Damped Harmonic Oscillator: Damped Harmonic Oscillator Illustrated by Online Calculator

Muhammad Waqar Ul Hassan Awan

February 19, 2024

Revision History

Date	Version	Notes
19 Feb, 2024	1.0	Initial VnV document

Contents

1 Symbols, Abbreviations, and Acronyms **1**

2 General Information **2**
- 2.1 Summary 2
- 2.2 Objectives 2
- 2.3 Relevant Documentation 3

3 Plan **3**
- 3.1 Verification and Validation Team 3
- 3.2 SRS Verification Plan 3
- 3.3 Design Verification Plan 4
- 3.4 Verification and Validation Plan Verification Plan 4
- 3.5 Implementation Verification Plan 4
- 3.6 Automated Testing and Verification Tools 4
- 3.7 Software Validation Plan 5

4 System Test Description **5**
- 4.1 Tests for Functional Requirements 5
  - 4.1.1 Input Parameters Handling 5
  - 4.1.2 Display and Confirmation of Entered Values 5
  - 4.1.3 Calculation of Oscillator Dynamics 6
  - 4.1.4 Verification of Solution Correctness 6
  - 4.1.5 Output Presentation 7
- 4.2 Tests for Nonfunctional Requirements 7
  - 4.2.1 Accuracy Testing 7
  - 4.2.2 Usability Testing 8
  - 4.2.3 Maintainability Testing 8
  - 4.2.4 Portability Testing 9
- 4.3 Traceability Between Test Cases and Requirements 10
  - 4.3.1 Function Requirements 10
  - 4.3.2 Non-Function Requirements 10

1 Symbols, Abbreviations, and Acronyms

As part of the comprehensive Verification and Validation (V&V) Plan for this project, in order to ensure clarity and consistency in the use of symbols, abbreviations, and acronyms, we direct all project stakeholders and team members to the Software Requirements Specification (SRS) document. Specifically, Section 1 of the SRS document, along with its Subsections 1.2 and 1.3, provide a thorough definition and explanation of all symbols, abbreviations, and acronyms employed throughout the project documentation and implementation.

This document outlines the Verification and Validation (VnV) Plan for a project focused on simulating the dynamics of damped harmonic oscillators. It provides a comprehensive framework for ensuring the software's correctness, reliability, and usability in modeling oscillatory systems under various damping conditions. The plan covers a range of testing strategies, from input validation and functionality tests to non-functional assessments such as usability and maintainability. Additionally, it details the approach for aligning software implementation with the outlined requirements, ensuring a thorough evaluation across different operating systems. This roadmap guides the project's VnV efforts, aiming for high accuracy and user satisfaction.

2 General Information

2.1 Summary

The software being tested is named "Damped Harmonic Oscillator(DHO)". It is designed to simulate the dynamics of harmonic oscillators subjected to both linear and non-linear damping forces. DHO offers a detailed representation of oscillatory systems through numerical solutions to differential equations, providing insights into their behavior under various damping conditions.

2.2 Objectives

The primary objective is to build confidence in the software's correctness and ensure its reliability in modeling damped oscillatory systems accurately. Another crucial goal is to demonstrate the software's adequacy in usability for educational and research purposes in physics and engineering. Following are some of the objectives that are out of scope:

- We will not verify the quality of usability in terms of user interface design beyond basic functionality and accessibility. This is due to limited resources and the project's primary focus on the accuracy of simulation results.

- The project will assume that all external libraries used for numerical computations and graphical representations are already verified by their respective development teams. This assumption allows us to concentrate our resources on the core functionalities of our software.

2.3 Relevant Documentation

Relevant documentation includes the Software Requirement Specification (SRS) document, which outlines the functional and non-functional requirements of the project. Additionally, design documents, such as the Module Guide (MG) and Module Interface Specification (MIS), are crucial as they detail the software's architecture and interaction interfaces. These documents are relevant to the Verification and Validation (VnV) efforts as they provide a foundation for testing strategies, ensuring that every aspect of the software is tested against the specified requirements and designs. By aligning VnV activities with these documents, we ensure a comprehensive evaluation of the software's functionality, reliability, and usability.

