FOSSEE Summer Fellowship Report

On

Scilab Case Studies

Submitted by

Sanyam Bhavsar

Int. M.Tech Student, Data Science and Computation
VIT Bhopal University

Under the Guidance of

Prof. Prabhu Ramachandran

Department of Chemical Engineering
Indian Institute of Technology Bombay

Mentor:

Rashmi Patankar

July 2026

Acknowledgment

I would like to sincerely thank the FOSSEE Team at IIT Bombay for granting me the valuable opportunity to participate in the FOSSEE Summer Fellowship, 2026. My heartfelt appreciation goes to Prof. Prabhu Ramachandran for initiating this remarkable program that promotes open-source software development. I am especially grateful to Mrs. Rashmi Patankar for her essential guidance and continuous encouragement throughout the internship; her expertise significantly enhanced my learning journey and contributed greatly to the internship's success.

I also extend my sincere gratitude to VIT Bhopal University for providing a strong academic foundation that enriched this experience. Lastly, I am deeply thankful to my parents for their unwavering support, without which this achievement would not have been possible.

Contents

Chapter 1

Introduction

The FOSSEE (Free and Open Source Software in Education) initiative aims to promote the use of open-source software in technical education across India. Its primary goal is to reduce dependency on proprietary software in STEM disciplines by integrating free and open-source alternatives into academic curricula. As part of this effort, FOSSEE offers a summer fellowship program that enables students to explore, contribute to, and apply open-source tools to solve real-world engineering problems.

Scilab is one such powerful open-source platform used extensively for numerical computations. It features a high-level programming language and supports a broad range of applications, including signal and image processing, statistical analysis, numerical optimization, and dynamic system modeling. Its Xcos module allows for graphical modeling and simulation, while its GUI development capabilities enable the creation of interactive, application-specific tools for engineering use.

Case Study 1: Smart Irrigation System with Sensor Fault Detection using Scilab. Development of a Scilab-based smart irrigation system using simulated sensor readings for soil moisture, temperature, and humidity. The system incorporates sensor fault detection using range checking, sudden-jump detection, and residual-based checking with redundant sensors. Faulty readings are corrected before being used for irrigation control.

Case Study 2: Regenerative Braking Control for Electric Vehicles Using PI and Sliding Mode Controllers in Scilab/Xcos. Design and simulation of a regenerative braking system for an electric vehicle using Scilab/Xcos. Two control techniques, PI controller and Sliding Mode Controller, are implemented and compared. The SMC demonstrates better current tracking and improved energy recovery compared to the conventional PI controller.

Case Study 3: PCA-Based Facial Recognition Using Eigenface Algorithm in Scilab GUI. Implementation of the classical Eigenfaces algorithm for face recognition using Principal Component Analysis (PCA). The system is trained on the AT&T ORL Face Database and achieves 95% recognition accuracy. A custom GUI is developed natively in Scilab for training, testing, and recognition.

These studies demonstrate the versatility of Scilab and its toolboxes across different engineering domains, highlighting the importance of FOSS tools in academic research and education.

Chapter 2

Smart Irrigation System with Sensor Fault Detection using Scilab

2.1 Abstract

Water management is a vital aspect of modern agriculture, and smart irrigation systems can help utilize water more efficiently. This project focuses on developing a Scilab-based smart irrigation system that uses sensor readings such as soil moisture, temperature, and humidity to decide whether irrigation should be turned ON or OFF. The main aim is to simulate how automated irrigation can improve decision-making compared to manual monitoring. Along with irrigation control, this project also includes sensor fault detection using range checking, sudden-jump detection, and residual-based checking with redundant sensors. Faulty readings are corrected before they are used for irrigation control. The complete system is modeled in Scilab using simulated sensor data over time, and the results are shown using graphs of corrected soil moisture, residuals, temperature, humidity, and irrigation status.

2.2 Problem Statement

The primary objectives of this case study are:

1. To simulate soil moisture, temperature, and humidity data in Scilab.
2. To create redundant sensor readings for soil moisture and temperature.
3. To inject artificial faults into sensor data.
4. To detect faults using range check, sudden-jump check, and residual-based check.
5. To correct faulty values before irrigation decision-making.
6. To decide irrigation ON/OFF using corrected soil moisture.
7. To show the behavior of the system using Scilab graphs.

