CAS 741: System Verification and Validation Plan

Dynamical Systems: MPSim

Karol Serkis
serkiskj@mcmaster.ca
GitHub: karolserkis

December 18, 2018

1 Revision History

Date	Developer(s)	Notes
October 22, 2018	Karol Serkis	First revision of document
October 21, 2018	Karol Serkis	New Blank Project Template utilized
December 17, 2018	Karol Serkis	All GitHub issues and comments addressed
December 18, 2018	Karol Serkis	SRS & bibliography fixed, & program name fixed

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
R	Requirement
NFR	Non-functional Requirement

2.1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as 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

2.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The choice of symbols was made to be consistent with calculus, ordinary differentials (ODE), the Lagrangian, kinematics etc. The standard mathematical spaces are used (e.g. $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, etc.) as well as some additional spaces defined in the following table.

symbol	**space**	**unit**	**description**
g	$\mathbb{R}$	–	gravitational constant
m_1	$\mathbb{R}$	kg	mass of the 1st pendulum weight
m_2	$\mathbb{R}$	kg	mass of the 2nd pendulum weight
m_n	$\mathbb{R}$	kg	mass of the nth pendulum weight
l_1	$\mathbb{R}$	m	length of the 1st pendulum rod
l_2	$\mathbb{R}$	m	length of the 2nd pendulum rod
l_n	$\mathbb{R}$	m	length of the nth pendulum rod
θ_1	$\mathbb{R}$	$^\circ$	amplitude from the pivot point
θ_2	$\mathbb{R}$	$^\circ$	amplitude from the 1st pendulum weight
θ_n	$\mathbb{R}$	$^\circ$	amplitude from the nth pendulum weight
L	$\sum \mathbb{R}$	–	Pendulum system Lagrangian
T	$\sum \mathbb{R}$	–	Kinetic energy of system
V	$\sum \mathbb{R}$	–	Potential energy of system
$P(x)$	$\mathbb{Z} \times \mathbb{R} \implies \mathbb{R}$	–	Poincar map P projects point x onto point P(x)

2.3 Abbreviations and Acronyms

The symbols are listed in alphabetical order.

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
NF	Non-Functional Requirement
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
T	Theoretical Model

[You shouldn't include symbols, abbreviations etc that you don't actually use in this document. —SS]

Contents

List of Tables

This document discusses the verification and validation requirements and provides a detailed description of the testing that will be carried out on the MPSim [I made a comment about your program name in your SRS. When you get a chance to address the issue there, please address it here as well. I've revised the blank template to use a common file that defines the program name. You might want to make the same modification. —SS] Generator program. Complementary documents include the System Requirement Specifications, Module Guide and Module Interface Specification. The full documentation and implementation can be found here. After reading this document one should be able to create and run test cases to verify and validate MPSim [provide an introductory blurb and road map of the Verification and Validation plan —SS]

3 General Information

3.1 Summary

The purpose of this document is to provide a comprehensive plan for testing the MPSim software against the requirements described in the MPSim SRS. The MPSim will allow users to generate diagrams (e.g. Poincare mapping) and plot trajectories over time using two different ODE/DAE initial value problem solvers. See the instance models (IM) in the SRS for more details.

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

3.2 Objectives

The purpose of this document is to describe the objectives for the MPSim program solution that only focuses on multi-pendulum simulations 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. The implementation should be reliable and accurate.

The System VnV plan will in summary:

- Satisfy the requirements of Dr. Spencer Smith [LaTeX has a rule that it inserts two spaces at the end of a sentence. It detects a sentence

as a period followed by a capital letter. This comes up, for instance, with Dr. Smith. Since the period after Dr. isn't actually the end of a sentence, you need to tell LaTeX to insert one space. You do this either by Dr. Smith (if you don't mind a line-break between Dr. and Smith), or Dr. Spencer Smith (to force LaTeX to not insert a line break). —SS]and those outlined in the SRS.

