Semester-Long Internship Report

On

Scilab Signal Processing Toolbox development

Submitted by

Chandra Pratap

Under the guidance of

Prof. Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

and mentor

Ms. Rashmi Patankar

July 25, 2024

Acknowledgment

I would like to express my heartfelt gratitude to everyone who contributed to the successful completion of my internship. First and foremost, I am deeply thankful to my mentor Ms. Rashmi Patankar, Prof. Kumar Appaiah, and Prof. Kannan M. Moudgalya for their invaluable guidance, support, and encouragement throughout this journey. Their expertise, patience, and willingness to share their knowledge have been instrumental in my learning and growth. They provided me with the tools and confidence needed to tackle challenges and excel in my project.

Furthermore, I am grateful to the organization of FOSSEE for providing me with this wonderful opportunity. The experience has been instrumental in shaping my career aspirations and has given me a clearer understanding of the professional world.

I would like to acknowledge my professors and academic advisors for their continuous support and guidance, which laid the foundation for my internship. Their encouragement and advice have been invaluable throughout this process.

Lastly, I extend my appreciation to my family and friends for their unwavering support and encouragement. Their belief in my abilities and their constant motivation have been a source of strength for me. This experience has been incredibly enriching, and I am grateful to all who made it possible. Thank you all for being a part of this significant phase of my professional journey.

Contents

Chapter 1

Introduction

Scilab is a free and open-source, cross-platform numerical computational package and a high-level, numerically oriented programming language. It can be used for signal processing, statistical analysis, image enhancement, fluid dynamics simulations, numerical optimization, and modeling, simulation of explicit and implicit dynamical systems and (if the corresponding toolbox is installed) symbolic manipulations. Scilab is one of the two major open-source alternatives to MATLAB, the other one being GNU Octave. Scilab puts less emphasis on syntactic compatibility with MATLAB than Octave does, but is similar enough to easily transfer skills between the two systems.

IIT Bombay is leading the effort to popularise Scilab in India and Scilab Signal Processing Toolbox is one its endeavours towards the cause. This effort is part of the Free and Open source Software for Science and Engineering Education (FOSSEE) project, supported by the National Mission on Education through ICT of the Ministry of Education.

FOSSEE Scilab Signal Processing Toolbox is a comprehensive suite designed for the analysis, manipulation, and visualization of signals, developed and maintained by FOSSEE, IIT Bombay. It offers a wide range of functions that cover fundamental and advanced signal processing techniques, including filtering, spectral analysis, and time-frequency analysis. Users can implement various types of digital filters (such as FIR and IIR), perform Fourier transforms, and analyze the frequency content of signals. The toolbox also supports wavelet transform methods, which are essential for non-stationary signal analysis. With its intuitive interface and extensive testing, the Signal Processing Toolbox is a powerful tool for engineers, researchers, and educators working in fields like telecommunications, audio processing, and biomedical signal analysis.

Chapter 2

Signal Processing Toolbox development

2.1 Overview

The signal processing toolbox has been built and tested on Ubuntu 14.04 and the instructions to install the toolbox on one's own system can be found here. The toolbox is comprised of an amalgamation of various functions (listed in the directory `FOSSEE-Signal-Processing-Toolbox/macros/`) of mainly two categories: functions that perform the required computation natively in Scilab, and functions that pass the input from the user to be processed by Octave.

To convert functions of the latter type to the former type has been the primary goal of this internship project. The motivation for this is multifaceted, but a few are listed as following:

- Calling Octave's signal processing function using Scilab requires an intermediary toolbox, the FOSSEE Scilab-Octave Toolbox. This makes FSOT a dependency for Scilab Signal Processing Toolbox, which is unoptimal.
- Using Scilab-native code for computations is much more performant than having Octave deal with them and simply using Scilab as an interface.
- The FOSSEE Scilab-Octave Toolbox is still in development and hence, somewhat lackluster in features. For example, any functions that have boolean values, structs, or graphs/images as input or output cannot be accessed using the toolbox. This significantly diminishes its usefulness while handling signal processing functions.

This process of re-writing functions in native Scilab code involves some steps, which are explained from here onwards.

2.2 Development workflow

2.2.1 Coding

This is the most important (thus, the most tedious) part of the entire workflow. Since the signal processing functions in their present state rely on Octave to perform the required computations, it suffices to implement these functions in Scilab akin to how they're written in Octave.

Since Octave (like Scilab) is an open-source project, the source code for these functions is publicly available. Hence, it is a viable strategy to go through this source code, study the implementation, and implement the same in Scilab. This is the strategy that I adopted for this project and it served me good enough to completely re-write 26 functions within the duration of this internship.

However, it is important to note that a copy-paste job between Octave and Scilab will not work with this approach. This is because:

(i) Every Octave function is not available in Scilab. For instance, the function `fftfilt` performs FFT-based FIR filtering using overlap-add method and is used extensively in Octave's signal processing functions, but it isn't available in Scilab and needs to be reproduced using Scilab's `filter` function.

