Project Preparation

The first steps to setup a scalable software project are none trival and need a good strcuture for later project expands. Either to setup further new projects a well known scalable project structure helps to combine different software parts to bigger environment packages. Therefore a project preparation flow needs to be documented. It unifies the outcome of software projects and part guarantee certain quality aspects.

The following steps can be used as guidance to establish a propper Matlab project structure in general. Each step is documented with screenshots to give a comprehensible explanation.

Contents

See Also

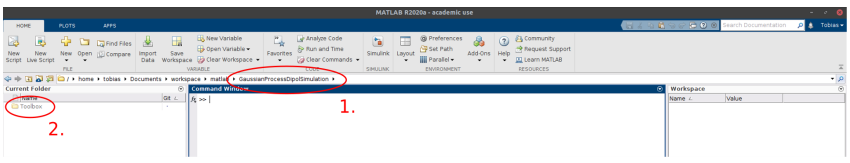
- Create a New Project From a Folder
- Add a Project to Source Control
- Setup Git Source Control
- Use Source Control with Projects
- Git Attributes
- Git Ignores
- Add Files to the Project
- Commit Modified Files to Source Control
- Clone Git Repository

Create Main Project Directory

The main project directory contains only two subfolders. The first one is the Toolbox folder where the project, m-files and other project files like documentation are placed. The folder is also called sandbox folder in Matlab project creation flows which is just another description for a project folder where the coding takes place. The second folder is a hidden Git repository folder which keeps the versionation in final. It is respectively seen a remote repository that establish basics to setup backup plans via Git clone or can be laterly replaced by remote repository on a server or a GitHub repository to work in common on the project.

First step:

1. Create an empty project folder, open Matlab navigate to folder path.
2. Right click in the Current Folder pane and create New> Folder "Toolbox".
3. Open a Git terminal and in the project directory and initialize an empty Git repository.



```
tobias@bean42:~/Documents/workspace/matlab$ cd GaussianProcessDipolSimulation/
tobias@bean42:~/Documents/workspace/matlab/GaussianProcessDipolSimulation$ ls
Toolbox
tobias@bean42:~/Documents/workspace/matlab/GaussianProcessDipolSimulation$ git init
Initialized empty Git repository in /home/tobias/Documents/workspace/matlab/GaussianProcessDipolSimulation/.git/
tobias@bean42:~/Documents/workspace/matlab/GaussianProcessDipolSimulation$ ls -a
.  ..  .git  Toolbox
tobias@bean42:~/Documents/workspace/matlab/GaussianProcessDipolSimulation$ git config -l
user.name=Tobias Wulf
user.email=46107549+TobiasWulf@users.noreply.github.com
core.editor=gedit
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
tobias@bean42:~/Documents/workspace/matlab/GaussianProcessDipolSimulation$
```

3.

Create Matlab Project with Git Support

In second it is needed to create the Matlab project files in a certain way to get full Git support and support for the Matlab help browser environment. In this use case the before created local Git repository is used as remote origin. So several settings are automatacally made during the creation process by Matlab and as mentioned before the "local remote" repository can be replaced later by a remote origin located on a server or GitHub. The Toolbox folder must be empty to process the following steps.

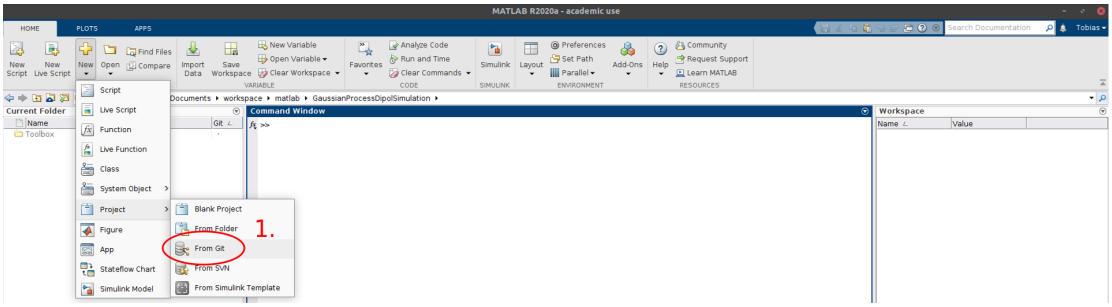
It is recommend to do no further Git actions on the created Git repository via Git terminal!

