System Verification and Validation Plan for Double Pendulum

Zhi Zhang

December 11, 2019

1 Revision History

Date	Version	Notes
Oct.30	1.0	Initial Draft
Dec.10	2.0	Fixed Issues

Contents

List of Tables

List of Figures

2 Symbols, Abbreviations and Acronyms

symbol	description
VnV	Verification and Validation
SRS	Software Requirements Specification
FR	Functional Requirements
NFR	Nonfunctional Requirements

This document provides an overview of the Verification and Validation(VnV) plan for Double Pendulum. The general information is introduced in section General Information. SRS, design, implementation verification plan and software validation plan is introduced in section Plan. In section System Test Description, tests for both FR and NFR of Double Pendulum will be discussed, with the traceability between test cases and requirements.

3 General Information

3.1 Summary

The software to be tested is Double Pendulum. This software allows user to input the initial conditions of double pendulum, calculates the equations for the movements of the two pendulums, and then output the result to a text file.

3.2 Objectives

The objective of the VnV plan is to verify the FR and NFR, as found in the SRS titled Double Pendulum, has been met. The most important requirement is the correctness of the software, the goal is to build confidence in the software correctness by comparing the outputs of the software with the Double Pendulum tool from myPhysicsLab.com Neumann. Adequate usability also needs to be achieved since the software is designed for any user who is interested in the motion of a double pendulum.

3.3 Relevant Documentation

The relevant documentations are:

- SRS Zhang (a)
- Unit VnV Plan Zhang (c)
- System VnV Report Zhang (b)
- Unit Vnv Report Zhang (d)

[You should cite these documents, including the GitHub url for their location. You should also mention all of the documents that you will write for this course, even though they are not all complete. —SS] [done —Author]

4 Plan

4.1 Verification and Validation Team

- Zhi Zhang
- Dr. Spencer Smith smiths and Bo Cao caobo1994 review the whole project, including all documents and codes.
- Deema Alomair deemaalmair1 reviews the SRS.
- Ao Dong Ao99 reviews MG and MIS.
- Sharon Wu sharyuwu reviews the VnV Plan.

[You should also list your colleagues in the class, because they will be helping you with the VnV efforts. —SS][Done. —Author]

4.2 SRS Verification Plan

The SRS of Double Pendulum will be verified in the following ways:(Michalski)

1. Feedback: Classmates, including all primary and secondary reviewers listed above, shall provide feedback on GitHub. They shall read the document and provide insight on how to improve the documents and the project.
2. Initial Review: The document shall be manually reviewed by the author using the SRS checklist upon its initial creation, as found in the CAS741 GitLab repository (Smith).
3. Second Review: The document shall be manually reviewed by the author using the SRS checklist after VnV completion, as found in the CAS741 GitLab repository (Smith).

4. Final Review: The document shall be manually reviewed by the author using the SRS checklist after MG and MIS development, as found in the CAS741 repository (Smith).

[The review would be more useful if it was more structured. Maybe you have some ideas for a way to structure the review? Maybe there can be a task-based review? —SS] [I think Peter's SRS Verification Plan is very detailed, so I refered to his plan. —Author]

4.3 Design Verification Plan

The design will be verified by ensuring that functional requirements and NFRs are tested, as listed in Section 3.2. The system functional requirements will be tested as outlined in Section 5.1, and the NFRs will be tested as outlined in Section 5.2. [As for the SRS, you can flesh this out in greater detail. Peter has a nice design verification plan in his document. —SS]

[Plans for design verification —SS] [Done. —Author]

4.4 Implementation Verification Plan

The implementation verification plan will include the followings:

- Code walkthroughs: both the developer and the domain expert will inspect the code to ensure all the functional requirements of the SRS are met and all modules of MIS document are implemented. A rubber duck debugging method will be followed. Any defects shall be immediately fixed. [How is this going to be done? How will the walkthrough be structured? What reference are you going to follow? —SS] [Updated. —Author]

- Unit testing: all modules are to be unit tested to ensure correctness, dynamic unit test will be carried out, more details of unit testing can be found in the Unit Testing Plan. [What technology will you use for unit testing? —SS] [Dynamic unit test. —Author]

- Dynamic testing: the software will be executed and all its functions will be tested manually by test team.

4.5 Software Validation Plan

Double Pendulum does not have a validation step. Validation is the process of comparing the outputs of models to experimental values. Double Pendulum only illustrates the motion of the pendulum in ideal condition ignoring air drag and other possible features that may affect the motion. So there is to need to validate the outputs of Double Pendulum. [I agree, but you should say why it is N/A. —SS] [Explained above. —Author]

5 System Test Description

5.1 Tests for Functional Requirements

There functional requirements are described in section 5.1 of the SRS. [You can cross-reference between documents in LaTeX —SS] [Copying the requirements here is a maintenance nightmare. —SS] [Done. —Author] Double Pendulum shall verify that the inputs are valid, shall guarantee the calculation is correct and the outputs are in the correct form.

