Module Interface Specification for Library of Linear Algebraic Equation Solver

Devi Prasad Reddy Guttapati

December 5, 2017

1 Revision History

[You commented out the input for Comments. I had to uncomment it so that I could add comments with the wss command. —SS]

Date	**Version**	**Notes**
Date 1	1.0	Initial Draft

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at: https://github.com/deviprasad135/CAS741/blob/master/Doc/SRS/CA.pdf

Contents

3 Introduction

The following document details the Module Interface Specifications for Library of Linear Algebraic Equation Solver. [You should usually avoid one sentence paragraphs. These two paragraphs would certainly make sense combined together. —SS]

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at the mentioned link below:

`https://github.com/deviprasad135/CAS741`.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1999), with the addition that template modules have been adapted from Ghezzi et al. (2002). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1999). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \ldots | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Library of Linear Algebraic Equation Solver.

Data Type	**Notation**	**Description**
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

The specification of Library of Linear Algebraic Equation Solver uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Library of Linear Algebraic Equation Solver uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

[Rather than Input Computing, you should just say Input. You don't really compute the input. Output is also a simpler name than Output Computing. —SS]

Level 1	**Level 2**
Hardware-Hiding	
Behaviour-Hiding	Input Computing Module Output computing Module Library of Linear Algebraic Equation Solver Module
Software Decision	Matrix Module Gaussian Elimination Module Gauss-Jordan Elimination Module

Table 1: Module Hierarchy

6 MIS of Library of Linear Algebraic Equation Solver Module

6.1 Module

LLAES

6.2 Uses

IC (Section 7), OC (Section 8), Matrix (Section 9), GE (Section 10), GJE (Section 11)

6.3 Syntax

Name	In	Out	Exceptions
Linear_Algebraic Equation_Solving Methods	Linear_Algebraic Equation_Solving Method $\in 1, 2$	-	-

[Rather than use 1, 2, you should introduce an enumerated type that has two values: GaussElim, GaussJordan. You could export this new type, rather than define it in the input field —SS]

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Access Routine Semantics

- transition: None
- output: None
- exception: None
- pseudocode:

```
if (option = 1)
    then solve using Gaussian Elimination Method
if (option = 2)
    then solve using Gauss−Jordan Elimination Method
```

[You have no transition, no output, no exceptions, so your module doesn't actually do anything. What you enter under pseudocode should actually be entered under transition. —SS]

[You actually aren't passing any arguments to your solver methods. —SS]

7 MIS of Input Computing Module

7.1 Module

IC

7.2 Uses

Matrix (Section 9), GE (Section 10), GJE (Section 11)

7.3 Syntax

Name	In	Out	Exceptions
A	$n \times n$ matrix $\in \mathbb{R}$ and n > 0	-	Complex Numbers
b	$n \times 1$ matrix $\in \mathbb{R}$ and n > 0	-	Complex Numbers

[Your type for A and b is not correct. The type for a matrix can be represented as $\mathbb{R}^{n\times n}$ —SS]

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Access Routine Semantics

- transition: None
- output: None
- exception: None

[This module doesn't actually do anything! There is not transition, not output and no exceptions. Rather than an Input module, you can just use multiple instances of your Matrix module to define the input to your solver module. —SS]

8 MIS of Output Computing Module

8.1 Module

OC

8.2 Uses

None

8.3 Syntax

Name	In	Out	Exceptions
x	-	$n \times 1$ matrix $\in \mathbb{R}$ and n > 0	-

[The type for x is not expressed correctly. —SS]

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Access Routine Semantics

- transition: None
- output: None
- exception: None
- pseudocode:

```
if (displayresult)
    {print(x);}
end if
```

[If you are printing output to the screen, or to a file, you need an environment variable. —SS]

9 MIS of Matrix Module

9.1 Module

Matrix

9.2 Uses

GE (Section 10), GJE (Section 11), OC (Section 8)

9.3 Syntax

Name	In	Out	Exceptions
A	$\mathbb{R}$	$n \times n$ matrix $\in \mathbb{R}$ and n > 0	-
b	$\mathbb{R}$	$n \times 1$ matrix $\in \mathbb{R}$ and n > 0	-