2.3 Flowchart

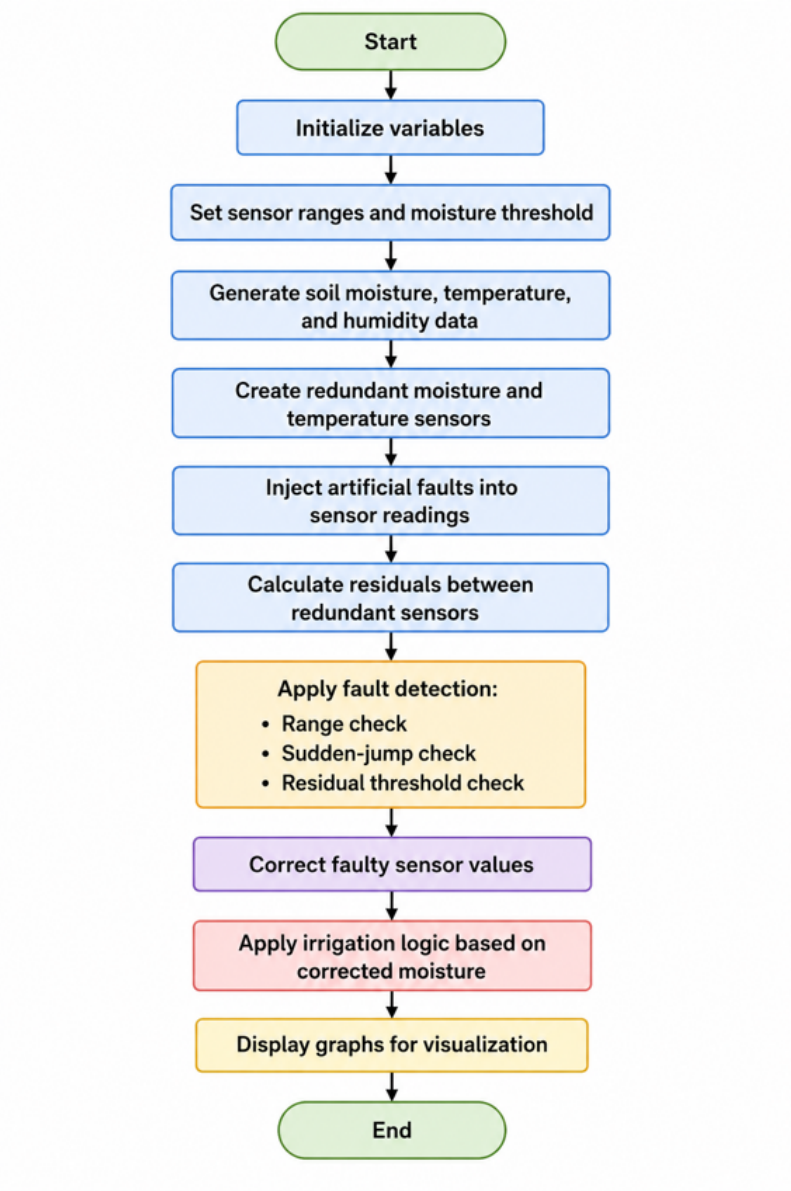

Figure 2.1: Flowchart of the Smart Irrigation System with Sensor Fault Detection

2.4 Results

The simulation generated sensor readings for soil moisture, temperature, and humidity over a fixed time period. Artificial faults were added to test the fault detection logic. The system successfully detected faulty readings using three methods: range checking, sudden-jump checking, and residual-based checking between redundant sensors. Faulty values were corrected before irrigation control, ensuring decisions remained unaffected by incorrect readings.

