Semester Long Internship Report

On

Scilab Control System Toolbox Development

Submitted by

Nikitha D

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

Mentor

Ms. Rashmi Patankar
Project Manager, Scilab Team, FOSSEE Project, IIT Bombay

Faculty Guide

Dr.V.Krishnaveni
Professor and Head,Department of ECE,
PSG College of Technology

September 1, 2025

Acknowledgment

I would like to express my sincere gratitude to the FOSSEE Team at IIT Bombay for giving me the opportunity to contribute to the development of the Scilab Control System Toolbox during my internship. This experience has been valuable in enhancing both my technical knowledge and practical skills.

I am thankful to Prof. Kannan M. Moudgalya, Department of Chemical Engineering, IIT Bombay, for creating this platform that enables students to apply their learning through open-source contributions and meaningful projects.

I extend my thanks to my mentor, Ms. Rashmi Patankar, Project Manager, Scilab Team, FOSSEE Project, for her constant guidance, support, and constructive feedback throughout the internship. Her insights helped me improve and gain confidence in my work.

I would also like to acknowledge the FOSSEE Team for maintaining a collaborative and motivating environment that encouraged continuous learning and growth.

My gratitude also goes to the Department of Electronics and Communication Engineering, PSG College of Technology, for providing me with a strong academic foundation and encouraging me to explore opportunities beyond classroom learning.

Finally, I thank Dr. V. Krishnaveni, Professor and Head of the Department of ECE, PSG College of Technology, for her constant guidance and encouragement, which inspired me to explore practical applications of Control Systems and contribute to this open-source project.

Contents

Chapter 1

Introduction

The Free/Libre and Open Source Software for Education (FOSSEE) is a national project supported by the Ministry of Education, Government of India. It is part of the National Mission on Education through ICT. The main aim of FOSSEE is to promote the use of free and open-source software in colleges and universities. By encouraging the use of such tools, the project helps reduce the need for expensive commercial software. FOSSEE works on different activities like software development, training programs, and creating learning materials. Through these efforts, it supports better learning and research in technical fields. It also motivates students and teachers to take part in the open-source community and gain hands-on experience with real-world projects.

Scilab is one of the software tools promoted by FOSSEE. It is a free and open-source platform used for scientific and engineering calculations. Scilab has its own programming language and supports a wide range of applications such as signal processing, image pro cessing, system simulation, data analysis, and optimization. It also includes tools for creating 2D and 3D plots, making it useful for visualizing data and results. Users can add more features to Scilab by installing toolboxes that expand its functionality. Because it is free and easy to use, Scilab is often chosen as an alternative to MATLAB in academic settings.

One of the important toolboxes available in Scilab is the Control System Toolbox, which is developed by the FOSSEE team at IIT Bombay. This toolbox is designed to help users study and work with control systems. It allows for modeling, analysis, and simulation of different types of systems. The toolbox has functions to plot root locus, Bode plots, and plot. These plots help in understanding system behavior and frequency response of the system. It also provides tools to analyze poles and zeros, which are important for checking the stability of a system. In addition, the toolbox supports PID controller design and helps users work with transfer functions and state-space models. Properties like controllability and observability can also be checked. Since all the features are built to work directly in Scilab, there is no need for extra software, which makes the toolbox fast and easy to use. It is a helpful tool for students and professionals in fields such as robotics, automation, and industrial control systems.

Chapter 2

Control System Toolbox Development

2.1 Overview

The Control System Toolbox developed as part of the FOSSEE initiative plays a significant role in supporting control engineering applications within Scilab. It serves as a vital resource for learners, educators, and practitioners who rely on free and open-source tools for system modeling and analysis. The toolbox includes a diverse set of functions that cover core aspects of control system design, ranging from foundational classical techniques to more advanced modern methods.

The Control System Toolbox in Scilab originally contained only a limited number of functions. Hence the internship goal was to replicate several control functions from GNU Octave to expand and extend the existing toolbox's capabilities. This work aimed to reduce dependence on external tools and provide a more integrated, Scilab-only solution for users. The motivations for this translation included:

- **Better integration:** Allowed greater alignment with Scilab's internal libraries and syntax, resulting in fewer runtime issues.
- **Enhancement of existing toolbox:** To strengthen and expand the capabilities of the current Control System Toolbox by adding new functions and improving existing ones.

The process of development required a combination of technical understanding, creative problem-solving, and rigorous testing. Each function was carefully redesigned or rewritten to match its intended behavior. Special attention was paid to numerical stability, syntax compatibility, and documentation. The following sections detail the development strategy, function design, and key results achieved during the internship.

2.2 Development Workflow

The development process followed these main steps:

- The functionality of the original Octave functions was examined in detail.
- Their logic was translated into Scilab using equivalent syntax and operations.
- Any missing or unsupported features in Scilab were identified and implemented separately.
- Each function was carefully tested, and improvements were made to ensure accuracy and reliability.

