

Project Title: System Verification and  
Validation Plan for  
Helmholtz-Coil-Current-Calculator-CAS741

Reyhaneh Norouziani

February 19, 2024

## Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

# Contents

<b>1</b>	<b>Symbols, Abbreviations, and Acronyms</b>	<b>iv</b>
<b>2</b>	<b>General Information</b>	<b>1</b>
2.1	Summary . . . . .	1
2.2	Objectives . . . . .	1
2.3	Relevant Documentation . . . . .	1
<b>3</b>	<b>Plan</b>	<b>1</b>
3.1	Verification and Validation Team . . . . .	1
3.2	SRS Verification Plan . . . . .	2
3.3	Design Verification Plan . . . . .	3
3.4	Verification and Validation Plan Verification Plan . . . . .	3
3.5	Implementation Verification Plan . . . . .	4
3.6	Automated Testing and Verification Tools . . . . .	4
3.7	Software Validation Plan . . . . .	5
<b>4</b>	<b>System Test Description</b>	<b>5</b>
4.1	Tests for Functional Requirements . . . . .	5
4.1.1	Inputs . . . . .	5
4.1.2	Outputs . . . . .	6
4.2	Tests for Nonfunctional Requirements . . . . .	7
4.2.1	Accuracy . . . . .	7
4.2.2	Usability . . . . .	7
4.2.3	Maintainability . . . . .	7
4.2.4	Portability . . . . .	8
4.3	Traceability Between Test Cases and Requirements . . . . .	8
<b>5</b>	<b>Unit Test Description</b>	<b>8</b>
5.1	Unit Testing Scope . . . . .	9
5.2	Tests for Functional Requirements . . . . .	9
5.2.1	Module 1 . . . . .	9
5.2.2	Module 2 . . . . .	10
5.3	Tests for Nonfunctional Requirements . . . . .	10
5.3.1	Module ? . . . . .	10
5.3.2	Module ? . . . . .	11
5.4	Traceability Between Test Cases and Modules . . . . .	11

<b>6</b>	<b>Appendix</b>	<b>12</b>
6.1	Usability Test Survey Questions? . . . . .	12

## List of Tables

1	Verification and Validation Team . . . . .	2
2	Relationship Between Test Cases and Requirements . . . . .	8

## List of Figures

# 1 Symbols, Abbreviations, and Acronyms

Symbol/Abbreviation/Acronym	Description
VnV	Verification and Validation
SRS	System Requirements Specification
MG	Module Guide
MIS	Module Interface Specification

## 2 General Information

### 2.1 Summary

This document outlines the validation and verification (VnV) plan for the Helmholtz-Coil-Current-Calculator-CAS741, a specialized software tool for calculating the precise currents required in each coil of a three-axis Helmholtz coil system to achieve specific target magnetic force and torque. The VnV plan will ensure the software’s functionality, accuracy, and reliability in facilitating the precise control of magnetic environments for a wide range of applications.

### 2.2 Objectives

The objectives within this Verification and Validation (VnV) plan are centered on ensuring the software’s proficiency in precisely calculating coil currents necessary for achieving specified magnetic force and torque within a three-axis Helmholtz coil system. Additionally, it aims to confirm the clarity and user-friendliness of the software’s visual outputs, making certain they are easily understandable and provide valuable assistance for users’ research endeavors. This integrated approach emphasizes both accuracy in computational outputs and accessibility in data presentation, underpinning the software’s utility in scientific research.

### 2.3 Relevant Documentation

For a detailed overview of the Helmholtz-Coil-Current-Calculator-CAS741’s objectives, functional requirements, and operational guidelines, please refer to the [System Requirements Specification \(SRS\)](#) document.

## 3 Plan

### 3.1 Verification and Validation Team

This section describes the members of the Verification and Validation plan.

<b>Name</b>	<b>Document</b>	<b>Role</b>	<b>Description</b>
Dr. Spencer Smith	All	Instructor/Reviewer	Review the documents, design, and documentation style.
Reyhaneh Norouziani	All	Author	Create and manage all the documents, create the VnV plan, perform the VnV testing, verify the implementation.
Tanya Djavaherpour	All	Domain Expert Reviewer	Review all the documents.
Yi-Leng Chen	SRS	Secondary Reviewer	Review the SRS document.
Yidng Li	VnV Plan	Secondary Reviewer	Review the VnV plan.
Atiyeh Sayadi	MG + MIS	Secondary Reviewer	Review the MG and MIS document.

Table 1: Verification and Validation Team

### 3.2 SRS Verification Plan

The verification process for the Helmholtz-Coil-Current-Calculator-CAS741's SRS document will proceed as follows:

1. An initial review will be conducted by assigned members (Dr. Spencer Smith, Tanya Djavaherpour, Yi-Leng Chen) using the provided [SRS Checklist](#), developed by Dr. Smith. This review will be carried out manually.
2. Reviewers can provide feedback to the author by creating issues on Github.