(ii) There does not exist a one-one relationship between the functions in Octave and Scilab. For instance, Octave's `fft(A)` function performs the required fourier transform along the first non-singleton dimension of a matrix `A` by default, but this is different from Scilab's default behaviour and needs to be replicated using `fft(A, -1, firstnonsingleton(A))`.

However, it is still worthwhile to note that more or less most of the syntax is similar between Scilab and Octave and that is what makes it possible to write a good number of functions in a short time period, with little to no knowledge of the overall logic behind the functions themselves.

2.2.2 Documentation

Since the functions are to be implemented akin to how they're in Octave, it is a fine strategy to use the documentation in Octave as a reference for the documentation in the Scilab functions. The documentation for a function sits just below the function's declaration, and is written as one big comment block. There are four components to a function's documentation:

1. Calling Sequence: The order of evaluation for the function for the set of given paramaters.

2. Parameters: The expected values for the function's parameters. This section outlines the type as well as the acceptable values for these parameters.

3. Description: A comprehensive description of the function. This section outlines default behaviours, expected input and output and how to interpret them, dependencies and other such details.

4. Examples: An example to demonstrate the correct usage of the function.

It is worth nothing that in the process of re-writing functions to use Scilab code, the documentation part does not require a lot of modifications because the intended behaviour for the function is, most of the times, already well-documented and in-line with their Octave counterparts.

2.2.3 Testing

Unlike the documentation section, this part of the development procedure is non-existent in both the existing implementations and the Octave function files. Hence, it is the function author's responsibility to thouroughly test the code to ensure a robust implementation. There are three Scilab functions that can be employed to acheive this:

1. `assert_checkerror(instr, expectedmsg)`: Takes as input two strings, `instr` which is an expression that throws an error on evaluation and `expectedmsg` which is the expected error message when `instr` is evaluated. Throws an error in the case of a mismatch.

2. `assert_checkequal(computed, expected)`: Takes an input two Scilab objects (like vectors, matrices, real/imaginary numbers) and throws an error if equality doesn't hold between these objects.

3. `assert_checkalmostequal(computed, expected, reltol)`: Takes as input two Scilab objects `computed` and `expected`, and a real number `reltol`. Throws an error if the difference between `computed` and `expected` is greater than the relative tolerance denoted by `reltol`.

Using these three Scilab functionalities, the signal processing functions can be tested to a satisfiable degree. The tests appear at the bottom of the `.sci` function files and are commented out after their validity is asserted. This is equivalent to how function tests are written in the GNU Octave project.

2.3 Current status

Using the workflow outlined prior, I have been able to implement the following 29 functions in Scilab within the duration of this internship:

1. `arma_rnd.sci`

2. `autoreg_matrix.sci`

3. `cceps.sci`

4. `clustersegment.sci`

5. `detrend1.sci`

6. `dst1.sci`

7. durbinlevinson.sci
8. fftshift1.sci
9. filter2.sci
10. hurst.sci
11. ifftshift1.sci
12. ifht.sci
13. morlet.sci
14. pei_tseng_notch.sci
15. polystab.sci
16. ar_psd.sci
17. schtrig.sci
18. sinetone.sci
19. sinewave.sci
20. spencer.sci
21. stft.sci
22. synthesis.sci
23. tf2sos.sci
24. tfestimate.sci
25. triang.sci
26. tripuls.sci
27. fwhm.sci
28. fwhmjlt.sci
29. pburg.sci

These functions can be found in my GitHub repository. The other intern for this project, Abinash Singh, has also written around 28 functions which leaves the following 23 functions that still need to be re-written in Scilab:

1. pyulear.sci
2. cl2bp.sci
3. dwt.sci

4. fftw1.sci
5. findpeaks.sci
6. fracshift.sci
7. freqs.sci
8. freqz.sci
9. decimate.sci
10. invfreqs.sci
11. fwht.sci
12. ifwht.sci
13. iirlp2mb.sci
14. impinvar.sci
15. impz.sci
16. invfreq.sci
17. invimpinvar.sci
18. marcumq.sci
19. pburg.sci
20. invfreqz.sci
21. resample.sci
22. sosfilt.sci
23. ultrwin.sci

Chapter 3

Scilab-Octave Toolbox development

3.1 Overview

The FOSSEE Scilab-Octave Toolbox is a practical tool designed to enable the execution of Octave functions directly within the Scilab environment. With this toolbox, users can seamlessly run Octave scripts and functions in Scilab, combining the strengths of both platforms. This enhances the overall computational experience by allowing access to Scilab's powerful visualization and modeling capabilities alongside Octave's extensive numerical computation libraries. The Scilab-Octave Toolbox is a valuable resource for anyone looking to maximize the potential of these two powerful tools.

