System Verification and Validation Plan for IP Simulator

Mina Mahdipour

April 19, 2023

1 Revision History

Date	Version	Notes
February 13, 2023	1.0	Created the first version of VnV
February 13, 2023	1.1	Added the plan part
February 17, 2023	1.2	Added system test description
February 18, 2023	1.3	Updated test section
February 20, 2023	1.5	Updated NFRs test cases
April 15, 2023	2	Updated all the documents

Contents

List of Tables

2 Symbols, Abbreviations and Acronyms

symbol	description
FR	Functional Requirements
MIS	Module Interface Specification
NFR	Nonfunctional Requirements
N/A	Not Applicable
VnV	Verification and Validation
SRS	Software Requirements Specification
T	Test
TC	Test Case

Section 1 of SRS document can be referred by the reader for complete symbols used within the system.

This document outlines the roadmap of the verification and validation plan for the Inverted Pendulum (IP) Simulator to help ensure (but not prove) the correctness and completeness of the program. It includes a plan for testing the functional and non-functional requirements, an overview of the system tests, and an outline of the unit tests.

3 General Information

3.1 Summary

This document provides the validation and verification plans for the IP Simulator Software. This software aims to simulate and describe the behavior of a cart-pendulum system in the presence of different values of external force. To do that, it will calculate the cart position and pendulum's angle after applying a time-varying force.

3.2 Objectives

The objectives of this document are listed below:

- Establish confidence in software reliability.
- Demonstrate adequate usability.
- Verify and validate the final product.

Performance testing, stress testing, and maintainability testing are outside of the scope of this document.

3.3 Relevant Documentation

Problem Statement presents an overview of the problem. The requirements and an outline of the solution for the IP Simulator are captured in the Software Requirements Specification, and verification of SRS is also a part of VnV process. The software design information will be captured in the Module Interface Specification. The final report of running the test which described in this document (unit test in section 6), will be shown in VnV Report.

4 Plan

This section provides a roadmap of plans for different steps of producing the final product.

4.1 Verification and Validation Team

Our team details are described in table 1:

Table 1: Verification and Validation Team

Role	**Responsibility**	**Assignee**	**Document**
Supervisor	Review and provide feedback	Dr. Spencer Smith	All documents
Author	Create the products and refine them by feedback	Mina Mahdipour	All documents
Domain Expert/ Reviewer	Review and provide feedback	Deesha Patel	All documents
Secondary Reviewer	Review and provide feedback	Maryam Valian	SRS
Secondary Reviewer	Review and provide feedback	Karen Wang	VnV plan
Secondary Reviewer	Review and provide feedback	Joachim de Fourestier	MG and MIS

4.2 SRS Verification Plan

The SRS has been independently peer-reviewed by members of the VnV team according to SRS-Checklist, consisting of the domain expert Deesha Patel and the SRS reviewer, Maryam Valian, as well as Dr. Spencer Smith. The SRS document has been published to GitHub. Any issues identified during the review were tracked and verified in Github platform.

4.3 Design Verification Plan

The design document MIS will be reviewed by the MIS review team, consisting of the domain expert, Deesha Patel and the MIS reviewer, Joachim de Fourestier, as well as Dr. Spencer Smith. The MIS document will be published to GitHub. Defects will be addressed with issues on the GitHub platform. There is also MIS-Checklist, that can be used.

4.4 Verification and Validation Plan Verification Plan

The VnV plan document will be reviewed by the VnV plan review team, consisting of the domain expert Deesha Patel and the reviewer, Karen Wang as well as Dr. Spencer Smith. This document will be published to GitHub. Defects will be addressed with issues on the GitHub platform. There is also VnV-Checklist, that can be used.

4.5 Implementation Verification Plan

The IP Simulator will be developed in Python programming language. The implementation of the software will be tested under the test cases listed in section 5. The tests will either be automated or performed manually.
Static analysis will be done using the Code Walkthrough technique by the domain expert (Deesha Patel). During a meeting, the author presents the document under review, while the domain expert will run the test cases over the code by hand. She may also ask the author any relevant questions and discuss her findings with the author [3]. Visual Studio Code will be used for program debugging and for checking syntax errors.

4.6 Automated Testing and Verification Tools

Automated testing and verification tools will be extensively used in the development of IP Simulator for automation at different levels. These tools include git, a distributed version-control system for tracking changes in source code during software development, Pytest framework for unit, functional, and integration automated testing, and Flake8 to check for errors, enforces coding standards, and identifies code complexity issues.

4.7 Software Validation Plan

There are no plans for the validation of the IP Simulator software.

