System Verification and Validation Plan of a Library of Simplex Method Solvers (LoSMS) [put your library name in the title —SS][done —HZ]

Hanane Zlitni

October 22, 2018

1 Revision History

Date	Version	Notes
December 17, 2018	2.0	Final Draft (includes making the document consistent with the rest of the deliverables)
December 16, 2018	1.2	Applied Dr. Smith's Comments
December 06, 2018	1.1	Applied Brooks MacLachlan's Comments Posted on GitHub
October 22, 2018	1.0	First Draft

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
V&V	Verification and Validation
LoSMS	Library of Simplex Method Solvers
CA	Commonality Analysis
SRS	Software Requirements Specification
A	Assumption
IM	Instance Model
$s.\ t.$	Subject to
Z	Optimal solution of the objective function
x_1, x_2, x_3	Decision variables

Contents

List of Tables

This document describes the system verification and validation (V&V) plan for the Library of Simplex Method Solvers (LoSMS) tool. [I recently modified the blank project template to put the (equivalent of) the famname command in a common file, which can be shared between all your files. You might want to do the same. —SS][done —HZ] It is based on the tool's commonality analysis (CA) that can be found, along with the full documentation of LoSMS, at: https://github.com/hananezlitni/HZ-CAS741-Project. [nice to include this link —SS][thank you —HZ]

The V&V plan starts by providing general information about the tool and this document in Section 3. Then, Section 4 provides additional details about the plan, which includes information about the V&V team, the SRS, design and implementation verification plans and the software validation plan. This is followed by the system test description in Section 5, which consists of tests for the tool's functional and nonfunctional requirements and traceability between test cases and requirements. The document is concluded by Section 6 which describes the techniques for static verification.

3 General Information

3.1 Summary

The software under test, LoSMS, is a general-purpose program family that facilitates obtaining the optimal solution of a linear program, using the simplex method, given the objective function, the objective function goal (maximization or minimization) and the linear constraints. Since the simplex algorithm is widely used in various fields, LoSMS is intended to be used by people from different backgrounds to help them optimize parameters of their choice.

3.2 Objectives

The objective of this verification and validation plan is to build confidence in the correctness of the LoSMS tool (i.e. it produces the correct output for the corresponding inputs), while providing satisfactory usability.

3.3 References

Different sections in this document refer to the tool's CA which can be found at: https://github.com/hananezlitni/HZ-CAS741-Project/blob/master/docs/SRS/CA.pdf. [You should reference your SRS here. —SS][done —HZ]

4 Plan

4.1 Verification and Validation Team

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

4.2 SRS Verification Plan

The CA for the LoSMS tool will be verified by getting feedback from 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][done —HZ] and my CAS 741 classmates which comes from formal reviews of the document.

The feedback from Dr. Smith is expressed as comments written in the actual document. The feedback from the CAS 741 students, on the other hand, is expressed as issues posted in the project's GitHub repository. An Excel sheet "Repos.xlsx" present in the course's GitHub repository contains the names of the reviewers for each documentation (Brooks MacLachlan for this document). After getting the comments, I apply them one by one and make changes in the document. [You could be more specific (using Repos.xlsx) on who exactly is going to review your documentation. —SS][done —HZ]

4.3 Design Verification Plan

LoSMS's design documents will be verified by getting feedback from Dr. Spencer Smith and my CAS 741 classmates which comes from formal reviews of the documents.

The feedback from Dr. Smith is expressed as comments written in the documents, whereas the feedback from the CAS 741 students is expressed as issues posted in the project's GitHub repository. After getting the comments, I apply them one by one and make changes in the documents.

4.4 Implementation Verification Plan

The implementation of the LoSMS tool will be verified dynamically by executing the test cases detailed in this plan and the Unit V&V Plan using testing frameworks (e.g. Python's unittest). Based on the errors I get, the implementation will be modified.

If the time permits, the implementation of the LoSMS tool will be statically verified by performing code review with Dr. Spencer Smith and my CAS 741 classmates using the GitHub issue tracker.

4.5 Software Validation Plan

Not applicable for LoSMS.

5 System Test Description

System testing for the LoSMS tool ensures that the correct inputs produce the correct outputs. The test cases in this section are derived from the instance models and the requirements detailed in the tool's CA, found at: https://github.com/hananezlitni/HZ-CAS741-Project/blob/master/docs/SRS/CA.pdf.

5.1 Tests for Functional Requirements

This section contains the test cases related to the tool's functional requirements (e.g. handling input errors, calculating the solution and producing the output). They are categorized based on the linear programming problem goal (maximization and minimization). This section is based on the CA which can be found at https://github.com/hananezlitni/HZ-CAS741-Project/blob/master/docs/SRS/CA.pdf.

