

Verification and Validation Report: IP Simulator

Mina Mahdipour

April 19, 2023

1 Revision History

Date	Version	Notes
April 16, 2023	1.0	First draft of report
April 18, 2023	1.1	Added non functional and unit tests

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
TC	Test Case
SRS	Software Requirements Specification
VnV	Verification and Validation
IP Simulator	Inverted Pendulum Simulator
MG	Module Guide
MIS	Module Interface Specification

Section 1 of [SRS](#) document can be referred by the reader for complete symbols used within the system.

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Functional Requirements Evaluation	1
3.1	Static Testing	1
3.1.1	Code Walkthrough	1
3.2	Dynamic Testing	1
3.2.1	Input Validation	1
3.2.2	Output Validation	5
4	Nonfunctional Requirements Evaluation	6
4.1	Portability	6
4.2	Usability	7
4.3	Accuracy	11
5	Comparison to Existing Implementation	16
6	Unit Testing	16
6.1	Tests for Functional Requirements	16
6.1.1	Constant Parameter Module	16
6.1.2	Input Parameters Module	17
6.1.3	Motion ODE Module	17
6.1.4	ODE Solver Module	17
6.1.5	Output Module	17
6.1.6	Plotting Module	18
6.1.7	IP Control Module	18
7	Changes Due to Testing	18
8	Automated Testing	19
9	Trace to Requirements	19
10	Trace to Modules	20
11	Code Coverage Metrics	20

List of Tables

1	TC-Valid-Inputs	1
2	The results of TC-Valid-Inputs	2
3	TC-V-Zero-Input- b	2
4	The results of TC-V-Zero-Input- b	2
5	TC-V-Zero-Input- f	2
6	The results of TC-V-Zero-Input- f	3
7	TC-IP-Out-1	5
8	The results of TC-IP-Out-1	5
9	TC-IP-Out-2	6
10	The results of TC-IP-Out-2	6
11	TC-IP-Out-3	6
12	The results of TC-IP-Out-3	7
13	Test Case-2	7
14	Test Case-2	8
15	TC-normal values with zero function	11
16	Relative error for TC-normal values with zero function	11
17	TC-normal values with zero function in a shorter time	12
18	Relative error for TC-normal values with zero function in a shorter time	12
19	TC-positive normal values in longer time	13
20	Relative Error for TC-positive normal values in longer time	13
21	TC-positive normal values in a shorter time	14
22	Relative Error for TC-positive normal values in shorter time	14
23	TC-normal positive values with different initial conditions	15
24	Relative Error for TC-normal positive values with different initial conditions	15
25	Traceability Between Test Cases and Requirements	20
26	Traceability Between Test Cases and modules	20

List of Figures

1	Tests for checking the validity of mass of the cart	3
2	Tests for checking the validity of mass of the pendulum	4
3	Tests for checking the validity of length of pendulum	4
4	Tests for checking the validity of cart friction	5

5	The mean values and variances of all the responses to each question	8
6	Scale Means value per person	9
7	Mean value per item	9
8	Mean values and variance for each factor	10
9	TC-normal values with zero function	12
10	TC-normal values with zero function in a shorter time	13
11	TC-positive normal values in a longer time	14
12	TC-positive normal values in shorter time	15
13	TC-normal positive values with different initial conditions	16
14	The result of running unit tests	18
15	Running flake8 on the modules	19
16	Code Coverage	21

This document provides a summary of the verification and validation (VnV) of the IP Simulator software. The test cases listed in the [VnV plan](#) were executed, and this document contains a summary of the results. Sections 3 and 4 summarize the results of the functional and non-functional requirements respectively. Subsequent sections summarize the test results in detail.

3 Functional Requirements Evaluation

3.1 Static Testing

3.1.1 Code Walkthrough

Unfortunately, because of time, I could not do the static test.

3.2 Dynamic Testing

This section outlines the results of tests in table 2 in section 5.1 of the [VnV plan](#). All the tests are related to input validation.

3.2.1 Input Validation

Table 1 sets up the environment to run the test TC-Valid-Inputs, table 2 shows the expected output that is the result of running [1], and actual outputs are the outputs of the IP Simulator.

Table 1: TC-Valid-Inputs

Initial Conditions				Inputs					Time
x_i	dx_i	θ_i	$\dot{\theta}_i$	m_p	m_c	l_p	f	b	duration
0	0	0.1	0	0.3	0.4	1.5	20	0.1	1.0000

Table 3 sets up the environment to run the test TC-V-Zero-Input- b , table 4 shows the expected output that is the result of running [1], and actual

Table 2: The results of TC-Valid-Inputs

Expected Output		Actual Output	
x_i	θi	x_i	θi
6.4817	0.5560	6.4483	0.5583

outputs are the outputs of the IP Simulator.

Table 3: TC-V-Zero-Input- b

Initial Conditions				Inputs					Time
x_i	dx_i	θi	$\dot{\theta}_i$	m_p	m_c	l_p	f	b	Time
0	0	0.1	0	0.5	0.2	0.3	50	0	6.000

Table 4: The results of TC-V-Zero-Input- b

Expected Output		Actual Output	
x_i	θi	x_i	θi
1043	0.57	1047.60	0.53

Table 5 sets up the environment to run the test TC-V-Zero-Input- f , table 6 shows the expected output that is the result of running [1], and actual outputs are the outputs of the IP Simulator.