5 System Test Description

This section will define the tests to ensure IP Simulator meets the functional requirements seen in section 5 of the SRS document for simulating inverted pendulum system. The subsections combine several requirements that are separated based on common ideas.

5.1 Tests for Functional Requirements

This section contains the system test cases for the functional requirements which are described in the SRS.

5.1.1 Input validation

This section includes tests to verify that the software rejects invalid input values such as non-numerical values or values outside of specified range. These tests also help to make sure that the IP Simulator accepts valid input values within specified range, FR1 and FR2 of requirements in section 5 of the SRS.

Input Constraints Test

1. TC-Valid-Inputs: Valid Inputs

   Control: Automatic

   Initial State: N/A

   Input: Set of input values as TC-Valid-Inputs test case in Table 2.

   Output: x, the position of the cart equals 0.1410 and the angle of the pendulum, θ, is 0.3643 when time, t, is 0.1 second. The initial values of x_i, $\dot{x_i}$, and $\dot{\theta} i$ are zero and θi is 0.1.

   Test Case Derivation: Derived from IP Simulator, as a normal and valid set of inputs.

   How the test will be performed: Automatic test using Pytest.

Table 2: Input Validation Test Cases

	Input					**Expected Output**	
TC	m_p	m_c	l	F	b	*Validity*	*Output*
Valid-Inputs	0.3	0.4	1.5	20	0.1	Yes	t= 1 x= 6.4817 θ= 0.5560
I-Zero-Input-m_p	0	0.5	0.3	50	0.2	No	Error: invalid input
I-Zero-Input-m_c	0.2	0	0.3	50	0.2	No	Error: invalid input
I-Zero-Input-l_p	0.2	0.5	0	50	0.2	No	Error: invalid input
V-Zero-Input-b	0.2	0.5	0.3	50	0	Yes	t= 6 x=1023 θ= 0.57
V-Zero-Input-f	0.2	0.5	0.3	0	0.2	Yes	t= 1 x=0.3121 θ= 0.8265
Neg-Input-m_p	-0.2	0.5	0.3	50	0.2	No	Error: invalid input
Neg-Input-m_c	0.2	-0.5	0.3	50	0.2	No	Error: invalid input
Neg-Input-l_p	0.2	0.5	-0.3	50	0.2	No	Error: invalid input
Neg-Input-f	0.2	0.5	0.3	-50	0.2	No	Error: invalid input
Neg-Input-b	0.2	0.5	0.3	50	-0.2	No	Error: invalid input
OutofBound-m_p	51	0.5	0.3	50	0.2	No	Error: invalid input
OutofBound-m_c	0.2	51	0.3	50	0.2	No	Error: invalid input
OutofBound-l_p	0.2	0.5	11	50	0.2	No	Error: invalid input
Null-Input-m_p		0.5	0.3	50	0.2	No	Error: invalid input
Null-Input-m_c	0.2		0.3	50	0.2	No	Error: invalid input
Null-Input-l_p	0.2	0.5		50	0.2	No	Error: invalid input
Null-Input-f	0.2	0.5	0.3		0.2	No	Error: invalid input
Null-Input-b	0.2	0.5	0.3	50		No	Error: invalid input

2. TC-I-Zero-Input: Zero Inputs

   Control: Automatic

   Initial State: N/A

   Input: Set of invalid zero inputs as I-Zero-Input-m_p, I-Zero-Input-m_c, and I-Zero-Input-l_p test cases in Table 2.

   Output: Error message of invalid input because of zero inputs.

   Test Case Derivation: Zero input for the mass of the pendulum or mass

of the cart, or the length of the cart is not valid.

How the test will be performed: Automatic test using Pytest.

3. TC-V-Zero-Input-b: Valid and Zero Inputs

   Control: Automatic

   Initial State: N/A

   Input: Set of input values as V-Zero-Input-b test case in Table 2.

   Output: x, the position of the cart and θ, the angle of the pendulum

   Test Case Derivation: Derived from IP Simulator, as a normal and valid set of inputs. How the test will be performed: Automatic test using Pytest.

4. TC-V-Zero-Input-f: Valid and Zero Inputs

   Control: Automatic

   Initial State: N/A

   Input: Set of input values as V-Zero-Input-f test case in Table 2.

   Output: x, the position of the cart and θ, the angle of the pendulum

   Test Case Derivation: Derived from IP Simulator, as a normal and valid set of inputs. How the test will be performed: Automatic test using Pytest.

