], and
[http://www.wikicreole.org/wiki/Creole1.0 Creole Wiki].
These formats are called
'gwiki', 'mwiki', and 'cwiki', respectively.
Transformation from Doconce to these formats is done by
Terminal > doconce format gwiki mydoc.do.txt
Terminal> doconce format mwiki mydoc.do.txt
Terminal> doconce format cwiki mydoc.do.txt
} } }
The produced MediaWiki can be tested in the [http://en.wikibooks.org/wiki/Sandbo
x sandbox of
wikibooks.org]. The format
works well with Wikipedia, Wikibooks, and
[http://doconce.shoutwiki.com/wiki/Doconce_demo_ShoutWiki],
but not always well elsewhere
(see [http://doconcedemo.jumpwiki.com/wiki/First demo this example]).
Large MediaWiki documents can be made with the
[http://en.wikipedia.org/w/index.php?title=Special:Book&bookcmd=book_creator Boo
k creator].
From the MediaWiki format one can go to other formats with aid
of [http://pediapress.com/code/ mwlib]. This means that one can
easily use Doconce to write [http://en.wikibooks.org Wikibooks]
and publish these in PDF and MediaWiki format, while
at the same time, the book can also be published as a
standard LaTeX book, a Sphinx web document, or a collection of HTML files.
The Googlecode wiki document, 'mydoc.gwiki', is most conveniently stored
in a directory which is a clone of the wiki part of the Googlecode project.
This is far easier than copying and pasting the entire text into the
wiki editor in a web browser.
When the Doconce file contains figures, each figure filename must in
the '.gwiki' file be replaced by a URL where the figure is
available. There are instructions in the file for doing this. Usually,
one performs this substitution automatically (see next section).
==== Tweaking the Doconce Output ====
Occasionally, one would like to tweak the output in a certain format
from Doconce. One example is figure filenames when transforming
Doconce to reStructuredText. Since Doconce does not know if the
'.rst' file is going to be filtered to LaTeX or HTML, it cannot know
if '.eps' or '.png' is the most appropriate image filename.
The solution is to use a text substitution command or code with, e.g., sed,
perl, python, or scitools subst, to automatically edit the output file
from Doconce. It is then wise to run Doconce and the editing commands
from a script to automate all steps in going from Doconce to the final
format(s). The 'make.sh' files in 'docs/manual' and 'docs/tutorial'
```

```
tutorial.gwiki
constitute comprehensive examples on how such scripts can be made.
==== Demos ====
The current text is generated from a Doconce format stored in the file
docs/tutorial/tutorial.do.txt
} } }
The file 'make.sh' in the 'tutorial' directory of the
Doconce source code contains a demo of how to produce a variety of
formats. The source of this tutorial, 'tutorial.do.txt' is the starting point. Running 'make.sh' and studying the various generated files and comparing them with the original 'tutorial.do.txt' file,
gives a quick introduction to how Doconce is used in a real case.
[https://doconce.googlecode.com/hg/doc/demos/tutorial/index.html Here]
is a sample of how this tutorial looks in different formats.
There is another demo in the 'docs/manual' directory which
translates the more comprehensive documentation, 'manual.do.txt', to
various formats. The 'make.sh' script runs a set of translations.
== Installation of Doconce and its Dependencies ==
==== Doconce ====
Doconce itself is pure Python code hosted at http://code.google.com/p/doconce.
Its installation from the
Mercurial ('hg') source follows the standard procedure:
{ { {
# Doconce
hg clone https://doconce.googlecode.com/hg/ doconce
cd doconce
sudo python setup.py install
cd ..
} } }
Since Doconce is frequently updated, it is recommended to use the
above procedure and whenever a problem occurs, make sure to
update to the most recent version:
{ { {
cd doconce
hq pull
hq update
sudo python setup.py install
}}}
Debian GNU/Linux users can also run
{ { {
sudo apt-get install doconce
}}}
This installs the latest release and not the most updated and bugfixed
version.
On Ubuntu one needs to run
{ { {
sudo add-apt-repository ppa:scitools/ppa
sudo apt-get update
sudo apt-get install doconce
} } }
```

```
tutorial.gwiki
==== Dependencies ====
==== Preprocessors ====
If you make use of the [http://code.google.com/p/preprocess Preprocess]
preprocessor, this program must be installed:
svn checkout http://preprocess.googlecode.com/svn/trunk/ preprocess
cd preprocess
cd doconce
sudo python setup.py install
} } }
A much more advanced alternative to Preprocess is
[http://www.makotemplates.org Mako]. Its installation is most
conveniently done by 'pip',
{ { {
pip install Mako
}}
This command requires 'pip' to be installed. On Debian Linux systems,
such as Ubuntu, the installation is simply done by
sudo apt-get install python-pip
Alternatively, one can install from the 'pip' [http://pypi.python.org/pypi/pip s
ource code].
Mako can also be installed directly from
[http://www.makotemplates.org/download.html source]: download the
tarball, pack it out, go to the directory and run
the usual 'sudo python setup.py install'.
==== Image file handling ====
Different output formats require different formats of image files.
For example, PostScript or Encapuslated PostScript is required for 'latex'
output, while HTML needs JPEG, GIF, or PNG formats.
Doconce calls up programs from the ImageMagick suite for converting
image files to a proper format if needed. The [http://www.imagemagick.org/script
/index.php ImageMagick suite] can be installed on all major platforms.
On Debian Linux (including Ubuntu) systems one can simply write
sudo apt-get install imagemagick
} } }
The convenience program 'doconce combine_images', for combining several
images into one, will use 'montage' and 'convert' from ImageMagick and
the 'pdftk', 'pdfnup', and 'pdfcrop' programs from the 'texlive-extra-utils'
Debian package. The latter gets installed by
{ { {
sudo apt-get install texlive-extra-utils
} } }
```

```
tutorial.gwiki
==== Spellcheck ====
The utility 'doconce spellcheck' applies the 'ispell' program for
spellcheck. On Debian (including Ubuntu) it is installed by
sudo apt-get install ispell
}}}
==== Ptex2tex for LaTeX Output ====
To make LaTeX documents with very flexible choice of typesetting of
verbatim code blocks you need [http://code.google.com/p/ptex2tex ptex2tex],
which is installed by
svn checkout http://ptex2tex.googlecode.com/svn/trunk/ ptex2tex
cd ptex2tex
sudo python setup.py install
It may happen that you need additional style files, you can run
a script, 'cp2texmf.sh':
cd latex
sh cp2texmf.sh # copy stylefiles to ~/texmf directory
cd ../..
} } }
This script copies some special stylefiles that
that 'ptex2tex' potentially makes use of. Some more standard stylefiles
are also needed. These are installed by
sudo apt-get install texlive-latex-recommended texlive-latex-extra
on Debian Linux (including Ubuntu) systems. TeXShop on Mac comes with
the necessary stylefiles (if not, they can be found by googling and installed
manually in the '~/texmf/tex/latex/misc' directory).
Note that the 'doconce ptex2tex' command, which needs no installation
beyond Doconce itself, can be used as a simpler alternative to the 'ptex2tex'
program.
The *minted* LaTeX style is offered by 'ptex2tex' and 'doconce ptext2tex'
is popular among many
users. This style requires the package [http://pygments.org Pygments]
to be installed. On Debian Linux,
{ { {
sudo apt-get install python-pygments
}}}
Alternatively, the package can be installed manually:
hg clone ssh://hg@bitbucket.org/birkenfeld/pygments-main pygments
cd pygments
sudo python setup.py install
} } }
If you use the minted style together with 'ptex2tex', you have to
```

```
tutorial.gwiki
enable it by the '-DMINTED' command-line argument to 'ptex2tex'.
This is not necessary if you run the alternative 'doconce ptex2tex' program.
All
use of the minted style requires the '-shell-escape' command-line
argument when running LaTeX, i.e., 'latex -shell-escape' or 'pdflatex
-shell-escape'.
<wiki:comment> Say something about anslistings.sty </wiki:comment>
==== reStructuredText (reST) Output ====
The 'rst' output from Doconce allows further transformation to LaTeX,
{\tt HTML, XML, OpenOffice, and so on, through the [http://docutils.sourceforge.net details.sourceforge.net details.sourceforg
ocutils] package. The installation of the
most recent version can be done by
{ { {
svn checkout http://docutils.svn.sourceforge.net/svnroot/docutils/trunk/docutils
cd docutils
sudo python setup.py install
}}}
To use the OpenOffice suite you will typically on Debian systems install
sudo apt-get install unovonv libreoffice libreoffice-dmaths
} } }
There is a possibility to create PDF files from reST documents
using ReportLab instead of LaTeX. The enabling software is
[http://code.google.com/p/rst2pdf rst2pdf]. Either download the tarball
or clone the svn repository, go to the 'rst2pdf' directory and
run the usual 'sudo python setup.py install'.
Output to 'sphinx' requires of course the
[http://sphinx.pocoo.org Sphinx software],
installed by
hg clone https://bitbucket.org/birkenfeld/sphinx
cd sphinx
sudo python setup.py install
cd ..
} } }
==== Markdown and Pandoc Output ====
The Doconce format 'pandoc' outputs the document in the Pandoc
extended Markdown format, which via the 'pandoc' program can be translated to a range of other formats. Installation of [http://johnmacfarlane.n
et/pandoc/ Pandoc], written in Haskell, is most
easily done by
{ { {
sudo apt-get install pandoc
on Debian (Ubuntu) systems.
```

```
Printed by hpl
                                           tutorial.gwiki
==== Epydoc Output ====
When the output format is 'epydoc' one needs that program too, installed
by
}}
svn co https://epydoc.svn.sourceforge.net/svnroot/epydoc/trunk/epydoc epydoc
cd epydoc
sudo make install
cd ..
} } }
*Remark.* Several of the packages above installed from source code are also available in Debian-based system through the
'apt-get install' command. However, we recommend installation directly
from the version control system repository as there might be important
updates and bug fixes. For 'svn' directories, go to the directory, run 'svn update', and then 'sudo python setup.py install'. For Mercurial ('hg') directories, go to the directory, run
'hg pull; hg update', and then 'sudo python setup.py install'.
```

n n

<!-- Missing: FIGURE, MOVIE, environments -->

- % Doconce: Document Once, Include Anywhere
- % Hans Petter Langtangen at Simula Research Laboratory and University of Oslo
- % Jan 16, 2013
  - \* When writing a note, report, manual, etc., do you find it difficult to choose the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?
- \* Do you need to write documents in varying formats but find it difficult to remember all the typesetting details of various formats like [LaTeX](http://refcards.com/docs/silvermanj/amslatex/LaTeXRefCard.v2.0.pdf), [HTML](http://www.htmlcodetutorial.com/), [reStructuredText](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html), [Sphinx](http://sphinx.pocoo.org/contents.html), and [wiki](http://code.google.com/p/support/wiki/WikiSyntax)? Would it be convenient

to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?

\* Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.

What Does Doconce Look Like?

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. Here are som examples.

- \* Bullet lists arise from lines starting with '\*'.
- \* \*Emphasized words\* are surrounded by `\*`.
- \* \_Words in boldface\_ are surrounded by underscores.
- \* Words from computer code are enclosed in back quotes and then typeset 'verbatim (in a monospace font)'.
- \* Section headings are recognied by equality ('=') signs before and after the title, and the number of '=' signs indicates the level of the section: 7 for main section, 5 for subsection, and 3 for subsubsection.
- \* Paragraph headings are recognized by a double underscore before and after the heading.
- \* The abstract of a document starts with \*Abstract\* as paragraph heading, and all text up to the next heading makes up the abstract,

- \* Blocks of computer code can easily be included by placing '!bc' (begin code) and '!ec' (end code) commands at separate lines before and after the code block.
- \* Blocks of computer code can also be imported from source files.
- \* Blocks of LaTeX mathematics can easily be included by placing '!bt' (begin TeX) and '!et' (end TeX) commands at separate lines before and after the math block.
- \* There is support for both LaTeX and text-like inline mathematics.
- \* Figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported.
- \* Invisible comments in the output format can be inserted throughout the text.
- \* Visible comments can be inserted so that authors and readers can comment upon the text (and at any time turn on/off output of such comments).
- \* There is an exercise environment with many advanced features.
- \* With a preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text.
- \* With Mako one can also have Python code embedded in the Doconce document and thereby parameterize the text (e.g., one text can describe programming in two languages).

Here is an example of some simple text written in the Doconce format:

Ordinary text looks like ordinary text, and the tags used for \_boldface\_ words, \*emphasized\* words, and 'computer' words look natural in plain text. Lists are typeset as you would do in email,

\* item 1

\label{my:first:sec}

- \* item 2
- \* item 3

Lists can also have automatically numbered items instead of bullets,

- o item 1
- o item 2
- o item 3

URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl". If the word is URL, the URL itself becomes the link name, as in "URL": "tutorial.do.txt".

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to Section ref{my:first:sec}.

Doconce also allows inline comments of the form [name: comment] (with a space after 'name:'), e.g., such as [hpl: here I will make some remarks to the text]. Inline comments can be removed from the output by a command-line argument (see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

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

# lines beginning with # are comment lines

The Doconce text above results in the following little document:

A Subsection with Sample Text

\_\_\_\_\_

Ordinary text looks like ordinary text, and the tags used for \_boldface\_ words, \*emphasized\* words, and 'computer' words look natural in plain text. Lists are typeset as you would do in an email,

- \* item 1
- \* item 2
- \* item 3

Lists can also have numbered items instead of bullets, just use an 'o' (for ordered) instead of the asterisk:

- 1. item 1
- 2. item 2
- 3. item 3

URLs with a link word are possible, as in [hpl](http://folk.uio.no/hpl). If the word is URL, the URL itself becomes the link name, as in <tutorial.do.txt>.

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to the section [A Subsection with Sample Text](#t).

Doconce also allows inline comments such as [hpl: here I will make some remarks to the text] for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see the section [From Doconce to Other Formats](#s) for an example).

Tables are also supperted, e.g.,

| " tutorial md                                                                                                                                                                                                                                                                                                                                         | ,, |  |  |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|--|--|
| " tutorial.md                                                                                                                                                                                                                                                                                                                                         | ,, |  |  |
| time velocity acceleration                                                                                                                                                                                                                                                                                                                            |    |  |  |
| 0.0 1.4186 -5.01<br>2.0 1.376512 11.919<br>4.0 1.1E+1 14.717624                                                                                                                                                                                                                                                                                       |    |  |  |
| Mathematics and Computer Code                                                                                                                                                                                                                                                                                                                         |    |  |  |
| Inline mathematics, such as $nu = \sin(x)$ , allows the formula to be specified both as LaTeX and as plain text. This results in a professional LaTeX typesetting, but in other formats the text version normally looks better than raw LaTeX mathematics with backslashes. An inline formula like $nu = \sin(x)$ is typeset as                       |    |  |  |
| <pre>%                                      </pre>                                                                                                                                                                                                                                                                                                    |    |  |  |
| The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula. If you write a lot of mathematics, only the output formats 'latex', 'pdflatex', 'html', 'sphinx', and 'pandoc' are of interest and all these support inline LaTeX mathematics so then you will naturally drop the pipe symbol and write just        |    |  |  |
| <pre>\$\nu = \sin(x)\$</pre>                                                                                                                                                                                                                                                                                                                          |    |  |  |
| However, if you want more textual formats, like plain text or reStructuredText, the text after the pipe symbol may help to make the math formula more readable if there are backslahes or other special LaTeX symbols in the LaTeX code.  Blocks of mathematics are typeset with raw LaTeX, inside '!bt' and '!et' (begin TeX, end TeX) instructions: |    |  |  |
| <pre>!bt \begin{align} {\partial u\over\partial t} &amp;= \nabla^2 u + f, \label{myeq1}\\ {\partial v\over\partial t} &amp;= \nabla\cdot(q(u)\nabla v) + g \end{align} !et</pre>                                                                                                                                                                      |    |  |  |
| Note: !bt and !et (and !bc and !ec below) are used to illustrate tex and code blocks in inside verbatim blocks and are replaced by !bt, !et, !bc, and !ec after all other formatting is finished The result looks like this:                                                                                                                          |    |  |  |
| <pre>\$\$ \begin{align}</pre>                                                                                                                                                                                                                                                                                                                         |    |  |  |

```
tutorial.md
{\partial u\over\partial t} \&= \nabla^2 u + f, \label{myeq1}
{\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
Of course, such blocks only looks nice in formats with support
for LaTeX mathematics, and here the align environment in particular
(this includes 'latex', 'pdflatex', 'html', and 'sphinx'). The raw LaTeX syntax appears in simpler formats, but can still be useful
for those who can read LaTeX syntax.
You can have blocks of computer code, starting and ending with
'!bc' and '!ec' instructions, respectively.
!bc pycod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)
import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
Such blocks are formatted as
  from math import sin, pi
def myfunc(x):
   return sin(pi*x)
import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
A code block must come after some plain sentence (at least for successful
output to 'sphinx', 'rst', and ASCII-close formats),
not directly after a section/paragraph heading or a table.
One can also copy computer code directly from files, either the
complete file or specified parts. Computer code is then never
duplicated in the documentation (important for the principle of
avoiding copying information!).
Another document can be included by writing `# #include "mynote.do.txt"`
at the beginning of a line. Doconce documents have extension 'do.txt'. The 'do' part stands for doconce, while the trailing '.txt' denotes a text document so that editors gives you
plain text editing capabilities.
Macros (Newcommands), Cross-References, Index, and Bibliography
Doconce supports a type of macros via a LaTeX-style *newcommand*
construction. The newcommands defined in a file with name
```

'newcommand\_replace.tex' are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names 'newcommands.tex' and 'newcommands\_keep.tex' are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by '!bt' and '!et' in 'newcommands\_keep.tex' to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used elsewhere throughout the text will usually be placed in 'newcommands\_replace.tex' and expanded by Doconce. The definitions of newcommands in the 'newcommands\*.tex' files \*must\* appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the 'doc/manual/manual.do.txt' file (see the [demo page](https://doconce.googlecode.com/hg/doc/demos/manual/index.html) for various formats of this document).

<!-- Example on including another Doconce file (using preprocess): -->

From Doconce to Other Formats

Transformation of a Doconce document 'mydoc.do.txt' to various other formats applies the script 'doconce format':

~~~~~~~{.Bash} Terminal> doconce format format mydoc.do.txt

Terminary december refined format in accordance

or just

-----{.Bash}

Terminal> doconce format format mydoc

Preprocessing

\_\_\_\_\_\_

The 'preprocess' and 'mako' programs are used to preprocess the file, and options to 'preprocess' and/or 'mako' can be added after the filename. For example,

# preprocess

aid of this package. The command-line argument '--no-pygments-html' turns off the use of Pygments and makes code blocks appear with plain ('pre') HTML tags. The option '--pygments-html-linenos' turns on line numbers in Pygments-formatted code blocks.

