Test Plan for the Library of ODE Solvers (LODES)

Paul Aoanan

October 16, 2017

1 Revision History

Date	Version	Notes
October 16, 2017	1.0	Initial draft.

2 Symbols, Abbreviations and Acronyms

The following table lists the symbols, abbreviations and acronyms used in the Test Plan. The software library's Commonality Analysis (CA) tables provide supplementary items in addition to the ones listed below.

symbol	**description**
CA	Commonality Analysis
IDE	Integrated Development Environment
IVP	Initial Value Problem
ODE	Ordinary Differential Equation
LODES	Library of ODE Solvers
SRS	Software Requirements Specification
T	Test
O	Output

Table 1: Symbols, Abbreviations, and Acronyms used in the Test Plan

Contents

List of Tables

List of Figures

3 General Information

The following section provides an overview of the Test Plan for the Library of Ordinary Differential Equation (ODE) Solvers.
This section explains the purpose of this document, the scope of the system, and an overview of the following sections.

3.1 Purpose

The main purpose of this document (the Test Plan) is to describe the verification and validation process that will be used to test the functionality of LODES. This document closely follows the requirements and governs the subsequent testing activities. This document is intended to be used as a reference for all testing and will be used to increase confidence in the software implementation.

This document will be used as a guide and starting point for the Test Report. The test cases listed in this document will be executed and the output will be analyzed to uncover errors, increase confidence and correctness in the software.

3.2 Scope

The scope of the testing is limited to the Library of ODE Solvers. Given the appropriate inputs, each program in LODES is intended to find the solution to an Initial Value Problem (IVP).

3.3 Overview of Document

The following sections provide more detail about the testing of LODES. Information about the testing process is provided and the software specifications that were discussed in the Commonality Analysis are stated. The evaluation process that will be followed during testing is outlined and test cases for both the system testing and unit testing are provided.

4 Plan

This section provides a description of the software that is being tested, the team that will perform the testing, the approach to automated testing, the tools to be used for verification, and the non-testing based verification.

4.1 Software Description

The software being tested is the Library of ODE Solvers. Given the ODE, initial values of x and y, and the final value of x, the programs calculate the final value of y through the use of numerical methods.

4.2 Test Team

The test team that will execute the test cases, write and review the Test Report consists of:

- Paul Aoanan
- To be determined (A separate test report will be reviewed by an independent individual)

4.3 Automated Testing Approach

Automated testing will be implemented for LODES. It will consist of developing test cases that target the boundary conditions in the code

4.4 Verification Tools

The verification tools to be used will be the following:

1. Unit Testing Framework
   A Unit Testing Framework designed in MATLAB that will compare MATLAB's own functional programs with LODES' running the same inputs will be implemented.

   The following algorithm will be implemented to compare the results:

$$\Delta_{\text{relative}} = \frac{\text{Result}_{\text{MATLAB}} - \text{Result}_{\text{LODES}}}{\text{Result}_{\text{MATLAB}}}$$

2. Static Analyer
   The program's IDE (MATLAB) will be used as a Static Analyzer tool for program debugging and for checking syntax errors.

3. Continuous Integration
   The source code and the project repository is located in GitHub at: https://github.com/aoananp/cas741/. It provides the Build Server functionality to fully maintain and document the software through its lifecycle. As well, it provides the compare functionality for future regression testing and analysis of code updates.

4. Code Coverage Tool
   Due to the commercial nature of MATLAB, only commercial code coverage tools are viable for use due to the maturity, increased confidence, and detailed documentation that they offer. Other coverage tools may be considered, but no code coverage tool will be considered in the scope of this test plan due to budget constraints.

4.5 Non-Testing Based Verification

LODES will undergo the following non-testing based verification activities:

Code Inspection

LODES will undergo an initial desk review by the developer. Whilst it is preferred to have an independent body conducting this verification activity, the developer himself will peruse his own code for syntax errors and correct program calls. This code inspection activity provides the initial sanity check for the developer and the software.

Code Walkthrough

LODES will undergo code walkthrough by the developer. Again, whilst it is preferred to have an independent body conducting this verification activity, the developer himself will peruse his code and reference the Commonality Analysis for algorithm adherence. This activity will also involve logic analysis, loop and recursion boundary tracing (using by-hand test cases), passing of variables and references, and if the code is programmatically correct.

Symbolic Execution

Generally, symbolic execution will be performed using the boundary input conditions. Test conditions will be analyzed and executed by-hand. Generally, inputs in, on, and around the boundary conditions are chosen.

5 System Test Description

System testing will be executed to provide increased confidence that LODES will achieve the goals defined in the Commonality Analysis. It uses a "black box" approach wherein it tests the system as a whole through the use of input and output analysis.

5.1 Tests for Faulty Input

5.1.1 Input

The input will be based on the Assumptions table in the Commonality Analysis. Each test will correspond to an entry from the assumptions item whilst altering a specific input variable to a non-permissible value. The list of inputs is in order with the entries in the table.

Table 2: Fauly Input Test Cases

Number	Input	Expected Outcome
01	ODE Function Call $\notin$ {euler741, trap741, heun741, rk4741} $\cup$ MATLAB functions	error: undefined function call
02	$f(x, y) = y'' + y' + x + 2$	error: input out of bounds

5.2 Tests for Functional Requirements

5.2.1 Faulty Input Exception

Input Assumption Tests

1. FT-1

   Type: Static

   Initial State: The code "as-is"

   Input: The source code as the input to this test

   Output: Error / exception prompts

   How test will be performed: The code will undergo desk review to check for the presence of error exception handling prompts if the inputs do not adhere to the boundaries defined in the input assumptions

2. FT-2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State: The code "as-is"

   Input: The source code as the input to this test

   Output: Error / exception prompts

   How test will be performed: The code will undergo desk review to check for the presence of error exception handling prompts if the inputs do not adhere to the boundaries defined in the input assumptions

3. FT-3

   Type: Static

4. FT-4

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.2.2 Area of Testing2

...

5.3 Tests for Nonfunctional Requirements

5.3.1 Area of Testing1

Title for Test

1. test-id1

   Type:

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.3.2 Area of Testing2

...

5.4 Traceability Between Test Cases and Requirements

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 Usability Survey Questions?

This is a section that would be appropriate for some teams.