The Development and Implementation of a Computer Linear Algebra System with Java

Dan Dai

BSc (Hons) in Computer Science

2005

The Development and Implementation of a Computer Linear Algebra System with Java

Submitted by Dan Dai

COPYRIGHT

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Batchelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specially acknowledged, it is the work of the author.

Signed: …………………………….

Abstract

This development project presents a simplified matrix-oriented computer linear algebra system based on Java technology. This system is designed to be capable of solving common linear algebra computation problems. The system is mainly intended to be used by beginners to the world of linear algebra. Other users like computer scientists and mathematicians may also wish to use this system as a portable linear algebra computation tool alternatively to large and expensive general purpose computer algebra systems. Therefore, the system is aiming to provide all basic facilities required by these users. In this version, algorithms are implemented for common matrix arithmetic operations, matrix determinant and trace calculation and matrix inverse, ad-joint and transpose computation. To solve systems of linear equations, two algorithms (Cramer's rule and Gauss-Jordan elimination) are also included. A graphical interface with save/load facility is another important feature of the system.

Unlike other computer algebra packages in the market today, this system only concentrates on the dealing with linear algebra problems. As this result, the whole application can be packed into binary with a very reasonable size. Also thanks to the Java technology, the system could virtually be run at almost all software and hardware platforms with a same complied binary. With all these properties, the portability of the delivered system is ensured.

In this dissertation, the complete design ideas and implementation details will be described. The future development plan of the system is also discussed.

Acknowledgements

I would like to thank my project supervisor, Dr Dan S Richardson for his on going support, help, advice and valuable suggestions throughout the project. Also thanks to my families and friends for their supports and encouragements.

Contents

Chapter 1

Introduction

Since the first computer algebra systems appear in the early 1970s, numerous symbolic mathematics software projects have been developed. Among all these projects, many can be used to solve linear algebra problems, whereas most of such software projects aim to deliver general purpose computer algebra systems that cover all aspects of algebraic computation. However, these systems may not suitable for all users.

Most CASs (computer algebra systems) are commercial software, which means the user have to pay for the binary application and don't have access to the entire source code. As the result, users of such systems have no idea about the exact implementation. Therefore, they are not suitable for users who want to know more about the detail algorithms.

The ability to deal with linear algebra problems is only one of many features that general purpose computer algebra systems provide. Such systems with large sets of functionalities are indeed very welcomed by scientists, mathematicians and engineers in every subject. These professionals are the main users of such system as they directly benefit from various embedded components that provide capabilities such like graphics, statistics and numerical computations. However, users who don't require these features also have to pay for them. On the other hand, with these components, the systems require more storage spaces to install. For users who purely interested in linear algebra functions, they are totally not necessary.

Rich in functionality also leads to more complex software. To use any of these software packages, users are supposed to learn at least part of the command syntaxes. In most cases, there are hundreds of them listed in the user manual. Along with different usage of each command (i.e. in many case, each command has different parameters options), massive materials must be read before starting using the systems. For users only want linear algebra functions, obviously, a lot of time and energy will be wasted on learning these syntaxes, which are mostly not related to the topic they interested.

There are some relatively small applications that can be used to deal with linear algebra problems. However, they may have all kinds of limitations such as lack of functionalities and probabilities.

This new computer algebra system aims to provide a system that solves the problems of current software packages in some aspects. Thus, the objective of this dissertation project is to deliver an open-source Java application that focused on performing common matrix-oriented linear algebra operations. The system will be mainly designed for linear algebra students. It will need to provide a user friendly interface for easy user interaction, meanwhile, be small in terms of storage size. It will also need to be portable and easy to install on various hardware and software platforms.

Chapter 2

Literature review

2.1 Introduction

Humans make errors, but the computers don't. More and more scientists and mathematician prefer to use computers to solving algebraic related problems, because these problems can be too complicate to solve by human hands. As this result, numerous computer algebra systems have been developed. Among modern computer algebra systems, many of them have the capability of solving linear algebra problem. These systems are tend to be very powerful bus very complex, thus a small linear algebra focused system with sufficient functionality is very useful to many linear algebra beginner.

This chapter will discuss the general background of computer algebra systems along with the concepts of linear algebra. In order to have a more in-depth understanding of the problems, the principles of matrix and a few common linear algebra operations will be introduced. At the end of the chapter, several existing systems will be reviewed and possible implantation methods of the new system will also be discussed.

2.2 The Background of Computer Algebra System

A computer algebra system (CAS) is a software program that facilitates symbolic mathematics. The core functionality of a CAS is manipulation of mathematical expressions in symbolic form. (Wikipedia, 2004)

Mathematical expressions in the a CAS could be variable with data type like numbers and matrices, polynomials, ordinary mathematics functions with parameters like square and mod, arithmetic symbols like addition and multiplication and many more depend on different CAS design. Manipulations of the these expressions typically include simplification, substitution of values for expressions, changing the form of expression, differentiation, factorization, numeric operations, solving integration and differential equations, solving system of linear/non-linear equations and matrix operations. (. *Computer algebra system)*

Computer algebra systems first appear in early 1970s. They became more and more popular and commonly used by researchers in many subjects. Today, most widely used systems are commercial systems like Maple, Mathematica, MuPAD and REDUCE. A number of free software like Axiom and Yacas are also very popular. It is worthwhile to gain experience and knowledge from studying these systems.

2.3 The Computer Linear Algebra System

Linear algebra is the branch of mathematics concerned with the study of vectors, vector spaces (or linear spaces), linear transformations, and systems of linear equations. It is the study of linear sets of equations and their transformations. Linear algebra is widely used in both abstract algebra and functional analysis. (*Linear algebra)*

As above, we can clear see a computer linear algebra system can be the subsystem of a general purpose computer algebra system. The central object of this project is to isolate the linear algebra functions from general computer algebra systems. Then integrate them with other facilities to produce a new computer linear algebra system.

The main functionality of a computer linear algebra system will normally include:

- Common matrix manipulation: For example, transpose;
- Row operation calculator: Interactively perform a sequence of elementary row operations on the given m x n matrix A;
- Transforming a matrix to row echelon form: Find a matrix in row echelon form that is row equivalent to the given m x n matrix A;
- Transforming a matrix to reduced row echelon form: Find the matrix in reduced row echelon form that is row equivalent to the given m x n matrix A;
- Solving a linear system of equations: Solve the given linear system of m equations in n unknowns;
- Calculating the inverse using row operations: Find (if possible) the inverse of the given n x n matrix A;
- Calculating the determinant using row operations: Calculate the determinant of the given n x n matrix A.
- Performing matrix arithmetic: Addition, subtraction, scalar, multiplication, etc…

2.4 The Mathematic Background

It's important to have an in-depth understanding of the basic principles of linear algebra operations. In this section, concepts of linear algebra foundation elements and some basic algorithms will be investigated.

2.4.1 Matrix

In mathematics, a matrix is a rectangular table of elements of ring-like algebraic structure (*Matrix (mathematics))*. For this project, the entries of a matrix are only supposed to be real numbers. The numbers in the matrix are called the elements of matrix. The horizontal lines in a matrix are called rows and the vertical lines are called columns. Rows are labelled from the top of the matrix, columns from left. A matrix is constructed by a certain number of rows and certain number of columns. The location of an element in the matrix is usually described by giving the row and column in which it lies. Matrix with m rows and n columns, called m by n matrix as a rectangular array of numbers usually presented in the following terms. See Figure 2.1:

$$\begin{array}{cc} \longleftarrow \text{ n columns } \longrightarrow & \\ \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} & \begin{array}{c} \uparrow \\ \text{m} \\ \text{rows} \\ \downarrow \end{array} \end{array}$$

Figure 2. 1

A matrix is called square matrix if it has the same number of rows as columns. A matrix is called sub-matrix if it is obtained by deleting certain rows or/and columns of another matrix. A matrix consisting of one row or one column is correspondingly called row matrix and column matrix. The main diagonal of a square matrix is the elements from top left corner to the bottom right corner. The trace of a square matrix is the sum of its main diagonal entries. A diagonal matrix is a square matrix which only the main diagonal entries are non-zero. The main diagonal elements themselves can be zero or non zero. A diagonal matrix with 1s in all main diagonal entries is called identity matrix. [William, G]

2.4.2 Matrix Addition and Subtraction

The algorithms for matrices addition and subtraction are quite trivial. The sum of matrix A and matrix B is obtained by adding together the corresponding elements of A and B (See Figure 2.2).

$$\mathrm{A}+\mathrm{B}=\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix}$$

Figure 2. 2

Same as ordinary arithmetic operations, matrix substation is actually another way of addition (See Figure 2.3). The only constraint for matrix addition and subtraction is the size of both matrices must be same. Otherwise, the sum or difference does not exist.

$$\mathrm{A}-\mathrm{B}=\begin{bmatrix} a_{11}+(-1)b_{11} & a_{12}+(-1)b_{12} & \cdots & a_{1n}+(-1)b_{1n} \\ a_{21}+(-1)b_{21} & a_{22}+(-1)b_{22} & \cdots & a_{2n}+(-1)b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}+(-1)b_{m1} & a_{m2}+(-1)b_{m2} & \cdots & a_{mn}+(-1)b_{mn} \end{bmatrix}$$

Figure 2. 3

2.4.3 Scalar Multiplication of Matrices

Let A be the matrix and c be a number. The scalar multiple of A by c is denoted as cA and the number c is called scalar. The result of scalar multiplication is obtained by multiplying every element of A by c (See Figure 2.4). The size of the result matrix will be same as the size of A.

$$\mathrm{cA}=\begin{bmatrix} c\times a_{11} & c\times a_{12} & \cdots & c\times a_{1n} \\ c\times a_{21} & c\times a_{22} & \cdots & c\times a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ c\times a_{m1} & c\times a_{m2} & \cdots & c\times a_{mn} \end{bmatrix}$$

Figure 2. 4

2.4.4 Matrices Multiplication

Let A and B be matrices, the multiplication of A and B is denoted as AB. Matrix multiplication is an associative operation, which means A(BC) and (AB)C are not equivalent. To multiply A and B, the number of columns in A must be the same as the number of rows in B. Otherwise, the product dose not exist. To obtain the result of AB, the element in row *i* and column *j* is the result of multiplying the corresponding elements of row *i* of A and column *j* of B and adding them together (See Figure 2.5). This is the most natural algorithm of multiplying two matrices.

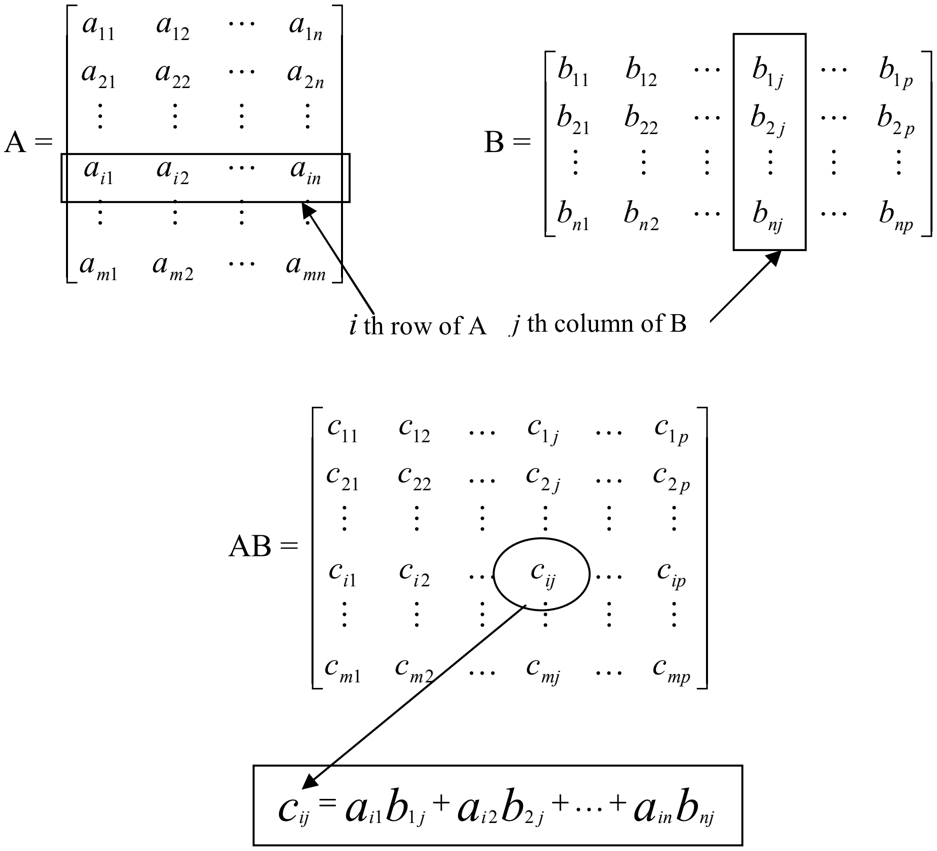

A =
B =
i th row of A
j th column of B
AB =
$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj}$

Figure 2. 5

In Figure 2.5, A is an m × n matrix; B is an n × p matrix. The column number of A and the row number of B are both equal to n. Thus the product exists. The row number of result matrix equals to m (the row number of A). The column number of result matrix equals to q (the column number of the B). The result AB is an m × p matrix.

There are other algorithms exist for matrix multiplication. One of most famous algorithms for fast matrix multiplication is introduced by V. Strassen in 1969[V.Strassen]. This algorithm can be used to multiply square matrix and has a complexity of $\Theta(m^{\lg(7)})$ they where lg(7) ≈ 2.807, m is corresponding to the size of an m × m square matrix[Steven H, Elaine M.]. Compared to the trivial algorithm above, there is a significant improvement.

2.4.5 Matrix Transpose

To obtain the transpose of a matrix, simply translates the matrix columns into rows (or other way round, rows into columns). In other words, the (*i, j*) element of a matrix becomes the (*j, i*) element of its transposed matrix (See Figure 2.6). Thus if the size of such matrix is m × n, the size of the transposed matrix will become n × m. The transpose of matrix A is denoted as A^t.

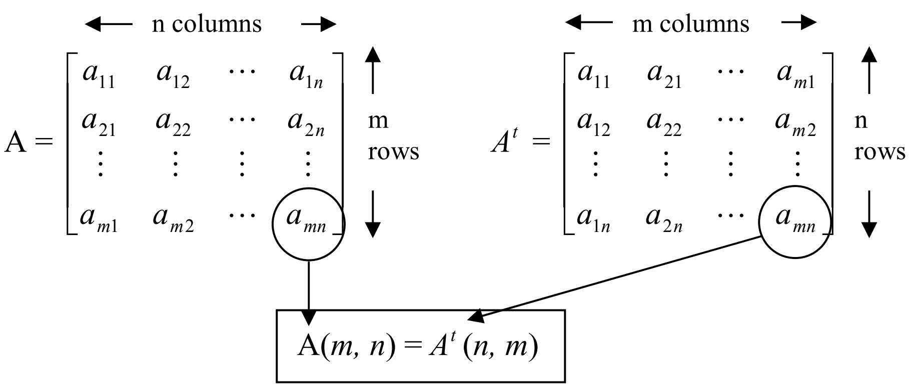

n columns
m rows
$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$
m columns
n rows
$A^t = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}$
A(*m, n*) = A^t (*n, m*)

Figure 2. 6

2.4.6 Determinant of a Matrix

Let A be a square matrix, the determinant of A is denoted as |A|.
If A is a one by one matrix, the determinant of A is the value of A(1, 1);
If A is a two by two matrix, the determinant of A is A(1, 1)A(2, 2) – A(2, 1)A(1, 2). This is obtained by calculating the difference of the products of the two diagonals of the matrix (See Figure 2.7);

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

Figure 2. 7

If A is a three by three matrix, the determinant of A is A(1, 1)A(2, 2)A(3,3) + A(1, 3)A(2, 1)A(3, 2) + A(1, 2)A(2, 3)A(3, 1) – A(1, 3)A(2, 2)A(3, 1) – A(1, 1)A(2, 3)A(3, 2) – A(1, 2)A(2, 1)A(3, 3).

The general solution of obtaining the determinant of an m × m square matrix is quite complicated. Let A be an m × m matrix, the minor of element a_{ij} is denoted as M_{ij} and is the determinant of the matrix that remains after deleting row *i* and column *j* of A. The cofactor of element a_{ij} is denoted as C_{ij} and is given by C_{ij} = $(-1)^{i+j}$ M_{ij}. Thus the determinant of a square matrix is the sum of the products of the elements of any row or column and their cofactors.
ith row expansion: |A| = $a_{1j}C_{i1} + a_{i2}C_{i2} + \cdots + a_{im}C_{im}$
jth column expansion: |A| = $a_{1j}C_{1j} + a_{2j}C_{2j} + \cdots + a_{mj}C_{mj}$

[William, G]
The following formula (Figure 2.8) is the general way of computing determinant. It was introduced by Gottfried Leibniz with what is now known as the *Leibniz formula*:

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)}$$

Figure 2. 8

The sum is computed over all permutations σ of the numbers {1,2,...,n} and sgn(σ) denotes the signature of the permutation σ: +1 if σ is an even permutation and −1 if it is odd. See even and odd permutations for an explanation of even/odd permutations. This formula contains n! (Factorial) summands and is therefore impractical to use it to calculate determinants for large n.

Gauss algorithm is a general way of computing determinants of matrices. It consists of following rules:
If *A* is a triangular matrix, i.e. A(*i, j*) = 0 whenever *i* > *j*, then det(A) = A(1, 1)A(1, 2)…A(n, n)
If *B* results from *A* by exchanging two rows or columns, then det(B) = - det(A)
If *B* results from *A* by multiplying one row or column with the number *c*, then det(B) = c det(A)
If *B* results from *A* by adding a multiple of one row to another row, or a multiple of one column to another column, then det(B) = det(A)
[*Determinant*]

A matrix is called singular matrix if its determinant equals to zero, otherwise, it is called non-singular.

2.4.7 Adjoint of a Matrix

Let A be a n × n square matrix. Let c_{ij} be the cofactor ofa_{ij}. The matrix whose (i, j)th element is c_{ij} is called the matrix of cofactors. The transpose of this matrix is called the adjoint of A and denoted as adj(A) (Gareth Williams P142). See Figure 2.9.

$$\underbrace{\begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nn} \end{bmatrix}}_{\text{matrix of cofactors}} \xrightarrow{\text{Transpose}} \underbrace{\begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nn} \end{bmatrix}}_{\text{adjoint matrix}}$$

Figure 2. 9

2.4.8 Matrix Inverse

Let A be a square matrix with |A| ≠ 0. A is invertible with
A^{-1} = (1 / |A|) adj(A)

A square matrix is invertible if and only if its determinant doesn't equal to zero. In other words, inverse of a matrix exists if and only if the matrix is non-singular.
(Gareth Williams P144)

2.4.9 System of Linear Equations

In the world of linear algebra, Cramer's rule, Gaussian Elimination and Gauss-Jordan elimination are the two common and effective methods to solve the systems of linear equations.

A system of linear equations is a set of equations, such as (See Figure 2.10):

$$\begin{cases} 5x_1 - 4x_2 - 5x_3 = 2 \\ 4x_1 - 3x_2 + x_3 = -1 \\ -x_1 + \tfrac{1}{2}x_2 - x_3 = 0 \end{cases}$$

Figure 2. 10

Generally, a system with m unknowns (represented by $x_{1,}\, x_1 \ldots x_{\mathrm{m})}$ and n equations can be written into the following format (See Figure 2.11). The coefficients are represented as a_{ij}.

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1m}x_m = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2m}x_m = b_2 \\ \vdots \qquad\quad \vdots \qquad\qquad\quad \vdots \qquad \vdots \\ \vdots \qquad\quad \vdots \qquad\qquad\quad \vdots \qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nm}x_m = b_n, \end{array}$$

Figure 2. 11

To solve the systems of equations in linear algebra, it is important to translate the format using in Figure 2.11 into another format by separating the coefficients and unknowns. Therefore, the translation of system used in Figure 2.11 is represented in Figure 2.12.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \qquad \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \qquad = \qquad \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

matrix of coefficients matrix of unknowns matrix of constants

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2m} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n1} & \cdots & a_{nm} & b_n \end{bmatrix}$$

Augmented matrix

Figure 2. 12

Augmented matrix (See Figure 2.12) is obtained by combining of coefficients matrix and constants matrix.
[*System of linear equations*]

2.4.9.1 Cramer's Rule

Let AX = B be the system of n linear equations in n variables. A is coefficients matrix, X is matrix of unknowns and B is the constants matrix. If the determinant of A (i.e. |A|) does not equal to zero, the system has a unique solution (See Figure 2.13).

$$x_1 = \frac{|A_1|}{|A|},\ x_2 = \frac{|A_2|}{|A|},\ x_3 = \frac{|A_3|}{|A|},\ \ldots,\ x_n = \frac{|A_n|}{|A|}$$

matrix A_i is obtained by replacing column *i* of matrix *A* with matrix B

Figure 2. 13

(Gareth Williams)

2.4.9.2 Gaussian Elimination and Gauss-Jordan Elimination

To understand Gauss-Jordan Elimination and Gaussian Elimination, it's important to introduce the concepts of echelon form and reduced echelon form. The following four conditions can be used to examine whether a matrix is in echelon form or reduced echelon form.

1. Any rows consisting entirely of zeros are grouped at the bottom of the matrix.
2. The first nonzero element of each other row is 1. This element is called a leading 1.
3. The leading 1 of each row after the first is positioned to the right of the leading 1 of the previous row.
4. All other elements in a column that contains a leading 1 are zero.

(Gareth Williams)

A matrix is in echelon form if it satisfies the condition 1, 2 and 3. A matrix satisfies all the above four conditions is in reduced echelon form.

The procedures of Gauss-Jordan Elimination:

- Write down the augmented matrix of the system of linear equations.
- Derive the reduced echelon form of the augmented matrix using elementary row operations. This is done by creating leading 1s, then zeros above and below each leading 1, column by column starting with the first column.
- Write down the system of equations corresponding to the reduced echelon form. This system gives the solution.

(Gareth Williams)

The procedures of Gaussian Elimination:

- Write down the augmented matrix of the system of linear equations.
- Find an echelon form of the augmented matrix using elementary row operations. This is done by creating leading 1s, the zero below each leading 1, column by column starting with the first column.
- Write down the system of equations corresponding to the echelon form.
- Use back substitution to arrive the solution.

(Gareth Williams)

By comparing the algorithms of Gauss-Jordan Elimination and Gaussian Elimination, it is not possible to determine which algorithm is better. It's better to choose different algorithm under different situations.

2.5 Reviews of existing systems

There are numerous computer algebra systems in the world today. It's not possible to review all of them. On the other hand, it is not difficult and expensive to review some CAS as they are commercial software. In this section, two systems – Maple and Reduce will be reviewed. This is because they are easy to get hold of and have close relations to linear algebra. Each review is mainly consisted with general introduction, linear algebra functions description and user interface analysis.

2.5.1 Maple

This review is based on Maple 6 for Microsoft Windows operating system.

2.5.1.1 General introduction

Maple is a general computer algebra system. It's very powerful and it is currently widely used in many subjects and research areas. This section is a discussion of general Maple usage.

In Maple commands, a semicolon must be placed at the end of any Maple command and the Return key must be pressed for Maple to act on the input. Maple will then show the result of the command. To suppress the showing of the result, a colon may be used instead of a semicolon. Maple then acts on the command but does not show the results. However, it stores any assignments made internally. Failure to use the proper terminating punctuation will result in a warning message.

User could define functions in Maple. These functions are called procedures. Each procedure can take almost any thing as arguments.
Example:

```
F := proc(x,y) x^2 + y^2 end;
```

This procedure has two formal parameters, x and y, which returns the sum of power of x and power of y.

