



# Summer Fellowship Report

On

## Scilab Signal Processing Toolbox development

Submitted by

**Abinash Singh**

Under the guidance of

**Prof.Kannan M. Moudgalya**
Chemical Engineering Department
IIT Bombay

Mentor

**Ms. Rashmi Patankar**

July 30, 2024

# Acknowledgment

I am deeply grateful to my mentor Ms. Rashmi Patankar for her invaluable guidance, support, and encouragement throughout my internship with the FOSSEE Team at IIT Bombay.

Their expertise and patience have been pivotal in my learning and professional growth during this period.

I would also like to sincerely thank Prof. Kannan M. Moudgalya and Prof. Kumar Appaih for their insightful guidance, which has significantly shaped my understanding of open-source systems.

Additionally, I wish to express my heartfelt appreciation to my friends who supported me in completing the screening tasks for this internship. Their encouragement and assistance were crucial in overcoming challenges and achieving milestones.

I remain dedicated to contributing to Scilab and other FOSSEE initiatives. In the coming months, my primary focus will be on advancing the signal processing toolkit further. I am enthusiastic about the opportunities ahead and eager to make meaningful contributions to these projects.

I am thankful to everyone who has been part of this journey, supporting and inspiring me along the way. Your unwavering belief in my capabilities has made this experience incredibly rewarding.

# Contents


**1 Introduction** **3**

**2 Signal Processing Toolbox Development** **4**
- 2.1 Overview . . . 4
- 2.2 Development Workflow . . . 5
  - 2.2.1 Reading Octave Implementation . . . 5
  - 2.2.2 Line By Line Translation . . . 5
  - 2.2.3 Comparing Inbuilt Functions And Writing Missing Ones . . . 6
  - 2.2.4 Test And Iterate . . . 6
- 2.3 Current Status . . . 7
  - 2.3.1 Documentation pattern . . . 7
  - 2.3.2 Functions Completed . . . 8
  - 2.3.3 Incomplete implementations and FIXME issues . . . 8

**3 Scilab-Octave Toolbox Devlopment** **9**
- 3.1 Overview . . . 9
- 3.2 Current Status . . . 10
- 3.3 Common Errors And Fixes . . . 11

**4 Learnings** **12**

**5 Conclusion** **13**

# 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 is 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 the Scilab Signal Processing Toolbox is one of its endeavors toward 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 ed- educators working in fields like telecommunications, audio processing, and biomedical signal analysis.

# Chapter 2

# Signal Processing Toolbox Development

## 2.1 Overview

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 follows:

- Calling Octave's signal processing function using Scilab requires an intermediary toolbox, the FOSSEE Scilab-Octave Toolbox. This makes FSOT and Octave a dependency for the Scilab Signal Processing Toolbox, which is non-optimal.
- Using Scilab-native code for computations is much more performant than having Octave deal with them and simply uses Scilab as an interface.
- The FOSSEE Scilab-Octave Toolbox is still in development and hence, somewhat lacking 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

The workflow underwent significant changes as I gained experience during this internship. Here is a summary of the workflow:

- Reading the Octave implementation
- Translating line by line
- Comparing the functions and writing missing ones
- Testing and iterating

### 2.2.1 Reading Octave Implementation

This analysis is crucial for planning our code translation process from Octave to Scilab. It's important to consider that certain sub-functions in Octave may not have direct equivalents in Scilab.

To find the Octave documentation for a desired function, search online and determine which package it belongs to. Typically, functions will belong to one of two packages:

1. Octave's signal package: You can access the source code intended for translation at https://octave.sourceforge.io/signal/ Most functions are written in pure Octave, although some may be implemented in C++.
2. Octave's core package: Download the source code of the latest version of Octave and search for the desired function within the downloaded folder.

The outcomes of this process include:

- Estimation of time constraints.
- Identification of sub-functions that are not available in Scilab.
- Assessment of the complexity involved in the translation process.