3. The author, Reyhaneh Norouziani, is responsible for addressing the issues raised by both primary and secondary reviewers, as well as incorporating any suggestions provided by the instructor, Dr. Spencer Smith.

### 3.3 Design Verification Plan

The verification process for the Helmholtz-Coil-Current-Calculator-CAS741's Module Guide (MG), and Module Interface Specification (MIS) will proceed as follows:

1. An initial review will be conducted by assigned members (Dr. Spencer Smith, Tanya Djavaherpour, Atiyeh Sayadi) using the provided [MG Checklist](#) and [MIS Checklist](#), developed by Dr. Smith. This review will be carried out manually.
2. Reviewers can provide feedback to the author by creating issues on Github.
3. The author, Reyhaneh Norouziani, is responsible for addressing the issues raised by both primary and secondary reviewers, as well as incorporating any suggestions provided by the instructor, Dr. Spencer Smith.

### 3.4 Verification and Validation Plan Verification Plan

The verification process for the Helmholtz-Coil-Current-Calculator-CAS741's Verification and Validation Plan will proceed as follows:

1. An initial review will be conducted by assigned members (Dr. Spencer Smith, Tanya Djavaherpour, Yiding Li) using the provided [VnV Checklist](#), developed by Dr. Smith. This review will be carried out manually.
2. Reviewers can provide feedback to the author by creating issues on Github.
3. The author, Reyhaneh Norouziani, is responsible for addressing the issues raised by both primary and secondary reviewers, as well as incorporating any suggestions provided by the instructor, Dr. Spencer Smith.



### 3.5 Implementation Verification Plan

The Implementation process will consist of both static and dynamic testing methodologies.

**Static testing:**

- Static testing:
  - Code Walkthrough: The author (Reyhaneh Norouziani) will provide the final version of the original code to the domain expert (Tanya Djavaherpour). The domain expert will then manually examine critical test cases, share her findings, and actively participate in discussions. Additionally, she can seek clarification on any aspects of the code as needed.
  - PyFlakes: PyFlakes, a static code analysis tool, will be utilized to identify common programming errors and potential issues in the Helmholtz-Coil-Current-Calculator-CAS741 codebase. This automated process will help ensure code quality and identify any inconsistencies or errors that may have been overlooked during manual code review by the author.
- Dynamic testing:
  - Test cases: Test cases for all the tests mentioned in [subsection 3.7](#) will be executed. These tests aim to validate both functional and non-functional requirements outlined in the [SRS](#) document (Sections 5.1 and 5.2).

### 3.6 Automated Testing and Verification Tools

The verification and validation process for the Three-axis Helmholtz Coil System Current Calculator (Helmholtz-Coil-Current-Calculator-CAS741) involves automated testing using the pytest library in Python. One of the automated test cases focuses on verifying the accuracy of the calculated force and torque generated by the input currents. This test ensures that the software performs within acceptable tolerances, as specified by the system requirements. By utilizing pytest for automated testing, we aim to streamline the verification process and maintain the reliability and accuracy of the software across various scenarios. Additionally, pytest allows for the implementation

of test cases that validate input data against predefined constraints and verify that output results adhere to expected criteria.

### 3.7 Software Validation Plan

For validation purposes, the project will be demonstrated to Dr. Onaizah, whose research involves the intersection of magnetics and robotics for biomedical applications, including the development of small-scale robotic tools. Given her expertise in working with three-axis Helmholtz Coil Systems, her feedback will be invaluable in assessing the suitability of the program to meet the needs of its intended users, including scientists, engineers, and clinicians working in the field of medical robotics and magnetic applications. Any feedback received from this validation activity will be used to refine and enhance the functionality of the Three-axis Helmholtz Coil System Current Calculator further.

## 4 System Test Description

### 4.1 Tests for Functional Requirements

#### 4.1.1 Inputs

This section ensures that the input and output limitations specified in requirement R1 for Helmholtz-Coil-Current-Calculator-CAS741 are verified. These constraints can be found in the "Input Data Constraints" section of the [SRS](#) document.

1. **T1:** Test input for meeting the constraints

Control: Automatic

Initial State: Pending input

Input: The inputs mentioned in [SRS](#) as Table 1: Input Variables

Output: If the inputs meet the constraints, the program will display "Valid"; otherwise, it will present an error message indicating the incorrect input along with the reason for the error.

Test Case Derivation: The expected output is justified based on the predefined input constraints specified in the System Requirements Specification [SRS](#) document. These constraints ensure that the input data adhere to the predefined criteria necessary for accurate calculations within the Three-axis Helmholtz Coil System Current Calculator (Helmholtz-Coil-Current-Calculator-CAS741).

How test will be performed: The automatic test is performed using pytest.

