Open-source framework for documentation of scientific software written on MATLAB-compatible programming languages

Mikhail V. Konnik and James Welsh
School of Electrical Engineering and Computer Science, The University of Newcastle, Callaghan 2308 Australia

ABSTRACT
Numerical simulators for adaptive optics systems have become an essential tool for the research and development of the future advanced astronomical instruments. However, growing software code of the numerical simulator makes it difficult to continue to support the code itself. The problem of adequate documentation of the astronomical software for adaptive optics simulators may complicate the development since the documentation must contain up-to-date schemes and mathematical descriptions implemented in the software code. Although most modern programming environments like MATLAB or Octave have in-built documentation abilities, they are often insufficient for the description of a typical adaptive optics simulator code.

This paper describes a general cross-platform framework for the documentation of scientific software using open-source tools such as \LaTeX, mercurial, Doxygen, and Perl. Using the Perl script that translates M-files MATLAB comments into C-like, one can use Doxygen to generate and update the documentation for the scientific source code. The documentation generated by this framework contains the current code description with mathematical formulas, images, and bibliographical references.

A detailed description of the framework components is presented as well as the guidelines for the framework deployment. Examples of the code documentation for the scripts and functions of a MATLAB-based adaptive optics simulator are provided.

Keywords: software documentation, Doxygen, \LaTeX, numerical simulators

1. INTRODUCTION
The documentation is an essential part of any software code. This is especially true for scientific software that usually contains implementations of complicated mathematical models. The specific problem of the scientific software is that its documentation should contain mathematical formulas, graphical schemes, and bibliographic references to describe the implemented models. Without appropriate documentation, the problem of code support and further development becomes sometimes more difficult than the actual research and development.

The existing methods of the MATLAB-compatible code documentation rely on the MATLAB’s ability to output the short help\(^1\) if the comments are inserted at the beginning of the M-file.\(^2\) A major drawback of this method is that the output will be text-only, without pictures and mathematical formulas. Furthermore, such documentation is difficult to print or convert into other formats. Another approach is to use documentation tools such as Doxygen\(^3\) that allows to include mathematical formulas and pictures. However, the Doxygen does not support MATLAB source code comments directly. There were attempts\(^4,5\) to use the open-source parser Byson with Doxygen to translate the comments into a code that can be parsed by the Doxygen. However, the authors\(^4\) presented only an outline of such a method and concentrated mostly on documentation of MATLAB classes and not scripts or functions.

This tutorial paper provides the description of a complete open-source framework for the documentation of scripts and functions written on MATLAB and Octave. The described framework is the result of the development of MATLAB-based optical simulator software.\(^6\) The aim of the paper is to give the guidelines and descriptions

Further author information: Mikhail V. Konnik: mail@mvkonnik.info, James Welsh: James.Welsh@newcastle.edu.au
to use the Doxygen-based documentation system for the scientific software development. The description of the framework is done for Linux, but the installation steps are similar for Windows, Mac OS, and FreeBSD.

The novelty of the paper is the description of the whole framework based on open-source tools for the effective documentation of the scientific software written on MATLAB-compatible languages. The paper can be of interest for academic researchers and engineers dealing with scientific software development.

2. THE COMPONENTS OF THE DOCUMENTATION FRAMEWORK

The documentation framework consists of open-source tools that can be run on most of operation systems such as Linux, Mac OS, FreeBSD, and Windows. The framework contains the following components:

1. \LaTeX scientific typesetting language;
2. Doxygen software documentation generator;
3. distributed version control system (mercurial, git etc.).
4. the script on Perl language for on-the-fly translation of comments into the documentation.

Those components and their installation procedures will be briefly described in this section. The description is done for Debian GNU/Linux operating system; the steps are similar for other operation systems.