The HTML file can be embedded in a template if the Doconce document does not have a title (because then there will be no header and footer in the HTML file). The template file must contain valid HTML code and can have three "slots": '%(title)s' for a title, '%(date)s' for a date, and '%(main)s' for the main body of text, i.e., the Doconce document translated to HTML. The title becomes the first heading in the Doconce document, and the date is extracted from the 'DATE:' line, if present. With the template feature one can easily embed the text in the look and feel of a website. The template can be extracted from the source code of a page at the site; just insert '%(title)s' and '%(date)s' at appropriate places and replace the main bod of text by '%(main)s'. Here is an example:

-----{.Bash}
Terminal> doconce format html mydoc --html-template=mytemplate.html

### Blogs

Doconce can be used for writing blogs provided the blog site accepts raw HTML code. Google's Blogger service ('blogname.blogspot.com') is particularly well suited since it also allows extensive LaTeX mathematics via MathJax.

Write the blog text as a Doconce document without any title, author, and date. Then generate HTML as described above. Copy the text and paste it into the text area in the blog, making sure the input format is HTML.

On Google's Blogger service you can use Doconce to generate blogs with LaTeX mathematics and pretty (pygmentized) blocks of computer code. See a [blog example](http://doconce.blogspot.no) for details on blogging.

\*Warning.\* In the comments after the blog one cannot paste raw HTML code with Ma thJax

scripts so there is no support for mathematics in the comments.

WordPress ('wordpress.com') allows raw HTML code in blogs, but has very limited

LaTeX support, basically only formulas. The '--wordpress' option to 'doconce' modifies the HTML code such that all equations are typeset in a way that is acceptable to WordPress.

There is a [doconce example](http://doconce.wordpress.com) on blogging with mathematics and code on WordPress.

Pandoc and Markdown
----Output in Pandoc's extended Markdown format results from
------{Bash}
Terminal> doconce format pandoc mydoc

tutorial.md The name of the output file is 'mydoc.mkd'. From this format one can go to numerous other formats: Terminal> pandoc -R -t mediawiki -o mydoc.mwk --toc mydoc.mkd Pandoc supports 'latex', 'html', 'odt' (OpenOffice), 'docx' (Microsoft Word), 'rtf', 'texinfo', to mention some. The '-R' option makes Pandoc pass raw HTML or LaTeX to the output format instead of ignoring it, while the '--toc' option generates a table of contents. See the [Pandoc documentation](http://johnmacfarlane.net/pandoc/README.html) for the many features of the 'pandoc' program. Pandoc is useful to go from LaTeX mathematics to, e.g., HTML or MS Word. There are two ways (experiment to find the best one for your document): 'doconce format pandoc' and then translating using 'pandoc', or 'doconce format latex', and then going from LaTeX to the desired format using 'pandoc'. Here is an example on the latter strategy: ~~~~~~~~~~~~~{.Bash} Terminal> doconce format latex mydoc Terminal> doconce ptex2tex mydoc Terminal> doconce replace '\Verb!' '\verb!' mydoc.tex Terminal> pandoc -f latex -t docx -o mydoc.docx mydoc.tex When we go through 'pandoc', only single equations or 'align\*' environments are well understood. Note that Doconce applies the 'Verb' macro from the 'fancyvrb' package while 'pandoc' only supports the standard 'verb' construction for inline verbatim text. Moreover, quite some additional 'doconce replace' and 'doconce subst' edits might be needed on the '.mkd' or '.tex' files to successfully have mathematics that is well translated to MS Word. Also when going to reStructuredText using Pandoc, it can be advantageous to go via LaTeX. Here is an example where we take a Doconce snippet (without title, author, and date), maybe with some unnumbered equations, and quickly generate HTML with mathematics displayed my MathJax: Terminal> doconce format pandoc mydoc Terminal> pandoc -t html -o mydoc.html -s --mathjax mydoc.mkd The '-s' option adds a proper header and footer to the 'mydoc.html' file. This recipe is a quick way of makeing HTML notes with (some) mathematics. LaTeX Making a LaTeX file 'mydoc.tex' from 'mydoc.do.txt' is done in two steps: <!-- Note: putting code blocks inside a list is not successful in many --> <!-- formats - the text may be messed up. A better choice is a paragraph --> <!-- environment, as used here. -->

| " tutorial.md  | "                  |
|--|--------------------|
| *Step 1.* Filter the doconce text to a pre–LaTeX form 'mydoc.p.tex' for the 'ptex2tex' program (or 'doconce ptex2tex'):  |                    |
| Terminal> doconce format latex mydoc   | ,                  |
| LaTeX-specific commands ("newcommands") in math formulas and simil can be placed in files 'newcommands.tex', 'newcommands_keep.tex', or 'newcommands_replace.tex' (see the section [Macros (Newcommands), Cy)).  If these files are present, they are included in the LaTeX document so that your commands are defined.                |                    |
| An option '—latex—printed' makes some adjustments for documents aimed at being printed. For example, links to web resources are associated with a footnote listing the complete web address (URL).   |                    |
| *Step 2.* Run 'ptex2tex' (if you have it) to make a standard LaTeX file,   |                    |
|  | ~{.Bash}           |
| In case you do not have 'ptex2tex', you may run a (very) simplified version  |                    |
| Terminal> doconce ptex2tex mydoc   | ,                  |
| Note that Doconce generates a '.p.tex' file with some preprocessor macros that can be used to steer certain properties of the LaTeX document. For example, to turn on the Helvetica font instead of the standard Computer Modern font, run   |                    |
| Terminal> ptex2tex -DHELVETICA mydoc Terminal> doconce ptex2tex mydoc -DHELVETICA # alternative  | ~{.Bash}<br>~~~~~~ |
| The title, authors, and date are by default typeset in a non-standard way to enable a nicer treatment of multiple authors having institutions in common. However, the standard LaTeX "maketitle" heading is also available through '-DLATEX_HEADING=traditional'. A separate titlepage can be generate by '-DLATEX_HEADING=titlepage'. | 5                  |
| Preprocessor variables to be defined or undefined are  |                    |
| * 'BOOK' for the "book" documentclass rather than the standard "article" class (necessary if you apply chapter headings)   |                    |
| * 'PALATINO' for the Palatino font   |                    |
| * 'HELVETIA' for the Helvetica font  |                    |
| * 'A4PAPER' for A4 paper size  |                    |

tutorial.md

- \* 'A6PAPER' for A6 paper size (suitable for reading on small devices)
- \* 'MOVIE15' for using the movie15 LaTeX package to display movies
- \* 'PREAMBLE' to turn the LaTeX preamble on or off (i.e., complete document or document to be included elsewhere)
- \* 'MINTED' for inclusion of the minted package (which requires 'latex' or 'pdflatex' to be run with the '-shell-escape' option)

The 'ptex2tex' tool makes it possible to easily switch between many different fancy formattings of computer or verbatim code in LaTeX documents. After any '!bc' command in the Doconce source you can insert verbatim block styles as defined in your '.ptex2tex.cfg' file, e.g., '!bc sys' for a terminal session, where 'sys' is set to a certain environment in '.ptex2tex.cfg' (e.g., 'CodeTerminal'). There are about 40 styles to choose from, and you can easily add new ones.

Also the 'doconce ptex2tex' command supports preprocessor directives for processing the '.p.tex' file. The command allows specifications of code environments as well. Here is an example:

Terminal> doconce ptex2tex mydoc -DLATEX\_HEADING=traditional \ -DPALATINO -DA6PAPER \

"sys=\begin{quote}\begin{verbatim}@\end{verbatim}\end{quote}"\ fpro=minted fcod=minted shcod=Verbatim envir=ans:nt

Note that '@' must be used to separate the begin and end LaTeX commands, unless only the environment name is given (such as 'minted' above, which implies '\begin{minted} {fortran}' and '\end{minted}' as begin and end for blocks inside '!bc fpro' and '!ec'). Specifying 'envir=ans:nt' means that all other environments are typeset with the 'anslistings.sty' package, e.g., '!bc cppcod' will then result in '\begin $\{c++\}$ '. If no environments like 'sys', 'fpro', or the common 'envir' are defined on the command line, the plain 'begin{verbatim}' and '\end{verbatim}' used.

\*Step 2b (optional).\* Edit the 'mydoc.tex' file to your needs. For example, you may want to substitute 'section' by 'section\*' to avoid numbering of sections, you may want to insert linebreaks (and perhaps space) in the title, etc. This can be automatically edited with the aid of the 'doconce replace' and 'doconce subst' commands. The former works with substituting text directly, while the latter performs substitutions using regular expressions. Here are two examples:

A lot of tailored fixes to the LaTeX document can be done by an appropriate set of text replacements and regular expression substitutions. You are anyway encourged to make a script for

| " tutorial.md "  |
|--|
| generating PDF from the LaTeX file.  |
| *Step 3.* Compile 'mydoc.tex' and create the PDF file:   |
| {.Bash}  |
| Terminal> latex mydoc Terminal> latex mydoc Terminal> makeindex mydoc # if index Terminal> bibitem mydoc # if bibliography Terminal> latex mydoc Terminal> dvipdf mydoc  |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   |
| If one wishes to run 'ptex2tex' and use the minted LaTeX package for typesetting code blocks ('Minted_Python', 'Minted_Cpp', etc., in 'ptex2tex' specified through the '*pro' and '*cod' variables in '.ptex2tex.cfg' or '\$HOME/.ptex2tex.cfg'), the minted LaTeX package is needed. This package is included by running 'ptex2tex' with the '-DMINTED' option: |
| ~~~~~~{.Bash}  |
| Terminal> ptex2tex -DMINTED mydoc  |
| In this case, 'latex' must be run with the '-shell-escape' option:   |
| Terminal> latex -shell-escape mydoc Terminal> makeindex mydoc # if index Terminal> bibitem mydoc # if bibliography Terminal> latex -shell-escape mydoc Terminal> bibitem mydoc # if bibliography Terminal> latex -shell-escape mydoc Terminal> dvipdf mydoc  |
| When running 'doconce ptex2tex mydoc envir=minted' (or other minted specifications with 'doconce ptex2tex'), the minted package is automatically included so there is no need for the '-DMINTED' option.   |
| PDFLaTeX<br>   |
| Running 'pdflatex' instead of 'latex' follows almost the same steps, but the start is  |
| Terminal> doconce format latex mydoc   |
| Then 'ptex2tex' is run as explained above, and finally   |
| Terminal> pdflatex -shell-escape mydoc Terminal> makeindex mydoc # if index Terminal> bibitem mydoc # if bibliography Terminal> pdflatex -shell-escape mydoc   |

| " tutorial.md "  |
|--|
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  |
| Plain ASCII Text   |
| We can go from Doconce "back to" plain untagged text suitable for viewing in terminal windows, inclusion in email text, or for insertion in computer source code:  |
| Terminal> doconce format plain mydoc.do.txt # results in mydoc.txt   |
| reStructuredText   |
| Going from Doconce to reStructuredText gives a lot of possibilities to go to other formats. First we filter the Doconce text to a reStructuredText file 'mydoc.rst':   |
| ~~~~~~~~{.Bash}  |
| Terminal> doconce format rst mydoc.do.txt  |
| We may now produce various other formats:  |
| Terminal> rst2html.py mydoc.rst > mydoc.html # html Terminal> rst2latex.py mydoc.rst > mydoc.tex # latex Terminal> rst2xml.py mydoc.rst > mydoc.xml # XML Terminal> rst2odt.py mydoc.rst > mydoc.odt # OpenOffice  |
| The OpenOffice file 'mydoc.odt' can be loaded into OpenOffice and saved in, among other things, the RTF format or the Microsoft Word format. However, it is more convenient to use the program 'unovonv' to convert between the many formats OpenOffice supports *on the command line*. Run                                |
| Terminal> unoconv —show  |
| to see all the formats that are supported. For example, the following commands take 'mydoc.odt' to Microsoft Office Open XML format, classic MS Word format, and PDF:  |
| Towningly uncome form mydes odt  |
| Terminal> unoconv -f ooxml mydoc.odt Terminal> unoconv -f doc mydoc.odt Terminal> unoconv -f pdf mydoc.odt   |
| *Remark about Mathematical Typesetting.* At the time of this writing, there is no easy way to go from Doconce and LaTeX mathematics to reST and further to OpenOffice and the "MS Word world". Mathematics is only fully supported by 'latex' as output and to a wide extent also supported by the 'sphinx' output format. |

tutorial.md \* <http://ubuntuforums.org/showthread.php?t=1033441> \* <http://tug.org/utilities/texconv/textopc.html> \* <http://nileshbansal.blogspot.com/2007/12/latex-to-openofficeword.html> Sphinx Sphinx documents demand quite some steps in their creation. We have automated most of the steps through the 'doconce sphinx\_dir' command: Terminal> doconce sphinx\_dir author="authors' names" \ title="some title" version=1.0 dirname=sphinxdir \ theme=mytheme file1 file2 file3 ... The keywords 'author', 'title', and 'version' are used in the headings of the Sphinx document. By default, 'version' is 1.0 and the script will try to deduce authors and title from the doconce files 'file1'. 'file2', etc. that together represent the whole document. Note that none of the individual Doconce files 'file1', 'file2', etc. should include the rest as their union makes up the whole document. The default value of 'dirname' is 'sphinx-rootdir'. The 'theme' keyword is used to set the theme for design of HTML output from Sphinx (the default theme is ''default''). With a single-file document in 'mydoc.do.txt' one often just runs Terminal> doconce sphinx\_dir mydoc and then an appropriate Sphinx directory 'sphinx-rootdir' is made with relevant files. The 'doconce sphinx\_dir' command generates a script 'automake\_sphinx.py' for compiling the Sphinx document into an HTML document. One can either run 'automake sphinx.py' or perform the steps in the script manually, possibly with necessary modifications. You should at least read the script prior to executing it to have some idea of what is done.

The 'doconce sphinx\_dir' script copies directories named 'figs' or 'figures' over to the Sphinx directory so that figures are accessible in the Sphinx compilation. If figures or movies are located in other directories, 'automake\_sphinx.py' must be edited accordingly. Files, to which there are local links (not 'http:' or 'file:' URLs), must be placed in the '\_static' subdirectory of the Sphinx directory. The utility 'doconce sphinxfix\_localURLs' is run to check for local links in the Doconce file: for each such link, say 'dir1/dir2/myfile.txt' it replaces the link by '\_static/myfile.txt' and copies 'dir1/dir2/myfile.txt' to a local '\_static' directory (in the same directory as the script is run). However, we recommend instead that the writer of the document places files in '\_static' or lets a script do it automatically. The user must copy all '\_static/\*' files to the ' static' subdirectory of the Sphinx directory. It may be wise to

tutorial.md always put files, to which there are local links in the Doconce document, in a '\_static' or '\_static-name' directory and use these local links. Then links do not need to be modified when creating a Sphinx version of the document. Doconce comes with a collection of HTML themes for Sphinx documents. These are packed out in the Sphinx directory, the 'conf.py' configuration file for Sphinx is edited accordingly, and a script 'make-themes.sh' can make HTML documents with one or more themes. For example, to realize the themes 'fenics' and 'pyramid', one writes Terminal> ./make-themes.sh fenics pyramid The resulting directories with HTML documents are '\_build/html\_fenics' and '\_build/html\_pyramid', respectively. Without arguments, 'make-themes.sh' makes all available themes (!). If it is not desirable to use the autogenerated scripts explained above, here is the complete manual procedure of generating a Sphinx document from a file 'mydoc.do.txt'. \*Step 1.\* Translate Doconce into the Sphinx format: ------{.Bash} Terminal> doconce format sphinx mydoc \*Step 2.\* Create a Sphinx root directory either manually or by using the interactive 'sphinx-quickstart' program. Here is a scripted version of the steps with the latter: mkdir sphinx–rootdir sphinx-quickstart <<EOF</pre> sphinx-rootdir n Name of My Sphinx Document Author version version .rst index n y n n n n y n n y y EOF