These steps only proceed the project setup further Matlab framework functionality is added later.

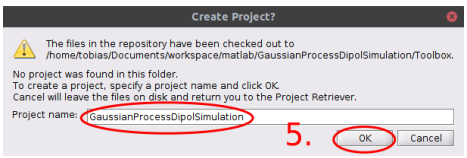
Second step:

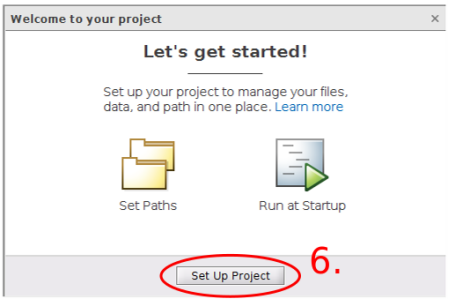
1. In the created main project directory create a New > Project > From Git.
2. Change the repository path to the hidden Git repository path in the main project directory.
3. Change the sandbox path to the Toolbox path in the main project directory.
4. Click Retrieve.
5. Enter the project name given by the main project directory name and click OK.
6. Click on Set Up Project and skip the two follwing steps via Next and Finish.
7. Switch to Toolbox directory by double click on the folder in the Current Folder pane, open the created Matlab project file with a double click and check source control information under PROJECT tab by clicking Git Details.
8. Add a short project summary by click on Details under the ENVIRONMENT section of the PROJECT tab.
9. Click Apply.
10. Click OK.

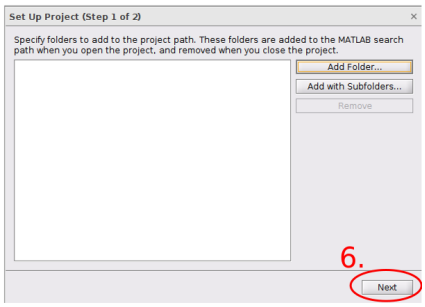
The project itself is under source control now.

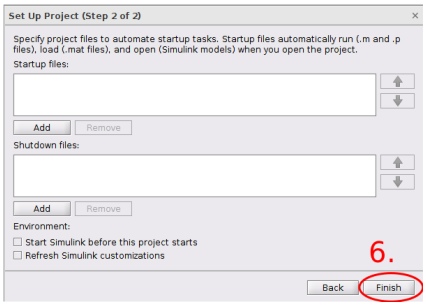


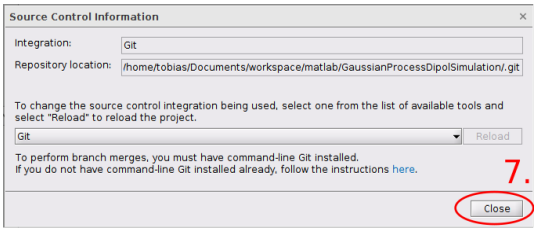

New Project From Source Control
Source control tool:
Git
Repository path:
Change...
New
Sandbox:
/home/tobias/Documents/workspace/matlab/GaussianProcessDipolSimulation/Toolbox
Source control information:
To perform branch merges, you must have command-line Git installed.
If you do not have command-line Git installed already, follow the instructions here.
2.
3.
4.
Retrieve
Cancel