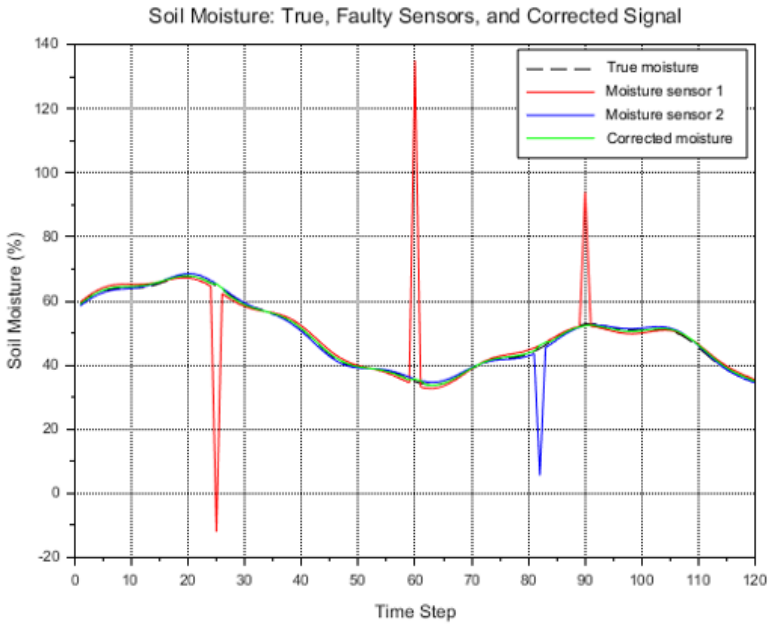

Figure 2.2: Soil moisture readings and corrected soil moisture versus time

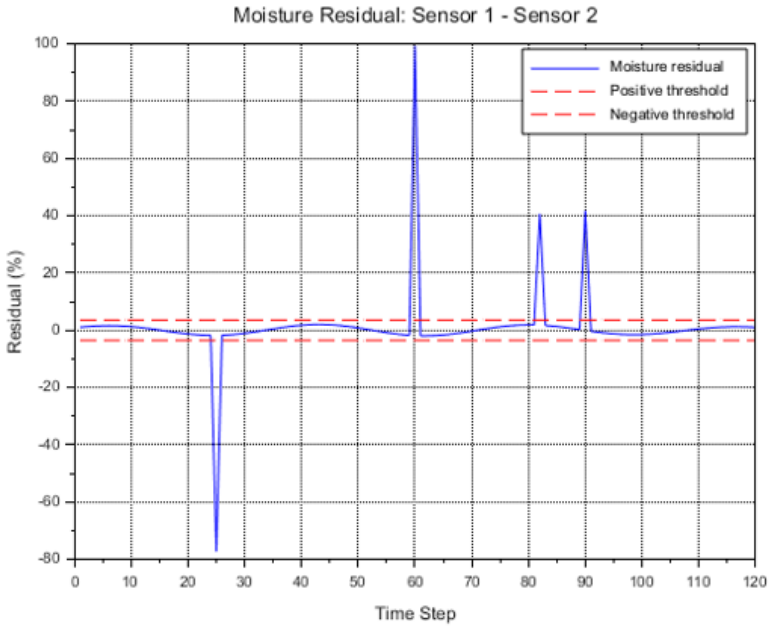

Figure 2.3: Moisture residual versus time

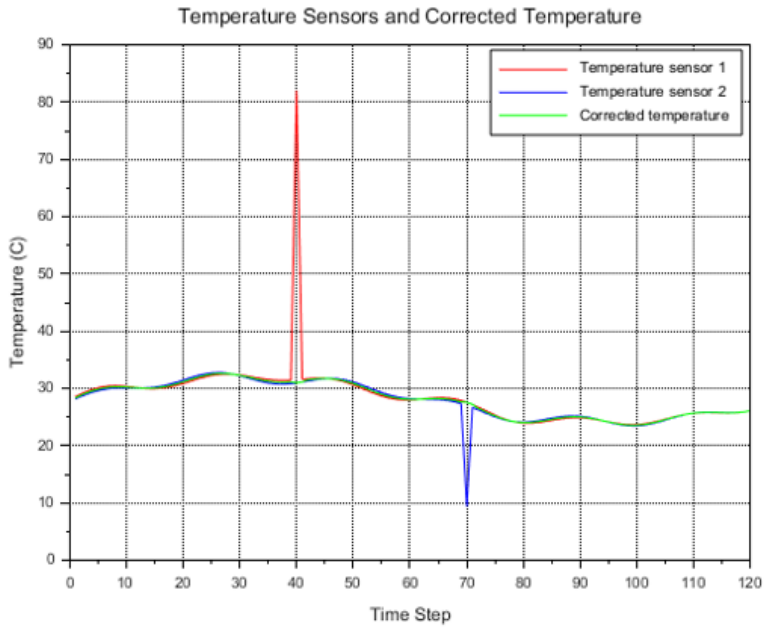

Figure 2.4: Temperature sensor readings and corrected temperature versus time

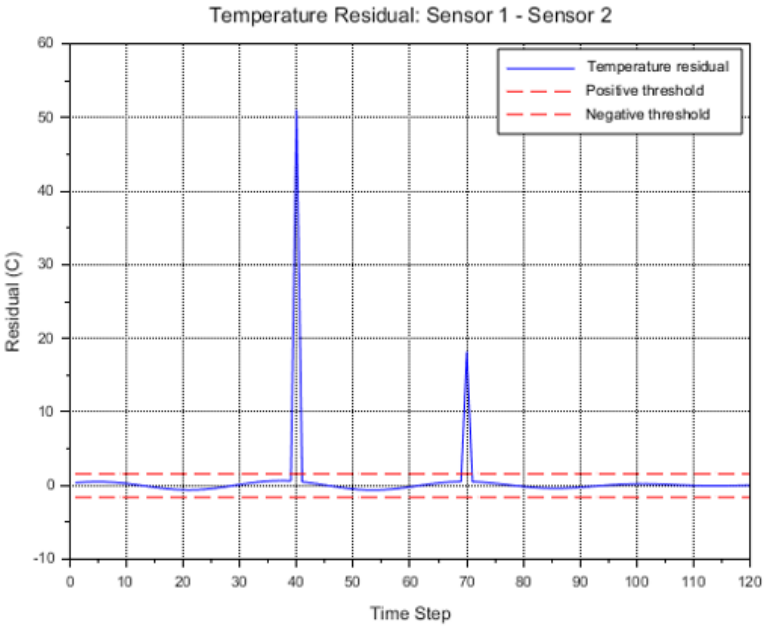

Figure 2.5: Temperature residual versus time

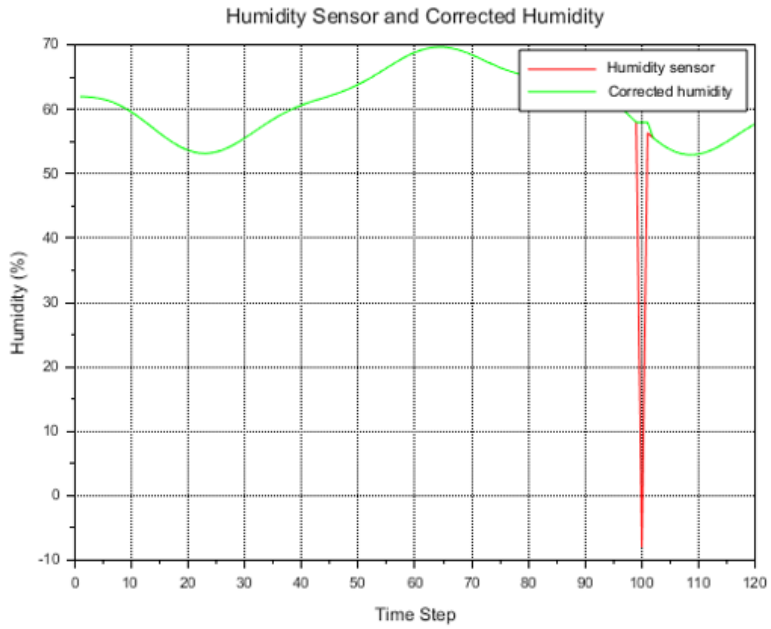

Figure 2.6: Humidity reading and corrected humidity versus time

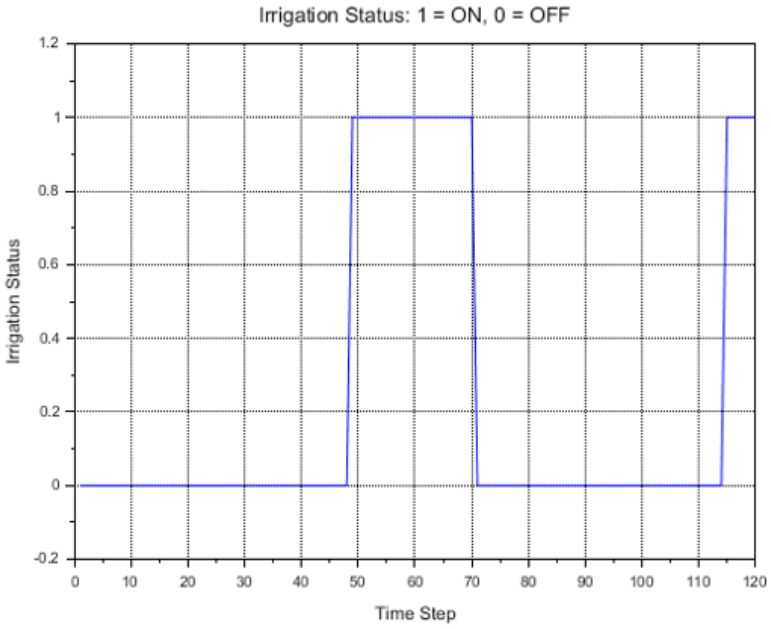

Figure 2.7: Irrigation status versus time (1 = ON, 0 = OFF)

Chapter 3

Regenerative Braking Control for Electric Vehicles Using PI and Sliding Mode Controllers in Scilab/Xcos

3.1 Abstract

Regenerative braking has become an important feature in modern electric vehicles. Unlike conventional braking systems that waste kinetic energy as heat, regenerative braking converts a portion of this energy into electrical energy and stores it back in the battery. This project aims to design and simulate a regenerative braking system for an electric vehicle using Scilab/Xcos and evaluate the performance of different control strategies. The proposed model includes vehicle dynamics, DC motor model, battery charging through regenerative braking, and controller-based duty cycle regulation. Two control techniques, namely the Proportional-Integral (PI) controller and Sliding Mode Controller (SMC), are implemented and compared. The primary objective is to analyze their ability to regulate braking current and maximize energy recovery during vehicle deceleration. Simulation results indicate that both controllers successfully achieve regenerative braking and battery charging. However, the Sliding Mode Controller provides better current tracking and improved energy recovery compared to the conventional PI controller.