2.2.1 Studying Reference Implementations

Before beginning the implementation, it was essential to analyze each control function in detail. This included understanding its purpose and the mathematical principles involved. For functions derived from Octave, reference materials such as documentation and source code were typically available on Octave Forge.

Important tasks during this stage included:

- Studying the mathematical foundation and expected behavior of the function.
- Identifying dependencies, including sub-functions and specific Octave features.
- Assessing the complexity of translation by comparing Octave constructs with Scilab's capabilities.

2.2.2 Line-by-Line Translation to Scilab

This stage focused on converting the original Octave/MATLAB code into Scilab-compatible syntax and structure. While both environments share several programming concepts, there are important differences that had to be carefully addressed:

- **Control structures:** Octave uses explicit keywords such as endif and endfor to close blocks, whereas Scilab uses a single end for all block closures.
- **Constants and logical values:** Scilab uses predefined constants like
- **String handling and indexing:** The way Scilab manages strings and performs indexing operations often differs, requiring adjustments to maintain expected behavior.
- **Error handling:** Octave's usage() function had to be replaced with Scilab's error("message") function for displaying custom error messages.

In some cases, direct one-to-one translation was not possible, which called for creative reimplementation using Scilab-specific features and logic to ensure functional accuracy.

2.2.3 Handling Missing or Incompatible Functions

During the translation process, several difficulties arose due to missing or incompatible functions in Scilab. To overcome these issues, alternative strategies were employed:

- **Function reimplementation:** When no direct equivalent existed in Scilab, new versions of those functions were created using native Scilab programming constructs.
- **Differences in behavior:** Some functions with the same name in both Octave and Scilab behaved differently. In these cases, the intended functionality was reproduced by writing custom code tailored for Scilab.

In some instances, the original reference code used data types such as structs that Scilab does not fully support. These cases were addressed by either developing algorithmic substitutes or bypassing those parts of the implementation altogether.

2.2.4 Testing and Iteration

Testing played a crucial role throughout the development process. Often, test cases were available at the end of the original source files in the reference implementation. When such tests were missing or inadequate, new test cases were created to verify the accuracy and reliability of the Scilab implementations. Various input scenarios, including edge cases, were examined to achieve thorough validation. Each function was repeatedly refined and tested until its results closely matched those of the original reference.

2.2.5 Leveraging AI for Translation and Debugging

An important part of the workflow was the use of AI (Artificial Intelligence) assistance tools to speed up the translation process. AI proved valuable at several stages of development:

- **Algorithm Extraction:** AI was used to interpret the underlying algorithms from the original Octave code, helping to understand the logical flow before starting the Scilab implementation.
- **Code Translation:** Octave code fragments were converted into Scilab syntax with AI support, ensuring functionality was preserved and suggesting equivalent Scilab functions where Octave features were not directly supported.
- **Code Structuring:** After drafting the Scilab version, AI assisted in improving indentation, formatting, and applying clear naming conventions, which enhanced readability and maintainability.
- **Conceptual Understanding:** AI explanations helped in grasping new technical concepts that were essential for building accurate Scilab functions.
- **Error Analysis and Debugging:** In cases of partial or unexpected outputs, AI analyzed error messages and suggested corrections or alternative approaches, which reduced time spent searching documentation and allowed more focus on the mathematical aspects of each function.

2.3 Current Status

Following the previously outlined workflow, I have successfully translated 25 functions from Octave to Scilab.

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 test cases and documentation in a README file for each function.

2.3.1 Documentation Pattern

Since the goal was to replicate the functions as they exist in Octave, Octave's documentation served as the primary reference for documenting the corresponding Scilab functions.

The documentation for each function is placed on the top of its declaration and consists of a single comprehensive comment block. This documentation typically includes four key parts:

- **Calling Sequence:** The order in which the function evaluates its parameters.
- **Parameters:** Details about the expected inputs, including their types and valid ranges.

- **Description:** An in-depth explanation of the function's behavior, including default settings, input-output expectations, dependencies, and other relevant information.
- **Examples:** Sample usage illustrating how the function should be used correctly.

Because the functions were rewritten to maintain their original behavior, the documentation required minimal changes, as the existing descriptions generally aligned well with the Octave versions.

All the work done during the internship can be found on GitHub:
https://github.com/nikithad14/Scilab-control-system-toolbox-development-functions

2.3.2 Functions Completed

S.No	Function name	Function name in octave	Dependencies
1	append	append	
2	db2mag	db2mag	
3	diff_iddata	@iddata/diff	
4	dssdata	dssdata	_sys_data_
5	horzcat_iddata	@iddata/horzcat	
6	ifft_iddata	@iddata/ifft	
7	inv_ss	inv	_sys_inverse_
8	isminimumphase	isminimumphase	zero
9	issiso	issiso	
10	isstable	isstable	_is_stable_ , pole
11	merge_iddata	@iddata/merge	
12	mldivide	mldivide	
13	mpower	mpower	
14	mrdivide	mrdivide	
15	nkshift_iddata	@iddata/nkshift	
16	parallel	parallel	
17	pole	pole	_pole_
18	repsys	repsys	
19	series	series	
20	size_lti	@lti/size	
21	times	times	_times_
22	transpose	transpose	_transpose_
23	uminus	uminus	
24	uplus	uplus	
25	end_lti	end	