5. TC-Neg-Input: Negative Inputs

   Control: Automatic

   Initial State: N/A

   Input: Set of input values as Neg-Input-m_p, Neg-Input-m_c, Neg-Input-l_p, Neg-Input-f, and Neg-Input-b test cases in Table 2.

   Output: Error message of invalid negative input.

   Test Case Derivation: Inputs can not be negative values.

   How the test will be performed: Automatic test using Pytest.

6. TC-OutofBound: Out of Bound Inputs

   Control: Automatic

   Initial State: N/A

Input: Set of input values as OutofBound-m_p, OutofBound-m_c, and OutofBound-l_p test cases in Table.2.

Output: Error message of inputs because of constraints.

Test Case Derivation: The user should prepare values for inputs in the valid range for each.

How the test will be performed: Automatic test using Pytest.

7. TC-Null-Input: Null Inputs

   Control: Automatic

   Initial State: N/A

   Input: Set of input values as Null-Input-m_c, Null-Input-m_p, Null-Input-l_p, Null-Input-f, and Null-Input-b test cases, in Table.2.

   Output: Error message of invalid missing input.

   Test Case Derivation: The user should provide all the inputs without missing them.

   How the test will be performed: Automatic test using Pytest.

5.1.2 Output Validity

This section presents tests to make confidence in the out put and also error handling of the software, which means the system can generate the expected results from valid inputs and otherwise handle unexpected errors gracefully and provides clear error messages when errors occur. The other goal of these tests is to verify that the system simulates the cart moving back and forth and that the pendulum rotates according to input values, as the R3 and R4 of requirements in section 5 of the SRS has defined. All the outputs are measured at time equals to 3.000 seconds.

1. TC-IP-Out-1

   Control: Automatic

   Initial State: the software has run by the input values of TC-IP-Out-1 in Table 3 and the initial conditions are $x_i = 0.2$, $\dot{x_i} = 0$, $\theta = 0.3$, $\dot{\theta} i = 0$.

   Input: input variables like TC-IP-Out-1 in Table 3.

Table 3: Output Validation Test Cases

	Input					**Expected Output**	
TC	m_p	m_c	l	F	b	*Output*	
TC-IP-Out-1	0.2	1.5	1.3	50	0.1	$x = 109.000$	$\theta = 0.6984$
TC-IP-Out-2	2.5	0.1	0.3	20	0	$x = 21.6257$	$\theta = 0.4034$
TC-IP-Out-3	0.5	4	1.3	2	0.2	$x = 11.0721$	$\theta = 0.9152$
TC-IP-Out-4	0	51	-0.3	50	-0.2	Value Error	

Output: The x will be 109.000 m and θ equals to 0.6984 rad.
Test Case Derivation: Derived form IP Simulator, as a normal and valid set of inputs have been provided.
How test will be performed: It will be performed by IP Simulator and Pytest.

2. TC-IP-Out-2

   Control: Automatic

   Initial State: the software has run by the input values of TC-IP-Out-2 in Table 3 and the initial conditions are $x_i = 0$, $\dot{x_i} = 0$, $\theta = 0.1$, $\dot{\theta} i = 0$.
   Input: valid inputs like TC-IP-Out-2 in Table 3.
   Output: The x will be 21.6257 m and θ equals to 0.4034 rad.
   How test will be performed: It will be performed by IP Simulator, Pytest and Flake8.

3. TC-IP-Out-3

   Control: Automatic

   Initial State: the software has run by the input values of TC-IP-Out-3 in Table 3 and the initial conditions are $x_i = 0$, $\dot{x_i} = 0.2$, $\theta = 0.1$, $\dot{\theta} i = 0$.
   Input: valid inputs like TC-IP-Out-3 in Table 3.
   Output: The x will be 11.0721 m and θ equals to 0.9152 rad.
   How test will be performed: It will be performed by IP Simulator, Pytest and Flake8.

4. TC-IP-Out-4

   Control: Automatic

   Initial State: the software has run by the input values of TC-IP-Out-4 in Table 3 and the initial conditions are $x_i = 0$, $\dot{x_i} = 0.2$, $\theta = 0.1$, $\dot{\theta i} = 0$.
   Input: invalid inputs like TC-IP-Out-4 in Table 3.
   Output: The Value Error will be raised.
   How test will be performed: It will be performed by IP Simulator, Pytest and Flake8.

5.2 Tests for Nonfunctional Requirements

5.2.1 Portability

Testing the portability of IP Simulator

1. T1: Portability

   Type: Manual

   Initial State: None.

   Requirement ID(s): NFR-1

   Input/Condition: Executes all tests in 5.1.1 and 5.1.2 in the IP Simulator in different operating systems, including Linux, Windows, and MacOS.