### 2.2.2 Line By Line Translation

This task requires meticulous line-by-line translation for successful adaptation. The focus is primarily on syntax differences between Octave and Scilab.

Here are some key differences to note. If you encounter difficulties, refer to the documentation for Octave and Scilab:

- In Octave, if, else, for, while, and switch statements should be closed with endif, endfor, endwhile, and endswitch, respectively. In Scilab, you simply use "end".
- Constants like! , pi, eps, j, true, and false are defined differently: in Octave as !, pi, eps, j, true/false, and in Scilab as ˜ %pi, %i, %eps, %T /%F.

- "print_usage" function is not available in Scilab so replace it with some error("message")
- The overall syntax is quite similar, but these differences are important to keep in mind.

### 2.2.3 Comparing Inbuilt Functions And Writing Missing Ones

Now that you have identified which sub-functions are present in the code and checked their availability in Scilab, there is still a possibility of encountering errors. Scilab and Octave functions with the same name may behave differently.

The best approach is to consult the documentation for both platforms and adjust the implementation accordingly. If the differences are significant, consider writing a wrapper function or creating separate implementations.

Octave Scilab equivalents

| Octave | Scilab |
|---|---|
| length(a) | max(size(a)) |
| string | s̃tring |
| end | $ |

Another example is When comparing the max and min functions between Scilab and Octave, it's important to note a difference: Octave computes min/max along the columns of a matrix, whereas Scilab computes max/min over the entire matrix.

Paying attention to these nuances is crucial to avoid potential errors.

Now, regarding sub-functions that are unavailable in Scilab, you can follow a similar workflow: consult Octave's documentation, obtain the source code, and translate it to Scilab.

Challenges may arise with functions implemented in C++. In such cases, you can extract the algorithm from the C++ code and attempt to implement it directly in Scilab or utilize Scilab's C++ API. Alternatively, consider creating a wrapper C++ function to interface with the code and dynamically link it to Scilab. However, I don't recommend this if you do not have a good command of C++.

### 2.2.4 Test And Iterate

This is a crucial but often tedious part of the workflow. To lighten the load, you can find test cases at the bottom of the Octave source code files. Attempt these first, and If you encounter difficulties running some test cases, don't worry—create your own examples instead.

Compare the outputs of these examples with those from Octave and your implemented function. Ensure to write diverse test cases covering different calling sequences, aiming for at least one example for each type.

## 2.3 Current Status

Following the previously outlined workflow, I have successfully translated the following:

- 27 functions independently.
- 2 functions in collaboration with another intern.

In total, our team has translated approximately 60 functions.
Each function is accompanied by documentation at the top of its file and test cases at the bottom.

Furthermore, in my repository, you can find additional test cases and documentation in a README file for each function. As part of my plans for the next semester's DSP lab, I intend to include real-world examples in these README files, beyond the scope of this internship.

### 2.3.1 Documentation pattern

Since the functions are intended to mirror their implementation in Octave, it's prudent to use Octave's documentation as a reference for documenting the corresponding Scilab functions.

Additionally, consult Matlab's documentation for these functions, as Octave may have certain bugs. If your test cases fail despite following all steps, compare your code's output with Matlab's to troubleshoot further.

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 parameters.
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 behaviors, expected input and output, and how to interpret them, dependencies and other such data
4. Examples: An example to demonstrate the correct usage of the function.

It is worth noting 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 behavior for the function is, most of the time, already well-documented and in line with their Octave counterparts.

