tutorial.do.txt

```
TITLE: Doconce: Document Once, Include Anywhere
AUTHOR: Hans Petter Langtangen at Simula Research Laboratory and University of O
slo
DATE: September 10, 2010


# lines beginning with # are comment lines

 * 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 at some later stage eventually go
   with a particular format?

 * Do you find it problematic that 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
   place, and include it anywhere?

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


======= The Doconce Concept  =======

Doconce is two things:

  o Doconce is a working strategy for documenting software in a single
    place and avoiding duplication of information. The slogan is:
    "Write once, include anywhere". This requires that what you write
    can be transformed to many different formats for a variety of
    documents (manuals, tutorials, books, doc strings, source code
    documentation, etc.).

  o Doconce is a simple and minimally tagged markup language that can
    be used for the above purpose. That is, the Doconce format look
    like ordinary ASCII text (much like what you would use in an
    email), but the text can be transformed to numerous other formats,
    including HTML, Wiki, LaTeX, PDF, reStructuredText (reST), Sphinx,
    Epytext, and also plain text (where non-obvious formatting/tags are
    removed for clear reading in, e.g., emails). From reStructuredText
    you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the
    latter to RTF and MS Word.

Doconce was particularly written for the following sample applications:

  * Large books written in LaTeX, but where many pieces (computer demos,
    projects, examples) can be written in Doconce to appear in other
    contexts in other formats, including plain HTML, Sphinx, or MS Word.

  * Software documentation, primarily Python doc strings, which one wants
    to appear as plain untagged text for viewing in Pydoc, as reStructuredText
    for use with Sphinx, as wiki text when publishing the software at
    googlecode.com, and as LaTeX integrated in, e.g., a master's thesis.

  * Quick memos, which start as plain text in email, then some small
    amount of Doconce tagging is added, before the memos can appear as
    MS Word documents or in wikis.
```

tutorial.do.txt

```
======= 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. For example,

  * bullet lists arise from lines starting with an asterisk,

  * *emphasized words* are surrounded by asterisks,

  * _words in boldface_ are surrounded by underscores,

  * words from computer code are enclosed in back quotes and
    then typeset verbatim,

  * blocks of computer code can easily be included, also from source files,

  * blocks of LaTeX mathematics can easily be included,

  * there is support for both LaTeX and text-like inline mathematics,

  * figures with captions, URLs with links, labels and references
    are supported,

  * comments can be inserted throughout the text,

  * a preprocessor (much like the C preprocessor) is integrated so
    other documents (files) can be included and large portions of text
    can be defined in or out of the text.

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 an 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 http://folk.uio.no/hpl<hpl>.
Just a file link goes like 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
Chapter ref{my:first:sec}.

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    |
  |------------------------------|
!ec
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 http://folk.uio.no/hpl<hpl>.
Just a file link goes like 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
Chapter ref{my:first:sec}.

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 $\nu = \sin(x)$|$v = sin(x)$,
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 $\nu = \sin(x)$|$v = sin(x)$ is
typeset as
!bc
$\nu = \sin(x)$|$v = sin(x)$
!ec
The pipe symbol acts as a delimiter between LaTeX code and the plain text
version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside
```

tutorial.do.txt

```
‘!bt‘ and ‘!et‘ (begin tex / end tex) instructions.
The result looks like this:
!bt
\begin{eqnarray}
{\partial u\over\partial t} &=& \nabla^2 u + f,\label{myeq1}\\
{\partial v\over\partial t} &=& \nabla\cdot(q(u)\nabla v) + g
\end{eqnarray}
!et
Of course, such blocks only looks nice in LaTeX. The raw
LaTeX syntax appears in all other 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. Such blocks look like
!bc cod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec
It is possible to add a specification of a (ptex2tex-style)
environment for typesetting the verbatim code block, e.g., ‘!bc xxx‘
where ‘xxx‘ is an identifier like ‘pycod‘ for code snippet in Python,
‘sys‘ for terminal session, etc. When Doconce is filtered to LaTeX,
these identifiers are used as in ptex2tex and defined in a
configuration file ‘.ptext2tex.cfg‘, while when filtering
to Sphinx, one can have a comment line in the Doconce file for
mapping the identifiers to legal language names for Sphinx (which equals
the legal language names for Pygments):
!bc
 # sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console
!ec
By default, ‘pro‘ and ‘cod‘ are ‘python‘, ‘sys‘ is ‘console‘,
while ‘xpro‘ and ‘xcod‘ are computer language specific for ‘x‘
in ‘f‘ (Fortran), ‘c‘ (C), ‘cpp‘ (C++), and ‘py‘ (Python).
# ‘rb‘ (Ruby), ‘pl‘ (Perl), and ‘sh‘ (Unix shell).

# (Any sphinx code-block comment, whether inside verbatim code
# blocks or outside, yields a mapping between bc arguments
# and computer languages. In case of muliple definitions, the
# first one is used.)

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!). A complete file is typeset
with ‘!bc pro‘, while a part of a file is copied into a ‘!bc cod‘
environment. What ‘pro‘ and ‘cod‘ mean is then defined through
a ‘.ptex2tex.cfg‘ file for LaTeX and a ‘sphinx code-blocks‘
comment for Sphinx.

Another document can be included by writing ‘#include "mynote.do.txt"‘
on a line starting with (another) hash sign.  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 the
right writing enviroment for plain text.
```

tutorial.do.txt

```
===== 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
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 'docs/manual/manual.do.txt' file (see the
https://doconce.googlecode.com/hg/trunk/docs/demos/manual/index.html<demo
page> for various formats of this document).


# Example on including another Doconce file:

# #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.
https://doconce.googlecode.com/hg/trunk/docs/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.

===== Dependencies =====

Doconce depends on the Python package
http://code.google.com/p/preprocess/<preprocess>. To make LaTeX
documents (without going through the reStructuredText format) you also
need http://code.google.com/p/ptex2tex<ptex2tex> and some style files
that ptex2tex potentially makes use of. Going from reStructuredText
to formats such as XML, OpenOffice, HTML, and LaTeX requires
http://docutils.sourceforge.net/<docutils>. Making Sphinx documents
requires of course http://sphinx.pocoo.org<sphinx>.

======= Warning/Disclaimer =======

Doconce can be viewed is a unified interface to a variety of
typesetting formats. This interface is minimal in the sense that a
lot of typesetting features are not supported, for example, footnotes
and bibliography. For many documents the simple Doconce format is
sufficient, while in other cases you need more sophisticated
formats. Then you can just filter the Doconce text to a more
approprite format and continue working in this format only. For
example, reStructuredText is a good alternative: it is more tagged
than Doconce and cannot be filtered to plain, untagged text, or wiki,
and the LaTeX output is not at all as clean, but it also has a lot
more typesetting and tagging features than Doconce.

Doconce: Document Once, Include Anywhere

Hans Petter Langtangen[1,2]

[1]Simula Research Laboratory

[2]University of Oslo

September 10, 2010

- 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 at some later stage eventually go with a particular format?
- Do you find it problematic that 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 place, and include it anywhere?

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

1 The Doconce Concept

Doconce is two things:

1. Doconce is a working strategy for documenting software in a single place and avoiding duplication of information. The slogan is: "Write once, include anywhere". This requires that what you write can be transformed to many different formats for a variety of documents (manuals, tutorials, books, doc strings, source code documentation, etc.).
2. Doconce is a simple and minimally tagged markup language that can be used for the above purpose. That is, the Doconce format look like ordinary ASCII text (much like what you would use in an email), but the text can be transformed to numerous other formats, including HTML, Wiki, LaTeX, PDF, reStructuredText (reST), Sphinx, Epytext, and also plain text (where non-obvious formatting/tags are removed for clear reading in, e.g., emails). From reStructuredText you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the latter to RTF and MS Word.

Doconce was particularly written for the following sample applications:

- Large books written in LaTeX, but where many pieces (computer demos, projects, examples) can be written in Doconce to appear in other contexts in other formats, including plain HTML, Sphinx, or MS Word.
- Software documentation, primarily Python doc strings, which one wants to appear as plain untagged text for viewing in Pydoc, as reStructured-Text for use with Sphinx, as wiki text when publishing the software at googlecode.com, and as LaTeX integrated in, e.g., a master's thesis.
- Quick memos, which start as plain text in email, then some small amount of Doconce tagging is added, before the memos can appear as MS Word documents or in wikis.

2 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. For example,

- bullet lists arise from lines starting with an asterisk,
- *emphasized words* are surrounded by asterisks,
- **words in boldface** are surrounded by underscores,
- words from computer code are enclosed in back quotes and then typeset verbatim,
- blocks of computer code can easily be included, also from source files,
- blocks of LaTeX mathematics can easily be included,
- there is support for both LaTeX and text-like inline mathematics,
- figures with captions, URLs with links, labels and references are supported,
- comments can be inserted throughout the text,
- a preprocessor (much like the C preprocessor) is integrated so other documents (files) can be included and large portions of text can be defined in or out of the text.

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 an 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 http://folk.uio.no/hpl<hpl>.
Just a file link goes like 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
Chapter ref{my:first:sec}.

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    |
  |--------------------------------|
```

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. Just a file link goes like 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 Chapter 2.1.

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

```
$\nu = \sin(x)$|$v = sin(x)$
```

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside bt! and et! (begin tex / end tex) instructions. The result looks like this:

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

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

Of course, such blocks only looks nice in LaTeX. The raw LaTeX syntax appears in all other 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. Such blocks look like

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

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

It is possible to add a specification of a (ptex2tex-style) environment for typesetting the verbatim code block, e.g., bc xxx! where `xxx` is an identifier like `pycod` for code snippet in Python, `sys` for terminal session, etc. When Doconce is filtered to LaTeX, these identifiers are used as in ptex2tex and defined in a configuration file `.ptext2tex.cfg`, while when filtering to Sphinx, one can have a comment line in the Doconce file for mapping the identifiers to legal language names for Sphinx (which equals the legal language names for Pygments):

```
# sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console
```

By default, `pro` and `cod` are `python`, `sys` is `console`, while `xpro` and `xcod` are computer language specific for `x` in `f` (Fortran), `c` (C), `cpp` (C++), and `py` (Python).

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!). A complete file is typeset with bc pro!, while a part of a file is copied into a bc cod! environment. What `pro` and `cod` mean is then defined through a `.ptex2tex.cfg` file for LaTeX and a `sphinx code-blocks` comment for Sphinx.

Another document can be included by writing `#include "mynote.do.txt"` on a line starting with (another) hash sign. 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 the right writing enviroment for plain text.

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 `docs/manual/manual.do.txt` file (see the demo page for various formats of this document).

3 From Doconce to Other Formats

Transformation of a Doconce document to various other formats applies the script `doconce2format`:

Terminal

```
Unix/DOS> doconce2format format mydoc.do.txt
```

The `preprocess` program is always used to preprocess the file first, and options to `preprocess` can be added after the filename. For example,

Terminal

```
Unix/DOS> doconce2format LaTeX mydoc.do.txt -Dextra_sections
```

The variable `FORMAT` is always defined as the current format when running `preprocess`. 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"`.

3.1 HTML

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

Terminal

```
Unix/DOS> doconce2format HTML mydoc.do.txt
```

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

3.2 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 `ptex2tex`:

Terminal

```
Unix/DOS> doconce2format LaTeX mydoc.do.txt
```

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

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

Terminal

```
Unix/DOS> ptex2tex mydoc
```

or just perform a plain copy,

Terminal

```
Unix/DOS> cp mydoc.p.tex mydoc.tex
```

Doconce generates a `.p.tex` file with some preprocessor macros. For example, to enable font Helvetica instead of the standard Computer Modern font,

Terminal

```
Unix/DOS> ptex2tex -DHELVETICA mydoc
```

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. The standard LaTeX "maketitle" heading is also available through

Terminal

```
Unix/DOS> ptex2tex -DTRAD_LATEX_HEADING mydoc
```

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 sys! command in the Doconce source you can insert verbatim block styles as defined in your `.ptex2tex.cfg` file, e.g., bc sys cod! for a code snippet, where `cod` is set to a certain environment in `.ptex2tex.cfg` (e.g., `CodeIntended`). There are over 30 styles to choose from.

