Tips and Tricks

The present document covers a number an items which you should find useful during your PhD career here. All the examples in this document are guaranteed to work on a Unix platform (MacOS, Linux), and are also adaptable to Windows systems.

1 Git and Github

1.1 Brief description

Git is "a version control system that is used for system development". Not only can it keep track of all the changes imparted to code undergoing development by a single user, but its functionalities are extremely handy when several developers interact with the same piece of code concurrently.
Github is a website that offers a server architecture that you can synchronize with the local copy of the code. This remote code storage location is called " remote repository" in Git jargon. Repositories can be

- public: everybody can see the code, contribute or pull (i.e retrieve the most recent version of the code) from the repository
- private: the repository's owner is by default the only one able to perform any of the previously listed actions, unless authorized contributors are added to the repository access list.

Github permits one to create an infinite number of public repositories for free. Private repositories are typically not free, but you can upgrade your Github account to a Premium account (which allows you to create private repositories) thanks to your CU Boulder student status.
Note that alternatives to Github such as Bitbucket do exist. A summary of Git workflow is provided on Figure 1 along with the main Git commands.

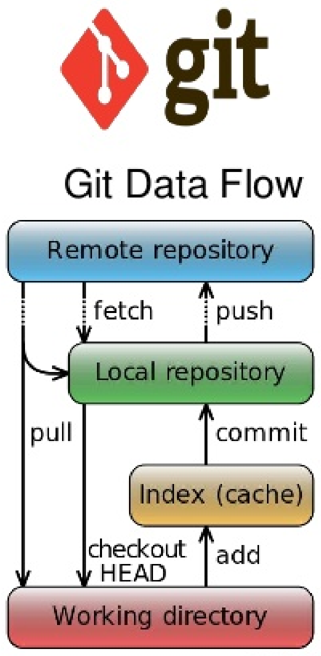


Figure 1: A summary of Git workflow taken from here

Figure 1 encompasses a good chunk of what you need to know about Git. The following sections will show you two important examples:

1. How to create a local repository on your computer, a remote repository on Github and synchronize the two. In particular, the *init*, *add*, *commit* and *push* commands will be exemplified.

2. How to contribute to code in a collaborative environment.

1.2 How to set up a local/remote repository

Create the local repository

We begin by opening a terminal window into the folder where we want to create our repository. In this example, the repository will be in a folder named *NewStudentHandout* located in the *Documents* folder of a Unix system. On a Mac, the terminal command *pwd* would return */Users/my_username/Documents/NewStudentHandout* when executed from *NewStudentHandout*. Once we have made sure that the window terminal is in the right folder, let's type

Listing 1: Creation of the local repository

```
git init
```

If everything goes well, the following should appear on the terminal window:

Listing 2: Successful git init message

```
Initialized empty Git repository in /Users/bbercovici/Documents/NewStudentHandout/.git/
```

This command initiates the local git repository by creating the hidden directory *./git* and populates it with a number of configuration files. These files are the backbone of your local repository, so make sure not to alter them.

As noted by Git, our local repository is empty. Note that the working directory content is unrelated to the local repository content as those are two separate entities as shown on Figure 1. We thus have to use *add* to add files to the index cache, and *commit* this index to have those files added to our local repository.
Let us start with the *add* command. It can be used in two different ways, as shown below (assuming that *myfile.txt* does exist in the working directory):

Listing 3: Add files to the cache

```
git add myfile.txt
# the index cache only contains myfile.txt
git add *
# the index cache now contains all files in the working directory that were not excluded in .gitignore
```

Another useful command is *status*, which highlights files depending on whether they have been modified since the last commit. Since *git status* lists myfile.txt in green, we are now ready to commit it to the local repository using the following command:

Listing 4: commit files to the local repository

```
git commit -m "first commit: myfile.txt added to the local repo"
```

The ”-m” options allows us to append a message to the commit. After the content of the local repository is pushed to remote repository, the other developers will be able to read the message and quickly understand what this commit was about.

Create an account on Github

The title of this section is quite self-explanatory. Make sure you remember your username/password as you will need them shortly!

Create the remote repository on Github

Log on to Github and click on ”New repository” in the drop-down list next to your profile picture as show on Figure 2.