| " tutorial.md   | " |
|---|---|
| The autogenerated 'conf.py' file may need some edits if you want to specific layout (Sphinx themes) of HTML pages. The 'doconce sphinx_dir' generator makes an extended 'conv.py' file where, among other things, several useful Sphinx extensions are included.  |   |
| *Step 3.* Copy the 'mydoc.rst' file to the Sphinx root directory:   |   |
| If you have figures in your document, the relative paths to those will be invalid when you work with 'mydoc.rst' in the 'sphinx—rootdir' directory. Either edit 'mydoc.rst' so that figure file paths are correct, or simply copy your figure directories to 'sphinx—rootdir'.  Links to local files in 'mydoc.rst' must be modified to links to files in the '_static' directory, see comment above.  *Step 4.* Edit the generated 'index.rst' file so that 'mydoc.rst' is included, i.e., add 'mydoc' to the 'toctree' section so that it becomes |   |
| toctree:: :maxdepth: 2 mydoc  |   |
| (The spaces before 'mydoc' are important!)  *Step 5.* Generate, for instance, an HTML version of the Sphinx source:   |   |
| make clean # remove old versions make html  |   |
| Sphinx can generate a range of different formats: standalone HTML, HTML in separate directories with 'index.html' files, a large single HTML file, JSON files, various help files (the qthelp, HTML, and Devhelp projects), epub, LaTeX, PDF (via LaTeX), pure text, man pages, and Texinfo files.  |   |
| *Step 6.* View the result:  |   |
| Note that verbatim code blocks can be typeset in a variety of ways depending the argument that follows '!bc': 'cod' gives Python ('code-block:: python' in Sphinx syntax) and 'cppcod' gives C++, but all such arguments can be customized both for Sphinx and LaTeX output.  |   |

tutorial.md

#### Wiki Formats

There are many different wiki formats, but Doconce only supports three: [Googlecode wiki](http://code.google.com/p/support/wiki/WikiSyntax), [MediaWiki](http://www.mediawiki.org/wiki/Help:Formatting), and [Creole Wiki](http://www.wikicreole.org/wiki/Creole1.0). These formats are called 'gwiki', 'mwiki', and 'cwiki', respectively. Transformation from Doconce to these formats is done by

~~~~~~{.Bash}

Terminal> doconce format gwiki mydoc.do.txt Terminal> doconce format mwiki mydoc.do.txt Terminal> doconce format cwiki mydoc.do.txt

The produced MediaWiki can be tested in the [sandbox of wikibooks.org](http://en.wikibooks.org/wiki/Sandbox). The format works well with Wikipedia, Wikibooks, and [ShoutWiki](http://doconce.shoutwiki.com/wiki/Doconce\_demo), but not always well elsewhere (see [this example](http://doconcedemo.jumpwiki.com/wiki/First\_demo)).

Large MediaWiki documents can be made with the [Book creator](http://en.wikipedia.org/w/index.php?title=Special:Book&bookcmd=book\_creator). From the MediaWiki format one can go to other formats with aid of [mwlib](http://pediapress.com/code/). This means that one can easily use Doconce to write [Wikibooks](http://en.wikibooks.org) and publish these in PDF and MediaWiki format, while at the same time, the book can also be published as a standard LaTeX book, a Sphinx web document, or a collection of HTML files.

The Googlecode wiki document, 'mydoc.gwiki', is most conveniently stored in a directory which is a clone of the wiki part of the Googlecode project. This is far easier than copying and pasting the entire text into the wiki editor in a web browser.

When the Doconce file contains figures, each figure filename must in the '.gwiki' file be replaced by a URL where the figure is available. There are instructions in the file for doing this. Usually, one performs this substitution automatically (see next section).

Tweaking the Doconce Output

Occasionally, one would like to tweak the output in a certain format from Doconce. One example is figure filenames when transforming Doconce to reStructuredText. Since Doconce does not know if the '.rst' file is going to be filtered to LaTeX or HTML, it cannot know if '.eps' or '.png' is the most appropriate image filename. The solution is to use a text substitution command or code with, e.g., sed, perl, python, or scitools subst, to automatically edit the output file from Doconce. It is then wise to run Doconce and the editing commands from a script to automate all steps in going from Doconce to the final format(s). The 'make.sh' files in 'docs/manual' and 'docs/tutorial' constitute comprehensive examples on how such scripts can be made.

| " tutorial.md                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | ,, |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Demos                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |    |
| The current text is generated from a Doconce format stored in the file                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |    |
| docs/tutorial.do.txt                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |    |
| The file 'make.sh' in the 'tutorial' directory of the Doconce source code contains a demo of how to produce a variety of formats. The source of this tutorial, 'tutorial.do.txt' is the starting point. Running 'make.sh' and studying the various generated files and comparing them with the original 'tutorial.do.txt' file, gives a quick introduction to how Doconce is used in a real case. [Here](https://doconce.googlecode.com/hg/doc/demos/tutorial/index.html) is a sample of how this tutorial looks in different formats. |    |
| There is another demo in the 'docs/manual' directory which translates the more comprehensive documentation, 'manual.do.txt', to various formats. The 'make.sh' script runs a set of translations.                                                                                                                                                                                                                                                                                                                                      |    |
| Installation of Doconce and its Dependencies                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |    |
| Doconce<br>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |    |
| Doconce itself is pure Python code hosted at <a href="http://code.google.com/p/doconce">http://code.google.com/p/doconce</a> . Its installation from the Mercurial ('hg') source follows the standard procedure:                                                                                                                                                                                                                                                                                                                       |    |
| # Doconce hg clone https://doconce.googlecode.com/hg/ doconce cd doconce sudo python setup.py install cd                                                                                                                                                                                                                                                                                                                                                                                                                               |    |
| Since Doconce is frequently updated, it is recommended to use the above procedure and whenever a problem occurs, make sure to update to the most recent version:                                                                                                                                                                                                                                                                                                                                                                       |    |
| cd doconce hg pull hg update sudo python setup.py install                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |
| Debian GNU/Linux users can also run                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |    |
| sudo apt-get install doconce                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |    |
| This installs the latest release and not the most updated and bugfixed                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |    |

| " tutorial.md "                                                                                                                                                                                                                                                                                                                                                                                                       |    |  |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|--|
| version. On Ubuntu one needs to run                                                                                                                                                                                                                                                                                                                                                                                   |    |  |
| sudo add—apt—repository ppa:scitools/ppa sudo apt—get update sudo apt—get install doconce                                                                                                                                                                                                                                                                                                                             |    |  |
| Dependencies                                                                                                                                                                                                                                                                                                                                                                                                          |    |  |
| Preprocessors ~~~~~~                                                                                                                                                                                                                                                                                                                                                                                                  |    |  |
| If you make use of the [Preprocess](http://code.google.com/p/preprocess) preprocessor, this program must be installed:                                                                                                                                                                                                                                                                                                |    |  |
| svn checkout http://preprocess.googlecode.com/svn/trunk/ preprocess cd preprocess cd doconce sudo python setup.py install cd                                                                                                                                                                                                                                                                                          |    |  |
| A much more advanced alternative to Preprocess is [Mako](http://www.makotemplates.org). Its installation is most conveniently done by 'pip',                                                                                                                                                                                                                                                                          |    |  |
| pip install Mako                                                                                                                                                                                                                                                                                                                                                                                                      |    |  |
| This command requires 'pip' to be installed. On Debian Linux systems, such as Ubuntu, the installation is simply done by                                                                                                                                                                                                                                                                                              |    |  |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~                                                                                                                                                                                                                                                                                                                                                                                |    |  |
| sudo apt–get install python–pip                                                                                                                                                                                                                                                                                                                                                                                       |    |  |
| Alternatively, one can install from the 'pip' [source code](http://pypi.python.org/pypi/pip).                                                                                                                                                                                                                                                                                                                         |    |  |
| Mako can also be installed directly from [source](http://www.makotemplates.org/download.html): download the tarball, pack it out, go to the directory and run the usual 'sudo python setup.py install'.                                                                                                                                                                                                               |    |  |
| Image file handling                                                                                                                                                                                                                                                                                                                                                                                                   |    |  |
| Different output formats require different formats of image files. For example, PostScript or Encapuslated PostScript is required for 'latex' output, while HTML needs JPEG, GIF, or PNG formats. Doconce calls up programs from the ImageMagick suite for converting image files to a proper format if needed. The [ImageMagick suite](http://www.imagemagick.org/script/index.php) calls a proper format if needed. | an |  |

| " tutorial.md                                                                                                                                                                                                                                                            | ,, |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| be installed on all major platforms. On Debian Linux (including Ubuntu) systems one can simply write                                                                                                                                                                     |    |
| {.Bash} sudo apt–get install imagemagick                                                                                                                                                                                                                                 |    |
| The convenience program 'doconce combine_images', for combining several images into one, will use 'montage' and 'convert' from ImageMagick and the 'pdftk', 'pdfnup', and 'pdfcrop' programs from the 'texlive–extra–utils' Debian package. The latter gets installed by |    |
|                                                                                                                                                                                                                                                                          |    |
| Spellcheck ~~~~~~                                                                                                                                                                                                                                                        |    |
| The utility 'doconce spellcheck' applies the 'ispell' program for spellcheck. On Debian (including Ubuntu) it is installed by                                                                                                                                            |    |
| {.Bash} sudo apt-get install ispell                                                                                                                                                                                                                                      |    |
| Ptex2tex for LaTeX Output                                                                                                                                                                                                                                                |    |
| To make LaTeX documents with very flexible choice of typesetting of verbatim code blocks you need [ptex2tex](http://code.google.com/p/ptex2tex), which is installed by                                                                                                   |    |
| svn checkout http://ptex2tex.googlecode.com/svn/trunk/ ptex2tex cd ptex2tex sudo python setup.py install                                                                                                                                                                 |    |
| It may happen that you need additional style files, you can run a script, 'cp2texmf.sh':                                                                                                                                                                                 |    |
| cd latex sh cp2texmf.sh # copy stylefiles to ~/texmf directory cd/                                                                                                                                                                                                       |    |
| This script copies some special stylefiles that that 'ptex2tex' potentially makes use of. Some more standard stylefiles are also needed. These are installed by                                                                                                          |    |
| ~~~~~~{.Bash}                                                                                                                                                                                                                                                            |    |

| " tutorial.md                                                                                                                                                                                                   | ,,                      |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| sudo apt-get install texlive-latex-recommended texlive-latex-extra                                                                                                                                              |                         |
| on Debian Linux (including Ubuntu) systems. TeXShop on Mac comes with the necessary stylefiles (if not, they can be found by googling and installed manually in the '~/texmf/tex/latex/misc' directory).        |                         |
| Note that the 'doconce ptex2tex' command, which needs no installation beyond Doconce itself, can be used as a simpler alternative to the 'ptex2tex' program.                                                    |                         |
| The *minted* LaTeX style is offered by 'ptex2tex' and 'doconce ptext2tex' is popular among many users. This style requires the package [Pygments](http://pygments.org) to be installed. On Debian Linux,        |                         |
| <pre>sudo apt-get install python-pygments </pre>                                                                                                                                                                |                         |
| Alternatively, the package can be installed manually:                                                                                                                                                           |                         |
| hg clone ssh://hg@bitbucket.org/birkenfeld/pygments-main pygments cd pygments sudo python setup.py install                                                                                                      |                         |
| If you use the minted style together with 'ptex2tex', you have to enable it by the '-DMINTED' command-line argument to 'ptex2tex'. This is not necessary if you run the alternative 'doconce ptex2tex' program. |                         |
| All use of the minted style requires the '-shell-escape' command-line argument when running LaTeX, i.e., 'latex -shell-escape' or 'pdflatex -shell-escape'.                                                     |                         |
| Say something about anslistings.sty                                                                                                                                                                             |                         |
| reStructuredText (reST) Output                                                                                                                                                                                  |                         |
| The 'rst' output from Doconce allows further transformation to LaTeX, HTML, XML, OpenOffice, and so on, through the [docutils](http://docutils.sourceforge.net) particle the most recent version can be done by | ckage. The installation |
| svn checkout http://docutils.svn.sourceforge.net/svnroot/docutils/trunk/docutils cd docutils sudo python setup.py install cd                                                                                    |                         |
| To use the OpenOffice suite you will typically on Debian systems install                                                                                                                                        |                         |
| {.Bash} sudo apt-get install unovonv libreoffice libreoffice-dmaths                                                                                                                                             |                         |

| " tutorial.md "                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| There is a possibility to create PDF files from reST documents using ReportLab instead of LaTeX. The enabling software is [rst2pdf](http://code.google.com/p/rst2pdf). Either download the tarball or clone the svn repository, go to the 'rst2pdf' directory and run the usual 'sudo python setup.py install'.                                                                                                                                                                                                                |
| Output to 'sphinx' requires of course the [Sphinx software](http://sphinx.pocoo.org), installed by                                                                                                                                                                                                                                                                                                                                                                                                                             |
| hg clone https://bitbucket.org/birkenfeld/sphinx cd sphinx sudo python setup.py install cd                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Markdown and Pandoc Output                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| The Doconce format 'pandoc' outputs the document in the Pandoc extended Markdown format, which via the 'pandoc' program can be translated to a range of other formats. Installation of [Pandoc](http://johnmacfarlane.net/pandoc/), written in Haskell, is most easily done by                                                                                                                                                                                                                                                 |
| sudo apt-get install pandoc                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| on Debian (Ubuntu) systems.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Epydoc Output                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| When the output format is 'epydoc' one needs that program too, installed by                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| svn co https://epydoc.svn.sourceforge.net/svnroot/epydoc/trunk/epydoc epydoc epydoc sudo make install cd                                                                                                                                                                                                                                                                                                                                                                                                                       |
| *Remark.* Several of the packages above installed from source code are also available in Debian—based system through the 'apt—get install' command. However, we recommend installation directly from the version control system repository as there might be important updates and bug fixes. For 'svn' directories, go to the directory, run 'svn update', and then 'sudo python setup.py install'. For Mercurial ('hg') directories, go to the directory, run 'hg pull; hg update', and then 'sudo python setup.py install'. |

# Doconce: Document Once, Include Anywhere Documentation

Release 1.0

**Hans Petter Langtangen** 

# **CONTENTS**

| 1 | Doconce: Document Once, Include Anywhere                                                                                                                                                                                               |  |  |
|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| 2 | What Does Doconce Look Like?  2.1 A Subsection with Sample Text                                                                                                                                                                        |  |  |
| 3 | 110m Deconce to Other 101mm                                                                                                                                                                                                            |  |  |
|   | 3.1 Preprocessing 3.2 Removal of inline comments 3.3 HTML 3.4 Blogs 3.5 Pandoc and Markdown 3.6 LaTeX 3.7 PDFLaTeX 3.8 Plain ASCII Text 3.9 reStructuredText 3.10 Sphinx 3.11 Wiki Formats 3.12 Tweaking the Doconce Output 3.13 Demos |  |  |
| 4 |                                                                                                                                                                                                                                        |  |  |
| 5 | Indices and tables                                                                                                                                                                                                                     |  |  |

Contents:

CONTENTS 1

2 CONTENTS

# DOCONCE: DOCUMENT ONCE, INCLUDE ANYWHERE

Author Hans Petter Langtangen

Date Jan 16, 2013

- When writing a note, report, manual, etc., do you find it difficult to choose the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?
- Do you need to write documents in varying formats but find it difficult to remember all the typesetting details of various formats like LaTeX, HTML, reStructuredText, Sphinx, and wiki? Would it be convenient to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?
- Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.



# WHAT DOES DOCONCE LOOK LIKE?

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. Here are som examples.

- Bullet lists arise from lines starting with \*.
- Emphasized words are surrounded by \*.
- Words in boldface are surrounded by underscores.
- Words from computer code are enclosed in back quotes and then typeset verbatim (in a monospace font).
- Section headings are recognied by equality (=) signs before and after the title, and the number of = signs indicates the level of the section: 7 for main section, 5 for subsection, and 3 for subsubsection.
- Paragraph headings are recognized by a double underscore before and after the heading.
- The abstract of a document starts with *Abstract* as paragraph heading, and all text up to the next heading makes up the abstract,
- Blocks of computer code can easily be included by placing !bc (begin code) and !ec (end code) commands at separate lines before and after the code block.
- Blocks of computer code can also be imported from source files.
- Blocks of LaTeX mathematics can easily be included by placing !bt (begin TeX) and !et (end TeX) commands at separate lines before and after the math block.
- There is support for both LaTeX and text-like inline mathematics.
- Figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported.
- Invisible comments in the output format can be inserted throughout the text.
- Visible comments can be inserted so that authors and readers can comment upon the text (and at any time turn on/off output of such comments).
- There is an exercise environment with many advanced features.
- With a preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text.
- With Mako one can also have Python code embedded in the Doconce document and thereby parameterize the text (e.g., one text can describe programming in two languages).

Here is an example of some simple text written in the Doconce format:

```
==== A Subsection with Sample Text =====
label{my:first:sec}
Ordinary text looks like ordinary text, and the tags used for
_boldface_ words, *emphasized* words, and 'computer' words look
natural in plain text. Lists are typeset as you would do in email,
 * item 1
  * item 2
  * item 3
Lists can also have automatically numbered items instead of bullets,
 o item 1
 o item 2
 o item 3
URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl".
If the word is URL, the URL itself becomes the link name,
as in "URL": "tutorial.do.txt".
References to sections may use logical names as labels (e.g., a
"label" command right after the section title), as in the reference to
Section ref{my:first:sec}.
Doconce also allows inline comments of the form [name: comment] (with
a space after 'name:'), e.g., such as [hpl: here I will make some
remarks to the text]. Inline comments can be removed from the output
by a command-line argument (see Section ref{doconce2formats} for an
example).
Tables are also supperted, e.g.,
  |time | velocity | acceleration |
  | 0.0 | 1.4186 | -5.01
  | 2.0 | 1.376512 | 11.919
  | 4.0 | 1.1E+1 | 14.717624
```

# lines beginning with # are comment lines

The Doconce text above results in the following little document:

# 2.1 A Subsection with Sample Text

Ordinary text looks like ordinary text, and the tags used for **boldface** words, *emphasized* words, and computer words look natural in plain text. Lists are typeset as you would do in an email,

- item 1
- item 2
- item 3

Lists can also have numbered items instead of bullets, just use an o (for ordered) instead of the asterisk:

- 1. item 1
- 2. item 2
- 3. item 3

URLs with a link word are possible, as in hpl. If the word is URL, the URL itself becomes the link name, as in tutorial.do.txt.

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to the section A Subsection with Sample Text.

Doconce also allows inline comments such as (**hpl**: here I will make some remarks to the text) for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see the section *From Doconce to Other Formats* for an example).

Tables are also supperted, e.g.,

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

# 2.2 Mathematics and Computer Code

Inline mathematics, such as  $\nu = \sin(x)$ , allows the formula to be specified both as LaTeX and as plain text. This results in a professional LaTeX typesetting, but in other formats the text version normally looks better than raw LaTeX mathematics with backslashes. An inline formula like  $\nu = \sin(x)$  is typeset as

```
\alpha = \sin(x) = \sin(x)
```

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula. If you write a lot of mathematics, only the output formats latex, pdflatex, html, sphinx, and pandoc are of interest and all these support inline LaTeX mathematics so then you will naturally drop the pipe symbol and write just

```
nu = \sin(x)
```

However, if you want more textual formats, like plain text or reStructuredText, the text after the pipe symbol may help to make the math formula more readable if there are backslahes or other special LaTeX symbols in the LaTeX code.

Blocks of mathematics are typeset with raw LaTeX, inside !bt and !et (begin TeX, end TeX) instructions:

```
!bt
\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f, label{myeq1}\\
{\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
!et
```

The result looks like this:

$$\frac{\partial u}{\partial t} = \nabla^2 u + f,\tag{2.1}$$

$$\frac{\partial v}{\partial t} = \nabla \cdot (q(u)\nabla v) + g$$

Of course, such blocks only looks nice in formats with support for LaTeX mathematics, and here the align environment in particular (this includes latex, pdflatex, html, and sphinx). The raw LaTeX syntax appears in simpler formats, but can still be useful for those who can read LaTeX syntax.

You can have blocks of computer code, starting and ending with !bc and !ec instructions, respectively.