Step 3. Compile `mydoc.tex` with `latex -shell-escape` and create the PDF file:

Terminal

```
Unix/DOS> latex -shell-escape mydoc
Unix/DOS> latex -shell-escape mydoc
Unix/DOS> makeindex mydoc   # if index
Unix/DOS> bibitem mydoc     # if bibliography
Unix/DOS> latex -shell-escape mydoc
Unix/DOS> dvipdf mydoc
```

The `-shell-escape` option is required because `ptex2tex` inserts an include of the `minted.sty` style, which applies the `pygments` to format code, and this program cannot be run from `latex` without the `-shell-escape` option.

3.3 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

```
Unix/DOS> doconce2format plain mydoc.do.txt  # results in mydoc.txt
```

3.4 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

```
Unix/DOS> doconce2format rst mydoc.do.txt
```

We may now produce various other formats:

Terminal

```
Unix/DOS> rst2html.py  mydoc.rst > mydoc.html # HTML
Unix/DOS> rst2latex.py mydoc.rst > mydoc.tex  # LaTeX
Unix/DOS> rst2xml.py   mydoc.rst > mydoc.xml  # XML
Unix/DOS> 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. That is, one can easily go from Doconce to Microsoft Word.

3.5 Sphinx

Sphinx documents can be created from a Doconce source in a few steps.

Step 1. Translate Doconce into the Sphinx dialect of the reStructuredText format:

Terminal

```
Unix/DOS> doconce2format sphinx mydoc.do.txt
```

Step 2. Create a Sphinx root directory with a `conf.py` file, 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
sphinx-rootdir
n

_
Name of My Sphinx Document
Author
version
version
.rst
index
n
y
```

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

Step 3. Move the `tutorial.rst` file to the Sphinx root directory:

Terminal

```
Unix/DOS> mv 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 directory to `sphinx-rootdir` (if all figures are located in a subdirectory).

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:

Terminal

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

Many other formats are also possible.

Step 6. View the result:

Terminal

```
Unix/DOS> 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.

3.6 Google Code Wiki

There are several different wiki dialects, but Doconce only support the one used by Google Code. The transformation to this format, called `gwiki` to explicitly mark it as the Google Code dialect, is done by

Terminal

```
Unix/DOS> doconce2format gwiki mydoc.do.txt
```

You can then open a new wiki page for your Google Code project, copy the `mydoc.gwiki` output file from `doconce2format` and paste the file contents into the wiki page. Press **Preview** or **Save Page** to see the formatted result.

When the Doconce file contains figures, each figure filename must 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).

3.7 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.

3.8 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.

3.9 Dependencies

Doconce depends on the Python package preprocess. To make LaTeX documents (without going through the reStructuredText format) you also need ptex2tex and some style files that ptex2tex potentially makes use of. Going from reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX requires docutils. Making Sphinx documents requires of course sphinx.

4 Warning/Disclaimer

Doconce can be viewed is a unified interface to a variety of typesetting formats. This interface is minimal in the sense that a lot of typesetting features are not supported, for example, footnotes and bibliography. For many documents the simple Doconce format is sufficient, while in other cases you need more sophisticated formats. Then you can just filter the Doconce text to a more approprite format and continue working in this format only. For example, reStructuredText is a good alternative: it is more tagged than Doconce and cannot be filtered to plain, untagged text, or wiki, and the LaTeX output is not at all as clean, but it also has a lot more typesetting and tagging features than Doconce.

Doconce: Document Once, Include Anywhere

Author: Hans Petter Langtangen

Date: September 10, 2010

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

The Doconce Concept

Doconce is two things:

1. Doconce is a working strategy for documenting software in a single place and avoiding duplication of information. The slogan is: "Write once, include anywhere". This requires that what you write can be transformed to many different formats for a variety of documents (manuals, tutorials, books, doc strings, source code documentation, etc.).
2. Doconce is a simple and minimally tagged markup language that can be used for the above purpose. That is, the Doconce format look like ordinary ASCII text (much like what you would use in an email), but the text can be transformed to numerous other formats, including HTML, Wiki, LaTeX, PDF, reStructuredText (reST), Sphinx, Epytext, and also plain text (where non-obvious formatting/tags are removed for clear reading in, e.g., emails). From reStructuredText you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the latter to RTF and MS Word.

Doconce was particularly written for the following sample applications:

- Large books written in LaTeX, but where many pieces (computer demos, projects, examples) can be written in Doconce to appear in other contexts in other formats, including plain HTML, Sphinx, or MS Word.
- Software documentation, primarily Python doc strings, which one wants to appear as plain untagged text for viewing in Pydoc, as reStructuredText for use with Sphinx, as wiki text when publishing the software at googlecode.com, and as LaTeX integrated in, e.g., a master's thesis.
- Quick memos, which start as plain text in email, then some small amount of Doconce tagging is added, before the memos can appear as MS Word documents or in wikis.

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. For example,

- bullet lists arise from lines starting with an asterisk,
- *emphasized words* are surrounded by asterisks,

- **words in boldface** are surrounded by underscores,
- words from computer code are enclosed in back quotes and then typeset verbatim,
- blocks of computer code can easily be included, also from source files,
- blocks of LaTeX mathematics can easily be included,
- there is support for both LaTeX and text-like inline mathematics,
- figures with captions, URLs with links, labels and references are supported,
- comments can be inserted throughout the text,
- a preprocessor (much like the C preprocessor) is integrated so other documents (files) can be included and large portions of text can be defined in or out of the text.

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 an 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 http://folk.uio.no/hpl<hpl>.
Just a file link goes like 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
Chapter ref{my:first:sec}.

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    |
  |--------------------------------|
```

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. Just a file link goes like 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 chapter A Subsection with Sample Text.

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:

```
$\nu = \sin(x)$|$v = sin(x)$
```

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside `!bt` and `!et` (begin tex / end tex) instructions. The result looks like this:

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

Of course, such blocks only looks nice in LaTeX. The raw LaTeX syntax appears in all other 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. Such blocks look like:

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

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

It is possible to add a specification of a (ptex2tex-style) environment for typesetting the verbatim code block, e.g., `!bc xxx` where `xxx` is an identifier like `pycod` for code snippet in Python, `sys` for terminal session, etc. When Doconce is filtered to LaTeX, these identifiers are used as in ptex2tex and defined in a configuration file `.ptext2tex.cfg`, while when filtering to Sphinx, one can have a comment line in the Doconce file for mapping the identifiers to legal language names for Sphinx (which equals the legal language names for Pygments):

```
# sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console
```

By default, `pro` and `cod` are `python`, `sys` is `console`, while `xpro` and `xcod` are computer language specific for `x` in `f` (Fortran), `c` (C), `cpp` (C++), and `py` (Python). .. `rb` (Ruby), `pl` (Perl), and `sh` (Unix shell).

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!). A complete file is typeset with `!bc pro`, while a part of a file is copied into a `!bc cod` environment. What `pro` and `cod` mean is then defined through a `.ptex2tex.cfg` file for LaTeX and a `sphinx code-blocks` comment for Sphinx.

Another document can be included by writing `#include "mynote.do.txt"` on a line starting with (another) hash sign. 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 the right writing enviroment for plain text.

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 `docs/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 to various other formats applies the script `doconce2format`:

```
Unix/DOS> doconce2format format mydoc.do.txt
```

The `preprocess` program is always used to preprocess the file first, and options to `preprocess` can be added after the filename. For example:

```
Unix/DOS> doconce2format LaTeX mydoc.do.txt -Dextra_sections
```

The variable `FORMAT` is always defined as the current format when running `preprocess`. 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"`.

HTML

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

```
Unix/DOS> doconce2format HTML mydoc.do.txt
```

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

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.

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

```
Unix/DOS> doconce2format LaTeX mydoc.do.txt
```

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.

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

```
Unix/DOS> ptex2tex mydoc
```

or just perform a plain copy:

```
Unix/DOS> cp mydoc.p.tex mydoc.tex
```

Doconce generates a `.p.tex` file with some preprocessor macros. For example, to enable font Helvetica instead of the standard Computer Modern font:

```
Unix/DOS> ptex2tex -DHELVETICA mydoc
```

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. The standard LaTeX "maketitle" heading is also available through:

```
Unix/DOS> ptex2tex -DTRAD_LATEX_HEADING mydoc
```

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 sys` command in the Doconce source you can insert verbatim block styles as defined in your `.ptex2tex.cfg` file, e.g., `!bc sys cod` for a code snippet, where `cod` is set to a certain environment in `.ptex2tex.cfg` (e.g., `CodeIntended`). There are over 30 styles to choose from.

Step 3. Compile `mydoc.tex` with `latex -shell-escape` and create the PDF file:

```
Unix/DOS> latex -shell-escape mydoc
Unix/DOS> latex -shell-escape mydoc
Unix/DOS> makeindex mydoc   # if index
Unix/DOS> bibitem mydoc     # if bibliography
Unix/DOS> latex -shell-escape mydoc
Unix/DOS> dvipdf mydoc
```

The `-shell-escape` option is required because `ptex2tex` inserts an include of the `minted.sty` style, which applies the `pygments` to format code, and this program cannot be run from `latex` without the `-shell-escape` option.

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:

```
Unix/DOS> doconce2format 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`:

```
Unix/DOS> doconce2format rst mydoc.do.txt
```

We may now produce various other formats:

```
Unix/DOS> rst2html.py  mydoc.rst > mydoc.html # HTML
Unix/DOS> rst2latex.py mydoc.rst > mydoc.tex  # LaTeX
Unix/DOS> rst2xml.py   mydoc.rst > mydoc.xml  # XML
Unix/DOS> 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. That is, one can easily go from Doconce to Microsoft Word.

Sphinx

Sphinx documents can be created from a Doconce source in a few steps.

Step 1. Translate Doconce into the Sphinx dialect of the reStructuredText format:

```
Unix/DOS> doconce2format sphinx mydoc.do.txt
```

Step 2. Create a Sphinx root directory with a `conf.py` file, 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
n
y
n
n
y
y
y
EOF
```

Step 3. Move the `tutorial.rst` file to the Sphinx root directory:

```
Unix/DOS> mv 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 directory to `sphinx-rootdir` (if all figures are located in a subdirectory).

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
```

Many other formats are also possible.

Step 6. View the result:

```
Unix/DOS> 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.

Google Code Wiki

There are several different wiki dialects, but Doconce only support the one used by Google Code. The transformation to this format, called `gwiki` to explicitly mark it as the Google Code dialect, is done by:

```
Unix/DOS> doconce2format gwiki mydoc.do.txt
```

You can then open a new wiki page for your Google Code project, copy the `mydoc.gwiki` output file from `doconce2format` and paste the file contents into the wiki page. Press **Preview** or **Save Page** to see the formatted result.

When the Doconce file contains figures, each figure filename must 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.

Dependencies

Doconce depends on the Python package preprocess. To make LaTeX documents (without going through the reStructuredText format) you also need ptex2tex and some style files that ptex2tex potentially makes use of. Going from reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX requires docutils. Making Sphinx documents requires of course sphinx.

Warning/Disclaimer

Doconce can be viewed is a unified interface to a variety of typesetting formats. This interface is minimal in the sense that a lot of typesetting features are not supported, for example, footnotes and bibliography. For many documents the simple Doconce format is sufficient, while in other cases you need more sophisticated formats. Then you can just filter the Doconce text to a more approprite format and continue working in this format only. For example, reStructuredText is a good alternative: it is more tagged than Doconce and cannot be filtered to plain, untagged text, or wiki, and the LaTeX output is not at all as clean, but it also has a lot more typesetting and tagging features than Doconce.

Doconce Tutorial Documentation

Release 1.0

H. P. Langtangen

September 10, 2010

CONTENTS

Contents:

CHAPTER

ONE

DOCONCE: DOCUMENT ONCE, INCLUDE ANYWHERE

Author Hans Petter Langtangen

Date September 10, 2010

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

CHAPTER

TWO

THE DOCONCE CONCEPT

Doconce is two things:

1. Doconce is a working strategy for documenting software in a single place and avoiding duplication of information. The slogan is: "Write once, include anywhere". This requires that what you write can be transformed to many different formats for a variety of documents (manuals, tutorials, books, doc strings, source code documentation, etc.).
2. Doconce is a simple and minimally tagged markup language that can be used for the above purpose. That is, the Doconce format look like ordinary ASCII text (much like what you would use in an email), but the text can be transformed to numerous other formats, including HTML, Wiki, LaTeX, PDF, reStructuredText (reST), Sphinx, Epytext, and also plain text (where non-obvious formatting/tags are removed for clear reading in, e.g., emails). From reStructuredText you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the latter to RTF and MS Word.

Doconce was particularly written for the following sample applications:

- Large books written in LaTeX, but where many pieces (computer demos, projects, examples) can be written in Doconce to appear in other contexts in other formats, including plain HTML, Sphinx, or MS Word.
- Software documentation, primarily Python doc strings, which one wants to appear as plain untagged text for viewing in Pydoc, as reStructuredText for use with Sphinx, as wiki text when publishing the software at google-code.com, and as LaTeX integrated in, e.g., a master's thesis.
- Quick memos, which start as plain text in email, then some small amount of Doconce tagging is added, before the memos can appear as MS Word documents or in wikis.

CHAPTER

THREE

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. For example,

- bullet lists arise from lines starting with an asterisk,
- *emphasized words* are surrounded by asterisks,
- **words in boldface** are surrounded by underscores,
- words from computer code are enclosed in back quotes and then typeset verbatim,
- blocks of computer code can easily be included, also from source files,
- blocks of LaTeX mathematics can easily be included,
- there is support for both LaTeX and text-like inline mathematics,
- figures with captions, URLs with links, labels and references are supported,
- comments can be inserted throughout the text,
- a preprocessor (much like the C preprocessor) is integrated so other documents (files) can be included and large portions of text can be defined in or out of the text.

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 an 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 http://folk.uio.no/hpl<hpl>.
Just a file link goes like 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
```

```
Chapter ref{my:first:sec}.

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    |
  |-------------------------------|
```

The Doconce text above results in the following little document:

3.1 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. Just a file link goes like 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 chapter *A Subsection with Sample Text*.

