Project Title: System Verification and Validation Plan for Inverted Pendulum Control Systems

Morteza Mirzaei

February 18, 2024

Revision History

Date	Version	Notes
2024-02-18	1.0	Initial Release.

Contents

List of Tables

List of Figures

1 Symbols, Abbreviations, and Acronyms

For complete symbols used within the system, please refer the section 1 in SRS document. Table 1 defines some abbreviations and acronyms used in this document.

Table 1: Table of Abbreviations and Acronyms.

symbol	**description**
SRS	Software Requirement Specification
VnV	Verification and Validation
MIS	Module Interface Specification
IPCS	Inverted Pendulum Control Systems
TC	Test Case

This document outlines the framework for Verification and Validation (VnV), delineating the objectives, scope, and methodologies for confirming the correctness and suitability of the software solution. Verification entails scrutinizing whether the software aligns with predetermined requirements and design parameters, whereas validation ensures it meets user expectations and demands effectively. Through this VnV framework, the team can uphold the integrity, dependability, and utility of the software, while also identifying and rectifying any flaws or discrepancies before deployment or release.

2 General Information

2.1 Summary

This document presents a verification and validation plan for Inverted Pendulum Control Systems, which aims to simulate, control, and visualize an inverted pendulum on a cart.

2.2 Objectives

The objectives outlined in this Verification and Validation plan are as follows:

- Validate the software's capability to precisely simulate an inverted pendulum.
- Verify the proper functioning of the control system to sustain the pendulum in an upright position.
- Confirm the correctness and user-friendliness of the visualization output.

2.3 Relevant Documentation

Refer to the Software Requirements Specification for comprehensive information regarding the objectives, requirements, assumptions, and underlying principles of the software.

3 Plan

This section will outline the strategy for validating the documentation and software for Inverted Pendulum Control Systems. The main components to be validated include the Software Requirements Specification (SRS), design, Verification and Validation (VnV) plan, and implementation.

3.1 Verification and Validation Team

The team members and their responsibilities are specified in Table 2.

Table 2: Verification and Validation Team

Name	**Document**	**Role**	**Description**
Morteza Mirzaei	All	Author	Create and manage all the documents, create the VnV plan, perform the VnV testing, verify the implementation.
Dr. Spencer Smith	All	Instructor/Reviewer	Review the documents, design and documentation style.
Seyed Ali Mousavi	All	Domain Expert Reviewer	Review all the documents.
Al Jubair Hossain	SRS	Secondary Reviewer	Review the SRS document
Hunter Ceranic	VnV Plan	Secondary Reviewer	Review the VnV plan.
Adrian Sochaniwsky	MG + MIS	Secondary Reviewer	Review the MG and MIS document.

3.2 SRS Verification Plan

The SRS document will be verified in the following way:

1. Initial review will be performed by the assigned members (Dr. Spencer Smith, Seyed Ali Mousavi, and Al Jubair Hossain). For this, a manual review will be conducted using the provided SRS Checklist and following additional checklist gather from here.

    - □ Are all requirements actually requirements, not design or implementation solutions?
    - □ Is each requirement uniquely and correctly identified?
    - □ Is each requirement verifiable by testing, demonstration, review, or analysis?
    - □ Do any requirements conflict with or duplicate other requirements?
    - □ Is each requirement in scope for the project?
    - □ Are all requirements written at a consistent and appropriate level of detail?
    - □ Do the requirements provide an adequate basis for design?

2. The reviewers will provide feedback to the author by creating an issue on GitHub.

3. Morteza Mirzaei, the author, is responsible for addressing the issues.

3.3 Design Verification Plan

Initial review will be performed by the assigned members (Dr. Spencer Smith, Seyed Ali Mousavi, and Adrian Sochaniwsky). For this, a manual review will be conducted using the following checklist.

- □ Does the design meet all specified requirements outlined in the design specification document?
- □ Are all interfaces between design components specified clearly?
- □ Are all components of the design testable separately?