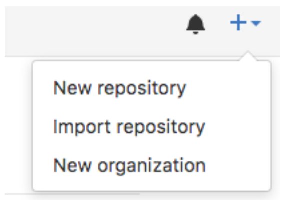
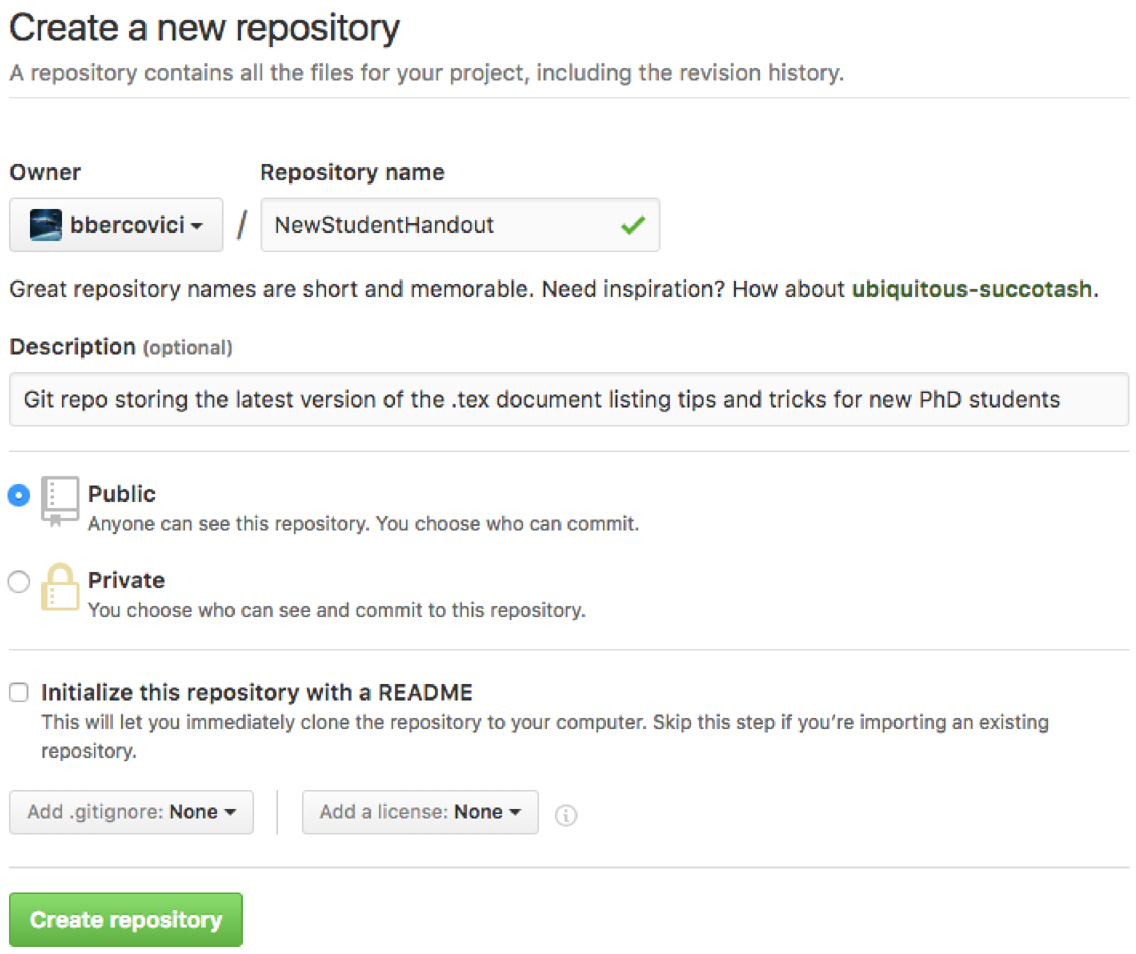


Figure 2: New repository

The following window (shown on Figure 3) allows you to choose several important settings for your remote repository.


Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: bbercovici / Repository name: NewStudentHandout

Great repository names are short and memorable. Need inspiration? How about **ubiquitous-succotash**.

Description (optional): Git repo storing the latest version of the .tex document listing tips and tricks for new PhD students

[✓ Public] Anyone can see this repository. You choose who can commit.

Private — You choose who can see and commit to this repository.