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

3.2 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

```
$\nu = \sin(x)$|$v = sin(x)$
```

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside `!bt` and `!et` (begin tex / end tex) instructions. The result looks like this:

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

Of course, such blocks only looks nice in LaTeX. The raw LaTeX syntax appears in all other 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. Such blocks look like

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

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

It is possible to add a specification of a (ptex2tex-style) environment for typesetting the verbatim code block, e.g., `!bc xxx` where `xxx` is an identifier like `pycod` for code snippet in Python, `sys` for terminal session, etc. When Doconce is filtered to LaTeX, these identifiers are used as in ptex2tex and defined in a configuration file `.ptext2tex.cfg`, while when filtering to Sphinx, one can have a comment line in the Doconce file for mapping the identifiers to legal language names for Sphinx (which equals the legal language names for Pygments):

```
# sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console
```

By default, `pro` and `cod` are `python`, `sys` is `console`, while `xpro` and `xcod` are computer language specific for `x` in `f` (Fortran), `c` (C), `cpp` (C++), and `py` (Python). .. `rb` (Ruby), `pl` (Perl), and `sh` (Unix shell).

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!). A complete file is typeset with `!bc pro`, while a part of a file is copied into a `!bc cod` environment. What `pro` and `cod` mean is then defined through a `.ptex2tex.cfg` file for LaTeX and a `sphinx code-blocks` comment for Sphinx.

Another document can be included by writing `#include "mynote.do.txt"` on a line starting with (another) hash sign. 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 the right writing enviroment for plain text.

3.3 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 `docs/manual/manual.do.txt` file (see the demo page for various formats of this document).

CHAPTER

FOUR

FROM DOCONCE TO OTHER FORMATS

Transformation of a Doconce document to various other formats applies the script `doconce2format`:

```
Unix/DOS> doconce2format format mydoc.do.txt
```

The `preprocess` program is always used to preprocess the file first, and options to `preprocess` can be added after the filename. For example,

```
Unix/DOS> doconce2format LaTeX mydoc.do.txt -Dextra_sections
```

The variable `FORMAT` is always defined as the current format when running `preprocess`. 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"`.

4.1 HTML

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

```
Unix/DOS> doconce2format HTML mydoc.do.txt
```

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

4.2 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.

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

```
Unix/DOS> doconce2format LaTeX mydoc.do.txt
```

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.

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

```
Unix/DOS> ptex2tex mydoc
```

or just perform a plain copy,

```
Unix/DOS> cp mydoc.p.tex mydoc.tex
```

Doconce generates a `.p.tex` file with some preprocessor macros. For example, to enable font Helvetica instead of the standard Computer Modern font,

```
Unix/DOS> ptex2tex -DHELVETICA mydoc
```

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. The standard LaTeX "maketitle" heading is also available through

```
Unix/DOS> ptex2tex -DTRAD_LATEX_HEADING mydoc
```

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 sys` command in the Doconce source you can insert verbatim block styles as defined in your `.ptex2tex.cfg` file, e.g., `!bc sys cod` for a code snippet, where `cod` is set to a certain environment in `.ptex2tex.cfg` (e.g., `CodeIntended`). There are over 30 styles to choose from.

Step 3. Compile `mydoc.tex` with `latex -shell-escape` and create the PDF file:

```
Unix/DOS> latex -shell-escape mydoc
Unix/DOS> latex -shell-escape mydoc
Unix/DOS> makeindex mydoc   # if index
Unix/DOS> bibitem mydoc     # if bibliography
Unix/DOS> latex -shell-escape mydoc
Unix/DOS> dvipdf mydoc
```

The `-shell-escape` option is required because `ptex2tex` inserts an include of the `minted.sty` style, which applies the `pygments` to format code, and this program cannot be run from `latex` without the `-shell-escape` option.

4.3 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:

```
Unix/DOS> doconce2format plain mydoc.do.txt  # results in mydoc.txt
```

4.4 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`:

```
Unix/DOS> doconce2format rst mydoc.do.txt
```

We may now produce various other formats:

```
Unix/DOS> rst2html.py  mydoc.rst > mydoc.html # HTML
Unix/DOS> rst2latex.py mydoc.rst > mydoc.tex  # LaTeX
Unix/DOS> rst2xml.py   mydoc.rst > mydoc.xml  # XML
Unix/DOS> 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. That is, one can easily go from Doconce to Microsoft Word.

4.5 Sphinx

Sphinx documents can be created from a Doconce source in a few steps.

Step 1. Translate Doconce into the Sphinx dialect of the reStructuredText format:

```
Unix/DOS> doconce2format sphinx mydoc.do.txt
```

Step 2. Create a Sphinx root directory with a `conf.py` file, 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
n
y
n
n
y
y
y
EOF
```

Step 3. Move the `tutorial.rst` file to the Sphinx root directory:

```
Unix/DOS> mv 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 directory to `sphinx-rootdir` (if all figures are located in a subdirectory).

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
```

Many other formats are also possible.

Step 6. View the result:

```
Unix/DOS> 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.

4.6 Google Code Wiki

There are several different wiki dialects, but Doconce only support the one used by Google Code. The transformation to this format, called `gwiki` to explicitly mark it as the Google Code dialect, is done by

```
Unix/DOS> doconce2format gwiki mydoc.do.txt
```

You can then open a new wiki page for your Google Code project, copy the `mydoc.gwiki` output file from `doconce2format` and paste the file contents into the wiki page. Press **Preview** or **Save Page** to see the formatted result.

When the Doconce file contains figures, each figure filename must 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).

4.7 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.

4.8 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.

4.9 Dependencies

Doconce depends on the Python package preprocess. To make LaTeX documents (without going through the reStructuredText format) you also need ptex2tex and some style files that ptex2tex potentially makes use of. Going

from reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX requires docutils. Making Sphinx documents requires of course sphinx.

CHAPTER

FIVE

WARNING/DISCLAIMER

Doconce can be viewed is a unified interface to a variety of typesetting formats. This interface is minimal in the sense that a lot of typesetting features are not supported, for example, footnotes and bibliography. For many documents the simple Doconce format is sufficient, while in other cases you need more sophisticated formats. Then you can just filter the Doconce text to a more approprite format and continue working in this format only. For example, reStructuredText is a good alternative: it is more tagged than Doconce and cannot be filtered to plain, untagged text, or wiki, and the LaTeX output is not at all as clean, but it also has a lot more typesetting and tagging features than Doconce.

CHAPTER

SIX

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

tutorial.txt

```
Doconce: Document Once, Include Anywhere
========================================

Hans Petter Langtangen [1, 2]

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


Date: September 10, 2010

 * 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 at some later stage eventually go
   with a particular format?

 * Do you find it problematic that 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
   place, and include it anywhere?

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


The Doconce Concept
===================

Doconce is two things:

 1. Doconce is a working strategy for documenting software in a single
    place and avoiding duplication of information. The slogan is:
    "Write once, include anywhere". This requires that what you write
    can be transformed to many different formats for a variety of
    documents (manuals, tutorials, books, doc strings, source code
    documentation, etc.).

 2. Doconce is a simple and minimally tagged markup language that can
    be used for the above purpose. That is, the Doconce format look
    like ordinary ASCII text (much like what you would use in an
    email), but the text can be transformed to numerous other formats,
    including HTML, Wiki, LaTeX, PDF, reStructuredText (reST), Sphinx,
    Epytext, and also plain text (where non-obvious formatting/tags are
    removed for clear reading in, e.g., emails). From reStructuredText
    you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the
    latter to RTF and MS Word.

Doconce was particularly written for the following sample applications:

  * Large books written in LaTeX, but where many pieces (computer demos,
    projects, examples) can be written in Doconce to appear in other
    contexts in other formats, including plain HTML, Sphinx, or MS Word.

  * Software documentation, primarily Python doc strings, which one wants
    to appear as plain untagged text for viewing in Pydoc, as reStructuredText
    for use with Sphinx, as wiki text when publishing the software at
    googlecode.com, and as LaTeX integrated in, e.g., a master's thesis.
```

tutorial.txt

```
  * Quick memos, which start as plain text in email, then some small
    amount of Doconce tagging is added, before the memos can appear as
    MS Word documents or in wikis.

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. For example,

  * bullet lists arise from lines starting with an asterisk,

  * *emphasized words* are surrounded by asterisks,

  * _words in boldface_ are surrounded by underscores,

  * words from computer code are enclosed in back quotes and
    then typeset verbatim,

  * blocks of computer code can easily be included, also from source files,

  * blocks of LaTeX mathematics can easily be included,

  * there is support for both LaTeX and text-like inline mathematics,

  * figures with captions, URLs with links, labels and references
    are supported,

  * comments can be inserted throughout the text,

  * a preprocessor (much like the C preprocessor) is integrated so
    other documents (files) can be included and large portions of text
    can be defined in or out of the text.

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 an 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 http://folk.uio.no/hpl<hpl>.
        Just a file link goes like URL:"tutorial.do.txt". References
        to sections may use logical names as labels (e.g., a "label" command rig
ht
        after the section title), as in the reference to
```

tutorial.txt

```
        Chapter ref{my:first:sec}.

        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    |
          |--------------------------------|


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).
Just a file link goes like 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 chapter "A Subsection with Sample Text".

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.
```

tutorial.txt

```
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.

Blocks of mathematics are better typeset with raw LaTeX, inside::

The result looks like this::

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

Of course, such blocks only looks nice in LaTeX. The raw
LaTeX syntax appears in all other 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  cod
        from math import sin, pi
        def myfunc(x):
            return sin(pi*x)

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


It is possible to add a specification of a (ptex2tex-style)
environment for typesetting the verbatim code block, e.g.::

where xxx is an identifier like pycod for code snippet in Python,
sys for terminal session, etc. When Doconce is filtered to LaTeX,
these identifiers are used as in ptex2tex and defined in a
configuration file .ptext2tex.cfg, while when filtering
to Sphinx, one can have a comment line in the Doconce file for
mapping the identifiers to legal language names for Sphinx (which equals
the legal language names for Pygments)::

         # sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console


By default, pro and cod are python, sys is console,
while xpro and xcod are computer language specific for x
in f (Fortran), c (C), cpp (C++), and py (Python).


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!). A complete file is typeset
with::
```

```
environment. What pro and cod mean is then defined through
a .ptex2tex.cfg file for LaTeX and a sphinx code-blocks
comment for Sphinx.

Another document can be included by writing #include "mynote.do.txt"
on a line starting with (another) hash sign.  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 the
right writing enviroment for plain text.


