



Summer Fellowship Report

On

Spice Simulations using Kicad nightly builds

Submitted by

Akshay NH and Athul MS

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

July 3, 2018

Acknowledgment

We are extremely thankful to Prof . Kannan Moudgalya for guiding and motivating us throughout the FOSSEE fellowship programme. We would also like to thank our mentors Mrs.Gloria Nandihal and Mr.Athul George for their immense support and advice. Moreover, we are grateful to our fellow friends Mudit Joshi and Ashutosh Gangwar for assisting us with their programming knowledge. Lastly, we extend our warm gratitude to the managers and staff of FOSSEE for their co-operation and assistance.

Contents

1	Introduction	3
1.1	About Scilab Signal Processing Toolbox	3
1.2	git	4
1.3	Travis CI	5
2	Structure of the Signal Processing toolbox	7
2.1	Visible Files	7
2.1.1	loader.sce	8
2.1.2	builder.sce	8
2.1.3	demos	8
2.1.4	macros	8
2.1.5	etc	8
2.1.6	jar	8
2.1.7	cleaner.sce	9
2.1.8	testfile.sce	9
2.1.9	unloader	9
2.1.10	help	9
2.2	Hidden Files	9
2.2.1	.gitignore	9
2.2.2	.travis.yml	10
3	Issues faced and Solutions	11

Chapter 1

Introduction

1.1 About Scilab Signal Processing Toolbox

The Scilab Signal Processing toolbox that we are working on has around 280 functionalities. We had been given a task of fixing and developing around 85 functions of the toolbox. This toolbox tries to match the toolbox of same kind in other computational softwares like Octave or MATLAB in almost all of the cases. There are few exceptions in Scilab ,for example, the function "filter" in scilab accepts only real vectors or matrices as input arguments but not so in the case of MATLAB and Octave. But these limitations on the output are seen quite less and we have worked on it, seeing to that the values are matched.

The toolbox has algorithms with highly diverse complexities in the implementations. There are algorithms ranging from simple implementation of windows to complex signal processing algorithms like MUSIC(pmusic and rootmusic) algorithms, fir1, fir2 and icceps.

Almost fifteen windows necessary for signal processing have been added to the toolbox.

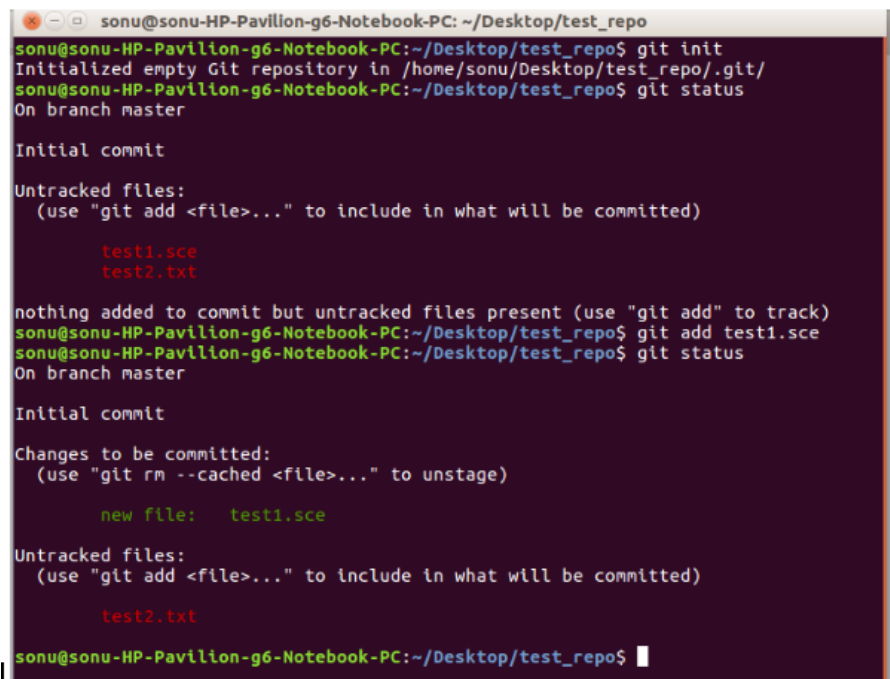
We have also added all the necessary documents and help files for each and every functions that we have worked on.

So ,Scilab being an open source software, most of the signal processing algorithms that are implemented in MATLAB can be implemented easily for free using our toolbox.

1.2 git

Git is a version control system for tracking changes happening in files of computer, Git along with web based hosting services for repository (like GitHub,) can be used to coordinating work among multiple people across the world.

We have made the toolbox in our systems as git repositories(local repository) to push it to github (to create a remote repository).

