Verification and Validation Report: Inverted Pendulum Control Systems (IPCS)

Morteza Mirzaei

April 16, 2024

1 Revision History

Date	**Version**	**Notes**
2024-04-13	1.0	Initial Release.

2 Symbols, Abbreviations and Acronyms

symbol	**description**
T	Test
UT	Unit Test

Contents

List of Tables

List of Figures

This document reports the results of executing the VnV Plan.

3 Functional Requirements Evaluation

This section covers the evaluation of the functional requirements.

3.1 test-input-1

This test checks if the software returns proper exceptions for invalid inputs and works properly for valid inputs. This test is implemented in the file https://github.com/mirzaim/ipcs/blob/main/test/test_integration.py, and the function name is `test_input_1`. The test passed successfully.

```
test_integration.py::test_input_1[0.0-0.0--90.0-16.0-80.0-1.0-False]  PASSED
test_integration.py::test_input_1[abc-0.0--90.0-16.0-80.0-1.0-True]   PASSED
test_integration.py::test_input_1[0.0-abc--90.0-16.0-80.0-1.0-True]   PASSED
test_integration.py::test_input_1[0.0-0.0-abc-16.0-80.0-1.0-True]     PASSED
test_integration.py::test_input_1[0.0-0.0--90.0-abc-80.0-1.0-True]    PASSED
test_integration.py::test_input_1[0.0-0.0--90.0-16.0-abc-1.0-True]    PASSED
test_integration.py::test_input_1[0.0-0.0--90.0-16.0-80.0-abc-True]   PASSED
test_integration.py::test_input_1[0.0-0.0--90.0--16.0-80.0-1.0-True]  PASSED
test_integration.py::test_input_1[0.0-0.0--90.0-16.0--80.0-1.0-True]  PASSED
test_integration.py::test_input_1[0.0-0.0--90.0-16.0-80.0--1.0-True]  PASSED
test_integration.py::test_input_1[0.0-0.0--90.0-100.0-80.0-1.0-True]  PASSED
test_integration.py::test_input_1[0.0-0.0--90.0-16.0-100.0-1.0-True]  PASSED
test_integration.py::test_input_1[0.0-0.0--90.0-16.0-80.0-50.0-True]  PASSED
test_integration.py::test_input_1[0.0-0.0--90.0-5.0-80.0-1.0-True]    PASSED
test_integration.py::test_input_1[0.0-0.0--90.0-90.0-80.0-1.0-True]   PASSED
test_integration.py::test_input_1[0.0--20.0--90.0-16.0-80.0-1.0-True] PASSED
test_integration.py::test_input_1[0.0-20.0--90.0-16.0-80.0-1.0-True]  PASSED
```

3.2 test-sim-1

This section focuses on testing the world simulator responsible for executing the inverted pendulum physics. The testing procedure involves verifying that given the initial position of the pendulum the location of the pendulum is accurately determined after a specified duration of time. This test is implemented in the file https://github.com/mirzaim/ipcs/blob/main/test/

test_integration.py, and the function name is `test_sim_1`. If the relative error between the expected and actual values is less than 5%, the test is considered successful. All the test passed successfully.

```
test_integration.py::test_sim_1[0.0-0.0--90.0-16.0-80.0-1.0-5.0-0.25]   PASSED
test_integration.py::test_sim_1[0.0--5.0--90.0-16.0-80.0-1.0-5.0--4.75] PASSED
test_integration.py::test_sim_1[0.0-5.0--90.0-16.0-80.0-1.0-5.0-5.25]   PASSED
test_integration.py::test_sim_1[0.0-0.0-0.0-16.0-80.0-1.0-6.0-0.08]     PASSED
test_integration.py::test_sim_1[0.0-0.0--45.0-16.0-80.0-1.0-2.5--0.3]   PASSED
test_integration.py::test_sim_1[0.0-0.0--90.0-30.0-80.0-1.0-1.5-0.8]    PASSED
test_integration.py::test_sim_1[0.0-0.0--90.0-16.0-50.0-1.0-1.45-1.3]   PASSED
test_integration.py::test_sim_1[0.0-0.0--90.0-16.0-80.0-20.0-5.0-0.25]  PASSED
test_integration.py::test_sim_1[5.0-0.0--90.0-16.0-80.0-1.0-2.7-0.65]   PASSED
test_integration.py::test_sim_1[-5.0-0.0--90.0-16.0-80.0-1.0-5.5-0.0]   PASSED
```