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::

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 docs/manual/manual.do.txt file (see the
demo
page (https://doconce.googlecode.com/hg/trunk/docs/demos/manual/index.html) for
various formats of this document).



From Doconce to Other Formats
=============================

Transformation of a Doconce document to various other
formats applies the script doconce2format::

        Unix/DOS> doconce2format format mydoc.do.txt
```

tutorial.txt

```
The preprocess program is always used to preprocess the file first,
and options to preprocess can be added after the filename. For example::

        Unix/DOS> doconce2format LaTeX mydoc.do.txt -Dextra_sections


The variable FORMAT is always defined as the current format when
running preprocess. 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".


HTML
----

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

        Unix/DOS> doconce2format HTML mydoc.do.txt


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

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
     ptex2tex::

        Unix/DOS> doconce2format LaTeX mydoc.do.txt


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.

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

        Unix/DOS> ptex2tex mydoc


or just perform a plain copy::

        Unix/DOS> cp mydoc.p.tex mydoc.tex


Doconce generates a .p.tex file with some preprocessor macros.
For example, to enable font Helvetica instead of the standard
Computer Modern font::

        Unix/DOS> ptex2tex -DHELVETICA mydoc
```

tutorial.txt

```
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. The standard LaTeX "maketitle" heading
is also available through::

        Unix/DOS> ptex2tex -DTRAD_LATEX_HEADING mydoc



The ptex2tex tool makes it possible to easily switch between many
different fancy formattings of computer or verbatim code in LaTeX
documents. After any::

insert verbatim block styles as defined in your .ptex2tex.cfg
file, e.g.::

a certain environment in .ptex2tex.cfg (e.g., CodeIntended).
There are over 30 styles to choose from.

*Step 3.* Compile mydoc.tex with latex -shell-escape
and create the PDF file::

        Unix/DOS> latex -shell-escape mydoc
        Unix/DOS> latex -shell-escape mydoc
        Unix/DOS> makeindex mydoc   # if index
        Unix/DOS> bibitem mydoc     # if bibliography
        Unix/DOS> latex -shell-escape mydoc
        Unix/DOS> dvipdf mydoc


The -shell-escape option is required because ptex2tex inserts an include
of the minted.sty style, which applies the pygments to format code,
and this program cannot be run from latex without the -shell-escape option.


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::

        Unix/DOS> doconce2format 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::

        Unix/DOS> doconce2format rst mydoc.do.txt


We may now produce various other formats::

        Unix/DOS> rst2html.py  mydoc.rst > mydoc.html # HTML
```

```
        Unix/DOS> rst2latex.py mydoc.rst > mydoc.tex  # LaTeX
        Unix/DOS> rst2xml.py   mydoc.rst > mydoc.xml  # XML
        Unix/DOS> 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.
That is, one can easily go from Doconce to Microsoft Word.

Sphinx
------

Sphinx documents can be created from a Doconce source in a few steps.

*Step 1.* Translate Doconce into the Sphinx dialect of
the reStructuredText format::

        Unix/DOS> doconce2format sphinx mydoc.do.txt



*Step 2.* Create a Sphinx root directory with a conf.py file,
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
        n
        y
        n
        n
        y
        y
        y
        EOF



*Step 3.* Move the tutorial.rst file to the Sphinx root directory::

        Unix/DOS> mv 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 directory to sphinx-rootdir (if all figures
are located in a subdirectory).

*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


Many other formats are also possible.

*Step 6.* View the result::

        Unix/DOS> firefox _build/html/index.html



Note that verbatim code blocks can be typeset in a variety of ways
depending the argument that follows::

(code-block:: python in Sphinx syntax) and cppcod gives C++, but
all such arguments can be customized both for Sphinx and LaTeX output.



Google Code Wiki
----------------

There are several different wiki dialects, but Doconce only support the
one used by Google Code (http://code.google.com/p/support/wiki/WikiSyntax).
The transformation to this format, called gwiki to explicitly mark
it as the Google Code dialect, is done by::

        Unix/DOS> doconce2format gwiki mydoc.do.txt


You can then open a new wiki page for your Google Code project, copy
the mydoc.gwiki output file from doconce2format and paste the
file contents into the wiki page. Press _Preview_ or _Save Page_ to
see the formatted result.

When the Doconce file contains figures, each figure filename must 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).
```

tutorial.txt

```
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/trunk/docs/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.

Dependencies
------------

Doconce depends on the Python package
preprocess (http://code.google.com/p/preprocess/).  To make LaTeX
documents (without going through the reStructuredText format) you also
need ptex2tex (http://code.google.com/p/ptex2tex) and some style files
that ptex2tex potentially makes use of.  Going from reStructuredText
to formats such as XML, OpenOffice, HTML, and LaTeX requires
docutils (http://docutils.sourceforge.net/).  Making Sphinx documents
requires of course sphinx (http://sphinx.pocoo.org).


Warning/Disclaimer
==================

Doconce can be viewed is a unified interface to a variety of
typesetting formats.  This interface is minimal in the sense that a
lot of typesetting features are not supported, for example, footnotes
and bibliography. For many documents the simple Doconce format is
sufficient, while in other cases you need more sophisticated
formats. Then you can just filter the Doconce text to a more
```

```
approprite format and continue working in this format only.  For
example, reStructuredText is a good alternative: it is more tagged
than Doconce and cannot be filtered to plain, untagged text, or wiki,
and the LaTeX output is not at all as clean, but it also has a lot
more typesetting and tagging features than Doconce.
```

tutorial.epytext

```
TITLE: Doconce: Document Once, Include Anywhere
BY: Hans Petter Langtangen (Simula Research Laboratory, and University of Oslo)D
ATE: September 10, 2010



 − 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 at some later stage eventually go
   with a particular format?
 − Do you find it problematic that 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
   place, and include it anywhere?

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


The Doconce Concept
===================

Doconce is two things:

 1. Doconce is a working strategy for documenting software in a single
    place and avoiding duplication of information. The slogan is:
    "Write once, include anywhere". This requires that what you write
    can be transformed to many different formats for a variety of
    documents (manuals, tutorials, books, doc strings, source code
    documentation, etc.).
 2. Doconce is a simple and minimally tagged markup language that can
    be used for the above purpose. That is, the Doconce format look
    like ordinary ASCII text (much like what you would use in an
    email), but the text can be transformed to numerous other formats,
    including HTML, Wiki, LaTeX, PDF, reStructuredText (reST), Sphinx,
    Epytext, and also plain text (where non−obvious formatting/tags are
    removed for clear reading in, e.g., emails). From reStructuredText
    you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the
    latter to RTF and MS Word.

Doconce was particularly written for the following sample applications:

  − Large books written in LaTeX, but where many pieces (computer demos,
    projects, examples) can be written in Doconce to appear in other
    contexts in other formats, including plain HTML, Sphinx, or MS Word.
  − Software documentation, primarily Python doc strings, which one wants
    to appear as plain untagged text for viewing in Pydoc, as reStructuredText
    for use with Sphinx, as wiki text when publishing the software at
    googlecode.com, and as LaTeX integrated in, e.g., a master's thesis.
  − Quick memos, which start as plain text in email, then some small
    amount of Doconce tagging is added, before the memos can appear as
    MS Word documents or in wikis.

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. For example,

  - bullet lists arise from lines starting with an asterisk,
  - I{emphasized words} are surrounded by asterisks,
  - B{words in boldface} are surrounded by underscores,
  - words from computer code are enclosed in back quotes and
    then typeset verbatim,
  - blocks of computer code can easily be included, also from source files,
  - blocks of LaTeX mathematics can easily be included,
  - there is support for both LaTeX and text-like inline mathematics,
  - figures with captions, URLs with links, labels and references
    are supported,
  - comments can be inserted throughout the text,
  - a preprocessor (much like the C preprocessor) is integrated so
    other documents (files) can be included and large portions of text
    can be defined in or out of the text.

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 an 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 http://folk.uio.no/hpl<hpl>.
        Just a file link goes like URL:"tutorial.do.txt". References
        to sections may use logical names as labels (e.g., a "label" command rig
ht
        after the section title), as in the reference to
        Chapter ref{my:first:sec}.

        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    |
          |--------------------------------|


The Doconce text above results in the following little document:

A Subsection with Sample Text
-----------------------------
```

tutorial.epytext

```
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>}.
Just a file link goes like 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 chapter "A Subsection with Sample Text".

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.

Blocks of mathematics are better typeset with raw LaTeX, inside
C{!bt} and C{!et} (begin tex / end tex) instructions.
The result looks like this::


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

tutorial.epytext

```
Of course, such blocks only looks nice in LaTeX. The raw
LaTeX syntax appears in all other 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. Such blocks look like::

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

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


It is possible to add a specification of a (ptex2tex-style)
environment for typesetting the verbatim code block, e.g., C{!bc xxx}
where C{xxx} is an identifier like C{pycod} for code snippet in Python,
C{sys} for terminal session, etc. When Doconce is filtered to LaTeX,
these identifiers are used as in ptex2tex and defined in a
configuration file C{.ptext2tex.cfg}, while when filtering
to Sphinx, one can have a comment line in the Doconce file for
mapping the identifiers to legal language names for Sphinx (which equals
the legal language names for Pygments)::

         # sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console


By default, C{pro} and C{cod} are C{python}, C{sys} is C{console},
while C{xpro} and C{xcod} are computer language specific for C{x}
in C{f} (Fortran), C{c} (C), C{cpp} (C++), and C{py} (Python).


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!). A complete file is typeset
with C{!bc pro}, while a part of a file is copied into a C{!bc cod}
environment. What C{pro} and C{cod} mean is then defined through
a C{.ptex2tex.cfg} file for LaTeX and a C{sphinx code-blocks}
comment for Sphinx.

Another document can be included by writing C{#include "mynote.do.txt"}
on a line starting with (another) hash sign.  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 the
right writing enviroment for plain text.


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{docs/manual/manual.do.txt} file (see the
U{demo
page<https://doconce.googlecode.com/hg/trunk/docs/demos/manual/index.html>} for
various formats of this document).



From Doconce to Other Formats
=============================

Transformation of a Doconce document to various other
formats applies the script C{doconce2format}:
!bc   sys
        Unix/DOS> doconce2format format mydoc.do.txt


The C{preprocess} program is always used to preprocess the file first,
and options to C{preprocess} can be added after the filename. For example::

        Unix/DOS> doconce2format LaTeX mydoc.do.txt −Dextra_sections


The variable C{FORMAT} is always defined as the current format when
running C{preprocess}. 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"}.


HTML
----

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

        Unix/DOS> doconce2format HTML mydoc.do.txt
```

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

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
     C{ptex2tex}:
!bc   sys
        Unix/DOS> doconce2format LaTeX mydoc.do.txt


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.

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

        Unix/DOS> ptex2tex mydoc


or just perform a plain copy::

        Unix/DOS> cp mydoc.p.tex mydoc.tex


Doconce generates a C{.p.tex} file with some preprocessor macros.
For example, to enable font Helvetica instead of the standard
Computer Modern font::

        Unix/DOS> ptex2tex -DHELVETICA mydoc


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. The standard LaTeX "maketitle" heading
is also available through::

        Unix/DOS> ptex2tex -DTRAD_LATEX_HEADING mydoc



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 sys} command in the Doconce source you can
insert verbatim block styles as defined in your C{.ptex2tex.cfg}
file, e.g., C{!bc sys cod} for a code snippet, where C{cod} is set to
a certain environment in C{.ptex2tex.cfg} (e.g., C{CodeIntended}).
There are over 30 styles to choose from.

I{Step 3.} Compile C{mydoc.tex} with C{latex -shell-escape}
and create the PDF file::

        Unix/DOS> latex -shell-escape mydoc
```

```
        Unix/DOS> latex -shell-escape mydoc
        Unix/DOS> makeindex mydoc   # if index
        Unix/DOS> bibitem mydoc     # if bibliography
        Unix/DOS> latex -shell-escape mydoc
        Unix/DOS> dvipdf mydoc


The C{-shell-escape} option is required because C{ptex2tex} inserts an include
of the C{minted.sty} style, which applies the C{pygments} to format code,
and this program cannot be run from C{latex} without the C{-shell-escape} option
.