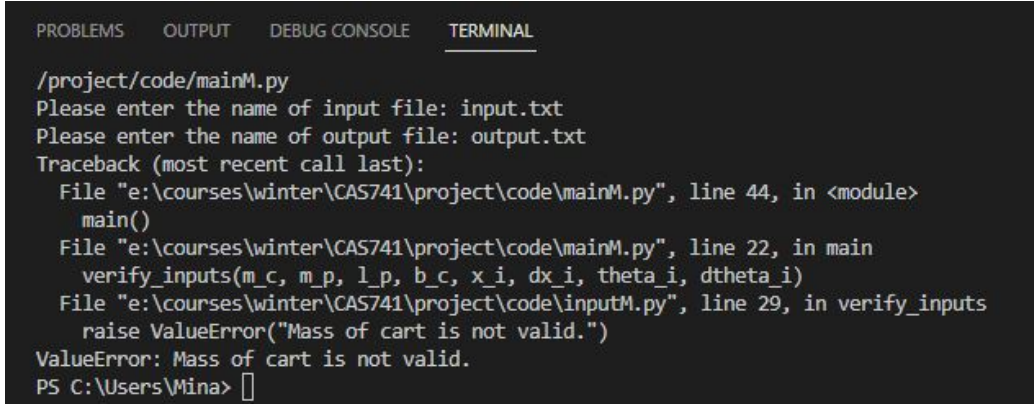
Table 5: TC-V-Zero-Input- f

Initial Conditions				Inputs					Time
x_i	dx_i	θi	$\dot{\theta}_i$	m_p	m_c	l_p	f	b	Time
0	0	0.1	0	0.5	0.2	0.3	0	0.2	1.000

All the tests related to checking the validity of m_c , including I-Zero-Input- m_c , Neg-Input- m_c , OutofBound- m_c , and Null-Input- m_c are passed successfully and figure 1 shows the result that a Value Error has raised.

Table 6: The results of TC-V-Zero-Input- f

Expected Output		Actual Output	
x_i	θ_i	x_i	θ_i
0.3121	0.8265	0.3120	0.8260



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

/project/code/mainM.py
Please enter the name of input file: input.txt
Please enter the name of output file: output.txt
Traceback (most recent call last):
  File "e:\courses\winter\CAS741\project\code\mainM.py", line 44, in <module>
    main()
  File "e:\courses\winter\CAS741\project\code\mainM.py", line 22, in main
    verify_inputs(m_c, m_p, l_p, b_c, x_i, dx_i, theta_i, dtheta_i)
  File "e:\courses\winter\CAS741\project\code\inputM.py", line 29, in verify_inputs
    raise ValueError("Mass of cart is not valid.")
ValueError: Mass of cart is not valid.
PS C:\Users\Mina> 

```

Figure 1: Tests for checking the validity of mass of the cart

All the tests related to checking the validity of m_p , including I-Zero-Input- m_p , Neg-Input- m_p , OutofBound- m_p , and Null-Input- m_p are passed and figure 2 shows the result that a Value Error has raised.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

/project/code/mainM.py
Please enter the name of input file: input.txt
Please enter the name of output file: output.txt
Traceback (most recent call last):
  File "e:\courses\winter\CAS741\project\code\mainM.py", line 44, in <module>
    main()
  File "e:\courses\winter\CAS741\project\code\mainM.py", line 22, in main
    verify_inputs(m_c, m_p, l_p, b_c, x_i, dx_i, theta_i, dtheta_i)
  File "e:\courses\winter\CAS741\project\code\inputM.py", line 33, in verify_inputs
    raise ValueError("Mass of pendulum is not valid.")
ValueError: Mass of pendulum is not valid.
PS C:\Users\Mina> |
```

Figure 2: Tests for checking the validity of mass of the pendulum

All the tests related to checking the validity of l_p , including I-Zero-Input- l_p , Neg-Input- l_p , OutofBound- l_p , and Null-Input- l_p are passed and figure 3 shows the result that a Value Error has raised.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

/project/code/mainM.py
Please enter the name of input file: input.txt
Please enter the name of output file: output.txt
Traceback (most recent call last):
  File "e:\courses\winter\CAS741\project\code\mainM.py", line 44, in <module>
    main()
  File "e:\courses\winter\CAS741\project\code\mainM.py", line 22, in main
    verify_inputs(m_c, m_p, l_p, b_c, x_i, dx_i, theta_i, dtheta_i)
  File "e:\courses\winter\CAS741\project\code\inputM.py", line 37, in verify_inputs
    raise ValueError("Length of pendulum is not valid.")
ValueError: Length of pendulum is not valid.
PS C:\Users\Mina> |
```

Figure 3: Tests for checking the validity of length of pendulum

All the tests related to checking the validity of b , including Neg-Input- b and Null-Input- b are passed and figure 4 shows the result that a Value Error has been raised.

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

/project/code/mainM.py
Please enter the name of input file: input.txt
Please enter the name of output file: output.txt
Traceback (most recent call last):
  File "e:\courses\winter\CAS741\project\code\mainM.py", line 44, in <module>
    main()
  File "e:\courses\winter\CAS741\project\code\mainM.py", line 22, in main
    verify_inputs(m_c, m_p, l_p, b_c, x_i, dx_i, theta_i, dtheta_i)
  File "e:\courses\winter\CAS741\project\code\inputM.py", line 40, in verify_inputs
    raise ValueError("Friction of cart is not valid.")
ValueError: Friction of cart is not valid.
PS C:\Users\Mina>

```

Figure 4: Tests for checking the validity of cart friction

3.2.2 Output Validation

This section outline the validation of outputs according to table 3 in section 5.1.2 in [VnV plan](#). The expected values are from [1] while actual outputs are the result of running the IP Simulator. Table 8 shows the outputs considering the initial condition corresponding to table 7.