Detailed test plan for the five functional requirements will be covered in the next five subsections.

5.1.1 Input Verification

Invalid Inputs

1. Non-positive Mass

   Control: Automatic

   Initial State: Double Pendulum is started and running

   Input:

   - $m_1 = 0$, $m_2 =$ -10, $L_1 = 10$, $L_2 = 10$, $\theta_1 = 10$, $\theta_2 = 10$, $g = 9.8$
   - $m_1 = 0$, $m_2 = 0$, $L_1 = 10$, $L_2 = 10$, $\theta_1 = 10$, $\theta_2 = 10$, $g = 9.8$
   - $m_1 =$ -10, $m_2 = 0$, $L_1 = 10$, $L_2 = 10$, $\theta_1 = 10$, $\theta_2 = 10$, $g = 9.8$
   - $m_1 =$ -10, $m_2 =$ -10, $L_1 = 10$, $L_2 = 10$, $\theta_1 = 10$, $\theta_2 = 10$, $g = 9.8$

   Output: Error message of “Please input positive value for the mass”. [Quotes are done using “quote” —SS] [Fixed —Author]

Test Case Derivation: Check if the NEGATIVE_MASS exception is raised and error message pops up.

How test will be performed: Unit Test framework will feed function with the aforementioned input, verify that the correct exception is raised and the error message is displayed. [No, this test should be automated. Rather than waiting for an error message, you should have unit tests that verify that the correct exception is raised. —SS] [updated. —Author]

2. Non-positive Length of Rods

   Control: Manual

   Initial State: Double Pendulum is started and running

   Input:

   - $m_1 = 10$, $m_2 = 20$, $L_1 = 0$, $L_2 = 0$, $\theta_1 = 10$, $\theta_2 = 10$, $g = 9.8$
   - $m_1 = 10$, $m_2 = 20$, $L_1 = 0$, $L_2 = $ -10, $\theta_1 = 10$, $\theta_2 = 10$, $g = 9.8$
   - $m_1 = 10$, $m_2 = 20$, $L_1 = $ -10, $L_2 = 0$, $\theta_1 = 10$, $\theta_2 = 10$, $g = 9.8$
   - $m_1 = 10$, $m_2 = 20$, $L_1 = $ -10, $L_2 = $ -10, $\theta_1 = 10$, $\theta_2 = 10$, $g = 9.8$

   [You could use tables to make this document less repetitive. Define the base test case and then give the delta to define the other test cases. —SS]

   Output: An error message of“”Please input positive value for the the [proof read —SS][fixed. —Author] length of rods”.

   Test Case Derivation: Check if the error message pops up.

   How test will be performed: Manually input the above the Input data to the software by test team.

3. Zero Starting Angle

   Control: Manual

   Initial State: Double Pendulum is started and running

   Input:

   - $m_1 = 10$, $m_2 = 20$, $L_1 = 10$, $L_2 = 20$, $\theta_1 = 0$, $\theta_2 = 0$, $g = 9.8$

Output: An error message of "Please input at least one non-zero value for the starting angles".

Test Case Derivation: Check if the error message pops up.

How test will be performed: Manually input the above the Input data to the software by test team.

Valid Inputs

1. Valid Values

   Control: Manual

   Initial State: Double Pendulum is started and running

   Input: $m_1 = 10$ $m_2 = 10$ $L_1 = 10$ $L_2 = 10$ $\theta_1 = 10$ $\theta_2 = 10$ $g = 9.8$

   - $m_1 = 10$, $m_2 = 20$, $L_1 = 10$, $L_2 = 10$, $\theta_1 = 0$, $\theta_2 = 10$, $g = 9.8$
   - $m_1 = 10$, $m_2 = 20$, $L_1 = 10$, $L_2 = 20$, $\theta_1 = 10$, $\theta_2 = 0$, $g = 9.8$

   Output: The output file is generated and the graphs are displayed.

   Test Case Derivation: Check if there is output file and the graphs.

   How test will be performed: Manually input the above the Input data to the software by test team.

5.1.2 Outputs Verification

This section covers requirements R.3, R.4 and R.5.

Outputs Correctness Tests

1. Correct Output

   Type: Dynamic, Manual

   Initial State: Double Pendulum takes valid inputs and generate an output file

   Input/Condition: Compare a sequence of outputs generated over the same time steps between Double Pendulum and Matlab ode23 solver with the same inputs.

Output/Result: Check if the relative error between each value at each time step is within 0.2.

How test will be performed: Test team will manually input the same data into myPhysicsLab.com Neumann and compare the graphs.