☐ Initialize this repository with a README — This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

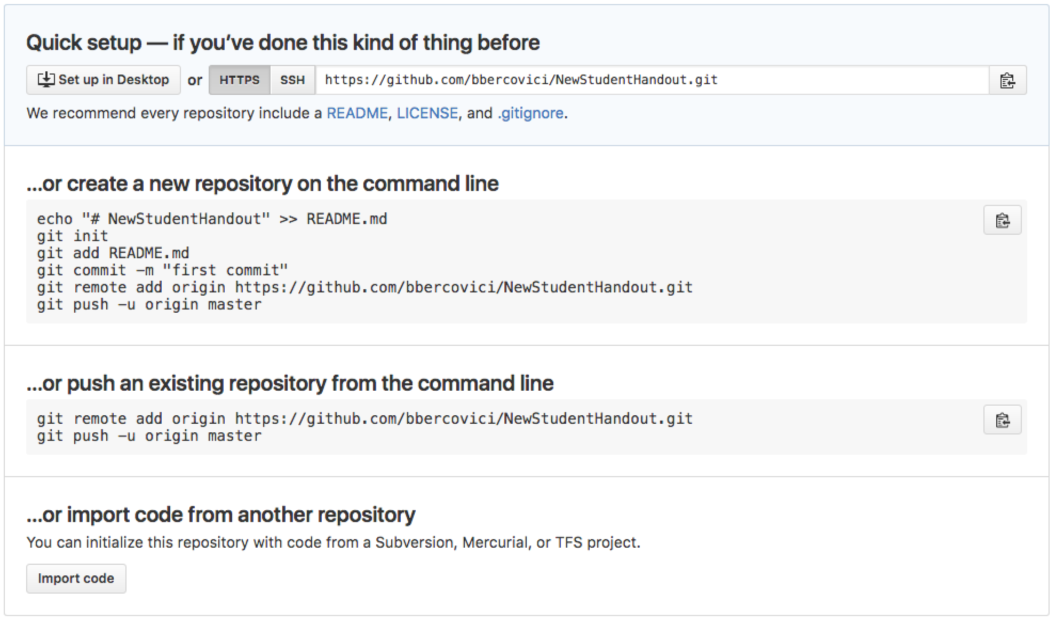
Figure 3: New repository settings

- Choose whether the repository will be made public or private (the latter being only available if you have upgraded to a premium account).
- Initialize the repository with a README.md file.
  Leave this box unchecked if your local repository is already created, as this file can be added later on manually without risking conflicts
- Add a .gitignore file. This file lists all the file extensions that must be ignored by Git. Especially handy if there are specific file extensions that you never want to see included in a commit.
- Add a license file.

When you are ready, simply click on Create Repository and proceed to the next step.

Connect the local repository to the remote repository

You should now be seeing the same page as on Figure 4.


Quick setup — if you've done this kind of thing before

https://github.com/bbercovici/NewStudentHandout.git

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# NewStudentHandout" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/bbercovici/NewStudentHandout.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/bbercovici/NewStudentHandout.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Figure 4: Final setup options

The third option is what we want to do, since we have already created a remote repository. We can copy and paste the two lines under ”...push an existing repository from the command line” into our terminal. Start with the following:

Listing 5: Add the remote repository location

```
git remote add origin https://github.com/bbercovici/NewStudentHandout.git
```

This will add the remote repository address to one of the hidden configuration files. Finally, push the local repository to the remote.

Listing 6: Push the local repository to the remote

```
git push -u origin master
```

We are now all set! We can keep working in our local directory, add files containing new content to the index using *git add*, commit the changes to the local repository using *git commit*, and finally pushing the changes to the remote using *git push origin master* (note that the -u option was not used here). Now is also the time to create a .gitignore file and a README.md file.

1.3 Contribute to code in a collaborative environment

2 LaTeX

This section walks you through the most critical, time-consuming, less-rewarding steps of the download & installation of the LaTeX distribution, to the point where you can compile and edit your own documents.

There are two things that you need in order to do so:

1. The LaTeX distribution: the (huge) software package that will compile your raw .tex document and turn them into neat pdf documents.
2. The TeX editor. This is the software that you will use to create and edit .tex documents. It also provides an interface with the compiler.

The procedure is slightly different depending on whether you are using Windows or MacOS.