Table 7: TC-IP-Out-1

Initial Conditions				Inputs					Time
x_i	dx_i	θ_i	$\dot{\theta}_i$	m_p	m_c	l_p	f	b	Time
0.2	0	0.3	0	0.2	1.5	1.3	50	0.1	3.000

Table 8: The results of TC-IP-Out-1

Expected Output		Actual Output	
x_i	θ_i	x_i	θ_i
109.000	0.6984	109.1997	0.7002

Table 9 investigates TC-IP-Out-2 of table 3 in VnV plan while table 10 displays the outputs.

Table 9: TC-IP-Out-2

Initial Conditions				Inputs					Time
x_i	dx_i	θ_i	$\dot{\theta}_i$	m_p	m_c	l_p	f	b	Time
0	0	0.1	0	2.5	0.1	0.3	20	0	3.000

Table 10: The results of TC-IP-Out-2

Expected Output		Actual Output	
x_i	θ_i	x_i	θ_i
21.6257	0.4034	21.5857	0.3934

Table 12 has the output for environment has been set up by table 11 values.

Table 11: TC-IP-Out-3

Initial Conditions				Inputs					Time
x_i	dx_i	θ_i	$\dot{\theta}_i$	m_p	m_c	l_p	f	b	Time
0	0.2	0.1	0	0.5	4	1.3	2	0.2	7.000

The initial values of parameters in table 13 will produce the outputs are shown in table 14.

4 Nonfunctional Requirements Evaluation

This section describes the tests that have been done to investigate the non-functional requirements including portability, usability, and correctness. All the test run manually as mentioned in section 5.2 of [VnV plan](#).

4.1 Portability

According to section 5.2.1 of [VnV plan](#), to verify the portability of the tool, the software with the tests described in VnV plan have been performed manually in different OS, including Linux, Windows, and MacOS.

The software has been developed and tested in Python 3.10.2 in Microsoft

Table 12: The results of TC-IP-Out-3

Expected Output		Actual Output	
x_i	θ_i	x_i	θ_i
11.0721	0.9152	11.1207	0.9282

Table 13: Test Case-2

Initial Conditions				Inputs					Time
x_i	dx_i	θ_i	$\dot{\theta}_i$	m_p	m_c	l_p	f	b	Time
0	0	0.3	0	0.7	0.4	1	0	0	4.000

Windows 10 Enterprise and all the test passed.

It is also run on Ubuntu 21.04 and passed the tests, however, we could not run the program on MacOS because of lacking the system resources.

4.2 Usability

To evaluate the usability of the tool, we use the [2], as discussed in section 5.2.2 of [VnV plan](#). The system has been tested against the usability test and survey by different people with varying levels of experience with inverted pendulum systems. The tester team includes an undergraduate Physics student, a high school student, and the domain expert who has a Ph.D. in Physics. At first, the software, the tests described in VnV plan, and the questionnaire were delivered to the tester team. They run the list of tests manually and tried any input test considering input criteria. Then they completed the questionnaire.

This survey considers six different factors (attractiveness, perspicuity, efficiency, dependability, stimulation, and novelty) through twenty-six randomized questions to minimize answer tendencies.

Using the data analysis tool which is prepared by [2], the feedback from the survey is analysed and the results are shown below. The tool does not produce an overall score for the user experience.

Figure 5 display the mean of values assigned to each question by all the tester team. It is worth mentioning that the green up arrow besides the mean values

Table 14: Test Case-2

Expected Output		Actual Output	
x_i	θ_i	x_i	θ_i
0.0000	0.0485	0.0001	0.0453

means all the values are greater than mean value in benchmarks and figure 7 shows the mean value per each of items for usability test of IP Simulator.

Item	Mean	Variance	Std. Dev.	No.	Left	Right	Scale
1	2.0	0.0	0.0	3	annoying	enjoyable	Attractiveness
2	2.3	0.3	0.6	3	not understandable	understandable	Perspicuity
3	2.0	0.0	0.0	3	creative	dull	Novelty
4	2.7	0.3	0.6	3	easy to learn	difficult to learn	Perspicuity
5	1.7	1.3	1.2	3	valuable	inferior	Stimulation
6	2.3	0.3	0.6	3	boring	exciting	Stimulation
7	2.0	1.0	1.0	3	not interesting	interesting	Stimulation
8	2.7	0.3	0.6	3	unpredictable	predictable	Dependability
9	2.0	1.0	1.0	3	fast	slow	Efficiency
10	1.7	0.3	0.6	3	inventive	conventional	Novelty
11	2.3	1.3	1.2	3	obstructive	supportive	Dependability
12	2.7	0.3	0.6	3	good	bad	Attractiveness
13	2.3	0.3	0.6	3	complicated	easy	Perspicuity
14	2.0	0.0	0.0	3	unlikable	pleasing	Attractiveness
15	2.3	0.3	0.6	3	usual	leading edge	Novelty
16	2.3	0.3	0.6	3	unpleasant	pleasant	Attractiveness
17	2.0	1.0	1.0	3	secure	not secure	Dependability
18	2.3	0.3	0.6	3	motivating	demotivating	Stimulation
19	2.3	0.3	0.6	3	meets expectations	does not meet expectations	Dependability
20	1.7	0.3	0.6	3	inefficient	efficient	Efficiency
21	2.0	1.0	1.0	3	clear	confusing	Perspicuity
22	2.7	0.3	0.6	3	impractical	practical	Efficiency
23	2.3	0.3	0.6	3	organized	cluttered	Efficiency
24	2.3	0.3	0.6	3	attractive	unattractive	Attractiveness
25	2.0	0.0	0.0	3	friendly	unfriendly	Attractiveness
26	1.3	0.3	0.6	3	conservative	innovative	Novelty