2.1 \LaTeX scientific typesetting language

Although the \LaTeX is a mature tool specifically for the academic community, office word processors like Microsoft Word or OpenOffice are still widely used. The problem is that the text typeset in word processors is usually not re-usable: one cannot use it for the documentation of the source code without loss of formatting and mathematical formulas. Such a problem is important since without appropriate documentation, it is often difficult to understand the implemented mathematical model in the source code. Contrary, the text typeset in \LaTeX with mathematical formulas, pictures, and bibliography can be easily re-used for making posters, presentations, books, and for the source code documentation itself without almost any changes. Therefore, the \LaTeX is necessary for writing a documentation for the scientific software in order to include the mathematical formulas, bibliographic citations and graphical schemes to clarify the implemented models in scientific software.

The installation of \LaTeX in Debian GNU/Linux is straightforward:

\$ aptitude install tex-common texlive-base texlive-base-bin texlive-common texlive-latex-base texlive-latex-recommended

There is a port of \LaTeX for Windows called MiKTeX (accessible from www.miktex.org), and Mac users will have to use MacTeX package (www.tug.org/mactex/).

2.2 Doxygen software documentation generator

Doxygen is an open-source documentation generation tool that supports C/C++, Java, Python, VHDL, C#, and other programming languages. Doxygen is able to generate the documentation out of comments and structured description of the software routine. The output can be in \LaTeX, HTML, XML, RTF, and UNIX “man” pages.

However, Doxygen does not directly support MATLAB or GNU/Octave languages that are frequently used in the academic environment. As a solution to this problem, the Perl script was proposed that translates comments written in MATLAB M-files into C/C++ type that can be processed by the Doxygen. Such a solution has a distinct advantage: it is simple to install and use for the documentation of scientific software. The script and its installation is described in more details in the subsection 2.4. The alternative solutions are briefly discussed in Section 5.

\* Doxygen, at least to the time of writing the article (October 2011), does not support the citations using standard cite command. However, the patch was proposed for the solution of this issue.
Installation of Doxygen in Debian GNU/Linux is performed by:

```bash
$ aptitude install doxygen doxygen-gui
```

Source code and binary distributions for Mac OS and Windows are available as well.³

### 2.3 Version Control System

Using a version control system (VCS) is obvious to professional software developers; however, for the researchers it is generally less common. It has been noted that the biggest obstacle for adoption of VCS in academic environment is the need to study large amount of new commands for a VCS.¹² The development process of scientific software is usually less complicated than the conventional software. Therefore, only a few basic commands¹ are really needed for simple yet effective control of the development process:

- initializing the repository (`hg init`);
- adding new files to the repository (`hg add`);
- removing files from the repository (`hg rm`);
- checking the state of the repository (`hg st`);
- merging the files (`hg merge`);
- reverting the changes of files (`hg revert`);
- committing to the repository (`hg ci -m "commit message"`);
- pulling the changes from another repository / web web-based hosting service (`hg pull https://user@bitbucket.org/project/currentproj`);
- pushing the changes to another repository / web web-based hosting service (`hg push /home/user/backups/currentproj`);

Those commands are usually enough even for a rather complex project.

There are many web-based services for software projects that use version control systems; such web services can be used for synchronisation of repository by other researchers. As an example, [www.bitbucket.org](http://www.bitbucket.org) can be mentioned for the Mercurial DVCS.

### 2.4 Script for on-the-fly comments translation

Perl is a general-purpose scripting language that is widely used for the automation of routine tasks. In Linux, it is used for many system scripts and is usually installed by default. In this framework, Perl is used for execution of the conversion script, which translates on-the-fly the comments in MATLAB scripts and functions into C/C++ comments for the Doxygen. The installation of Perl in Linux is therefore not required. In other systems like Windows and Mac OS, basic Perl interpreter is usually provided with MATLAB itself (e.g., for MATLAB R2010b, it is in `C:\Program Files\MATLAB\R2010b\sys\perl\win32\bin\perl.exe`).