- Build confidence in the software correctness.
- Verify usability and effectiveness.
- The tests outlined in this document are limited to the verification of the requirements given in the MPSim SRS.
- The tests outlined in this document are limited to dynamic tests only. No formal static testing (code walk-through, code inspections, etc.) will be carried out.
- The MPSim software will be written in Python. The testing of implementations in other languages will not be considered in this document.

[State what is intended to be accomplished. The objective will be around the qualities that are most important for your project. You might have something like: "build confidence in the software correctness," "demonstrate adequate usability." etc. You won't list all of the qualities, just those that are most important. —SS]

3.3 Relevant Documentation

SRS document:

- Karol Serkis. System Requirements Specification for MPSim . Github, 2018.

[Reference relevant documentation. This will definitely include your SRS —SS]

[A hypperlink to your GitHub documentation would be great. —SS]

4 Plan

4.1 Verification and Validation Team

The test team includes the following members: Karol Serkis

The verification and validation team consists of (possibly Dr. Spencer Smith) classmates and I.

[Probably just you. :-) —SS]

4.2 SRS Verification Plan

The SRS verification plan consists of feedback from Dr. Spencer Smith and CAS 741 classmates. They will provide feedback regarding model assumptions, constraints and goals. Feedback from classmates and Dr. Smith will criticize the document outline, readability and requirements.
[Using Repos.xlsx you can be more specific about how is going to provide you with feedback. —SS] [You could mention the use of the GitHub issue tracker. —SS]

[List any approaches you intend to use for SRS verification. This may just be ad hoc feedback from reviewers, like your classmates, or you may have something more rigorous/systematic in mind.. —SS]

4.3 Design Verification Plan

The design verification plan will simply involve inspection of the software by Dr. Spencer Smith and CAS 741 classmates.
[Same comment as above. —SS]

[Plans for design verification —SS]

4.4 Implementation Verification Plan

The implementation verification plan consists of two parts. The first part is a software verification. It will be completed by myself any others who use MPSim and will involve verification that basic that basic software features have been implemented successfully, such as allowing the user to successfully utilize the software to full-fill its basic responsibilities (ie generate diagrams (e.g. Poincare mapping) and plot trajectories). More can be found in the

appendix. The second part involves running software tests outlined in sections 5 (System Test) and 6 (Static Verification). Unit testing will also be performed.

[You should at least point to the tests listed in this document and the unit testing plan. —SS]

4.5 Software Validation Plan

See SRS for more information in references —
External examples (tentative):

- [1] Differential-Algebraic Equations by Taylor Series
  http://www.cas.mcmaster.ca/~nedialk/daets/
- [2] Multi-body Lagrangian Simulations
  https://www.youtube.com/channel/UCCuLchOxOWOyoNE9KOCYlVQ
- [3] Poincar map
  https://en.wikipedia.org/wiki/Poincar%C3%A9_map

[I don't know how to use this information. How does it help? How does it apply? —SS]

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

5 System Test Description

5.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

1. test-Rin1

   Initial State: -

   Input: Initial input for the pendula weights in chain. See SRS and appendix for more info.

Output: Error or pass message.

How test will be performed: Combinations of inputting non-numerical values as input (such as letters), or numerical values outside of their respective constraints, will be considered. A successful test in these instances will be an error message.
I will also test cases with each variable in the input having an acceptable numerical value. A successful test in this case will be a pass message.

2. test-Rplt

   Initial State: -

   Input: Similar to test-Rin1

   Output: Pass of fail message (Is there a graph/plot or not)

   How test will be performed: This test will simply check to see if the trajectory plot or Poincare plot is plotted and if the picture agrees with the related physics. There should be no plot for the failed test cases of test-Rin1.

3. test-Rstl

   Initial State: -

   Input: Similar to test-Rin1

   Output: Binary Variable (Verification of stability)

   How test will be performed: The stability results will be compared with the stability analysis in (ref1).

