Test Plan for the Library of Linear Algebraic Equation Solver

Devi Prasad Reddy Guttapati

November 16, 2017

]

1 Revision History

Date	Version	Notes
November 16, 2017	1.0	Initial draft

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
O	Output
CA	Commonality Analysis
IM	Instance Module
ϵ	Difference

Contents

List of Tables

3 General Information

The following section gives an overview of the Verification and Validation (V & V) plan for the Library of Linear Algebraic Equation Solver. This section also explains the purpose, scope and overview of the document.

[An explicit web-link to your GitHub repo would be nice. —SS]

3.1 Purpose

The purpose of this document is to plan the Verification and Validation process for the Library of Linear Algebraic Equation Solver. The main purpose of this document is to check whether the Library of Linear Algebraic Equation Solver meets the specifications and fulfill its intended purpose. This document will be used as the reference and guidance for testing the Library of Linear Algebraic Equation Solver.

3.2 Scope

The scope of testing is limited to Library of Linear Algebraic Equation Solver. The library includes two linear algebraic solving functions. The scope is limited to these two solving functions.

[Reference your SRS document —SS]

[Explicitly state programming language —SS]

3.3 Overview of Document

The following sections provides in depth information about the V & V of Library of Linear Algebraic Equation Solver. The following sections also provides information about automated testing approach, testing tools. Test cases for system testing and unit testing are provided.

4 Plan

4.1 Software Description

Software which is tested is Library of Linear Algebraic Equation Solver. The library contains two linear algebraic equation solving algorithms. Given the initial values of algebraic equation A and B, the program calculates the final

value x by using numerical methods. [When you use mathematical symbols, put them in math mode (x). —SS]

4.2 Test Team

Devi Prasad Reddy Guttapati

4.3 Automated Testing Approach

Automated unit testing will be done for the Library of Linear Algebraic Equation Solver. Unit Testing Framework and R's Test Class will be used in a combination for automated testing. Syntax checking will be done automatically by the compiler. The aim of testing is 100% code coverage .

4.4 Verification Tools

The verification tools which will be used are as follows:

1. The programming language used is R. Comparison is done between the RStusio's [proof read —SS] own functional programs with the Library of Linear Algebraic Equation Solver by the Unit Testing Framework designed in RStudio. The following algorithm [This is an equation, not an algorithm —SS] is used for comparison.

$$\epsilon_{relative} = \frac{Result_{RStudio} - Result_{Library of Linear Algebraic Equation Solver}}{Result_{RStudio}}$$

   [The symbols in this equation are far too long. You also should use a non-italic font for the subscripts. For instance, you could use ϵ_{rel}, x_{R}, x_{LAES}, where x is the solution of the linear system of equations. You also need to do the equation component wise and then find the norm, so that you get one scalar value to represent the error. $\epsilon_{\text{rel}} = \frac{|x_{\text{R}} - x_{\text{LAES}}|}{|x|}$ —SS]

2. The framework for automated system testing is provided by using RStudio's Test Class .

3. For program debugging and for checking syntax errors the program's IDE (RStudio) will be used as a Static Analyzer tool . [proof read —SS]

4. covr which is an RStudio's library is used for code coverage.

5. RStudio's unit test library will be used for performing unit tests on the code.

4.5 Non-Testing Based Verification

Not Applicable [Your project is quite simple. To get a higher grade, you will need to do something beyond just testing. Others in the class are looking at code inspection and code walkthroughs. —SS]

5 System Test Description

System Test is done to verify whether the goals mentioned in Commonality Analysis are achieved. Input and Output analysis is used to test the system as a whole by black box testing approach.

5.1 Tests for Functional Requirements

5.1.1 Calculation Tests

Gaussian Elimination Method

1. **T-1: Simple Linear system involving two equations and two variables**

   Type: Functional, Automated, System.

