CAS 741: Unit Verification and Validation Plan

Dynamical Systems: Multi-Pendulum Simulation

Karol Serkis
serkiskj@mcmaster.ca
GitHub: karolserkis

December 16, 2018

1 Revision History

Date	Version	Notes
December 5, 2018	1.0	First full draft for submission

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at CAS-741-Pendula SRS [give url —SS] [You should remove these comments, so that the new comments that are added are easier to see. —SS]

[Also add any additional symbols, abbreviations or acronyms —SS]

The symbols are listed in alphabetical order.

symbol	**description**
A	Assumption
DD	Data Definition
FT	Functional Test
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
MG	Module Guide
MIS	Module Interface Specification
NF	Non-Functional Requirement
R	Requirement
SRS	Software Requirements Specification
T	Test
VnV	Verification and Validation

2.1 Table of Units

Throughout this document SI (Système International d'Unités) is employedas [proof read —SS] the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

symbol	unit	SI
m	length	metre
kg	mass	kilogram
s	time	second
°	angle	degree

Table 1: Table of Units

Contents

List of Tables

The purpose of the document is to provide the Unit Verification and Validation Plan for testing the Multi-Pendulum Simulation software with respect to the requirements (see SRS document). Unit Verification and Validation plan consists of outlining test cases for particular units of a software's modules. These tests are created to ensure that the units satisfy the software's functional and nonfunctional requirements. The tests can be traced to a particular module. The module should be traced to a particular requirement.

3 General Information

This documents is an Unit VnV Plan for the Multi-Pendulum Simulation program. The directory for this project can be found at GitHub: /karolserkis/CAS-741-Pendula/

3.1 Purpose

The purpose of this document is to describe the unit VnV plan and requirements for the Multi-Pendulum Simulation program solution that only focuses on multi-pendulum simulations (double & triple pendula and beyond) and tracking the chaotic motion of the system. It will allow users to generate diagrams (e.g. Poincare mapping) and plot trajectories over time using two different ODE/DAE initial value problem solvers. In the case of a double pendulum you have a new system that is dynamic and chaotic and requires a set of coupled ordinary differential equation solvers. Once one introduces multiple pendula the system becomes chaotic and interesting to model and simulate.

3.2 Scope

The scope of the Multi-Pendulum Simulation program is limited to the generation of diagrams and plot trajectories that are possible to run and compute on a local system.

The scope of the test plan is described below:

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

The Multi-Pendulum Simulation is limited to the user initialized inputs and the output of the Multi-Pendulum Simulation will either plot trajectories over time, generate diagrams, like Poincare mapping and limit the user to a specific duration of the simulation, in order to allow diagrams and trajectory history to be saved. The user will be able to set a range of time and initialize the system.

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

M1: Hardware-Hiding Module (described in MG_Hardware)

M2: Multi-Pendulum Simulation Control Module (described in MG_Control)

M3: Multi-Pendulum Simulation GUI Module (described MG_GUI)

M4: User Input Parameters Module (described in MG_InputFormat)

M5: Lagrangian Module (described in MG_LA)

M6: Hamiltonian Module (described in MG_HA)

M7: Data Structure Module (described in MG_DataStruct)

M8: Generic GUI/Plot Module (described in MG_GUIplot)

M9: Generic Trajectory Simulation GUI Module (described in MG_SIM)

[You should leave the hardware hiding module out of scope. The control module is usually also left out of scope, since it is covered in the system vnv plan. —SS]

4 Plan

4.1 Test Team

The Verification and Validation team consists of:
Karol Serkis [Probably just you. :-) —SS]

4.2 Automated Testing Approach

Unit-based scripts can be created for testing the modules in Multi-Pendulum Simulation and be tested automatically with scripts written in pytest. See section on verification tools for more information.

4.3 Verification Tools

Python has a few open source unit testing frameworks, namely,
pytest: `https://docs.pytest.org`. Unit-based scripts can be created for testing the modules in Multi-Pendulum Simulation . The functions and classes that automate this process are detailed in the following link: `https://github.com/pytest-dev/pytest`.
[Thoughts on what tools to use, such as the following: unit testing framework, valgrind, static analyzer, make, continuous integration, test coverage tool, etc. —SS]

4.4 Non-Testing Based Verification

The non-testing based verfication [spell check —SS] will involve a code inspection by myself and Prof Smith during final submission. He will inspect each of the modules to guarentee [spell check —SS] that the physical equations have been implemented succesfully, [spell check —SS] verify that the software is designed to be maintainable and manageable and complete the two surveys in the appendix.
[I will be reviewing your final documentation (for the purpose of grading), but the scope of the review you are proposing for me is beyond the time I have available. —SS] [List any approaches like code inspection, code walkthrough, symbolic execution etc. Enter not applicable if that is the case. —SS]

5 Unit Test Description

The modules that are being unit tested are specificed in the MIS (`https://github.com/karolserkis/CAS-741-Pendula/blob/master/docs/`). The unit VnV plan ensure that each module is behaving accurately and that each module is satisfying the SRS requirements.