```
!bc pycod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec

Such blocks are formatted as
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
```

A code block must come after some plain sentence (at least for successful output to sphinx, rst, and ASCII-close formats), not directly after a section/paragraph heading or a table.

One can also copy computer code directly from files, either the complete file or specified parts. Computer code is then never duplicated in the documentation (important for the principle of avoiding copying information!).

Another document can be included by writing # #include "mynote.do.txt" at the beginning of a line. Doconce documents have extension do.txt. The do part stands for doconce, while the trailing .txt denotes a text document so that editors gives you plain text editing capabilities.

# 2.3 Macros (Newcommands), Cross-References, Index, and Bibliography

Doconce supports a type of macros via a LaTeX-style newcommand construction. The newcommands defined in a file with name newcommand\_replace.tex are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names newcommands.tex and newcommands\_keep.tex are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by !bt and !et in newcommands\_keep.tex to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used elsewhere throughout the text will usually be placed in newcommands\_replace.tex and expanded by Doconce. The definitions of newcommands in the newcommands\*.tex files must appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the doc/manual/manual.do.txt file (see the demo page for various formats of this document).

# FROM DOCONCE TO OTHER FORMATS

Transformation of a Doconce document mydoc.do.txt to various other formats applies the script doconce format:

```
Terminal> doconce format format mydoc.do.txt or just
```

Terminal> doconce format format mydoc

# 3.1 Preprocessing

The preprocess and make programs are used to preprocess the file, and options to preprocess and/or make can be added after the filename. For example,

```
Terminal> doconce format latex mydoc -Dextra_sections -DVAR1=5  # preprocess Terminal> doconce format latex yourdoc extra_sections=True VAR1=5  # mako
```

The variable FORMAT is always defined as the current format when running preprocess or make. That is, in the last example, FORMAT is defined as latex. Inside the Doconce document one can then perform format specific actions through tests like #if FORMAT == "latex" (for preprocess) or % if FORMAT == "latex": (for make).

### 3.2 Removal of inline comments

The command-line arguments —no-preprocess and —no-make turn off running preprocess and make, respectively.

Inline comments in the text are removed from the output by

```
Terminal> doconce format latex mydoc --skip_inline_comments
```

One can also remove all such comments from the original Doconce file by running:

```
Terminal> doconce remove_inline_comments mydoc
```

This action is convenient when a Doconce document reaches its final form and comments by different authors should be removed.

#### **3.3 HTML**

Making an HTML version of a Doconce file mydoc.do.txt is performed by

Terminal> doconce format html mydoc

The resulting file mydoc.html can be loaded into any web browser for viewing.

The HTML style can be defined either in the header of the HTML file or in an external CSS file. The latter is enabled by the command-line argument --css=filename. There is a default style with blue headings, and a style with the solarized color palette, specified by the --html-solarized command line argument. If there is no file with name filename in the --css=filename specification, the blue or solarized styles are written to filename and linked from the HTML document. You can provide your own style sheet either by replacing the content inside the style tags or by specifying a CSS file through the --css=filename option.

If the Pygments package (including the pygmentize program) is installed, code blocks are typeset with aid of this package. The command-line argument --no-pygments-html turns off the use of Pygments and makes code blocks appear with plain (pre) HTML tags. The option --pygments-html-linenos turns on line numbers in Pygments-formatted code blocks.

The HTML file can be embedded in a template if the Doconce document does not have a title (because then there will be no header and footer in the HTML file). The template file must contain valid HTML code and can have three "slots": % (title) s for a title, % (date) s for a date, and % (main) s for the main body of text, i.e., the Doconce document translated to HTML. The title becomes the first heading in the Doconce document, and the date is extracted from the DATE: line, if present. With the template feature one can easily embed the text in the look and feel of a website. The template can be extracted from the source code of a page at the site; just insert % (title) s and % (date) s at appropriate places and replace the main bod of text by % (main) s. Here is an example:

Terminal> doconce format html mydoc --html-template=mytemplate.html

# 3.4 Blogs

Doconce can be used for writing blogs provided the blog site accepts raw HTML code. Google's Blogger service (blogname.blogspot.com) is particularly well suited since it also allows extensive LaTeX mathematics via MathJax. Write the blog text as a Doconce document without any title, author, and date. Then generate HTML as described above. Copy the text and paste it into the text area in the blog, making sure the input format is HTML. On Google's Blogger service you can use Doconce to generate blogs with LaTeX mathematics and pretty (pygmentized) blocks of computer code. See a blog example for details on blogging.

**Warning:** In the comments after the blog one cannot paste raw HTML code with MathJax scripts so there is no support for mathematics in the comments.

WordPress (wordpress.com) allows raw HTML code in blogs, but has very limited LaTeX support, basically only formulas. The --wordpress option to doconce modifies the HTML code such that all equations are typeset in a way that is acceptable to WordPress. There is a doconce example on blogging with mathematics and code on WordPress.

### 3.5 Pandoc and Markdown

Output in Pandoc's extended Markdown format results from

```
Terminal> doconce format pandoc mydoc
```

The name of the output file is mydoc.mkd. From this format one can go to numerous other formats:

```
Terminal> pandoc -R -t mediawiki -o mydoc.mwk --toc mydoc.mkd
```

Pandoc supports latex, html, odt (OpenOffice), docx (Microsoft Word), rtf, texinfo, to mention some. The -R option makes Pandoc pass raw HTML or LaTeX to the output format instead of ignoring it, while the --toc option generates a table of contents. See the Pandoc documentation for the many features of the pandoc program.

Pandoc is useful to go from LaTeX mathematics to, e.g., HTML or MS Word. There are two ways (experiment to find the best one for your document): doconce format pandoc and then translating using pandoc, or doconce format latex, and then going from LaTeX to the desired format using pandoc. Here is an example on the latter strategy:

```
Terminal> doconce format latex mydoc
Terminal> doconce ptex2tex mydoc
Terminal> doconce replace '\Verb!' '\verb!' mydoc.tex
Terminal> pandoc -f latex -t docx -o mydoc.docx mydoc.tex
```

When we go through pandoc, only single equations or align\* environments are well understood.

Note that Doconce applies the Verb macro from the fancyvrb package while pandoc only supports the standard verb construction for inline verbatim text. Moreover, quite some additional doconce replace and doconce subst edits might be needed on the .mkd or .tex files to successfully have mathematics that is well translated to MS Word. Also when going to reStructuredText using Pandoc, it can be advantageous to go via LaTeX.

Here is an example where we take a Doconce snippet (without title, author, and date), maybe with some unnumbered equations, and quickly generate HTML with mathematics displayed my MathJax:

```
Terminal> doconce format pandoc mydoc
Terminal> pandoc -t html -o mydoc.html -s --mathjax mydoc.mkd
```

The -s option adds a proper header and footer to the mydoc.html file. This recipe is a quick way of makeing HTML notes with (some) mathematics.

#### 3.6 LaTeX

Making a LaTeX file mydoc.tex from mydoc.do.txt is done in two steps: .. Note: putting code blocks inside a list is not successful in many

Step 1. Filter the doconce text to a pre-LaTeX form mydoc.p.tex for the ptex2tex program (or doconce ptex2tex):

```
Terminal> doconce format latex mydoc
```

LaTeX-specific commands ("newcommands") in math formulas and similar can be placed in files newcommands.tex, newcommands\_keep.tex, or newcommands\_replace.tex (see the section *Macros* (*Newcommands*), *Cross-References*, *Index*, *and Bibliography*). If these files are present, they are included in the LaTeX document so that your commands are defined.

An option ——latex—printed makes some adjustments for documents aimed at being printed. For example, links to web resources are associated with a footnote listing the complete web address (URL).

Step 2. Run ptex2tex (if you have it) to make a standard LaTeX file,

```
Terminal> ptex2tex mydoc
```

3.6. LaTeX 11

In case you do not have ptex2tex, you may run a (very) simplified version:

```
Terminal> doconce ptex2tex mydoc
```

Note that Doconce generates a .p.tex file with some preprocessor macros that can be used to steer certain properties of the LaTeX document. For example, to turn on the Helvetica font instead of the standard Computer Modern font, run

```
Terminal> ptex2tex -DHELVETICA mydoc
Terminal> doconce ptex2tex mydoc -DHELVETICA # alternative
```

The title, authors, and date are by default typeset in a non-standard way to enable a nicer treatment of multiple authors having institutions in common. However, the standard LaTeX "maketitle" heading is also available through <code>-DLATEX\_HEADING=traditional</code>. A separate titlepage can be generate by <code>-DLATEX\_HEADING=titlepage</code>.

Preprocessor variables to be defined or undefined are

- BOOK for the "book" documentclass rather than the standard "article" class (necessary if you apply chapter headings)
- PALATINO for the Palatino font
- HELVETIA for the Helvetica font
- A4PAPER for A4 paper size
- A6PAPER for A6 paper size (suitable for reading on small devices)
- MOVIE15 for using the movie15 LaTeX package to display movies
- PREAMBLE to turn the LaTeX preamble on or off (i.e., complete document or document to be included elsewhere)
- MINTED for inclusion of the minted package (which requires latex or pdflatex to be run with the -shell-escape option)

The ptex2tex tool makes it possible to easily switch between many different fancy formattings of computer or verbatim code in LaTeX documents. After any !bc command in the Doconce source you can insert verbatim block styles as defined in your .ptex2tex.cfg file, e.g., !bc sys for a terminal session, where sys is set to a certain environment in .ptex2tex.cfg (e.g., CodeTerminal). There are about 40 styles to choose from, and you can easily add new ones.

Also the doconce ptex2tex command supports preprocessor directives for processing the .p.tex file. The command allows specifications of code environments as well. Here is an example:

```
Terminal> doconce ptex2tex mydoc -DLATEX_HEADING=traditional \
    -DPALATINO -DA6PAPER \
    "sys=\begin{quote}\begin{verbatim}@\end{verbatim}\end{quote}" \
    fpro=minted fcod=minted shcod=Verbatim envir=ans:nt
```

Note that @ must be used to separate the begin and end LaTeX commands, unless only the environment name is given (such as minted above, which implies \begin{minted} {fortran} and \end{minted} as begin and end for blocks inside !bc fpro and !ec). Specifying envir=ans:nt means that all other environments are typeset with the anslistings.sty package, e.g., !bc cppcod will then result in \begin{c++}. If no environments like sys, fpro, or the common envir are defined on the command line, the plain \begin{verbatim} and \end{verbatim} used.

Step 2b (optional). Edit the mydoc.tex file to your needs. For example, you may want to substitute section by section\* to avoid numbering of sections, you may want to insert linebreaks (and perhaps space) in the title, etc. This can be automatically edited with the aid of the doconce replace and doconce subst commands. The

former works with substituting text directly, while the latter performs substitutions using regular expressions. Here are two examples:

A lot of tailored fixes to the LaTeX document can be done by an appropriate set of text replacements and regular expression substitutions. You are anyway encourged to make a script for generating PDF from the LaTeX file.

Step 3. Compile mydoc.tex and create the PDF file:

```
Terminal> latex mydoc
Terminal> latex mydoc
Terminal> makeindex mydoc  # if index
Terminal> bibitem mydoc  # if bibliography
Terminal> latex mydoc
Terminal> dvipdf mydoc
```

If one wishes to run ptex2tex and use the minted LaTeX package for typesetting code blocks (Minted\_Python, Minted\_Cpp, etc., in ptex2tex specified through the \*pro and \*cod variables in .ptex2tex.cfg or \$HOME/.ptex2tex.cfg), the minted LaTeX package is needed. This package is included by running ptex2tex with the -DMINTED option:

```
Terminal> ptex2tex -DMINTED mydoc
```

In this case, latex must be run with the -shell-escape option:

```
Terminal> latex -shell-escape mydoc
Terminal> latex -shell-escape mydoc
Terminal> makeindex mydoc  # if index
Terminal> bibitem mydoc  # if bibliography
Terminal> latex -shell-escape mydoc
Terminal> dvipdf mydoc
```

When running doconce ptex2tex mydoc envir=minted (or other minted specifications with doconce ptex2tex), the minted package is automatically included so there is no need for the -DMINTED option.

#### 3.7 PDFLaTeX

Running pdflatex instead of latex follows almost the same steps, but the start is

```
Terminal> doconce format latex mydoc
```

Then ptex2tex is run as explained above, and finally

```
Terminal> pdflatex -shell-escape mydoc
Terminal> makeindex mydoc  # if index
Terminal> bibitem mydoc  # if bibliography
Terminal> pdflatex -shell-escape mydoc
```

### 3.8 Plain ASCII Text

We can go from Doconce "back to" plain untagged text suitable for viewing in terminal windows, inclusion in email text, or for insertion in computer source code:

3.7. PDFLaTeX

```
Terminal> doconce format plain mydoc.do.txt # results in mydoc.txt
```

#### 3.9 reStructuredText

Going from Doconce to reStructuredText gives a lot of possibilities to go to other formats. First we filter the Doconce text to a reStructuredText file mydoc.rst:

```
Terminal> doconce format rst mydoc.do.txt
```

We may now produce various other formats:

```
Terminal> rst2html.py mydoc.rst > mydoc.html # html
Terminal> rst2latex.py mydoc.rst > mydoc.tex # latex
Terminal> rst2xml.py mydoc.rst > mydoc.xml # XML
Terminal> rst2odt.py mydoc.rst > mydoc.odt # OpenOffice
```

The OpenOffice file mydoc.odt can be loaded into OpenOffice and saved in, among other things, the RTF format or the Microsoft Word format. However, it is more convenient to use the program unovonv to convert between the many formats OpenOffice supports on the command line. Run

```
Terminal> unoconv --show
```

to see all the formats that are supported. For example, the following commands take mydoc.odt to Microsoft Office Open XML format, classic MS Word format, and PDF:

```
Terminal> unoconv -f ooxml mydoc.odt
Terminal> unoconv -f doc mydoc.odt
Terminal> unoconv -f pdf mydoc.odt
```

Remark about Mathematical Typesetting. At the time of this writing, there is no easy way to go from Doconce and LaTeX mathematics to reST and further to OpenOffice and the "MS Word world". Mathematics is only fully supported by latex as output and to a wide extent also supported by the sphinx output format. Some links for going from LaTeX to Word are listed below.

- http://ubuntuforums.org/showthread.php?t=1033441
- http://tug.org/utilities/texconv/textopc.html
- http://nileshbansal.blogspot.com/2007/12/latex-to-openofficeword.html

# 3.10 Sphinx

Sphinx documents demand quite some steps in their creation. We have automated most of the steps through the doconce sphinx dir command:

The keywords author, title, and version are used in the headings of the Sphinx document. By default, version is 1.0 and the script will try to deduce authors and title from the doconce files file1, file2, etc. that together represent the whole document. Note that none of the individual Doconce files file1, file2, etc. should include the rest as their union makes up the whole document. The default value of dirname is sphinx-rootdir. The theme keyword is used to set the theme for design of HTML output from Sphinx (the default theme is 'default').

With a single-file document in mydoc.do.txt one often just runs

```
Terminal> doconce sphinx_dir mydoc
```

and then an appropriate Sphinx directory sphinx-rootdir is made with relevant files.

The doconce sphinx\_dir command generates a script automake\_sphinx.py for compiling the Sphinx document into an HTML document. One can either run automake\_sphinx.py or perform the steps in the script manually, possibly with necessary modifications. You should at least read the script prior to executing it to have some idea of what is done.

The doconce sphinx\_dir script copies directories named figs or figures over to the Sphinx directory so that figures are accessible in the Sphinx compilation. If figures or movies are located in other directories, automake\_sphinx.py must be edited accordingly. Files, to which there are local links (not http: or file: URLs), must be placed in the \_static subdirectory of the Sphinx directory. The utility doconce sphinxfix\_localURLs is run to check for local links in the Doconce file: for each such link, say dir1/dir2/myfile.txt it replaces the link by \_static/myfile.txt and copies dir1/dir2/myfile.txt to a local \_static directory (in the same directory as the script is run). However, we recommend instead that the writer of the document places files in \_static or lets a script do it automatically. The user must copy all \_static/\* files to the \_static subdirectory of the Sphinx directory. It may be wise to always put files, to which there are local links in the Doconce document, in a \_static or \_static-name directory and use these local links. Then links do not need to be modified when creating a Sphinx version of the document.

Doconce comes with a collection of HTML themes for Sphinx documents. These are packed out in the Sphinx directory, the <code>conf.py</code> configuration file for Sphinx is edited accordingly, and a script <code>make-themes.sh</code> can make HTML documents with one or more themes. For example, to realize the themes <code>fenics</code> and <code>pyramid</code>, one writes