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::

        Unix/DOS> doconce2format 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}:
!bc   sys
        Unix/DOS> doconce2format rst mydoc.do.txt


We may now produce various other formats::

        Unix/DOS> rst2html.py  mydoc.rst > mydoc.html # HTML
        Unix/DOS> rst2latex.py mydoc.rst > mydoc.tex  # LaTeX
        Unix/DOS> rst2xml.py   mydoc.rst > mydoc.xml  # XML
        Unix/DOS> 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.
That is, one can easily go from Doconce to Microsoft Word.

Sphinx
------

Sphinx documents can be created from a Doconce source in a few steps.

I{Step 1.} Translate Doconce into the Sphinx dialect of
the reStructuredText format::

        Unix/DOS> doconce2format sphinx mydoc.do.txt



I{Step 2.} Create a Sphinx root directory with a C{conf.py} file,
```

```
either manually or by using the interactive C{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
        n
        y
        n
        n
        y
        y
        y
        EOF



I{Step 3.} Move the C{tutorial.rst} file to the Sphinx root directory::

        Unix/DOS> mv 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 directory to C{sphinx-rootdir} (if all figures
are located in a subdirectory).

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


Many other formats are also possible.
```

```
I{Step 6.} View the result::

        Unix/DOS> 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.



Google Code Wiki
----------------

There are several different wiki dialects, but Doconce only support the
one used by U{Google Code<http://code.google.com/p/support/wiki/WikiSyntax>}.
The transformation to this format, called C{gwiki} to explicitly mark
it as the Google Code dialect, is done by::

        Unix/DOS> doconce2format gwiki mydoc.do.txt


You can then open a new wiki page for your Google Code project, copy
the C{mydoc.gwiki} output file from C{doconce2format} and paste the
file contents into the wiki page. Press B{Preview} or B{Save Page} to
see the formatted result.

When the Doconce file contains figures, each figure filename must 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
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.


Demos
-----

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

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

tutorial.epytext

```
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/trunk/docs/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.

Dependencies
------------

Doconce depends on the Python package
U{preprocess<http://code.google.com/p/preprocess/>}.  To make LaTeX
documents (without going through the reStructuredText format) you also
need U{ptex2tex<http://code.google.com/p/ptex2tex>} and some style files
that ptex2tex potentially makes use of.  Going from reStructuredText
to formats such as XML, OpenOffice, HTML, and LaTeX requires
U{docutils<http://docutils.sourceforge.net/>}.  Making Sphinx documents
requires of course U{sphinx<http://sphinx.pocoo.org>}.


Warning/Disclaimer
==================

Doconce can be viewed is a unified interface to a variety of
typesetting formats.  This interface is minimal in the sense that a
lot of typesetting features are not supported, for example, footnotes
and bibliography. For many documents the simple Doconce format is
sufficient, while in other cases you need more sophisticated
formats. Then you can just filter the Doconce text to a more
approprite format and continue working in this format only.  For
example, reStructuredText is a good alternative: it is more tagged
than Doconce and cannot be filtered to plain, untagged text, or wiki,
and the LaTeX output is not at all as clean, but it also has a lot
more typesetting and tagging features than Doconce.
```

tutorial.gwiki

```
#summary Doconce: Document Once, Include Anywhere
<wiki:toc max_depth="2" />
By *Hans Petter Langtangen*

==== September 10, 2010 ====

<wiki:comment> lines beginning with # are comment lines </wiki:comment>


 * 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.  Woul
d it be convenient to   start with some very simple text-like format that easily
 converts   to the formats listed above, and at some later stage eventually go
 with a particular format?
 * Do you find it problematic that you have the same information   scattered aro
und in different documents in different typesetting   formats? Would it be a goo
d idea to write things once, in one   place, and include it anywhere?

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



== The Doconce Concept ==

Doconce is two things:


 # Doconce is a working strategy for documenting software in a single    place a
nd avoiding duplication of information. The slogan is:    "Write once, include a
nywhere". This requires that what you write    can be transformed to many differ
ent formats for a variety of    documents (manuals, tutorials, books, doc string
s, source code    documentation, etc.).
 # Doconce is a simple and minimally tagged markup language that can    be used
for the above purpose. That is, the Doconce format look    like ordinary ASCII t
ext (much like what you would use in an    email), but the text can be transform
ed to numerous other formats,    including HTML, Wiki, LaTeX, PDF, reStructuredT
ext (reST), Sphinx,    Epytext, and also plain text (where non-obvious formattin
g/tags are    removed for clear reading in, e.g., emails). From reStructuredText
    you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the    latter to R
TF and MS Word.

Doconce was particularly written for the following sample applications:


  * Large books written in LaTeX, but where many pieces (computer demos,    proj
ects, examples) can be written in Doconce to appear in other    contexts in othe
r formats, including plain HTML, Sphinx, or MS Word.
  * Software documentation, primarily Python doc strings, which one wants    to
appear as plain untagged text for viewing in Pydoc, as reStructuredText    for u
se with Sphinx, as wiki text when publishing the software at    googlecode.com,
and as LaTeX integrated in, e.g., a master's thesis.
  * Quick memos, which start as plain text in email, then some small    amount o
f Doconce tagging is added, before the memos can appear as    MS Word documents
or in wikis.



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

tutorial.gwiki

```
Doconce text looks like ordinary text, but there are some almost invisible
text constructions that allow you to control the formating. For example,


  * bullet lists arise from lines starting with an asterisk,
  * *emphasized words* are surrounded by asterisks,
  * *words in boldface* are surrounded by underscores,
  * words from computer code are enclosed in back quotes and     then typeset ve
rbatim,
  * blocks of computer code can easily be included, also from source files,
  * blocks of LaTeX mathematics can easily be included,
  * there is support for both LaTeX and text-like inline mathematics,
  * figures with captions, URLs with links, labels and references    are support
ed,
  * comments can be inserted throughout the text,
  * a preprocessor (much like the C preprocessor) is integrated so    other docu
ments (files) can be included and large portions of text    can be defined in or
 out of the text.

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 an 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 http://folk.uio.no/hpl<hpl>.
Just a file link goes like 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
Chapter ref{my:first:sec}.

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    |
  |--------------------------------|
}}}
The Doconce text above results in the following little document:

==== A Subsection with Sample Text ====
```

tutorial.gwiki

```
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].
Just a file link goes like 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 chapter [#A_Subsection_with_Sample_Text].

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
{{{
$\nu = \sin(x)$|$v = sin(x)$
}}}
The pipe symbol acts as a delimiter between LaTeX code and the plain text
version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside
‘!bt‘ and ‘!et‘ (begin tex / end tex) instructions.
The result looks like this:
{{{
\begin{eqnarray}
{\partial u\over\partial t} &=& \nabla^2 u + f,\label{myeq1}\\
{\partial v\over\partial t} &=& \nabla\cdot(q(u)\nabla v) + g
\end{eqnarray}
}}}
Of course, such blocks only looks nice in LaTeX. The raw
LaTeX syntax appears in all other 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. Such blocks look like
{{{
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
}}}
It is possible to add a specification of a (ptex2tex-style)
environment for typesetting the verbatim code block, e.g., '!bc xxx'
where 'xxx' is an identifier like 'pycod' for code snippet in Python,
'sys' for terminal session, etc. When Doconce is filtered to LaTeX,
these identifiers are used as in ptex2tex and defined in a
configuration file '.ptext2tex.cfg', while when filtering
to Sphinx, one can have a comment line in the Doconce file for
mapping the identifiers to legal language names for Sphinx (which equals
the legal language names for Pygments):
{{{
 # sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console
}}}
By default, 'pro' and 'cod' are 'python', 'sys' is 'console',
while 'xpro' and 'xcod' are computer language specific for 'x'
in 'f' (Fortran), 'c' (C), 'cpp' (C++), and 'py' (Python).
<wiki:comment> 'rb' (Ruby), 'pl' (Perl), and 'sh' (Unix shell). </wiki:comment>

<wiki:comment> (Any sphinx code-block comment, whether inside verbatim code </wi
ki:comment>
<wiki:comment> blocks or outside, yields a mapping between bc arguments </wiki:c
omment>
<wiki:comment> and computer languages. In case of muliple definitions, the </wik
i:comment>
<wiki:comment> first one is used.) </wiki:comment>

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!). A complete file is typeset
with '!bc pro', while a part of a file is copied into a '!bc cod'
environment. What 'pro' and 'cod' mean is then defined through
a '.ptex2tex.cfg' file for LaTeX and a 'sphinx code-blocks'
comment for Sphinx.

Another document can be included by writing '#include "mynote.do.txt"'
on a line starting with (another) hash sign.  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 the
right writing enviroment for plain text.

==== 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 ‘docs/manual/manual.do.txt‘ file (see the
[https://doconce.googlecode.com/hg/trunk/docs/demos/manual/index.html demo
page] for various formats of this document).


<wiki:comment> Example on including another Doconce file: </wiki:comment>



== From Doconce to Other Formats ==

Transformation of a Doconce document to various other
formats applies the script ‘doconce2format‘:
{{{
Unix/DOS> doconce2format format mydoc.do.txt
}}}
The ‘preprocess‘ program is always used to preprocess the file first,
and options to ‘preprocess‘ can be added after the filename. For example,
{{{
Unix/DOS> doconce2format LaTeX mydoc.do.txt −Dextra_sections
}}}
The variable ‘FORMAT‘ is always defined as the current format when
running ‘preprocess‘. 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"‘.

==== HTML ====

Making an HTML version of a Doconce file ‘mydoc.do.txt‘
is performed by
{{{
Unix/DOS> doconce2format HTML mydoc.do.txt
}}}
The resulting file ‘mydoc.html‘ can be loaded into any web browser for viewing.

==== 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
     ‘ptex2tex‘:
{{{
Unix/DOS> doconce2format LaTeX mydoc.do.txt
}}}
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.

*Step 2.* Run ‘ptex2tex‘ (if you have it) to make a standard LaTeX file,
{{{
Unix/DOS> ptex2tex mydoc
}}}
or just perform a plain copy,
{{{
Unix/DOS> cp mydoc.p.tex mydoc.tex
}}}
Doconce generates a ‘.p.tex‘ file with some preprocessor macros.
For example, to enable font Helvetica instead of the standard
Computer Modern font,
{{{
Unix/DOS> ptex2tex −DHELVETICA mydoc
}}}
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. The standard LaTeX "maketitle" heading
is also available through
{{{
Unix/DOS> ptex2tex −DTRAD_LATEX_HEADING mydoc
}}}

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 sys‘ command in the Doconce source you can
insert verbatim block styles as defined in your ‘.ptex2tex.cfg‘
file, e.g., ‘!bc sys cod‘ for a code snippet, where ‘cod‘ is set to
a certain environment in ‘.ptex2tex.cfg‘ (e.g., ‘CodeIntended‘).
There are over 30 styles to choose from.

*Step 3.* Compile ‘mydoc.tex‘ with ‘latex −shell−escape‘
and create the PDF file:
{{{
Unix/DOS> latex −shell−escape mydoc
Unix/DOS> latex −shell−escape mydoc
Unix/DOS> makeindex mydoc   # if index
Unix/DOS> bibitem mydoc     # if bibliography
Unix/DOS> latex −shell−escape mydoc
Unix/DOS> dvipdf mydoc
}}}
The ‘−shell−escape‘ option is required because ‘ptex2tex‘ inserts an include
of the ‘minted.sty‘ style, which applies the ‘pygments‘ to format code,
```

```
and this program cannot be run from ‘latex‘ without the ‘−shell−escape‘ option.

==== 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:
{{{
Unix/DOS> doconce2format 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‘:
{{{
Unix/DOS> doconce2format rst mydoc.do.txt
}}}
We may now produce various other formats:
{{{
Unix/DOS> rst2html.py  mydoc.rst > mydoc.html # HTML
Unix/DOS> rst2latex.py mydoc.rst > mydoc.tex  # LaTeX
Unix/DOS> rst2xml.py   mydoc.rst > mydoc.xml  # XML
Unix/DOS> 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.
That is, one can easily go from Doconce to Microsoft Word.

==== Sphinx ====

Sphinx documents can be created from a Doconce source in a few steps.

*Step 1.* Translate Doconce into the Sphinx dialect of
the reStructuredText format:
{{{
Unix/DOS> doconce2format sphinx mydoc.do.txt
}}}

*Step 2.* Create a Sphinx root directory with a ‘conf.py‘ file,
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
n
y
n
n
y
y
y
EOF
}}}

*Step 3.* Move the ‘tutorial.rst‘ file to the Sphinx root directory:
{{{
Unix/DOS> mv 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 directory to ‘sphinx-rootdir‘ (if all figures
are located in a subdirectory).

