LoSMS: Unit Verification and Validation Plan for a Library of Simplex Method Solvers

Hanane Zlitni

December 3, 2018

1 Revision History

Date	Version	Notes
December 6, 2018	1.1	Applied Brooks MacLachlan's Comments Posted on GitHub
December 3, 2018	1.0	First Draft

Contents

List of Tables

2 Symbols, Abbreviations and Acronyms

The following are symbols, abbreviations or acronyms used in this document:

symbol	**description**
T	Test
Z	Optimal solution(s) of the objective function
K	The points where the optimal solution(s) occur
MIS	Module Interface Specification
V&V	Verification and Validation

This document describes the unit Verification and Validation (V&V) plan for the Library of Simplex Method Solvers (LoSMS) tool. It is intended to be a refinement of the tool's system V&V plan by providing test cases based on the modules in the library's Module Interface Specification (MIS) document. The MIS, along with the full documentation of LoSMS, can be found at: https://github.com/hananezlitni/HZ-CAS741-Project.

The unit V&V plan starts by providing general information about the tool in Section 3. Then, Section 4 provides additional details about the plan, which include information about the V&V team, automated testing and verification tools and non-testing based verification. This is followed by the unit test description in Section 5, which consists of tests for the library's functional and nonfunctional requirements, categorized based on the modules in the MIS, and traceability between the test cases and modules.

3 General Information

3.1 Purpose

The software under test, LoSMS, is a general-purpose program family that is intended to be used by people from various backgrounds. It facilitates obtaining the optimal solution of a linear program using the simplex method. It accepts the objective function, the objective function goal (maximization or minimization) and the linear constraints and outputs the optimum of the linear programming problem.

3.2 Scope

All the modules, except for the output module, will be verified using the test cases in this document and the System V&V Plan (found at: https://github.com/hananezlitni/HZ-CAS741-Project/blob/master/docs/VnVPlan/SystVnVPlan/SystVnVPlan.pdf). The reason for excluding the output module is because its sole responsibility is to display the solution and there are no calculations included in it. Therefore, priority was given for the other modules.

4 Plan

4.1 Verification and Validation Team

The verification and validation team consists of one member: Hanane Zlitni.

4.2 Automated Testing and Verification Tools

PyTest, a unit testing framework for Python, will be used for automated testing. Code coverage would be achieved by selecting test cases that cover every possible statement and branch. In addition, Python's Mock library will be used to replace parts of the modules with mock objects and check their behaviour.

4.3 Non-Testing Based Verification

Not applicable for LoSMS.

5 Unit Test Description

The test cases discussed in this section were selected to ensure that the library does correctly what it is intended to do while maintaining quality. The test cases are derived from the tool's MIS document which can be found at: https://github.com/hananezlitni/HZ-CAS741-Project/blob/master/docs/Design/MIS/MIS.pdf

5.1 Tests for Functional Requirements

The following are the test cases related to the tool's functional requirements and categorized based on the modules in the MIS document.

5.1.1 Input Module

The input module is responsible for receiving the necessary inputs, verifying them and throwing an exception in case of a violation.

The test cases T6-T9 in 5.1.3 in the System V&V Plan cover the cases related to the input module. The document can be found at: https://github.

com/hananezlitni/HZ-CAS741-Project/blob/master/docs/VnVPlan/SystVnVPlan/SystVnVPlan.pdf

[fix labels of test cases in system plan and use cross-ref here —HZ]

5.1.2 Tableau Module

The tableau module is responsible for operations related to the simplex tableau, such as setting it up, updating its values and converting the tableau to the canonical form.

Since the conversion includes the execution of most of the module's access programs (found in the MIS document Zlitni (2018)), the following test cases were selected to test each possible case in toCanonical() - which helps achieving branch coverage.

In addition, a test case was selected for updateTableau() to cover the case when negating the objective function is needed.

1. **T1: Test canonical form conversion for maximization problems with less than or equal to inequalities**

   Control: Automatic

   Initial State: sTableau $= \begin{bmatrix} -2 & 3 & -1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 10 \\ 4 & -3 & 1 & 0 & 3 \\ 2 & 1 & -1 & 0 & 10 \end{bmatrix}$

   Input: -

   Output: No output, but the state changes to:

   sTableau $= \begin{bmatrix} -2 & 3 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 10 \\ 4 & -3 & 1 & 0 & 0 & 1 & 0 & 3 \\ 2 & 1 & -1 & 0 & 0 & 0 & 1 & 10 \end{bmatrix}$

   Test Case Derivation: Section 7 in the MIS (Zlitni (2018))

   How test will be performed: PyTest and/or Mock

   [Explanation: Since the linear constraints are less than or equal to (from calling getLCsType()), the slack variables are added to the matrix —HZ]

2. **T2: Test canonical form conversion for less than or equal to inequalities: adding slack variables**

   Control: Automatic

   Initial State: sTableau = $\begin{bmatrix} 2 & -3 & 1 & 0 \\ 3 & 4 & 0 & 24 \\ 7 & 4 & 0 & 16 \end{bmatrix}$

   Input: -

   Output: No output, but the state changes to:

   sTableau = $\begin{bmatrix} -2 & 3 & -1 & 0 & 0 & 0 \\ 3 & 4 & 0 & 1 & 0 & 24 \\ 7 & 4 & 0 & 0 & 1 & 16 \end{bmatrix}$

   Test Case Derivation: Section 7 in the MIS (Zlitni (2018))

   How test will be performed: PyTest and/or Mock

3. **T3: Test canonical form conversion for minimization problems: the transition of *wasMin***