```
Terminal> ./make-themes.sh fenics pyramid
```

The resulting directories with HTML documents are \_build/html\_fenics and \_build/html\_pyramid, respectively. Without arguments, make-themes.sh makes all available themes (!).

If it is not desirable to use the autogenerated scripts explained above, here is the complete manual procedure of generating a Sphinx document from a file mydoc.do.txt.

Step 1. Translate Doconce into the Sphinx format:

```
Terminal> doconce format sphinx mydoc
```

Step 2. Create a Sphinx root directory either manually or by using the interactive sphinx-quickstart program. Here is a scripted version of the steps with the latter:

```
mkdir sphinx-rootdir
sphinx-quickstart <<EOF
sphinx-rootdir
n
_
Name of My Sphinx Document
Author
version
version
.rst
index
n
y
n
n
n</pre>
```

3.10. Sphinx 15

```
y
n
n
y
y
y
EOF
```

The autogenerated <code>conf.py</code> file may need some edits if you want to specific layout (Sphinx themes) of HTML pages. The <code>doconce sphinx\_dir</code> generator makes an extended <code>conv.py</code> file where, among other things, several useful Sphinx extensions are included.

*Step 3.* Copy the mydoc.rst file to the Sphinx root directory:

```
Terminal > cp mydoc.rst sphinx-rootdir
```

If you have figures in your document, the relative paths to those will be invalid when you work with mydoc.rst in the sphinx-rootdir directory. Either edit mydoc.rst so that figure file paths are correct, or simply copy your figure directories to sphinx-rootdir. Links to local files in mydoc.rst must be modified to links to files in the \_static directory, see comment above.

Step 4. Edit the generated index.rst file so that mydoc.rst is included, i.e., add mydoc to the toctree section so that it becomes

```
.. toctree::
:maxdepth: 2
mydoc
```

(The spaces before mydoc are important!)

Step 5. Generate, for instance, an HTML version of the Sphinx source:

```
make clean # remove old versions
make html
```

Sphinx can generate a range of different formats: standalone HTML, HTML in separate directories with index.html files, a large single HTML file, JSON files, various help files (the qthelp, HTML, and Devhelp projects), epub, LaTeX, PDF (via LaTeX), pure text, man pages, and Texinfo files.

Step 6. View the result:

```
Terminal> firefox _build/html/index.html
```

Note that verbatim code blocks can be typeset in a variety of ways depending the argument that follows !bc: cod gives Python (code-block:: python in Sphinx syntax) and cppcod gives C++, but all such arguments can be customized both for Sphinx and LaTeX output.

#### 3.11 Wiki Formats

There are many different wiki formats, but Doconce only supports three: Googlecode wiki, MediaWiki, and Creole Wiki. These formats are called gwiki, mwiki, and cwiki, respectively. Transformation from Doconce to these formats is done by

```
Terminal> doconce format gwiki mydoc.do.txt
Terminal> doconce format mwiki mydoc.do.txt
Terminal> doconce format cwiki mydoc.do.txt
```

The produced MediaWiki can be tested in the sandbox of wikibooks.org. The format works well with Wikipedia, Wikibooks, and ShoutWiki, but not always well elsewhere (see this example).

Large MediaWiki documents can be made with the Book creator. From the MediaWiki format one can go to other formats with aid of mwlib. This means that one can easily use Doconce to write Wikibooks and publish these in PDF and MediaWiki format, while at the same time, the book can also be published as a standard LaTeX book, a Sphinx web document, or a collection of HTML files.

The Googlecode wiki document, mydoc.gwiki, is most conveniently stored in a directory which is a clone of the wiki part of the Googlecode project. This is far easier than copying and pasting the entire text into the wiki editor in a web browser.

When the Doconce file contains figures, each figure filename must in the .gwiki file be replaced by a URL where the figure is available. There are instructions in the file for doing this. Usually, one performs this substitution automatically (see next section).

## 3.12 Tweaking the Doconce Output

Occasionally, one would like to tweak the output in a certain format from Doconce. One example is figure filenames when transforming Doconce to reStructuredText. Since Doconce does not know if the .rst file is going to be filtered to LaTeX or HTML, it cannot know if .eps or .png is the most appropriate image filename. The solution is to use a text substitution command or code with, e.g., sed, perl, python, or scitools subst, to automatically edit the output file from Doconce. It is then wise to run Doconce and the editing commands from a script to automate all steps in going from Doconce to the final format(s). The make.sh files in docs/manual and docs/tutorial constitute comprehensive examples on how such scripts can be made.

## **3.13 Demos**

The current text is generated from a Doconce format stored in the file

docs/tutorial/tutorial.do.txt

The file make.sh in the tutorial directory of the Doconce source code contains a demo of how to produce a variety of formats. The source of this tutorial, tutorial.do.txt is the starting point. Running make.sh and studying the various generated files and comparing them with the original tutorial.do.txt file, gives a quick introduction to how Doconce is used in a real case. Here is a sample of how this tutorial looks in different formats.

There is another demo in the docs/manual directory which translates the more comprehensive documentation, manual.do.txt, to various formats. The make.sh script runs a set of translations.

| Doconce: Document Once, Include Anywhere Documentation, Release 1.0 |  |
|---------------------------------------------------------------------|--|
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |
|                                                                     |  |

**CHAPTER** 

**FOUR** 

## INSTALLATION OF DOCONCE AND ITS DEPENDENCIES

#### 4.1 Doconce

Doconce itself is pure Python code hosted at http://code.google.com/p/doconce. Its installation from the Mercurial (hg) source follows the standard procedure:

```
# Doconce
hg clone https://doconce.googlecode.com/hg/ doconce
cd doconce
sudo python setup.py install
cd ..
```

Since Doconce is frequently updated, it is recommended to use the above procedure and whenever a problem occurs, make sure to update to the most recent version:

```
cd doconce
hg pull
hg update
sudo python setup.py install
```

#### Debian GNU/Linux users can also run

```
sudo apt-get install doconce
```

This installs the latest release and not the most updated and bugfixed version. On Ubuntu one needs to run

```
sudo add-apt-repository ppa:scitools/ppa
sudo apt-get update
sudo apt-get install doconce
```

#### 4.2 Dependencies

#### 4.2.1 Preprocessors

If you make use of the Preprocess preprocessor, this program must be installed:

```
svn checkout http://preprocess.googlecode.com/svn/trunk/ preprocess
cd preprocess
cd doconce
```

```
sudo python setup.py install
cd ..
```

A much more advanced alternative to Preprocess is Mako. Its installation is most conveniently done by pip,

```
pip install Mako
```

This command requires pip to be installed. On Debian Linux systems, such as Ubuntu, the installation is simply done by

```
sudo apt-get install python-pip
```

Alternatively, one can install from the pip source code.

Mako can also be installed directly from source: download the tarball, pack it out, go to the directory and run the usual sudo python setup.py install.

#### 4.2.2 Image file handling

Different output formats require different formats of image files. For example, PostScript or Encapuslated PostScript is required for latex output, while HTML needs JPEG, GIF, or PNG formats. Doconce calls up programs from the ImageMagick suite for converting image files to a proper format if needed. The ImageMagick suite can be installed on all major platforms. On Debian Linux (including Ubuntu) systems one can simply write

```
sudo apt-get install imagemagick
```

The convenience program doconce combine\_images, for combining several images into one, will use montage and convert from ImageMagick and the pdftk, pdfnup, and pdfcrop programs from the texlive-extra-utils Debian package. The latter gets installed by

```
sudo apt-get install texlive-extra-utils
```

#### 4.2.3 Spellcheck

The utility doconce spellcheck applies the ispell program for spellcheck. On Debian (including Ubuntu) it is installed by

```
sudo apt-get install ispell
```

#### 4.2.4 Ptex2tex for LaTeX Output

To make LaTeX documents with very flexible choice of typesetting of verbatim code blocks you need ptex2tex, which is installed by

```
svn checkout http://ptex2tex.googlecode.com/svn/trunk/ ptex2tex
cd ptex2tex
sudo python setup.py install
```

It may happen that you need additional style files, you can run a script, cp2texmf.sh:

```
cd latex
sh cp2texmf.sh # copy stylefiles to ~/texmf directory
cd ../..
```

This script copies some special stylefiles that that ptex2tex potentially makes use of. Some more standard stylefiles are also needed. These are installed by

```
sudo apt-get install texlive-latex-recommended texlive-latex-extra
```

on Debian Linux (including Ubuntu) systems. TeXShop on Mac comes with the necessary stylefiles (if not, they can be found by googling and installed manually in the  $\sim/\text{texmf/tex/latex/misc}$  directory).

Note that the doconce ptex2tex command, which needs no installation beyond Doconce itself, can be used as a simpler alternative to the ptex2tex program.

The *minted* LaTeX style is offered by ptex2tex and doconce ptext2tex is popular among many users. This style requires the package Pygments to be installed. On Debian Linux,

```
sudo apt-get install python-pygments
```

Alternatively, the package can be installed manually:

```
hg clone ssh://hg@bitbucket.org/birkenfeld/pygments-main pygments cd pygments sudo python setup.py install
```

If you use the minted style together with ptex2tex, you have to enable it by the -DMINTED command-line argument to ptex2tex. This is not necessary if you run the alternative doconce ptex2tex program.

All use of the minted style requires the -shell-escape command-line argument when running LaTeX, i.e., latex -shell-escape or pdflatex -shell-escape.

#### 4.2.5 reStructuredText (reST) Output

The rst output from Doconce allows further transformation to LaTeX, HTML, XML, OpenOffice, and so on, through the docutils package. The installation of the most recent version can be done by

```
svn checkout http://docutils.svn.sourceforge.net/svnroot/docutils/trunk/docutils
cd docutils
sudo python setup.py install
cd ..
```

To use the OpenOffice suite you will typically on Debian systems install

```
sudo apt-get install unovonv libreoffice libreoffice-dmaths
```

There is a possibility to create PDF files from reST documents using ReportLab instead of LaTeX. The enabling software is rst2pdf. Either download the tarball or clone the svn repository, go to the rst2pdf directory and run the usual sudo python setup.py install.

Output to sphinx requires of course the Sphinx software, installed by

```
hg clone https://bitbucket.org/birkenfeld/sphinx
cd sphinx
sudo python setup.py install
cd ..
```

#### 4.2.6 Markdown and Pandoc Output

The Doconce format pandoc outputs the document in the Pandoc extended Markdown format, which via the pandoc program can be translated to a range of other formats. Installation of Pandoc, written in Haskell, is most easily done by

4.2. Dependencies 21

```
sudo apt-get install pandoc on Debian (Ubuntu) systems.
```

#### 4.2.7 Epydoc Output

When the output format is epydoc one needs that program too, installed by

```
svn co https://epydoc.svn.sourceforge.net/svnroot/epydoc/trunk/epydoc epydoc
cd epydoc
sudo make install
cd ..
```

Remark. Several of the packages above installed from source code are also available in Debian-based system through the apt-get install command. However, we recommend installation directly from the version control system repository as there might be important updates and bug fixes. For svn directories, go to the directory, run svn update, and then sudo python setup.py install. For Mercurial (hg) directories, go to the directory, run hg pull; hg update, and then sudo python setup.py install.

**CHAPTER** 

**FIVE** 

### **INDICES AND TABLES**

- genindex
- modindex
- search

```
tutorial.xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE document PUBLIC "+//IDN docutils.sourceforge.net//DTD Docutils Generic</pre>
//EN//XML" "http://docutils.sourceforge.net/docs/ref/docutils.dtd">
<!-- Generated by Docutils 0.9 -->
<document source="tutorial.rst"><comment xml:space="preserve">Automatically gene
rated reST file from Doconce source
(http://code.google.com/p/doconce/)</comment><comment xml:space="preserve">Missi
ng: FIGURE, MOVIE, environments</comment><section ids="doconce-document-once-inc
lude-anywhere" names="doconce:\ document\ once,\ include\ anywhere"><title>Docon
ce: Document Once, Include Anywhere</title><field_list><field><field_name>Author
</field_name><field_body><paragraph>Hans Petter Langtangen</paragraph></field_bo
dy></field><field><field_name>Date</field_name><field_body><paragraph>Jan 16, 20
13</paragraph><bullet_list bullet="*"><list_item><paragraph>When writing a note,
report, manual, etc., do you find it difficult
to choose the typesetting format? That is, to choose between plain
(email-like) text, wiki, Word/OpenOffice, LaTeX, HTML,
reStructuredText, Sphinx, XML, etc. Would it be convenient to
start with some very simple text-like format that easily converts
to the formats listed above, and then at some later stage
eventually go with a particular format?</paragraph></list_item><list_item><parag
raph>Do you need to write documents in varying formats but find it
difficult to remember all the typesetting details of various
formats like <reference name="LaTeX" refuri="http://refcards.com/docs/silvermanj
/amslatex/LaTeXRefCard.v2.0.pdf">LaTeX</reference><target ids="latex" names="lat
ex" refuri="http://refcards.com/docs/silvermanj/amslatex/LaTeXRefCard.v2.0.pdf"/
>, <reference name="HTML" refuri="http://www.htmlcodetutorial.com/">HTML</refere
nce><target ids="html" names="html" refuri="http://www.htmlcodetutorial.com/"/>,
 <reference name="reStructuredText" refuri="http://docutils.sourceforge.net/docs</pre>
/ref/rst/restructuredtext.html">reStructuredText</reference><target ids="restruc
turedtext" names="restructuredtext" refuri="http://docutils.sourceforge.net/docs
/ref/rst/restructuredtext.html"/>, <reference name="Sphinx" refuri="http://sphin</pre>
x.pocoo.org/contents.html">Sphinx</reference><target ids="sphinx" names="sphinx"
refuri="http://sphinx.pocoo.org/contents.html"/>, and <reference name="wiki" re
furi="http://code.google.com/p/support/wiki/WikiSyntax">wiki</reference><target</pre>
ids="wiki" names="wiki" refuri="http://code.google.com/p/support/wiki/WikiSyntax
"/>? Would it be convenient
to generate the typesetting details of a particular format from a
very simple text-like format with minimal tagging?</paragraph></list_item><list_
item><paragraph>Do you have the same information scattered around in different
documents in different typesetting formats? Would it be a good idea
to write things once, in one format, stored in one place, and
include it anywhere?</list_item></bullet_list></field_body></field>
/field_list><paragraph>If any of these questions are of interest, you should kee
p on reading.</paragraph></section><section ids="what-does-doconce-look-like" na
mes="what\ does\ doconce\ look\ like?"><title>What Does Doconce Look Like?</titl
e><paragraph>Doconce text looks like ordinary text, but there are some almost in
visible
text constructions that allow you to control the formating. Here are
som examples.<block_quote><bullet_list bullet="*"><list_item><paragr</pre>
aph>Bullet lists arise from lines starting with teral>*</literal>.</paragraph
></list_item><list_item><paragraph><emphasis>Emphasized words</emphasis> are sur
rounded by <literal>*feral>*/list_item>feralph><s</pre>
trong>Words in boldface</strong> are surrounded by underscores.</paragraph></lis
t_item><list_item><paragraph>Words from computer code are enclosed in back quote
s and
then typeset teral>verbatim (in a monospace font)
t_item><list_item><paragraph>Section headings are recognied by equality (<litera
l>=</literal>) signs before
and after the title, and the number of <literal>=</literal> signs indicates the
```