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

   How the test will be performed: The test will be performed manually by executing the software and tests in different OS, including Linux, Windows, and MacOS. A successful test is specified by the verification of each test for functional requirements.

5.2.2 Usability

Testing the Usability of IP Simulator

1. T1: Usability

   Testing the usability will determine if the users have efficient interaction with the software and also the software is easy to use or not. The system

will be tested against a usability test and survey which the details are in the Appendix 7.

Type: Manual

Initial State: The IP Simulator runs.

Requirement ID(s): NFR-2

Input/Condition: Any tests considering inputs criteria can be run by a group of individuals who have varying levels of experience with inverted pendulum systems and the software, including at least an undergraduate Physics student, a high school student, and the domain expert. We will administer the questionnaire to the participants while they use the software and observe their behavior and note any issues they encounter.

Output/Result: According to the feedback from the survey, the usability of the software will be tested.

How the test will be performed: Manually, as it is a test of usability by the user, and thus requires the test to be run by the users.

5.2.3 Accuracy

Testing the Accuracy of IP Simulator

1. T1: Accuracy

   The accuracy test should be done by comparing the results with a similar system that is available on the Internet and has been developed using MATLAB. [1]

   Type: Manual

   Initial State: The IP Simulator and a similar project in MATLAB with the same inputs run.

   Requirement ID(s): NFR-3

   Input/Condition: All of the tests in 5.1.2 will be done.

   Output/Result: Comparing the outputs of IP Simulator to the result of [1] using the relative error. If the results from running the IP Simulator considered as IP Measured Value and the results of running [1] as Actual Value, the relative error can be calculated as below:
   $AbsoluteError = |ActualValue - IPMeasuredValue|$

and then:
$RelativeError = (\frac{AbsoluteError}{ActualValue})$

The acceptable value for $RelativeError$ is less than 0.01.

How the test will be performed: Manually, as both projects should be run manually.

5.3 Traceability Between Test Cases and Requirements

A traceability between test cases and requirements is shown in table 4.

Table 4: Traceability Matrix between tests and requirements

Test Cases	R1	R2	R3	R4	NFR1	NFR2	NFR3
5.1.1	X	X					
5.1.2			X	X			
5.2.1					X		
5.2.2						X	
5.2.3							X

6 Unit Test Description

We will use Pytest framework for unit, functional, and integration automated testing.

6.1 Constant Parameter Module

This module is suppose to store the constants for the software, including constraints on input values, simulation variables and gravity. The *constantM.py* implements this requirement and using Pytest this module is tested and verified by running *test_constantM.py*.

6.2 Input Parameters Module

This module is implemented in *inputM.py*, has three functions, the read_inputs reads all the inputs from the file that users inserted, as described in section 6 in MIS. The constrains on the parameters are checked in verify_inputs using Constant Parameter Module, and then if all the constraints are met, convert_inputs prepares the parameters for the calculation process. *Test_inputM.py* checks the function of this module.

6.3 Motion ODE Module

OdeM.py implements Motion ODE Module, and defines the motion equation for the cart and pendulum with their specification.

6.4 ODE Solver Module

ODEs defined in the motion ODE module in addition to initial conditions that Input Parameters Module reads, are passed to ODE Solver Module that is implemented in *odeSolverM.py*. In this module by using *Scipy* library, the ODEs will be solved.

6.5 Output Module

OutputM.py is responsible to verify the outputs and writes them to the output file and to do that, it has two functions, verify_output, which checks the constrains on the generated outputs and write_output_to_file function that gives the output file and the output and then writes output to the specified file. The output includes an array of time and corresponding values of position of the cart, velocity of the cart, the angle position of the pendulum, and the velocity of the pendulum.

6.6 Plotting Module

This module gives the time and the output of ODE solver module and shows the results as a graph in *plotM.py*

6.7 IP Control Module

IP control module runs all the modules in a right way in $mainM.py$. $Test\_mainM.py$ verifies the function of output, plotting modules together with all modules.

References

[1] Huthaifa AL-Khazraji. Matlab code of inverted pendulum on cart using ode45 (animation), Apr 2022. URL https://www.youtube.com/watch?v=glf8HC5CiyE&t=162s.

[2] Andreas Hinderks, Martin Schrepp, and Jörg Thomaschewski. User experience questionnaire. URL https://www.ueq-online.org/.

[3] Swati Tawde. Code walkthrough. URL https://www.educba.com/code-walkthrough/.

7 Appendix

7.1 Usability Survey Questions?

To test usability of the IP Simulator, the Hinderks et al. questionnaire [2] will be used.