*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
}}}
Many other formats are also possible.

*Step 6.* View the result:
{{{
Unix/DOS> 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:comment> Desired extension: sphinx can utilize a "pycod" or "c++cod" </wik
i:comment>
<wiki:comment> instruction as currently done in latex for ptex2tex and write </w
iki:comment>
<wiki:comment> out the right code block name accordingly. </wiki:comment>

==== Google Code Wiki ====

There are several different wiki dialects, but Doconce only support the
one used by [http://code.google.com/p/support/wiki/WikiSyntax Google Code].
The transformation to this format, called ‘gwiki‘ to explicitly mark
```

```
it as the Google Code dialect, is done by
{{{
Unix/DOS> doconce2format gwiki mydoc.do.txt
}}}
You can then open a new wiki page for your Google Code project, copy
the ‘mydoc.gwiki‘ output file from ‘doconce2format‘ and paste the
file contents into the wiki page. Press *Preview* or *Save Page* to
see the formatted result.

When the Doconce file contains figures, each figure filename must 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.
[https://doconce.googlecode.com/hg/trunk/docs/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.

==== Dependencies ====

Doconce depends on the Python package
[http://code.google.com/p/preprocess/ preprocess].  To make LaTeX
documents (without going through the reStructuredText format) you also
need [http://code.google.com/p/ptex2tex ptex2tex] and some style files
that ptex2tex potentially makes use of.  Going from reStructuredText
to formats such as XML, OpenOffice, HTML, and LaTeX requires
[http://docutils.sourceforge.net/ docutils].  Making Sphinx documents
requires of course [http://sphinx.pocoo.org sphinx].
```

tutorial.gwiki

```
== Warning/Disclaimer ==

Doconce can be viewed is a unified interface to a variety of
typesetting formats.  This interface is minimal in the sense that a
lot of typesetting features are not supported, for example, footnotes
and bibliography. For many documents the simple Doconce format is
sufficient, while in other cases you need more sophisticated
formats. Then you can just filter the Doconce text to a more
approprite format and continue working in this format only.  For
example, reStructuredText is a good alternative: it is more tagged
than Doconce and cannot be filtered to plain, untagged text, or wiki,
and the LaTeX output is not at all as clean, but it also has a lot
more typesetting and tagging features than Doconce.
```

Doconce Tutorial Documentation

Release 1.0

H. P. Langtangen

September 10, 2010

CONTENTS

Contents:

CHAPTER

ONE

DOCONCE: DOCUMENT ONCE, INCLUDE ANYWHERE

Author Hans Petter Langtangen

Date September 10, 2010

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

CHAPTER

TWO

THE DOCONCE CONCEPT

Doconce is two things:

1. Doconce is a working strategy for documenting software in a single place and avoiding duplication of information. The slogan is: "Write once, include anywhere". This requires that what you write can be transformed to many different formats for a variety of documents (manuals, tutorials, books, doc strings, source code documentation, etc.).
2. Doconce is a simple and minimally tagged markup language that can be used for the above purpose. That is, the Doconce format look like ordinary ASCII text (much like what you would use in an email), but the text can be transformed to numerous other formats, including HTML, Wiki, LaTeX, PDF, reStructuredText (reST), Sphinx, Epytext, and also plain text (where non-obvious formatting/tags are removed for clear reading in, e.g., emails). From reStructuredText you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the latter to RTF and MS Word.

Doconce was particularly written for the following sample applications:

- Large books written in LaTeX, but where many pieces (computer demos, projects, examples) can be written in Doconce to appear in other contexts in other formats, including plain HTML, Sphinx, or MS Word.
- Software documentation, primarily Python doc strings, which one wants to appear as plain untagged text for viewing in Pydoc, as reStructuredText for use with Sphinx, as wiki text when publishing the software at google-code.com, and as LaTeX integrated in, e.g., a master's thesis.
- Quick memos, which start as plain text in email, then some small amount of Doconce tagging is added, before the memos can appear as MS Word documents or in wikis.

CHAPTER

THREE

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. For example,

- bullet lists arise from lines starting with an asterisk,
- *emphasized words* are surrounded by asterisks,
- **words in boldface** are surrounded by underscores,
- words from computer code are enclosed in back quotes and then typeset verbatim,
- blocks of computer code can easily be included, also from source files,
- blocks of LaTeX mathematics can easily be included,
- there is support for both LaTeX and text-like inline mathematics,
- figures with captions, URLs with links, labels and references are supported,
- comments can be inserted throughout the text,
- a preprocessor (much like the C preprocessor) is integrated so other documents (files) can be included and large portions of text can be defined in or out of the text.

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 an 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 http://folk.uio.no/hpl<hpl>.
Just a file link goes like 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
```

```
Chapter ref{my:first:sec}.

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    |
  |-------------------------------|
```

The Doconce text above results in the following little document:

3.1 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. Just a file link goes like 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 chapter *A Subsection with Sample Text*.

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

3.2 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

```
$\nu = \sin(x)$|$v = sin(x)$
```

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside `!bt` and `!et` (begin tex / end tex) instructions. The result looks like this:

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

Of course, such blocks only looks nice in LaTeX. The raw LaTeX syntax appears in all other 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. Such blocks look like

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

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

It is possible to add a specification of a (ptex2tex-style) environment for typesetting the verbatim code block, e.g., `!bc xxx` where `xxx` is an identifier like `pycod` for code snippet in Python, `sys` for terminal session, etc. When Doconce is filtered to LaTeX, these identifiers are used as in ptex2tex and defined in a configuration file `.ptext2tex.cfg`, while when filtering to Sphinx, one can have a comment line in the Doconce file for mapping the identifiers to legal language names for Sphinx (which equals the legal language names for Pygments):

```
# sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console
```

By default, `pro` and `cod` are `python`, `sys` is `console`, while `xpro` and `xcod` are computer language specific for `x` in `f` (Fortran), `c` (C), `cpp` (C++), and `py` (Python). .. `rb` (Ruby), `pl` (Perl), and `sh` (Unix shell).

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!). A complete file is typeset with `!bc pro`, while a part of a file is copied into a `!bc cod` environment. What `pro` and `cod` mean is then defined through a `.ptex2tex.cfg` file for LaTeX and a `sphinx code-blocks` comment for Sphinx.

Another document can be included by writing `#include "mynote.do.txt"` on a line starting with (another) hash sign. 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 the right writing enviroment for plain text.

3.3 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 `docs/manual/manual.do.txt` file (see the demo page for various formats of this document).

CHAPTER

FOUR

FROM DOCONCE TO OTHER FORMATS

Transformation of a Doconce document to various other formats applies the script `doconce2format`:

```
Unix/DOS> doconce2format format mydoc.do.txt
```

The `preprocess` program is always used to preprocess the file first, and options to `preprocess` can be added after the filename. For example,

```
Unix/DOS> doconce2format LaTeX mydoc.do.txt -Dextra_sections
```

The variable `FORMAT` is always defined as the current format when running `preprocess`. 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"`.

4.1 HTML

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

```
Unix/DOS> doconce2format HTML mydoc.do.txt
```

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

4.2 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.

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

```
Unix/DOS> doconce2format LaTeX mydoc.do.txt
```

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.

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

```
Unix/DOS> ptex2tex mydoc
```

or just perform a plain copy,

```
Unix/DOS> cp mydoc.p.tex mydoc.tex
```

Doconce generates a `.p.tex` file with some preprocessor macros. For example, to enable font Helvetica instead of the standard Computer Modern font,

```
Unix/DOS> ptex2tex -DHELVETICA mydoc
```

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. The standard LaTeX "maketitle" heading is also available through

```
Unix/DOS> ptex2tex -DTRAD_LATEX_HEADING mydoc
```

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 sys` command in the Doconce source you can insert verbatim block styles as defined in your `.ptex2tex.cfg` file, e.g., `!bc sys cod` for a code snippet, where `cod` is set to a certain environment in `.ptex2tex.cfg` (e.g., `CodeIntended`). There are over 30 styles to choose from.

Step 3. Compile `mydoc.tex` with `latex -shell-escape` and create the PDF file:

```
Unix/DOS> latex -shell-escape mydoc
Unix/DOS> latex -shell-escape mydoc
Unix/DOS> makeindex mydoc   # if index
Unix/DOS> bibitem mydoc     # if bibliography
Unix/DOS> latex -shell-escape mydoc
Unix/DOS> dvipdf mydoc
```

The `-shell-escape` option is required because `ptex2tex` inserts an include of the `minted.sty` style, which applies the `pygments` to format code, and this program cannot be run from `latex` without the `-shell-escape` option.

4.3 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:

```
Unix/DOS> doconce2format plain mydoc.do.txt  # results in mydoc.txt
```

4.4 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`:

```
Unix/DOS> doconce2format rst mydoc.do.txt
```

We may now produce various other formats:

```
Unix/DOS> rst2html.py  mydoc.rst > mydoc.html # HTML
Unix/DOS> rst2latex.py mydoc.rst > mydoc.tex  # LaTeX
Unix/DOS> rst2xml.py   mydoc.rst > mydoc.xml  # XML
Unix/DOS> 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. That is, one can easily go from Doconce to Microsoft Word.

4.5 Sphinx

Sphinx documents can be created from a Doconce source in a few steps.

Step 1. Translate Doconce into the Sphinx dialect of the reStructuredText format:

```
Unix/DOS> doconce2format sphinx mydoc.do.txt
```

Step 2. Create a Sphinx root directory with a `conf.py` file, 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
n
y
n
n
y
y
y
EOF
```

Step 3. Move the `tutorial.rst` file to the Sphinx root directory:

```
Unix/DOS> mv 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 directory to `sphinx-rootdir` (if all figures are located in a subdirectory).

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
```

Many other formats are also possible.

Step 6. View the result:

```
Unix/DOS> 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.

4.6 Google Code Wiki

There are several different wiki dialects, but Doconce only support the one used by Google Code. The transformation to this format, called `gwiki` to explicitly mark it as the Google Code dialect, is done by

```
Unix/DOS> doconce2format gwiki mydoc.do.txt
```

You can then open a new wiki page for your Google Code project, copy the `mydoc.gwiki` output file from `doconce2format` and paste the file contents into the wiki page. Press **Preview** or **Save Page** to see the formatted result.

When the Doconce file contains figures, each figure filename must 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).

4.7 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.

4.8 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.

4.9 Dependencies

Doconce depends on the Python package preprocess. To make LaTeX documents (without going through the reStructuredText format) you also need ptex2tex and some style files that ptex2tex potentially makes use of. Going

from reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX requires docutils. Making Sphinx documents requires of course sphinx.

CHAPTER

FIVE

WARNING/DISCLAIMER

Doconce can be viewed is a unified interface to a variety of typesetting formats. This interface is minimal in the sense that a lot of typesetting features are not supported, for example, footnotes and bibliography. For many documents the simple Doconce format is sufficient, while in other cases you need more sophisticated formats. Then you can just filter the Doconce text to a more approprite format and continue working in this format only. For example, reStructuredText is a good alternative: it is more tagged than Doconce and cannot be filtered to plain, untagged text, or wiki, and the LaTeX output is not at all as clean, but it also has a lot more typesetting and tagging features than Doconce.

CHAPTER

SIX

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
//EN//XML" "http://docutils.sourceforge.net/docs/ref/docutils.dtd">
<!-- Generated by Docutils 0.8 -->
<document source="tutorial.rst"><section ids="doconce-document-once-include-anyw
here" names="doconce:\ document\ once,\ include\ anywhere"><title>Doconce: Docum
ent Once, Include Anywhere</title><field_list><field><field_name>Author</field_n
ame><field_body><paragraph>Hans Petter Langtangen</paragraph></field_body></fiel
d><field><field_name>Date</field_name><field_body><paragraph>September 10, 2010<
/paragraph></field_body></field></field_list><comment xml:space="preserve">lines
 beginning with # are comment lines

* 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 at some later stage eventually go
  with a particular format?

* Do you find it problematic that 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
  place, and include it anywhere?</comment><paragraph>If any of these questions
are of interest, you should keep on reading.</paragraph></section><section ids="
the-doconce-concept" names="the\ doconce\ concept"><title>The Doconce Concept</t
itle><paragraph>Doconce is two things:</paragraph><block_quote><enumerated_list
enumtype="arabic" prefix="" suffix="."><list_item><paragraph>Doconce is a workin
g strategy for documenting software in a single
place and avoiding duplication of information. The slogan is:
&amp;quot;Write once, include anywhere&amp;quot;. This requires that what you write
can be transformed to many different formats for a variety of
documents (manuals, tutorials, books, doc strings, source code
documentation, etc.).</paragraph></list_item><list_item><paragraph>Doconce is a
simple and minimally tagged markup language that can
be used for the above purpose. That is, the Doconce format look
like ordinary ASCII text (much like what you would use in an
email), but the text can be transformed to numerous other formats,
including HTML, Wiki, LaTeX, PDF, reStructuredText (reST), Sphinx,
Epytext, and also plain text (where non-obvious formatting/tags are
removed for clear reading in, e.g., emails). From reStructuredText
you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the
latter to RTF and MS Word.</paragraph></list_item></enumerated_list></block_quot
e><paragraph>Doconce was particularly written for the following sample applicati
ons:</paragraph><block_quote><bullet_list bullet="*"><list_item><paragraph>Large
 books written in LaTeX, but where many pieces (computer demos,
projects, examples) can be written in Doconce to appear in other
contexts in other formats, including plain HTML, Sphinx, or MS Word.</paragraph>
</list_item><list_item><paragraph>Software documentation, primarily Python doc s
trings, which one wants
to appear as plain untagged text for viewing in Pydoc, as reStructuredText
for use with Sphinx, as wiki text when publishing the software at
googlecode.com, and as LaTeX integrated in, e.g., a master's thesis.</paragraph>
</list_item><list_item><paragraph>Quick memos, which start as plain text in emai
l, then some small
amount of Doconce tagging is added, before the memos can appear as
MS Word documents or in wikis.</paragraph></list_item></bullet_list></block_quot
e></section><section ids="what-does-doconce-look-like" names="what\ does\ doconc
e\ look\ like?"><title>What Does Doconce Look Like?</title><paragraph>Doconce te
```