```
tutorial.xml
level of the section: 7 for main section, 5 for subsection, and
3 for subsubsection.</paragraph></list_item><list_item><paragraph>Paragraph head
ings are recognized by a double underscore
before and after the heading.</paragraph></list item><paragraph>The a
bstract of a document starts with <emphasis>Abstract</emphasis> as paragraph
heading, and all text up to the next heading makes up the abstract, </paragraph><
/list_item><list_item><paragraph>Blocks of computer code can easily be included
by placing
<literal>!bc</literal> (begin code) and <literal>!ec</literal> (end code) comman
ds at separate lines
before and after the code block.</paragraph></list_item><paragraph>Bl
ocks of computer code can also be imported from source files.</paragraph></list_
item><list_item><paragraph>Blocks of LaTeX mathematics can easily be included by
placing
<literal>!bt</literal> (begin TeX) and <literal>!et</literal> (end TeX) commands
at separate lines
before and after the math block.</paragraph></list_item><paragraph>Th
ere is support for both LaTeX and text-like inline mathematics.</paragraph></lis
t_item><list_item><paragraph>Figures and movies with captions, simple tables,
URLs with links, index list, labels and references are supported.</paragraph></l
ist_item><list_item><paragraph>Invisible comments in the output format can be in
serted throughout
the text.</paragraph></list_item><list_item><paragraph>Visible comments can be i
nserted so that authors and readers can
comment upon the text (and at any time turn on/off output of such
comments).</paragraph></list_item><list_item><paragraph>There is an exercise env
ironment with many advanced features./list_item><list_item><paragra</pre>
ph>With a preprocessor, Preprocess or Mako, one can include
other documents (files) and large portions of text can be defined
in or out of the text.</paragraph></list_item><list_item><paragraph>With Mako on
e can also have Python code
embedded in the Doconce document and thereby parameterize the
text (e.g., one text can describe programming in two languages).</paragraph>
st_item></bullet_list></block_quote><paragraph>Here is an example of some simple
text written in the Doconce format:</paragraph><literal_block xml:space="preser
ve">==== A Subsection with Sample Text =====
label{my:first:sec}
Ordinary text looks like ordinary text, and the tags used for
boldface words, *emphasized* words, and 'computer' words look
natural in plain text. Lists are typeset as you would do in email,
  * item 1
  * item 2
  * item 3
Lists can also have automatically numbered items instead of bullets,
 o item 1
  o item 2
  o item 3
URLs with a link word are possible, as in "hpl": "http://folk.uio
.no/hpl".
If the word is URL, the URL itself becomes the link name,
as in " URL": " tutorial.do.txt".
References to sections may use logical names as labels (e.g., a
```

" label" command right after the section title), as in the reference to

Section ref{my:first:sec}.

Doconce also allows inline comments of the form [name: comment] (with a space after 'name:'), e.g., such as [hpl: here I will make some remarks to the text]. Inline comments can be removed from the output by a command-line argument (see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

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

# lines beginning with # are comment lines</literal\_block><paragraph>The Doconce
 text above results in the following little document:</paragraph><target refid="
 my-first-sec"/><section ids="a-subsection-with-sample-text my-first-sec" names="
 a\ subsection\ with\ sample\ text my:first:sec"><title>A Subsection with Sample
 Text</title><paragraph>Ordinary text looks like ordinary text, and the tags used
 for

<strong>boldface</strong> words, <emphasis>emphasized</emphasis> words, and <lit
eral>computer</literal> words look

natural in plain text. Lists are typeset as you would do in an email,</paragrap
h><block\_quote><bullet\_list bullet="\*"><list\_item><paragraph>item 1</paragraph><
/list\_item><list\_item><paragraph>item 2</paragraph></list\_item><list\_item><paragraph>item 3</paragraph></list\_item></bullet\_list></block\_quote><paragraph>Lists
can also have numbered items instead of bullets, just use an literal>o

(for ordered) instead of the asterisk:</paragraph><block\_quote><enumerated\_list enumtype="arabic" prefix="" suffix="."><list\_item><paragraph>item 1</paragraph></list\_item><list\_item><paragraph></list\_item><list\_item><paragraph></list\_item><list\_item><paragraph></list\_item><paragraph>UR Ls with a link word are possible, as in <reference name="hpl" refuri="http://folk.uio.no/hpl">hpl</reference><target ids="hpl" names="hpl" refuri="http://folk.uio.no/hpl"/>.

If the word is URL, the URL itself becomes the link name,

as in <reference name="tutorial.do.txt" refuri="tutorial.do.txt">tutorial.do.txt </reference><target ids="tutorial-do-txt" names="tutorial.do.txt" refuri="tutorial.do.txt"/>.</paragraph>References to sections may use logical names as labels (e.g., a

" label" command right after the section title), as in the reference to the section <reference name="A Subsection with Sample Text" refid="a-subsection-with-sample-text">A Subsection with Sample Text</reference>.</paragraph><paragraph>Doconce also allows inline comments such as (<strong>hpl</strong>: here I will make

some remarks to the text) for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see the section <reference name="From Doconce to Other Formats" refid="from-doc

once-to-other-formats">From Doconce to Other Formats</reference> for an example)
.</paragraph><paragraph>Tables are also supperted, e.g.,</paragraph><tgro
up cols="3"><colspec colwidth="12"/><colspec colwidth="12"/><colspec colwidth="1"
2"/><thead><row><entry><paragraph>time</paragraph></entry><entry><paragraph>velo
city</paragraph></entry><entry><paragraph>acceleration</paragraph></entry><paragraph>1
.4186</paragraph></entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry><entry

```
tutorial.xml
entry><paragraph>2.0</paragraph></entry><entry><paragraph>1.376512</paragraph></
entry><entry><paragraph>11.919</paragraph></entry></row><entry><paragraph>4
.0</paragraph></entry><entry><paragraph>1.1E+1</paragraph></entry><entry><paragr
aph>14.717624</paragraph></entry></row></section><secti
on ids="mathematics-and-computer-code" names="mathematics\ and\ computer\ code">
<title>Mathematics and Computer Code</title><paragraph>Inline mathematics, such
as v = \sin(x),
allows the formula to be specified both as LaTeX and as plain text.
This results in a professional LaTeX typesetting, but in other formats
the text version normally looks better than raw LaTeX mathematics with
backslashes. An inline formula like v = sin(x) is
typeset as:</paragraph><literal_block xml:space="preserve">\n = \sin(x) $ | $v =
sin(x)$</literal_block><paragraph>The pipe symbol acts as a delimiter between La
TeX code and the plain text
version of the formula. If you write a lot of mathematics, only the
output formats <literal>latex</literal>, <literal>pdflatex</literal>, <literal>h
tml</literal>, teral>sphinx</literal>, and teral>pandoc</literal>
are of interest
and all these support inline LaTeX mathematics so then you will naturally
drop the pipe symbol and write just:/paragraph>teral_block xml:space="preser"
ve">$\nu = \sin(x)$</literal_block><paragraph>However, if you want more textual
formats, like plain text or reStructuredText,
the text after the pipe symbol may help to make the math formula more readable
if there are backslahes or other special LaTeX symbols in the LaTeX code.</parag
raph><paragraph>Blocks of mathematics are typeset with raw LaTeX, inside
<literal>!bt</literal> and <literal>!et</literal> (begin TeX, end TeX) instructi
ons:</paragraph><literal_block xml:space="preserve">!bt
\begin{align}
 \partial u\over\partial t\ &= \nabla^2 u + f, label\{myeq1\}\\
{\partial v\over\partial t} & amp; = \nabla\cdot(q(u)\nabla v) + q
\end{align}
!et</literal_block><comment xml:space="preserve">Note: !bt and !et (and !bc and
!ec below) are used to illustrate</comment><comment xml:space="preserve">tex and
 code blocks in inside verbatim blocks and are replaced</comment><comment xml:sp</pre>
ace="preserve">by !bt, !et, !bc, and !ec after all other formatting is finished.
</comment><paragraph>The result looks like this:</paragraph><literal_block xml:s</pre>
pace="preserve">\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f, label{myeq1}\\
{\partial v\over\partial t} & amp; = \nabla\cdot(q(u)\nabla v) + g
\end{align}</literal_block><paragraph>Of course, such blocks only looks nice in
formats with support
for LaTeX mathematics, and here the align environment in particular
(this includes <literal>latex</literal>, <literal>pdflatex</literal>, <literal>h
tml</literal>, and teral>sphinx</literal>). The raw
LaTeX syntax appears in simpler formats, but can still be useful
for those who can read LaTeX syntax.</paragraph><paragraph>You can have blocks o
f computer code, starting and ending with
<literal>!bc</literal> and <literal>!ec</literal> instructions, respectively:
aragraph><literal_block xml:space="preserve">!bc pycod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)
import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec</literal_block><paragraph>Such blocks are formatted as:</paragraph><literal_</pre>
block xml:space="preserve">from math import sin, pi
```

def myfunc(x):

return sin(pi\*x)

```
tutorial.xml
import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)literal_block><paragraph>A code b
lock must come after some plain sentence (at least for successful
output to teral>sphinxliteral>, <literal>rst</literal>, and ASCII-close for
mats),
not directly after a section/paragraph heading or a table.</paragraph><paragraph
>One can also copy computer code directly from files, either the
complete file or specified parts. Computer code is then never
duplicated in the documentation (important for the principle of
avoiding copying information!).</paragraph><paragraph>Another document can be in
cluded by writing eral># #include "mynote.do.txt"
at the beginning of a line. Doconce documents have
extension cliteral>do.txt</literal>. The <literal>do</literal> part stands for d
oconce, while the
trailing teral>.txteliteral> denotes a text document so that editors gives y
plain text editing capabilities.</paragraph><target refid="newcommands"/></secti
on><section ids="macros-newcommands-cross-references-index-and-bibliography newc
ommands" names="macros\ (newcommands),\ cross-references,\ index,\ and\ bibliogr
aphy newcommands"><title>Macros (Newcommands), Cross-References, Index, and Bibl
iography</title><paragraph>Doconce supports a type of macros via a LaTeX-style <
emphasis>newcommand</emphasis>
               The newcommands defined in a file with name
construction.
<literal>newcommand_replace.tex</literal> are expanded when Doconce is filtered
other formats, except for LaTeX (since LaTeX performs the expansion
itself). Newcommands in files with names teral>newcommands.tex</literal> and
<literal>newcommands_keep.tex</literal> are kept unaltered when Doconce text is
filtered to other formats, except for the Sphinx format. Since Sphinx
understands LaTeX math, but not newcommands if the Sphinx output is
HTML, it makes most sense to expand all newcommands. Normally, a user
will put all newcommands that appear in math blocks surrounded by
<literal>!bt</literal> and <literal>!et</literal> in <literal>newcommands_keep.t
exexliteral> to keep them unchanged, at
least if they contribute to make the raw LaTeX math text easier to
read in the formats that cannot render LaTeX. Newcommands used
elsewhere throughout the text will usually be placed in
<literal>newcommands_replace.tex</literal> and expanded by Doconce.
                                                                       The definit
newcommands in the teral>newcommands*.texfiles <emphasis>must</emp
hasis> appear on a single
line (multi-line newcommands are too hard to parse with regular
expressions).</paragraph><paragraph>Recent versions of Doconce also offer cross
referencing, typically one
can define labels below (sub)sections, in figure captions, or in
equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx
formats. Citations to literature, with an accompanying bibliography in
a file, are also supported. The syntax of labels, references,
citations, and the bibliography closely resembles that of LaTeX,
making it easy for Doconce documents to be integrated in LaTeX
```

Example on including another Doconce file (using preprocess):</comment><target r

for various formats of this document).</paragraph><comment xml:space="preserve">

e" refuri="https://doconce.googlecode.com/hg/doc/demos/manual/index.html"/>

syntax we refer to the teral>doc/manual/manual.do.txt</literal> file (see the <reference name="demo page" refuri="https://doconce.googlecode.com/hg/doc/demos/</pre> manual/index.html">demo page</reference><target ids="demo-page" names="demo\ pag

projects (manuals, books). For further details on functionality and

efid="doconce2formats"/></section></section>ids="from-doconce-to-otherformats doconce2formats" names="from\ doconce\ to\ other\ formats doconce2format s"><title>From Doconce to Other Formats</title><paragraph>Transformation of a Do conce document literal>mydoc.do.txt formats applies the script teral>doconce format l block xml:space="preserve">Terminal&qt; doconce format format mydoc.do.txt teral\_block><paragraph>or just:</paragraph><literal\_block xml:space="preserve">T erminal> doconce format format mydoc</literal\_block><section ids="preprocessi"</pre> ng" names="preprocessing"><title>Preprocessing</title><paragraph>The teral>pr eprocess</literal> and teral>mako</literal> programs are used to preprocess t he file, and options to <literal>preprocessand/or <literal>mako/literal > can be added after the filename. For example:</paragraph>teral\_block xml:space="preserve">Terminal&g t; doconce format latex mydoc -Dextra\_sections -DVAR1=5 # preprocess Terminal> doconce format latex yourdoc extra\_sections=True VAR1=5 # mako</li teral\_block><paragraph>The variable <literal>FORMAT</literal> is always defined as the current format when running teral>preprocessliteral> or literal>makoliteral>. That is, in the e last example, teral>FORMAT</literal> is defined as <literal>latexlatexlatex</or> format specific actions through tests like teral>#if FORMAT == "latex&qu ot;</literal> (for <literal>preprocess</literal>) or <literal>% if FORMAT == &quot;latex&quot; :</literal> (for <literal>mako</literal>)./paragraph></section><section ids="re"</pre> moval-of-inline-comments" names="removal\ of\ inline\ comments"><title>Removal o f inline comments</title><comment xml:space="preserve">mention notes also</comme nt><paragraph>The command-line arguments teral>--no-preprocess</literal> and <literal>--no-mako</literal> turn off running teral>preprocess</literal> and teral>mako</literal>, respectively. </paragraph><paragraph>Inline comments in the text are removed from the output b y:</paragraph>teral\_block xml:space="preserve">Terminal&gt; doconce format la tex mydoc --skip\_inline\_comments ll such comments from the original Doconce file by running:file by running:file by running: once remove\_inline\_comments mydoc</literal\_block><paragraph>This action is conve nient when a Doconce document reaches its final form and comments by different authors should be removed.</paragraph></section><secti on dupnames="html" ids="id1"><title>HTML</title><paragraph>Making an HTML versio n of a Doconce file teral>mydoc.do.txt is performed by:</paragraph>literal\_block xml:space="preserve">Terminal&gt; doc once format html mydocliteral\_block><paragraph>The resulting file literal>myd oc.html</literal> can be loaded into any web browser for viewing.</paragraph><pa ragraph>The HTML style can be defined either in the header of the HTML file or in an external CSS file. The latter is enabled by the command-line argument teral>--css=filenameliteral>. There is a default style with blue h eadings, and a style with the <reference name="solarized" refuri="http://ethanschoonover. com/solarized">solarized</reference><target ids="solarized" names="solarized" re furi="http://ethanschoonover.com/solarized"/> color palette, specified by the teral>--html-solarized</literal> command line

specification, the blue or solarized styles are written to <literal>filename teral> and linked from the HTML document. You can provide your own style sheet

argument. If there is no file with name <literal>filename</literal> in the <lite

ral>--css=filename</literal>

either by replacing the content inside the teral>style</literal> tags or by specifying a CSS file through the teral>--css=filename</literal> option.</par

# tutorial.xml agraph><paragraph>If the Pygments package (including the teral>pygmentize is installed, code blocks are typeset with aid of this package. The command-line argument teral>--no-pygments-html turns off the use of Pygments and makes code blocks appear with

plain (teral>pre
the disc of fygments and makes code blocks appear with plain (teral>pre
the disc of fygments and makes code blocks appear with plain (teral>pre
the disc of fygments and makes code blocks appear with plain plain (the property and plain plain

on line numbers in Pygments-formatted code blocks.</paragraph><paragraph>The HTM L file can be embedded in a template if the Doconce document

does not have a title (because then there will be

no header and footer in the HTML file). The template file must contain valid HTML code and can have three "slots": </l></l></l></

<literal>%(date)s</literal> for a date, and <literal>%(main)s</literal> for the
main body of text, i.e., the

Doconce document translated to HTML. The title becomes the first heading in the Doconce document, and the date is extracted from the

<literal>DATE:</literal> line, if present. With the template feature one can eas
ily embed

the text in the look and feel of a website. The template can be extracted from the source code of a page at the site; just insert teral>%(title)sral> and

<literal>%(date)s</literal> at appropriate places and replace the main bod of te
xt

by teral>%(main)sliteral>. Here is an example:</paragraph><literal\_block xm l:space="preserve">Terminal&gt; doconce format html mydoc --html-template=mytemp late.htmlliteral\_block></section><section ids="blogs" names="blogs"><title>Blogs</title><paragraph>Doconce can be used for writing blogs provided the blog sit e accepts

raw HTML code. Google's Blogger service (teral>blogname.blogspot.com

is particularly well suited since it also allows extensive LaTeX mathematics via MathJax.

Write the blog text as a Doconce document without any title, author, and date. Then generate HTML as described above. Copy the text and paste it into the text area in the blog, making sure the input format is HTML. On Google's Blogger service you can use Doconce to generate blogs with LaTeX mathematics and pretty (pygmentized) blocks of computer code.

See a <reference name="blog example" refuri="http://doconce.blogspot.no">blog example</reference><target ids="blog-example" names="blog\ example" refuri="http://doconce.blogspot.no"/> for details on blogging.</paragraph><warning><paragraph>In the comments after the blog one cannot paste raw HTML code with MathJax

scripts so there is no support for mathematics in the comments.</paragraph></war ning><paragraph>WordPress (literal>wordpress.comliteral>) allows raw HTML cod e in blogs,

but has very limited

LaTeX support, basically only formulas. The teral>--wordpressliteral> option to

teral>doconce</literal> modifies the HTML code such that all equations are ty
peset

in a way that is acceptable to WordPress.

There is a <reference name="doconce example" refuri="http://doconce.wordpress.com">doconce example</reference><target ids="doconce-example" names="doconce\ example" refuri="http://doconce.wordpress.com"/>

on blogging with mathematics and code on WordPress.</paragraph></section><section nids="pandoc-and-markdown" names="pandoc\ and\ markdown"><title>Pandoc and Markdown</title><paragraph>Output in Pandoc's extended Markdown format results from: </paragraph>literal\_block xml:space="preserve">Terminal&gt; doconce format pandoc's extended Markdown format results from: </paragraph>

"

oc mydoc</literal\_block><paragraph>The name of the output file is teral>mydoc .mkd</literal>.

From this format one can go to numerous other formats:</paragraph><literal\_block xml:space="preserve">Terminal&gt; pandoc -R -t mediawiki -o mydoc.mwk --toc myd oc.mkd</literal\_block><paragraph>Pandoc supports <literal>latex</literal>, <literal>html</literal>, <literal>odt</literal> (OpenOffice), <literal>docx</literal> (Microsoft

Pandoc pass raw HTML or LaTeX to the output format instead of ignoring it, while the teral>--tocliteral> option generates a table of contents.

See the <reference name="Pandoc documentation" refuri="http://johnmacfarlane.net/pandoc/README.html">Pandoc documentation</reference><target ids="pandoc-documentation" names="pandoc\ documentation" refuri="http://johnmacfarlane.net/pandoc/README.html"/>

for the many features of the eliteral>pandoceliteral> program.</paragraph><paragraph>Pandoc is useful to go from LaTeX mathematics to, e.g., HTML or MS Word. There are two ways (experiment to find the best one for your document):

teral>doconce format pandoc</literal> and then translating using teral>pan doc</literal>, or

teral>doconce format latex</literal>, and then going from LaTeX to the desire
d format

using <literal>pandoc</literal>.

Here is an example on the latter strategy:</paragraph><literal\_block xml:space="preserve">Terminal&gt; doconce format latex mydoc

Terminal> doconce ptex2tex mydoc

Terminal> doconce replace '\Verb!' '\verb!' mydoc.tex

Terminal> pandoc -f latex -t docx -o mydoc.docx mydoc.tex
literal\_block><par agraph>When we go through literal>pandoc
only single equations or iteral>align\*

environments are well understood.</paragraph><paragraph>Note that Doconce applie s the eral>Verberal> macro from the eral>fancyvrberal> package while eral>pandoceral> only supports the standard eral>verberal> construction for

inline verbatim text. Moreover, quite some additional <literal>doconce
replace</literal> and <literal>doconce subst</literal> edits might be needed on
the <literal>.mkd</literal> or

<literal>.tex</literal> files to successfully have mathematics that is well tran
slated

to MS Word. Also when going to reStructuredText using Pandoc, it can be advantageous to go via LaTeX.</paragraph>cparagraph>Here is an example where we take a Doconce snippet (without title, author,

and date), maybe with some unnumbered equations, and quickly generate

HTML with mathematics displayed my MathJax:</paragraph><literal\_block xml:space= "preserve">Terminal&gt; doconce format pandoc mydoc

Terminal> pandoc -t html -o mydoc.html -s --mathjax mydoc.mkd</literal\_block> <paragraph>The teral>-s</literal> option adds a proper header and footer to t he teral>mydoc.html</literal> file.

This recipe is a quick way of makeing HTML notes with (some) mathematics.</paragraph></section><section dupnames="latex" ids="id2"><title>LaTeX</title><paragraph>Making a LaTeX file literal>mydoc.tex</literal> from literal>mydoc.do.txt

.. Note: putting code blocks inside a list is not successful in many</paragraph> <comment xml:space="preserve">formats - the text may be messed up. A better choi ce is a paragraph</comment><comment xml:space="preserve">environment, as used he re.</comment><paragraph><emphasis>Step 1.</emphasis> Filter the doconce text to a pre-LaTeX form literal>mydoc.p.tex</literal> for

the teral>ptex2texprogram (or teral>doconce ptex2tex>/paragraph>teral\_block xml:space="preserve">Terminal> doconce format lat

,,