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. —SS][done —HZ]

5.1.1 Tests for Solving Maximization Linear Programs

The following are the test cases related to solving maximization linear programs. It covers both cases of having and not having an optimal solution. This is based on the tool's CA which can be found at https://github.com/hananezlitni/HZ-CAS741-Project/blob/master/docs/SRS/CA.pdf.

[Note: I excluded the case where there are multiple optimal solutions because after many tries, I wasn't able to achieve this case in the code. I wanted to include a paragraph about why I excluded it, but at this point in the document I think we're still abstract and not thinking about the code yet. —HZ]

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. —SS][done —HZ]

1. **T1: Optimal Solution**

   Control: Automatic

   Initial State: -

   Input: $\max Z = 2x_1 - 3x_2 + x_3$
   $s.t.$ $x_1 + x_2 + x_3 \leq 10$
   $4x_1 - 3x_2 + x_3 \leq 3$
   $2x_1 + x_2 - x_3 \leq 10$

   Output: $Z = 3,\ x_1 = 0,\ x_2 = 0,\ x_3 = 3$

   How test will be performed: Unit testing using Unittest

   Test Case Derivation: IM1 in Zlitni (2018) [good to have a field for test case derivation. I might have to add that to the blank template. :-) —SS][thank you (: —HZ]

2. **T2: No Optimal Solution**

   Control: Automatic

   Initial State: -

Input: $\max Z = 2x_1 + x_2$
$s.t. \quad x_1 - x_2 \leq 10$
$2x_1 - x_2 \leq 40$

Output: Exception message: "This linear program does not have an optimal solution"

How test will be performed: Unit testing using Unittest

Test Case Derivation: IM1 in Zlitni (2018) and Niu

5.1.2 Tests for Solving Minimization Linear Programs

The following are the test cases related to solving minimization linear programs. It covers both cases of having and not having an optimal solution. This is based on the tool's CA which can be found at https://github.com/hananezlitni/HZ-CAS741-Project/blob/master/docs/SRS/CA.pdf.

1. **T3: Optimal Solution**

   Control: Automatic

   Initial State: -

   Input: $\min Z = -2x_1 + 3x_2$
   $s.t. \quad 3x_1 + 4x_2 \leq 24$
   $7x_1 + 4x_2 \leq 16$

   Output: $Z = -4.5714,\ x_1 = 2.2857,\ x_2 = 0$

   How test will be performed: Unit testing using Unittest

   Test Case Derivation: IM2 in Zlitni (2018)

   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. **T4: No Optimal Solution**

   Control: Automatic

   Initial State: -

Input: $\min Z = 3x_1 + 14x_2$

$$
\begin{aligned}
s.\,t. \quad & -x_1 - 5x_2 \leq -6 \\
& -x_1 - 4x_2 \leq -5 \\
& -x_1 - 3x_2 \leq -4 \\
& -x_1 - 2x_2 \leq -5 \\
& -x_1 - x_2 \leq -6
\end{aligned}
$$

Output: Exception message: "This linear program does not have an optimal solution"

How test will be performed: Unit testing using Unittest

Test Case Derivation: IM2 in Zlitni (2018)

5.1.3 Tests for Faulty Inputs

The following test cases verify that the received inputs are not empty. It is based on the tool's CA found at: https://github.com/hananezlitni/HZ-CAS741-Project/blob/master/docs/SRS/CA.pdf.

1. **T5: No Objective Function**

   Control: Automatic

   Initial State: -

   Input: min

   $$
   \begin{aligned}
   s.\,t. \quad & 3x_1 + 4x_2 \leq 24 \\
   & 7x_1 + 4x_2 \leq 16
   \end{aligned}
   $$

   Output: Exception message: "At least one input is missing"

   How test will be performed: Unit testing using Unittest

   Test Case Derivation: IM1 & IM2 in Zlitni (2018)

2. **T6: No Linear Constraints**

   Control: Automatic

   Initial State: -

   Input: $\min Z = -2x_1 + 3x_2$

   Output: Exception message: "At least one input is missing"

How test will be performed: Unit testing using Unittest

Test Case Derivation: IM1 & IM2 in Zlitni (2018)

[Seems like good coverage. —SS]

5.2 Tests for Nonfunctional Requirements

5.2.1 Usability

1. **T7: Test for the Usability of LoSMS**

   Type: Usability Testing

   Initial State: -

   Input/Condition: -

   Output/Result: -

   How test will be performed: For testing purposes, I will be implementing a command line handling tool that will use LoSMS. For the usability testing, I will be asking participants to try the tool [you should clarify that you are going to make a tool using your library —SS][done —HZ] then answer the usability survey questions (see Section 7.2). The goal is to ensure that the library provided the services the users requested and that they are satisfied with the results obtained.