[The test cases are specific enough. Each test case should only test one thing. That means the output cannot have an "or" in it. There should be enough information for someone to write the test without having to look anything up, or having to consult with you. You want to give specific values for the inputs. —SS]

5.2 Tests for Nonfunctional Requirements

1. test-NFR1

   Type: Static

Initial State: -

Input/Condition: MPSim Python code

Output/Result: Pass or Fail

How test will be performed: The software will be manually read by the developer and his supervisor to see if there is a more effective code structure to allow implementation of plot trajectories and Poincare plot and related ODE solvers utilized. [Static tests like this can be very helpful, but you should try to be more rigorous. Is there a checklist to follow? Is there a process for conducting code walkthroughs? —SS]

2. test-NFR2

   Type: Manual

   Initial State: Software system with prescribed input.

   Input: Input all inital user values to the ODE equation/Lagrangian equation.

   Output: Pass or fail (See SRS for equations).

   How test will be performed: The output ought fit with the physics and math simulations for MPSim . This test will loop through a various set of inputs to see if the plot does not go out of bounds or fail with overflow.

3. test-NFR3

   Type: Manual

   Initial State: Software system with prescribed input.

   Input: See SRS and appendix for more detail.

   Output: Pass or fail.

   How test will be performed: The output will be tested against the Simulations referenced in the SRS that are already available to compare. Also reference Dr. Ned's software and other related.

[Your test cases need to be more specific. What are the inputs for your tests and what are the expected outputs. For the expected outputs you need to say how you are going to calculate the error. Your output will likely be a sequence of values that you need to compare to the "correct" sequence of values. You should consider using the ratio of the Euclidean norm of the residual vector to the Euclidean norm of the expected output vector. —SS]

5.3 Traceability Between Test Cases and Requirements

	Rin	Rplt	Rstl	NFR1	NFR2	NFR3
test-Rin	X	X	X	X	X	X
test-Rplt	X	X		X	X	X
test-Rstl	X					
test-NFR1	X	X		X	X	X
test-NFR2	X	X		X	X	X
test-NFR3	X	X		X	X	X

Table 1: Traceability Between Test

[Provide a table that shows which test cases are supporting which requirements. —SS]

6 Static Verification Techniques

- Code inspection : I will go through the code to see if each step is correct with respect to the mathematical theory. In particular I will ensure that:
  - any discretization of functions is performed accurately from the pivot points.
  - plot trajectories are within scope
  - ensure a variable is not accidentally overwritten or cleared.
  - variables are being used in the right context. For example, equal increment in all used Cartesian coordinates.
  - functions from other packages are being used in the right context. For example, some packages have different standards for constants.
- Code walk-through: Dr. Spencer Smith and I will go through the code together to ensure that:
  - I correctly implemented the mathematical theory and numerical algorithms.

– I made the code manageable and maintainable for future use.

[These ideas are a good starting point. I suggest that you flesh them out more. Also, you can delete this section and move the contents to Section 4.4 (Implementation Verification Plan.) I've modified the template after reading several student examples. It turns out that this section and Section 4.4 are really covering the same topic. —SS]

[In this section give the details of any plans for static verification of the implementation. Potential techniques include code walk-through, code inspection, static analyzers, etc. —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 Usability Survey Questions?

- How long did it take to learn how to run the software?
- Was it easy to interpret the output?
- What aspects of this software do you feel need improvement?
- Was the output received and processed in an adequate amount of time?
- How does this program compare with other similar software?

[Did you reference this survey from the body of your report? I couldn't find a reference. You should connect this to a NFR test related to usability. I like your questions, but try to put some time into making them less ambiguous. Was it easy to interpret the output is a difficult question to answer. Instead you could ask them how long it took them to interpret the output? (You want to ask questions that are measurable and unambiguous.) —SS] [This is a section that would be appropriate for some projects. —SS]