3 Plan

This section details the verification and validation (VnV) plan for the harmonic oscillator simulation project, designed and implemented as an individual, focusing on backend calculations and REST API functionality using Python/Flask, with ReactJS for the frontend.

3.1 Verification and Validation Team

As the sole contributor to this project, I will be responsible for all testing efforts. To enhance the robustness of the VnV process, I will take the help of two peers for review purposes: a primary reviewer and a secondary reviewer. These reviewers will provide feedback and insights to ensure the software's quality and checking if the requirements are met. Their roles will involve critiquing the design, testing strategy, and overall identification of any potential issues or areas for improvement in the project.

3.2 SRS Verification Plan

The SRS verification will involve a detailed review against a comprehensive checklist I will create, ensuring all functional and non-functional requirements are met. This checklist will be based on the specifications detailed in the SRS document. The SRS verification will also involve a review process, with the inclusion of a primary and a secondary reviewer.

3.3 Design Verification Plan

Given the individual nature of this project, design verification will focus on ensuring that the software's architecture and system design align with the outlined specifications and best practices for Python/Flask and ReactJS applications. The design verification will also incorporate a dual-peer review process, involving a primary and a secondary reviewer, to ensure a thorough examination of the software's architecture and system design.

3.4 Verification and Validation Plan Verification Plan

This step will involve a critical review of the VnV plan itself to ensure its adequacy and comprehensiveness. Adjustments will be made based on findings to improve the overall VnV strategy. The verification and validation (VnV) plan itself will undergo a review process involving both a primary and a secondary peer reviewer.

3.5 Implementation Verification Plan

The Implementation Verification Plan will include dynamic testing with unit tests primarily utilizing the PyTest framework for the Python/Flask backend. PyTest is chosen for its extensive support for testing Python applications, allowing for efficient testing of individual components and integration points. Static verification methods will include code walkthroughs, code inspections, and the use of static analyzers like Flake8 or Pylint to ensure code quality and adherence to coding standards. This blend of dynamic and static testing ensures a thorough validation of the software's functionality and reliability.

3.6 Automated Testing and Verification Tools

For automated testing, the PyTest framework will be used for unit testing the Python/Flask components, due to its robustness and ease of use. For static code analysis, Flake8 will serve as the primary linter, ensuring code quality and adherence to Python coding standards. GitHub Actions will automate the CI/CD pipeline, running tests and linting on every push to ensure continuous quality. Code coverage metrics will be summarized using the coverage.py tool, providing insights into untested parts of the codebase to improve test completeness.

3.7 Software Validation Plan

For software validation, I will compare the simulation results with those from external tools such as the GeoGebra damped harmonic oscillator calculator and other reputable online calculators to ensure accuracy. This external validation serves as an effective method for confirming the software's correctness against established benchmarks. Additionally, task-based inspections and peer reviews will be utilized to verify that the requirements document captures the correct requirements, providing a comprehensive validation approach from both a technical and user-experience perspective.

4 System Test Description

4.1 Tests for Functional Requirements

Given the project's focus on accurately modeling and simulating damped harmonic oscillators, the testing strategy is organized around key functional areas derived from the SRS document.

4.1.1 Input Parameters Handling

1. test-id1

   Control: Manual

   Initial State: Application loaded with no data entered.

   Input: Varying valid and invalid values for each parameter.

   Output: Acceptance of valid inputs and rejection or error messages for invalid inputs according to constraints in SRS section 4.2.6.

   Test Case Derivation: Verify input parameters for mass, spring constant, damping coefficient, initial displacement, and velocity.

   How test will be performed: Manually enter data into the software and observe behavior.

4.1.2 Display and Confirmation of Entered Values

1. test-id1

Control: Manual

Initial State: Parameters entered and awaiting confirmation.

Input: Parameters for simulation.

Output: Accurately display entered values for user review before simulation starts.

Test Case Derivation: Ensure entered values are correctly displayed for user confirmation.

How test will be performed: Check if the UI reflects the exact input values given by the user.