   Control: Automatic

   Initial State: wasMin = False

   Input: -

   Output: No output, but the state changes to:

   wasMin = True

   Test Case Derivation: Section 7 in the MIS (Zlitni (2018))

   How test will be performed: PyTest and/or Mock

4. **T4: Test updateTableau() for negating the objective function**

   Control: Automatic

   Initial State: sTableau = $\begin{bmatrix} 2 & -3 & 1 & 0 \\ 3 & 4 & 0 & 24 \\ 7 & 4 & 0 & 16 \end{bmatrix}$

   Input: 0

   Output: No output, but the state changes to:

$$\text{sTableau} = \begin{bmatrix} -2 & 3 & -1 & 0 & 0 & 0 \\ 3 & 4 & 0 & 1 & 0 & 24 \\ 7 & 4 & 0 & 0 & 1 & 16 \end{bmatrix}$$

Test Case Derivation: Section 7 in the MIS (Zlitni (2018))

How test will be performed: PyTest and/or Mock

[Explanation: T1 and T2 are also tests for updateTableau(), since the conversion to the canonical form cannot be done without updating the tableau. Since they don't cover if updateTableau() was called for converting maximization to minimization (negating the first row in the tableau), I added T3 to test that case. —HZ]

5.1.3 Simplex Solver Module

The simplex solver module is responsible for performing the simplex method steps to calculate the optimum and the points where it occurs.

The following test cases were selected to cover the cases for solving minimization and maximization problems.

1. **T5: Test solveLP() for minimization problems: value of Z**

   Control: Automatic

   Initial State: Z = [ ]

   Input: $\begin{bmatrix} 2 & -3 & 1 & 0 & 0 & 0 \\ 3 & 4 & 0 & 1 & 0 & 24 \\ 7 & 4 & 0 & 0 & 1 & 16 \end{bmatrix}$, True

   Output: No output, but the state changes to:

   Z = [-4.57]

   Test Case Derivation: Section 8 in the MIS (Zlitni (2018))

   How test will be performed: PyTest and/or Mock

   Note: Since this test case includes the comparison of floats and they're usually not exactly equal, the function *math.isclose()* that Python 3 provides will be used. By using this function, a small difference between the two floats being compared will be allowed. (Source: Pranskevichus and Selivanov (2015)).

2. **T6: Test solveLP() for minimization problems: value of K**

   Control: Automatic

   Initial State: K = [ ]

   Input: $\begin{bmatrix} 2 & -3 & 1 & 0 & 0 & 0 \\ 3 & 4 & 0 & 1 & 0 & 24 \\ 7 & 4 & 0 & 0 & 1 & 16 \end{bmatrix}$, True

   Output: No output, but the state changes to:

   K = [2.29 , 0]

   Test Case Derivation: Section 8 in the MIS (Zlitni (2018))

   How test will be performed: PyTest and/or Mock

   Note: Since this test case includes the comparison of floats and they're usually not exactly equal, the function *math.isclose()* that Python 3 provides will be used. By using this function, a small difference between the two floats being compared will be allowed. (Source: Pranskevichus and Selivanov (2015)).

3. **T7: Test solveLP() for maximization problems: value of Z**

   Control: Automatic

   Initial State: Z = [ ]

   Input: $\begin{bmatrix} -2 & 3 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 10 \\ 4 & -3 & 1 & 0 & 0 & 1 & 0 & 3 \\ 2 & 1 & -1 & 0 & 0 & 0 & 1 & 10 \end{bmatrix}$, False

   Output: No output, but the state changes to:

   Z = [3]

   Test Case Derivation: Section 8 in the MIS (Zlitni (2018))

   How test will be performed: PyTest and/or Mock

4. **T8: Test solveLP() for maximization problems: value of K**

   Control: Automatic

   Initial State: K = [ ]

Input: $\begin{bmatrix} -2 & 3 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 10 \\ 4 & -3 & 1 & 0 & 0 & 1 & 0 & 3 \\ 2 & 1 & -1 & 0 & 0 & 0 & 1 & 10 \end{bmatrix}$, False

Output: No output, but the state changes to:

K = [0, 0, 3]

Test Case Derivation: Section 8 in the MIS (Zlitni (2018))

How test will be performed: PyTest and/or Mock

5.2 Tests for Nonfunctional Requirements

The test cases T11-T15 in section 5.2 in the System V&V Plan cover the nonfunctional requirements (qualities) that are important for LoSMS. The document can be found at: https://github.com/hananezlitni/HZ-CAS741-Project/blob/master/docs/VnVPlan/SystVnVPlan/SystVnVPlan.pdf

5.3 Traceability Between Test Cases and Modules

Test Case Number	Module
T6-T9 in System V&V Plan	Input Module
T1	Tableau Module
T2	Tableau Module
T3	Tableau Module
T4	Tableau Module
T5	Simplex Solver Module
T6	Simplex Solver Module
T7	Simplex Solver Module
T8	Simplex Solver Module

Table 1: Traceability Between Test Cases and Modules

References

Elvis Pranskevichus and Yury Selivanov. What's new in python 3.5, 2015. URL https://docs.python.org/3/whatsnew/3.5.html#pep-485-a-function-for-testing-approximate-equality.

Hanane Zlitni. Module interface specification for a library of simplex method solvers (losms), 2018. URL https://github.com/hananezlitni/HZ-CAS741-Project/blob/master/docs/Design/MIS/MIS.pdf.

6 Appendix

This section provides additional content related to this document.

6.1 Symbolic Parameters

There are no symbolic parameters used in this document.