Figure 5: The mean values and variances of all the responses to each question

Figure 6 shows the scaled value for all six factors and the tester team.

Scale means per person					
Attractiveness	Perspicuity	Efficiency	Dependability	Stimulation	Novelty
2.33	2.50	1.75	2.75	2.00	1.75
2.33	2.25	2.50	1.75	2.25	1.75
2.00	2.25	2.25	2.50	2.00	2.00

Figure 6: Scale Means value per person

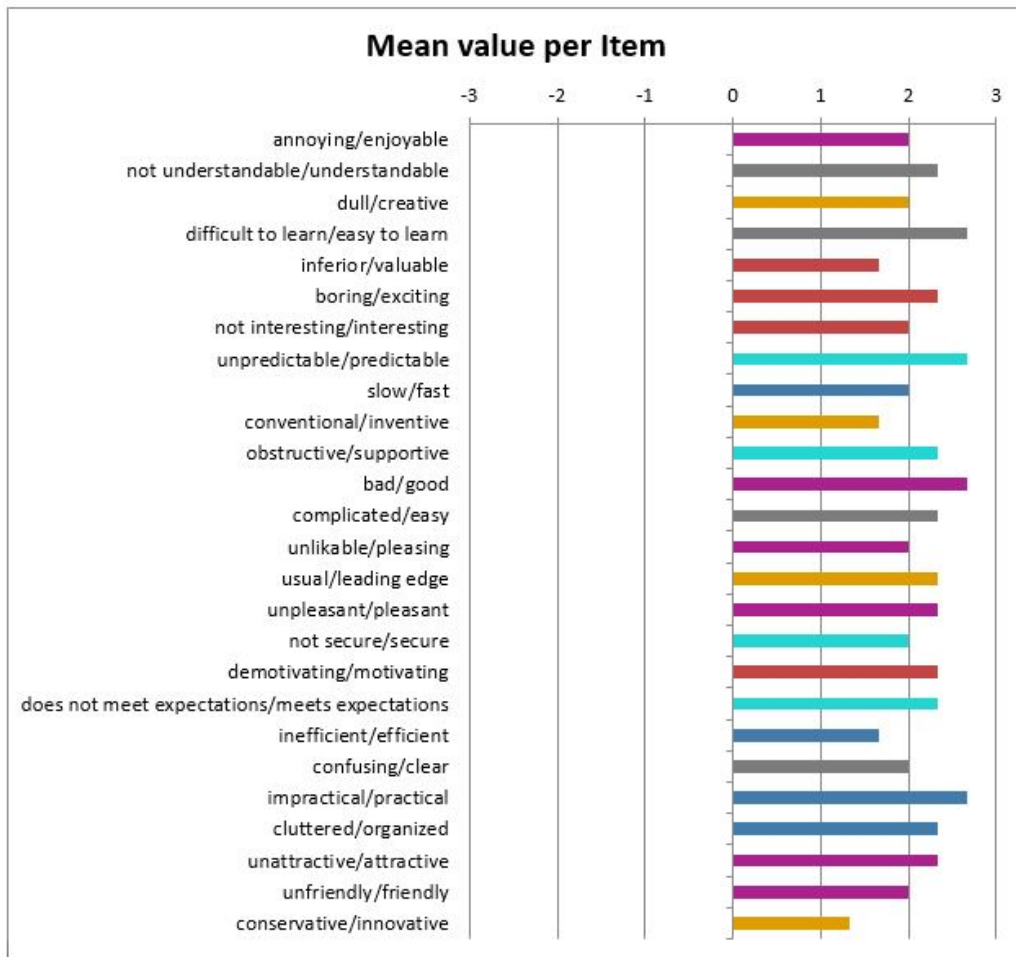


Figure 7: Mean value per item

As shown in figure 8, all the mean values for factors measured in this survey are higher than the mean value for benchmarks.