tutorial.xml

```
xt looks like ordinary text, but there are some almost invisible
text constructions that allow you to control the formating. For example,</paragr
aph><block_quote><bullet_list bullet="*"><list_item><paragraph>bullet lists aris
e from lines starting with an asterisk,</paragraph></list_item><list_item><parag
raph><emphasis>emphasized words</emphasis> are surrounded by asterisks,</paragra
ph></list_item><list_item><paragraph><strong>words in boldface</strong> are surr
ounded by underscores,</paragraph></list_item><list_item><paragraph>words from c
omputer code are enclosed in back quotes and
then typeset verbatim,</paragraph></list_item><list_item><paragraph>blocks of co
mputer code can easily be included, also from source files,</paragraph></list_it
em><list_item><paragraph>blocks of LaTeX mathematics can easily be included,</pa
ragraph></list_item><list_item><paragraph>there is support for both LaTeX and te
xt-like inline mathematics,</paragraph></list_item><list_item><paragraph>figures
 with captions, URLs with links, labels and references
are supported,</paragraph></list_item><list_item><paragraph>comments can be inse
rted throughout the text,</paragraph></list_item><list_item><paragraph>a preproc
essor (much like the C preprocessor) is integrated so
other documents (files) can be included and large portions of text
can be defined in or out of the text.</paragraph></list_item></bullet_list></blo
ck_quote><paragraph>Here is an example of some simple text written in the Doconc
e format:</paragraph><literal_block xml:space="preserve">===== 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 automatically numbered items instead of bullets,

  o item 1
  o item 2
  o item 3

URLs with a link word are possible, as in http://folk.uio.no/hpl&lt;hpl&gt;.
Just a file link goes like URL:&quot;tutorial.do.txt&quot;. References
to sections may use logical names as labels (e.g., a &quot;label&quot; command r
ight
after the section title), as in the reference to
Chapter ref{my:first:sec}.

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    |
  |-------------------------------|</literal_block><paragraph>The Doconce text
above results in the following little document:</paragraph><target refid="my-fir
st-sec"/><section ids="a-subsection-with-sample-text my-first-sec" names="a\ sub
section\ 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
```

tutorial.xml

```
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><parag
raph>item 3</paragraph></list_item></bullet_list></block_quote><paragraph>Lists
can also have numbered items instead of bullets, just use an <literal>o</literal
>
(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>item 2</paragraph></list_item><list_item><parag
raph>item 3</paragraph></list_item></enumerated_list></block_quote><paragraph>UR
Ls with a link word are possible, as in <reference name="hpl" refuri="http://fol
k.uio.no/hpl">hpl</reference><target ids="hpl" names="hpl" refuri="http://folk.u
io.no/hpl"/>.
Just a file link goes like <reference name="tutorial.do.txt" refuri="tutorial.do
.txt">tutorial.do.txt</reference><target ids="tutorial-do-txt" names="tutorial.d
o.txt" refuri="tutorial.do.txt"/>. References
to sections may use logical names as labels (e.g., a &amp;quot;label&amp;quot; command r
ight
after the section title), as in the reference to
the chapter <reference name="A Subsection with Sample Text" refid="a-subsection-
with-sample-text">A Subsection with Sample Text</reference>.</paragraph><paragra
ph>Tables are also supperted, e.g.,</paragraph><table><tgroup cols="3"><colspec
colwidth="12"/><colspec colwidth="12"/><colspec colwidth="12"/><thead><row><entr
y><paragraph>time</paragraph></entry><entry><paragraph>velocity</paragraph></ent
ry><entry><paragraph>acceleration</paragraph></entry></row></thead><tbody><row><
entry><paragraph>0.0</paragraph></entry><entry><paragraph>1.4186</paragraph></en
try><entry><paragraph>-5.01</paragraph></entry></row><row><entry><paragraph>2.0<
/paragraph></entry><entry><paragraph>1.376512</paragraph></entry><entry><paragra
ph>11.919</paragraph></entry></row><row><entry><paragraph>4.0</paragraph></entry
><entry><paragraph>1.1E+1</paragraph></entry><entry><paragraph>14.717624</paragr
aph></entry></row></tbody></tgroup></table></section><section ids="mathematics-a
nd-computer-code" names="mathematics\ and\ computer\ code"><title>Mathematics an
d 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">$\nu = \sin(x)$|$v =
sin(x)$</literal_block><paragraph>The pipe symbol acts as a delimiter between La
TeX code and the plain text
version of the formula.</paragraph><paragraph>Blocks of mathematics are better t
ypeset with raw LaTeX, inside
<literal>!bt</literal> and <literal>!et</literal> (begin tex / end tex) instruct
ions.
The result looks like this:</paragraph><literal_block xml:space="preserve">\begi
n{eqnarray}
{\partial u\over\partial t} &amp;=&amp; \nabla^2 u + f,\label{myeq1}\\
{\partial v\over\partial t} &amp;=&amp; \nabla\cdot(q(u)\nabla v) + g
\end{eqnarray}</literal_block><paragraph>Of course, such blocks only looks nice
in LaTeX. The raw
LaTeX syntax appears in all other formats (but can still be useful
for those who can read LaTeX syntax).</paragraph><paragraph>You can have blocks
of computer code, starting and ending with
<literal>!bc</literal> and <literal>!ec</literal> instructions, respectively. Su
ch blocks look like:</paragraph><literal_block xml:space="preserve">from math im
port sin, pi
def myfunc(x):
    return sin(pi*x)
```

tutorial.xml

```
import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)</literal_block><paragraph>It is po
ssible to add a specification of a (ptex2tex-style)
environment for typesetting the verbatim code block, e.g., <literal>!bc xxx</lit
eral>
where <literal>xxx</literal> is an identifier like <literal>pycod</literal> for
code snippet in Python,
<literal>sys</literal> for terminal session, etc. When Doconce is filtered to La
TeX,
these identifiers are used as in ptex2tex and defined in a
configuration file <literal>.ptext2tex.cfg</literal>, while when filtering
to Sphinx, one can have a comment line in the Doconce file for
mapping the identifiers to legal language names for Sphinx (which equals
the legal language names for Pygments):</paragraph><literal_block xml:space="pre
serve"># sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console</literal
_block><paragraph>By default, <literal>pro</literal> and <literal>cod</literal>
are <literal>python</literal>, <literal>sys</literal> is <literal>console</liter
al>,
while <literal>xpro</literal> and <literal>xcod</literal> are computer language
specific for <literal>x</literal>
in <literal>f</literal> (Fortran), <literal>c</literal> (C), <literal>cpp</liter
al> (C++), and <literal>py</literal> (Python).
.. <literal>rb</literal> (Ruby), <literal>pl</literal> (Perl), and <literal>sh</
literal> (Unix shell).</paragraph><comment xml:space="preserve">(Any sphinx code
-block comment, whether inside verbatim code</comment><comment xml:space="preser
ve">blocks or outside, yields a mapping between bc arguments</comment><comment x
ml:space="preserve">and computer languages. In case of muliple definitions, the<
/comment><comment xml:space="preserve">first one is used.)</comment><paragraph>O
ne 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!). A complete file is typeset
with <literal>!bc pro</literal>, while a part of a file is copied into a <litera
l>!bc cod</literal>
environment. What <literal>pro</literal> and <literal>cod</literal> mean is then
 defined through
a <literal>.ptex2tex.cfg</literal> file for LaTeX and a <literal>sphinx code-blo
cks</literal>
comment for Sphinx.</paragraph><paragraph>Another document can be included by wr
iting <literal>#include &amp;quot;mynote.do.txt&amp;quot;</literal>
on a line starting with (another) hash sign.  Doconce documents have
extension <literal>do.txt</literal>. The <literal>do</literal> part stands for d
oconce, while the
trailing <literal>.txt</literal> denotes a text document so that editors gives y
ou the
right writing enviroment for plain text.</paragraph><target refid="newcommands"/
></section><section ids="macros-newcommands-cross-references-index-and-bibliogra
phy newcommands" names="macros\ (newcommands),\ cross-references,\ index,\ and\
bibliography newcommands"><title>Macros (Newcommands), Cross-References, Index,
and Bibliography</title><paragraph>Doconce supports a type of macros via a LaTeX
-style <emphasis>newcommand</emphasis>
construction.  The newcommands defined in a file with name
<literal>newcommand_replace.tex</literal> are expanded when Doconce is filtered
to
other formats, except for LaTeX (since LaTeX performs the expansion
itself).  Newcommands in files with names <literal>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
```

tutorial.xml

```
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
ex</literal> 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
ions of
newcommands in the <literal>newcommands*.tex</literal> files <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
projects (manuals, books). For further details on functionality and
syntax we refer to the <literal>docs/manual/manual.do.txt</literal> file (see th
e
<reference name="demo page" refuri="https://doconce.googlecode.com/hg/trunk/docs
/demos/manual/index.html">demo
page</reference><target ids="demo−page" names="demo\ page" refuri="https://docon
ce.googlecode.com/hg/trunk/docs/demos/manual/index.html"/> for various formats o
f this document).</paragraph><comment xml:space="preserve">Example on including
another Doconce file:</comment></section></section><section ids="from−doconce−to
−other−formats" names="from\ doconce\ to\ other\ formats"><title>From Doconce to
 Other Formats</title><paragraph>Transformation of a Doconce document to various
 other
formats applies the script <literal>doconce2format</literal>:</paragraph><litera
l_block xml:space="preserve">Unix/DOS&amp;gt; doconce2format format mydoc.do.txt</li
teral_block><paragraph>The <literal>preprocess</literal> program is always used
to preprocess the file first,
and options to <literal>preprocess</literal> can be added after the filename. Fo
r example:</paragraph><literal_block xml:space="preserve">Unix/DOS&amp;gt; doconce2f
ormat LaTeX mydoc.do.txt −Dextra_sections</literal_block><paragraph>The variable
 <literal>FORMAT</literal> is always defined as the current format when
running <literal>preprocess</literal>. That is, in the last example, <literal>FO
RMAT</literal> is
defined as <literal>LaTeX</literal>. Inside the Doconce document one can then pe
rform
format specific actions through tests like <literal>#if FORMAT == &amp;quot;LaTeX&qu
ot;</literal>.</paragraph><section ids="html" names="html"><title>HTML</title><p
aragraph>Making an HTML version of a Doconce file <literal>mydoc.do.txt</literal
>
is performed by:</paragraph><literal_block xml:space="preserve">Unix/DOS&amp;gt; doc
once2format HTML mydoc.do.txt</literal_block><paragraph>The resulting file <lite
ral>mydoc.html</literal> can be loaded into any web browser for viewing.</paragr
aph></section><section ids="latex" names="latex"><title>LaTeX</title><paragraph>
Making a LaTeX file <literal>mydoc.tex</literal> from <literal>mydoc.do.txt</lit
eral> 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.</paragraph><definition_list><definition_list_item>
```