All of my work is available in my GitHub repository GITHUB

### 2.3.2 Functions Completed

| S.No | Function | Dependencies/custom functions |
|---|---|---|
| 1 | fftn | |
| 2 | fht | |
| 3 | fft1 | |
| 4 | fft21 | |
| 5 | fftconv | |
| 6 | ifft1 | |
| 7 | ifft2 | |
| 8 | ifftn | |
| 9 | idct2 | |
| 10 | idct1 | idct1 |
| 11 | idst1 | dst1 |
| 12 | czt | fft1 , ifft1 |
| 13 | xcorr2 | |
| 14 | shanwavf | |
| 15 | rceps | fft1 , ifft1 |
| 16 | pulstran | |
| 17 | hilbert1 | fft1 , ifft1 , ipermute |
| 18 | grpdelay | fft1 |
| 19 | pwelch | fft1 |
| 20 | tfe | pwelch |
| 21 | mscohere | pwelch |
| 22 | cpsd | pwelch |
| 23 | cohere | pwelch |
| 24 | arch_fit | autoreg_matrix , ols |
| 25 | arch_test | ols |
| 26 | spectral_xdf | fft1 |
| 27 | spectral_adf | ifft1 , fft1 |
| 28 | unwrap2 | ipermute |
| 39 | cplxreal | cplxpair , ipermute |

### 2.3.3 Incomplete implementations and FIXME issues

No issues

# Chapter 3

# Scilab-Octave Toolbox Devlopment

## 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 system can be found on 2 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 Current Status

During my internship, I dedicated two weeks to resolving issues with the FOSSEE Scilab-Octave toolbox to make it compatible with the latest operating systems, Octave, and Scilab versions.

This effort aimed to streamline the development of the signal processing toolbox by avoiding the need to rewrite signal processing functions in Scilab. Instead, Octave could handle the computationally intensive tasks.

Throughout that time, my primary focus was on fixing the FSOT toolbox for the newest versions of Octave and Scilab. Despite my efforts, I encountered challenges that prevented me from successfully resolving all issues within the allotted time. However, I remain committed to continuing this work beyond the internship.
Key challenges I faced during debugging included:

- Insufficient documentation for the Octave C++ API, necessitating the examination of source code for insights. Changes in the Octave C++ API across recent versions led to unresolved symbol errors.
- Compatibility issues have resulted in segmentation faults when invoking the octave_fun function after building the toolbox for Scilab 2024 and Octave 8.4 .
- Stack trace is complex to decode.
- The need to redesign data structures to efficiently manage intermediate values.

Analyzing the stack trace and test file in the src folder concludes that there is no issue in octave c++ API, There are some new changes in some Scilab's object files in the latest version. The stack trace is given below :

```
[abinash-INBOOK-Y2-PLUS:16896] Signal: Segmentation fault (11)
[abinash-INBOOK-Y2-PLUS:16896] Signal code: Address not mapped (1)
[abinash-INBOOK-Y2-PLUS:16896] Failing at address: 0xec3f1870

Call stack:
   1: 0xe66b29 <JVM_handle_linux_signal>        (/usr/lib/jvm/default-java/lib/server/libjvm.so)
   2: 0x45320  < >                              (/lib/x86_64-linux-gnu/libc.so.6)
   3: 0x30b7   <sci_octave_fun>                 (/home/abinash/fsot_work/fossee-scilab-octave-toolbox/sci_gateway/cpp//libscilab_octave.so)
   4: 0x4996b3 <types::WrapCFunction::call(std::vector<types::InternalType*, std::allocator<types::InternalType*> >&, std::unordered_map<std::__cxx1
raits<wchar_t>, std::allocator<wchar_t> >, types::InternalType*, std::hash<std::__cxx11> (/usr/lib/x86_64-linux-gnu/scilab/libsciast.so.2024)
   5: 0x48eff8 <types::Callable::invoke(std::vector<types::InternalType*, std::allocator<types::InternalType*> >&, std::unordered_map<std::__cxx11::
ts<wchar_t>, std::allocator<wchar_t> >, types::InternalType*, std::hash<std::__cxx11::b> (/usr/lib/x86_64-linux-gnu/scilab/libsciast.so.2024)
   6: 0x1b15fb <ast::RunVisitorT<ast::ExecVisitor>::visitprivate(ast::CallExp const&)> (/usr/lib/x86_64-linux-gnu/scilab/libsciast.so.2024)
   7: 0x193e55 <ast::RunVisitorT<ast::ExecVisitor>::visitprivate(ast::SeqExp const&)> (/usr/lib/x86_64-linux-gnu/scilab/libsciast.so.2024)
   8: 0x1da7d0 <StaticRunner::launch()>         (/usr/lib/x86_64-linux-gnu/scilab/libscilab-cli.so.2024)
   9: 0x1ce7ca <RunScilabEngine>                (/usr/lib/x86_64-linux-gnu/scilab/libscilab-cli.so.2024)
  10: 0x25f3   <main>                           (/usr/bin/scilab-bin)
  11: 0x2a1ca  < >                              (/lib/x86_64-linux-gnu/libc.so.6)
  12: 0x2a28b  <__libc_start_main>              (/lib/x86_64-linux-gnu/libc.so.6)
  13: 0x2fb5   < >                              (/usr/bin/scilab-bin)
End of stack
```