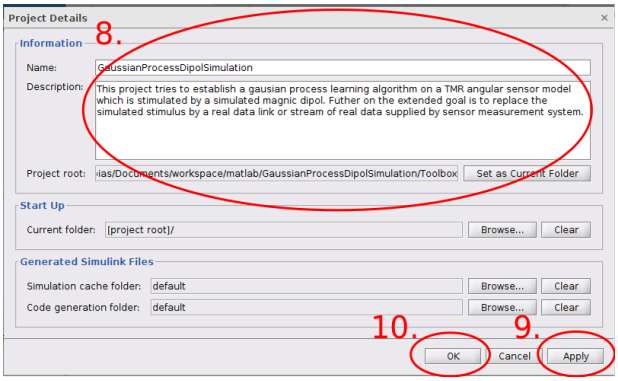

Create Project?
The files in the repository have been checked out to
/home/tobias/Documents/workspace/matlab/GaussianProcessDipolSimulation/Toolbox.
No project was found in this folder.
To create a project, specify a project name and click OK.
Cancel will leave the files on disk and return you to the Project Retriever.
Project name:
GaussianProcessDipolSimulation
5.
OK
Cancel


Welcome to your project
Let's get started!
Set up your project to manage your files,
data, and path in one place. Learn more
Set Paths
Run at Startup
Set Up Project
6.


Set Up Project (Step 1 of 2)
Specify folders to add to the project path. These folders are added to the MATLAB search
path when you open the project, and removed when you close the project.
Add Folder...
Add with Subfolders...
Remove
6.
Next


Set Up Project (Step 2 of 2)
Specify project files to automate startup tasks. Startup files automatically run (.m and .p
files), load (.mat files), and open (Simulink models) when you open the project.
Startup files:
Add
Remove
Shutdown files:
Add
Remove
Environment:
Start Simulink before this project starts
Refresh Simulink customizations
6.
Back
Finish





Registrate Binaries to Git and Prepare Git Ignore Cases

The root of Git is to work as text file versioner. Source code files are just text files. So git versionates, tags and merges them in various ways in work flow process. That means git edits files. This point can be critical if git does edit a binary file and corrupts it, so that is not executable any more. Therefore binary files must be registrated to Git. Another good reason is to registrated binary or other none text files beccause Git performs no automatic merges on file if they are not known text files. To keep the versionating Git makes a taged copy of that file every time the file changed. That can be a very junk of memory and lets repository expands to wide.

To prevent Git for misshandling binaries it is able to regestrate them in a certain file and mark the file types how to handle them in progress. The file is called .gitattributes must be placed in the Git assinged working directory which is the sandbox folder for Matlab projects. The .gitattributes file itself is hidden.

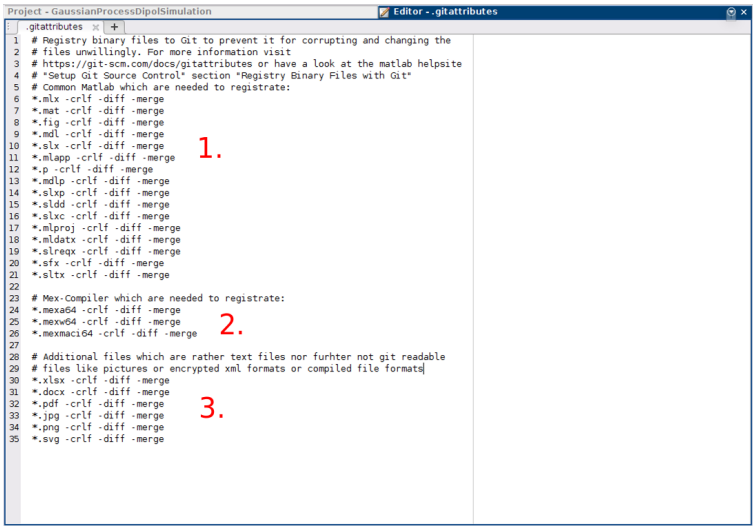
Three options are needed to mark a file type as binary. The -crlf option disables end of line conversion and the -diff option in combination with the -merge option to mark the file as binary.