A terminal window titled 'sonu@sonu-HP-Pavilion-g6-Notebook-PC: ~/Desktop/test_repo' showing the execution of git commands. The user runs 'git init', which initializes an empty Git repository. Then they run 'git status', which shows they are on the master branch and lists untracked files 'test1.sce' and 'test2.txt'. They then run 'git add test1.sce', which stages the file for commit. A final 'git status' shows 'test1.sce' as a new file to be committed and 'test2.txt' as still untracked.

```
sonu@sonu-HP-Pavilion-g6-Notebook-PC: ~/Desktop/test_repo
sonu@sonu-HP-Pavilion-g6-Notebook-PC:~/Desktop/test_repo$ git init
Initialized empty Git repository in /home/sonu/Desktop/test_repo/.git/
sonu@sonu-HP-Pavilion-g6-Notebook-PC:~/Desktop/test_repo$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        test1.sce
        test2.txt

nothing added to commit but untracked files present (use "git add" to track)
sonu@sonu-HP-Pavilion-g6-Notebook-PC:~/Desktop/test_repo$ git add test1.sce
sonu@sonu-HP-Pavilion-g6-Notebook-PC:~/Desktop/test_repo$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   test1.sce

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        test2.txt

sonu@sonu-HP-Pavilion-g6-Notebook-PC:~/Desktop/test_repo$
```

Figure 1.1: git commands

1.3 Travis CI

Travis is a continuous integration tool , where we can deploy and test our projects which are in the github. We linked our github repository to our travis CI account and then we initiated a build in travis. Travis just looks into the .travis.yml file in the repository , and based on the script that we have mentioned , it builds our project.



Figure 1.2: The basic configuration of the travis engine

We basically build the testfile.sce of the toolbox in Travis CI, which is used to validate the functions present in the toolbox.

As seen in .travis.yml file, commands:

`sudo apt-get install scilab` ...installs scilab in travis engine

`Scilab -nw -f testfile.sce` ...executes testfile.sce in a scilab window on the terminal without gui.

On successful completion of the build we get an indication as shown:

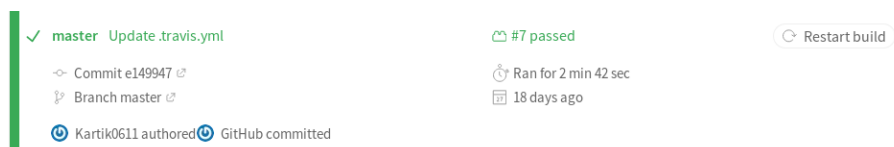


Figure 1.3: Successful build on travis

If the build fails, then we get an indication as follows:

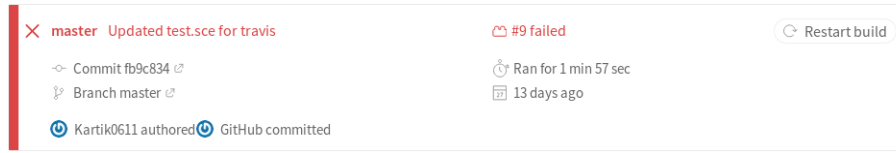


Figure 1.4: Failure of a build in travis

It indicates that we have to revisit the testfile to fix the errors or bugs.

After each and every build, be it successful or failed builds, we get notification mails from travis regarding the commit information and the build status.

So each time, we triggered the builds by the entering the desired commit messages of the git repository, to test the toolbox.

Chapter 2

Structure of the Signal Processing toolbox

The toolbox has two types of files:

2.1 Visible Files

- loader.sce
- builder.sce
- demos
- etc
- jar
- cleaner.sce
- testfile.sce
- unloader.sce
- README.md
- help

The primary step that one has to do, to use this toolbox for the first time is that , one has to essentially go through the README.md file of the toolbox. This file has the text providing the guidance on how to use the toolbox.

2.1.1 loader.sce

Its basically a file that calls for the function `FOSSEE_Signal_processing_toolbox.start` present in the `etc` folder. This has to be executed first on opening the toolbox, using the command `"exec loader.sce"`. in the scilab console.

2.1.2 builder.sce

This file builds the macros,help and the loader.sce files. Type the command `"exec builder.sce"` in the scilab console to execute this file. Both these have to be executed every time, to load the toolbox for usage.

2.1.3 demos

This folder includes the examples taken from <https://in.mathworks.com/help/signal/examples/dft-estimation-with-the-goertzel-algorithm.html>, which is a function used for DFT Estimation with the Goertzel Algorithm.

2.1.4 macros

Its the main folder of the toolbox, which has all the signal processing functions of the toolbox.

2.1.5 etc

Its a folder which consists of `FOSSEE_Signal_processing_toolbox.start` and `FOSSEE_Signal_processing_toolbox.quit` files. These files are used to start and exit files of the toolbox.

`.start` file It will run a script when loader.sce is run from root toolbox directory. It will run loader files in all of the directories and link to important library.

`.quit` file It will run a script when unloader.sce is run from root toolbox directory. It will unlink all of the important libraries.

2.1.6 jar

This folder has `scilab_en_US.jar` file. This file is basically a Java Archive package file format typically used to aggregate many Java

class files and associated metadata and resources (text,images etc.) into a file for distribution.

2.1.7 cleaner.sce

This file is generated by builder.sce. On executing this file, we actually delete the loader.sce and the unloader.sce files. One caution to the users is that **do not edit this file**.

2.1.8 testfile.sce

This file is used to validate all the functions present in the toolbox macros. This file is executed in the Travis CI (Continuous Integration).