Retrieving the LaTeX Distribution

Windows

Two versions of the LaTeX distribution installer are available on the MikTeX website

- If your computer is less than two years old, you can download the MiKTeX 2.9.5870 Net Installer 64-bits executable
- If you are not sure whether you can run the 64-bits executable, download the MiKTeX 2.9.5870 Net 32-bits executable

1. Run the installer once the file is downloaded.
2. When asked, click Download MiKTeX to download the distribution.
3. When asked whether the executable should download the Basic or Complete MiKTeX distribution, choose Complete
4. Choose the folder where the installation files will be downloaded
5. Pick a download source geographically close from CU in the server list that should pop up
6. Start the download...

...which might take a while...

7. Once the download is over, go to the folder where you saved the LaTeX distribution files. Look for setup-2.9.3959.exe (the only executable file in this folder) and run it.
8. Once the installation process is over, you have a complete LaTeX distribution installed on your PC and are almost good to go!

MacOS

The MacTex distribution can be found there. Click on the link at the center of the page to get the whole distribution. The file weighs 2.5 GB so the download might take a while .

Getting the Tex Editor

From my own experience, I strongly recommend Texmaker. The interface is neat and the autocomplete functionality (that predicts the rest of the command you are typing) rocks.
This software works on Windows, MacOS and Linux. The version you need can be found on the download page. Installing Texmaker does not require any special instructions so you should be fine!

Editing and compiling this tutorial

First of all, download the following archive and extract it into a folder of your choice.

The *preamble.tex* file lists all the packages that are needed in order to compile some commands that would otherwise be unknown to the Latex compiler. I have been using Latex and constantly updating this list of packages for nearly 4 years, so it should be comprehensive enough now[1].

[1] The *float* package that you will see listed in the preamble is one of these packages that come in handy. Without it, including figures in your document can lead to layout disasters...

Once this is done, open Texmaker. Figure 5 shows what you should see as you have no tex document currently opened.

