Unit Verification and Validation Plan for Double Pendulum

Zhi Zhang

December 18, 2019

1 Revision History

Date	Version	Notes
Dec.3, 2019	1.0	Initial Draft

Contents

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
VnV	Verification and Validation
DP	Double Pendulum

This document provides an overview of the unit Verification and Validation(VnV) for Double Pendulum. It lays out the purpose, methods, and test cases for the VnV procedure.

3 General Information

3.1 Purpose

The purpose of this document is to describe the unit VnV plan of the software Double Pendulum(DP). This software gets input parameters from users to initialize the double pendulum system, then predict the motion of the system using ODE initial value problem solvers, output the result and displays the graph of the motions.

3.2 Scope

The scope of DP is limited to the generate the output files and plot the diagram of change in the angles for two rods. The scope of the test plan is described below:

- Double Pendulum will be written in Python.
- The proposed test plan is focusing on system and unit testing to verify the functional and nonfunctional requirements of DP.

The DP is limited to the user initialized inputs and output of the DP will be a text file and the generated graphs of the angular position of two pendulums.

The modules that might be tested for under Unit VnV plan that are traced to the SRS requirements are:

M2: Input Parameters Module

M3: Output Format Module

M4: Acceleration Equations Module

M5: Control Module

M6: Sequence Data Structure Module

M7: ODE Solver Module

M8: Plotting Module

4 Plan

4.1 Verification and Validation Team

- Zhi Zhang

4.2 Automated Testing and Verification Tools

Unit-based scripts can be created for testing the modules in DP and be tested automatically with scripts written in pytest. The test scripts using pytest will cycle through an array of data and check an assert statement.

4.3 Non-Testing Based Verification

The non-testing based verification will involve a code inspection by myself during final submission. Dr. Smith will run the software to ensure the functional requirements are implemented.

5 Unit Test Description

The modules that are being unit tested are specified in the MIS(https://github.com/best-zhang-zhi/CAS741Project/blob/master/Double%20Pendulum/docs/Design/MIS/MIS.pdf). The Unit VnV plan ensures that each module is behaving accurately and that each module is satisfying the SRS(https://github.com/karolserkis/CAS-741-Pendula/blob/master/docs/SRS/SRS.pdf) requirements.

5.1 Tests for Functional Requirements

5.1.1 Input Parameters Module

The following tests are created to ensure that the input parameters module is validate the user input data. This first step is critical as all of the other modules rely on these parameters.

1. test-Input

   Type: Automatic

   Initial State: DP is opened

   Input: DP import the input file

   Output: assert=True

   Test Case Derivation: The data inputted by the user should match the state variable from InParams.

   How test will be performed: Random number values ranging from 1 to 100 will be inputed in the environment. This value should be exactly equal to the state variables of Input. An assert statement will return True if these are equal.

2. test-Input

   Type: Automatic

   Initial State: DP is opened

   Input: DP import the input file with invalid input data

   Output: assert=True

   Test Case Derivation: The system should detect the invalid data, and raise exception.

   How test will be performed: Invalid inputs will be inputed in the environment. The system should raise the correct exception.

5.1.2 Acceleration Equations Module

1. test-Acceleration

   Type: Manual

   Initial State: DP is opened

   Input: Acceleration module is called

   Output: equations of $\theta_1(t)$ and $\theta_2(t)$

Test Case Derivation: Output should match the equations calculated by hand

How test will be performed: Different set of valid input will be passed to acceleration module, and the test team will check if the generated equations match the hand calculated equations.

5.1.3 ODE Solver Module

1. test-ODE

   Type: Automatic

   Initial State: DP is opened

   Input: ODE module called

   Output: list of $\theta_1(t)$ and $\theta_2(t)$ data generated

   Test Case Derivation: θ_1 and θ_2 values should be generated, and will be compared with the results generated by MATLAB with the same input data.

   How test will be performed: Unit test will call the function with valid inputs, and check if the output data matches the result from MATLAB.

5.1.4 Output Module

1. test-Control

   Type: Manual

   Initial State: DP is opened

   Input: main Module is called

   Output: output.txt file generated

   Test Case Derivation: Output should contain a list of $\theta_1(t)$ and $\theta_2(t)$ data.

   How test will be performed: Unit test will call the function with valid inputs, and check if the output.txt file is generated in the project folder.

5.1.5 Plotting Module

1. test-Plot

   Type: Automatic

   Initial State: DP is opened

   Input: Plot module called

   Output: Graph of $\theta_1(t)$ and $\theta_2(t)$ displayed

   Test Case Derivation: The graph be the plots of θ_1 and θ_2 values

   How test will be performed: Unit test will call the function with the inputs, and check if the output files are generated.

5.2 Tests for Nonfunctional Requirements

Testing for nonfunctional requirements is not a part of unit test, it is introduced in System VnV Plan(https://github.com/best-zhang-zhi/CAS741Project/blob/master/Double%20Pendulum/docs/VnVPlan/SystVnVPlan/SystVnVPlan.pdf)

5.3 Traceability Between Test Cases and 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 1 shows the dependencies between the test cases and the requirements.

Test Cases	Modules
5.1.1	Input Parameters Module
5.1.2	Acceleration Equations Module
5.1.3	ODE Solver Module
5.1.4	Output Format Module
5.1.5	Plotting Module

Table 1: Traceability Matrix showing the connections between modules and tests

6 Appendix

6.1 Symbolic Parameters

There is no symbolic parameters for Double Pendulum.