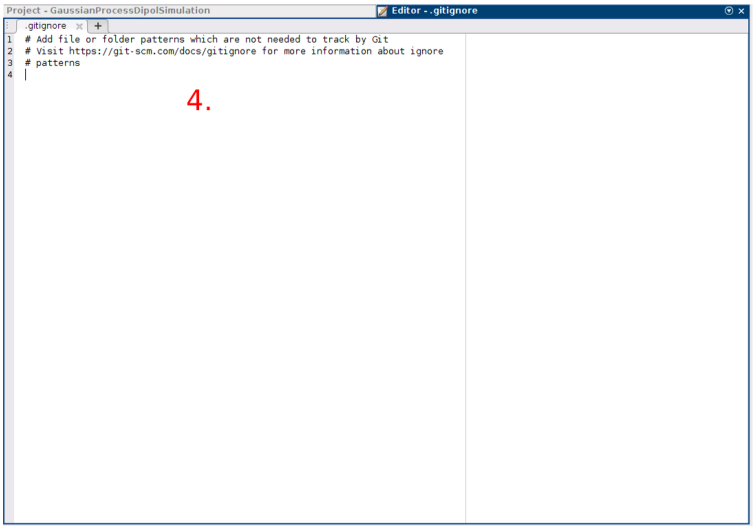
In addition to that it is possible to delclare several ignore cases to Git. So certain directories or file types are not touched or are left out from source control. This is done in .gitignore file. The must be placed in the sandbox folder too.

From the sandbox directory enter in the Matlab command prompt edit .gittatributes and edit .gitignore and save both files. The files are not shown in Current Folder pane (hidden files). Edit both files in the Matlab editor and save the files.

Third step:

1. Add common Matlab file types to .gitattributes.
2. Add Matlab compiler file types to .gitattributes.
3. Add other file types which can be appear during the work to .gitattributes.
4. Add ignore cases to .gitignore if needed.





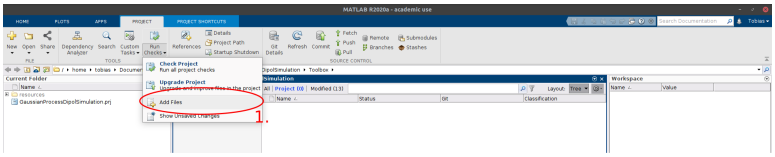
Checkout Project State and Do an Initial Commit

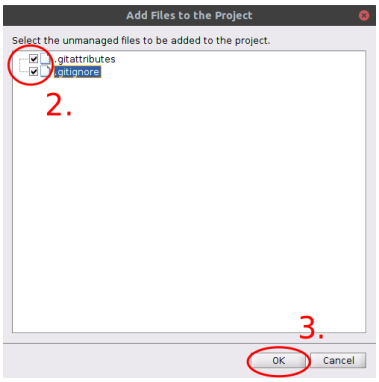
The main part is done. It just needs a few further step to save the work and add the created files to the project.

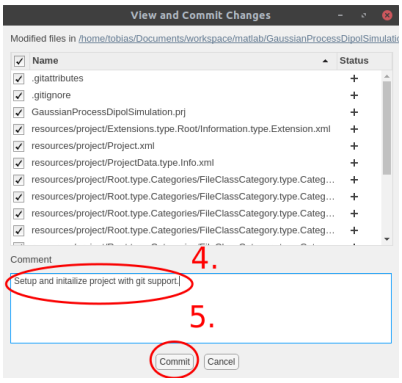
Fourth step:

1. Add created files to the project. In the PROJECT tab under TOOLS section click Run Checks > Add Files.
2. Check the files to add to the project.
3. Click OK.
4. Right click in the white space of Current Folder pane and click Source Control> View and Commit Changes... and add commemt to the commit.
5. Click Commit.

The project is now initialized.