3.3 test-control-1

The testing procedure involves verifying that given the initial position of the pendulum the control system can keeping the pendulum upright for 10 seconds after 30 seconds. The test is implemented in the file https://github.com/mirzaim/ipcs/blob/main/test/test_integration.py and the function name is `test_control_1`. All the test passed successfully.

```
test_integration.py::test_control_1[0.0-0.0--90.0-16.0-80.0-1.0]  PASSED
test_integration.py::test_control_1[0.0--5.0--90.0-16.0-80.0-1.0] PASSED
test_integration.py::test_control_1[0.0-5.0--90.0-16.0-80.0-1.0]  PASSED
test_integration.py::test_control_1[0.0-0.0-0.0-16.0-80.0-1.0]    PASSED
test_integration.py::test_control_1[0.0-0.0--45.0-16.0-80.0-1.0]  PASSED
test_integration.py::test_control_1[0.0-0.0--90.0-30.0-80.0-1.0]  PASSED
test_integration.py::test_control_1[0.0-0.0--90.0-16.0-50.0-1.0]  PASSED
test_integration.py::test_control_1[0.0-0.0--90.0-16.0-80.0-20.0] PASSED
test_integration.py::test_control_1[5.0-0.0--90.0-16.0-80.0-1.0]  PASSED
test_integration.py::test_control_1[-5.0-0.0--90.0-16.0-80.0-1.0] PASSED
```

3.4 test-vis-1

This test checks the correctness of the visualization. We took 4 screenshots (Figure 1, Figure 2, Figure 3, and Figure 4) of the visualization with different inputs. The correctness of the visualization is evaluated and all the test passed successfully.