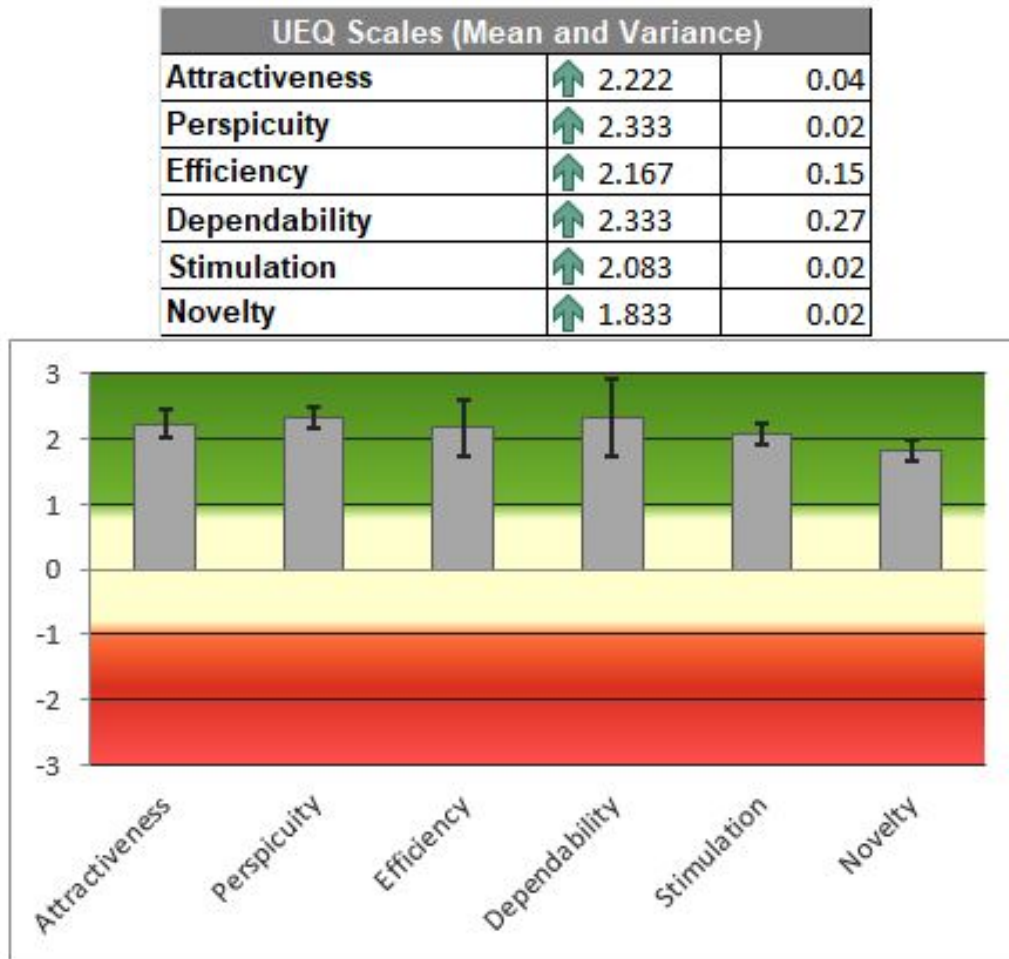


Figure 8: Mean values and variance for each factor

4.3 Accuracy

As described in section 5.2.3 [VnV plan](#), to verify the accuracy of the software, we use the pseudo-oracle and compare the results of running IP Simulator software with the results of [1] in the similar situation, with the same input values and the same initial conditions.

Then we calculate the relative error and the error less than 0.01 is good enough to accept the accuracy of the program.

the process has been done for different inputs and all the relative errors are in the acceptance range. Table 15 sets up the test with normal input and a zero value for function and table 16 shows the maximum relative error for all the calculated values during the 20 seconds of simulation. Figure 9 also shows the graphs of running two programs, the left picture shows the result of the IP Simulator in Python and the right one shows the results of running [1] in Matlab.

Table 15: TC-normal values with zero function

Initial Conditions				Inputs					Time
x_i	dx_i	θi	$\dot{\theta}_i$	m_p	m_c	l_p	f	b	duration
0	0	0.1	0	0.3	0.4	1	0	0.1	[0, 20]

Table 16: Relative error for TC-normal values with zero function

Relative Error			
x_i	dx_i	θi	$\dot{\theta}_i$
1.0000e-02	2.2130e-03	3.7380e-02	1.1755e-04

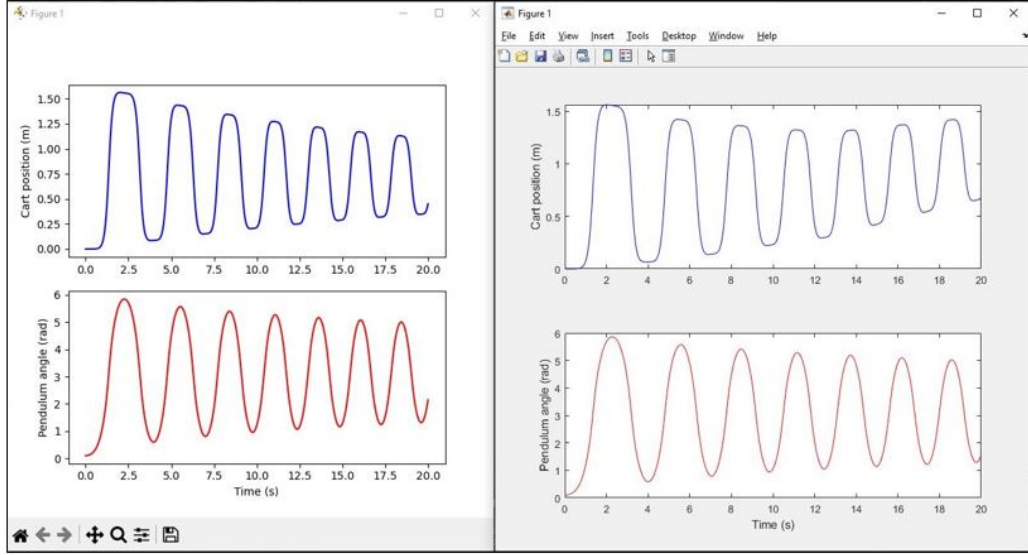


Figure 9: TC-normal values with zero function

The TC-normal values with zero function are repeated with the smaller time range, the previous test run during $[0, 20]$, while this test runs from 0 to 5 seconds. Table 17 shows the parameter values, table 18 shows the relative error, and figure 10 is a picture of running IP Simulator and [1].