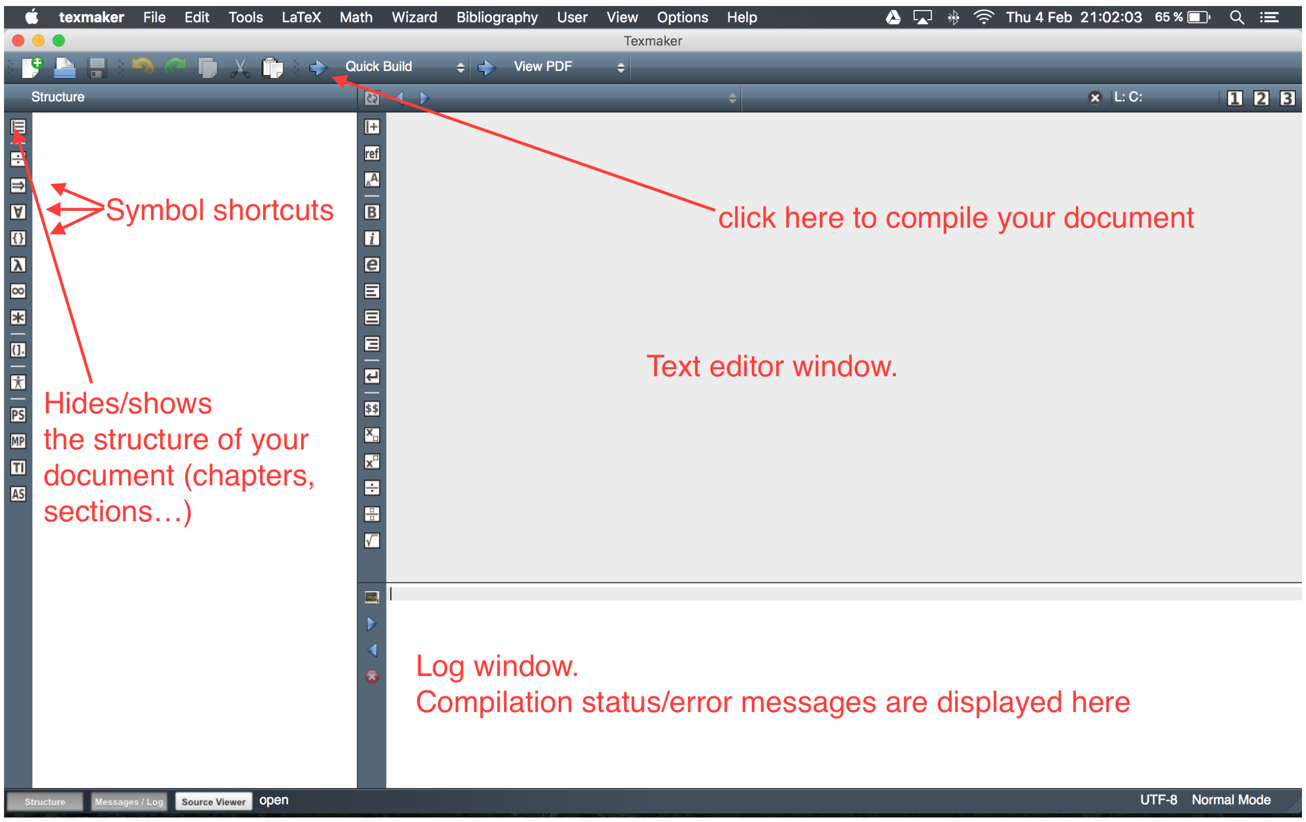
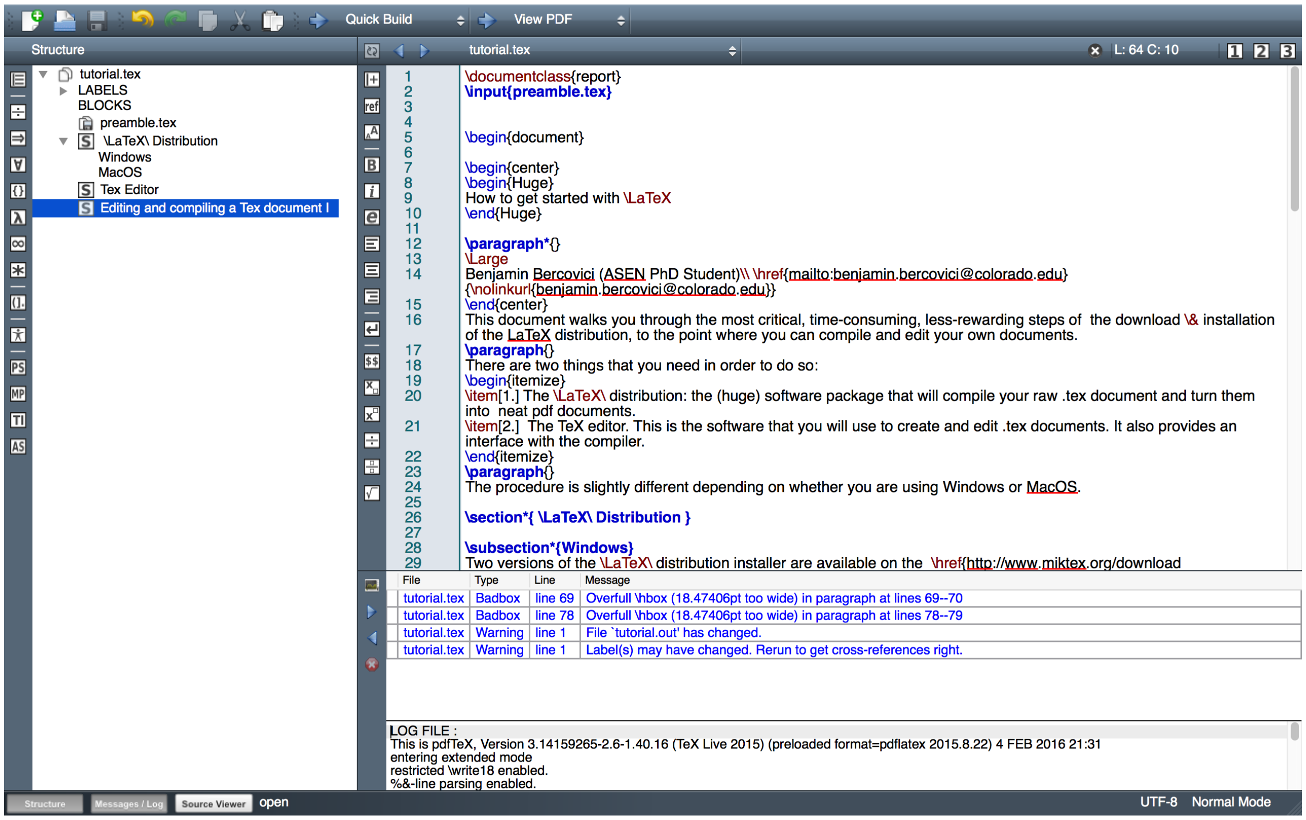

texmaker File Edit Tools LaTeX Math Wizard Bibliography User View Options Help
Thu 4 Feb 21:02:03 65 %
Texmaker
Quick Build
View PDF
Structure
L: C:
Symbol shortcuts
click here to compile your document
Text editor window.
Hides/shows the structure of your document (chapters, sections…)
Log window.
Compilation status/error messages are displayed here
Structure
Messages / Log
Source Viewer
open
UTF-8
Normal Mode