The Perl script that translates the comments written in MATLAB M-fils into C/C++ type that can be processed by the Doxygen can be downloaded from MatlabCentral website.¹¹ The installation consists of:

1. copy the file `Doxyfile` to the directory of the MATLAB project files (e.g., project files are in `/home/user/-matlab/currentproject`);

2. copy the script file `m2cpp.pl` to the same directory of the MATLAB project files as for `Doxyfile`;

¹Those commands are given for Mercurial DVCS; however, they are similar for other version control systems.
3. edit the file **Doxyfile** using any text editor to configure the following settings:

```plaintext
PROJECT_NAME = substitute project name
FILTER_PATTERNS = *.m=./m2cpp.pl
OUTPUT_DIRECTORY = ./Doc/
INPUT = .
PERL_PATH = /usr/bin/perl
STRIP_FROM_PATH = .
```

Dots at the end of the paths mean that the files are located at the current directory and in subdirectories. The documentation will be located at **OUTPUT_DIRECTORY** that is ./Doc/ in this example.

One should be aware that the script is written under Windows, which means that users of other operation systems have to correct the first string of the script *m2cpp.pl* for correct path to Perl interpreter. In case of Linux/UNIX it will be:

```plaintext
#! /usr/bin/perl
```

instead of

```plaintext
#! /usr/bin/perl.exe
```

by default. The script in UNIX must have proper privileges for the execution, which can be changed by the command:

```plaintext
chmod 755 m2cpp.pl
```

A copy of the script can be put into the directory with the software code or placed in one of the system’s directories.

### 3. CONFIGURATION OF THE DOCUMENTATION FRAMEWORK

Although some components of the framework may be already familiar to the reader, there are some considerations to make in case of the research software. For instance, if one uses specific extension packages for **\LaTeX** documentation, they must be included in *Doxyfile* configuration file for the Doxygen to process the mathematical formulas. This section gives the necessary guidelines for the effective usage of the described documentation framework.

#### 3.1 Using **\LaTeX** commands for Doxygen

The mathematical formulas in the documentation of MATLAB M-files must be placed in `\f$ ... \f$` environment. In case of complex mathematical formulas, one need to mention the required **\LaTeX** extension packages in *Doxyfile*:

```plaintext
#-------------------------------------------------
# Configuration options related to the LaTeX output
#-------------------------------------------------
GENERATE_LATEX = YES
LATEX_OUTPUT = latex
LATEX_CMD_NAME = latex
MAKEINDEX_CMD_NAME = makeindex
PAPER_TYPE = a4
```
In this case, we added *amssymb*, *amsfonts*, *amsmath*, *mathtext* packages. This allows using more complicated mathematical expressions in the documentation of MATLAB M-files.

### 3.2 Inserting an image in the documentation

Often it is necessary to insert a picture or a scheme in the documentation to clarify the internals of the implemented algorithm. This can be done using the code:

```plaintext
%> @image html FraunhoferDiffraction.png
%> @image latex FraunhoferDiffraction.eps "Caption" width=10cm
```

The location of images must be defined in the Doxyfile using the string:

```plaintext
IMAGE_PATH = ./Reference/images/
```

The supported images formats are JPEG, PNG, GIF and EPS (for \LaTeX documentation).

### 3.3 Using Version Control System for scientific software

When a version control system (VCS) is used along with Doxygen, it can be difficult to check which files were changed since there will be a long list of temporary files. To avoid this, most of VCS allow making a list of ignored files. The documentation can be put in such a list to track only the software code files (the documentation is regenerated from the code itself).

In Mercurial DVCS, such list of ignored files is called `.hgignore` and placed in directory of a software project. The content of `.hgignore` can be the following:

```plaintext
aux
log
toc
documentation/html/
documentation/latex/
```

First three lines contain the mask to ignore \LaTeX temporary files, and the last two lines forces Mercurial to ignore the documentation generated from the source code. For other VCS, such a file with ignore list is called similarly (e.g., `.gitignore` for git DVCS).

