Project Title: System Verification and Validation Plan for 3D-H3C

Reyhaneh Norouziani

April 16, 2024

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

Contents

List of Tables

List of Figures

1 Symbols, Abbreviations, and Acronyms

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

2 General Information

2.1 Summary

This document outlines the validation and verification (VnV) plan for the 3D-H3C, 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.

Name	Document	Role	Description
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 3D-H3C'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 3D-H3C'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 3D-H3C'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 3D-H3C 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 (3D-H3C) 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 3D-H3C 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 (3D-H3C).

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 codebase'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

5.1 Unit Testing Scope

The scope of the unit tests will focus on the functionality of each module within the software, with an exception made for the OutputFormat module. The OutputFormat module is excluded from unit testing as it contains only a single, simple method responsible for displaying two lines of output, which does not warrant the setup of formal unit tests. Testing for this module will be conducted manually to ensure the output is formatted and displayed correctly. The rest of the unit tests will exclude testing external library functionalities.

5.2 Tests for Functional Requirements

5.2.1 CoilT Module

This module tests the initialization and parameter validation of the CoilT class, which represents individual coils in the system. The tests are implemented in TestCoilT.py and are designed to ensure that coil objects are correctly instantiated and that their parameters are validated according to the specified rules. Each test is named after the specific condition it checks, making it straightforward to understand what each test validates.

UT1: **Load Coils Valid Data**

UT2: **Load Coils File Not Found**

UT3: **Load Coils Invalid JSON**

UT4: **Create Coil from Data Valid**

5.2.2 InputFormat Module

This module tests the functionality of the InputFormat class, which is responsible for loading and validating coil configuration data from JSON files. The tests are implemented in TestInputFormat.py and are designed to ensure that the class accurately handles various data input scenarios. Each test is named to reflect the specific functionality it assesses, facilitating clear and understandable test coverage.

UT5: **Valid Initialization**

UT6: **Invalid Radius**

UT7: **Invalid Distance**

UT8: **Invalid Turns**

UT9: **Invalid Current**

UT10: **Load Params Valid Input**

UT11: **Load Params Invalid Magnetic Moment**

UT12: **Load Params Invalid Magnetic Torque**

UT13: **Load Params Invalid Magnetic Force**

UT14: **Load Params Invalid Combination**

5.2.3 MagneticCoreTest Module

This module tests the functionality of various methods within the MagneticCore Module. The tests are implemented in TestMagneticCore.py, and the names of the tests are directly derived from the methods they evaluate. This naming convention ensures clarity and helps easily identify which method each test is designed to validate.

UT15: **Initialization Test**

UT16: **Calculate Current of Target Force**

UT17: **Calculate Current of Target Torque**

UT18: **Magnetic Torque Calculation**

UT19: **Magnetic Force Calculation**

UT20: **Magnetic Field Helmholtz**

UT21: **Derivatives of Magnetic Field Maxwell at Center**

5.2.4 OutputVerification Module

This module tests the functionality of the OutputVerification class, which is responsible for ensuring the accuracy of calculated outputs based on predefined criteria. The tests are implemented in TestOutputVerification.py. Each test is named after the specific functionality or condition it verifies.

UT22: **Currents Within Range Valid**

UT23: **Currents Within Range Invalid**

UT24: **Accuracy Valid**

UT25: **Accuracy Force Invalid**

UT26: **Accuracy Torque Invalid**

UT27: **Similarity Perfect Match**

UT28: **Similarity Opposite Direction**

UT29: **Similarity Zero Vector**

5.3 Tests for Nonfunctional Requirements

None.

5.4 Traceability Between Test Cases and Modules

Table 3: Traceability Between Test Cases and Modules

Module	Tests
T1	UT1 to UT21
T2	UT22, UT23
T3	UT24 to UT29

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?