```
tutorial.xml
ex mydoc</literal_block><paragraph>LaTeX-specific commands (&quot;newcommands&qu
ot;) in math formulas and similar
can be placed in files teral>newcommands.tex</literal>, teral>newcommands_
keep.tex</literal>, or
<literal>newcommands_replace.tex</literal> (see the section <reference name="Mac</pre>
ros (Newcommands), Cross-References, Index, and Bibliography refid="macros-newc
ommands-cross-references-index-and-bibliography">Macros (Newcommands), Cross-Ref
erences, Index, and Bibliography</reference>).
If these files are present, they are included in the LaTeX document
so that your commands are defined.</paragraph><paragraph>An option <literal>--la
tex-printed</literal> makes some adjustments for documents
aimed at being printed. For example, links to web resources are
associated with a footnote listing the complete web address (URL).</paragraph><p
aragraph><emphasis>Step 2.</emphasis> Run teral>ptex2tex
ve it) to make a standard LaTeX file:</paragraph><literal_block xml:space="prese"
rve">Terminal> ptex2tex mydoc</literal_block><paragraph>In case you do not ha
ve <literal>ptex2tex</literal>, you may run a (very) simplified version:
aph>aph>teral_block xml:space="preserve">Terminal> doconce ptex2tex mydoc</lit
eral_block><paragraph>Note that Doconce generates a literal>.p.tex
le with some preprocessor macros
that can be used to steer certain properties of the LaTeX document.
For example, to turn on the Helvetica font instead of the standard
Computer Modern font, run:</paragraph>literal_block xml:space="preserve">Termin
al> ptex2tex -DHELVETICA mydoc
Terminal> doconce ptex2tex mydoc -DHELVETICA # alternative</literal_block><p
aragraph>The title, authors, and date are by default typeset in a non-standard
way to enable a nicer treatment of multiple authors having
institutions in common. However, the standard LaTeX " maketitle" headin
is also available through teral>-DLATEX_HEADING=traditional
A separate titlepage can be generate by
<literal>-DLATEX_HEADING=titlepage</literal>./paragraph>Preprocessor
 variables to be defined or undefined are</paragraph><block_quote><bullet_list b
ullet="*"><list_item><paragraph><literal>BOOK</literal> for the &quot;book&quot;
documentclass rather than the standard
" article" class (necessary if you apply chapter headings)</paragraph><
/list_item><list_item><paragraph><literal>PALATINO</literal> for the Palatino fo
nt</paragraph></list_item><list_item><literal>HELVETIA</literal> for
the Helvetica font</paragraph></list_item><list_item><paragraph><literal>A4PAPER
</literal> for A4 paper size</paragraph></list_item><list_item><paragraph><liter</pre>
al>A6PAPER</literal> for A6 paper size (suitable for reading on small devices)</
paragraph></list_item><paragraph><literal>MOVIE15</literal> for using
 the movie15 LaTeX package to display movies</paragraph></list_item><
paragraph>teral>PREAMBLEteral> to turn the LaTeX preamble on or off (i.e.
 complete document
or document to be included elsewhere)</paragraph></list_item><paragraph
ph>teral>MINTED</literal> for inclusion of the minted package (which requires
 <literal>latex</literal>
or teral>pdflatex</literal> to be run with the teral>-shell-escape</literal>
l> option)
teral>ptex2tex</literal> tool makes it possible to easily switch between many
different fancy formattings of computer or verbatim code in LaTeX
documents. After any teral>!bc</literal> command in the Doconce source you ca
insert verbatim block styles as defined in your teral>.ptex2tex.cfg
file, e.g., e.g., teral>!bc sys</literal> for a terminal session, where teral>sy
```

rminal</literal>).

a certain environment in teral>.ptex2tex.cfgeral> (e.g., <literal>CodeTe

sset to

```
tutorial.xml
There are about 40 styles to choose from, and you can easily add
new ones.</paragraph><paragraph>Also the teral>doconce ptex2tex</literal> com
mand supports preprocessor directives
for processing the teral>.p.texfile. The command allows specificat
of code environments as well. Here is an example:</paragraph>teral block xml:
space="preserve">Terminal> doconce ptex2tex mydoc -DLATEX_HEADING=traditional
          -DPALATINO -DA6PAPER \
          " sys=\begin{quote}\begin{verbatim}@\end{verbatim}\end{quote}&quot
          fpro=minted fcod=minted shcod=Verbatim envir=ans:nt</literal_block><pa</pre>
ragraph>Note that teral>@</literal> must be used to separate the begin and en
commands, unless only the environment name is given (such as eral>minted
teral>
above, which implies teral>\begin{minted}{fortran}
d{minted}</literal> as
begin and end for blocks inside teral>!bc fpro</literal> and <literal>!ec
teral>). Specifying
<literal>envir=ans:nt</literal> means that all other environments are typeset wi
th the
<literal>anslistings.sty</literal> package, e.g., <literal>!bc cppcod</literal>
will then result in
<literal>\begin{c++}</literal>. If no environments like <literal>sys</literal>,
<literal>fpro</literal>, or the common
<literal>envir</literal> are defined on the command line, the plain <literal>\be
qin{verbatim}</literal>
and teral>\end{verbatim}used.</paragraph><paragraph><emphasis>Step
 2b (optional).</emphasis> Edit the teral>mydoc.tex</literal> file to your ne
eds.
For example, you may want to substitute teral>section</literal> by teral>s
ection*</literal> to
avoid numbering of sections, you may want to insert linebreaks
(and perhaps space) in the title, etc. This can be automatically
edited with the aid of the teral>doconce replace</literal> and teral>docon
ce subst</literal>
commands. The former works with substituting text directly, while the
latter performs substitutions using regular expressions.
Here are two examples:</paragraph>teral_block xml:space="preserve">Terminal&q
t; doconce replace 'section{' 'section*{' mydoc.tex
Terminal> doconce subst 'title\{(.+)Using (.+)\}' \
'title{\g<1&gt; \\\ [1.5mm] Using \g&lt;2&gt;' mydoc.tex</literal_block><paragraph>A lot of tailored fixes to the LaTeX document can be done by
an appropriate set of text replacements and regular expression
substitutions. You are anyway encourged to make a script for generating PDF from the LaTeX file.</paragraph><pemphasis>Step 3.</pemp
hasis> Compile teral>mydoc.tex</literal>
and create the PDF file:</paragraph>teral_block xml:space="preserve">Terminal
> latex mydoc
Terminal> latex mydoc
Terminal> makeindex mydoc  # if index
                               # if bibliography
Terminal> bibitem mydoc
Terminal> latex mydoc
Terminal> dvipdf mydoc</literal_block><paragraph>If one wishes to run <litera
1>ptex2tex</literal> and use the minted LaTeX package for
typesetting code blocks (<literal>Minted_Python</literal>, <literal>Minted_Cpp</
literal>, etc., in
<literal>ptex2tex</literal> specified through the <literal>*pro</literal> and <l</pre>
```

```
tutorial.xml
iteral>*cod</literal> variables in
<literal>.ptex2tex.cfg</literal> or <literal>$HOME/.ptex2tex.cfg</literal>), the
minted LaTeX package is
        This package is included by running teral>ptex2texeliteral> with the
<literal>-DMINTED</literal> option:
e">Terminal> ptex2tex -DMINTED mydoc</literal_block><paragraph>In this case,
<literal>latex</literal> must be run with the
<literal>-shell-escape</literal> option:cliteral_block xml:space="pred" option:
eserve">Terminal> latex -shell-escape mydoc
Terminal> latex -shell-escape mydoc
Terminal> makeindex mydoc
                             # if index
                              # if bibliography
Terminal> bibitem mydoc
Terminal> latex -shell-escape mydoc
Terminal> dvipdf mydoc</literal_block><paragraph>When running teral>doconc
e ptex2tex mydoc envir=minted</literal> (or other minted
specifications with teral>doconce ptex2tex</literal>), the minted package is
automatically
included so there is no need for the teral>-DMINTED</literal> option.</paragr
aph></section><section ids="pdflatex" names="pdflatex"><title>PDFLaTeX</title><p
aragraph>Running <literal>pdflatex</literal> instead of <literal>latex</literal>
 follows almost the same steps,
but the start is:</paragraph><literal block xml:space="preserve">Terminal&qt; do
conce format latex mydoc</literal block><paragraph>Then teral>ptex2tex</liter</pre>
al> is run as explained above, and finally:</paragraph><literal_block xml:space=
"preserve">Terminal> pdflatex -shell-escape mydoc
Terminal> makeindex mydoc
                             # if index
                              # if bibliography
Terminal&qt; bibitem mydoc
Terminal> pdflatex -shell-escape mydoc</literal_block></section><section ids=
"plain-ascii-text" names="plain\ ascii\ text"><title>Plain ASCII Text</title><pa
ragraph>We can go from Doconce " back to" plain untagged text suitable
for viewing
in terminal windows, inclusion in email text, or for insertion in
computer source code:/paragraph><literal_block xml:space="preserve">Terminal&gt
; doconce format plain mydoc.do.txt # results in mydoc.txt</literal_block></sec</pre>
tion><section dupnames="restructuredtext" ids="id3"><title>reStructuredText</tit
le><paragraph>Going from Doconce to reStructuredText gives a lot of possibilitie
go to other formats. First we filter the Doconce text to a
reStructuredText file teral>mydoc.rsteliteral>:</paragraph><literal block xm
l:space="preserve">Terminal> doconce format rst mydoc.do.txt</literal block><
paragraph>We may now produce various other formats:</paragraph>teral_block xm
l:space="preserve">Terminal> rst2html.py mydoc.rst > mydoc.html # html
Terminal> rst2latex.py mydoc.rst > mydoc.tex # latex
Terminal> rst2xml.py
                        mydoc.rst > mydoc.xml
                                                  # XML
Terminal> rst2odt.py
                        mydoc.rst > mydoc.odt # OpenOffice</literal_block>
<paragraph>The OpenOffice file <literal>mydoc.odt</literal> can be loaded into O
penOffice and
saved in, among other things, the RTF format or the Microsoft Word format.
However, it is more convenient to use the program teral>unovonv</literal>
to convert between the many formats OpenOffice supports <emphasis>on the command
 line</emphasis>.
Run:</paragraph>teral_block xml:space="preserve">Terminal&gt; unoconv --show<
/literal_block><paragraph>to see all the formats that are supported.
For example, the following commands take
teral>mydoc.odt</literal> to Microsoft Office Open XML format,
classic MS Word format, and PDF:</paragraph>teral_block xml:space="preserve">
Terminal&qt; unoconv -f ooxml mydoc.odt
Terminal> unoconv -f doc mydoc.odt
```

he

Terminal> unoconv -f pdf mydoc.odt</literal\_block><paragraph><emphasis>Remark about Mathematical Typesetting.</emphasis> At the time of this writing, there i s no easy way to go from Doconce and LaTeX mathematics to reST and further to OpenOffice and the " MS Word world". Mathematics is only fully supported by teral>latex output and to a wide extent also supported by the teral>sphinx ut format. Some links for going from LaTeX to Word are listed below.</paragraph><block\_quot e><bullet\_list bullet="\*"><list\_item><paragraph><reference name="http://ubuntufo</pre> rums.org/showthread.php?t=1033441" refuri="http://ubuntuforums.org/showthread.ph p?t=1033441">http://ubuntuforums.org/showthread.php?t=1033441</reference><target  $\verb|ids="http-ubuntuforums-org-showthread-php-t-1033441"| names="http://ubuntuforums-org-showthread-php-t-1033441"| names="http://ub$ .org/showthread.php?t=1033441" refuri="http://ubuntuforums.org/showthread.php?t= 1033441"/></paragraph></list\_item><list\_item><paragraph><reference name="http:// tug.org/utilities/texconv/textopc.html" refuri="http://tug.org/utilities/texconv /textopc.html">http://tug.org/utilities/texconv/textopc.html</reference><target ids="http-tug-org-utilities-texconv-textopc-html" names="http://tug.org/utilitie s/texconv/textopc.html" refuri="http://tug.org/utilities/texconv/textopc.html"/> </paragraph></list\_item><list\_item><paragraph><reference name="http://nileshbans")</pre> al.blogspot.com/2007/12/latex-to-openofficeword.html" refuri="http://nileshbansa 1.blogspot.com/2007/12/latex-to-openofficeword.html">http://nileshbansal.blogspo t.com/2007/12/latex-to-openofficeword.html</reference><target ids="http-nileshba nsal-blogspot-com-2007-12-latex-to-openofficeword-html" names="http://nileshbans al.blogspot.com/2007/12/latex-to-openofficeword.html" refuri="http://nileshbansa 1.blogspot.com/2007/12/latex-to-openofficeword.html"/></paragraph></list\_item></</pre> bullet\_list></block\_quote></section><section dupnames="sphinx" ids="id4"><title> Sphinx</title><paragraph>Sphinx documents demand quite some steps in their creat ion. We have automated most of the steps through the teral>doconce sphinx\_dir</literal> command:</pa ragraph><literal\_block xml:space="preserve">Terminal&gt; doconce sphinx\_dir auth or="authors' names" \ title="some title" version=1.0 dirname=sphinxdir \ theme=mytheme file1 file2 file3 ...</literal\_block><paragraph>The keyw ords teral>authorliteral>, teral>titleliteral>, and teral>version literal> are used in the headings of the Sphinx document. By default, teral>version</literal> is 1.0 and the sc will try to deduce authors and title from the doconce files teral>file1 <literal>file2</literal>, etc. that together represent the whole document. Note none of the individual Doconce files teral>file1teral>, teral>file2</l iteral>, etc. should include the rest as their union makes up the whole document. The default value of <literal>dirname
is <literal>sphinx-rootdir/lite ral>. The teral>theme</literal> keyword is used to set the theme for design of HTML output from Sphinx (the default theme is teral>'default' h>With a single-file document in teral>mydoc.do.txt runs:runs:/paragraph>teral\_block xml:space="preserve">Terminal> doconce sphinx \_dir mydoc</literal\_block><paragraph>and then an appropriate Sphinx directory <1 iteral>sphinx-rootdir</literal> is made with relevant files.</paragraph><paragraph>The teral>doconce sphinx\_dir command generates a script <literal>automake\_sphinx.py</literal> for compiling the Sphinx document into an document. One can either run teral>automake sphinx.py</literal> or perform t