Table 17: TC-normal values with zero function in a shorter time

Initial Conditions				Inputs					Time
x_i	dx_i	θ_i	$\dot{\theta}_i$	m_p	m_c	l_p	f	b	duration
0	0	0.1	0	0.3	0.4	1	0	0.1	$[0, 5]$

Table 18: Relative error for TC-normal values with zero function in a shorter time

Relative Error			
x_i	dx_i	θ_i	$\dot{\theta}_i$
0.2700e-01	1.34130e-02	4.7560e-02	0.3215e-03

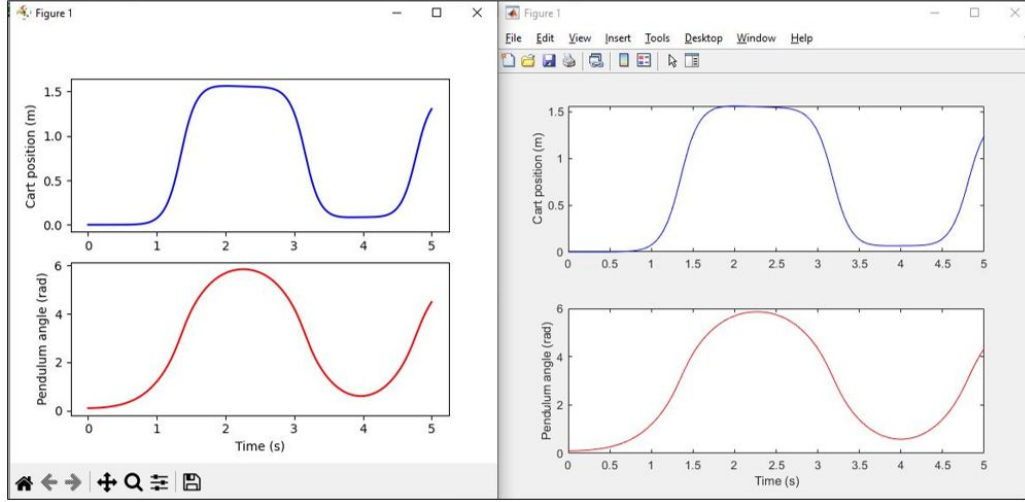


Figure 10: TC-normal values with zero function in a shorter time

In this test, the f that is the function exerted on the cart is not zero and it has the value of 20 and the time duration exceeded. Table 19 and table 20 show the set up and maximum relative error of this test. Figure 11 shows the graphs for this test.

Table 19: TC-positive normal values in longer time

Initial Conditions				Inputs					Time
x_i	dx_i	θ_i	$\dot{\theta}_i$	m_p	m_c	l_p	f	b	duration
0	0	0.1	0	0.3	0.4	1.5	20	0.1	[0, 30]

Table 20: Relative Error for TC-positive normal values in longer time

Relative Error			
x_i	dx_i	θ_i	$\dot{\theta}_i$
3.0000e-03	1.0802e-02	1.6030e-02	3.9120e-02

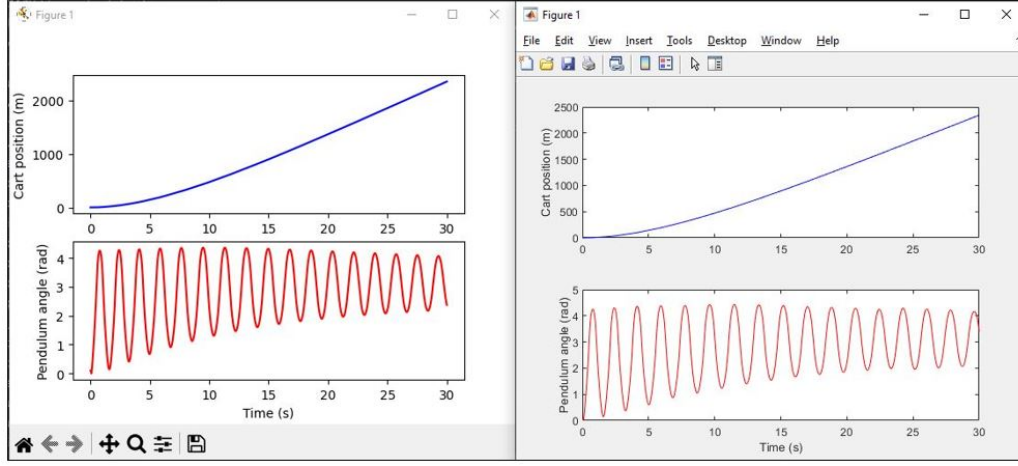


Figure 11: TC-positive normal values in a longer time

Table 21 shows the repetition of TC-positive normal values in a longer time test but in a shorter duration and table 22 shows the maximum relative error for this test while figure 12 is devoted to the graphs.