The Scilab-Octave toolbox has been built and tested on Linux Debian 10, Ubuntu 18.10 and 19.10 (64-bit), Windows 10 (64-bit) with Octave 4.4.1, 5.1.0 and Scilab 6.0.x. The instructions to install the toolbox on one's own system can be found here. The toolbox is currently capable of the following:

- Move matrices and vectors in and out of Octave via Scilab.
- Handle multiple inputs and outputs.
- Handle any size of inputs and outputs.
- Handle input and output of type 'string', 'double' and 'structure'.
- Call native functions of Octave.
- Load packages in Octave.
- Display error messages thrown by Octave.
- Call those functions provided by Octave packages that handle matrices and strings.

3.2 Testing

A crucial part of this internship was dedicated to testing the FOSSEE Scilab-Octave toolbox in light of the signal processing functions. This was done as a way to save time on the process of signal processing toolbox's development, as fixing the Scilab-Octave toolbox means a lesser need to adopt the tedious task of re-writing signal processing functions in Scilab and instead letting Octave handle the computationally challeging parts.

Within the duration of this internship, I was only able to test the FSOT toolbox for those signal processing functions that I had implemented in Scilab. To implement the required tests for these functions, it suffices to copy the 'tests' section from the function's `.sci` file and paste it in that version of the function that uses the FSOT toolbox. Since the tests are comprehensive enough to catch any breakage in the Scilab-only version, they can assure a good amount of validity in the Scilab-Octave version as well.

3.3 Current status

Using the method outlined prior, I have been able to test all 26 of the functions that I have worked on in the signal processing toolbox development project with the Scilab-Octave toolbox, and all of them work fine with the following being the exceptions:

- `clustersegment.sci`: Cannot pass cell objects between Octave and Scilab.
- `schtrig.sci`: Cannot pass boolean values between Octave and Scilab, need to use 0 (for false) and 1 (for true) instead.
- `tfestimate.sci`: Cannot uses functions with graphical output from Scilab.
- `pyulear.sci`: Works partially. Cannot uses for cases with graphical output.

Overall, the following is what still needs to be accomplished for the FSOT toolbox:

- Improve on the robustness of toolbox, like exiting gracefully in the case of errors.
- Handle Octave functions that use boolean, cells, hypermatrices, or some other data type.
- Handle plots generated by Octave.
- Build the toolbox for latest verion of Windows.
- Build the toolbox for the latest versions of Ubuntu and Debian.
- Make the code less prone to segmentation faults and memory leaks.

Chapter 4

Learnings

I have had a lot of great experience from this internship opportunity, some are enumarated below.

1. Technical Skills Enhancement:
    - Gained proficiency in Scilab, Octave, Git and GitHub.
    - Developed advanced coding skills like testing and documentation.
    - Improved understanding of technical concepts and practical applications.
2. Feedback and Continuous Improvement:
    - Learned to receive and act on constructive feedback.
    - Gained an understanding of the importance of continuous learning and improvement.
    - Developed the ability to self-assess and seek opportunities for growth.
3. Communication Skills:
    - Improved written and verbal communication skills through regular text conversations and virtual meetings.
    - Gained experience in collaborative communication within a team.
    - Gained experience in guiding others to new concepts effectively.
4. Professionalism and Work Ethic:
    - Developed a strong sense of professionalism in a workplace setting.
    - Learned the importance of punctuality, reliability, and accountability.
    - Gained experience in maintaining a professional demeanor in various situations.
5. Adaptability and Flexibility:
    - Learned to adapt to new environments and changing circumstances.
    - Gained experience in managing multiple tasks and shifting priorities.
    - Developed resilience and the ability to thrive in a dynamic work environment.

Chapter 5

Conclusion

My internship experience has been profoundly enriching and has significantly contributed to my professional and personal growth. Engaging in the development of the FOSSEE Scilab Signal Processing Toolbox and the FOSSEE Scilab-Octave Toolbox allowed me to apply knowledge in a practical setting, enhancing my skills in software development and problem solving in general. The challenges I encountered and overcame during these projects sharpened my problem-solving skills and provided a deeper understanding of working on open source projects in general.

Working on the Scilab Signal Processing Toolbox involved improving existing functionalities and ensuring the reliability and accuracy of the toolbox. This not only expanded my knowledge of signal processing but also taught me the importance of comprehensive documentation and testing. On the other hand, the Scilab-Octave Toolbox project provided me an opportunity to learn about build systems and software design.

This internship has helped outline my career aspirations. The hands-on experience has prepared me for future professional challenges. I am deeply grateful for this opportunity and confident that the skills and insights gained during this internship will be immensely beneficial in my career development. As I move forward, I am excited to leverage the knowledge and experience acquired to contribute meaningfully to the tech industry, especially the open source world, and achieve my professional goals.

Reference

- Scilab Wikipedia
- FOSSEE Scilab Signal Processing Toolbox
- FOSSEE Scilab-Octave Toolbox