- □ Is the design scalable to accommodate future growth or changes in requirements?
- □ Are the design components separated as much as possible?
- □ Does each design component have a logical task?

3.4 Verification and Validation Plan Verification Plan

The designated members (Dr. Spencer Smith, Seyed Ali Mousavi, and Hunter Ceranic) will conduct the initial review. This review will entail a manual assessment utilizing the following checklist.

- □ Does the VnV plan verify all the functional and non-functional requirements?
- □ Are all input and output pairs for test cases correct?
- □ Are roles and responsibilities clearly defined for individuals or teams involved in verification and validation activities?

3.5 Implementation Verification Plan

Inverted Pendulum Control Systems will be developed using the Python programming language. For static testing, we will use Flake8. Flake8 is a static code analysis tool for Python that checks code against coding style (PEP 8), detects syntax errors, and identifies potential bugs or problematic constructs. Additionally, we conduct code walkthroughs with domain expert (Ali Mousavi). For dynamic testing, we employ system and unit level testing, as explained in detail in sections Section 4 and Section 5.

3.6 Automated Testing and Verification Tools

- Pytest: A unit testing framework for Python that allows for easy and scalable testing of Python code.
- Flake8: A static code analysis tool for Python that checks code against coding style (PEP 8), detects syntax errors, and identifies potential bugs or problematic constructs.

- GitHub: This platform will be used for version control and collaboration.
- GitHub CI workflow: This will automate regression tests and checks that Inverted Pendulum Control Systems builds are passing before code is merged into a protected branch.
- Docker: A mechanism used to containerize applications, ensuring their installability and understandability throughout the project.

3.7 Software Validation Plan

No plan for software validation is currently in place.

4 System Test Description

4.1 Tests for Functional Requirements

4.1.1 Inputs and Outputs

This part verifies the input and output restrictions outlined in R1 for Inverted Pendulum Control Systems. To review these constraints, refer to the "Input Data Constraints" section in the SRS.

Table 3: Input validation test cases

Input						Output
F	x	θ	m	M	l	Valid
4	0	π	16	80	1	Yes
10	4	$\frac{3\pi}{4}$	4	200	20	No
20	0	π	6	20	30	No
-3	-5	π	-5	30	5	No

Inputs and Outputs test

1. test-inout-1

   Control: Automatic

Initial State: N/A.

Input: Specified in the Table 3.

Output: Specified in the Table 3.

Test Case Derivation: Directly from the requirement.

How test will be performed: At every stage, I input the data and assert the results by a testing framework.

4.1.2 Simulator

This part tests the world simulator responsible for executing the inverted pendulum physics. It assesses the requirements R2 and R3 for Inverted Pendulum Control Systems. Essentially, it verifies that given the initial position of the pendulum and a constant force acting on the cart, the location of the pendulum is determined after a specified duration of time.

Table 4: Output Validation Test Cases. The time step (dt) is 0.1 seconds. The results are calculated after 5 seconds.

Input						Output	
F	x	θ	m	M	l	x	θ
4	0	π	16	80	1	0.535	4.597
10	4	$\frac{3\pi}{4}$	4	50	20	6.368	4.005
20	0	π	6	20	4	9.897	3.869
-3	-5	π	4	30	5	-6.130	4.782

Simulator

1. test-sim-1

   Control: Automatic

   Initial State: Specified in the Table 4.

   Input: Specified in the Table 4.

   Output: Specified in the Table 4.

   Test Case Derivation: Solving the ODE's provided in the SRS with numberical analysis tools.

How test will be performed: At every stage, I input the data and assert the results by a testing framework.

4.1.3 Control System

This part tests the control system responsible for keeping the inverted pendulum upright. It assesses the requirements (R4) for Inverted Pendulum Control Systems. Essentially, it verifies that after a reasonable amount of time, the pendulum's angular position remains relatively stable in an upward position.