In Maple system, by typing "?" will display a help page. Even for advanced users, this help system is very useful. By entering the topic, the help will display all the related information.

2.5.1.2 The linear algebra package

The linear algebra package in Maple is called "`linalg`" package. By default, the linear algebra is not pre-loaded. By using "`with( )`" command, the maple system will load the package. For example, the following command is used to load the linear algebra package:

```
with(linalg):
```

The following examples (MathSci, 2004) are the common usage of linear algebra commands.

- Declare a matrix, where each row is put in [ ] with row entries separated by commas. Example:
  ```
  A := matrix ( [ [ row 1], ..., [row n] ] );
  ```

- Declare a matrix, where m is the number of rows, n is the number of columns, and then the numbers are entered between [ ] row-wise. Example:
  ```
  A := matrix ( m, n, [ numbers here ] );
  ```

- Evaluate a matrix, and displays the result. Example:
  ```
  evalm( name );
  ```

- Add two matrices A and B and stores the result in C -- assuming A and B have been declared and are the right sizes. Example:
  ```
  C := evalm(A + B);
  ```

- Multiplies two matrices A and B and stores the result in C -- assuming A and B have been declared and are the right sizes -- Note that matrix multiplication is &*, not *. Example:
  ```
  C := evalm(A &* B);
  ```

- Puts matrix A in reduced row echelon form. Example:
  ```
  rref(A);
  ```

- Declares a vector (column matrix) named b, where the numbers are separated by commas between [ ]. Example:
  ```
  b := vector( [ numbers here ] );
  ```

- Augments matrix A with vector b, building a new matrix with b added to A as an additional, last column, can also be used to build a matrix out of a set of column vectors. Example:
  ```
  augment(A, b);
  ```

- Does the forward elimination step of Gaussian Elimination on A. Example:
  ```
  gausselim(A);
  ```

- Does the back substitution step of Gaussian Elimination on A. Example:
  ```
  linsolve(A, b);
  ```

- Does the back substitution step of Gaussian Elimination on A. Example:
  ```
  backsub(A);
  ```

- Calculates determinant of A. Example:
  `det(A);`

- Do a transposes on matrix A. Example:
  `transpose(A);`

- Inverse matrix A. Example:
  `inverse(A);`

- List eigenvalues for A. Example:
  `eigenvals(A);`

- Find eigenvalues, multiplicity and eigenvectors for A. Example:
  `eigenvects(A);`

2.5.1.3 Interface Analysis

Depending on installed operating systems and user preferences, it's possible to use Maple under two different interfaces. One is the classical command-line user interface. The other one is the Maple worksheet graphical user interface, which is commonly used today. The two different system share the same set of built in commands, thus both of them provide same functionalities.

Because this dissertation project aims to deliver a software system with graphical interface, thus it's appropriate to focus on the analysing the Maple graphical user interface.

The most important feature of Maple graphical interface version is the built in worksheet environment. Worksheet is an implementation of MDI (Multiple Document Interface) concept. Different worksheets will be able to have different command consoles and different sets of active variables without interfering each other. Maple also provides facilities to create, save, and open worksheets. Therefore, user could save the unfinished worksheets into disk and retrieve them later.

Other important features include:

- Toolbar for quick access functions
- Worksheet Print function
- Font and style change function
- Spell check function
- Very details user manual

2.5.2 REDUCE

This review is based on REDUCE 3.4.1 for Sun Solaris.

2.5.2.1 General introduction

REDUCE is an interactive program designed for general algebraic computations of interest to mathematicians, scientists and engineers. Its capabilities include (Uni-koeln, 1995):

- Expansion and ordering of polynomials and rational functions.
- Substitutions and pattern matching in a wide variety of forms.
- Automatic and user controlled simplification of expressions.
- Calculations with symbolic matrices.
- Arbitrary precision integer and real arithmetic.
- Facilities for defining new functions and extending program syntax.
- Analytic differentiation and integration.
- Factorization of polynomials.
- Facilities for the solution of a variety of algebraic equations.
- Facilities for the output of expressions in a variety of formats.
- Facilities for generating optimized numerical programs from symbolic input.
- Calculations with a wide variety of special functions.
- Dirac matrix calculations of interest to high energy physicists.

REDUCE has following main characteristics (Uni-koeln, 1995):

- Code stability: The system's critical components are highly reliable, stable and efficient.
- Wide user base: Many existing packages and templates can be used to speed up problem solving.
- Full source code to the public: The entire code sources are all available to the public. It is a valuable education resource.
- Flexible updating: As all source code is available, the developers from all over the world are able to check and patch the mistakes in the source. The user could easily reach these patches from the web.
- State-of-art algorithms: Most importantly, the system has all the current best algorithms.
- Algebraic focus: The system aims at being part of a complete scientific environment rather than being the complete environment itself. As a result, users can take advantage of other state-of-the-art systems specializing in numerical and graphical calculations, rather than depend on just one system to provide everything.
- Portability;
- Uniformity: The system makes sure that calculations will not behave differently on different machines/environments/systems;
- Low cost

2.5.2.2 Linear algebra package in REDUCE

Same as Maple, a linear package is provided in REDUCE. To load the linear algebra package in REDUCE, type:

```
load_package linalg;
```

In REDUCE, to initialize a matrices needs the following command syntax:

```
mat1 := mat((a,b,c),(d,e,f),(g,h,i));
```

This will produce:

$$\text{mat1} := \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

To access the element of matrix at row *i* and column *j,* use the following command syntax

```
mat1(i,j)
```

(Matt Rebbeck., 1994)

The full details of REDUCE linear algebra package can be found at: http://www.uni-koeln.de/REDUCE/3.6/doc/linalg/

2.5.2.3 Interface Analysis

Comparing with Maple's graphical user interface, the user interface for REDUCE is much less user-friendly. To a beginner user, the command-Line interface provided is quit difficult to use. Without a user manual, it's almost impossible to perform any operations. The help function is also purely text based and have a very limit search function to get the right content.

In REDUCE version 3.8, the system comes with a recently developed graphical user interface called XR. However, it's still not a part of the normal REDUCE distribution.

The main feature of XR includes:

- Two dimensional graphical formula output.
- Command history editing with bracket matching
- Utility features such as file logging.
- LaTeX or PostScript printer output and user font selection.
- An info-based help system

[XR Graphical User Interface for REDUCE]

2.6 The Project

Generally, the ideal functionality of the final delivered system should include:

- The system needs to perform all basic linear transformation
- The data must be represented to the user in a clear form, and can be read and understood easily.
- The system needs to allow the user input data easily.
- The system needs to have a fast response to the user's input.
- The interface of the system needs to provide facilities that allow easy user interaction.
- The size of the software package must be small.
- The system needs to be easy to install.
- The user must be able to save the unfinished work into files and reload the work later.

- A help system and user manual must be provided.

Chapter 3

Requirements Analysis

3.1 Introduction

The literature review has enabled us to begin thinking about the issues that are important to the design of computer linear algebra system. In this chapter, the needs, expectations and goals of the users of the final system will need be identified. The requirements analysis process is needed for requirement specification that produces a stable set of requirements.

3.2 System Interface Requirements

3.2.1 The Importance of User Interface

The system interface should be considered as one of the most important parts of the system. The only way that users interact with system is through the user interface. Without a good interface, it will be difficult for user to access all the system functions thus could lead to a bad overall system performance.

3.2.2 The Comparisons of User Interfaces

Pure Command Line User Inter

A pure command line user interface is easy to be implemented. However, from the literature review, it's easy to see that a pure command line interface could not even provide the basic help information when it's used by a beginner user. In order to use a system will this kind of interface, user will need to learn the system first. Thus, the system will not be easy to use. Now it's obviously, a pure command line interface can't be the best option.

It's possible to develop a command line user interface with some graphical components like menu bars. These components could provide facilities like saving/loading, getting help information and etc. However, all the system feedback such like history commands, operation results or system prompts will still be displayed on the command line console. This will tend to be very difficult for the user to read. On the other hand, the main method for user to interact with system is still typing commands into the console, thus at least some command syntaxes must be memorised. For these reasons, the majority of beginner users will still find it hard to use.

Developing a graphic interface with Java is not easy. Java provides two packages -- AWT and Swing for creating graphical components. Both of packages are large and complex. However, with a graphical user interface, a user could easily access all information and performing any actions by clicking some mouse buttons. Various ways are left for the designer to decide the way of system-user interaction. Majority users will have few difficulties to user the system. In few cases, when the user encounters a problem, a graphical user help will always be helpful. It's easy to see a pure graphical user interface will be the ideal option for the beginner users.

Below is a summary of benefits that a graphical user interface will bring:

- System could be generally simple to use.
- User could input commands easily
- System feedback and prompts could be more readable and understandable.
- The interface could provide sufficient instructions for users to follow.
- Layout of the interface could be clear and concise.
- Graphical help function could be provided in order to give more detailed explanation of functionality.

3.3 Algorithm Requirements

The aim of this project is to design a basic linear algebra system that simplified analogies of usual computer algebra systems like REDUCE, Maple. The user of the system should be able to perform a number of linear algebra operations. From the literature review, to ensure the basic functionality, the below algorithm will need to be included:

- Algorithm for matrix addition
- Algorithm for matrix subtraction
- Algorithm for matrix scalar
- Algorithm for trivial matrix multiplication
- Algorithm for computing matrix trace
- Algorithm for computing matrix determinant
- Algorithm for computing matrix inverse
- Algorithm for computing matrix ad-joint
- Solving the systems of linear equations with Cramer's Rule algorithm

Other algorithms such like Gaussian Elimination, Gauss-Jordan Elimination for solving the systems of linear equations and Strassen's algorithm for fast matrix multiplication should also be implemented. However, lacking these algorithms will not affect the functionality of the system. Thus, the implementation of such algorithm should not be in the first priority.

The efficiency of algorithms should also be taken into account. During the implementation, the most efficient algorithm will be implemented at a high priority.

Facilities for algorithm error handling should be built into algorithms to ensure all exceptions are handled properly. To provide these facilities, auxiliary functions should be implemented to perform operations such like comparing matrix values, comparing matrix sizes, checking matrix characteristics and etc…

3.4 System Constraints

3.4.1 Software Constraints

The reason to choose Java as the implementation language is because Java is powerful object-oriented language with decent packages built in that support graphical interface development.

The Java virtual machine also ensures the portability of the system. There are very few operating systems that don't have a JVM support. This means, without compiling the source code for different operating systems, the user would be able to execute the final system binary at nearly all popular operating systems.

3.4.2 Hardware Constraints

The whole system is developed with Java. As the nature of Java Programming language, theoretically, the system should be able to be run on the top of Java virtual environment at any hardware platforms. However, during the implementation of the system, characteristic of different file systems on various hardware platforms should be also taken into account.

3.5 System Robustness Requirements

Obviously, the error handing mechanism in algorithm is important to the robustness of the system. However, the facilities for other general error handling could also have large impact on the robustness. The system is supposed to interact with all kinks of users. But users' inputs could be random and unexpected. For example: If an invalid matrix is inputted by the user, what should the system response? It would be a clue to show how robust the system is. Therefore, during the implementation, special care should be put onto dealing with random user inputs.

3.5 General Usability

Usability of the system mainly consists with three factors:

- How interactive the user interface is?
- How fast does the system response to user?
- How reliable the system is?

The user interface should provide facilities for good user interaction. A bad user interface could result a complete unusable system.

The response time is close related to the efficiency of the algorithm. But it is related to the design of data storage and interface components layout.

Obviously, an unreliable system will surely result an unusable system.

3.6 Data Input Requirements

3.6.1 What Kind of Data to Input

Data input is the way of interacting between users and the system. For this system, a user's input mainly consists three parts:

- The variable input. This involves variable input and variable data (i.e. matrix) input.
- The command input, which tells the system what type of operation to perform.
- Worksheet save/load. This involves loading/saving the worksheet.

3.6.2 The method of data input

Variable input

There aren't too many options for variable input method. To input a variable, typing a name for representing the variable is necessary. The value of the variable should also be typed into system. The alternative way to input variable is to input a file with all variable details outside from the system. However, this could result increasing the complexity of the system.

Command input

One method to input commands is to build the system with a set of built-in commands. User could type in the right command with suitable parameters to the command line console. Thus the system should perform corresponding operation. Maple uses similar method for user commands input. The advantage of this method is it has great flexibility. The system can have a very large set of commands built in to provide nearly unlimited functionalities. The disadvantage is also quite obvious. To use the built in command, the user will have to know them before starting to use the system. This conflicts with the "easy to use" objective.

The alternative method is to build the system with a drop-down menu or combo box. User could then select/choose the corresponding operation to perform and specify parameter variables. The advantage of this method is users can use all supported functions without checking the user manuals. The disadvantage is the system could only have a limit number of functionality. Otherwise there will be massive items in the drop-down menu/combo box. Therefore, it will be difficult for the user to access certain functions.

The implementation of the first method is quite complicated. It involves syntax parsing and grammar analysis. If this method is chosen, a lot of time will be spent on writing the code for parser. Therefore, less time will be able to be spent on other parts of the project. It will then be very difficult to deliver a working system within the time constraint.

The objective of this dissertation project only aim to deliver a system with limited linear algebra functions. A drop-down list would hold at least 20 items without causing the selection problem. The total supported functions of this system should be less than 20. Therefore, the less flexible weakness of the second method should not be a big problem. As this result, an implementation of the second method would be more appropriate.

Worksheet save/load

The concept of worksheet is borrowed from Maple. It will be useful to save the unfinished work to a file. The user can open the saved file later to carry on the previous work.

3.7 Testing Requirements

Testing part is to test the system against all the functionalities within the system. The testing phase is below:

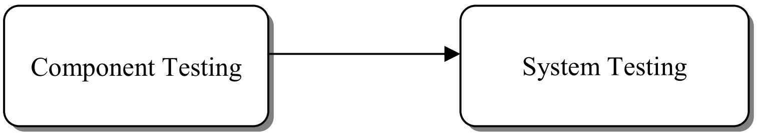
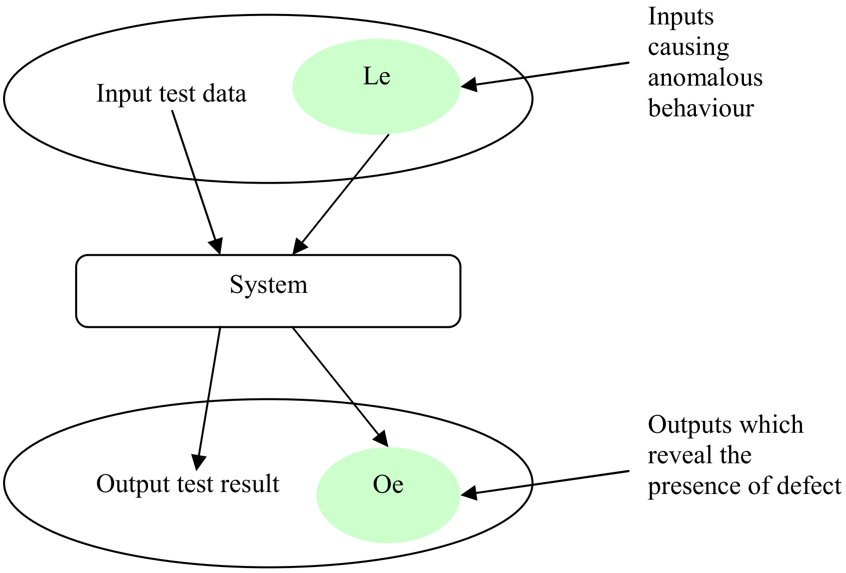


Figure 3. 2 Testing Phases

Component testing sometimes called unit testing is the process of testing individual components in the system. And unit testing have included white box testing and black box testing.


Input test data
Le
Inputs causing anomalous behaviour
System
Output test result
Oe
Outputs which reveal the presence of defect

(Sommerville, P538 P544)

Figure 3. 2 Black Box Testing

Integration testing is that where the test team have access to the source code of the system. Integration testing is mostly concerned with finding defects in the system.

System testing involves integrating two or more components that implement system functions or features and then testing this integrated system.
(Sommerville P540)

3.8 Training Requirements

No special training needs to be given before using the system. The user manual will be built into the system. Therefore, the user could check the user manual when a problem is encountered.

3.9 Development Resources and Constraints

3.9.1 The Development Resources

- One full time developer
- One experienced supervisor
- All computer resources provided by BUCS
- Two personal computers will Microsoft Windows operating system and Suse Linux 9.1 installed.
- Bath university library

3.9.2 The Development Constraints

The complete system with all development documents must be delivered before Monday, 16 May 2005.

Chapter 4

Requirement Specification

4.1 Introduction

This chapter is going to categorize the results that got from requirements analysis and produce a set of stable requirements.

4.2 Functional Requirements

The definition of functional requirements is:
Statements of services the system should provide, how the system should react to particular inputs how the system should behave in particular situation. In some cases, the functional requirements may also explicitly state what the system should not do. (Sommerville 2001)

1. The system should be able to perform the following linear algebra operations:
   - Matrices addition
     <u>Description:</u> For two matrices with the same sizes, the system should return the result. Error messages will return if the sizes are not the same.

   - Matrices subtraction
     <u>Description:</u> For two matrices subtraction with the same sizes, the system should return the result. Error messages return if the sizes are not the same.

   - Matrix multiplication
     <u>Description:</u> If column number of the first matrix is not same as the row number of the second matrix, this operation should not be performed. Error messages return if invalid matrix or incorrect information input. For two matrices, the sizes of them are not matched, and then the system could not return the result.

   - Matrix scalars
     <u>Description:</u> If a scalar number and a matrix input, result will return. Error messages return if invalid input found.

   - Matrix determinant
     <u>Description:</u> Return the determinant of a square matrix. Error messages return if no square matrix found.

   - Matrix inverse
     <u>Description:</u> Return the inverse of a non-singular square matrix**.** Error messages return if non-square matrix found.

   - Matrix ad-joint

Description: Return the ad-joint of a square matrix. Error messages return if non-square matrix found.

- Cramer's rule
  Description: Solving the system of linear equations that has a unique solution.Error messages return if no unique solution found.

- Gauss-Jordan Elimination
  Description: Return the reduced echelon form of a matrix.

2. The system should be able to store variables.
   Description: Matrix variables could be saved into a variable pool. The name and value of each variable should be recoded. The variable name and value should be able to be retrieved when needed.

3. The system should be able to delete variables.
   Description: Variables saved in the variable pool could be deleted.

4. The system should be able to display matrix properties.
   Description: Variable characteristics such like: matrix size, matrix value can be displayed.

5. The system should be able to create, save and load worksheet.
   Description: A worksheet contains data of current variable pool, and command history. Each worksheet can be saved as a file on the disk drive. This file can be opened to load the saved environment (i.e. variable pool and command history).

6. The system should provide facilities that allow user input command.
   Description: As discussed it the requirements analysis chapter. A drop down list should be used for command/operation selection. A dialog should appear for specifying the parameters for selected operation.

7. A history command should be provided.
   Description: It is used to display previous inputted commands. Certain functions like clearing the current command history should also be provided

8. The system should have a good error handing mechanism.
   Description: The system should be able to detect the error input caused by users. System should notify the user when an error is detected.

9. A user manual should come with the system.
   Description: Help could be provided when needed.

4.3 Non-Functional Requirements

The definition of functional requirements is:
These are constraints on the services or functions offered by the system.
(Summerville 2001)

1. The system should be easy to use.
   Description: The system should be easy to understand and follow without professional languages on the interface as an indicator.

2. The system should be robust.
   Description: The system should be running without any bugs, it could fully work under any environments to provide the nice functionality to users.

3. The system should be user friendly.
   Description: The layout of the system should be clear and well organized.

4. The system should be reliable.
   Description: The system should be able to run stably under different environment.

5. The system should response quickly.
   Description: The response time of the system should react as quickly as possible.

6. The system should be run at most hardware and software environment.
   Description: The system should be able to run at any JVM

4.4 Miscellaneous requirements

1. Documentation Requirements:
   - Abstract
   - Acknowledgement
   - Literature survey
   - Requirement analysis
   - Requirement specification
   - Design
   - Detailed Design and implementation
   - System testing
   - Evaluation
   - Conclusion
   - Bibliography
   - Appendices

2. Time constraints.
   Description: The deadline is set on 16th May, 2005.

3. Implement language requirement.
   Description: The system should be implemented with Java

4. Training requirement.
   Description: No special training needed. However a user manual should be provided.

5. The development resources constraints includes:
    - One full time developer
    - One experienced supervisor
    - All computer resources provided by BUCS
    - Two personal computers will Microsoft Windows operating system and Suse Linux 9.1 installed.
    - Bath university library

6. Maintenance requirements
    Description: The maintenance should be carried out by the coded and any maintenance patches, or upgrades will be distributed to the users.

7. System delivery requirements
    Description: The system should be delivered on a disk including the uncompelled source code and a ready to run version of the system. The binary should be packed into executable JAR file.

8. System testing Requirements
    Description: Unit testing and System testing should be included.

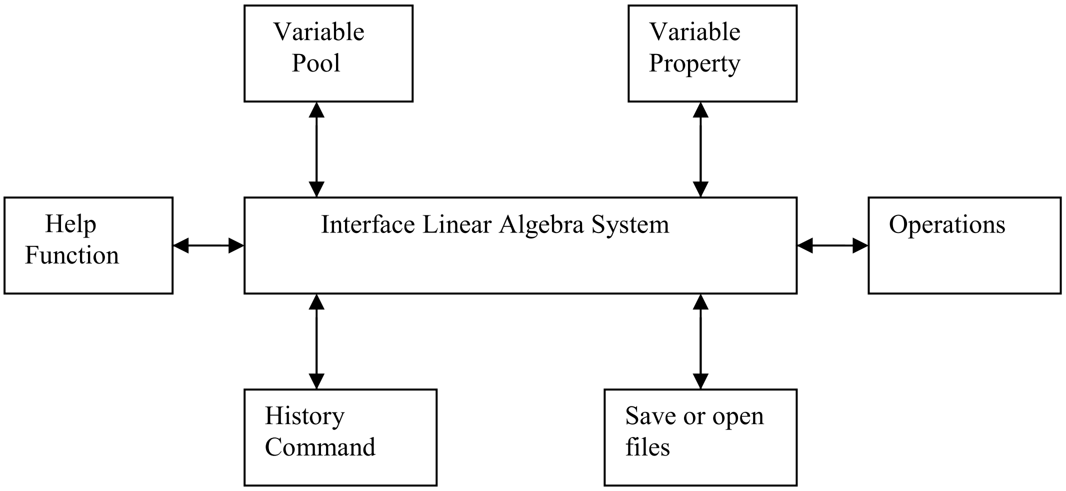
Chapter 5

Design

This section is going to outline the design ideas of computer linear algebra system. The project will focus on designing an interactive interface. How to design the interface will detail in this chapter rather than matrix algorithms.

5.1 Overall Design

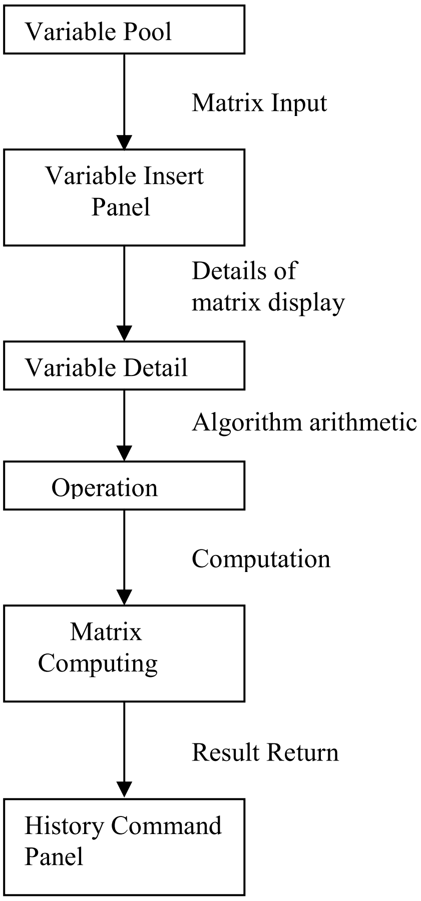
Overall design will consider the whole computer linear algebra system as a whole.