[You need to think about this module more. The input cannot be a single real number. The input is a sequence of real numbers. You also need state variables. You can look at the sequence module example in the lecture slides, combined with the examples for Abstract Data Types (ADTs). For an Abstract Data Type you need to use the "Template Module," rather than "Module." You don't actually represent A and b together in this module. The Matrix module defines a new ADT. You can create an A object and a b object. This about the constructor for each matrix and about the getters that makes sense. —SS]

9.4 Semantics

9.4.1 State Variables

None

9.4.2 Access Routine Semantics

- transition: None
- output: None
- exception: None
- pseudocode:

```
function{
            A = (Input of Real Numbers)
            n = $(length (A))^{1/2}$
A = matrix(A, row = n, col = n, byrow = TRUE)
}
```

10 MIS of Gaussian Elimination Module

10.1 Module

GE

10.2 Uses

This module is used to solve the system of Linear Algebraic Equations.

10.3 Syntax

Name	**In**	**Out**	**Exceptions**
A	$n \times n$ matrix $\in \mathbb{R}$ and n > 0	-	always_a_square_matrix no_singular_matrix
b	$n \times 1$ matrix $\in \mathbb{R}$ and n > 0	-	-
x	-	$n \times 1$ matrix $\in \mathbb{R}$ and n > 0	-

[The type for A and b should be Matrix. That is why you are defining a Matrix type. —SS]

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Access Routine Semantics

- transition: None
- output: None
- exception: None
- pseudocode:

```
for $k = 1$ to $n - 1$
    find a pivot p such that
    $|a_{pk}| \geq |a_{ik}|$ for $K \leq i \leq n$
    if $|a_{pk}| = 0$ do
        return "Singular Matrix"
        end the entire loop
    else
        interchange row p and k

    for $i = k + 1$ to $n$
        $factor_{ik} = \frac{a_{ik}}{a_{kk}}$
        for $j = k + 1$ to $n$
            $a_{ij} = a_{ij} - factor_{ik} * a_{kj}$
        end for
    end for
end for

$x_n = \frac{b'_n}{a_{nn}}$
for i in n-1 to 1
    for j in i+1 to n
        $sum = a_{ij}x_j$
    end for
    $x_i = \frac{b'_n - sum}{a_{ii}}$
end for
```

11 MIS of Gauss-Jordan Elimination Module

11.1 Module

GJE

11.2 Uses

This module is used to solve the system of Linear Algebraic Equations.

11.3 Syntax

Name	**In**	**Out**	**Exceptions**
A	$n \times n$ matrix $\in \mathbb{R}$ and n > 0	-	always_a_square_matrix no_singular_matrix
b	$n \times 1$ matrix $\in \mathbb{R}$ and n > 0	-	-
x	-	$n \times 1$ matrix $\in \mathbb{R}$ and n > 0	-

11.4 Semantics

11.4.1 State Variables

None

11.4.2 Access Routine Semantics

- transition: None
- output: None
- exception: None
- pseudocode:

```
for $k = 1$ to $n - 1$
     find a pivot p such that
     $|a_{pk}| \geq |a_{ik}|$ for $K \leq i \leq n$
     if $|a_{pk}| = 0$ do
          return "Singular Matrix"
          end the entire loop
     else
          interchange row p and k

     for $i = k + 1$ to $n$
          $factor_{ik} = \frac{a_{ik}}{a_{kk}}$
          for $j = k + 1$ to $n$
               $a_{ij} = a_{ij} - factor_{ik} * a_{kj}$
          end for
     end for
end for

Assuming that the matrix is not singular
for k = n to 2
     for i = k+1 to 1
          $factor_{ik} = \frac{a_{ik}}{a_{kk}}$
          for j = k−1 to 1
               $a_{ij} = a_{ij} - factor_{ik} * a_{kj}$
          end for
     end for
end for
```

```
for i in 1 to n
    $x_i = \frac{b_n^{'}}{a_{ii}}$
end for
```

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of software engineering*. Prentice Hall PTR, 2002.

Daniel Hoffman and Paul Strooper. Software design, automated testing, and maintenance–a practical approach. 1999.

[Why did you redo the Bib entry for HoffmanAndStrooper1995? You could have saved time (and improved the reference information) by just using the entry in the course repo, under References.bib. —SS]

12 Appendix

Not Applicable