5.2.2 Portability

1. **T8: Test for the Portability of LoSMS**

   Type: Static

   Initial State: -

   Input/Condition: -

   Output/Result: -

   How test will be performed: Running LoSMS on Mac, Windows and Linux operating systems

5.2.3 Accuracy

1. **T9: Test for the Accuracy of the Outputs**

   Type: Dynamic

   Initial State: -

   Input/Condition: -

   Output/Result: -

   How test will be performed: I plan to report the relative error of the expected output for each test case detailed in this document [great! —SS]

5.2.4 Correctness

1. **T10: Test for the Correctness of LoSMS**

   Type: Parallel Testing

   Initial State: -

   Input/Condition: -

   Output/Result: -

   How test will be performed: I plan to make a comparison between LoSMS and MatLab to evaluate the correctness of LoSMS. The problem to be tested is:

$$
\begin{aligned}
&\min Z = -x_1 - 2x_2 \\
&s.t. \quad 2x_1 + x_2 = 10 \\
&\qquad x_1 + x_2 = 6 \\
&\qquad -x_1 + x_2 = 2 \\
&\qquad -2x_1 + x_2 = 1
\end{aligned}
$$

5.2.5 Performance

1. **T11: Test for the Performance of LoSMS**

   Type: Parallel Testing

   Initial State: -

   Input/Condition: -

Output/Result: -

How test will be performed: I plan to make a comparison between LoSMS and MatLab to evaluate the performance of LoSMS by finding the time (in seconds) it took to solve a linear program. The problem to be tested is:

$$\begin{aligned} \min Z &= -x_1 - 2x_2 \\ s.t. \quad & 2x_1 + x_2 = 10 \\ & x_1 + x_2 = 6 \\ & -x_1 + x_2 = 2 \\ & -2x_1 + x_2 = 1 \end{aligned}$$

5.2.6 Stability

1. **T12: Test for the Stability of LoSMS Under Heavy Load**

   Type: Stress/Load Testing

   Initial State: -

   Input/Condition: -

   Output/Result: -

   How test will be performed: I plan to use a third-party tool that automates loading LoSMS with inputs so I can reach the greatest possible number of inputs (i.e. hundreds). Specifically, I plan to increase the number of parameters in the objective function and the number of linear constraints to check the stability of the library. [You need to be more specific. —SS][done. I tried to be as specific as possible and answer the questions given by Brooks in the issue tracker. —HZ]

5.3 Traceability Between Test Cases and Requirements

The following table describes the mapping between the test cases and requirements.

Test Case Number	Requirements
T1	R1, R**??**, R4, R5, R6
T2	R1, R**??**, R4, R5, R6
T3	R1, R**??**, R4, R5, R6
T4	R1, R**??**, R4, R5, R6
T5	R1, R2, R3
T6	R1, R2, R3
T7	NFR1
T8	NFR2
T9	NFR3
T10	NFR4
T11	NFR5
T12	NFR6

Table 1: Traceability Between Test Cases and Requirements

6 Static Verification Techniques

If the time permits, static verification of the LoSMS library implementation will performed using code review with Dr. Spencer Smith and my CAS 741 classmates (the comments will be in the form of GitHub issues).

References

Shun-Chen Niu. Special situations in the simplex algorithm. URL https://www.utdallas.edu/~scniu/OPRE-6201/documents/LP10-Special-Situations.pdf.

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. Commonality analysis of a library of simplex method solvers, 2018. URL https://github.com/hananezlitni/HZ-CAS741-Project/blob/master/docs/SRS/CA.pdf.

7 Appendix

This section provides additional content related to this system V&V plan.

7.1 Symbolic Parameters

There are no symbolic parameters used in this document.

7.2 Usability Survey Questions

1. Did LoSMS successfully provide all the services you requested?

   (Yes / No)

   Why have you chosen the above response?

2. How confident are you that LoSMS provided you with the correct results?

   (1 / 2 / 3 / 4 / 5) ; *(1) being not confident at all and (5) being very confident*

   Why have you given the above score?

3. How satisfied are you with the library's response time?

   (1 / 2 / 3 / 4 / 5) ; *(1) being not satisfied at all and (5) being very satisfied*

   Why have you given the above score?

4. How likely are you to recommend LoSMS to a friend?

   (1 / 2 / 3 / 4 / 5) ; *(1) being very unlikely and (5) being very likely*

   Why have you given the above score?

5. Rate your overall satisfaction with LoSMS out of 10

   (1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9 / 10)

   Why have you given the above score?

[Good survey questions. —SS][thank you —HZ]