Figure 5: Main menu

Opening the tutorial.tex file should get you the same view as on Figure 6:



```
\documentclass{report}
\input{preamble.tex}


\begin{document}

\begin{center}
\begin{Huge}
How to get started with \LaTeX
\end{Huge}

\paragraph*{}
\Large
Benjamin Bercovici (ASEN PhD Student)\\ \href{mailto:benjamin.bercovici@colorado.edu}
{\nolinkurl{benjamin.bercovici@colorado.edu}}
\end{center}
This document walks you through the most critical, time-consuming, less-rewarding steps of  the download \& installation
of the LaTeX distribution, to the point where you can compile and edit your own documents.
\paragraph{}
There are two things that you need in order to do so:
\begin{itemize}
\item[1.] The \LaTeX\ distribution: the (huge) software package that will compile your raw .tex document and turn them
into  neat pdf documents.
\item[2.]  The TeX editor. This is the software that you will use to create and edit .tex documents. It also provides an
interface with the compiler.
\end{itemize}
\paragraph{}
The procedure is slightly different depending on whether you are using Windows or MacOS.

\section*{ \LaTeX\ Distribution }

\subsection*{Windows}
Two versions of the \LaTeX\ distribution installer are available on the  \href{http://www.miktex.org/download
```

File	Type	Line	Message
tutorial.tex	Badbox	line 69	Overfull \hbox (18.47406pt too wide) in paragraph at lines 69--70
tutorial.tex	Badbox	line 78	Overfull \hbox (18.47406pt too wide) in paragraph at lines 78--79
tutorial.tex	Warning	line 1	File `tutorial.out' has changed.
tutorial.tex	Warning	line 1	Label(s) may have changed. Rerun to get cross-references right.

```
LOG FILE :
This is pdfTeX, Version 3.14159265-2.6-1.40.16 (TeX Live 2015) (preloaded format=pdflatex 2015.8.22) 4 FEB 2016 21:31
entering extended mode
restricted \write18 enabled.
%&-line parsing enabled.
```

Figure 6: Texmaker displaying the *tutorial.tex* file. Note that the document structure automatically appears on the left side of the window

If you take a look at the very top of the document, you will recognize the *preamble.tex* file that I mentioned before. The \input{*preamble.tex*} command will - as the name suggests - have *tutorial.tex* including the content of *preamble.tex*. It is equivalent to copying the content of *preamble.tex* and pasting it into *tutorial.tex*. Because the latter would be rather messy, listing all the packages you will use in an external file and using the input command is a much better practice.

Note that everything that begins with a \\[2] is a LaTeX command. This \\ tells the compiler that whatever follows this symbol has to be interpreted. Plain text is just considered as is.
Clicking on the *Quick Build* button will run the compiler, that will process the .tex file and generate a *tutorial.pdf* document in the same folder as the *tutorial.tex* file

Maths and LaTeX

There are thousands of tutorials available on the web addressing this topic. The greatest understatement of the day would be to say that LaTeX is really well suited to maths typesetting. Several examples based off former homeworks of mine are available there.

I strongly encourage you to open the document, compile it, see the pdf output, and then take a closer look at the .tex file to understand how mathematical expressions are formatted.

[2]The key that corresponds to this symbol is the one right above the Enter/Return Key.

Resources

Dr. Schaub has listed a number of references on his website which I encourage you to use. There is in particular a Latex cheat sheet that could come in handy.

3 Asymptote

3.1 Brief description

Asymptote is "a powerful descriptive vector graphics language that provides a natural coordinate-based framework for technical drawing." In other words, it is a compiled language that generates .eps or .pdf representing any 2D or 3D scene parametrized mathematically. Its hybrid syntax borrows from Python, C and Latex, which can be a bit bewildering at first. Raw asymptote code is customarily stored in the form of a .asy file, which can be edited using the text editor of your choice. The installation procedure depends on the platform you are using:

- MacOS X: Asymptote is already shipped with MacTex, so there is nothing to do besides retrieving the Latex distribution
- Windows: Download the latest executable file from the official Asymptote page. Make sure to have a running Latex distribution before attempting to run Asymptote!
- Linux: Similarly to MacOS X, Asymptote is shipped with the TexLive distribution.