#### 4.1.2 Outputs

1. **T2:** Test Output currents to not be greater than maximum acceptable electric current of each coil

Control: Automatic

Initial State: The output is calculated

Input: Calculated currents of each coils

Output: If the inputs do not meet the constraints, the program will display an error message indicating that the needed current is out of range.

Test Case Derivation: The expected output is justified based on the predefined constraints specified in the System Requirements Specification [SRS](#) document. These constraints define the maximum acceptable electric current for each coil within the Three-axis Helmholtz Coil System.

How test will be performed: The automatic test is performed using pytest.

2. **T3:** Test calculated currents Output are produce the target magnetic force and torque

Control: Automatic

Initial State: The output is calculated

Input: Calculated currents of each coils

Output: It calculates the generating force and torque from the input currents, and if they are within  $\pm 0.5\%$  of the target magnetic force and torque, it passes.

Test Case Derivation: The expected outcome is justified by the requirement outlined in the System Requirements Specification (SRS) document, which specifies that the calculated currents should produce a magnetic force and torque with a specific accuracy.

How test will be performed: The automatic test is performed using pytest.

## 4.2 Tests for Nonfunctional Requirements

### 4.2.1 Accuracy

**T4:** This matter is discussed in detail in Section [T3](#).

### 4.2.2 Usability

**T5:** The usability test will involve a group of users who will interact with the software to complete a specific task. Participants will be tasked with inputting the characteristics of a three-axis Helmholtz coil system, along with a provided target magnetic force and torque, and reading the calculated results. Following the completion of these tasks, participants will provide feedback on the software's user interface design and overall usability by completing a survey(see [6.1](#)).

### 4.2.3 Maintainability

**T6:** The Maintainability test for this project occurs through GitHub issue discussions, where reviewers and the instructor evaluate various aspects related to the software's lifespan. These discussions serve as a platform to assess the coupling of the software architecture, the quality of documentation, and other factors impacting maintainability. Through collaborative review and discussion in GitHub issues, reviewers and the instructor evaluate the code-base's readability, modularity, and adherence to coding standards, identifying areas for improvement to ensure the software remains easily maintainable and adaptable in the long term. This proactive approach to maintainability assessment through GitHub issues enables us to address potential issues early in the development process, fostering a more sustainable and efficient software lifecycle.

#### 4.2.4 Portability

**T7:** The Portability test for the project involves testing the program on various operating systems to ensure its compatibility and functionality across different platforms. The author will execute all test cases on a range of operating systems, including Windows and Linux distributions. Each test will verify that the program operates correctly and that all test cases are successfully executed without errors or discrepancies. Any issues encountered during testing will be documented and addressed to ensure that the software maintains consistent behavior across different platforms.

### 4.3 Traceability Between Test Cases and Requirements

	R1	R2	NFR1	NFR2	NFR3	NFR4
T1	x					
T2		x				
T3		x				
T4			x			
T5				x		
T6					x	
T7						x

Table 2: Relationship Between Test Cases and Requirements

## 5 Unit Test Description

[This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS]

[To save space and time, it may be an option to provide less detail in this section. For the unit tests you can potentially layout your testing strategy here. That is, you can explain how tests will be selected for each module. For instance, your test building approach could be test cases for each access program, including one test for normal behaviour and as many tests as needed

for edge cases. Rather than create the details of the input and output here, you could point to the unit testing code. For this to work, your code needs to be well-documented, with meaningful names for all of the tests. —SS]

## 5.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

## 5.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

### 5.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

#### 1. test-id1

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

2. test-id2

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

3. ...

## 5.2.2 Module 2

...

## 5.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

### 5.3.1 Module ?

1. test-id1

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

Type: Functional, Dynamic, Manual, Static etc.

Initial State:

Input:

Output:

How test will be performed:

### 5.3.2 Module ?

...

## 5.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

## References



## 6 Appendix

### 6.1 Usability Test Survey Questions?

1. On a scale of 1 to 5, how would you rate the overall ease of use of the software?
  - (a) Very difficult
  - (b) Difficult
  - (c) Neutral
  - (d) Easy
  - (e) Very easy
2. Were you able to input the characteristics of the three-axis Helmholtz coil system and the target magnetic force and torque without difficulty?
  - (a) Yes
  - (b) No (Please provide details)
3. How clear and understandable were the calculated results presented by the software?
  - (a) Very clear
  - (b) Clear
  - (c) Neutral
  - (d) Unclear
  - (e) Very unclear
4. Did you encounter any issues or challenges while using the software? If yes, please describe them briefly.
5. Do you have any additional comments or suggestions for improving the usability of the software?

## **Appendix — Reflection**

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

## Appendix — Reflection

[This section is not required for CAS 741 —SS]

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage etc. You should look to identify at least one item for each team member.
2. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?