This is an overview of the linear algebra system as a whole. It contains Variable Pool that for users input the matrix, Variable Property which can show the details of the matrix, Operations which are based on linear algebra mathematical background, Save or Open files which is for users save the matrix or open a file on the interface, history command which will display the result of computing, Help function which could help the users who are not familiar with the system.

5.2 Function flow model

The functions on the interface link each other; it could be shown as the model below:



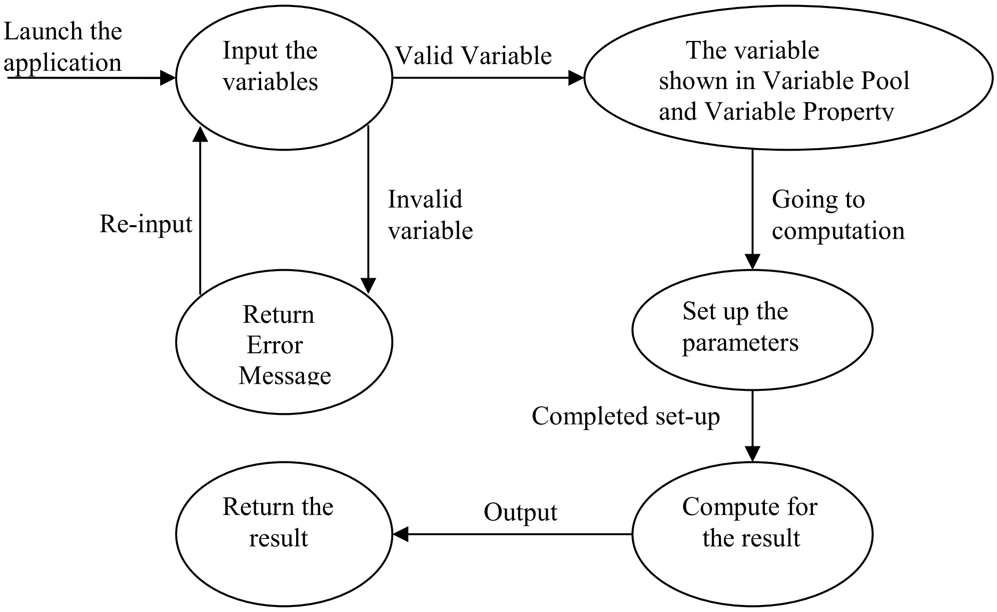
5.3 Data-flow models

Data-flow models are an intuitive way of showing how data is processed by a system. At the analysis level, they should be used to model the way in which data is processed in the system. The notation used in these models represents functional processing, data stores and data movements between functions. (Ian Summerville 2001)

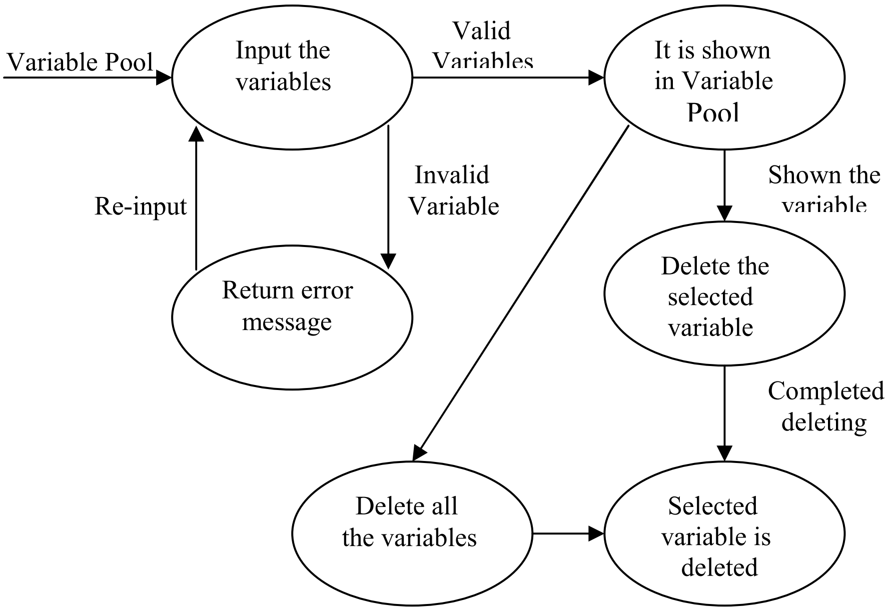
Data-flow models are used to show how data flows through a sequence of processing steps. The data-flow model will be described in this project as due to its characteristic, the data is transformed at each step before moving on to the next stage.

Data-flow models are valuable because tracking and documenting how the data associated because tracking and documenting how the data associated with a particular process moves through the system helps analysts understand what is going on. (Ian Summerville 2001)

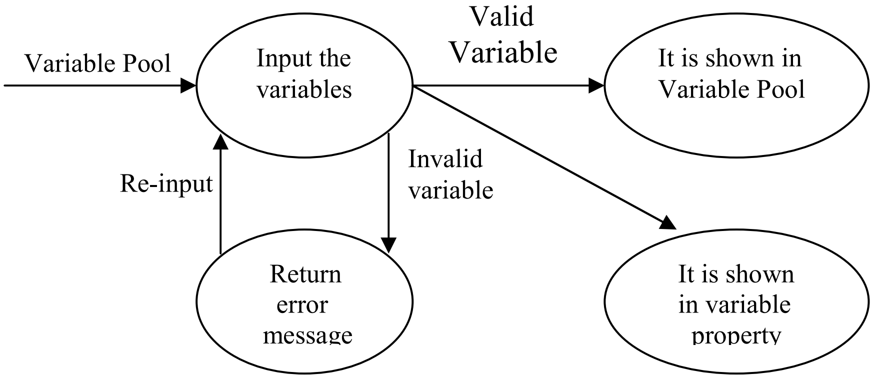
In this case, data-flow models will be involved to describe each stage data transformed. It will give the reader another perspective of the system, how the system is working internally.



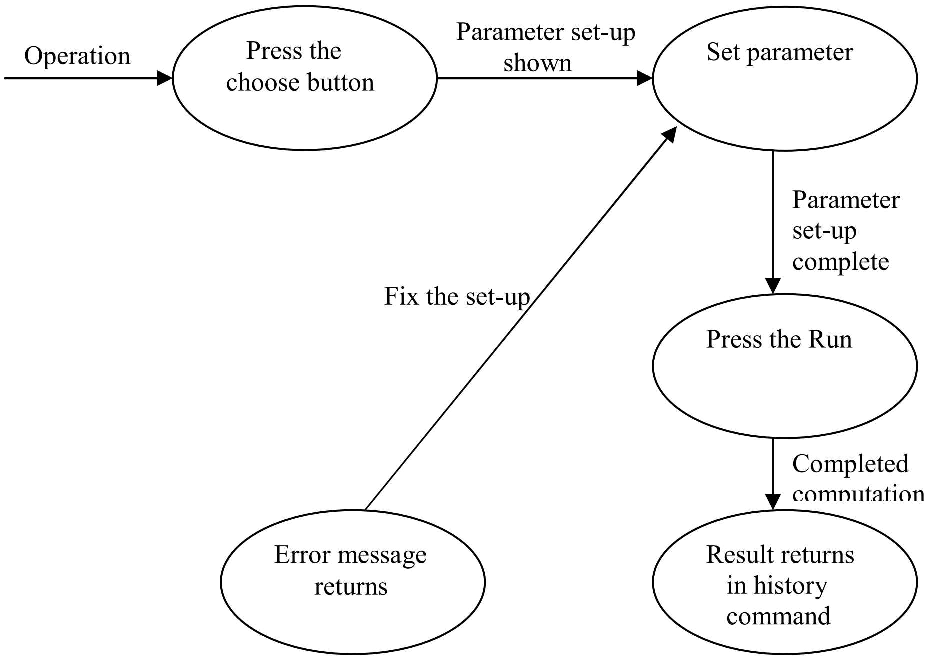
Data-flow model shows a functional perspective where each transformation represents a single function. This data-flow model shows an overview of the system working internally. It is showing how each function interacts with each other.



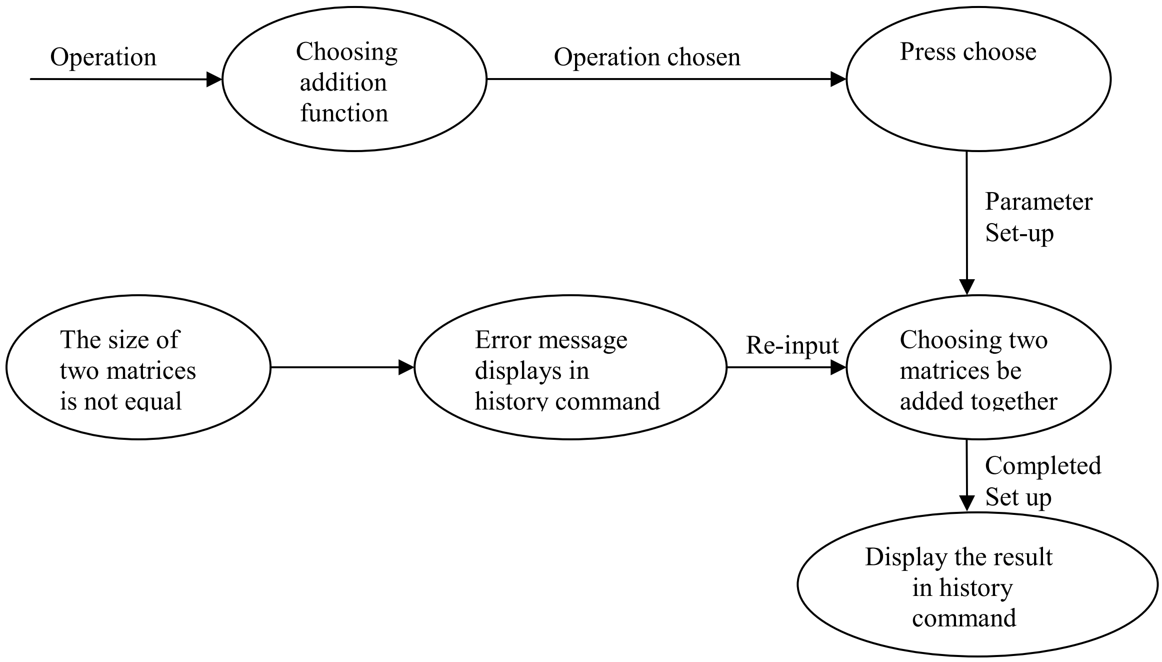
This data-flow model shows how the Variable Pool function works. There are two choices could be applied, delete the selected variable and delete all the variables in the Variable Pool.



This data-flow model shows how the Variable Property works. The input variable will be simultaneously appearing in Variable Pool and Variable Property.



This data-flow model shows how the Operation function works. The error messages return when the invalid matrix input in order to be fixed by users.


Operation
Choosing addition function
Operation chosen
Press choose
Parameter Set-up
The size of two matrices is not equal
Error message displays in history command
Re-input
Choosing two matrices be added together
Completed Set up
Display the result in history command

This data-flow model shows procedure how the addition operation works. In the process, the error message is provided by the interface for correcting the error being made.

5.4 Class design

This section will mention the main classes will be used in the implementation. The classes contain start class, menu class, console class, help class, MatrixTools class, MatrixAlgorithm class, operationDialog class, variablePool class, variableInputDialog class, mainPanel class, variableTablePanel class.

- start.class: start the hole application
- menu.class: menu items
- console.class: hold the mainframe and menu bar
- help.class: hold the help file
- MatrixTools.class: provide support functions like analysis user input
- MatrixAlgoritm.class: hold all the algorithm
- operationDialog.class: display dialog for specifying the input and result data
- variablePool.class: the hold actual variable objects
- variableInputDialog.class: the dialog for user to input variables
- mainPanel holds the main interface components including variable pool panel, variable details panel, operation choose panel, history command panel
- variableTablePanel.class display the table of variables, the variable details panel and also hold the table model
- Matrix.class: the new matrix data type

Chapter 6

Detailed Design and Implementation

6.1 The New Data Class – Matrix

By default, Java does not contain any packages that provide fully matrix support. It is important and necessary to design a new data type that provides the facilities to store and represent the proprieties of matrices. Because of the natural of Java programming language (i.e. Java Virtual Machine prevents the program access memory directly), it's nearly impossible to create a new complete data type as direct memory access is forbidden.

The most convenient way to design a new data type is to design a Matrix class by associating to an existing data type. There are several options left. Other data types like "int" and "long" only support integer storage. "float" is another possibility but it is less accurate than "double".

In Java specification, "double" is defined as 64-bit IEEE 754 floating-point value and "float" is defined as 32-bit IEEE754 floating point value [*Learning Java*]. The "double" data type should be a good choice for this system. This should be accurate enough for this project whose main users are supposed to be students.

The Matrix class will be one of the most important components of the system. Every variable is going to be stored in this format. The actual data of each matrix will be represented as a two dimensional double array. Other methods should be provided for manipulating and accessing the properties of each matrix.

One more detail will also need to be taken care of. In Java, the array index starts from zero. However, in the world of linear algebra, the matrix row and column number are counted from one. This problem should be taken into account during the implementation.

6.1.1 Matrix Data Representation - The Data Field

To represent matrix of real numbers, two dimensional array should be an ideal method. The element of each matrix could be easily allocated by giving a row index number and column index number.

The data field in the Matrix class will need to include these fields:

- row number
- column number
- value

"row number" is an integer holding the number of rows of the matrix. "column number" is an integer holding the number of columns of the matrix. Both of them could be used in the error checking. “value” is a two dimensional double array that holds the actual value of the matrix.

6.1.1 Building a Matrix – The Constructor

It is necessary to define different constructors for the Matrix class. Matrices are built by providing different parameters to the constructor.

The default constructor will create a 1 by 1 matrix that has value 0. The algorithm is:

```
matrix( ){
      row number = 1;
      column number = 1;
      value = 0;
}
```

Pass a two dimensional double array to the constructor. This will build a matrix that has the same value as the array. The algorithm is:

```
Matrix(data){
      row number = row number of data;
      column number = column number of data;
      value = data;
}
```

Pass a row number and a column number to the constructor. This will build a matrix that with given row and column number and an initial value 0 for every element. The algorithm is:

```
Matrix(row, column){
      row number = row;
      column number = column;
      value = 0;
}
```

Pass a row number, a column number and an initial value to the constructor. This will build a matrix with given row and column number with a given initial value for every element. The algorithm is:

```
Matrix(row, column, initial){
      row number = row;
      column number = column;
      value = initial;
}
```

6.2.2 Matrix Properties Retrieve

From time to time, properties of a matrix need to be retrieved. It is important to have access to various matrix properties without analysing the value of matrix every time. Thus the following methods are built in to serve this purpose.

Get the row number of a matrix. The algorithm is:

```
int getRowNumber( ){
      return row number;
}
```

Get the column number of a matrix. The algorithm is:

```
int getColumnNumber( ){
      return column number;
}
```

Get the element value of a matrix. By specifying the element by giving a row index number and a column index number, this method should return the value of the element. The algorithm is:

```
double getValue(row, col){
      return value[row-1][col-1];
}
```

Set the element value of a matrix. By giving a row index number and a column index number and a value, assign the corresponding matrix element with the given value. The algorithm is:

```
setValue(row, col, v){
      value[row-1][col-1] = v;
}
```

Check if a matrix is square matrix. Return true if a matrix is square, otherwise return false. The algorithm is:

```
boolean isSquareMatrix(){
      if(row number == column number){
            return true;
      }
      else{
            return false;
      }
}
```

6.2.3 Matrix Manipulating

Methods for manipulating matrices are necessary. They could be useful while performing some matrix operations such like determinant calculation and elementary row operations. Methods describe in this section provide these facilities:

- Access sub-matrices by specifying or deleting certain row/column index numbers.
- Row and column swapping.

Get the row vector of a matrix. By specifying the row index number, this method should return the corresponding row vector. The algorithm is:

```
double[] getRow(row){
      return value[row-1];
}
```

Get the column vector of a matrix. By specifying the column index, this method should return the corresponding column vector. The algorithm is:

```
double[] getColumn(col){
      for each row
            result[current row] = value[current row][col-1];
      end for
      return result;
}
```

Swap two rows of a matrix by giving two row index numbers. The algorithm is:

```
swapRow(rowOne, rowTwo){
    tempRow = getRow(rowOne);
    value[rowOne-1] = getRow(rowTwo);
    value[rowTwo-1] = tempRow;
}
```

Swap two columns of a matrix by giving two row index numbers. The algorithm is:

```
swapColumn(colOne, colTwo){
    tempColumn = getColumn(colOne);
    for each row
        value[current row][colOne-1] = value[current
        row][colTwo-1];
        value[current row][colTwo-1] = tempColumn[current
        row];
    end for
}
```

Remove a row of a matrix by giving the row index number. This should return a matrix with the specifying row deleted. The algorithm is:

```
Matrix removeRow(r){
    for each row to r
        for each column
            tempValue[current row][current column] =
            value[current row][current column];
        end for
    end for

    for each row from r-1
        for each column
            tempValue[current row][current column] =
            value[current row+1][current column];
        end for
    end for

    return new Matrix(tempValue);
}
```

Remove a column of a matrix by giving the column index number. This should return a matrix with the specifying column deleted. The algorithm is:

```
Matrix removeColumn(c){
    for each row
        for each column to c-1
            tempValue[current row][current column] =
            value[current row][current column];
        end for
    end for

    for each row
        for each column from c-1
            tempValue[current row][current column] =
            value[current row][current column+1];
        end for
    end for
```

```
        return new Matrix(tempValue);
}
```

6.2.3 Auxiliary Functions

Methods described in this section will provide facilities to:

- Compare element values of two matrices
- Compare size of two matrices
- Check whether a two dimensional array is legal to construct a matrix
- Produce a new matrix copy

The mechanism of Java object cloning is not suitable for performing a complete copy of objects ([10]). Thus, the matrix copy method could be particularly useful to create a new instance of matrix that need to be identical to another matrix in terms of value. The rest methods will be useful while performing error handing/checking.

Check if two matrices are identical (i.e. all corresponding elements of two matrix are equal). The algorithm is:

```
boolean isEqualTo(matrix){
        for each row
                for each column
                        if (matrix[current row][current column] !=
                        value[current row][current column]){
                                return false;
                        }
                }
        }
        return true;
}
```

Check if two matrices have the same size. Return true if the size is same, otherwise false. The algorithm is:

```
boolean isSameSize(matrix){
        if(matrix.getRowNumber() == row nummber &&
        matrix.getColumnNumber() == column number){
                return true;
        }
        else{
                return false;
        }
}
```

Check if a two dimensional double array can be used to construct a matrix. This is mainly for error checking while constructing a matrix. The algorithm is:

```
boolean isMatrix(data){
        if(data == null){
                return false;
        }
        else if(the column size in each row not the same){
                reuturn false;
        }
        else{
            return true;
        }
}
```

Create a new matrix by copying the entire elements of a matrix. The algorithm is:

```
Matrix copy(){
        for each row
            for each column
                temp[current row][current column] =
                value[current row][current column];
            end for
        end for

        return new Matrix(temp);
}
```

6.2 The Algorithms for Matrix Operations

In this section, the details of each algorithm will be analysed and implemented with pseudo code.

6.2.1 Matrix Addition and Subtraction

The algorithm for matrix addition and subtraction are similar, thus implementation of one algorithm will be sufficient to cover the other one.

Matrix addition and subtraction: → Matrix
Pre-condition: the size of two given matrix is same
Post-condition: return the result of addition/subtraction

```
Matrix matrixAddition(matrixOne, matrixTwo){
        for each row
            for each column
                matrixThree[current row][current column] =
                matrixOne[current row][current column] +
                matrixTwo[current row][current column];
            end for
        end for
        return matrixThree;
}
```

6.2.2 Matrix Scalar

Matrix scalar: → Matrix
Pre-condition: input one matrix
Post-condition: return the matrix after scalar

```
Matrix matrixScalar(matrixOne, scalar){
        for each row
            for each column
                matrixTwo[current row][current column] =
                matrixOne[current row][current column]*scalar;
            end for
```

```
      end for

      return matrixTwo;
}
```

6.2.3 Matrix Multiplication

Matrix multiplication: → Matrix
Pre-condition: input two matrix, matrixOne and matrixTwo. The column number of matrixOne and row number of matrixTwo must be equal.
Post-condition: return the matrix after multiplication

```
Matrix matrixMultiplication(matrixOne, matrixTwo){
      for each row
            for each column
                  for k from 1 to column number
                        matrixThree[current row][current column]
                        = matrixThree[current row][current
                        column]+ matrixOne[current
                        row][k]*matrixTwo[k][current column];
                  end for
            end for
      end for

      return matrixThree;
}
```

6.2.4 Matrix Transpose

Matrix transpose: → Matrix
Pre-condition: input one matrix
Post-condition: return the transposed matrix

```
Matrix matrixTranspose(matrixOne){
      for each row
            for each column
                  matrixTwo[current row][current column] =
                  matrixOne[current column][current row];
            end for
      end for

      return matrixTwo;
}
```

6.2.5 Determinant of Matrix

Matrix determinant: → double
Pre-condition: input a square matrix
Post-condition: return determinant of input matrix

```
double matrixDeterminant(matrixOne){
      if(matrixOne.rowNumber == 1 & matrixOne.columnNumber==
      1){
            determinant = matrixOne[1][1];
      }
```

```
        else if(matrixOne.rowNumber == 2 & matrixOne.columnNumber
        == 2){
                determinant = matrixOne[1][1]*matrixOne[2][2] -
                matrixOne[1][2]*matrixOne[2][1];
        }

        else{
                for each column
                        matrixTwo =
                        matrixOne.removeRow(1).removeColumn(current
                        column);
                        determinant = determinant + (-1)^(1 + current
                        column) * matrixOne[1][current column] *
                        matrixDeterminant(matrixTwo);
                end for
                }
        }
        return determinant;
}
```

6.2.6 Ad-joint Matrix

Matrix adjoint: → Matrix
Pre-condition: input a square matrix
Post-condition: return the adjoint matrix

```
Matrix matrixAdJoint(matrixOne){
        for each row
                for each column
                        matrixTemp =
                        matrixOne.removeRow(i).removeColumn(current
                        column);
                        matrixTwo[current row][current column] = (-
                        1)^(1 + current column) *
                        matrixDeterminant(matrixTemp);
                end for
        end for
        return matrixTranspose(matrixTwo);
}
```

6.2.7 Matrix Inverse

Matrix inverse: → Matrix
Pre-condition: input a square matrix whose determinant is not equal to zero
Post-condition: return the inverse matrix

```
Matrix matrixInverse(matrixOne){
        if(matrixDeterminant(matrixOne) == 0){
                return null;
        }
        else{
                matrixTwo = matrixScalar(matrixAdJoint(matrixOne),
                1/matrixDeterminant(matrixOne));
        }
```

```
    return matrixTwo;
}
```

6.2.8 Matrix Trace

Matrix trace: → double
Pre-condition: input a square matrix
Post-condition: return the matrix trace

```
double matrixTrace(matrixOne){
    for each row
        trace = trace + matrixOne[current row][current row];
    end for
    return trace;
}
```

6.3 Solving the System of Linear Equations

In this selection, two algorithms commonly used for solving the system of linear equations will be analysed. The implementation is in pseudo code.