Table 2.1: List of Completed Functions

2.3.3 Challenges Faced

Challenges in Developing Scilab-Native Implementations of Signal Processing Functions During the development of Scilab-native signal processing functions, challenges arose both before and after incorporating AI-based assistance. These can be categorized as follows:

Challenges Before Using AI Assistance

- **Unavailable Functions:** Several sub-functions used in Octave implementations were missing in Scilab, requiring re-implementation with equivalent logic.
- **Different Default Behaviors:** Functions like lyap behaved differently in Octave and Scilab, demanding careful alignment with Octave's expected behavior.
- **Unavailability of SLICOT Library:** Octave's Control System Toolbox relied on the SLICOT library (Fortran 77 routines for systems and control theory). Translating these into Scilab was complex.
- **Complex Data Types:** Octave supports advanced data types such as iddata, which are not natively available in Scilab.
- **Testing Difficulties:** Some functions lacked ready-made test cases, requiring manual creation of test inputs and comparison with Octave outputs for validation.

Challenges After Incorporating AI Assistance

- **Risk of Over-Simplification:** AI-generated translations sometimes ignored edge cases or subtle details present in the original Octave implementations.
- **Debugging AI-Generated Code:** AI outputs occasionally introduced syntactic or semantic errors that required manual debugging in Scilab.
- **Consistency Across Functions:** With AI accelerating development, additional effort was needed to maintain uniform coding style and seamless integration across different functions.
- **Validation Still Essential:** Despite AI support, rigorous testing against Octave remained crucial to ensure correctness, numerical stability, and reliability, particularly for edge cases.

2.3.4 Issues and Possible Improvements

- Documentation can be improved with practical examples.
- Error handling can be made more robust to prevent unexpected crashes.

Chapter 3

Learnings

I have had a lot of valuable learning experiences from this internship opportunity. Some are enumerated below:

1. **Control Systems Knowledge Enhancement:**
   - Gained a deeper understanding of control systems and their real-world applications.
   - Learned how to analyze, design, and simulate control systems effectively.
   - Improved conceptual clarity and problem-solving abilities related to system dynamics and feedback control.
2. **Technical Proficiency in Tools and Platforms:**
   - Learned to code in Scilab and Octave for mathematical modeling and simulations.
   - Applied Scilab and Octave to solve practical control system problems and perform simulations.
   - Gained hands-on experience using Git and GitHub for version control and collaborative project management.
3. **Time Management and Balance:**
   - Learned how to effectively manage time between academic responsibilities and internship tasks.
   - Developed better organizational and prioritization skills.
   - Gained insight into maintaining consistency in both academic and professional commitments.
4. **Professional Communication and Conduct:**
   - Learned how to communicate clearly and respectfully in a professional environment.
   - Understood the importance of writing professional emails and status updates.
   - Gained confidence in interacting with mentors, peers, and stakeholders.

Chapter 4

Conclusion

In conclusion, my internship at FOSSEE, IIT Bombay, working on the Scilab Control System Toolbox, has been a truly enriching and insightful journey.

Over the course of this internship, I've had the opportunity to contribute meaningfully to the open-source community by helping improve both the features and performance of the Control Systems Toolbox.

My main task involved converting control system functions written in Octave into native Scilab code. This development will make the Scilab Control System Toolbox more efficient, easier to maintain, and less reliant on external dependencies.Through careful planning, detailed coding, thorough testing, and continuous refinement, I was able to successfully convert around 19 functions to Scilab and provide clear documentation for each of them.

This internship has been more than just a technical learning experience – it's been a journey of growth, exploration, and collaboration. Being part of a project that contributes to open-source development helped me understand how individual efforts can make a broader impact.

I'm truly grateful to my mentor, Ms. Rashmi Patankar, for her constant support and guidance throughout this internship. Her insights and encouragement made a big difference in my learning. She was always approachable whenever I had doubts, and her timely feedback helped me improve continuously. Her structured approach to problem-solving also taught me the importance of clarity and precision in technical work. I also thank Dr.V Krishnaveni, HoD, Department of ECE and PSG College of Technology for their support and for encouraging students to participate in such meaningful open-source initiatives

Looking back, this experience has not only strengthened my technical skills but also deepened my interest in contributing to open-source tools for science and engineering. I'm excited to carry forward everything I've learned here into whatever comes next.

Reference

- https://gnu-octave.github.io/packages/control/
- https://gnu-octave.github.io/pkg-control/
- https://scilab.in/fossee-scilab-toolbox/control-system-toolbox
- https://github.com/nikithad14/Scilab-control-system-toolbox-development-functions