2. Graph Generated

   Type: Dynamic, Manual

   Initial State: Double Pendulum takes valid inputs and the output file is generated

   Input/Condition: -

   Output/Result: Graphs of $\theta_1(t)$ and $\theta_2(t)$ displayed on the screen of the software

   How test will be performed: Test team will manually press the Graph button and check if the graph is displayed [Aren't you also checking to see if the graph is correct? —SS]

[You are missing the most important tests. You want tests that the output is correct. Not in terms of the graphs (although you want that too), but in terms of the output calculations. The calculations of theta over time. You need to say where you are going to get the "correct" solution and how you are going to compare it. A good starting point would be to compare a sequence of outputs generated over the same times steps between your program and another program that does the same thing. You can then compare the relative error between each value at each time step. You can then summarize the entire test with a reasonable norm, like the infinity norm, or the Euclidean norm. You should have a variety of tests with more and more extreme inputs. —SS]

[It would be great to see a comparison between your output and the output of a validated solver. I haven't asked him, but I'm confident that Bo will help you to run Dr. Nedialkov's validated ODE solver. It would be great to see how quickly your answer gets outside of the correct answer. I think it will happen quicker than you think. :-) —SS]

5.2 Tests for Nonfunctional Requirements

There are five nonfunctional requirements of the system as specified in SRS.

5.2.1 Correctness and Verifiability

The correctness test covers the NFR 6, and the verifiability test covers the NFR 7. Both tests are to ensure that the software meets the SRS, and they are been tested in the section 5.1.2.

5.2.2 Maintainability

The maintainability test covers the NFR 9.

Maintainability Testing
Type: Manual, Nonfunctional, Static
Initial State: N/A
Input/Condition: Existing Double Pendulum system
Output/Result: N/A
How test will be performed: Test team will walkthrough the source code of Double Pendulum, verify if each module performs only one function, which ensures high cohesion and low coupling between modules. [You need to be more specific. How would someone actually do this test? —SS] [by code walkthrough —Author]

5.2.3 Re-usability

The re-usability test covers the NFR 8.

Re-usability Testing
Type: Manual, Nonfunctional
Initial State: N/A
Input/Condition: Existing Double Pendulum system
Output/Result: N/A
How test will be performed: Test team will perform code walk through to ensure that all codes are modularized. [This isn't a test, but it would be a good target for your code walkthrough exercise. You need to flesh out the details though. What would this walkthrough look like. How is modularity assessed? Also it is modularity that is being measured, not technically reusability. —SS]

5.2.4 Portability

The portability test covers the NFR 10.

Portability Testing

1. Portability on Windows System Type: Manual, Dynamic

   Initial State: Double Pendulum has been successfully installed on a Windows system

   Input/Condition: Perform all the functional tests listed in section 5.1.

   Output/Result: The software performs all functions successfully.

   How test will be performed: The test will be performed by test team manually.

2. Portability on MacOS system

   Type: Manual, Dynamic

   Initial State: Double Pendulum has been successfully installed on a MacOS system

   Input/Condition: Perform all the functional tests listed in section 5.1 .

   Output/Result: The software performs all functions successfully.

   How test will be performed: The test will be performed by test team manually.

[These tests are also not specific enough. A simpler approach would be to run your regression tests on both platforms and report the results. —SS]

5.3 Traceability Between Test Cases and Requirements

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
5.1.1	x	x								
5.1.2			x	x	x					
5.2.1				x	x	x	x			
5.2.2									x	
5.2.3								x		
5.2.4										x

Table 1: Traceability Between Test Cases and Requirements

References

Peter Michalski. System verification and validation plan for lattice boltzmann solver. URL https://github.com/peter-michalski/LatticeBoltzmannSolvers/blob/master/docs/VnVPlan/SystVnVPlan/SystVnVPlan.tex.

Erik Neumann. Double pendulum. URL https://www.myphysicslab.com/pendulum/double-pendulum-en.html.

Smith. SRS Checklist. URL https://gitlab.cas.mcmaster.ca/smiths/cas741/blob/master/BlankProjectTemplate/docs/SRS/SRS-Checklist.pdf.

Zhi Zhang. Software requirements specification, a. URL https://github.com/best-zhang-zhi/CAS741Project/blob/master/Double%20Pendulum/docs/SRS/SRS.pdf.

Zhi Zhang. Systemvnvreport, b. URL https://github.com/best-zhang-zhi/CAS741Project/blob/master/Double%20Pendulum/docs/VnVReport/SystVnVReport/SystVnVReport.pdf.

Zhi Zhang. Unitvnv, c. URL https://github.com/best-zhang-zhi/CAS741Project/blob/master/Double%20Pendulum/docs/VnVPlan/UnitVnVPlan/UnitVnVPlan.pdf.

Zhi Zhang. Unitvnvrport, d. URL https://github.com/best-zhang-zhi/CAS741Project/blob/master/Double%20Pendulum/docs/VnVReport/UnitVnVReport/UnitVnVReport.pdf.

[Your project is fairly straightforward Zhi. If you want to aim for a maximum mark in this course, you'll need to do more than just implement your project. One place where you could add some extra challenge to the project is to consider assessing usability. I'm open to discussion if there is something else you would like to add instead. Another idea was mentioned above - the idea of comparing to a validated ODE solver. Other interesting options would be to use the method of manufactured solutions for verification. —SS]