2.1.9 unloader

It unloads the toolbox.

2.1.10 help

It contains the help files of all the functions present in the toolbox(macros) as .xml files.

2.2 Hidden Files

These files have to included in the git repository before pushing it to the github. To view these files in the git repository use the command: "ls -a" on the terminal in the git repository directory.

- .gitignore
- .travis.yml

2.2.1 .gitignore

This file enables the user to explicitly tell git to ignore certain files. In our toolbox, we have included just a statement "*~", indicating that all the file names having the sign "~"("tilde") in the end(or suffixed) are ignored by git (when intentionally gone untracked).

2.2.2 .travis.yml

This file is included to facilitate the builds we can trigger in Travis CI.

Travis CI actually provides a default build environment and a default set of steps for each programming language. We can customize any step in this process in .travis.yml. Travis CI uses .travis.yml file in the root of our repository to learn about our project and how we want our builds to be executed. .travis.yml can be very minimalistic or have a lot of customization in it.

```
1 language: scilab
2
3 before_install:
4 - sudo apt-get install scilab
5
6
7
8
9 script:
10 - scilab -nw -f testfile.sce
```

Figure 2.1: .travis.yml file

The above figure shows the different sections involved in the .travis.yml file.

So each time when a build is triggered by the user, traxis engine actually downloads scilab (as per before_install section) and then a new scilab instance is opened in the terminal without any gui (-nw flag) and executes testfile.sce in the toolbox (because of the -f flag used here).

Chapter 3

Issues faced and Solutions

On working on the toolbox, we had faced many issues which we tried to counter them with our own automatations. The issues include :

- GFORTRAN lib issue

The version of Scilab that has to be used is Scilab 5.5.2 , to run these toolboxes.

We installed Scilab from the official website <http://www.scilab.org/download/previous> by choosing Linux OS 64-bit architecture.

To load the FOSSEE_Scilab-Octave-Interface_toolbox, we had to fix an issue.

The library file libgfortran.so.3 present in the location "scilab-5.5.2/lib/thirdparty/redist/" had to be moved to the previous directory to load it. This step was important to load the FOSSEE_Scilab-Octave-Interface_toolbox.

```

-->pwd
ans =

/home/shashi/Downloads/FOSSEE_Scilab-Octave_Interface_Toolbox

-->exec loader.sce

--> // This file is released under the 3-clause BSD license. See COPYING-BSD.

--> // Generated by builder.sce: Please, do not edit this file

-->try
--> getversion("scilab");
-->catch
--> error("Scilab 5.0 or more is required.");
-->end;

-->exec(get_absolute_file_path("loader.sce")+ "etc/" + "FOSSEE_Scilab-Octave_Interface_Toolbox.start");
Start FOSSEE Scilab-Octave Interface Toolbox
Load macros
Load gateways
link(lib_path + "/liboctave.so");
!--error 236
ink: The shared archive was not loaded: /home/shashi/Desktop/scilab-5.5.2/lib/thirdparty/redis/libgfortran.so.3: version `GFORTRAN_1.4' not
it line 50 of exec file called by :
tb_Octave_Interface_Toolbox.start")
it line 10 of exec file called by :
exec loader.sce

-->a=[1 2 3]
a =

1. 2. 3.

-->b=callOctave('transpose',a)
!--error 4
Undefined variable: callOctave

```

Figure 3.1: GFORTRAN 1.4 issue

- The output of few functions were matrices of huge dimensions. These matrices couldnt be included in the testfile.sce. So we worked on an automation of copying this matrix into a new text-file (if the matrix is square matrix, then we reshaped the matrix by stacking it column wise using command "matrix ") using the command-"fprintfMat("name_of_file ",desired_variable)" and saved it in macros folder . Then in the the testfile.sce , we scan this text file using the command "fscanfMat("name_of_file")" and store it in dummy variable for comparison.
- Few functions in the toolbox had same names as that of the inbuilt functions. This creates an issue as once the toolbox is built, we could not access the inbuilt functions of scilab. So we changed the names of these functions in the toolbox.The functions include wind.sci and oct_interp.sci.

- Also while porting the functions from octave to scilab, we faced an issue regarding the inconsistent operations of vectors. In octave, we can perform addition and subtraction between a row vector and column vector as shown

$$[1 \ 3 \ 4] - \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$

This operation is not possible in scilab and a small operation of such kind in a 350 line code was encountered. We figured out the operation going on in the backend and added that algorithm in to the corresponding functions. This operation was seen in the function "residue".

- Few functions that had to be ported to scilab ,from octave, had long tree of dependencies,i.e., these functions had inbuilt functions of octave which had to be built in scilab from scratch.(We called it as "Bottom Up approach") .

Reference

- http://docs.kicad-pcb.org/stable/en/getting_started_in_kicad.pdf
- <https://github.com/KiCad/kicad-source-mirror>
- <https://github.com/FOSSEE/eSim-Kicad-Simulations>
- <https://forum.kicad.info/>
- <http://www.ecircuitcenter.com/Basics.htm>
- <http://www.ecircuitcenter.com/SPICESummary.htm>