6.3.1 Cramer's Rule

Matrix determinant: → Matrix
Pre-condition: input a coefficient matrix and a constant matrix. The coefficient matrix must be square matrix. The constant matrix must be a matrix with a single column. The row number of coefficient matrix and constant matrix must be same
Post-condition: return the matrix of solutions

```
Matrix matrixCramersrule(matrixCoefficient, matrixConstant){
    det = matrixDeterminant(matrixCoefficient);
    if(det == 0){
        return null;
    }
    for i from 1 to row number of matrixCoefficient
        matrixTemp = matrixCoefficient.copy();
        for j from 1 to row number of matrixCoefficient
            matrixTemp[j][i] = matrixConstant[j][1];
        end for
        matrixSolution[i][1] =
        matrixDeterminant(matrixTemp)/det;
    end for

    return matrixSolution;
}
```

6.3.2 Gauss-Jordan Elimination

The algorithm of Gauss-Jordan Elimination mainly involves getting the reduced echelon form of the augmented matrix. The augmented matrix is obtained by combining the coefficient matrix and the constant matrix. Below are the three sub-algorithms for Gauss-Jordan Elimination implementation.

6.3.2.1 Combine Coefficient and Constant Matrix

Matrix combine: → Matrix
Pre-condition: input a coefficient matrix and a constant matrix. The coefficient matrix must be square matrix. The constant matrix must be a matrix with a single column. The row number of coefficient matrix and constant matrix must be same
Post-condition: return the augmented matrix

```
Matrix matrixCombine(matrixCoefficient, matrixConstant){
      for i from 1 to matrixCoefficient.getRowNumber()
            for j from 1 to matrixCoefficient.getColumnNumber()
                 matrixCombined[i][j] = matrixCoefficient[i][j];
            end for
      matrixCombined[i][matrixCoefficient.getColumnNumber()+1]
      = matrixConstant.getValue(i,1);
      end for

      return matrixCombined;
}
```

6.3.2.2 Get the Echelon Form of Augmented Matrix

Convert a matrix to Echelon Form by applying a sequence of row operations. This method should be placed at the Matrix class.

Matrix combine: → Matrix
Pre-condition: None
Post-condition: return the echelon form matrix

```
Matrix toEchelonForm(){
      Matrix matrixTemp = this.copy();
      MatrixTools tool = new MatrixTools();

      for m from 1 to matrixTemp.getRowNumber()
            for i from m to matrixTemp.getRowNumber()
                  temp = 1;
                  while(temp <= matrixTemp.getColumnNumber()&&
                  matrixTemp[i][temp] == 0){
                        temp++;
                  }
                  if(temp <= matrixTemp.getColumnNumber()){
                        p = matrixTemp[i][temp];
                        for j from temp to
                        matrixTemp.getColumnNumber()
                              q = matrixTemp[i][j]/p;
                              matrixTemp[i][j] =
                              matrixTemp[i][j]/p;
                        end for
                  }
            end for

            temp = 1;
            while(temp <= matrixTemp.getColumnNumber()&&
            matrixTemp[m][temp] == 0){
                  temp++;
```

```
        }
        if(temp <=matrixTemp.getColumnNumber()){
            for i from m+1 to matrixTemp.getRowNumber()
                if(matrixTemp[i][temp] != 0){
                    p =
                    matrixTemp[i][temp]/matrixTemp[m][t
                    emp];
                    for j from temp to
                    matrixTemp.getColumnNumber()
                        matrixTemp[i,j] =
                        matrixTemp[i][j] -
                        p*matrixTemp[m][j];
                    end for
                }
            end for
        }
    end for

    for i from 1 to matrixTemp.getRowNumber()
        pivotColumnIndex = 1;
        found = false;
        while(found == false & pivotColumnIndex <=
        matrixTemp.getColumnNumber()){
            if(matrixTemp[i][pivotColumnIndex] == 1){
                found = true;
                pivotIndexArray[i-1] = pivotColumnIndex;
            }
            else if(pivotColumnIndex ==
            matrixTemp.getColumnNumber()){
                pivotIndexArray[i-1] =
                matrixTemp.getColumnNumber()+1;
                found = true;
            }
            else{
                pivotColumnIndex++;
            }
        }
    end for

    for i from 0 to pivotIndexArray.lenth-1
        tempIndex = i;
        for j from i+1 to pivotIndexArray.length

    if(pivotIndexArray[j]<pivotIndexArray[tempIndex]){
                tempIndex = j;
            }
        end for
        temp = pivotIndexArray[i];
        pivotIndexArray[i] = pivotIndexArray[tempIndex];
        pivotIndexArray[tempIndex] = temp;
        matrixTemp.swapRow(i+1,tempIndex+1);
    }
    return matrixTemp;
}
```

6.3.2.3 Get the Reduced Echelon Form of Augmented Matrix

This method should be placed at the Matrix class.

Matrix combine: → Matrix
Pre-condition: None
Post-condition: return the reduced echelon form matrix

```
Matrix toReducedEchelonForm(){
      Matrix matrixTemp = this.toEchelonForm().copy();
      for i from 1 to matrixTemp.getRowNumber()
            pivotColumnIndex = 1;
            found = false;
            while(found == false & pivotColumnIndex <=
            matrixTemp.getColumnNumber()){
                  if(matrixTemp[i][pivotColumnIndex] == 1){
                        found = true;
                        for j from 1 to i

      if(matrixTemp[j][pivotColumnIndex] != 0){
                                    p =
                                    matrixTemp[j][pivotColumnIndex
                                    ]/matrixTemp[i][pivotColumnInd
                                    ex];
                                    for k from pivotColumnIndex to
                                    matrixTemp.getColumnNumber()
                                                matrixTemp[j][k] =
                                                matrixTemp[j][k] -
                                                p*matrixTemp[i][k];
                                    end for
                              }
                        end for
                        for j from i+1 to
                        matrixTemp.getRowNumber()

      if(matrixTemp[j][pivotColumnIndex] != 0){
                                    p =
                                    matrixTemp[j][pivotColumnIndex
                                    ]/matrixTemp[i][pivotColumnInd
                                    ex];
                                    for k from pivotColumnIndex to
                                    matrixTemp.getColumnNumber()
                                          matrixTemp[j][k] =
                                          matrixTemp[j][k] -
                                          p*matrixTemp[i][k];
                                    end for
                              }
                        end for
                  }
                  else if(pivotColumnIndex ==
                  matrixTemp.getColumnNumber()){
                        found = true;
                  }
                  else{
                        pivotColumnIndex++;
```

```
                }
            }
      end for
      return matrixTemp;
}
```

6.4 The Interface components and MVC

6.4.1 The Detailed Interface Design

One of the objectives of the project is to providing an interactive interface. This section will detail how the system is being built up.

The main screen

The main screen is the basic interface of the system. Every component could be accessed from here.

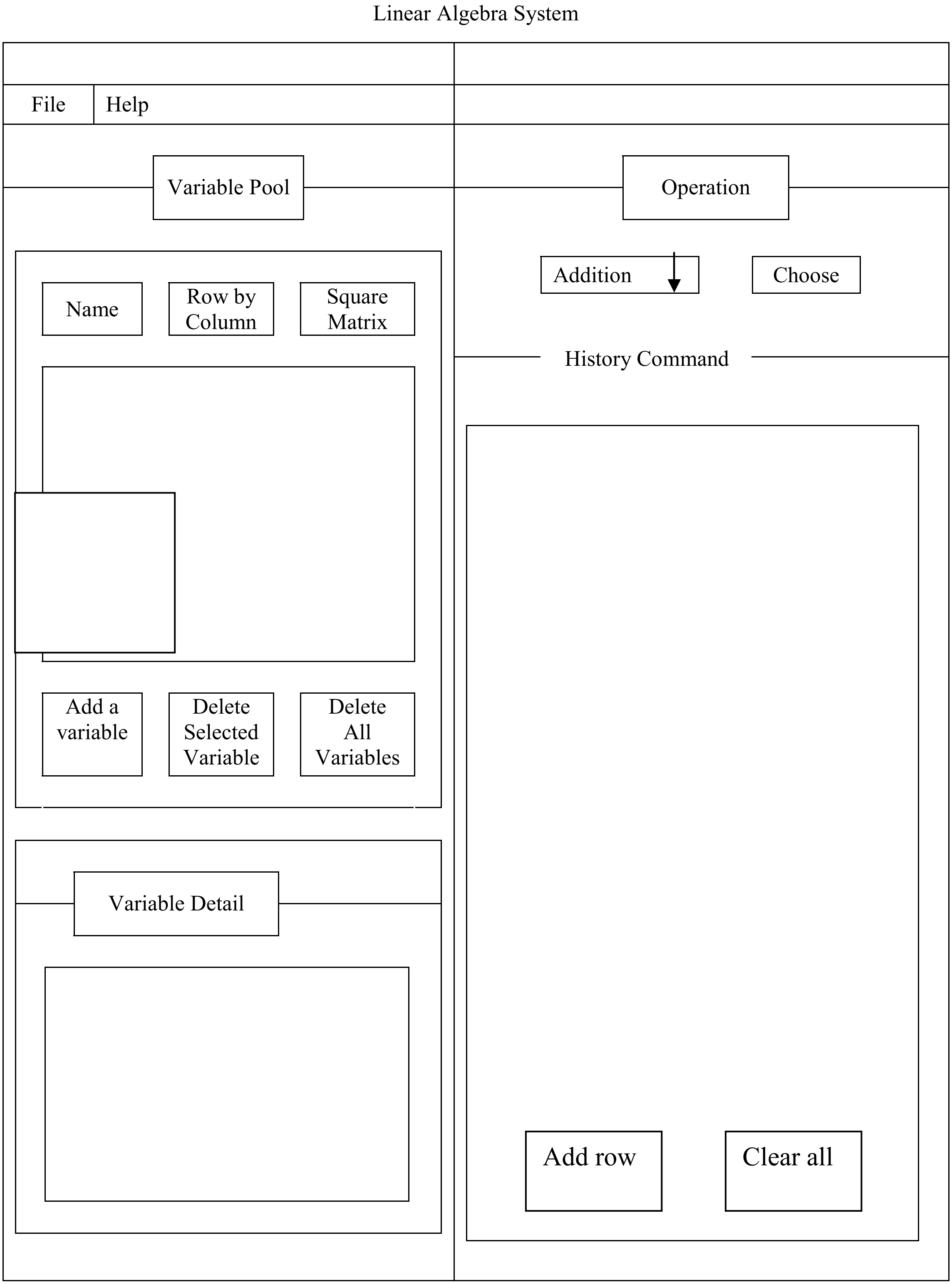
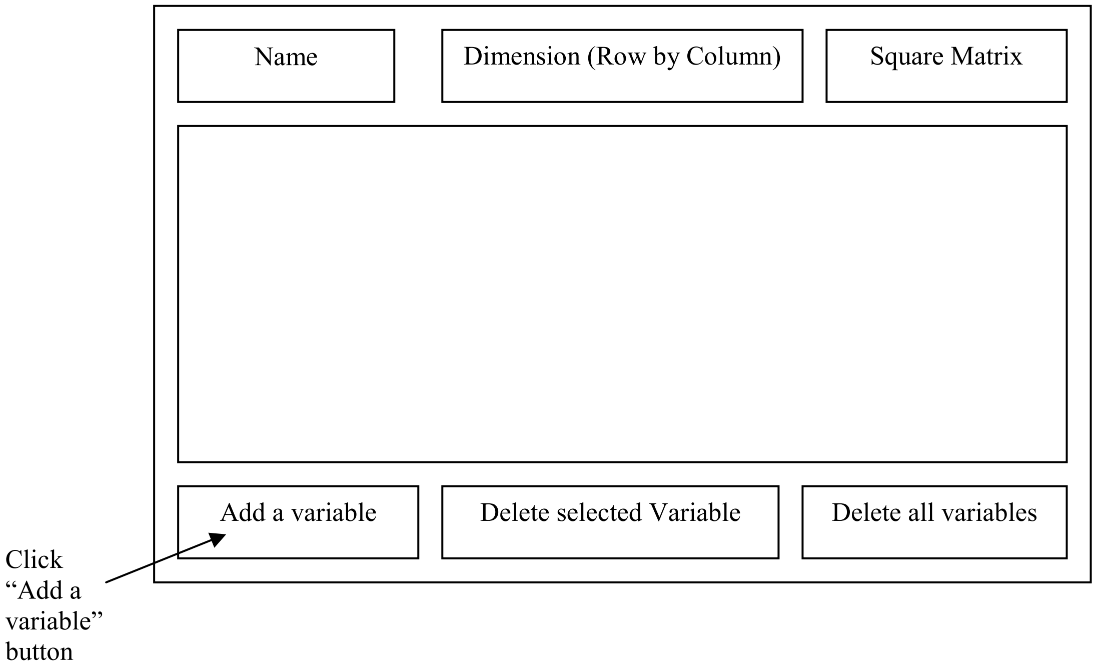


Figure 6. 1

Variable Pool

The variable pool panel is used to add or remove variables.



Click “Add a variable” button

Figure 6. 2

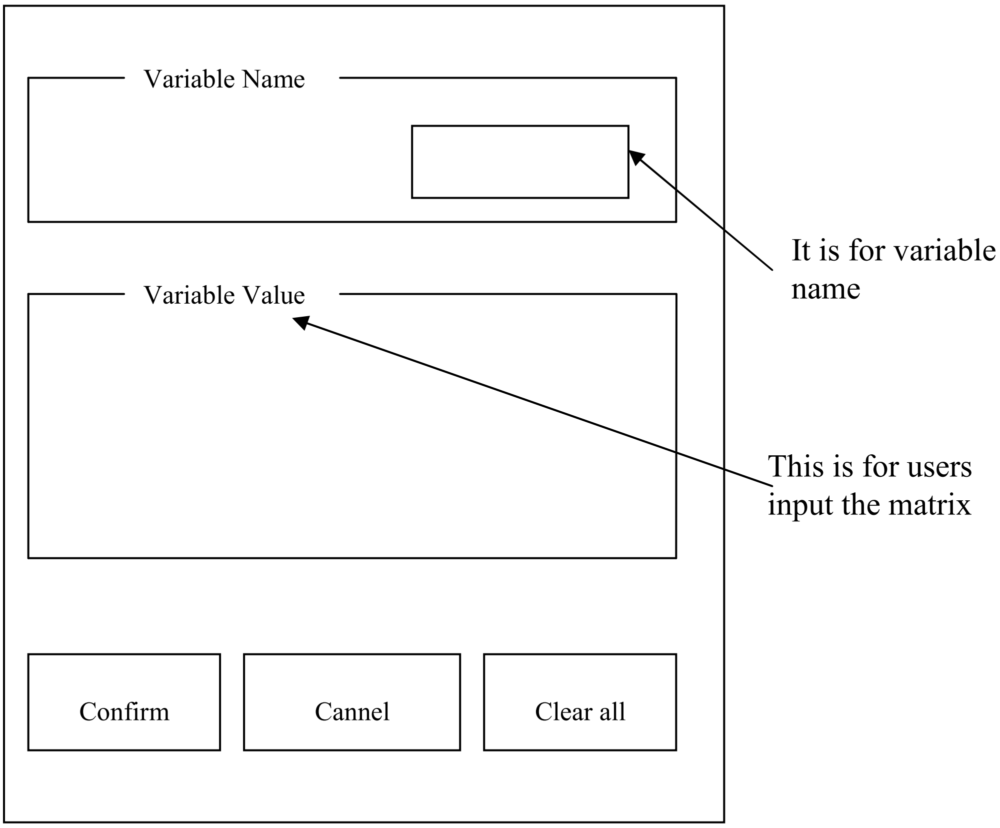
After clicking “add a variable” the button, the Variable Insert Panel will display for users to input the details of the variable.

After clicking “Delete selected variable” button, the selected variable will be deleted

After clicking “Delete all variables” button, all variables in the variable pool will be deleted

Variable Insert Panel

This is the Variable Insert Panel for which people could insert the details of a variable. If there are no errors, the details of the specific variable will be saved and displayed in variable pool. If an input error is detected, the system will prompt for warning information.



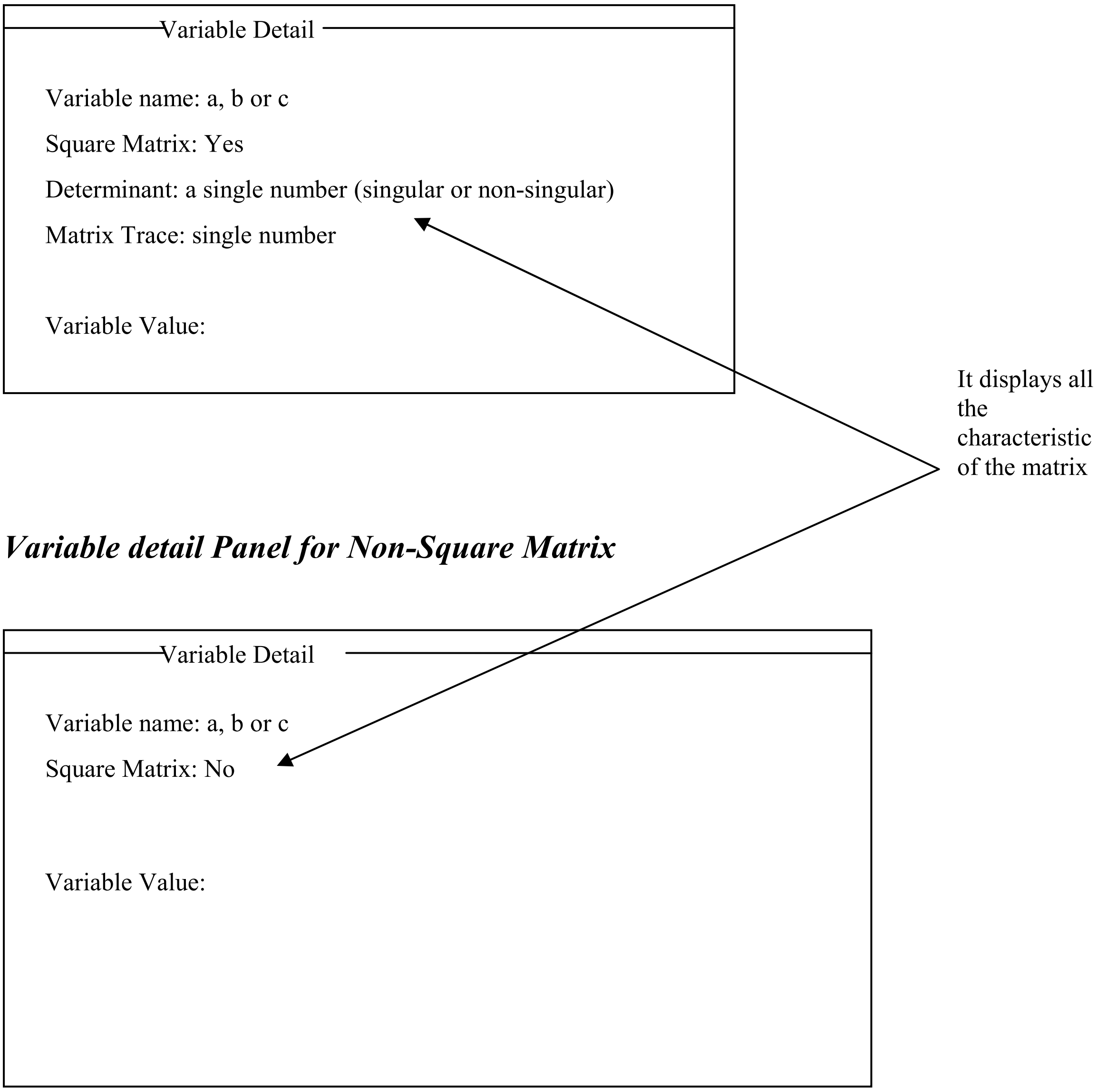
It is for variable name

This is for users input the matrix

Figure 6. 3

Variable Detail Panel for Square Matrix

This panel will display the details of selected variable.



Variable Detail

Variable name: a, b or c

Square Matrix: Yes

Determinant: a single number (singular or non-singular)

Matrix Trace: single number

Variable Value:

It displays all the characteristic of the matrix

Variable detail Panel for Non-Square Matrix

Variable Detail

Variable name: a, b or c

Square Matrix: No

Variable Value:

Figure 6. 4

Operation Panel

In this panel, there are two functions have been included.
One is operation choosing, the other one is history command which could display the result of a certain operation under matrix.