5.1 Tests for Functional Requirements

5.1.1 User Input Module

The following tests were created to ensure that the input parameters module is storing the correct values specified by the user. This first step is critical as all of the other modules rely on these quantities.

1. test-InParams

   Type: Automaic [spell check —SS]

   Initial State:

   Input: Input from Table of Units:

symbol	unit	SI
m	length	metre
kg	mass	kilogram
s	time	second
°	angle	degree

   Output: assert=True

Test Case Derivation: The environment variable inputted by the user should match the state variable meant to store the value.
Ex. units for length, mass and time cannot be negative.

How test will be performed: Integer numbers will be inputed in the environment. An assert statement will return true if these are equal.

[The output field should be the expected output, not an assert statement. I do not know how to do your test case. There should be enough information with a test case that someone could independently write it. You also aren't using the distinction between the environment variables and state variables correctly. —SS]

5.1.2 Multi-Pendulum Simulation Lagrangian Module Testing

The following tests were created to ensure that the input parameters module is storing the correct values specified by the user. This first step is critical as all of the other modules rely on these quantities.

1. test-Lagrangian

   Type: Automaic [spell check —SS]

   Initial State:

   Input: Input from Table of Units:

symbol	**unit**	**SI**
m	length	metre
kg	mass	kilogram
s	time	second
°	angle	degree

   Output: assert=True

   Test Case Derivation: From instance model of the SRS it follows that: Pendulum Lagrangian ($L = T - V$) (Refer to SRS for equations)

   How test will be performed: We will check to see if each pendula and plot behaves as expected. Testing the Kinetic Energy equation:

$$T = \frac{1}{2}m_1 v_1^2 + \frac{1}{2}m_2 v_2^2$$

$$= \frac{1}{2}m_1(\dot{x}_1^2 + \dot{y}_1^2) + \frac{1}{2}m_2(\dot{x}_2^2 + \dot{y}_2^2)$$

$$= \frac{1}{2}m_1 l_1^2 \dot{\theta}_1^2 + \frac{1}{2}m_2\left[l_1^2\dot{\theta}_1^2 + l_2^2\dot{\theta}_2^2 + 2l_1 l_2 \dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2)\right]$$

5.1.3 Multi-Pendulum Simulation Hamiltonian Module Testing

The following tests were created to ensure that the input parameters module is storing the correct values specified by the user. This first step is critical as all of the other modules rely on these quantities.

1. test-Hamiltonian

   Type: Automaic [spelling! —SS]

   Initial State:

   Input: Input from Table of Units:

symbol	**unit**	**SI**
m	length	metre
kg	mass	kilogram
s	time	second
°	angle	degree

   Output: assert=True

   Test Case Derivation: From instance model of the SRS it follows that: (Refer to SRS for equations)

   How test will be performed: We will check to see if each pendula and plot behaves as expected. Testing the Kinetic Energy equation:

$$T = \frac{1}{2}m_1 v_1^2 + \frac{1}{2}m_2 v_2^2$$

$$= \frac{1}{2}m_1(\dot{x}_1^2 + \dot{y}_1^2) + \frac{1}{2}m_2(\dot{x}_2^2 + \dot{y}_2^2)$$

$$= \frac{1}{2}m_1 l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2\left[l_1^2\dot{\theta}_1^2 + l_2^2\dot{\theta}_2^2 + 2l_1 l_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2)\right]$$

   [Your tests are not complete. You say that you will check whether the pendula behaves as expected, but you don't say how this will be determined. You give equations, but not how they will be compared. If you are looking at time histories, maybe you can calculate the norm of the difference between the calculated and expected time histories. Another option is to compare to Dr. Nedialkov's software. Again, you could calculate the norm of the residual of the difference between your calculations and Dr. Nedialkov's software and yours. —SS]

5.1.4 Plotting Module

1. test-Plotting-Poincareé

   Type: Manual

   Initial State:

   Input: User Input, Data structure
   Input from Table of Units:

symbol	unit	SI
m	length	metre
kg	mass	kilogram
s	time	second
°	angle	degree

   Output: Time pendulum trail plot and Poincareé map

   Test Case Derivation: The Lagrangian Module, Hamiltonian Module and Data structure module should produce valid output that corresponds to the reference material and examples.