3.2 Problem Statement

In conventional braking systems, a large amount of vehicle kinetic energy is dissipated as heat during deceleration. Regenerative braking offers a solution by converting this otherwise wasted energy into electrical energy and storing it in the battery. Regenerative braking reverses the motor's role: during braking, the motor acts as a generator. The back EMF of the spinning motor propels current back through the boost converter and into the battery. The efficiency of energy recovery largely depends on the controller used to regulate the braking current and converter duty

cycle.

The objective of this project is to develop a regenerative braking model and compare PI and Sliding Mode Controllers in terms of regenerative current control, converter duty-cycle regulation, battery charging performance, recovered energy, and vehicle braking response. The goal is to determine whether the Sliding Mode Controller can provide better energy recovery than the conventional PI controller.

3.3 Flowchart and Xcos Models

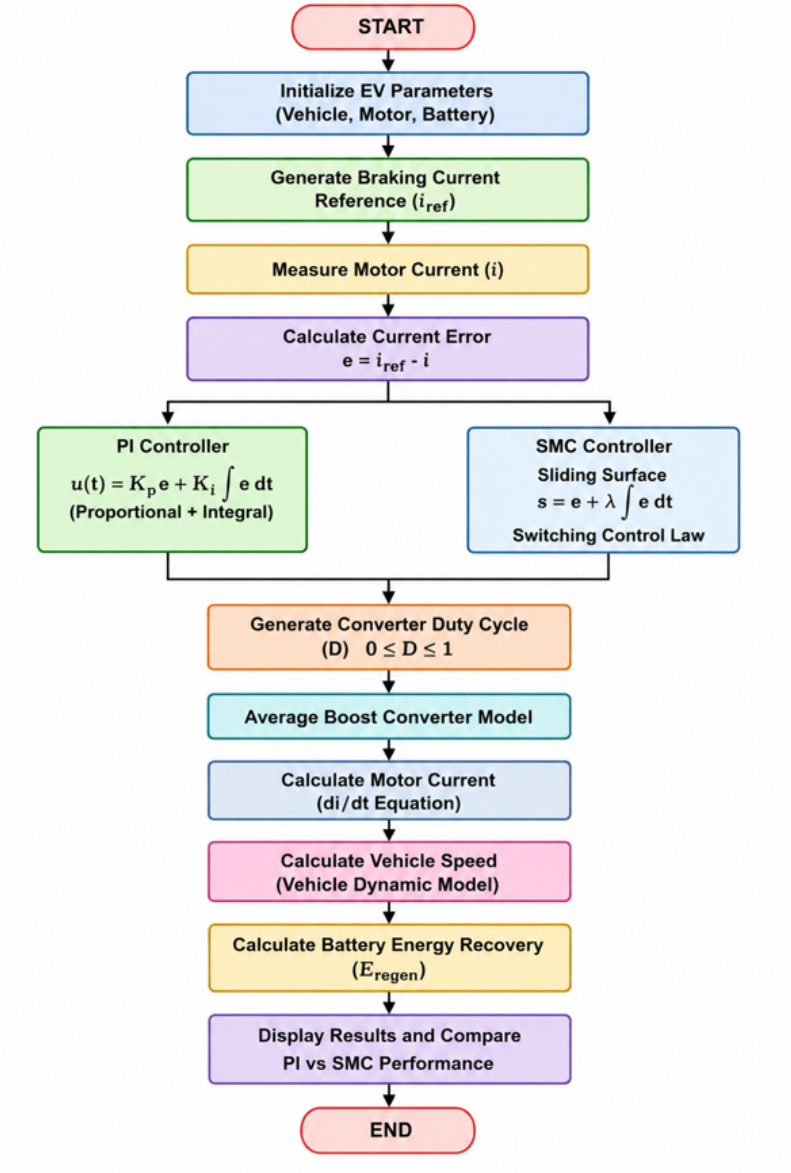

Figure 3.1: Flowchart of the Regenerative Braking Control System

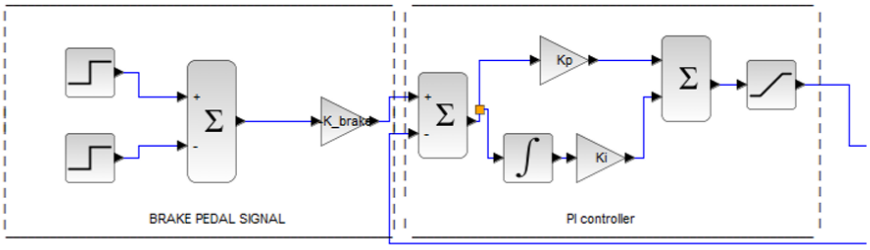

Figure 3.2: PI Controller Model in Xcos

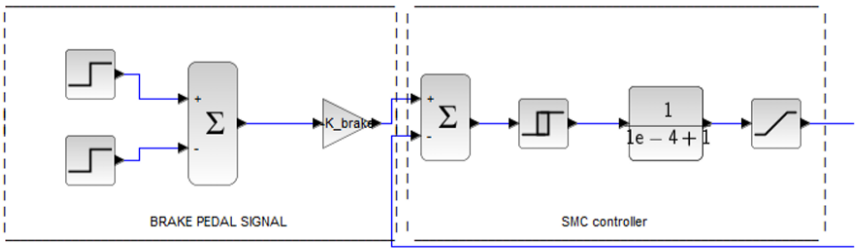

Figure 3.3: Sliding Mode Controller Model in Xcos

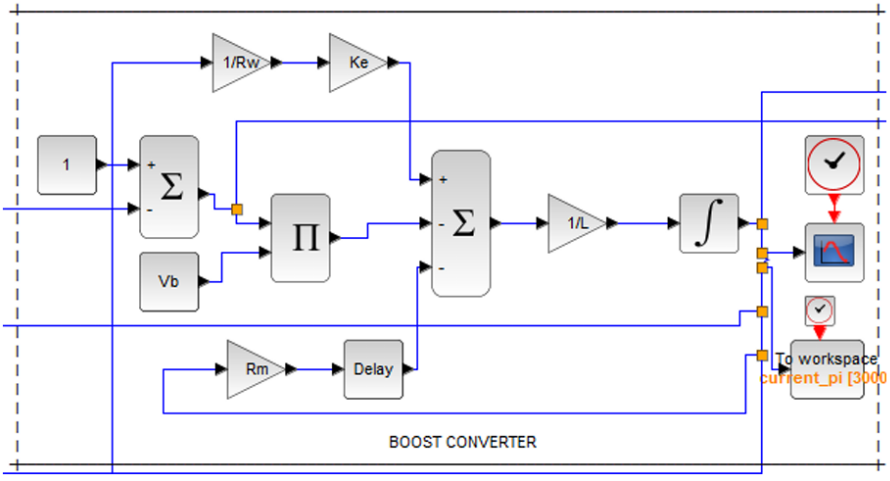

Figure 3.4: Boost Converter Model

3.4 Results

The regenerative braking system was successfully implemented and simulated for both PI and Sliding Mode Controllers using Xcos. The simulation was executed for 30 seconds with a single braking event between t = 5s and t = 15s.

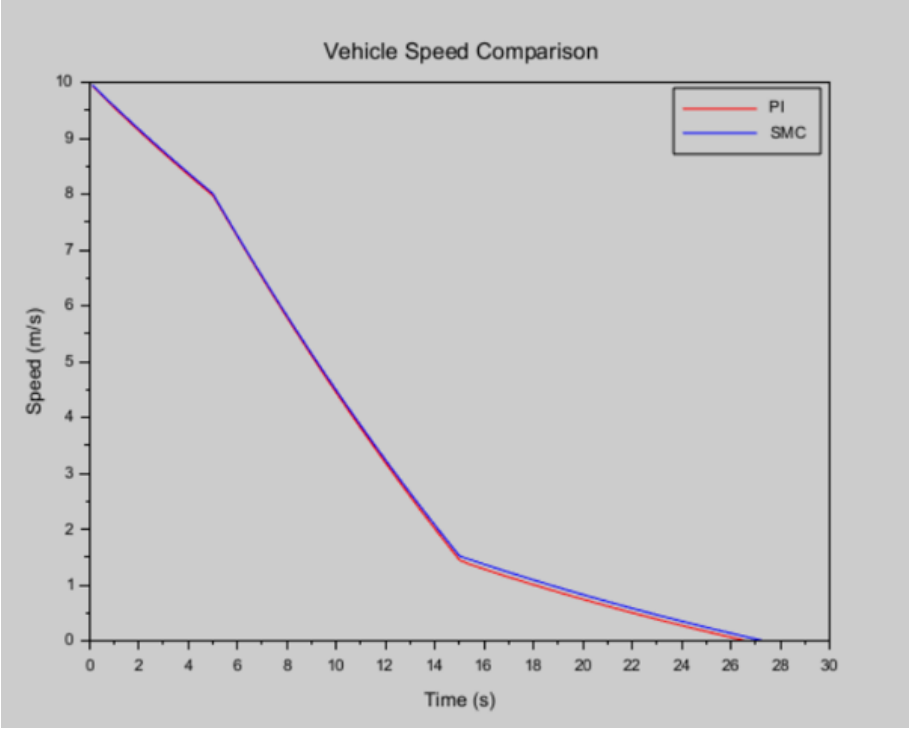

Figure 3.5: Vehicle Speed Comparison - PI (red) vs SMC (blue)

The SMC provides smoother speed regulation with reduced oscillations, indicating better current tracking and more efficient energy conversion.

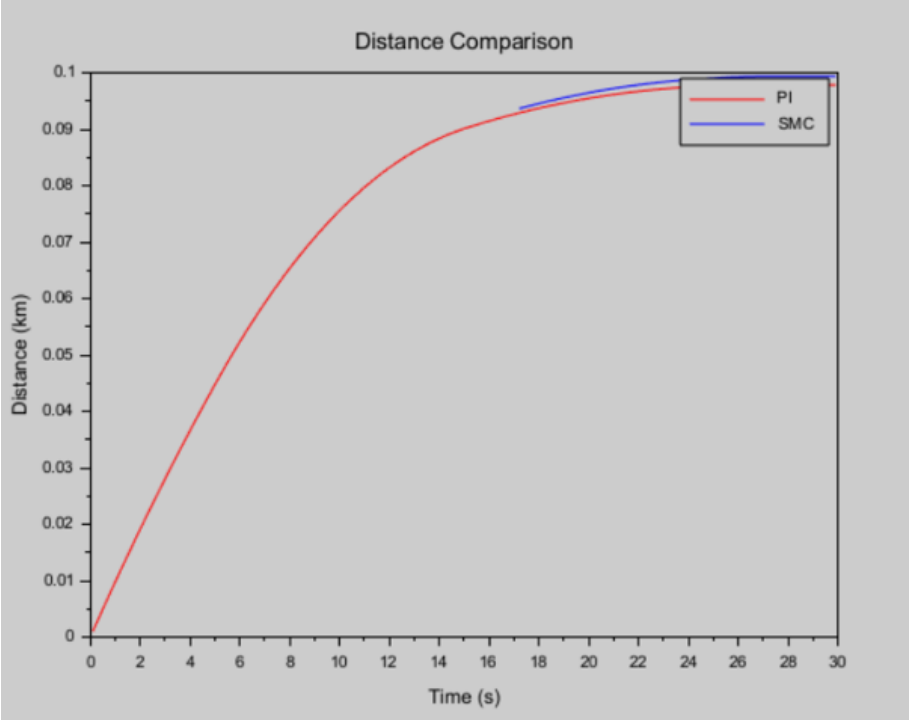

Figure 3.6: Distance Comparison - PI (red) vs SMC (blue)

The SMC achieved approximately 1.56% more distance compared to the PI con-

troller.