Control System

1. test-control-1

   Control: Automatic

   Initial State: Specified in the Table 4.

   Input: Specified in the Table 4.

   Output: θ should be between $-\frac{\pi}{4}$ and $\frac{\pi}{4}$ after 30 seconds.

   Test Case Derivation: Directly from the requirement.

   How test will be performed: input the data, and after 30 seconds, I will check if the value of θ remains within the specified range for 1 minute.

4.1.4 Visualization

This part verifies Requirement 5 (R5), ensuring that the visualization and graphical user interface (GUI) of the software meet the necessary standards.

Visualization

1. test-vis-1

   Control: Manual

   Initial State: Specified in the Table 4.

   Input: Specified in the Table 4.

Output: Visualization of the pendulum at different states.

Test Case Derivation: Directly from the requirement.

How test will be performed: First, Capture screenshots of the visualization at different states of the pendulum. Then, compare each screenshot with the expected state of the pendulum.

2. test-vis-2

   Type: Manual

   Initial State: N/A

   Input: A survey to be filled by the user. Ask them to provide feedback on clarity, ease of understanding, and any difficulties encountered.

   Output: an average score between 1 to 5 for each question.

   Test Case Derivation: Directly from the requirement.

   How test will be performed: After playing around with the visualization, users will be asked to fill out the following survey.

   - ☐ On a scale of 1 to 5, how clear is the visualization to you?
   - ☐ Are the location of the cart clear to you?
   - ☐ Are the angular position of the pendulum clear to you?
   - ☐ Could you get a sense of acceleration and velocity of the pendulum and cart?
   - ☐ Is the visualization smooth to you?

3. test-vis-3

   Type: Automatic

   Initial State: Specified in the Table 4.

   Input: Specified in the Table 4.

   Output: The frame rate of the visualization should be at least 30 frames per second.

   Test Case Derivation: Directly from the requirement.

   How test will be performed: Run the program and measure the frame rate of the visualization at every time step.

4.2 Tests for Nonfunctional Requirements

4.2.1 Accuracy

Accuracy

1. test-acc-1

   Type: Automatic

   Refer to Section 4.1.2.

4.2.2 Usability

Usability

1. test-useable-1

   Type: Automatic

   Refer to Section 4.1.4.

4.2.3 Maintainability

Maintainability

1. test-maintain-1

   Type: Manual

   Initial State: N/A

   Input/Condition: N/A

   Output/Result: A score from experts on the maintainability of the software.

   How test will be performed: We use survey in the Section 3.3 as a measure of maintainability. The design and survey will be given to the experts and they will provide feedback on the maintainability of the software.

4.2.4 Portability

Portability

1. test-port-1

   Type: Manual

   Initial State: N/A

   Input/Condition: test-sim-1 and est-vis-2.

   Output/Result: Successful test implies portability of the software.

   How test will be performed: The test will be performed manually by executing the software and tests on the Linux and macOS operating systems.

4.3 Traceability Between Test Cases and Requirements

Table 5: Relation of Test Cases and Requirements.

	R1	R2	R3	R4	R5	NFR1	NFR2	NFR3	NFR4
test-inout-1	X								
test-sim-1		X	X			X			
test-control-1				X					
test-vis-1					X				
test-vis-2					X				
test-vis-3					X				
test-acc-1		X	X			X			
test-useable-1					X		X		
test-maintain-1								X	
test-port-1									X

5 Unit Test Description

TBD. This section will be completed when the MIS is finished.

5.1 Unit Testing Scope

TBD. This section will be completed when the MIS is finished.

5.2 Tests for Functional Requirements

TBD. This section will be completed when the MIS is finished.

5.3 Tests for Nonfunctional Requirements

TBD. This section will be completed when the MIS is finished.

5.4 Traceability Between Test Cases and Modules

TBD. This section will be completed when the MIS is finished.