For further details on this toolbox, including its source code and project report, please refer to the relevant documentation.

## 3.3 Common Errors And Fixes

- **libfun. so not found** : Ensure libfun. so is added to the linker's path folder.
- **Error in -loctave and -loctaveinterp**: Add your Octave installation path to the linker's configuration file /etc/ld.so.conf.d/.
- **Symbol errors related to FFTW**: Install Scilab via the command line or attempt to build it from the source.

Refer to the FSOT toolbox GitHub repository for further assistance.

# Chapter 4

# Learnings

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

1. Technical Skills Enhancement:
    - Gained proficiency in Scilab, Linux, dynamic linking, c++ 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 growth opportunities.
3. 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 sit– uations.
4. 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- menu.

# Chapter 5

# Conclusion

In conclusion, my internship experience at FOSSEE, IIT Bombay, working on the development of the Scilab Signal Processing Toolbox and contributing to the Scilab-Octave Toolbox has been immensely rewarding and educational. Throughout this internship, I have made significant contributions to the open-source community, particularly in enhancing the functionality and performance of the Signal Processing Toolbox.

The primary focus of my work has been on translating Octave-based signal processing functions into native Scilab code. This transition aims to improve the toolbox's efficiency and reduce dependencies on external tools like the Scilab-Octave Toolbox. Through meticulous workflow planning, detailed function translations, rigorous testing, and iterative improvements, I have successfully converted approximately 29 functions to Scilab, alongside documenting each function comprehensively.

Furthermore, my involvement in resolving issues with the Scilab-Octave Toolbox has contributed to enhancing its compatibility with modern operating systems and software versions. This effort ensures a seamless integration of Octave functionalities within the Scilab environment, thereby expanding the usability and effectiveness of both tools for scientific and engineering applications.

Looking ahead, I am committed to continuing my contributions to FOSSEE projects, particularly in further developing the Signal Processing Toolbox. I plan to integrate real-world examples into the toolbox and continue collaborating with my peers to address any remaining challenges and implement new features.

I am deeply grateful to my mentor Ms. Rashmi Patankar for her unwavering support and guidance throughout this internship. Her expertise and encouragement have been instrumental in my professional development. I also thank Prof. Kannan M. Moudgalya and Prof. Kumar Appaih for their invaluable insights and my friends for their continuous support.

In conclusion, this internship has not only strengthened my technical skills but also reinforced my commitment to contributing to the advancement of open-source software for scientific and engineering education. I look forward to applying the knowledge and experience gained here in my future endeavors.

Signal processing Toolbox development

**Reference**

- https://github.com/abinash108/Signal-processing-toolkit-devlopment-
- https://github.com/FOSSEE/fossee-scilab-octave-toolbox
- https://octave.sourceforge.io/pkg-repository/signal/
- https://scilab.in/fossee-scilab-toolbox/signal-processing-toolbox