   Initial State: Not applicable

   Input: A = $\begin{bmatrix} 2 & 3 \\ 4 & 9 \end{bmatrix}$, b = $\begin{bmatrix} 6 \\ 15 \end{bmatrix}$

   Output: = x = $\begin{bmatrix} 3/2 \\ 1 \end{bmatrix}$, Success = true

   How test will be performed: Automated system test

2. **T-2: Linear system involving three equations and three variables**

   Type: Functional, Automated, System.

   Initial State: Not applicable

   Input: $A = \begin{bmatrix} 1 & 3 & -2 \\ 3 & 5 & 6 \\ 2 & 4 & 3 \end{bmatrix}$, $b = \begin{bmatrix} 5 \\ 7 \\ 8 \end{bmatrix}$

   Output: $x = \begin{bmatrix} -15 \\ 8 \\ 2 \end{bmatrix}$, Success = true

   How test will be performed: Automated system test

3. **T-3: Linear system involving six equations and six variables**

   Type: Functional, Automated, System.

   Initial State: Not applicable

   Input: $A = \begin{bmatrix} 1 & 3 & -2 \\ 3 & 5 & 6 \\ 2 & 4 & 3 \end{bmatrix}$, $b = \begin{bmatrix} 5 \\ 7 \\ 8 \end{bmatrix}$

   [The input is 3 equations, not 6 —SS]

   Output: $x = \begin{bmatrix} -15 \\ 8 \\ 2 \end{bmatrix}$, Success = true

   How test will be performed: Automated system test

4. **T-4: Linear system of equations which are singular**

   Type: Functional, Automated, System.

   Initial State: Not applicable

   Input: $A = \begin{bmatrix} 0 & 2 & -1 \\ 3 & -2 & 1 \\ 3 & 2 & -1 \end{bmatrix}$, $b = \begin{bmatrix} 5 \\ 7 \\ 8 \end{bmatrix}$

   Output: x = no solution, Success = true

How test will be performed: Automated system test

Gauss-Jordan Method [You shouldn't copy and paste the tests between the two methods. This is a maintenance headache. If you correct a mistake in one place, you have to also correct it in another. It is fine to use the same test for both, but only right it down once. You can then just reference the previous test the second time it comes up. —SS]

1. **T-5: Simple Linear system involving two equations and two variables**

   Type: Functional, Automated, System.

   Initial State: Not applicable

   Input: $\mathrm{A} = \begin{bmatrix} 2 & 3 \\ 4 & 9 \end{bmatrix}$, $\mathrm{b} = \begin{bmatrix} 6 \\ 15 \end{bmatrix}$

   Output: $= \mathrm{x} = \begin{bmatrix} 3/2 \\ 1 \end{bmatrix}$, Success = true

   How test will be performed: Automated system test

2. **T-6: Linear system involving three equations and three variables**

   Type: Functional, Automated, System.

   Initial State: Not applicable

   Input: $\mathrm{A} = \begin{bmatrix} 1 & 3 & -2 \\ 3 & 5 & 6 \\ 2 & 4 & 3 \end{bmatrix}$, $\mathrm{b} = \begin{bmatrix} 5 \\ 7 \\ 8 \end{bmatrix}$

   Output: $\mathrm{x} = \begin{bmatrix} -15 \\ 8 \\ 2 \end{bmatrix}$, Success = true

   How test will be performed: Automated system test

3. **T-7: Linear system involving six equations and six variables**

   Type: Functional, Automated, System.

Initial State: Not applicable

Input: $A = \begin{bmatrix} 1 & 1 & -2 & 1 & 3 & -1 \\ 2 & -1 & 1 & 2 & 2 & -3 \\ 1 & 3 & -3 & -1 & 2 & 1 \\ 5 & 2 & -1 & -1 & 2 & 1 \\ -3 & -1 & 2 & 3 & 1 & 3 \\ 4 & 3 & 1 & -6 & -3 & -2 \end{bmatrix}$, $b = \begin{bmatrix} 4 \\ 20 \\ -15 \\ -3 \\ 16 \\ -27 \end{bmatrix}$

Output: $x = \begin{bmatrix} 1 \\ -2 \\ 3 \\ 4 \\ 2 \\ -1 \end{bmatrix}$, Success = true

How test will be performed: Automated system test

4. **T-8: Linear system of equations which are singular**

   Type: Functional, Automated, System.