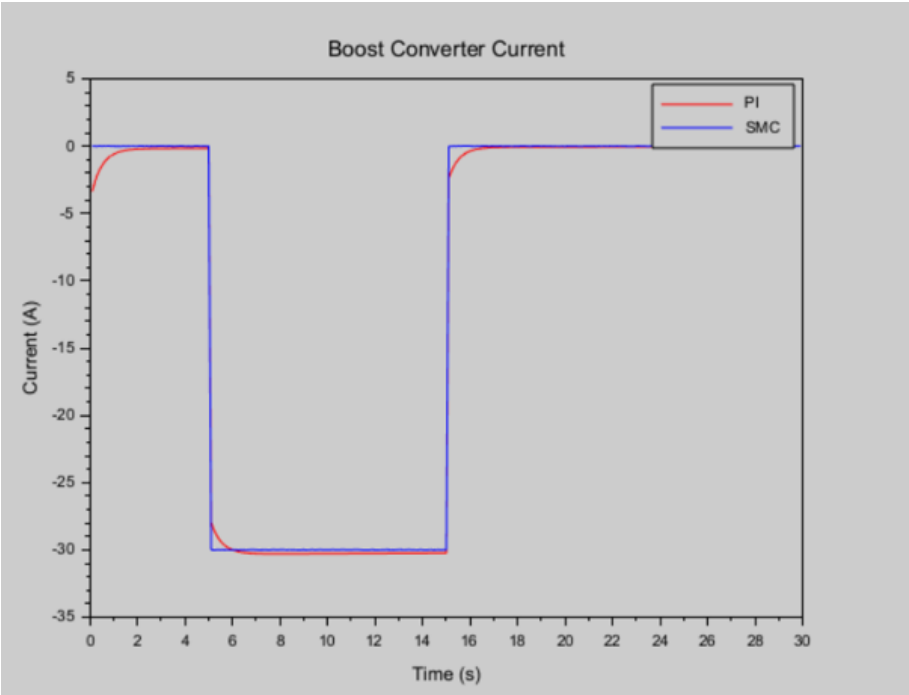

Figure 3.7: Boost Converter Current - PI (red) vs SMC (blue)

The PI controller responds with 7% undershoot at brake start and 8% overshoot at brake end. The SMC reaches target current in 40ms with less than 2% overshoot and ripple of only ±0.02A.

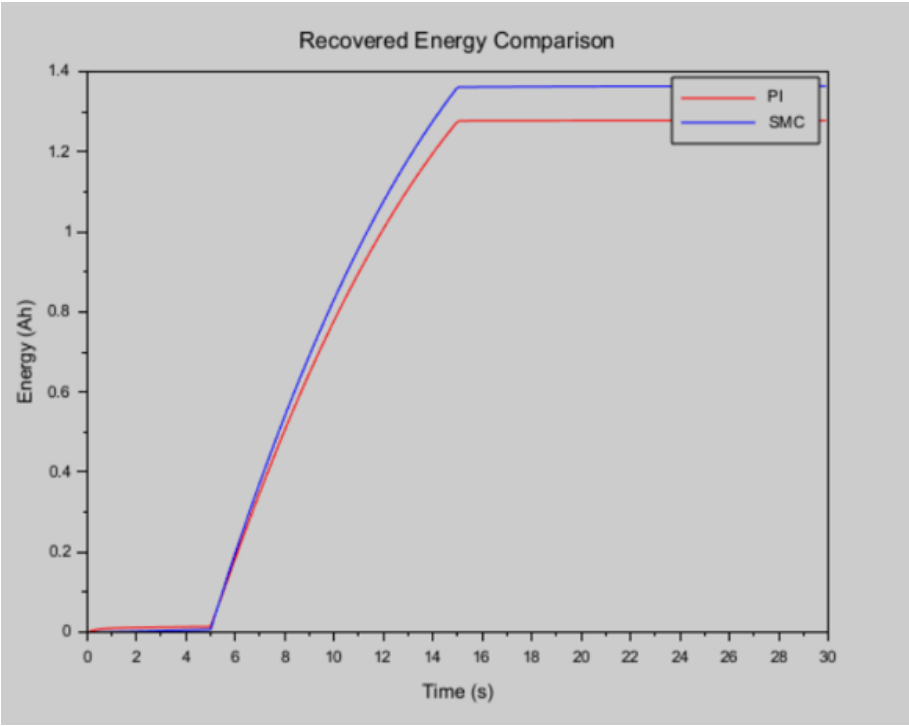

Figure 3.8: Recovered Energy Comparison - PI (red) vs SMC (blue)

The SMC recovers 6.76% more energy per braking event compared to the PI controller.

Parameter	**PI**	**SMC**	**Improvement**
Total Distance (km)	0.0978	0.0993	+1.56%
Recovered Energy (Wh)	1.2775	1.3639	+6.76%
Current Settling Time	1.5 s	0.04 s	SMC faster
Current Undershoot	7%	¡2%	SMC better
Current Ripple	±0.8 A	±0.02 A	SMC better

Chapter 4

PCA-Based Facial Recognition Using Eigenface Algorithm in Scilab GUI

4.1 Abstract

Facial recognition has become one of the key biometric technologies for security and access control. Modern research focuses on deep learning algorithms; however, classical statistical methods are much more simple and efficient. This case study implements the Eigenfaces algorithm for recognition using Principal Component Analysis (PCA). The algorithm was implemented inside the Scilab environment, where the Graphical User Interface (GUI) was designed for training the model, testing with own images, and viewing the results. The AT&T ORL Face Database was used; it contains 400 grayscale images of 40 persons. The first 8 pictures of every person were used for training (320 images), and the other 2 were used for testing. Every uploaded image was projected on the first 50 Eigenfaces to reduce its dimensionality from 10304 pixels to 50 weights. Using these weights, the Euclidean distance was calculated between the test image and every other image from the database.