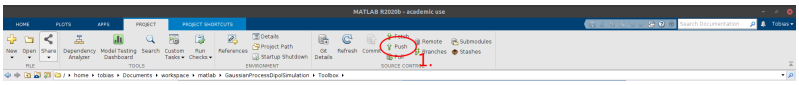


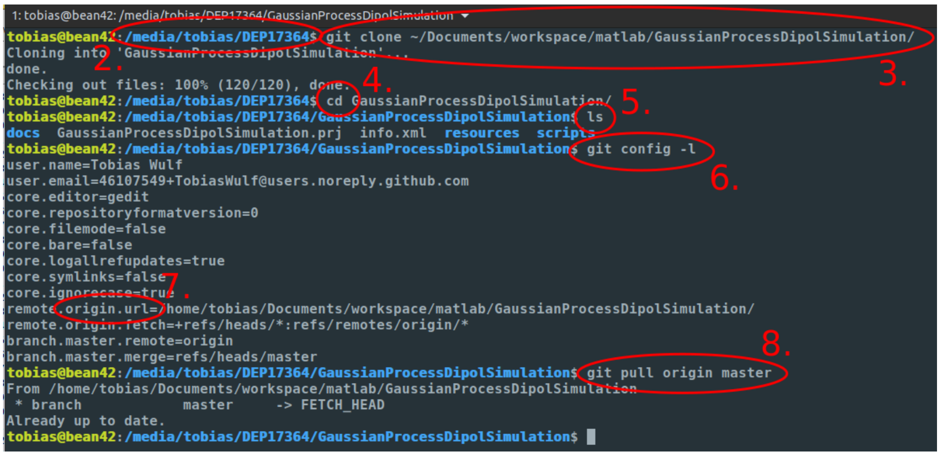


Push to Remote and Backup

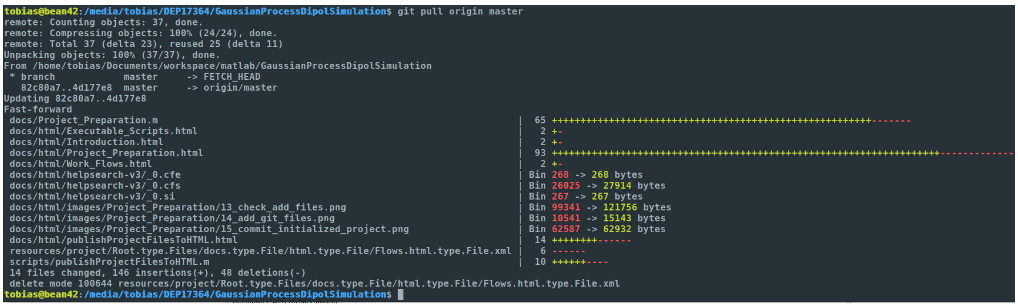
The project is ready to work with. Finally it needs a backup meachnism to save the done work after closing the Matlab session. Git and how the project is build up provide an easy way to make backups.

1. Push the committed changes to remote repository.
2. Insert a backup medium e.g. USB stick and open a git terminal there.
3. Clone the project remote repository from project directory.
4. Change the directory to cloned project.
5. Check if everything was cloned.
6. Check if the remote url fits to origin.
7. Pull from remote to check if everything is up to date.





If further changes are comitted to the project push again to the remote from Matlab environment and repeat update the backup from time to time by inserting your medium and make a fresh pull. Change the directory to the folder and just pull again. See below as an example how does it look like.



Port Remote Repository to GitHub

The remote repository is ported to GitHub laterly. Therfore some minimal changes are made manually to the local repository.

1. According to new rules on GitHub the master branch is renamed to main.
2. Due to that a new upstream is set to origin/main from origin/master
3. To fetch all casualties a merge was needed from origin/main on local main. The orign/master reference was included.
4. Change remote repository to GitHub URL https://github.com/TobiasWulf/GuassianProcessDipolSimulation.git
5. At the moment the GitHub repository is private and not visible in the web. After finishing the general work the repository will be set to publish in consultation with HAW TMR research project and team.
6. After publish on GitHub, clone or fork to work with.
7. The source code is hosted under MIT license.
8. Use GitHub flows to clone or fork and push changes to backup done work.
9. Toolbox folder is not needed anymore because remote is elswhere now
10. Re clone from remote to get new structurew without Toolbox folder

Created on September 30. 2020 by Tobias Wulf. Copyright Tobias Wulf 2020.