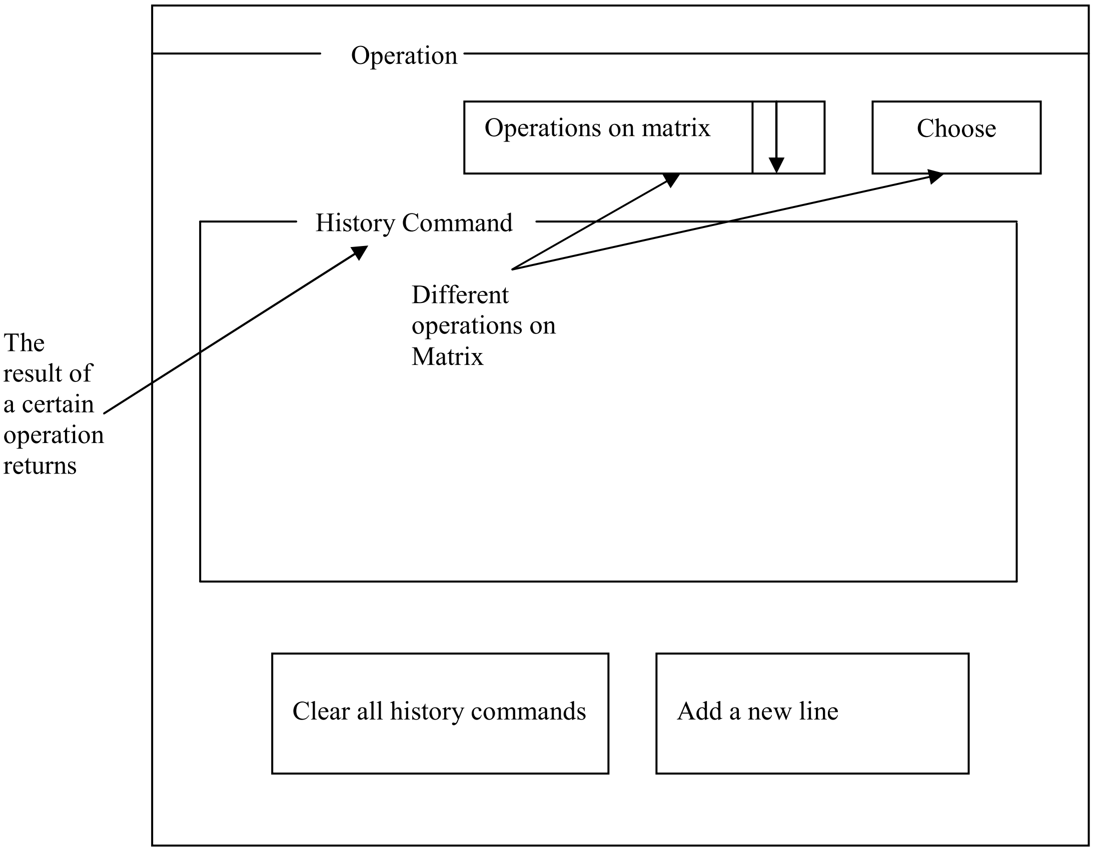
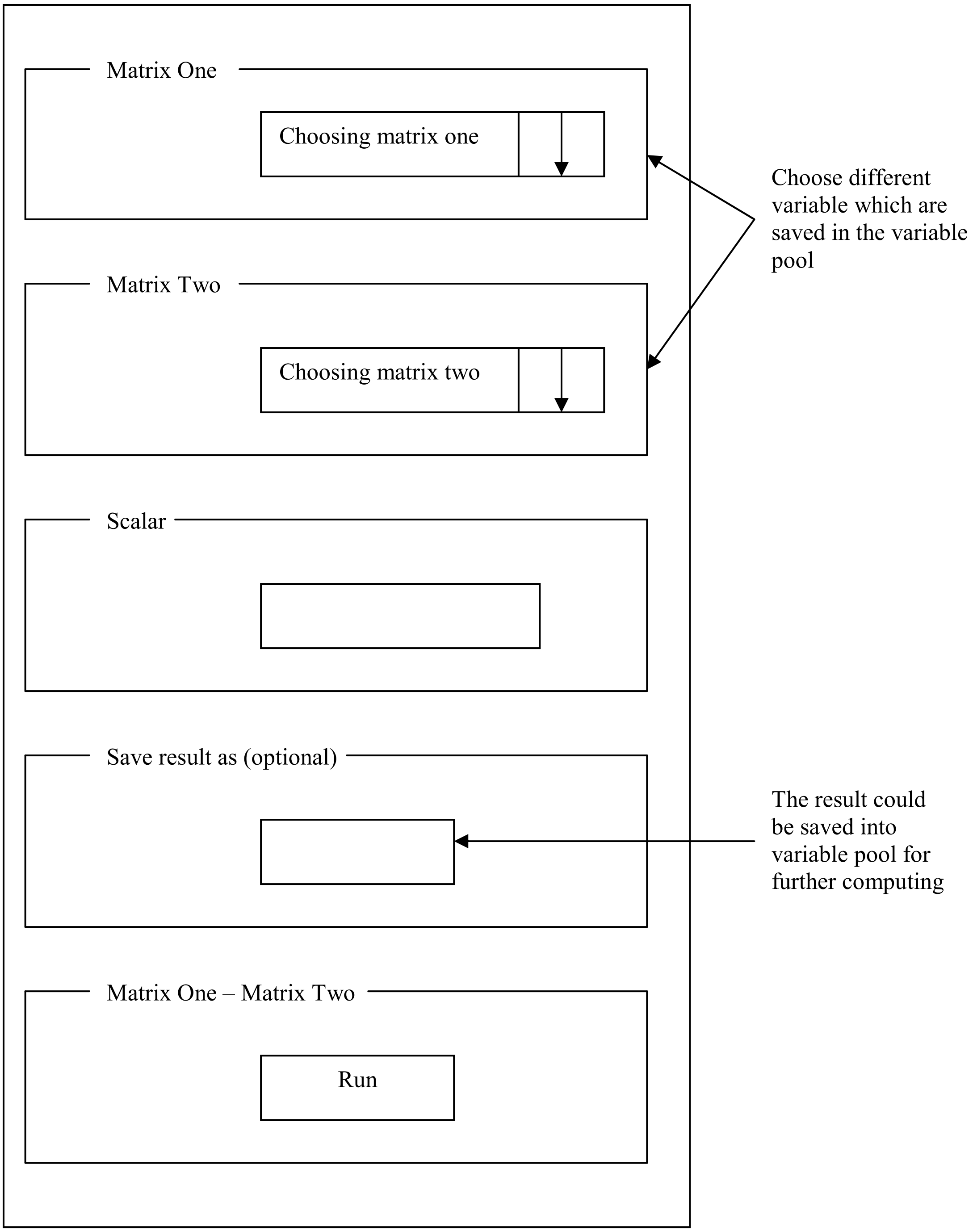

Operation
Operations on matrix
Choose
History Command
Different operations on Matrix
The result of a certain operation returns
Clear all history commands
Add a new line

Figure 6. 5

Parameter set-up panel

This panel is for parameters of matrices setup after the type of operation are defined. After specifying the parameters, click the “Run” button. The operation will be performed if the no error is detected. Otherwise, the system will prompt a warning message.



Choose different variable which are saved in the variable pool

The result could be saved into variable pool for further computing

Figure 6. 6

Menu bar

The Menu bar provides access to save/load and help functions. User could also exit the system from here.

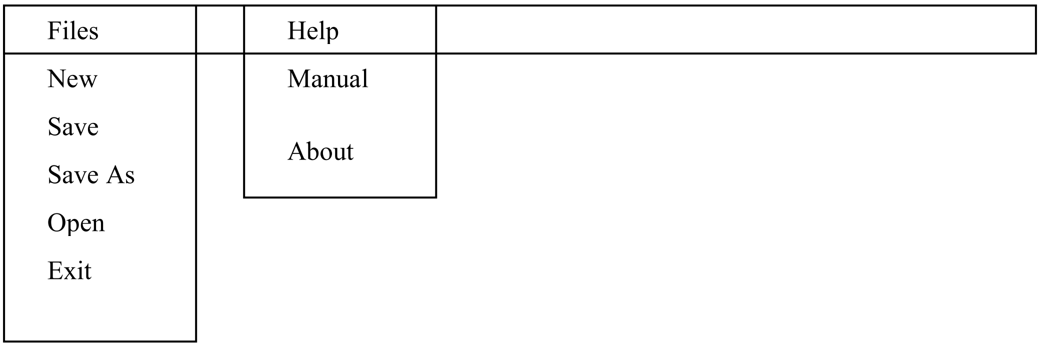


Figure 6. 7

6.4.4 The relation between components

(See Figure 6.8) The diagram shown below is an overview of the system, the main interface contains five main components, which are variable pool, variable property, operation, history command and menu bar.

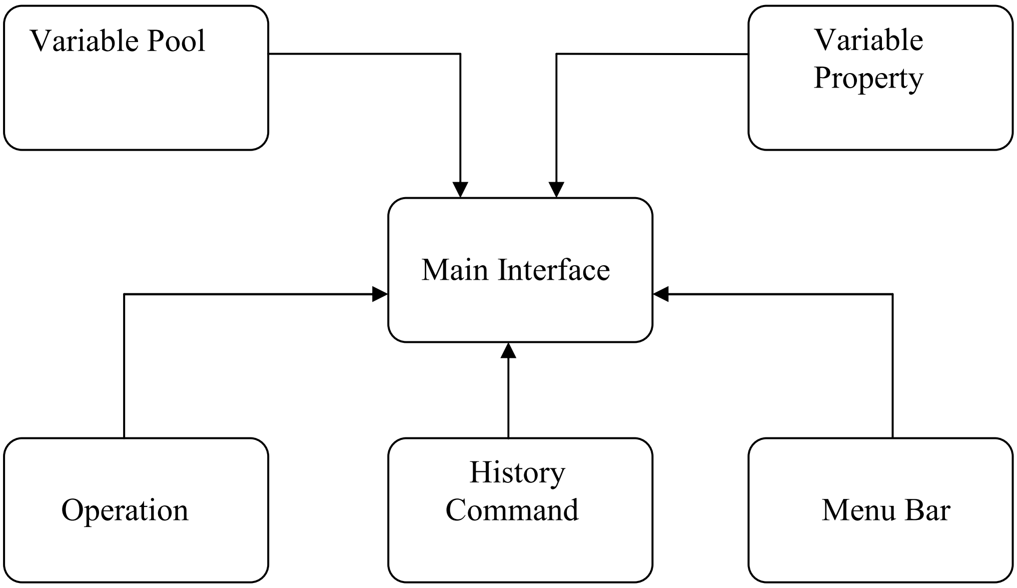


Figure 6. 8

(See Figure 6.9) The diagram below shows functions within the variable pool interact with each other. It has included six functionalities. They are “add a variable”, “delete selected”, “delete all variable”, “variable name”, “dimension row*column” and “square matrix”. The Variable Pool is the very first step for users to interact with the system. Name, Dimension (Row * Column) and Square matrix show the details of input matrix. For other three components add a variable, delete selected variable and delete all variable are functions on matrix. Matrix could be added and deleted.

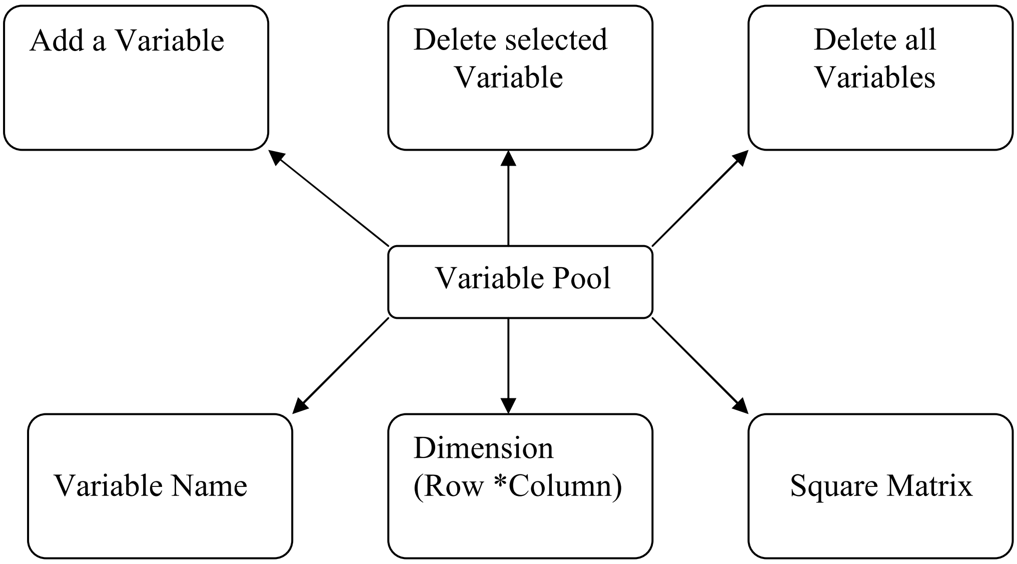


Figure 6. 9

(See Figure 6.10) The diagram below shows variable property, which could display all the information of the input matrix. It has interactive with variable pool and history command. The details that contain Variable name defined by users, square matrix or not, determinant, trace of matrix and the specific matrix of a new input matrix could be shown in variable property. If the result matrix is saved in variable pool for further use, the variable property will display the matrix result from history command.

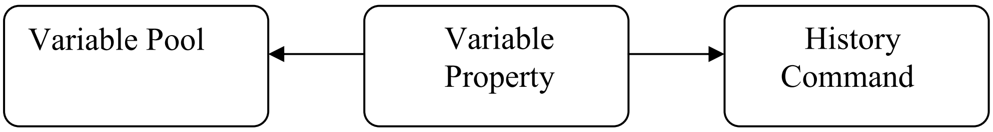


Figure 6. 10

(See Figure 6.11) Operation contains all the matrix built-in operations. The operation function could interact with variable pool, variable property, and parameter set-up and history command. Users are able to set matrices to be computed in Parameter set-up. They also could save the result matrix for further using, and then matrix result will display in variable pool, variable property and history command.

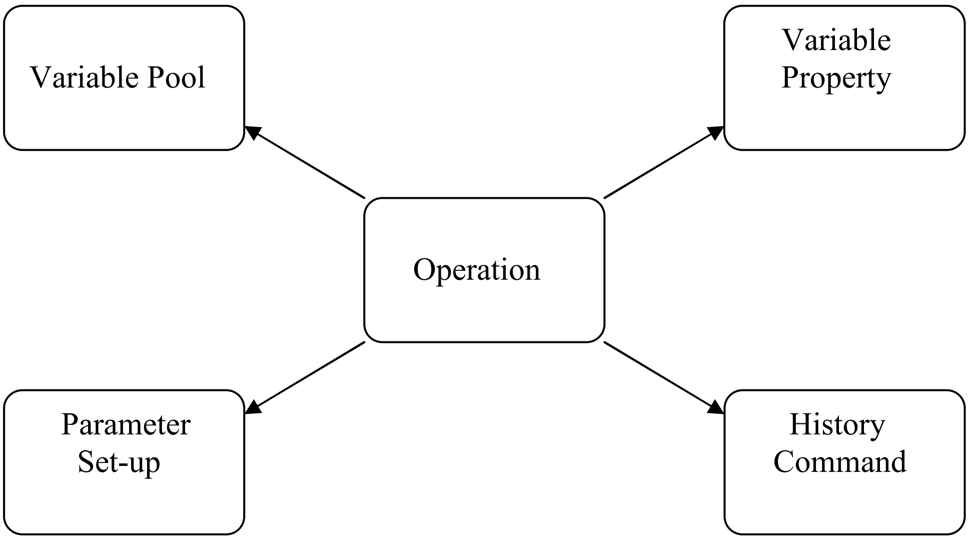


Figure 6. 11

(See Figure 6.12) The history command is linked with other five functions, which are "add a new line", "operation", "variable property", "clear all history commands" and "variable pool". It is able to clear all history commands in the history command, and add a new line in the history commands. It has been aforementioned, the result matrix that has been saved could display simultaneous in Variable Property, Variable Pool and History command.

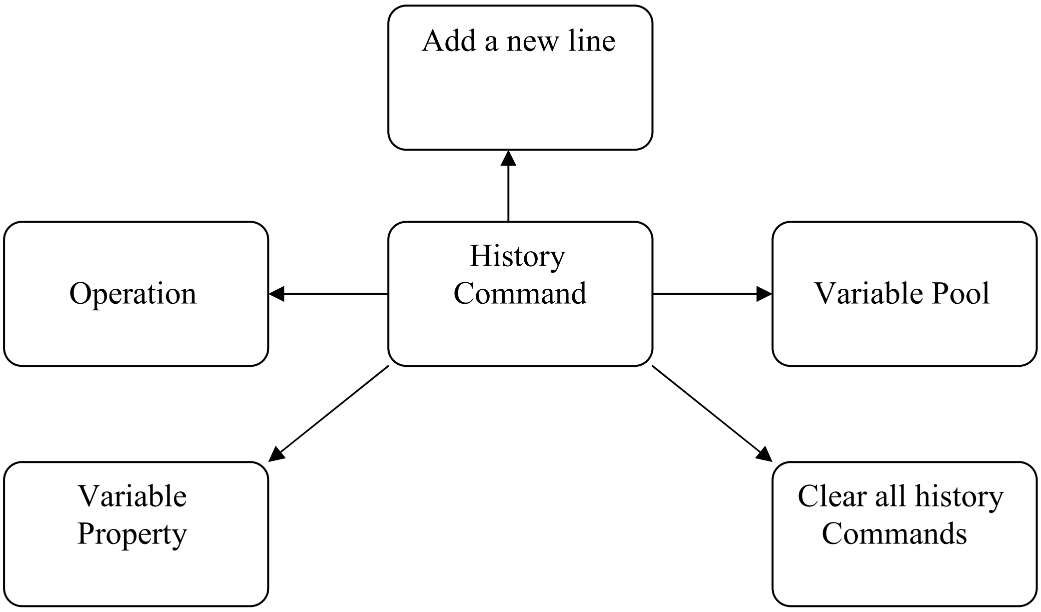


Figure 6. 12

(See Figure 6.13) The menu bar has included four activities, which are open files; help, save files, new worksheet perspective. Users could open the files to retrieve the matrix from the file, and save the matrix into the file for further use. Help function could supply for sufficient information for users using the system. New worksheet is to create a new worksheet; it will replace current unsaved data.

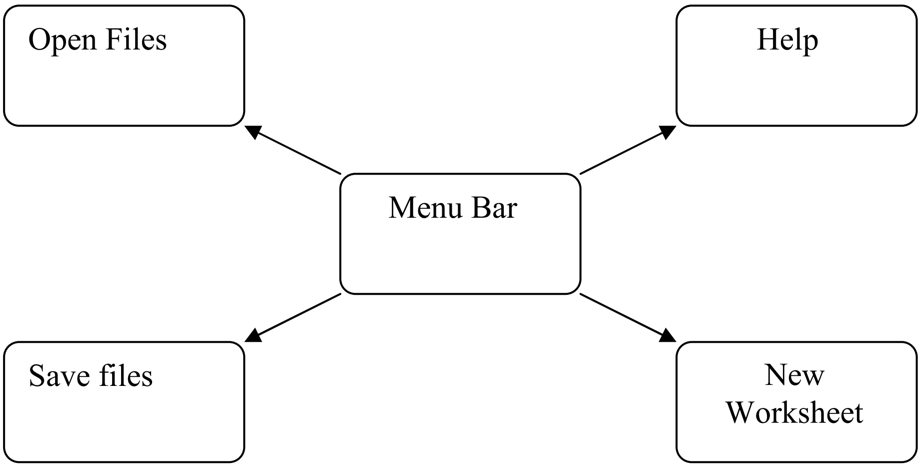


Figure 6. 13

6.4.3 The MVC

The concept of MVC (Model View Control) is important to the implementation of interface. Java has built in MVC support.

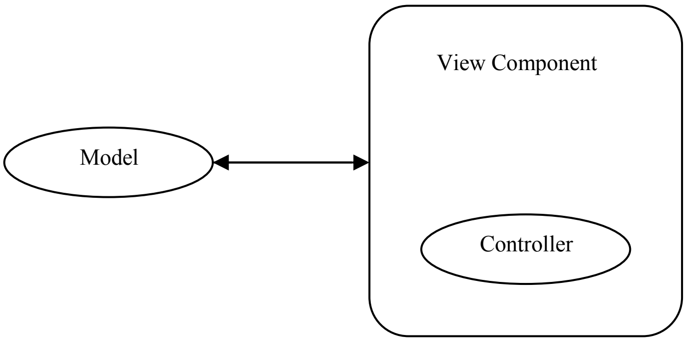


Figure 6. 14

To develop the user interface of this system with the MVC paradigm, it necessary to define the three constituents: Model, Controller and View.

- View comprises the visible GUI that the user sees.
- Model is the abstraction used in the application logic to represent the nature and state of the visual objects
- Controller enables the model and view components to communicate. Therefore, if a Controller updates the value holds in the Model, the view of the system will be reflected automatically.

The Model in the system should be implemented in the variableTable class. A Model called variableTableModel will then be created. The variableTableModel has the access to the entire variables in the variable pool. In order to get MVC work, all requests to update values in variable pool should through variableTableModel.

The Controller of the system should be implemented in the “Add variable”, “Del variable”, “Load”, and “Operation” functions. While these three functions need to perform changes in the variable pool, they should pass their requests the variableTableMode.

The only View component of the system should be varTable which is a JTable that displays contents of variable pool to the users. It will updated automatically once the Model is changes.

6.5 The Decimal Representation and the Rounding Problem

6.5.1The Decimal Representation

Because of the natural of computer architecture, some decimal numbers are not accurately represented in Java. For example:

```
double a1 = 1.1d;
double a2 = 3.0d;
System.out.println(a1*a2);
```

The expected result from the above Java code should be 3.3. But the actual result is 3.3000000000000003.

For general application, it may not be a problem as not everyone need vary accurate representations for decimal. In most situations, do a simple rounding is sufficient. However, in application such like linear algebra system, rounding is not a good option to choose.

Luckily, Java has a built-in solution to this problem. By using a package called java.math.BigDecimal, the above code can be converted to:

```
double a1 = 1.1d;
double a2 = 3.0d;
System.out.println(new
     BigDecimal(Double.toString(a1)).multiply(new
     BigDecimal(Double.toString(a2))));
```

The result will be 3.30, which is correct.

6.5.2 The Rounding Problem

The BigDecimal is a way to represent floating point. However, while performing the divide operations, a rounding method must be specified in case not dividable. This will sometimes cause unexpected result.

Consider the following example:

```
double a1 = 5.0d;
double a2 = 3.0d;

BigDecimal d1 = new BigDecimal(Double.toString(a1));
BigDecimal d2 = new BigDecimal(Double.toString(a2));

BigDecimal d3 =
d1.divide(d2,BigDecimal.ROUND_HALF_UP);
System.out.println(d3.multiply(d2));
```

The expected value is 5. The value printed out is 5.1 which is not expected. This sometimes causes serious trouble.

Consider the situation that while doing the row operations; an inaccurate matrix is obtained because of the rounding problem:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0.000000001 & 2 & 3 \\ 3 & 1 & 2 \end{bmatrix}$$

A[2][1] is expected to 0, but the actual value is 0.000000001. After one step of row operation, the A becomes:

$$A' = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 2/0.000000001 & 3/0.000000001 \\ 1 & 1/3 & 2/3 \end{bmatrix}$$

While the correct result should be:

$$A'' = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 3/2 \\ 1 & 1/3 & 2/3 \end{bmatrix}$$

A' and A'' is totally different matrix now.

The method described below could handle with this problem.

1. Suppose the user's input numbers are round with 8 decimal digits
2. After each operation, round the result with 16 decimal digits

The mechanism could be explained in the following example.
Suppose:

```
x  = 0.00000001;
y  = 0.00000001;
xy = 0.0000000000000001
```

x,y are the smallest positive number can be obtained.
xy is the smallest positive number can be obtained from multiplication
Thus if the any number with more 16 decimal digits appear, it's same to round it.

Chapter 7

System Testing

7.1 The techniques will be used in System testing

This section is going to test all the aspects of the system by using different testing techniques, which are unit testing, integration testing and system testing. The goal of this part is to meet all the functional requirements.

7.1.1 Unit testing

Unit testing is to test each component within the system. Test the individual component to ensure that they are working correctly for each of them before integrating them together.

Black box testing: the implementation of code is not available at all, it works as input something into the system, see what the system returns.

7.1.2 Integration Testing

“Integration testing can proceed in a number of different ways, which can be broadly characterised as top down or bottom up.” In top down integrations testing the high level control routines are tested first.

7.1.3 System Testing

The system testing will considerer the system as a whole, all the functional requirement should be tested to meet the original requirements.
The system testing is going to do the testing against the requirement specification.

7.2 Black box testing

This part is the black box testing for the interface, it should check that when a user requests a certain operation via the interface that their request is fully achieved. The testing result will represent testing is successful and testing fail.

Input	Expected Response	Actual Response	Testing Result
Add a valid matrix and specify a variable name that is not already in the variable pool	It is could be saved into variable pool, and the details of the variable will display in the Variable property panel	The result has been saved in the variable pool, the details of the variable display in the variable property.	Testing is successful.
Add a invalid matrix and specify a variable name that is not already in the variable pool	Invalid matrix input, it could not be saved into variable pool.	Please input a valid matrix, separated with ";".	Testing is successful.
Add a specified variable name	Add the matrix	Invalid matrix input, it could not be saved into variable pool.	Testing is ok.
Choose the function for the matrix.	The matrix could be computed under the specific operations.	The matrix could be computed under the specific operations.	Testing is successful.
Delete the selected variable	The variable in the variable pool could be removed	The variable is removed.	Testing is successful.
Delete all the variable	All the variables in the variable pool are removed.	The variables are removed.	Testing is successful
Clear the history command	All the information contained in the history command is removed.	Information are moved	Testing is successful
Calculate the inverse of a matrix as an example	The system should return the result without errors.	Errors free	Testing is successful.

7.3 System testing

This part is going to test all the aspects of the system to gain bug free software.

7.3.1 Testing for Variable Pool

Input	Expected Response	Actual Response	Testing Result
32.0 43.0 32.0 12.0 3.0 45.0 32.0 43.0 54.0	It can be saved in the variable pool, and the detail of this matrix will be shown in variable property as well	It is saved in the variable pool, and showed in variable property.	Testing is successful.