Asymptote files can be compiled directly from a Unix terminal. Simply type *asy myfile.asy* to compile the file and generate the output which by default will be named *myfile.pdf* or *myfile.eps*. The output format can be set using the *settings.outformat="pdf";* instruction in the asy file. On Windows, compilation can be taken care of by TexMaker.

3.2 Examples

Two Asymptote output files are presented below, to demonstrate the 2D and 3D capabilities of the language. Figure 7 represents a planetary flyby where the input angles are prescribed to their real value. The trajectory that is displayed is a conic (an hyperbola in this case) which corresponds to the true Keplerian orbit expected in this case.

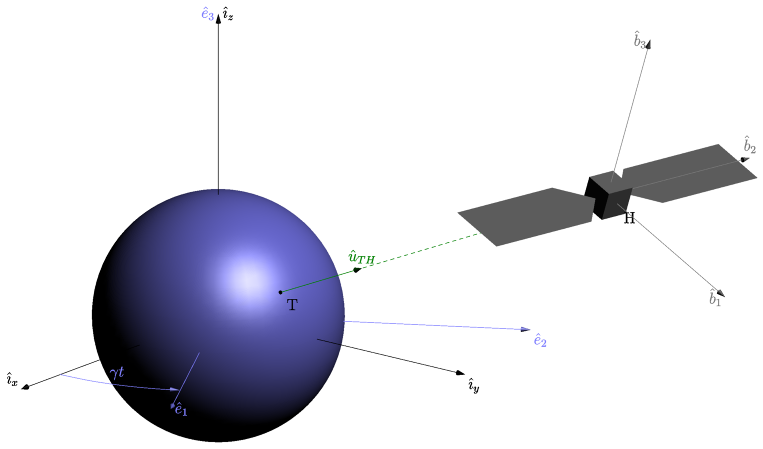


Figure 8: Satellite body-frame to inertial frame (3D scene)

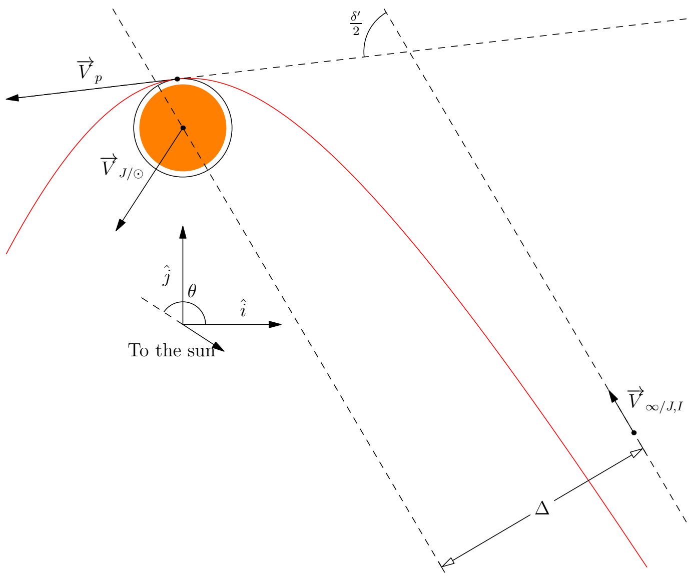


Figure 7: Planetary flyby (2D scene)

The example shown on Figure 8 is another illustration of Asymptote capabilities. Here, the body frame of a spacecraft is represented relative to an inertial frame. The spacecraft attitude is parametrized in terms of a set of 321 Euler angles (yaw, pitch, roll) that the user can edit. Recompiling the .asy file for a new attitude will generate the same scene with a different orientation, without requiring the user to edit manually the location of every single satellite feature.

3.3 More examples

The best compilation of examples I have found to date is probably this one for 2D examples and this one for 3D examples. As the reader will soon notice, these resources are unfortunately not in English. This being said, they are quite self explanatory due to the ubiquitous use of English in the Asymptote language in addition to the (almost) comprehensive list of Asymptote routines they provide. English documentation files, quite comprehensive but maybe less exhaustive can also be found here and another one here.