```
tutorial.xml
steps in the script manually, possibly with necessary modifications.
You should at least read the script prior to executing it to have
some idea of what is done.</paragraph><paragraph>The teral>doconce sphinx_dir
</literal> script copies directories named <literal>figs</literal> or
teral>figures/literal> over to the Sphinx directory so that figures are acce
in the Sphinx compilation. If figures or movies are located in other
directories, teral>automake_sphinx.py</literal> must be edited accordingly.
to which there are local links (not <literal>http:</literal> or <literal>file:</
literal> URLs), must be
placed in the eral>_staticeral> subdirectory of the Sphinx directory. T
utility teral>doconce sphinxfix localURLs
 links
in the Doconce file: for each such link, say teral>dir1/dir2/myfile.txt</lite
ral> it
replaces the link by teral>_static/myfile.txt</literal> and copies
<literal>dir1/dir2/myfile.txt</literal> to a local <literal>_static</literal> di
rectory (in the same
directory as the script is run). However, we recommend instead that
the writer of the document places files in teral>_staticteral> or lets a
do it automatically. The user must copy all teral> static/*</literal> files t
o the
<literal>_static</literal> subdirectory of the Sphinx directory. It may be wise
always put files, to which there are local links in the Doconce
document, in a teral>_staticor <literal>_static-namedir
ectory and use these
local links. Then links do not need to be modified when creating a
Sphinx version of the document.</paragraph><paragraph>Doconce comes with a colle
ction of HTML themes for Sphinx documents.
These are packed out in the Sphinx directory, the teral>conf.py</literal>
configuration file for Sphinx is edited accordingly, and a script
<literal>make-themes.sh</literal> can make HTML documents with one or more theme
s.
For example,
to realize the themes <literal>fenics/literal> and <literal>pyramid/literal>,
one writes:</paragraph><literal block xml:space="preserve">Terminal&gt; ./make-t
hemes.sh fenics pyramidliteral block><paragraph>The resulting directories with
HTML documents are <literal>_build/html_fenics</literal>
and and teral>_build/html_pyramid, respectively. Without arguments,
<literal>make-themes.sh</literal> makes all available themes (!).</paragraph><pa</pre>
ragraph>If it is not desirable to use the autogenerated scripts explained
above, here is the complete manual procedure of generating a
Sphinx document from a file teral>mydoc.do.txt</literal>.</paragraph><paragra</p>
ph><emphasis>Step 1.</emphasis> Translate Doconce into the Sphinx format:</parag
raph>teral_block xml:space="preserve">Terminal> doconce format sphinx mydo
c</literal_block><paragraph><emphasis>Step 2.</emphasis> Create a Sphinx root di
rectory
either manually or by using the interactive <literal>sphinx-quickstart</literal>
program. Here is a scripted version of the steps with the latter:</paragraph><li
teral_block xml:space="preserve">mkdir sphinx-rootdir
sphinx-quickstart <&lt;EOF
sphinx-rootdir
```

Name of My Sphinx Document

```
tutorial.xml
Author
version
version
.rst
index
n
У
n
n
n
n
У
n
n
У
У
У
EOF</literal_block><paragraph>The autogenerated <literal>conf.py</literal> file
may need some edits if you want to specific layout (Sphinx themes)
of HTML pages. The teral>doconce sphinx_dir
nded <literal>conv.py</literal>
file where, among other things, several useful Sphinx extensions
are included.</paragraph><emphasis>Step 3.</emphasis> Copy the emphasis> Copy the 
al>mydoc.rst</literal> file to the Sphinx root directory:</paragraph><literal_bl
ock xml:space="preserve">Terminal> cp mydoc.rst sphinx-rootdir</literal_block</pre>
><paragraph>If you have figures in your document, the relative paths to those wi
be invalid when you work with teral>mydoc.rst</literal> in the teral>sphin
x-rootdir</literal>
directory. Either edit teral>mydoc.rsteliteral> so that figure file paths ar
or simply copy your figure directories to teral>sphinx-rootdir
Links to local files in teral>mydoc.rst</literal> must be modified to links t
files in the teral>_staticdirectory, see comment above.</paragraph</li>
><paragraph><emphasis>Step 4.</emphasis> Edit the generated teral>index.rst</
literal> file so that teral>mydoc.rst</literal>
is included, i.e., add iteral>mydoc/literal> to the teral>toctree/literal
> section so that it becomes:</paragraph><literal_block xml:space="preserve">...
toctree::
     :maxdepth: 2
    mydoc</literal_block><paragraph>(The spaces before <literal>mydoc</literal> a
re important!)</paragraph><pmphasis>Step 5.</pmphasis> Generate, for
instance, an HTML version of the Sphinx source:</paragraph><literal_block xml:sp
ace="preserve">make clean # remove old versions
make html</literal_block><paragraph>Sphinx can generate a range of different for
mats:
standalone HTML, HTML in separate directories with teral>index.html
a large single HTML file, JSON files, various help files (the qthelp, HTML,
and Devhelp projects), epub, LaTeX, PDF (via LaTeX), pure text, man pages,
and Texinfo files.</paragraph><pmphasis>Step 6.</pmphasis> View the r
esult:</paragraph><literal_block xml:space="preserve">Terminal&gt; firefox _buil
d/html/index.html</literal_block><paragraph>Note that verbatim code blocks can b
e typeset in a variety of ways
depending the argument that follows <literal>!bc</literal>: cliteral>cod</literal
1> gives Python
(teral>code-block:: python</literal> in Sphinx syntax) and <literal>cppcod</l
```

ction><section ids="tweaking-the-doconce-output" names="tweaking\ the\ doconce\ output"><title>Tweaking the Doconce Output</title><paragraph>Occasionally, one w

```
tutorial.xml
ould like to tweak the output in a certain format
from Doconce. One example is figure filenames when transforming
Doconce to reStructuredText. Since Doconce does not know if the
<literal>.rst</literal> file is going to be filtered to LaTeX or HTML, it cannot
if teral>.epseliteral> or teral>.pnq</literal> is the most appropriate im
age filename.
The solution is to use a text substitution command or code with, e.g., sed,
perl, python, or scitools subst, to automatically edit the output file
from Doconce. It is then wise to run Doconce and the editing commands
from a script to automate all steps in going from Doconce to the final
format(s). The teral>make.sh/literal> files in teral>docs/manual
> and <literal>docs/tutorial</literal>
constitute comprehensive examples on how such scripts can be made.</paragraph></
section><section ids="demos" names="demos"><title>Demos</title><paragraph>The cu
rrent text is generated from a Doconce format stored in the file:</paragraph><li
teral_block xml:space="preserve">docs/tutorial/tutorial.do.txt</literal_block><p</pre>
aragraph>The file teral>make.shliteral> in the teral>tutorialliteral>
directory of the
Doconce source code contains a demo of how to produce a variety of
formats. The source of this tutorial, teral>tutorial.do.txt
starting point. Running literal>make.sh/literal> and studying the various gen
files and comparing them with the original <literal>tutorial.do.txt</literal> fi
gives a quick introduction to how Doconce is used in a real case.
<reference name="Here" refuri="https://doconce.googlecode.com/hg/doc/demos/tutor</pre>
ial/index.html">Here</reference><target ids="here" names="here" refuri="https://</pre>
doconce.googlecode.com/hg/doc/demos/tutorial/index.html"/>
is a sample of how this tutorial looks in different formats.</paragraph><paragra
ph>There is another demo in the teral>docs/manual
translates the more comprehensive documentation, teral>manual.do.txt
>, to
various formats. The literal>make.sh/literal> script runs a set of translation
s.</paragraph></section></section ids="installation-of-doconce-and-its-
dependencies" names="installation\ of\ doconce\ and\ its\ dependencies"><title>I
nstallation of Doconce and its Dependencies</title><section ids="doconce" names=
"doconce"><title>Doconce</title><paragraph>Doconce itself is pure Python code ho
sted at <reference name="http://code.google.com/p/doconce" refuri="http://code.g
oogle.com/p/doconce">http://code.google.com/p/doconce</reference><target ids="ht
tp-code-google-com-p-doconce names="http://code.google.com/p/doconce" refuri="h
ttp://code.google.com/p/doconce"/>. Its installation from the
Mercurial (teral>hg</literal>) source follows the standard procedure:</paragr
aph><literal_block xml:space="preserve"># Doconce
hg clone https://doconce.googlecode.com/hg/ doconce
cd doconce
sudo python setup.py install
cd ..</literal_block><paragraph>Since Doconce is frequently updated, it is recom
mended to use the
above procedure and whenever a problem occurs, make sure to
update to the most recent version:</paragraph><literal_block xml:space="preserve"
">cd doconce
hg pull
hg update
sudo python setup.py installliteral_block><paragraph>Debian GNU/Linux users ca
n also run:preserve>sudo apt-get install
doconce</literal_block><paragraph>This installs the latest release and not the m
```

ost updated and bugfixed

version.

```
tutorial.xml
On Ubuntu one needs to run:</paragraph><literal_block xml:space="preserve">sudo
add-apt-repository ppa:scitools/ppa
sudo apt-get update
sudo apt-get install doconce</literal block></section><section ids="dependencies"
" names="dependencies"><title>Dependencies</title><section ids="preprocessors" n
ames="preprocessors"><title>Preprocessors</title><paragraph>If you make use of t
he <reference name="Preprocess" refuri="http://code.google.com/p/preprocess">Pre
process</reference><target ids="preprocess" names="preprocess" refuri="http://co</pre>
de.google.com/p/preprocess"/>
preprocessor, this program must be installed:/paragraph><literal_block xml:spac</pre>
e="preserve">svn checkout http://preprocess.googlecode.com/svn/trunk/ preprocess
cd preprocess
cd doconce
sudo python setup.py install
cd ..</literal_block><paragraph>A much more advanced alternative to Preprocess i
<reference name="Mako" refuri="http://www.makotemplates.org">Mako</reference><ta</pre>
rget ids="mako" names="mako" refuri="http://www.makotemplates.org"/>. Its instal
lation is most
conveniently done by <literal>pip</literal>:/paragraph><literal_block xml:space</pre>
="preserve">pip install Mako</literal_block><paragraph>This command requires !
teral>pip</literal> to be installed. On Debian Linux systems,
such as Ubuntu, the installation is simply done by:</paragraph>teral_block xm
l:space="preserve">sudo apt-get install python-pip</literal_block><paragraph>Alt
ernatively, one can install from the teral>pip</literal> <reference name="sou
rce code" refuri="http://pypi.python.org/pypi/pip">source code</reference><targe
t ids="source-code" names="source\ code" refuri="http://pypi.python.org/pypi/pip
"/>.</paragraph><paragraph>Mako can also be installed directly from
<reference name="source" refuri="http://www.makotemplates.org/download.html">sou
rce</reference><target ids="source" names="source" refuri="http://www.makotempla
tes.org/download.html"/>: download the
tarball, pack it out, go to the directory and run
the usual teral>sudo python setup.py install
<section ids="image-file-handling" names="image\ file\ handling"><title>Image fi
le handling</title><paragraph>Different output formats require different formats
of image files.
For example, PostScript or Encapuslated PostScript is required for teral>late
x</literal>
output, while HTML needs JPEG, GIF, or PNG formats.
Doconce calls up programs from the ImageMagick suite for converting
image files to a proper format if needed. The <reference name="ImageMagick suite
" refuri="http://www.imagemagick.org/script/index.php">ImageMagick suite</refere
nce><target ids="imagemagick-suite" names="imagemagick\ suite" refuri="http://ww
w.imagemagick.org/script/index.php"/> can be installed on all major platforms.
On Debian Linux (including Ubuntu) systems one can simply write:</paragraph><lit
eral_block xml:space="preserve">sudo apt-get install imagemagick</literal_block>
<paragraph>The convenience program <literal>doconce combine_images</literal>, fo
r combining several
images into one, will use literal>montage/literal> and literal>convert/liter
al> from ImageMagick and
the teral>pdftkliteral>, pdfnup</literal>, and teral>pdfcrop
iteral> programs from the teral>texlive-extra-utils
Debian package. The latter gets installed by:</paragraph><literal_block xml:spac
e="preserve">sudo apt-get install texlive-extra-utils</literal_block></section><
section ids="spellcheck" names="spellcheck"><title>Spellcheck</title><paragraph>
The utility teral>doconce spellcheck</literal> applies the teral>ispell
iteral> program for
```

\_block xml:space="preserve">sudo apt-get install ispell</literal\_block></section ,

spellcheck. On Debian (including Ubuntu) it is installed by:</paragraph>teral

><section ids="ptex2tex-for-latex-output" names="ptex2tex\ for\ latex\ output">< title>Ptex2tex for LaTeX Output</title><paragraph>To make LaTeX documents with v ery flexible choice of typesetting of

verbatim code blocks you need <reference name="ptex2tex" refuri="http://code.goo
gle.com/p/ptex2tex">ptex2tex</reference><target ids="ptex2tex" names="ptex2tex"
refuri="http://code.google.com/p/ptex2tex"/>,

which is installed by:/paragraph><literal\_block xml:space="preserve">svn checko
ut http://ptex2tex.googlecode.com/svn/trunk/ ptex2tex
cd ptex2tex

sudo python setup.py install<lr><ld>ditional style files, you can run</ld>

a script, <literal>cp2texmf.sh</literal>:/paragraph><literal\_block xml:space="p
reserve">cd latex

sh cp2texmf.sh # copy stylefiles to ~/texmf directory

cd ../..</literal\_block><paragraph>This script copies some special stylefiles th
at

that teral>ptex2texpotentially makes use of. Some more standard st ylefiles

are also needed. These are installed by:</paragraph>teral\_block xml:space="preserve">sudo apt-get install texlive-latex-recommended texlive-latex-extraral\_block><paragraph>on Debian Linux (including Ubuntu) systems. TeXShop on Maccomes with

the necessary stylefiles (if not, they can be found by googling and installed manually in the teral>~/texmf/tex/latex/misc</literal> directory).</paragraph ><paragraph>Note that the <literal>doconce ptex2tex</literal> command, which nee ds no installation

beyond Doconce itself, can be used as a simpler alternative to the eral>ptex 2tex

program.</paragraph><paragraph>The <emphasis>minted</emphasis> LaTeX style is of
fered by teral>ptex2tex</literal> and <literal>doconce ptext2tex</literal>
is popular among many

users. This style requires the package <reference name="Pygments" refuri="http://pygments.org">Pygments</reference><target ids="pygments" names="pygments" refuri="http://pygments.org"/>

to be installed. On Debian Linux:</paragraph><literal\_block xml:space="preserve" > sudo apt-get install python-pygments
/literal\_block><paragraph>Alternatively, t he package can be installed manually:</paragraph><literal\_block xml:space="preserve">hg clone ssh://hg@bitbucket.org/birkenfeld/pygments-main pygments cd pygments

sudo python setup.py install<lr><ld>le together with literal>ptex2texliteral>, you have to

enable it by the enab

This is not necessary if you run the alternative literal>doconce ptex2tex/literal> program.

use of the minted style requires the eral>-shell-escape</literal> command-li ne

argument when running LaTeX, i.e., teral>latex -shell-escapeteral> or teral>pdflatex

-shell-escape</literal>.</paragraph><comment xml:space="preserve">Say something about anslistings.sty</comment></section><section ids="restructuredtext-rest-out put" names="restructuredtext\ (rest)\ output"><title>reStructuredText (reST) Out put</title><paragraph>The literal>rst</literal> output from Doconce allows furt her transformation to LaTeX,

HTML, XML, OpenOffice, and so on, through the <reference name="docutils" refuri= "http://docutils.sourceforge.net">docutils</reference><target ids="docutils" nam es="docutils" refuri="http://docutils.sourceforge.net"/> package. The installat ion of the

most recent version can be done by:</paragraph><literal block xml:space="preserv"

,,

```
tutorial.xml
e">svn checkout http://docutils.svn.sourceforge.net/svnroot/docutils/trunk/docut
ils
cd docutils
sudo python setup.py install
cd ..</literal_block><paragraph>To use the OpenOffice suite you will typically o
n Debian systems install:</paragraph><literal block xml:space="preserve">sudo ap
t-get install unovonv libreoffice libreoffice-dmaths</literal block><paragraph>T
here is a possibility to create PDF files from reST documents
using ReportLab instead of LaTeX. The enabling software is
<reference name="rst2pdf" refuri="http://code.google.com/p/rst2pdf">rst2pdf</ref</pre>
erence><target ids="rst2pdf" names="rst2pdf" refuri="http://code.google.com/p/rs</pre>
t2pdf"/>. Either download the tarball
or clone the svn repository, go to the teral>rst2pdf</literal> directory and
run the usual teral>sudo python setup.py install</literal>.</paragraph><parag
raph>Output to teral>sphinxliteral> requires of course the
<reference name="Sphinx software" refuri="http://sphinx.pocoo.org">Sphinx softwa
re</reference><target ids="sphinx-software" names="sphinx\ software" refuri="htt
p://sphinx.pocoo.org"/>,
installed by:</paragraph><literal_block xml:space="preserve">hg clone https://bi
tbucket.org/birkenfeld/sphinx
cd sphinx
sudo python setup.py install
cd ..</literal_block></section><section ids="markdown-and-pandoc-output" names="</pre>
markdown\ and\ pandoc\ output"><title>Markdown and Pandoc Output</title><paragra
ph>The Doconce format <literal>pandoc</literal> outputs the document in the Pand
\circ
extended Markdown format, which via the teral>pandoc</literal> program can be
translated to a range of other formats. Installation of <reference name="Pandoc"
refuri="http://johnmacfarlane.net/pandoc/">Pandoc</reference><target ids="pando
c" names="pandoc" refuri="http://johnmacfarlane.net/pandoc/"/>, written in Haske
ll, is most
easily done by:</paragraph><literal_block xml:space="preserve">sudo apt-get inst
all pandoc</literal_block><paragraph>on Debian (Ubuntu) systems.</paragraph></se
ction><section ids="epydoc-output" names="epydoc\ output"><title>Epydoc Output</
title><paragraph>When the output format is <literal>epydoc</literal> one needs t
hat program too, installed
by:</paragraph><literal_block xml:space="preserve">svn co https://epydoc.svn.sou
rceforge.net/svnroot/epydoc/trunk/epydoc epydoc
cd epydoc
sudo make install
cd ..</literal block><paragraph><emphasis>Remark.</emphasis> Several of the pack
ages above installed from source code
are also available in Debian-based system through the
<literal>apt-get install</literal> command. However, we recommend installation d
irectly
from the version control system repository as there might be important
updates and bug fixes. For teral>svn</literal> directories, go to the directo
ry,
```

run run run setup.py instal

teral>hg pull; hg update</literal>, and then <literal>sudo python setup.py in

Mercurial (teral>hg</literal>) directories, go to the directory, run

stall</literal>.</paragraph></section></section></document>

l</literal>. For