4.2 Problem Statement

The image from the AT&T ORL database has $112 \times 92 = 10304$ pixels, where every pixel's brightness is encoded as a grayscale value from 0 to 255. For recognition, such images need to be compared to each other pixel by pixel, which requires too many computations and too much memory. Additionally, images may have non-uniform background lighting, people may have different facial expressions or hairstyles, and image quality may vary for various reasons – all of these factors could make recognition inaccurate and unreliable.

Therefore, a recognition system needs to be able to identify a person in an image quickly and efficiently. Instead of comparing every pixel of two images, it would be much easier and more efficient to extract a minimal feature vector that

encodes a person's facial appearance in all images. Principal Component Analysis is a statistical method that finds the most varying directions in a dataset, then projects data onto these directions, thereby reducing dimensionality while retaining most of the information. These directions are then called Eigenfaces.

For the implementation, the Eigenfaces algorithm proposed by Turk and Pentland was used: each training image was converted to a vector, the mean image was computed, and a covariance matrix of the dataset was constructed. Then, the covariance matrix was used to find Eigenfaces and project each training image onto them, thus obtaining a weight vector for every training image. For recognition, a similar feature vector was extracted from the test image and compared to the database using the Euclidean distance between weight vectors.

A GUI was also designed in Scilab to make the application more user-friendly and allow users to train the model, test it on their images, reconstruct faces from weights, and measure overall recognition performance.