tutorial.xml

```
<term><emphasis>Step 1.</emphasis> Filter the doconce text to a pre-LaTeX form <
literal>mydoc.p.tex</literal> for</term><definition><paragraph><literal>ptex2tex
</literal>:</paragraph><literal_block xml:space="preserve">Unix/DOS&gt; doconce2
format LaTeX mydoc.do.txt</literal_block></definition></definition_list_item></d
efinition_list><paragraph>LaTeX-specific commands (&quot;newcommands&quot;) in m
ath formulas and similar
can be placed in files <literal>newcommands.tex</literal>, <literal>newcommands_
keep.tex</literal>, or
<literal>newcommands_replace.tex</literal> (see the section <reference name="Mac
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><emphasis>Step 2.</emph
asis> Run <literal>ptex2tex</literal> (if you have it) to make a standard LaTeX
file:</paragraph><literal_block xml:space="preserve">Unix/DOS&gt; ptex2tex mydoc
</literal_block><paragraph>or just perform a plain copy:</paragraph><literal_blo
ck xml:space="preserve">Unix/DOS&gt; cp mydoc.p.tex mydoc.tex</literal_block><pa
ragraph>Doconce generates a <literal>.p.tex</literal> file with some preprocesso
r macros.
For example, to enable font Helvetica instead of the standard
Computer Modern font:</paragraph><literal_block xml:space="preserve">Unix/DOS&gt
; ptex2tex -DHELVETICA mydoc</literal_block><paragraph>The title, authors, and d
ate are by default typeset in a non-standard
way to enable a nicer treatment of multiple authors having
institutions in common. The standard LaTeX &quot;maketitle&quot; heading
is also available through:</paragraph><literal_block xml:space="preserve">Unix/D
OS&gt; ptex2tex -DTRAD_LATEX_HEADING mydoc</literal_block><paragraph>The <litera
l>ptex2tex</literal> tool makes it possible to easily switch between many
different fancy formattings of computer or verbatim code in LaTeX
documents. After any <literal>!bc sys</literal> command in the Doconce source yo
u can
insert verbatim block styles as defined in your <literal>.ptex2tex.cfg</literal>
file, e.g., <literal>!bc sys cod</literal> for a code snippet, where <literal>co
d</literal> is set to
a certain environment in <literal>.ptex2tex.cfg</literal> (e.g., <literal>CodeIn
tended</literal>).
There are over 30 styles to choose from.</paragraph><paragraph><emphasis>Step 3.
</emphasis> Compile <literal>mydoc.tex</literal> with <literal>latex -shell-esca
pe</literal>
and create the PDF file:</paragraph><literal_block xml:space="preserve">Unix/DOS
&gt; latex -shell-escape mydoc
Unix/DOS&gt; latex -shell-escape mydoc
Unix/DOS&gt; makeindex mydoc   # if index
Unix/DOS&gt; bibitem mydoc     # if bibliography
Unix/DOS&gt; latex -shell-escape mydoc
Unix/DOS&gt; dvipdf mydoc</literal_block><paragraph>The <literal>-shell-escape</
literal> option is required because <literal>ptex2tex</literal> inserts an inclu
de
of the <literal>minted.sty</literal> style, which applies the <literal>pygments<
/literal> to format code,
and this program cannot be run from <literal>latex</literal> without the <litera
l>-shell-escape</literal> option.</paragraph></section><section ids="plain-ascii
-text" names="plain\ ascii\ text"><title>Plain ASCII Text</title><paragraph>We c
an go from Doconce &quot;back to&quot; 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">Unix/DOS&gt
; doconce2format plain mydoc.do.txt  # results in mydoc.txt</literal_block></sec
tion><section ids="restructuredtext" names="restructuredtext"><title>reStructure
```

tutorial.xml

```
dText</title><paragraph>Going from Doconce to reStructuredText gives a lot of po
ssibilities to
go to other formats. First we filter the Doconce text to a
reStructuredText file <literal>mydoc.rst</literal>:</paragraph><literal_block xm
l:space="preserve">Unix/DOS&gt; doconce2format rst mydoc.do.txt</literal_block><
paragraph>We may now produce various other formats:</paragraph><literal_block xm
l:space="preserve">Unix/DOS&gt; rst2html.py  mydoc.rst &gt; mydoc.html # HTML
Unix/DOS&gt; rst2latex.py mydoc.rst &gt; mydoc.tex  # LaTeX
Unix/DOS&gt; rst2xml.py   mydoc.rst &gt; mydoc.xml  # XML
Unix/DOS&gt; rst2odt.py   mydoc.rst &gt; 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.
That is, one can easily go from Doconce to Microsoft Word.</paragraph></section>
<section dupnames="sphinx" ids="sphinx"><title>Sphinx</title><paragraph>Sphinx d
ocuments can be created from a Doconce source in a few steps.</paragraph><paragr
aph><emphasis>Step 1.</emphasis> Translate Doconce into the Sphinx dialect of
the reStructuredText format:</paragraph><literal_block xml:space="preserve">Unix
/DOS&gt; doconce2format sphinx mydoc.do.txt</literal_block><paragraph><emphasis>
Step 2.</emphasis> Create a Sphinx root directory with a <literal>conf.py</liter
al> file,
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;&lt;EOF
sphinx-rootdir
n

_
Name of My Sphinx Document
Author
version
version
.rst
index
n
y
n
n
n
n
y
n
n
y
y
y
EOF</literal_block><paragraph><emphasis>Step 3.</emphasis> Move the <literal>tut
orial.rst</literal> file to the Sphinx root directory:</paragraph><literal_block
 xml:space="preserve">Unix/DOS&gt; mv mydoc.rst sphinx-rootdir</literal_block><p
aragraph>If you have figures in your document, the relative paths to those will
be invalid when you work with <literal>mydoc.rst</literal> in the <literal>sphin
x-rootdir</literal>
directory. Either edit <literal>mydoc.rst</literal> so that figure file paths ar
e correct,
or simply copy your figure directory to <literal>sphinx-rootdir</literal> (if al
l figures
are located in a subdirectory).</paragraph><paragraph><emphasis>Step 4.</emphasi
s> Edit the generated <literal>index.rst</literal> file so that <literal>mydoc.r
st</literal>
```

tutorial.xml

```
is included, i.e., add <literal>mydoc</literal> to the <literal>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><paragraph><emphasis>Step 5.</emphasis> 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>Many other formats are also possible.</parag
raph><paragraph><emphasis>Step 6.</emphasis> View the result:</paragraph><litera
l_block xml:space="preserve">Unix/DOS&gt; firefox _build/html/index.html</litera
l_block><paragraph>Note that verbatim code blocks can be typeset in a variety of
 ways
depending the argument that follows <literal>!bc</literal>: <literal>cod</litera
l> gives Python
(<literal>code-block:: python</literal> in Sphinx syntax) and <literal>cppcod</l
iteral> gives C++, but
all such arguments can be customized both for Sphinx and LaTeX output.</paragrap
h><comment xml:space="preserve">Desired extension: sphinx can utilize a &quot;py
cod&quot; or &quot;c++cod&quot;</comment><comment xml:space="preserve">instructi
on as currently done in latex for ptex2tex and write</comment><comment xml:space
="preserve">out the right code block name accordingly.</comment></section><secti
on ids="google-code-wiki" names="google\ code\ wiki"><title>Google Code Wiki</ti
tle><paragraph>There are several different wiki dialects, but Doconce only suppo
rt the
one used by <reference name="Google Code" refuri="http://code.google.com/p/suppo
rt/wiki/WikiSyntax">Google Code</reference><target ids="google-code" names="goog
le\ code" refuri="http://code.google.com/p/support/wiki/WikiSyntax"/>.
The transformation to this format, called <literal>gwiki</literal> to explicitly
 mark
it as the Google Code dialect, is done by:</paragraph><literal_block xml:space="
preserve">Unix/DOS&gt; doconce2format gwiki mydoc.do.txt</literal_block><paragra
ph>You can then open a new wiki page for your Google Code project, copy
the <literal>mydoc.gwiki</literal> output file from <literal>doconce2format</lit
eral> and paste the
file contents into the wiki page. Press <strong>Preview</strong> or <strong>Save
 Page</strong> to
see the formatted result.</paragraph><paragraph>When the Doconce file contains f
igures, each figure filename must 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).</paragraph></section><section ids="tweaking-th
e-doconce-output" names="tweaking\ the\ doconce\ output"><title>Tweaking the Doc
once Output</title><paragraph>Occasionally, one would like to tweak the output i
n 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
 know
if <literal>.eps</literal> or <literal>.png</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 <literal>make.sh</literal> files in <literal>docs/manual</literal
> and <literal>docs/tutorial</literal>
constitute comprehensive examples on how such scripts can be made.</paragraph></
```

tutorial.xml

```
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
aragraph>The file <literal>make.sh</literal> in the <literal>tutorial</literal>
directory of the
Doconce source code contains a demo of how to produce a variety of
formats.  The source of this tutorial, <literal>tutorial.do.txt</literal> is the
starting point.  Running <literal>make.sh</literal> and studying the various gen
erated
files and comparing them with the original <literal>tutorial.do.txt</literal> fi
le,
gives a quick introduction to how Doconce is used in a real case.
<reference name="Here" refuri="https://doconce.googlecode.com/hg/trunk/docs/demo
s/tutorial/index.html">Here</reference><target ids="here" names="here" refuri="h
ttps://doconce.googlecode.com/hg/trunk/docs/demos/tutorial/index.html"/>
is a sample of how this tutorial looks in different formats.</paragraph><paragra
ph>There is another demo in the <literal>docs/manual</literal> directory which
translates the more comprehensive documentation, <literal>manual.do.txt</literal
>, to
various formats. The <literal>make.sh</literal> script runs a set of translation
s.</paragraph></section><section ids="dependencies" names="dependencies"><title>
Dependencies</title><paragraph>Doconce depends on the Python package
<reference name="preprocess" refuri="http://code.google.com/p/preprocess/">prepr
ocess</reference><target ids="preprocess" names="preprocess" refuri="http://code
.google.com/p/preprocess/"/>.  To make LaTeX
documents (without going through the reStructuredText format) you also
need <reference name="ptex2tex" refuri="http://code.google.com/p/ptex2tex">ptex2
tex</reference><target ids="ptex2tex" names="ptex2tex" refuri="http://code.googl
e.com/p/ptex2tex"/> and some style files
that ptex2tex potentially makes use of.  Going from reStructuredText
to formats such as XML, OpenOffice, HTML, and LaTeX requires
<reference name="docutils" refuri="http://docutils.sourceforge.net/">docutils</r
eference><target ids="docutils" names="docutils" refuri="http://docutils.sourcef
orge.net/"/>.  Making Sphinx documents
requires of course <reference name="sphinx" refuri="http://sphinx.pocoo.org">sph
inx</reference><target ids="id1" names="sphinx" refuri="http://sphinx.pocoo.org"
/>.</paragraph></section></section><section ids="warning-disclaimer" names="warn
ing/disclaimer"><title>Warning/Disclaimer</title><paragraph>Doconce can be viewe
d is a unified interface to a variety of
typesetting formats.  This interface is minimal in the sense that a
lot of typesetting features are not supported, for example, footnotes
and bibliography. For many documents the simple Doconce format is
sufficient, while in other cases you need more sophisticated
formats. Then you can just filter the Doconce text to a more
approprite format and continue working in this format only.  For
example, reStructuredText is a good alternative: it is more tagged
than Doconce and cannot be filtered to plain, untagged text, or wiki,
and the LaTeX output is not at all as clean, but it also has a lot
more typesetting and tagging features than Doconce.</paragraph></section></docum
ent>
```