GIT

# Git Basics for Local Repository:

Starting a new repository:

* right click on the folder, select “Git Bash here”.
* Type “git init” to initialize a repository

Checking the status of the repository:

* Type “git status”
* Tells which files are tracked, untracked or have been edited

Adding a file to be tracked:

* Type “git add <FILE NAME>”
* This stages the file

Adding all files to be tracked:

* Type “git add .”

Adding all of a specific file to be tracked:

* Type “git add \*.html”
* Adds all the .html files

Committing to the repository (LONG METHOD):

* Type “git commit”
* Type “I” for insert mode
* Type your commit message at the top
* Click esc to exit insert mode
* Type “:wq” to commit

Committing to the repository (SHORT METHOD):

* Type “git commit –m ‘MESSAGE’ ”
* You first need to add the file then commit it

Staging files and committing all at once:

* Type “git commit –a –m ‘MESSAGE’ ”

Show the commit history:

* Type “git log”

Generating a file:

* Type “touch <FILENAME>”

Ignoring certain file types for a commit:

* Type “touch .gitignore”
* Edit the file and add the file type to be ignored eg “\*.log”

# Branching and Merging

Create a branch:

* Type “git branch <BRANCH NAME>”

Switch to a branch:

* Type “git checkout <BRANCH NAME>”
* Switches to the branch, changes all files in the folder
* Master is the name of the master branch

Merge branches

* Go to the master branch (git checkout master)
* Type “git merge <BRANCH NAME>”
* If there is a conflict it will be shown in the file:
  + <<<<< HEAD
  + Other branch code
  + \*\*\*\*\*\*\*\*\*\*
  + Master branch code
  + >>>>>>>> MASTER
* You need to manually fix the file

# Stash

If you make changes to a branch but do not commit the changes, and you want to switch to another branch, these uncommitted changes can cause problems In the other branch. You therefore stash the uncommitted changes before you switch branches, when you switch back you can re-apply these stashed changes

Stashing uncommitted changes:

* Type “git stash”

Re-applying the stashed changes:

* Type “git stash apply”

# Remote Repository

To see what remote repositories are available:

* Type “git remote”

Clone a remote repository to the current location:

* Type “git clone <URL>”
* Clones the repo to the current location
* Use cd to change the directory

Determining the URL of the repository:

* Type “git remote -v”
* Will give the push and fetch URLs

Get the latest changes of the repository:

* Type “git fetch origin”
* You need to manually merge

Get the latest changes and automatically merge:

* Type “git pull origin”

Commit changes to the server:

* Type “git push origin master”
* Commits the changes to the master branch

# GIT on Matlab (copied from mathworks site)

1. If you do not already have a .gitattributes file in your sandbox folder, create one at the MATLAB command prompt:

edit .gitattributes

1. Add these lines to the .gitattributes file:

\*.mat -crlf -diff -merge

\*.p -crlf -diff -merge

\*.slx -crlf -diff -merge

\*.mdl -crlf -diff -merge

These lines specify not to try automatic line feed, diff, and merge attempts for these types of files.

1. Check for other file types you use that you also need to register as binary to avoid corruption at check-in. Check for files such as .mdlp, .slxp, MEX-files (.mexa64, .mexmaci64, .mexw64), .xlsx, .jpg, .pdf,.docx, etc. Add a line to the attributes file for each file type you need.

Examples:

\*.mdlp -crlf -diff -merge

\*.slxp -crlf -diff -merge

\*.sldd -crlf -diff -merge

\*.mexa64 -crlf -diff -merge

\*.mexw64 -crlf -diff -merge

\*.mexmaci64 -crlf -diff -merge

\*.xlsx -crlf -diff -merge

\*.docx -crlf -diff -merge

\*.pdf -crlf -diff -merge

\*.jpg -crlf -diff -merge

\*.png -crlf -diff -merge

1. Restart MATLAB so you can start using the Git client.

# MATLAB AND GIT WRAPPER

<https://github.com/manur/MATLAB-git>

can now use git commands in the matlab workspace. Place the git.m file in the local directory or in any directory stored in the PATH variable