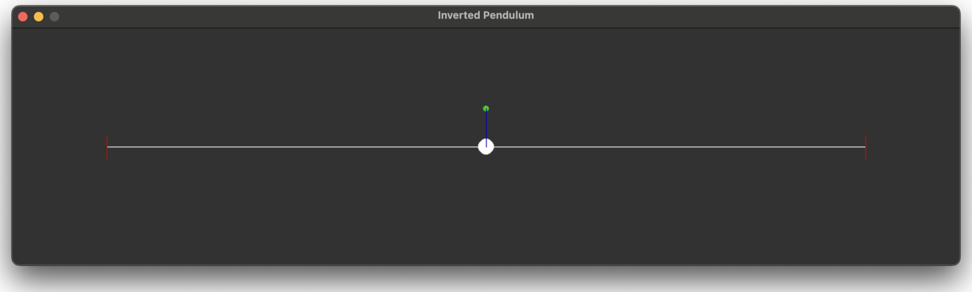

Figure 1: $x = 0$ and $\theta = 0$

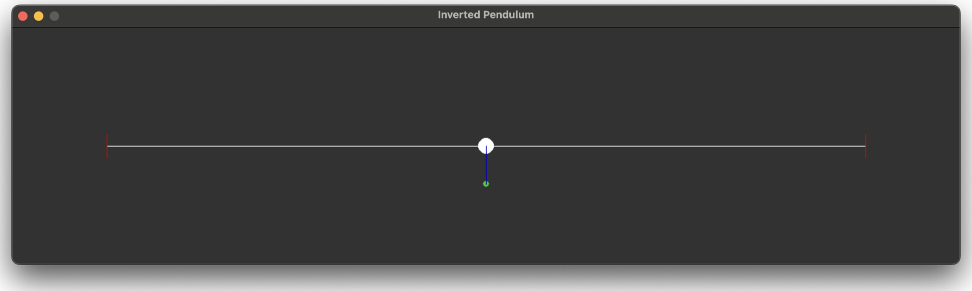

Figure 2: $x = 0$ and $\theta = \pi$

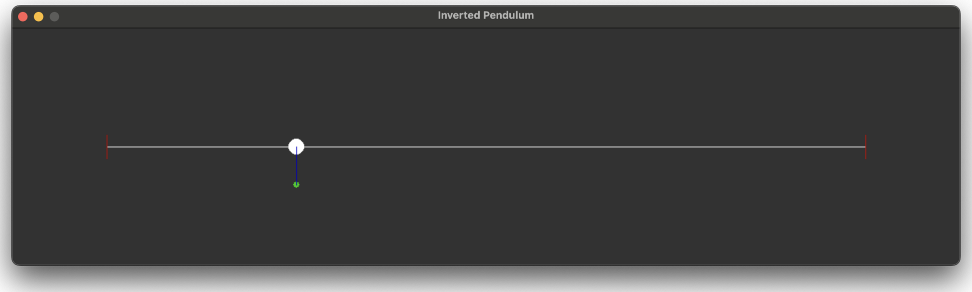

Figure 3: $x = -5$ and $\theta = \pi$

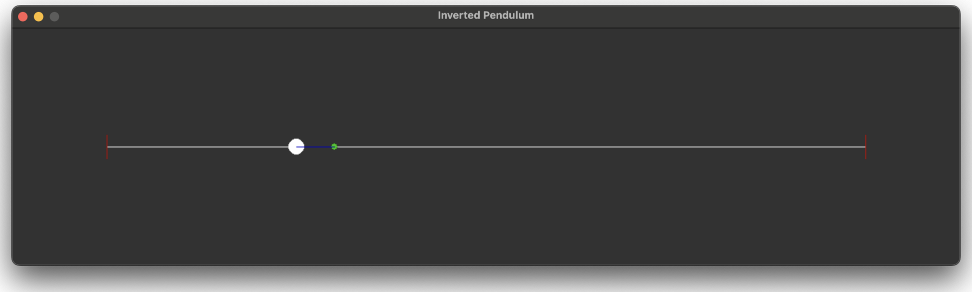

Figure 4: $x = -5$ and $\theta = \frac{\pi}{2}$

3.5 test-vis-2

This section focuses on testing the user-friendliness of the visualization aspect of the system. The survey results conducted by the domain expert are shown in Table 1. The average score obtained is 4.75, which indicates a successful outcome.

4 Nonfunctional Requirements Evaluation

This section covers the evaluation of the nonfunctional requirements.

Table 1: Domain expert feedback for test-vis-2. The average score is 4.75.

No.	Question	Answer
1.	On a scale of 1 to 5, how clear is the visualization to you?	5
2.	On a scale of 1 to 5, how clear is the location of the cart to you?	5
3.	On a scale of 1 to 5, how clear is the angular position of the pendulum to you?	5
4.	On a scale of 1 to 5, how smooth is the visualization to you?	4

4.1 Accuracy

The test is implemented in the file https://github.com/mirzaim/ipcs/blob/main/test/test_integration.py and the function name is `test_acc_1`. The average relative error for cart location is 1.2% and for pendulum angle is 2.4%. The test passed successfully.

4.2 Usability

This section focuses on testing the usability of the system. The survey results conducted by the domain expert are shown in Table 2. All the answers are positive, which indicates a successful outcome.

Table 2: Usability test survey, filled by the domain expert.

No.	Question	Answer
1.	Could you run the software on your operating system?	Yes
2.	Is system running smoothly on your computer?	Yes
3.	Is invalid input's message clear?	Yes
4.	Is text easy to read?	Yes

4.3 Maintainability

This section is dedicated to assessing the system's maintainability. The results of the survey, administered by a domain expert, are presented in Table 3. All responses received were affirmative, indicating a successful outcome.