   Initial State: Not applicable

   Input: $A = \begin{bmatrix} 0 & 2 & -1 \\ 3 & -2 & 1 \\ 3 & 2 & -1 \end{bmatrix}$, $b = \begin{bmatrix} 5 \\ 7 \\ 8 \end{bmatrix}$

   Output: x = no solution, Success = true

   How test will be performed: Automated system test

[You should be testing much larger systems than these for your functional requirements. You can generate matrices using the method of manufactured solutions that we discussed in class. You can also look up standard benchmark problems, like those given in the matrix market (you can find it with a google search). —SS]

5.2 Tests for Nonfunctional Requirements

5.2.1 Performance Requirements

Test for calculating speed

1. **T-9: Calculation speed of Library of Linear Algebraic Equation Solver.**

   Type: Non-Functional, Manual, Performance

   Initial State: Not Applicable

   Input/Condition: $\mathrm{A} = \begin{bmatrix} 1 & 3 & -2 \\ 3 & 5 & 6 \\ 2 & 4 & 3 \end{bmatrix}$, $\mathrm{b} = \begin{bmatrix} 5 \\ 7 \\ 8 \end{bmatrix}$

   Output/Result: Time

   How test will be performed: Comparing the time of Library of Linear Algebraic Equation Solver for different type and different size of input by increasing number of equations.

   [This problem is far too small to show any reasonable timing data. You need to be more specific about how you are going to vary the size of the benchmark problems. What are the axes of the graph you are going to plot? —SS]

5.2.2 Result Analysis

Accuracy

1. **T-10: Calculating the accuracy of Library of Linear Algebraic Equation Solver.**

   Type: Non-Functional, Manual, Accuracy

   Initial State: Not Applicable

   Input/Condition: $\mathrm{A} = \begin{bmatrix} 1 & 3 & -2 \\ 3 & 5 & 6 \\ 2 & 4 & 3 \end{bmatrix}$, $\mathrm{b} = \begin{bmatrix} 5 \\ 7 \\ 8 \end{bmatrix}$

   Output/Result: $\epsilon_{Relative}$ will be calculated by comparing the result obtained by Library of Linear Algebraic Equation Solver and the RStudio's own functional programs by the following algorithm

$$\epsilon_{relative} = \frac{Result_{RStudio} - Result_{Library of Linear Algebraic Equation Solver}}{Result_{RStudio}}$$

[What is the type of the terms in your equation? Are they reals, or sequences of reals? As mentioned previously, the right hand side should be vectors (sequences of reals) and the left hand side should be a scalar value. You'll need to use norms to turn the vectors into single numbers. —SS]

How test will be performed: Manually Comparing the result of Library of Linear Algebraic Equation Solver.

[You can automate this. You just write a program to do the calculations. —SS]

[What are the axes of the graph you are going to plot —SS]

5.3 Traceability Between Test Cases and Requirements

The following table shows the traceability mapping for test case and the Instance Models described in Commonality Analysis since CA does not include requirements.

Table 1: Requirements Traceability Matrix

Test Number	**CA Requirements**
T1	IM1
T2	IM1
T3	IM1
T4	IM1
T5	IM2
T6	IM2
T7	IM2
T8	IM2

6 Unit Testing plan

The testing plan for Library of Linear Algebraic Equation Solver is divided into several unique steps. These sequence of steps also includes specific function unit tests. The code will be divided into several individual units of code. A unit is a smallest component of code which can be tested. These units of code are tested if they are fit to use or not.

The code will be designed in several modules like Input/Output module, Gaussian Elimination Module, Gauss-Jordan Elimination Module. The basic plan for unit testing is that I will be using RStudio's unit testing packages. The tests from the functional requirements will also be used. The unit testing is a stage where we should test for code coverage. My aim is for 100% code coverage.

7 Appendix

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.

symbol	unit	description
ϵ	none	The measure of the difference between results obtained with Library of Linear Algebraic Equation Solver and RStudio.