### 4. Examples of documentation of scripts and functions using the described framework

Examples of MATLAB functions and scripts documentation are presented below to show the documentation framework in actual use. First, documentation of a MATLAB script from the adaptive optics simulator will be shown. Second, an example of MATLAB function documentation will be shown. The examples are aimed to show the structured approach in the documentation of scientific software with the described framework‡.

‡The classes can be documented in the similar manner; the examples are not added for brevity.
In order to generate the documentation, go to the root directory of the software project and give a command:

\$ doxygen Doxyfile

The process of documentation generation can take a long time (several minutes), especially if the code contains many mathematical formulas. The documentation is generated for all files at once; the output will be in HTML and LaTeX formats. To view the generated documentation, open a browser and go to documentation subdirectory (defined in OUTPUT_DIRECTORY string of Doxyfile configuration file that is ./Doc/ in this example), which is specified in the Doxyfile, and open index.html file.

4.1 Documentation for a MATLAB Script

The example of the documentation for a MATLAB script is the following:

```matlab
1 %> @file test_script_forshow.m
2 %>@brief This is a MATLAB script to show the main capabilities of the ...
3 documentation using Doxygen, LaTeX, and an on-the-fly Perl script.
4 %> @author Mikhail V. Konnik
5 %> @todo There are no current features to be implemented.
6 %> @bug No known bugs.
7 %> @date 22 August 2011
8 %>
9 %> @section script_forshow Documentation of a MATLAB script.
10 %> This is a test script to show general ideas for the documentation of the MATLAB ...
11 scripts. First of all, the mathematical formulas can be used directly from ...
12 yours LaTeX notes. For instance, consider Zernike functions, which can be even ...
13 \( \hat{f}^{m,n}(\rho,\theta) \) and odd \( \hat{f}^{-m,n}(\rho,\theta) \). Some text can ...
14 be highlighted as @b bold, other text like @c typewriter.
15 %> @attention The highlighted text may look different in different browsers; it ...
16 depends on the fonts configuration.
17 %>
18 %> If one needs to put a line brake, use @n command to do so. The documentation may ...
19 be divided into subsections or may refer to @ref function_forshow other function.
20 %>
21 %> @subsection script_forshowsubsect Notes on scripts documentation
22 %> The @verbatim \cite endverbatim command is not yet supported directly by ...
23 Doxygen; however, there is a patch that solves this problem. The patch will be ...
24 included in the future versions of Doxygen.
25 %> @note It is noteworthy that it is possible to document classes in MATLAB in the ...
26 same way.
27 clc
28 clear all
29 close all
30 addpath('tools','visualisators');
31 % ..... 
```

The listing of the code presented above contains typical structure of documentation of the scientific software written on MATLAB. The generated documentation for this function is presented on Fig. 1.
Figure 1. The documentation for the MATLAB script generated from the source code by Doxygen and on-the-fly Perl script (the HTML version, opened in a browser).

The mark-up of Doxygen is mostly self-explanatory; however, there are some aspects to be noted:
1. the actual file name of a routine and the file name defined after @file must coincide (otherwise the HTML documentation will contain empty page for this file);

2. the @bug and @todo commands may contain lists that start from %> - command, for example:

```
%> @todo The following must be done for this routine:
%> - random vectors must contain data in rows, not columns
%> - correct the other programs that use incorrect references of row vectors ... (instead of columns)
```

3. \TeX formulas must be in $ ...$ environment;

4. second word after @section (in this case is scriptforshow) is a label, which can be used for the reference from any other function/script;

5. if there is a mistake in the Doxygen's mark-up, the documentation must be regenerated for the entire project.

It is noteworthy that the commenting of a script or a function must not be interrupted: the sign %> should be placed at the beginning of each line even for an empty line. Otherwise Doxygen will ignore any lines with documentation after the first empty line, and these problems are often not easy to track down.