Table 21: TC-positive normal values in a shorter time

Initial Conditions				Inputs					Time
x_i	dx_i	θ_i	$\dot{\theta}_i$	m_p	m_c	l_p	f	b	duration
0	0	0.1	0	0.3	0.4	1.5	20	0.1	[0, 20]

Table 22: Relative Error for TC-positive normal values in shorter time

Relative Error			
x_i	dx_i	θ_i	$\dot{\theta}_i$
2.0000e-01	3.4484e-02	1.0000e-01	2.5804e-03

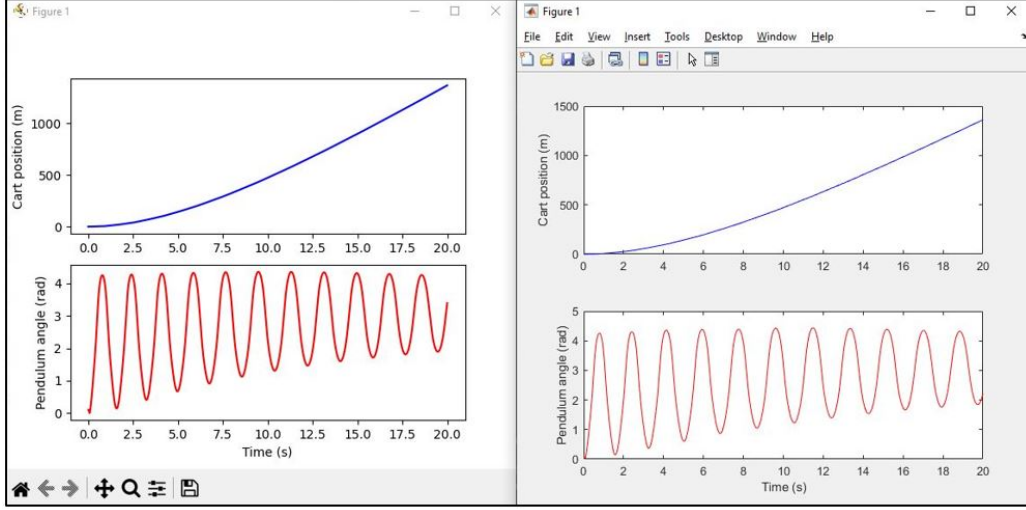


Figure 12: TC-positive normal values in shorter time

And in this test, the parameters for the cart and pendulum have changed and like other tests, table 23 shows the input parameters and table 24 shows the result of this test. Finally, the figure 13 shows the graphs of results.

Table 23: TC-normal positive values with different initial conditions

Initial Conditions				Inputs					Time
x_i	dx_i	θ_i	$\dot{\theta}_i$	m_p	m_c	l_p	f	b	duration
2	0	1.5	0	0.1	0.1	0.5	10	0.1	[0,10]

Table 24: Relative Error for TC-normal positive values with different initial conditions

Relative Error			
x_i	dx_i	θ_i	$\dot{\theta}_i$
2.3750e-03	3.2971e-02	2.5346e-01	1.6598e-02

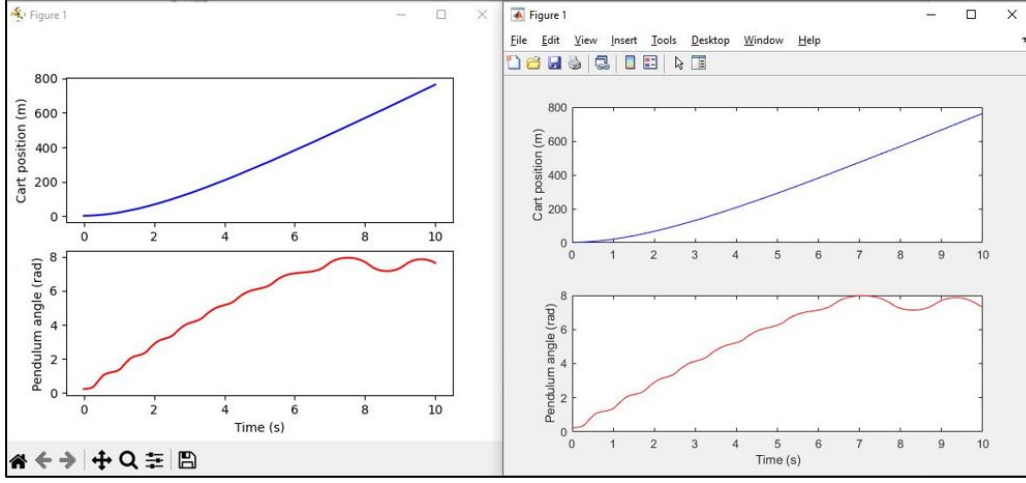


Figure 13: TC-normal positive values with different initial conditions

5 Comparison to Existing Implementation

The existing implementation of inverted pendulum that is used to compare the result of IP Simulator, is [1]. This program is implemented in Matlab and the details of comparison the results is discussed in section 4.3.

6 Unit Testing

6.1 Tests for Functional Requirements

We use [Pytest](#) framework for unit, functional, and integration automated testing.

6.1.1 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](#).

6.1.2 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 constraints 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.

6.1.3 Motion ODE Module

[OdeM.py](#) implements Motion ODE Module, and defines the motion equation for the cart and pendulum with their specification.

6.1.4 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](#) and using *Scipy* library in Python is solved.

To verify motion ODE and ODE solver modules, we compare the result of running [1] with the output of these modules in section [4.3](#).