32 43 32;
43 32;
	Invalid matrix.	Please input a valid matrix, and separated by “;”.	Testing is successful.

21 32 43;
43abc;
	Invalid matrix.	Please input a valid matrix, and separated by “;”.	Testing is successful.

“Random”	Invalid matrix.	Please input a valid matrix, and separated by “;”.	Testing is successful.
None	Empty	Please input a valid matrix, and separated by “;”.	Testing is successful

Delete variable	Expected Response	Actual Response	Testing Result
Choose the variable to be deleted	The variable has chosen could be removed from variable pool.	The variable has been removed from variable pool.	Testing is successful.

Delete all variable	Expected Response	Actual Response	Testing Result
This function provided could delete all the variables from variable pool	The variables from the variable pool will be definitely removed	All the variables have been removed	Testing is successful

7.3.2 Variable property

Display	Expected Response	Actual Response	Testing Result
Detail of each variable	The variable added into the variable pool could be displayed in variable property	Variable property displays the details of input variable	Testing is successful

7.3.3 Operation

7.3.3.1 Addition

Pre-defined matrices used for testing:

Matrix A, B, C:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad B = \begin{bmatrix} 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \end{bmatrix} \quad C = \begin{bmatrix} 32 & 54 & 32 \\ 43 & 54 & 43 \end{bmatrix}$$

Matrix One	Matrix Two	Save as	Expected Result in Variable Pool	Actual Result in Variable Pool	Expected Result in history command	Actual Result in Variable Pool	Testing Result
A	B	D	D will be added into the Variable Pool	D is added into the Variable Pool	Return the result of addition of A and B	Return the addition result of A and B	Testing is successful
A	B	C	The result cannot be saved into the Variable Pool.	The variable name has already been used, please choose another name	The result cannot return.	The result doesn't return	Testing is successful
A	C	E	E will not be added in the Variable Pool.	E is not added in the Variable Pool.	It will return error message instead of result	Error, the dimensions of 'a' and 'c' are incompatible for addition	Testing is successful

7.3.3.2 Subtraction

Pre-defined matrices used for testing:

Matrix A, B, C:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad B = \begin{bmatrix} 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \end{bmatrix} \quad C = \begin{bmatrix} 32 & 54 & 32 \\ 43 & 54 & 43 \end{bmatrix}$$

Matrix One	Matrix Two	Save as	Expected Result in Variable Pool	Actual Result in Variable Pool	Expected Result in history command	Actual Result in Variable Pool	Testing Result
A	B	D	D will be added into the Variable Pool	D is added into the Variable Pool	Return the result of subtraction of A and B	Return the subtraction result of A and B	Testing is successful
A	B	C	The result cannot be saved into the Variable Pool.	The variable name has already been used, please choose another name	The result cannot return.	The result doesn't return	Testing is successful
A	C	E	E will not be added in the Variable Pool.	E is not added in the Variable Pool	It will return error message instead of result	Error, the dimensions of 'a' and 'c' are incompatible for addition	Testing is successful

7.3.3.3 Scalar

Pre-defined matrices used for testing:

Matrix A, B, C:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Matrix One	Save as	Scalar	Expected result in Variable Pool	Actual result in Variable Pool	Expected result in History Command	Actual result in History Command	Testing Result
A	N/A	N/A	No Change	No Change	No Change	No Change	Testing Successful
A	B	2	B will be added in Variable Pool	B is added in Variable Pool	The result of scalar will return	The result returns	Testing Successful
A	N/A	2	No change	No change	The result of scalar will return	The result returns	Testing Successful
A	C	N/A	No change	No change	The result of scalar will not return	The result does not return	Testing Successful

7.3.3.4 Multiplication

Pre-defined matrices used for testing:

Matrix A, B, C:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad B = \begin{bmatrix} 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \end{bmatrix} \quad C = \begin{bmatrix} 32 & 54 & 32 \\ 43 & 54 & 43 \end{bmatrix}$$

Matrix One	Matrix Two	Save as	Expected Result in Variable Pool	Actual Result in Variable Pool	Expected Result in history command	Actual result in History Command	Testing Result
A	B	D	D will be added in Variable Pool	D is added in Variable Pool	Result will return	Result returns	Testing Successful
A	C	D	D will not be added in Variable Pool	D is not added in Variable Pool	Result will not return	Result does not return	Testing Successful
A	C	N/A	None	None	Result will not return	Error, the dimensions of ‘a’ and ‘c’ are incompatible for multiplication	Testing Successful

7.3.3.5 Transpose

Pre-defined matrices used for testing:

Matrix A:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Matrix One	Save as	Expected result in Variable Pool	Actual result in Variable Pool	Expected result in History Command	Actual result in History Command	Testing Result
A	C	C will be added in the Variable Pool	C is added in the Variable Pool	Result will return	Result returns	Testing Successful
A	N/A	None	None	Result will return	Result returns	Testing Successful
A	A	It will not be added in the Variable Pool	It is not added in the Variable Pool	Result will not return	Result does not return	Testing Successful

7.3.3.6 Inverse

Pre-defined matrices used for testing:

Matrix A:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad B = \begin{bmatrix} 43 & 654 & 43 \\ 643 & 4 & 54 \\ 12 & 98 & 76 \end{bmatrix} C = \begin{bmatrix} 32 & 54 & 32 \\ 43 & 54 & 43 \end{bmatrix}$$

Matrix One	Save as	Expected result in Variable Pool	Actual result in Variable Pool	Expected result in History Command	Actual result in Variable Pool	Testing Result
A	A	No change	No change	No change	No change	Testing Successful
A	B	B will be added in Variable Pool	B is added in Variable Pool	Determinant is zero, it will return no inverse	Error, inverse requires a non-singular matrix	Testing Successful
A	N/A	No change	No change	Determinant is zero, it will return no inverse	Error, inverse requires a non-singular matrix	Testing Successful
B	N/A	No change	No change	Inverse of B will return	Inverse of B returns	Testing Successful
C	N/A	No change	No change	Inverse of B will not return	Error: inverse requires a non-singular matrix	Testing Successful

7.3.3.7Ad-joint

Pre-defined matrices used for testing:

Matrix A, B:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad B = \begin{bmatrix} 32 & 54 & 32 \\ 43 & 54 & 43 \end{bmatrix}$$

Matrix One	Save as	Expected result in Variable Pool	Actual result in Variable Pool	Expected result in History Command	Actual result in History Command	Testing Result
A	A	No change	No change	No change	No change	Testing Successful
A	C	C will be added in Variable Pool	C is added in Variable Pool	Result will return	Result returns	Testing Successful
A	N/A	No change	No change	Result will return	Result returns	Testing Successful
B	N/A	No change	No change	Result will not return	Error, inverse requires a square matrix	Testing Successful

7.3.3.8 Cramer's Rule

Pre-defined matrices used for testing:
Matrix A, B:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 3 & 1 & -2 \\ 2 & 5 & 1 & -5 \\ 1 & 2 & 3 & 6 \end{bmatrix}$$

Matrix One	Save as	Expected result in Variable Pool	Actual result in Variable Pool	Expected result in History Command	Actual result in History Command	Testing Result
A	B	No change	No change	Determinant of the coefficient is equal to zero, there is no unique solution	No unique solution	Testing Successful
A	C	C will not be added in Variable Pool	C is not added in Variable Pool	Determinant of the coefficients is equal to zero, there is no unique solution	No unique solution	Testing Successful
B	C	C will be added in Variable Pool	C is added in Variable Pool	Result will return	Result returns	Testing Successful
B	N/A	None	None	Result will return	Result returns	Testing Successful
A	N/A	None	None	Determinant of the coefficients is equal to zero, there is no unique solution.	No unique solution	Testing Successful

7.3.3.9 Gauss-Jordan Elimination

Pre-defined matrices used for testing:
Matrix A:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Matrix One	Save as	Expected result in Variable Pool	Actual result in Variable Pool	Expected result in History Command	Actual result in History Command	Testing Result
A	B	No change	No change	No change	No change	Testing Successful
A	C	C will be added in Variable Pool	C is added in Variable Pool	Result will return	Result returns	Testing Successful
A	N/A	None	None	Result will return	Result returns	Testing Successful

8. Evaluation

8.1 The Final System

This dissertation project has presented the design of a computer linear algebra system. The project is coded in Java. The final system mainly consists two parts: the algorithm and the interface. The system has been packed into an executable JAR file.

8.2 Evaluation against the requirements

From the literature review and the requirements analysis, a set of requirements have been obtained. Now I am going to compare the final system with these initial requirements.

The system should be able to perform all the basic linear algebra operations.

The implementation of the final system does provide all the basic linear algebra operation including matrix addition and subtraction, matrix multiplication, matrix scalar, matrix determinant, matrix inverse, matrix adjoint, Cramer's rule and Gauss-Jordan Elimination.

However only the algorithms provide are all basic algorithms. Other algorithms like fast matrix multiplication are not implemented.

The system should be able to store variables

A fully working variable pool is indeed implemented. The variable stored in the system can be used for matrix operations.

The system should be able to delete variables

The variables in the variable pool can be deleted from the system. Once deleted, the variable will no longer available for matrix operation.

The system should be able to display matrix properties

Variable properties can be displayed in correctly.

The system should be able to create, save and load worksheet

The concept of worksheet is fully implemented. Once a worksheet is created, the variable pool and history command can be initialised successfully.
After loading a worksheet, the variable pool and history command could be loaded correctly. Saving the worksheet will create a file on the storage disk and can be loaded successfully.

The system should provide facilities that allow user input command

A drop-down list is provided for matrix operations choose. Once an operation is selected, by providing suitable parameters, the system will perform the operation.

A history command should be provided

The previous commands inputted into the system will be displayed.

The system should have a good error handing mechanism

From the testing, all error could be caught by the system. Thus the mechanism for error is perfect.

A user manual should come with the system

The user manual is indeed provided with the system.

Chapter 9

Conclusion

9.1 The Future Work

During the implementation, although most objectives of the system has been met. Quite a few things will be improved at the future release.

- The Matrix class uses a double to store the value of matrix. This should be sufficient for most situations. But it is not be accurate enough for scientific operation.
- It’s possible to build an API for linear algebra operation. Thus different programmers could use the different API for linear algebra system programming.
- The Gauss-Jordan elimination in this implementation only return the reduced echelon form of a matrix. No detailed solution has been given. In the future release, functions should be provided to analysis the reduced echelon form of matrix.
- A toolbar for quick command access could be provided in the future release.
- Short-cut keys should be implemented in the future release
- In this implementation, no functions for matrix copy and paste is provided. Should implement in the next release
- A function to print the worksheet will be useful
- More linear algebra functions should be provided in the future release.
- Facilities should be provided for large matrix storage.
- In this release, variable can’t be changed once created. Thus functions to edit a created variable could be useful.
- In this implementation, only one worksheet can be used at a time. Thus multiple worksheet function would be very useful.

9.2 Personal Remark

During developing the project, I have acquired a lot such as how to apply for different models for software design. For both coding experience and software experience have already been improved.

Also, in doing this project, I learned how to tackle the problems such as by using MVC (Model-View-Controller) to the design the system interface and also the problems in the algorithm part. However, I have not arranged a good time for the dissertation. Actually, I did a really good product. On the dissertation part I did not concentrate on it. I really know how to improve my project, I fully believe I am able to do a better one (include both software and documentation), if more time provided. However, I spent a lot of time on developing my program. I am capable to say my program is the best in this area, but with the lack of time, I could not spend more time to produce documentation. For future work on project, I will notice the time constraints to try the best to achieve the good result as I can.

Chapter 10

Bibliography

Char, B, W. (2003) *Maple 9: learning guide*. Maplesoft Press

Cohen, B. W. (2003) Computer algebra and symbolic computation: elementary algorithms. Peters Press

Chris Cannam. Announcement: XR Graphical User Interface for REDUCE. Available from: http://www.uni-koeln.de/REDUCE/reduce-forum/93/msg37.html (April 22, 2005)

Heck, A., 1996. *Introduction to Maple*. 2nd. Springer

MathSci. (2004) *Linear Algebra using Maple.* Available from: http://www.mathsci.appstate.edu/u/math/hph/maple/linalg.html (December 01.2004).

Matt Rebbeck. (1994) *A Linear Algebra package for REDUCE*. [WWW] http://www.reduce-algebra.com/docs/linalg.pdf (December 02, 2004)

Niemeyer, Patrick. 2002 *Learning Java.* O'Reilly

Pilbeam, B., 2002. *Computer linear algebra system*. Thesis (MSc). Bath University

Santamaria-Ortega, Laia(2002) *Computer linear algebra system*. University of Bath.

Sommerville, I., 2001. 6th ed. *Software engineering*. Addison Wesley

Steven H, Elaine M. *implementation of Strassen's Algorithm for Matrix Multiplication.* Available from: http://portal.acm.org/ft_gateway.cfm?id=369096&type=pdf&coll=GUIDE&dl=GUIDE&CFID=43966928&CFTOKEN=74409338(April 12, 2005)

The semantics of Java Object Copy. Available from: http://courses.dce.harvard.edu/~cscie160/Clone.htm (March 12, 2005)

Uni-koeln. (1995) *The REDUCE Computer Algebra System*. [WWW] http://www.uni-koeln.de/REDUCE/ (November 22, 2004)

V.Strassen. *Gaussian Elimination is not optimal* . Numer. Math., 13:354-356,1969.

Wikipedia. (2004*) Computer Algebra System*. [WWW] http://en.wikipedia.org/wiki/Linear_algebra (November 20, 2004)

Wikipedia. The Free Encyclopedia.2005. *Matrix (mathematics).* Available from: http://en.wikipedia.org/wiki/Matrix_%28mathematics%29 (April 30, 2005)

William, G., 2001.4th ed. *Linear algebra with application.* Jones and Bartlett.

Wikipedia. The Free Encyclopedia. 2005. *Computer algebra system*. Available from: http://en.wikipedia.org/wiki/Computer_algebra (April 10, 2005)

WOLFRAM Research. *Linear algebra*. Available from: http://mathworld.wolfram.com/LinearAlgebra.html(May 12, 2005)

wikipedia. The Free Encyclopedia.2005. *Determinant*. Available from: http://en.wikipedia.org/wiki/Determinant(April 11, 2005)

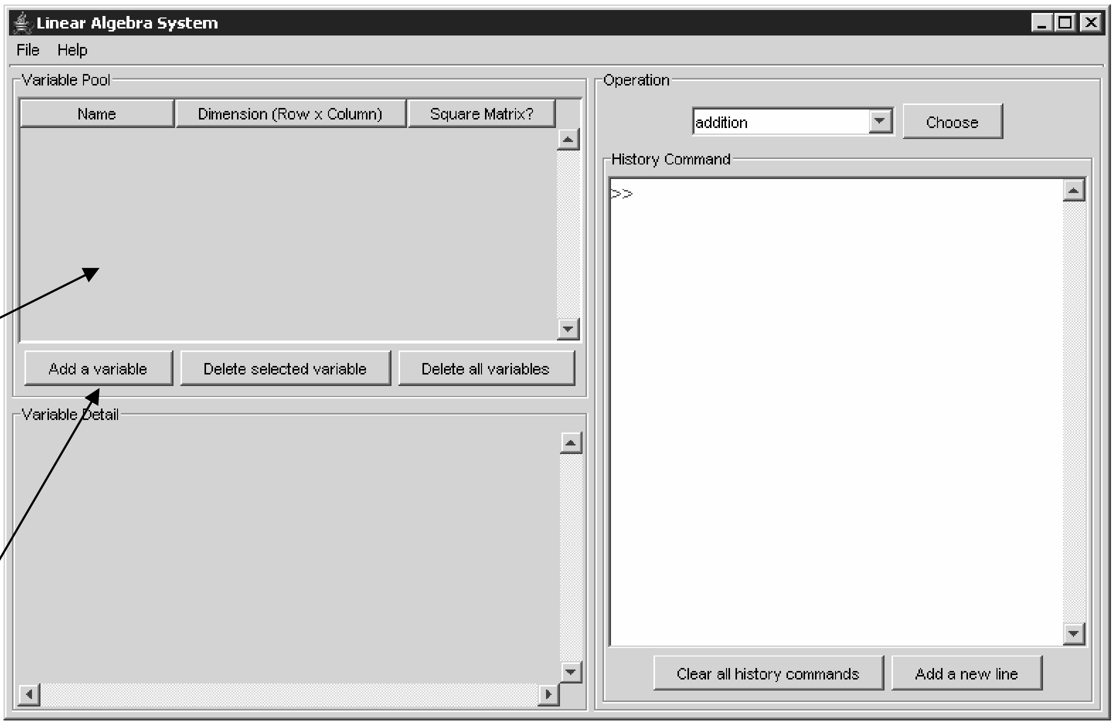
wikipedia. The Free Encyclopedia.2005. *System of linear equations*. Available from http://en.wikipedia.org/wiki/System_of_linear_equations (April 12, 2005)

11. Appendix

11.1 User Manual

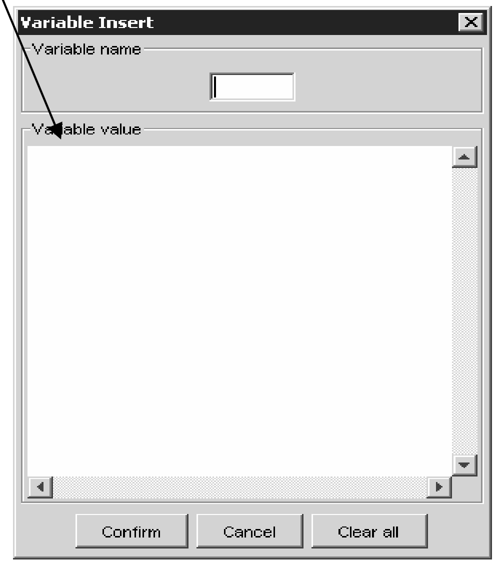
This part is a general instruction to the system. The main functionalities are going to be introduced

Below is the main screen of the system.

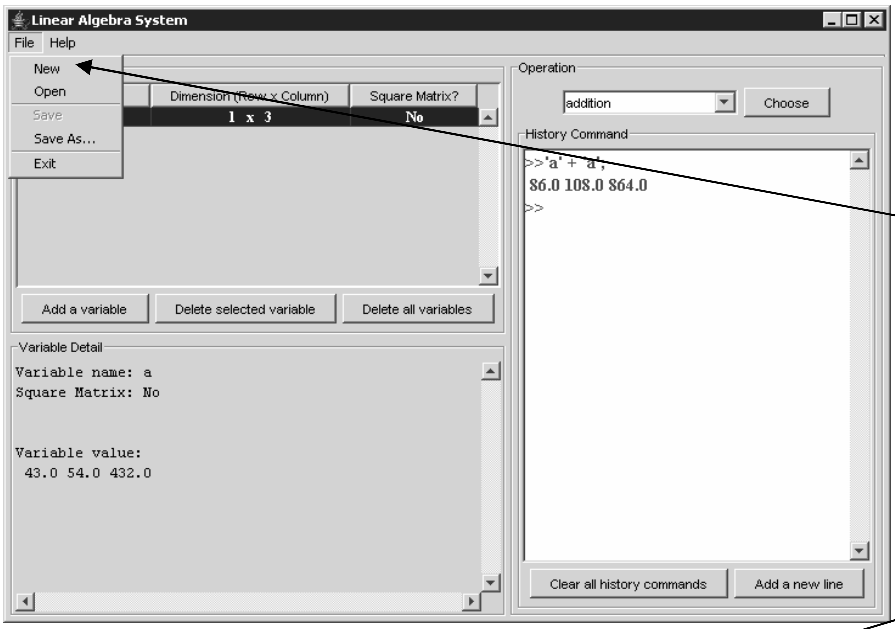


User inputs the variables into variable pool

Press Add a variable. Then the variable insert will display for users to input the matrix



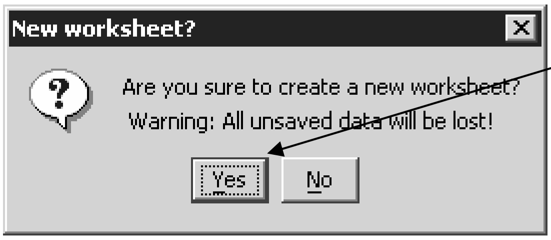
How to create a new worksheet



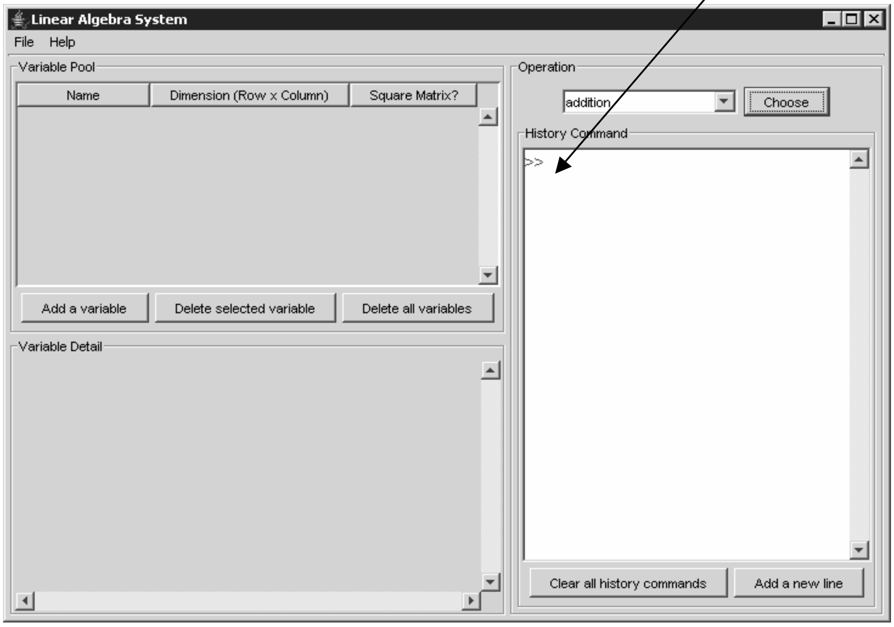
```
>>'a' + 'a';
 86.0 108.0 864.0
>>
```

```
Variable name: a
Square Matrix: No


Variable value:
 43.0 54.0 432.0
```

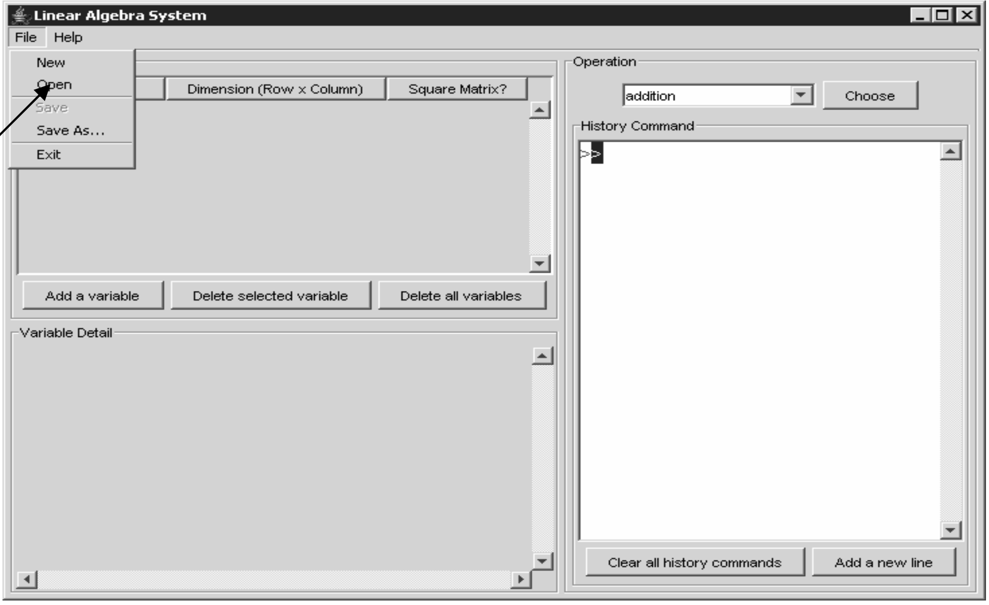
Click the file, and then click new. It will display confirmation screen. By pressing the yes button, the current worksheet will be created by a new one.