4.2 Documentation for a MATLAB Function

The documentation of a function is almost the same as for a script, except the description of the input parameters and return values. The example of the documented MATLAB function is the following:

```
%> @file tool_function_forshow.m
%> @brief This is an example of a MATLAB function to show the main capabilities of ... the documentation using Doxygen, LaTeX, and an on-the-fly Perl script.
%> @author Mikhail V. Konnik
%> @date 22 August 2011
%> @section functionforshow Documentation of a MATLAB function.
%> This test function shows how to document MATLAB functions in the described ... framework. The main points are:
%> - the names of input parameters (param) and return values (retval) must coincide ... with the names used in the function itself;
%> - it is a good practice to write the expected type/size of variables;
%> - the function(s) can contain brief descriptions as well.
%> All the rest mark-up is valid for functions, scripts, and classes.
%>======================================================================
%> @brief Test function to show the documentation of MATLAB functions.
%> @param grad_row = gradients matrix for ROWS [N-by-N]
%> @param grad_col = gradients matrix for COLUMNS [N-by-N]
%> @retval a = coefficients of reconstructed Zernike functions.
%> function a = tool_function_forshow(grad_row, grad_col);
%> ....
```

The conventions for Doxygen mark-up are the same as for the scripts. It is worth noting that the names of parameters @param and @retval must coincide with the names used in the function; errors during the processing of documentation will appear otherwise. The compiled documentation can be viewed in a browser (see Fig. 2).
Figure 2. MATLAB function documentation that is generated from the source code by Doxygen and on-the-fly Perl script (the HTML version, opened in a browser).
5. DISCUSSION OF THE FRAMEWORK COMPONENTS

The documentation framework discussed in this paper can be adjusted to meet the requirements of the specific research project. For instance, the choice of version control system (VCS) is based on the complexity of the project. The research software is often developed as a series of interconnected toolboxes that simulate physical processes. For instance, in our case of the adaptive optics simulator, the software consists of optics simulation part, sensor part, and control code part. These parts are being developed by separate teams of researchers. Therefore we came to the conclusion that a decentralised version control system (DVCS, like mercurial and git) can be more beneficial for the research software than traditional centralised VCS (like SVN).

The described Perl script,\textsuperscript{11} which translates on-the-fly the comments in MATLAB scripts and functions into C/C++ comments for the Doxygen, can be replaced as well. For instance, the filter written by Ben Heasly\textsuperscript{13} converts the comments from MATLAB scripts into the data for Doxygen. The alternative solution is provided by Ian Martins and is available for the download.\textsuperscript{14} The script on Perl\textsuperscript{11} used in this paper, however, is easier to use and can be included in the documentation framework with minimum efforts.

5.1 Working example of the framework

The examples of the code documentation, which are described in this paper, are freely available for downloading from the authors' development webpage:
http://code.google.com/p/latexdoxygencludemanagementexample/

Specifically, in the archive on the website one can find the example of documentation of a MATLAB's script test_script_forshow.m (see Subsection 4.1) and the function documentation example tool_function_forshow.m (see Subsection 4.2) are in the simulator folder. The compiled documentation is in help folder.

6. CONCLUSION

The problem addressed in this tutorial paper is a lack of consistent and convenient documentation frameworks for the scientific software. The research software code usually contains the implementation of complicated mathematical models, which must be described appropriately in the documentation. Therefore, the documentation framework must be able to handle mathematical symbols, graphical schemes and references.

The described open-source cross-platform framework attempts to provide a comprehensive solution for the documentation of the scientific software. The framework consists of \LaTeX \textsuperscript{2} scientific typesetting language, Doxygen software documentation generator, one of distributed version control system such as Mercurial, and the Perl script for on-the-fly translation of comments in MATLAB M-files into the documentation generated by Doxygen. The necessary guidelines for the installation and configuration of the framework components are given. The alternative components of the documentation framework are discussed. Examples of scripts and functions documentation should give a clear idea of the features of the described framework.

We hope that this tutorial paper will be helpful for better documentation of the scientific code, which is otherwise (and way too often) being rewritten from scratch.
REFERENCES