6.1.5 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 constraints on the generated outputs and `write_output_to_file` function that gives the output file and the output that 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 , writes this array to the

specified file.

6.1.6 Plotting Module

This module gives the time and the output of ODE solver module and shows the results as a graph in [plotM.py](#)

6.1.7 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 and with a lot of different test cases in input file.

```
E:\courses\winter\CAS741\project\code>pytest -p no:warnings
platform win32 -- Python 3.10.2, pytest-7.2.2, pluggy-1.0.0
rootdir: E:\courses\winter\CAS741\project\code
collected 9 items

test_checkFunction.py .
test_constantM.py .
test_inputM.py .....
test_mainM.py .

----- test session starts -----
----- 9 passed in 6.59s -----
```

Figure 14: The result of running unit tests

It is worth mentioning that IP Simulator accepts float number of function of time as the f , external applies to the cart. The function can be defined using *lambda* in python. *CheckFunction* module is responsible to dealing with the parameter user inputs for the function variable. [Test_checkfunction](#) is the unit test for verifying that this module works properly.

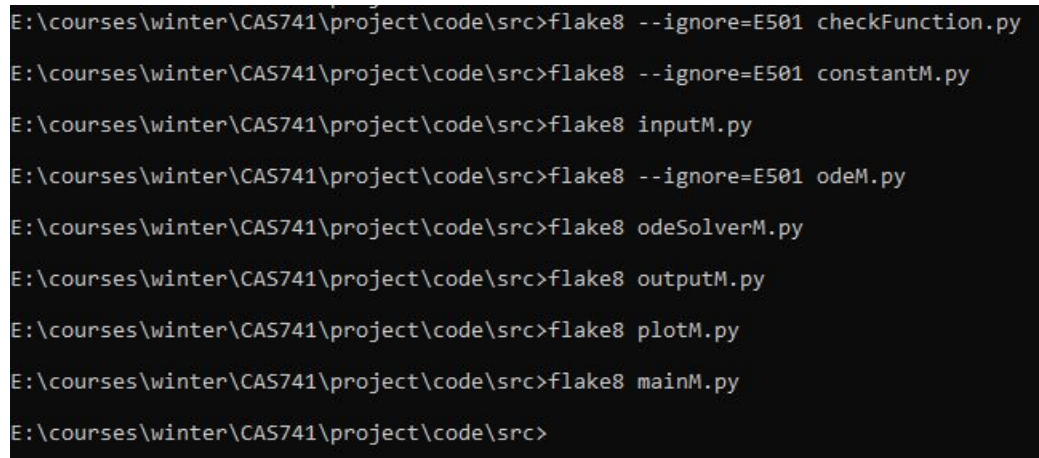
7 Changes Due to Testing

No major changes were implemented due to testing, and bugs in the IP Simulator software were continuously addressed throughout the course of the testing.

8 Automated Testing

We use [Pytest](#) framework for unit, functional, and integration automated testing, as discussed in section 6, [git](#), for tracking changes during software development, and [Flake8](#) to check for errors, enforces coding standards, and identifies code complexity issues.

Figure 15 shows the result of running flake8 on all modules. I decided to ignore *E501* which is related to the length of the lines which is longer than 79 characters.



```
E:\courses\winter\CAS741\project\code\src>flake8 --ignore=E501 checkFunction.py
E:\courses\winter\CAS741\project\code\src>flake8 --ignore=E501 constantM.py
E:\courses\winter\CAS741\project\code\src>flake8 inputM.py
E:\courses\winter\CAS741\project\code\src>flake8 --ignore=E501 odeM.py
E:\courses\winter\CAS741\project\code\src>flake8 odeSolverM.py
E:\courses\winter\CAS741\project\code\src>flake8 outputM.py
E:\courses\winter\CAS741\project\code\src>flake8 plotM.py
E:\courses\winter\CAS741\project\code\src>flake8 mainM.py
E:\courses\winter\CAS741\project\code\src>
```

Figure 15: Running flake8 on the modules

9 Trace to Requirements

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed.

Table 25 shows the dependencies between the test cases and the requirements. Requirements can be found in [SRS](#).

Requirements	Test section
R1	section 3
R2	section 3
R3	section 3
R4	section 3
NFR1	section 4.1
NFR2	section 4.2
NFR3	section 4.3

Table 25: Traceability Between Test Cases and Requirements

10 Trace to Modules

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Table 26 shows the dependencies between the test cases and the modules.

Requirements	Test section
Input Parameters Module	3, 6.1.2
Constants Parameters Module	6.1.1
Motion ODE Module	6.1.7
ODE Solver Module	6.1.7
Output Module	3, 6.1.7
Plotting Module	6.1.7
IP Control Module	6.1.7

Table 26: Traceability Between Test Cases and modules

11 Code Coverage Metrics

The following code coverage of each module is measured by Coverage. Figure 16 shows the code coverage.

Name	Stmts	Miss	Cover
checkFunction.py	13	3	77%
constantM.py	12	0	100%
inputM.py	43	4	91%
odeM.py	17	0	100%
odeSolverM.py	9	0	100%
outputM.py	18	9	50%
plotM.py	11	0	100%
test_checkFunction.py	10	0	100%
test_constantM.py	10	0	100%
test_inputM.py	45	0	100%
test_mainM.py	30	0	100%
TOTAL	218	16	93%

Figure 16: Code Coverage

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/>.