4.1.3 Calculation of Oscillator Dynamics

1. test-id1

   Control: Automated

   Initial State: Inputs validated and simulation ready.

   Input: Test scenarios with known outcomes.

   Output: Calculated values match expected theoretical or empirical results within defined tolerances.

   Test Case Derivation: Accuracy of displacement, velocity, and energy calculations over time.

   How test will be performed: Use automated scripts to compare software output against benchmark cases.

4.1.4 Verification of Solution Correctness

1. test-id1

   Control: Manual

   Initial State: Simulation completed with results generated.

   Input: Values for comparison.

   Output: Conformance to benchmarks with detailed reports on discrepancies.

Test Case Derivation: Solution verification against known values.

How test will be performed: Manual comparison to validate output against established benchmarks.

4.1.5 Output Presentation

1. test-id1

   Control: Manual

   Initial State: Simulation executed with results available.

   Input: User request for output display.

   Output: Outputs (displacement, velocity, energy) are presented in an easily interpretable format (graphs, tables).

   Test Case Derivation: Clarity and understandability of output presentation.

   How test will be performed: User evaluation of output presentation for clarity and completeness.

4.2 Tests for Nonfunctional Requirements

4.2.1 Accuracy Testing

1. test-id1

   Type: Manual

   Initial State: System initialized with predefined parameters for a damped oscillator.

   Input: Specific sets of input parameters for mass, spring constant, and damping coefficient.

   Output: Calculated displacement, velocity, and acceleration over time.

   How test will be performed: Compare the software output with analytical solutions numerical simulations to calculate relative error.

2. test-id2

Type: Automated

Initial State: System at rest with no initial input.

Input/Condition: Range of damping coefficients representing underdamped, critically damped, and overdamped scenarios.

Output/Result: System's response time to reach equilibrium or a steady state.

How test will be performed: Use automated testing tools to run simulations and verify accuracy against theoretical expectations, reporting relative errors.

4.2.2 Usability Testing

1. test-id1

   Type: Manual, User Survey

   Initial State: Users are provided with the software and basic instructions.

   Input: Tasks requiring users to input parameters, run simulations, and interpret results.

   Output: User feedback on ease of use, intuitiveness of the interface, and clarity of the results presentation.

   How test will be performed: Conduct a usability test followed by a survey to gather qualitative and quantitative data from users.

4.2.3 Maintainability Testing

1. test-id1

   Type: Manual

   Initial State: Existing software codebase and documentation.

   Input: Simulated changes to models and damping functions.

   Output: Effort required for implementation.

How test will be performed: Review and analysis by development team to estimate the effort needed for future modifications, ensuring it does not exceed 25% of the original development effort

4.2.4 Portability Testing

1. test-id1

   Type: Automated/Manual

   Initial State: Software ready for deployment.

   Input: Operation on Windows 10 and above, macOS Catalina and above, and popular Linux distributions such as Ubuntu 20.04 LTS.

   Output: Software runs without issues.

   How test will be performed: Run the software on various operating systems and Browsers and verify seamless operation, as outlined in the VnV Plan

4.3 Traceability Between Test Cases and Requirements

4.3.1 Function Requirements

Requirement ID	**Test Case ID**	**Description**
R1	test-id1	Verifies the software's ability to accept input parameters for mass, spring constant, damping coefficient, and initial conditions.
R2	test-id2	Ensures entered values are correctly displayed for user confirmation.
R3	test-id3	Tests the accuracy of displacement, velocity, and energy calculations over time.
R4	test-id4	Verifies the correctness of solutions against known benchmarks.
R5	test-id5	Assesses the clarity and understandability of the software's output presentation.

4.3.2 Non-Function Requirements

Requirement ID	**Test Case ID**	**Description**
NFR1	test-id1	Validates computational accuracy against theoretical values to within 0.10%.
NFR1	test-id2	Evaluates software usability through user surveys and usability tests.
NFR1	test-id3	Assesses maintainability by estimating effort required for future modifications and updates.
NFR1	test-id4	Checks software portability across Windows, macOS, and popular Linux distributions.