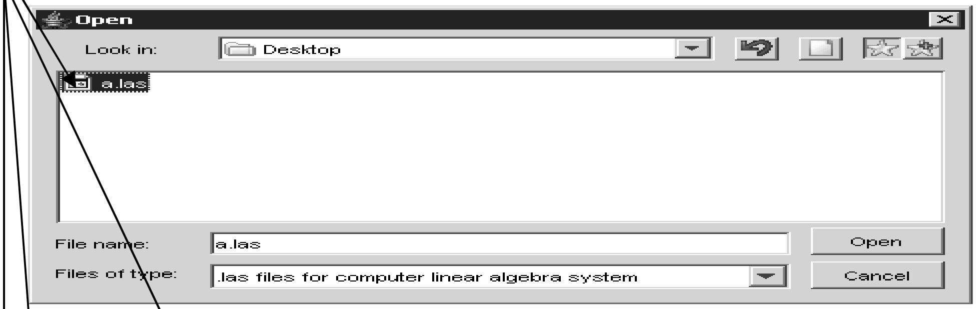
Are you sure to create a new worksheet?
Warning: All unsaved data will be lost!



How to open an existing worksheet

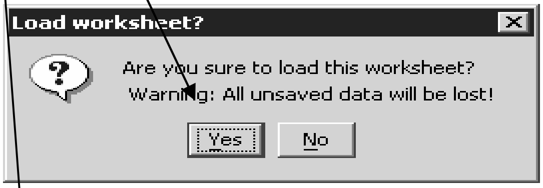


Open a worksheet by clicking open, and then the available file will display. It will send a warning dialog to ask users to save current worksheet. By clicking yes button, the file will be open

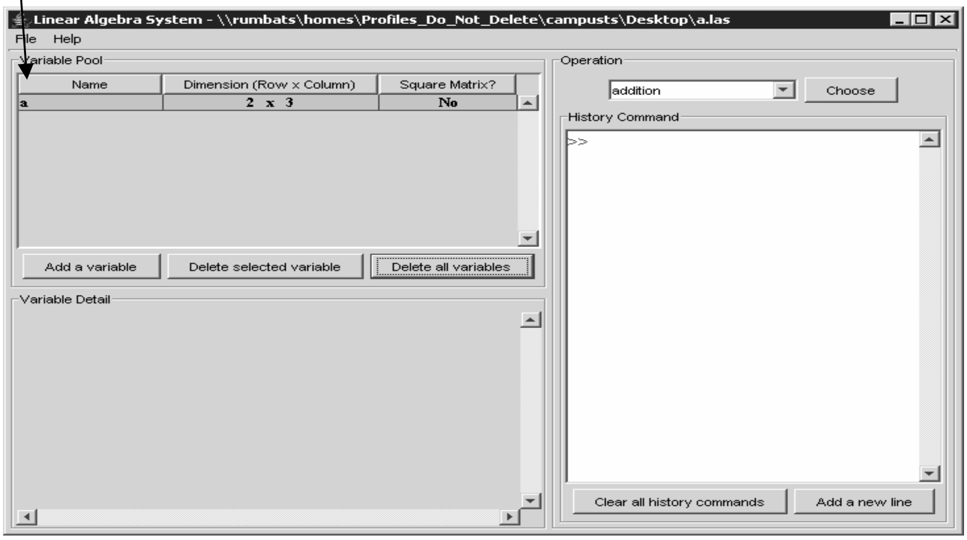


File name: a.las

Files of type: .las files for computer linear algebra system

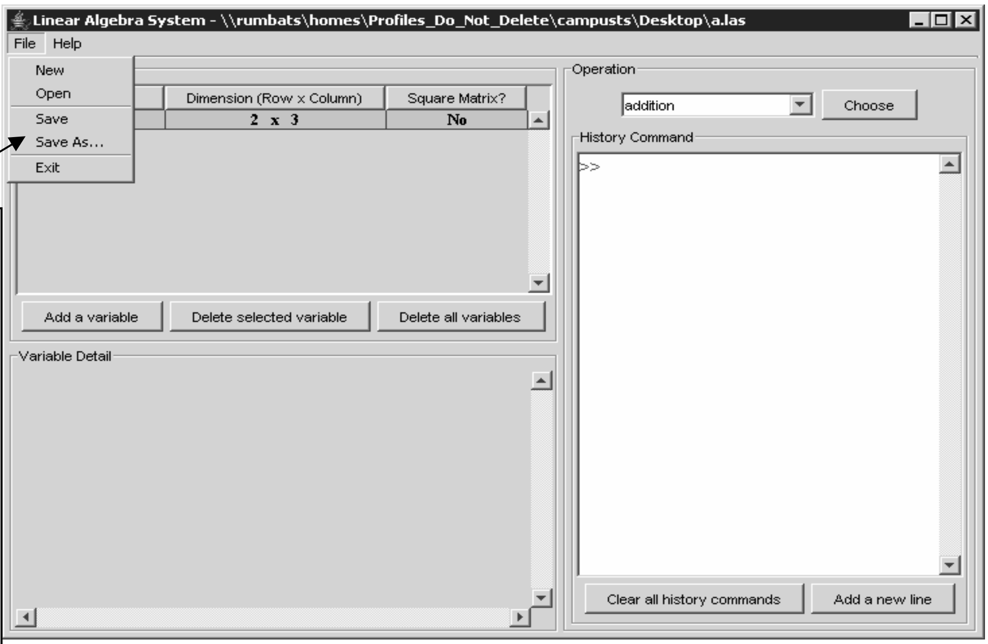


Are you sure to load this worksheet?
Warning: All unsaved data will be lost!

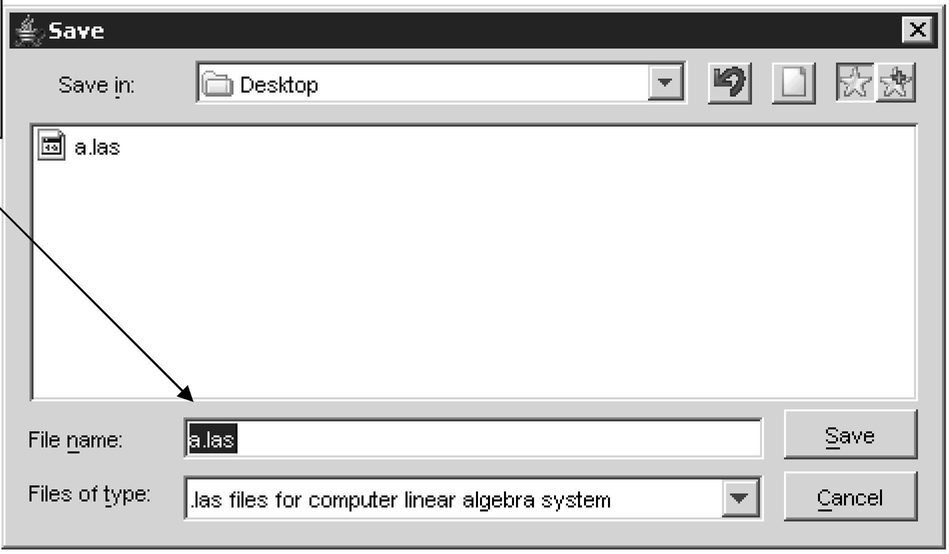


Name	Dimension (Row x Column)	Square Matrix?
a	2 x 3	No

How to save current worksheet



To save current worksheet by clicking save as from the file. Writing the name of the file you want to save.

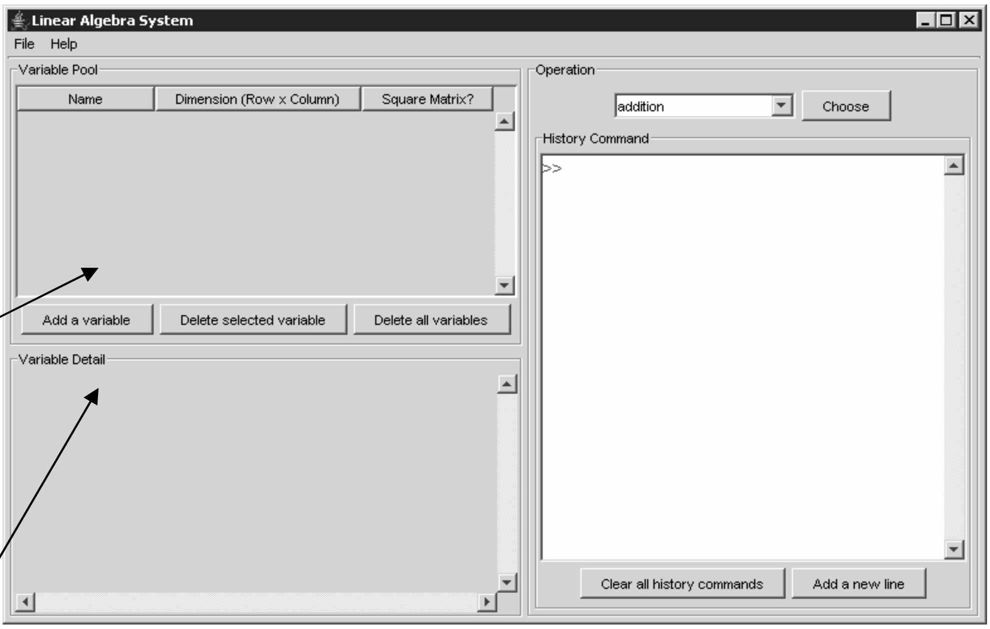


Save in: Desktop

File name: a.las

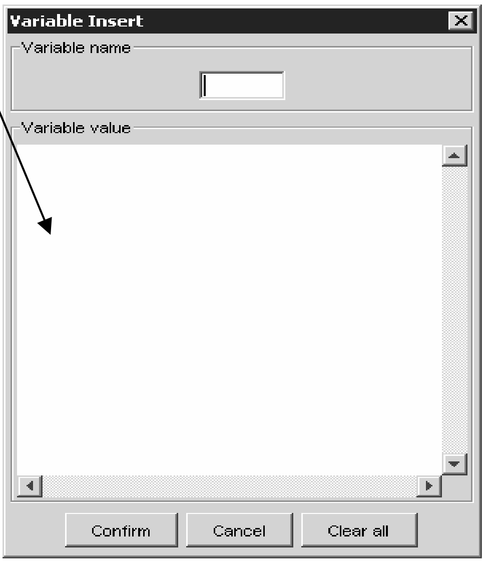
Files of type: .las files for computer linear algebra system

How to input a variable

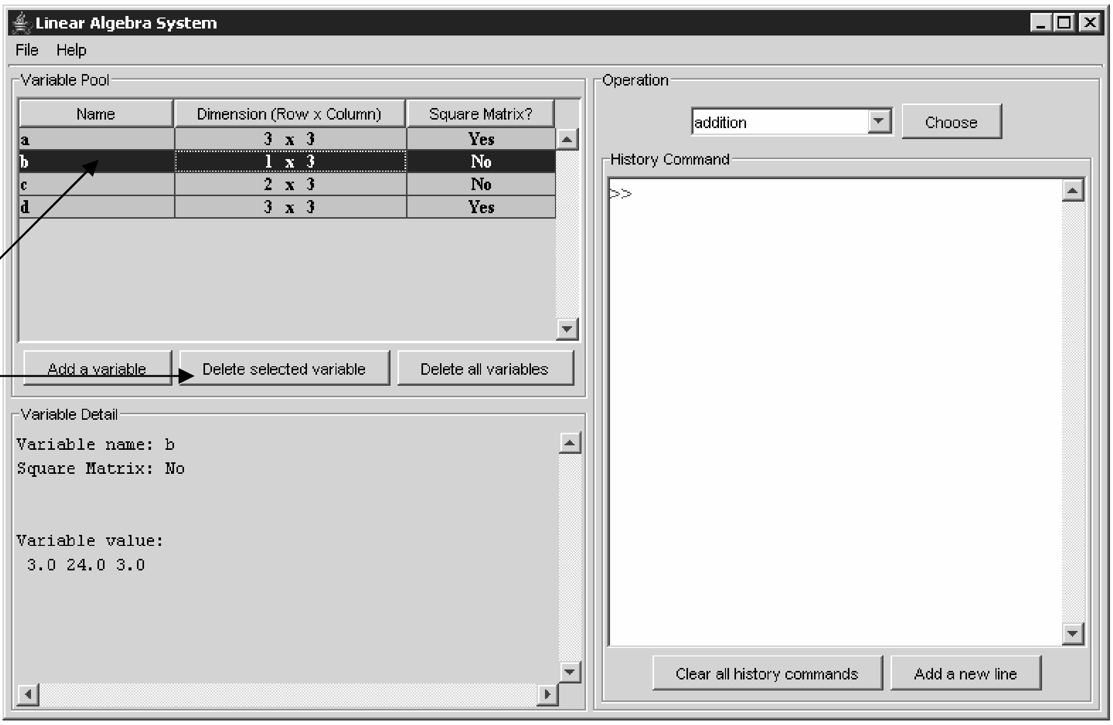


User inputs the variables into variable pool

Press Add a variable. Then the variable insert will display for users to input the matrix



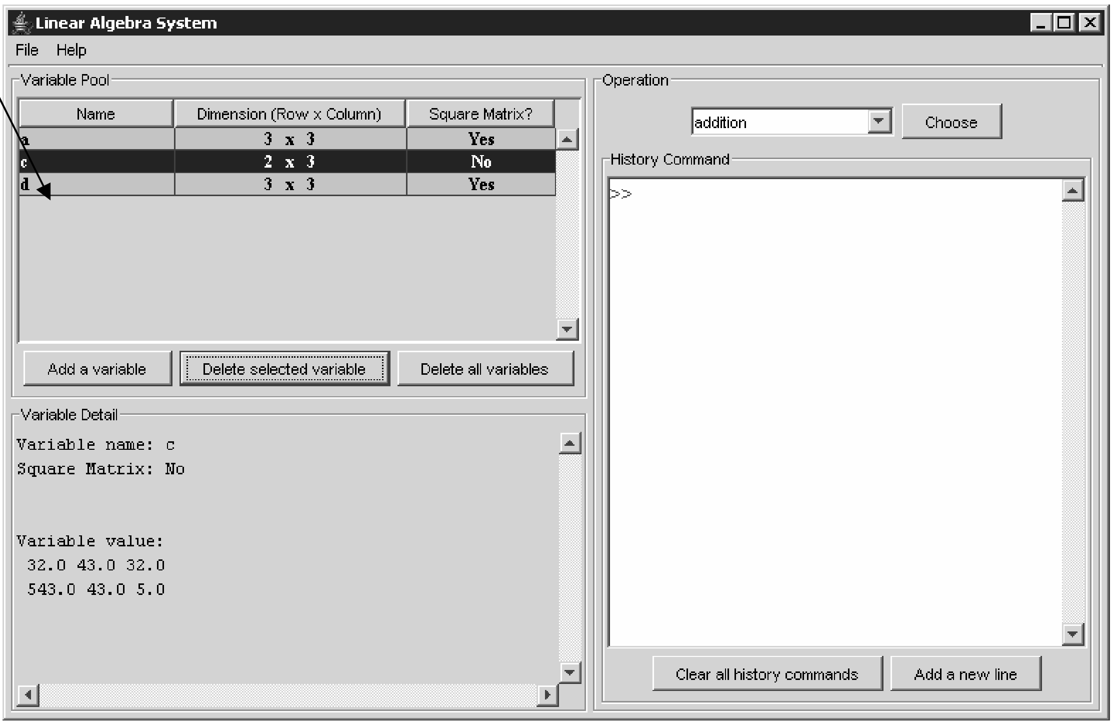
How to delete a variable



Name	Dimension (Row x Column)	Square Matrix?
a	3 x 3	Yes
b	1 x 3	No
c	2 x 3	No
d	3 x 3	Yes

```
Variable name: b
Square Matrix: No


Variable value:
 3.0 24.0 3.0
```

Delete selected variable and then press delete selected variable button.
b has been deleted

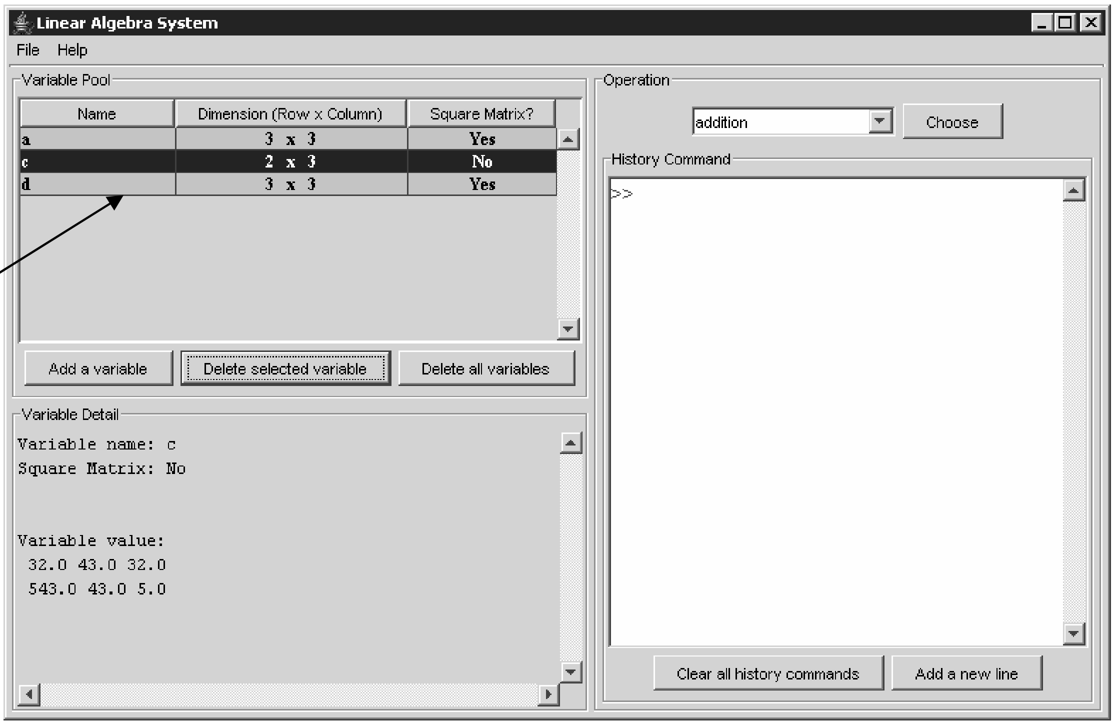


Name	Dimension (Row x Column)	Square Matrix?
a	3 x 3	Yes
c	2 x 3	No
d	3 x 3	Yes

```
Variable name: c
Square Matrix: No


Variable value:
 32.0 43.0 32.0
 543.0 43.0 5.0
```

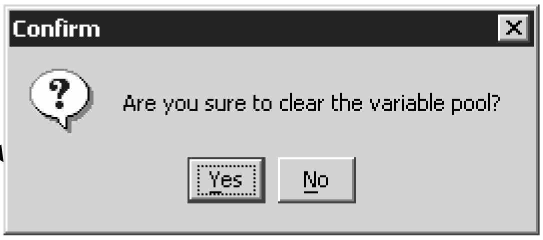
How to clear the variable pool

Clear the variable pool Press the delete all variables button, and then the dialog will display for confirm. After confirming, all the variables in the variable pool are deleted.

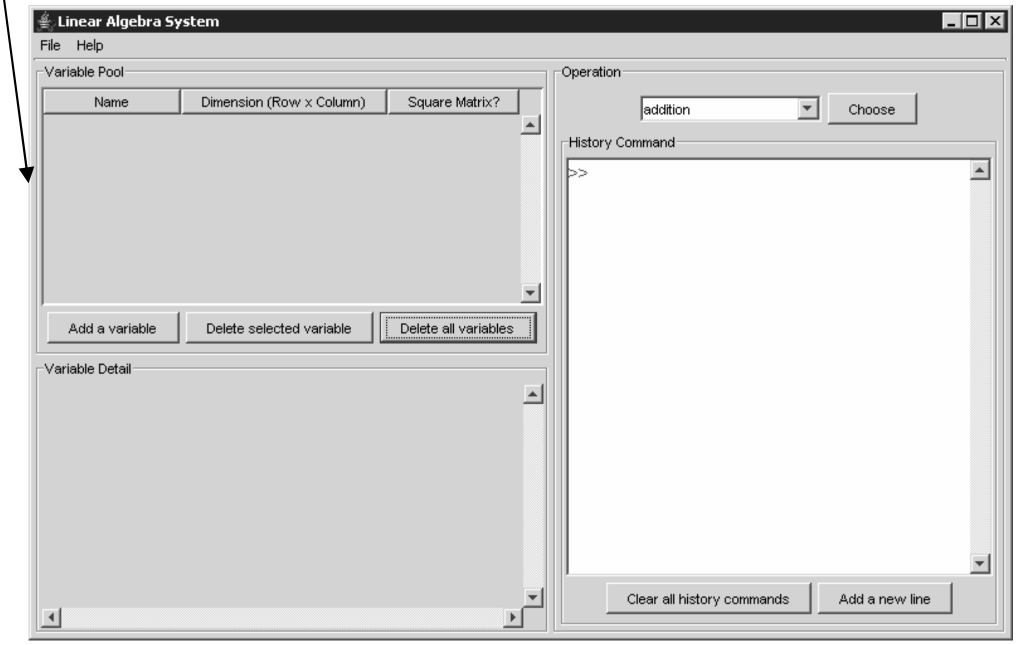


Name	Dimension (Row x Column)	Square Matrix?
a	3 x 3	Yes
c	2 x 3	No
d	3 x 3	Yes

```
Variable name: c
Square Matrix: No


Variable value:
 32.0 43.0 32.0
 543.0 43.0 5.0
```

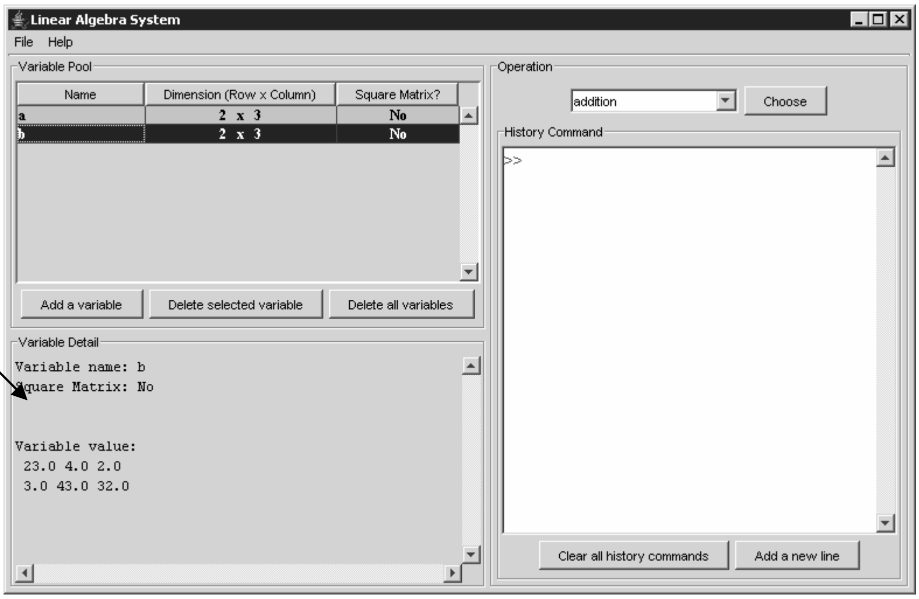


Are you sure to clear the variable pool?