   How test will be performed: I will inspect the plots to see if they resmble [spell check —SS] the examples in my references. See references [1] and [7] [use BibTeX - similar comment elsewhere —SS]

1. test-Plotting-Trajectory

   Type: Manual

   Initial State:

   Input: User Input, Data structure

$$\sum \mathbb{R}(\texttt{for Langrangian equation})$$

$$P(x) : \mathbb{Z} \times \mathbb{R} \implies \mathbb{R}$$

   Output: Trajectory plot of multiple pendula

   Test Case Derivation: The Lagrangian Module and Data structure module should produce valid output that corresponds to the reference material and examples.

   How test will be performed: I will inspect the plots to see if they resmble the examples in my references. See references [1],[2],[3],[4],[5] and [6].

5.1.5 GUI/Plot Module

1. test-GUI-Plot-Simulation

   Type: Manual

   Initial State:

   Input: User Input, Data structure
   Input from Table of Units:

symbol	unit	SI
m	length	metre
kg	mass	kilogram
s	time	second
°	angle	degree

   Output: GUI interface and pendulum trail plot with axis

   Test Case Derivation: The Lagrangian Module, Hamiltonian Module and Data structure module should produce valid output that corresponds to the reference material and examples.

   Check GUI for visual integrity

   How test will be performed: I will inspect the plots to see if they resmble the examples in my references. See references [1],[2],[3],[4],[5] and [6].

5.2 Tests for Nonfunctional Requirements

Planning for nonfunctional tests of units will not be relevant to Multi-Pendulum Simulation . For system tests related to Nonfunctional requirements please see the System Verification and Validation Plan at `https://github.com/karolserkis/CAS-741-Pendula/blob/master/docs/`.

5.3 Traceability Between Test Cases and Requirements

	T??	T??	T??	T??	DD??	IM??
T??	X	X	X	X	X	X
T??	X	X	X	X	X	X
T??	X	X	X	X	X	X
T??					X	X
DD??	X	X	X		X	X
IM??	X	X	X		X	X

Table 2: Traceability Matrix Showing the Connections Between Items of Different Sections

	T??	T??	T??	T??	DD??	IM??
R??	X	X	X		X	X
R??	X	X	X	X	X	X
R??	X	X	X		X	X
R??	X	X	X	X	X	X

Table 3: Traceability Matrix Showing the Connections Between Requirements and Instance Models

6 Unit Testing Plan

[Unit testing plans for internal functions and, if appropriate, output files —SS]

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Software Verification Checklist

- Did any of the inputs you entered provide suprising [spell check —SS] results? If yes, what were they?
- Were all of the plots legible?

7.3 Usability Survey Questions?

1. Were you able to setup and execute the program using the comment and makefile instructions alone, or did you require additional troubleshooting/help? If any steps were unclear, please explain how they might be improved.
2. Did the program accept your inital [spell check —SS] input?
3. Did the program output the trajectory plot simulation?
4. Did the program run the simulation to its completion and smoothly without issues?
5. Did the plot of pendulum movement pattern and/or Poincaré map?

References

- [1] Dynamics of multiple pendula
  `http://wmii.uwm.edu.pl/~doliwa/IS-2012/Szuminski-2012-Olsztyn.pdf`
- [2] Pendulum
  `https://en.wikipedia.org/wiki/Pendulum`
- [3] Pendulum (mathematics)
  `https://en.wikipedia.org/wiki/Pendulum_(mathematics)`
- [4] Double Pendulum
  `https://en.wikipedia.org/wiki/Double_pendulum`
- [4] Differential-Algebraic Equations by Taylor Series
  `http://www.cas.mcmaster.ca/~nedialk/daets/`

- [5] Multi-body Lagrangian Simulations
  https://www.youtube.com/channel/UCCuLchOx0W0yoNE9KOCYlVQ
- [6] The double pendulum: Lagrangian formulation
  https://diego.assencio.com/?index=1500c66ae7ab27bb0106467c68feebc6
- [7] Poincaré map
  https://en.wikipedia.org/wiki/Poincar%C3%A9_map
- [8] D. L. Parnas, "On the criteria to be used in decomposing systems into modules", Comm. ACM, vol. 15, pp. 1053-1058, December 1972.
- [9] D. Parnas, P. Clement, and D. M. Weiss, "The modular structure of complex systems", in International Conference on Software Engineering, pp. 408-419, 1984.
- [10] D. L. Parnas, "Designing software for ease of extension and contraction," in ICSE '78: Proceedings of the 3rd international conference on Software engineering, (Piscataway, NJ, USA), pp. 264-277, IEEE Press, 1978.
- [11] Smith and Lai(2005); Smith et al. (2007).
  See bib file that doesn't work for me for full citation. [The example in the blank project template works. Following that example should have worked for you as well. —SS]

[I haven't tracked them down, but LaTeXtells me that you have unresolved references. —SS]