Table 3: Maintainability test survey, filled by the domain expert.

No.	Question	Answer
1.	Does the design meet all specified requirements outlined in the design specification document?	Yes
2.	Are all interfaces between design components specified clearly?	Yes
3.	Are all components of the design testable separately?	Yes
4.	Is the design scalable to accommodate future growth or changes in requirements?	Yes
5.	Are the design components separated as much as possible?	Yes
6.	Does each design component have a logical task?	Yes

4.4 Portability

This section is dedicated to assessing the system's portability, for which we conducted the following two tests.

test-port-1

This test evaluates the system's portability across different operating systems, specifically focusing on Linux and Mac OS.

1. **Linux Testing:** The software is rigorously tested on Linux using GitHub workflows. After each push to the main branch, all tests are automatically executed on Ubuntu 22.04. The test results demonstrate successful execution.

2. **Mac OS Testing:** Similarly, testing is conducted on Mac OS using a MacBook Pro 2022 equipped with the M2 chip. All tests are executed on this platform, and once again, all tests pass successfully.

test-port-2

This test evaluates docker version of the software. Every update to the main branch is automatically built and pushed to the Docker Hub. The docker image is available at https://hub.docker.com/r/mirzaim/ipcs. The docker image can be run using the following command:

```
sudo docker run -v ./out/:/ipcs/out --rm -it mirzaim/ipcs:latest
or
sudo docker run -v ./out/:/ipcs/out \
                -v <path to config file>:/ipcs/configs/default.ini \
                --rm -it mirzaim/ipcs:latest
```

The test passed successfully.

5 Unit Testing

The files in the https://github.com/mirzaim/ipcs/tree/main/test directory that starts with `test_<module_name>.py` are the unit tests. We have 15 unit tests and 47 integration tests. All the unit tests passed successfully.

```
test_checker.py::test_conf[conf0]                       PASSED
test_checker.py::test_conf[conf1]                       PASSED
test_checker.py::test_conf[conf2]                       PASSED
test_conf.py::test_file_not_found                       PASSED
test_conf.py::test_no_input_exception                   PASSED
test_fuzzy.py::test_fuzzy[world_conf0-0.0]              PASSED
test_fuzzy.py::test_fuzzy[world_conf1-80.0]             PASSED
test_fuzzy.py::test_fuzzy[world_conf2-0.0]              PASSED
test_fuzzy.py::test_fuzzy[world_conf3-80.0]             PASSED
test_simulator.py::test_exception[-1.0]                 PASSED
test_simulator.py::test_exception[-0.1]                 PASSED
test_simulator.py::test_exception[0.0]                  PASSED
test_simulator.py::test_tick[0.0-0.0-0.0-6.185]         PASSED
test_simulator.py::test_tick[5.0-0.0-5.0-6.185]         PASSED
test_simulator.py::test_tick[0.0--90.0-0.0-4.712]       PASSED
```

6 Changes Due to Testing

N/A.

7 Automated Testing

The tests are set up to run automatically when a Pull Request is opened and after any commit is made to the `main` branch. The GitHub workflow installs the required dependencies and runs the pytest command. Additionally, after a successful pull request, the Docker image is built and pushed to the Docker Hub.

8 Trace to Requirements

The traceability between test cases and requirements is shown in Table 4.

Table 4: 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-acc-1		X	X			X			
test-useable-1					X		X		
test-maintain-1								X	
test-port-1									X
test-port-2									X

9 Trace to Modules

Traceability between test cases and modules is shown in Table 5.

10 Code Coverage Metrics

Code Coverage is out of scope of the VnV Plan.

Table 5: Traceability Between Test Cases and Modules

Module	Tests
World Checker	UT1
Config Reade	UT2, UT3
Fuzzy Controller	UT4
Simulation	UT5, UT6