How to check the detail of the variable property

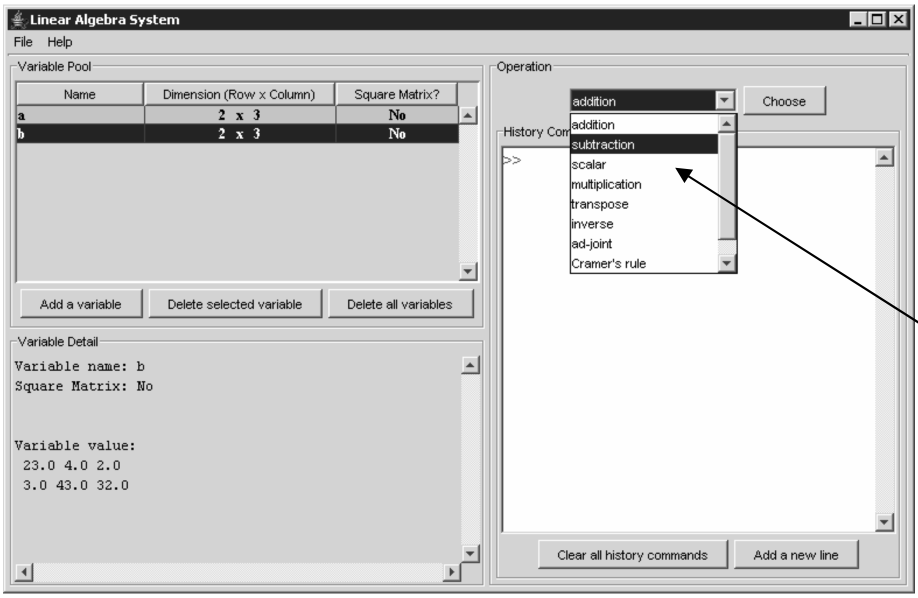
Details of the variable display in the variable property



Name	Dimension (Row x Column)	Square Matrix?
a	2 x 3	No
b	2 x 3	No

```
Variable name: b
Square Matrix: No


Variable value:
 23.0 4.0 2.0
 3.0 43.0 32.0
```

How to perform a matrix operation

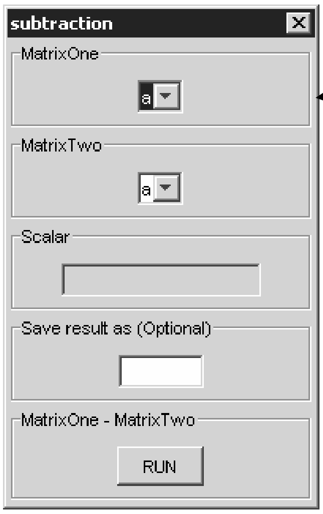


Name	Dimension (Row x Column)	Square Matrix?
a	2 x 3	No
b	2 x 3	No

```
Variable name: b
Square Matrix: No


Variable value:
 23.0 4.0 2.0
 3.0 43.0 32.0
```

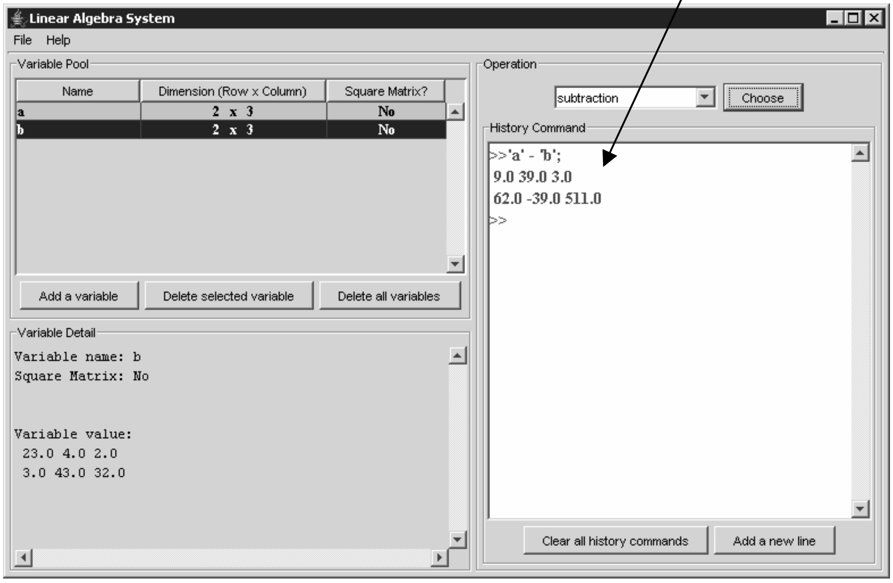
```
>>
```

Choosing the operation from the drop down menu, then click choose button. It will display parameter set-up. When the set-up is finished, click the RUN button to get the result.



MatrixOne: a

MatrixTwo: a



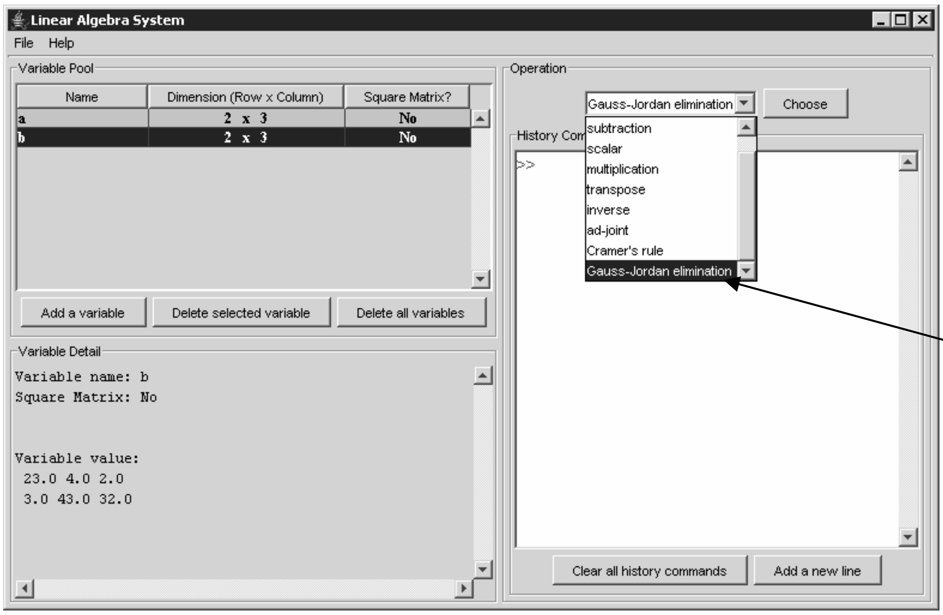
Operation: subtraction

Name	Dimension (Row x Column)	Square Matrix?
a	2 x 3	No
b	2 x 3	No

```
Variable name: b
Square Matrix: No


Variable value:
 23.0 4.0 2.0
 3.0 43.0 32.0
```

```
>>'a' - 'b';
 9.0 39.0 3.0
 62.0 -39.0 511.0
>>
```

How to solve the system of linear equation

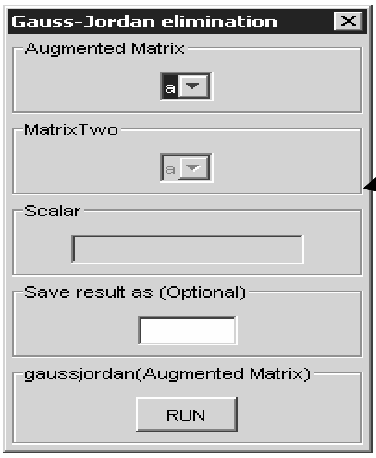


Name	Dimension (Row x Column)	Square Matrix?
a	2 x 3	No
b	2 x 3	No

```
Variable name: b
Square Matrix: No


Variable value:
 23.0 4.0 2.0
 3.0 43.0 32.0
```

```
>>
```

Choosing the Gauss-Jordan Elimination from the drop down menu, then click choose button. It will display parameter set-up. When the set-up is finished, click the RUN button to get the result.

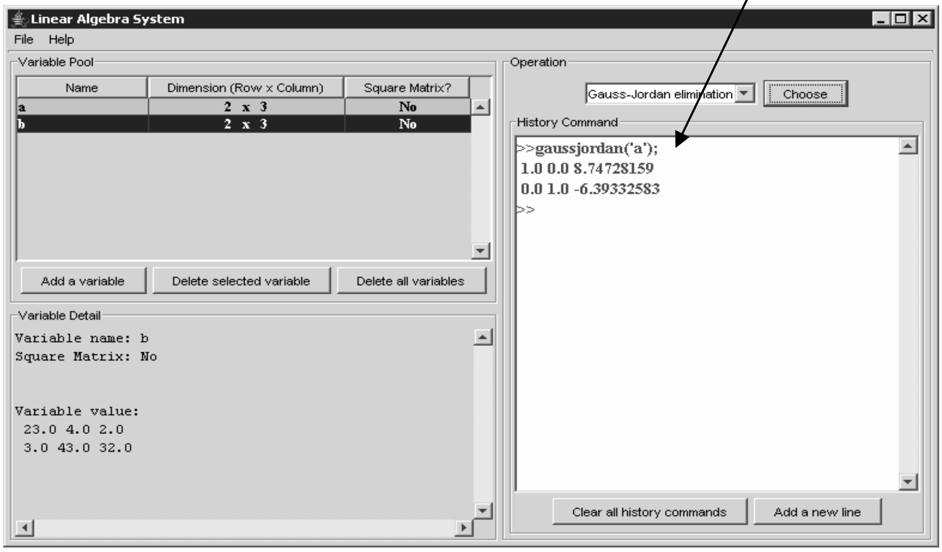


Augmented Matrix: a

MatrixTwo: a

Save result as (Optional)

gaussjordan(Augmented Matrix)

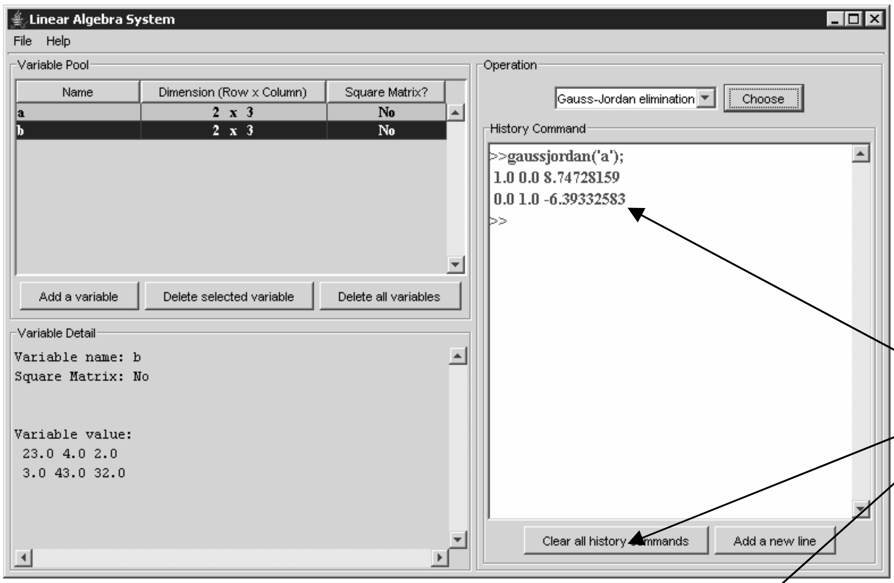


Name	Dimension (Row x Column)	Square Matrix?
a	2 x 3	No
b	2 x 3	No

```
Variable name: b
Square Matrix: No


Variable value:
 23.0 4.0 2.0
 3.0 43.0 32.0
```

```
>>gaussjordan('a');
 1.0 0.0 8.74728159
 0.0 1.0 -6.39332583
>>
```

How to operate the command history

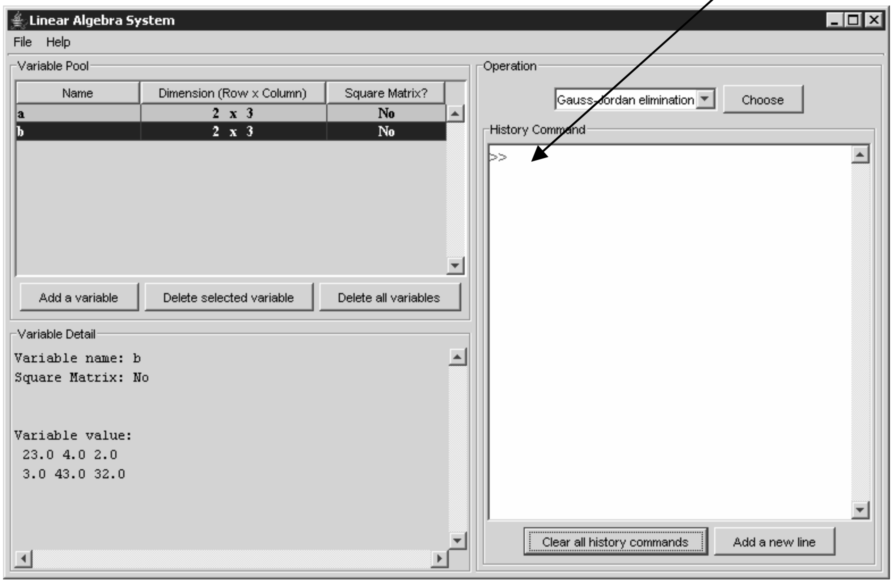


Name	Dimension (Row x Column)	Square Matrix?
a	2 x 3	No
b	2 x 3	No

```
Variable name: b
Square Matrix: No


Variable value:
 23.0 4.0 2.0
 3.0 43.0 32.0
```

```
>>gaussjordan('a');
1.0 0.0 8.74728159
0.0 1.0 -6.39332583
>>
```

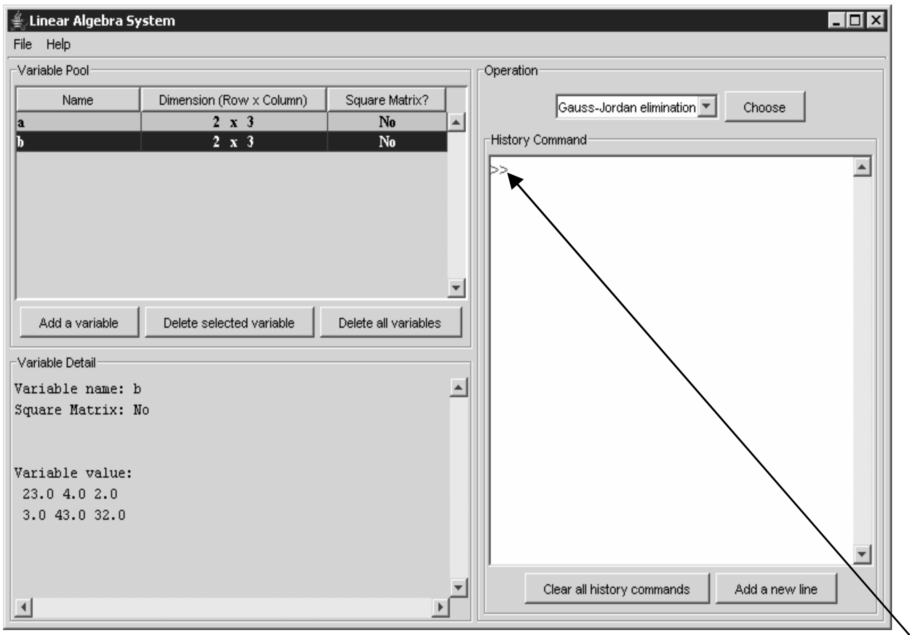
Clear the result in the history commands by pressing clear all history command button. Then all the information will be deleted.



Name	Dimension (Row x Column)	Square Matrix?
a	2 x 3	No
b	2 x 3	No

```
Variable name: b
Square Matrix: No


Variable value:
 23.0 4.0 2.0
 3.0 43.0 32.0
```

```
>>
```

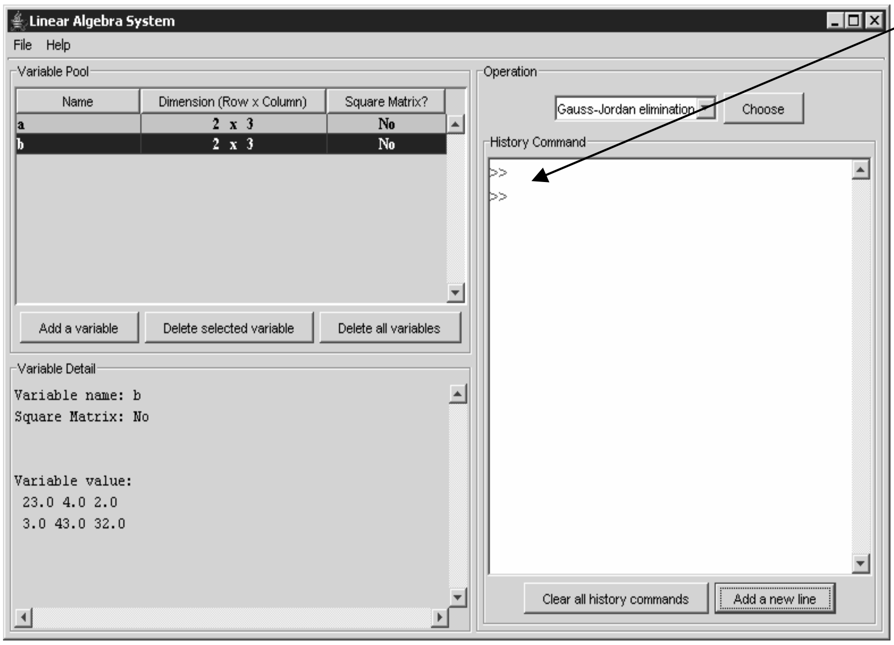


Name	Dimension (Row x Column)	Square Matrix?
a	2 x 3	No
b	2 x 3	No

```
Variable name: b
Square Matrix: No


Variable value:
 23.0 4.0 2.0
 3.0 43.0 32.0
```

```
>>
```

Add a new line from the previous screenshot in history command.

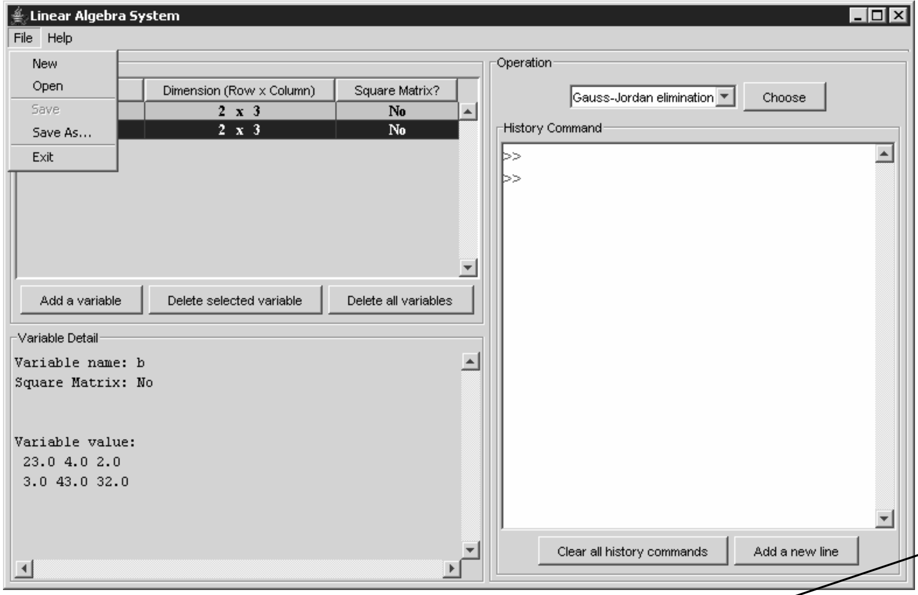


Name	Dimension (Row x Column)	Square Matrix?
a	2 x 3	No
b	2 x 3	No

```
Variable name: b
Square Matrix: No


Variable value:
 23.0 4.0 2.0
 3.0 43.0 32.0
```

```
>>
>>
```

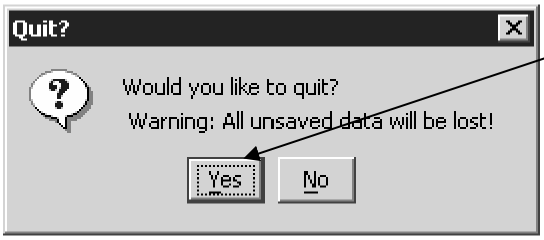
How to exit the system



Dimension (Row x Column)	Square Matrix?
2 x 3	No
2 x 3	No

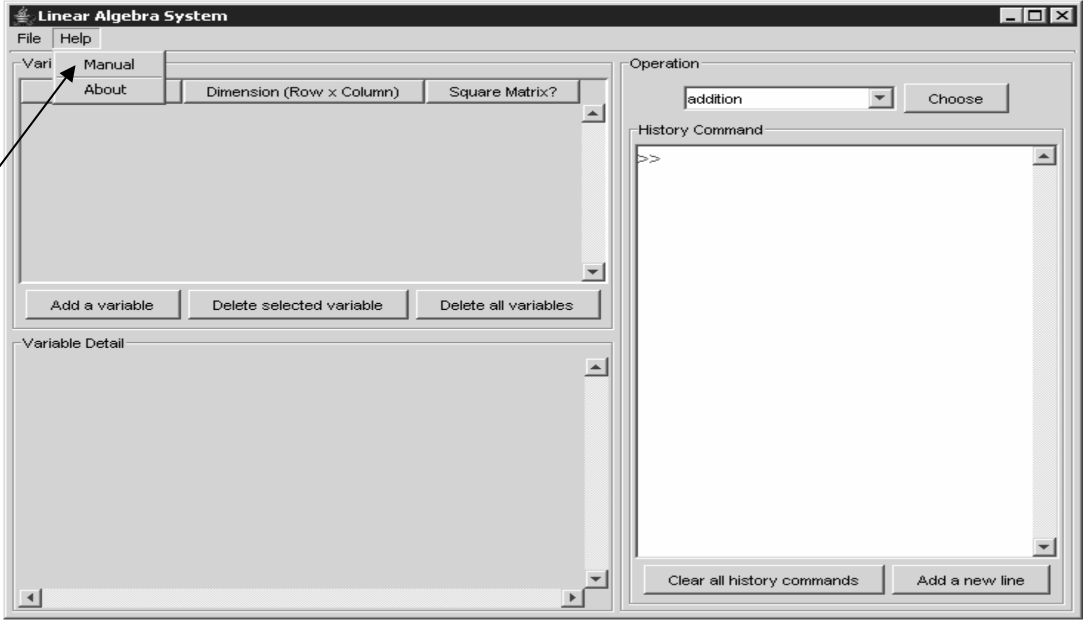
```
Variable name: b
Square Matrix: No


Variable value:
 23.0 4.0 2.0
 3.0 43.0 32.0
```

Click the file, and then click exit to close the system, it will notify you should save file. If you click yes, then the system is closed.



Would you like to quit?
Warning: All unsaved data will be lost!

How to access the help system



Get help by clicking the help menu

11.2 The Detail Code