4.3 Flowchart

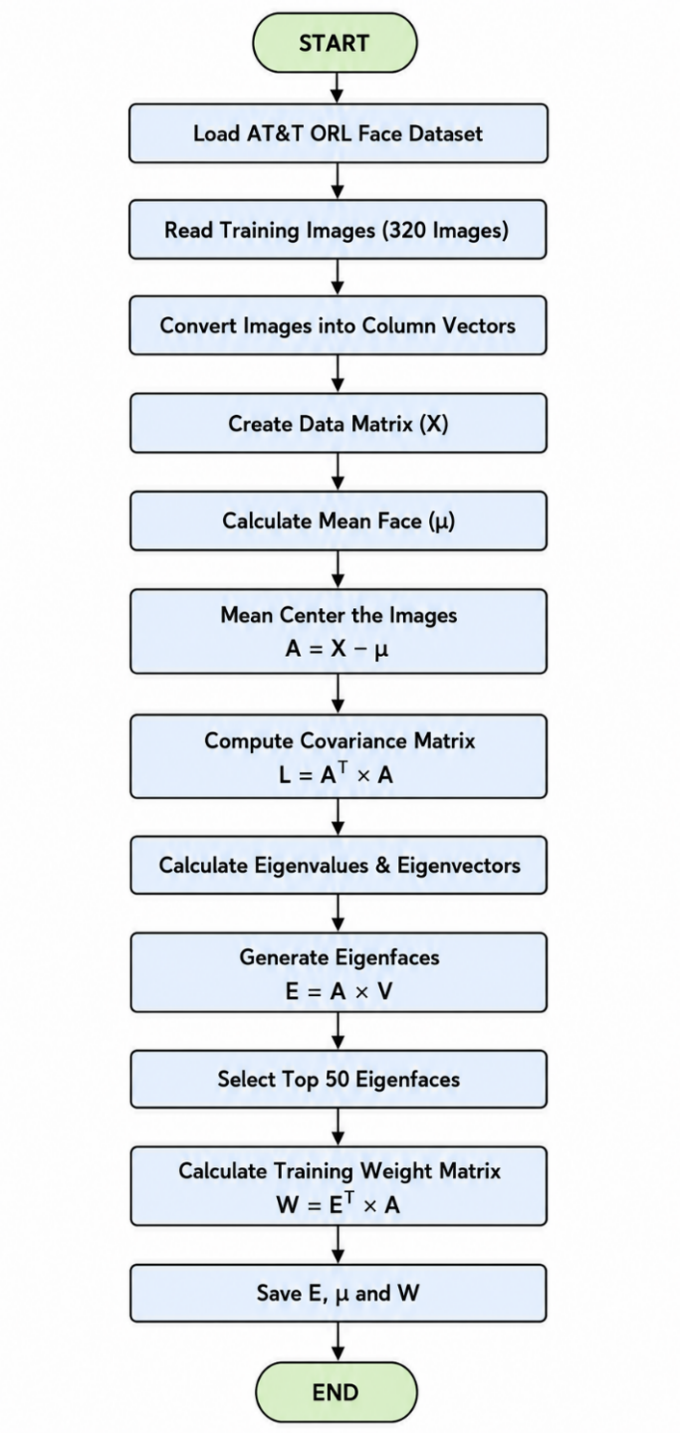

Figure 4.1: Training Phase Flowchart

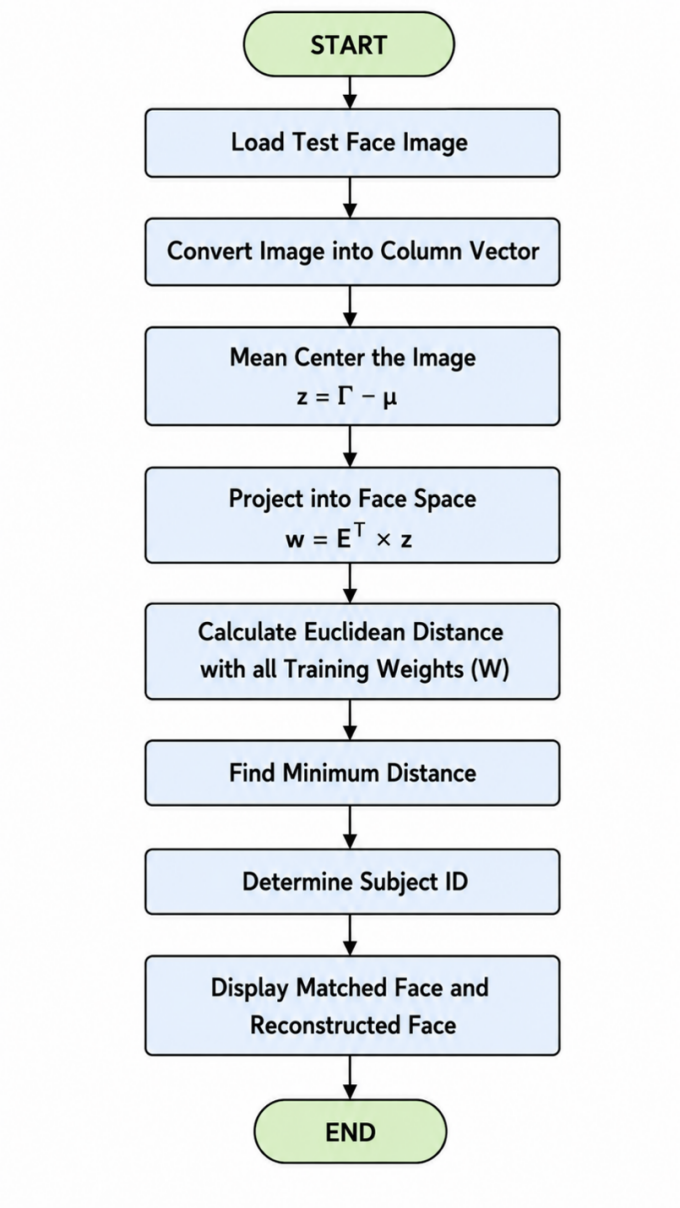

Figure 4.2: Recognition Phase Flowchart

4.4 Results

The facial recognition system based on Eigenfaces was successfully implemented and tested in the Scilab environment. The system went through training, recognition, reconstruction, and GUI testing.

Figure 4.3: Main GUI of the Eigenfaces Face Recognition System

The GUI provides buttons for training, recognition, reconstruction, and accuracy evaluation.

Figure 4.4: Mean Face and Eigenfaces from Training

The Mean Face is a blurred average of all training images. Eigenfaces encode transformations present in faces.

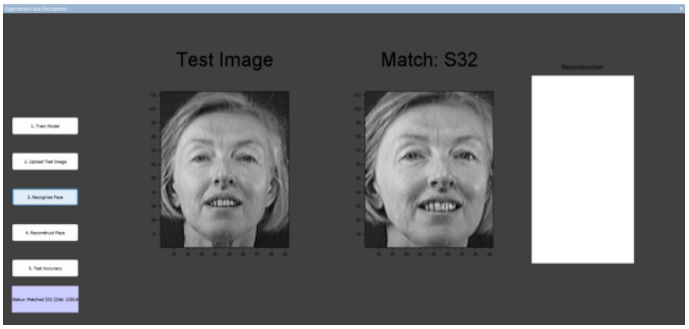

Figure 4.5: Recognition Result - Uploaded Test Image and Matched Face

The system successfully identified the correct subject from the ORL database.

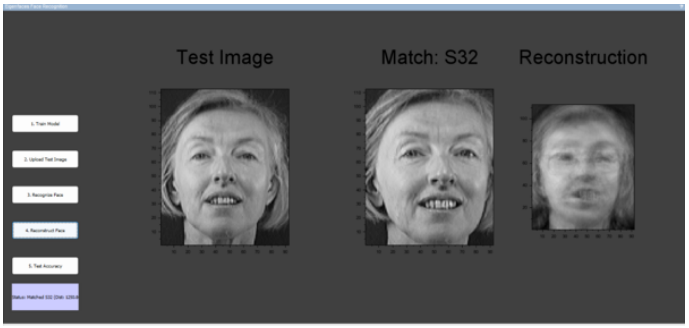

Figure 4.6: Reconstructed Face from Eigenface Coefficients

The reconstructed image, though slightly blurred, retains the overall facial structure.

The recognition accuracy was evaluated using 80 test images (40 people $\times$ 2 images). The system achieved 95% recognition accuracy.

Parameter	**Result**
Training Images	320
Test Images	80
Number of Subjects	40
Selected Eigenfaces	50
Recognition Method	Euclidean Distance
Recognition Accuracy	95%

Chapter 5

Key Learnings

The FOSSEE Summer Fellowship provided valuable hands-on experience in applying Scilab to diverse engineering problems. The key learnings from this experience are:

1. **Scilab and Xcos Proficiency:** Gained strong command over Scilab's numerical computation capabilities and Xcos graphical modeling for dynamic system simulation.
2. **Control System Design:** Practical experience in designing PI and Sliding Mode Controllers for regenerative braking systems, and analyzing their comparative performance.
3. **Sensor Fault Detection:** Implemented range checking, sudden-jump detection, and residual-based fault detection techniques for smart irrigation systems.
4. **Dimensionality Reduction:** Deepened understanding of Principal Component Analysis (PCA) through implementation of the Eigenfaces algorithm for facial recognition.
5. **GUI Development:** Learned to design interactive Graphical User Interfaces natively in Scilab.
6. **Technical Documentation:** Improved academic writing and LaTeX skills through report preparation.
7. **Open-Source Philosophy:** Reinforced the importance of Free and Open Source Software in making quality tools accessible for education and research.
8. **Problem-Solving and Self-Learning:** Enhanced ability to independently understand research papers, adapt concepts, and troubleshoot implementation issues in Scilab.

Overall, this fellowship has been a transformative experience, bridging theoretical knowledge with practical application and preparing me for future research and development work.

References

1. Automatic Irrigation System using Soil Moisture Sensor. ResearchGate Link
2. Sensor fault detection and isolation for smart irrigation using parity space approach. PDF Link
3. Energy-Regenerative Braking Control of Electric Vehicles Using Three-Phase BLDC Motors, Research Paper, 2014. MDPI Link
4. Matthew Turk and Alex Pentland. Eigenfaces for Recognition. PDF Link
5. Scilab - An Open Source Software. https://www.scilab.in/
6. FOSSEE Project, IIT Bombay. https://fossee.in/