SPL-1 Project Report

Implementation of Matrix Calculator and Linear System Solver

Submitted by

Muktadul Islam

BSSE Roll No. : 1215

BSSE Session: 2019-2020

Submitted to

Mohd. Zulfiquar Hafiz

Designation: Professor

Institute of Information Technology

Institute of Information Technology

University of Dhaka

[30-05-2022]

Table of Contents

Index of Figures

1. Introduction

Linear Systems are systems of equations in which the variables are never multiplied with each other but only with constants and then summed up. It is used to describe both static and dynamic relations between variables.

In the case of the description of static relations, systems of linear algebraic equations describe invariants between variables such as:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = c_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = c_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = c_3$$

Here, we are interested in the values of x_1 , x_2 and x_3 for which both equations hold. This system of equations can easily be written in matrix form:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \text{x} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Now, we can find the value of x_1 , x_2 and x_3 step by step using Cramer's Rules, Inverse Matrix, Gaussian Elimination, Gauss Jordan Method, Gauss Jacobi Iteration Method, Gauss Seidel Iteration Method in this project.

Encryption is the process of translating plain text data (plaintext) into something that appears to be random and meaningless (cipher text). Decryption is the process of converting cipher text back to plaintext.

The goal of every encryption algorithm is to make it as difficult as possible to decrypt the generated cipher text without using the key. If a really good encryption algorithm is used, then there's no technique significantly better than methodically trying every possible key. For such an algorithm, the longer the key, the more difficult it is to decrypt a piece of cipher text without possessing the key.

2. Background of the Project

2.1. Matrix

As this project is based on matrix, that why I had to implement all the matrix functionalities by using points of the matrix. During implementation of those functionalities I choose all efficient algorithms that I thought during my pervious semester. For instance, for determinant I used Upper Triangle matrix methods etc.

2.2. Linear System Solving

There are several ways to solve a linear system. In this project I had implemented 6 types of them.

2.2.1 Cramer's Rules

In linear algebra, Cramer's rule is an explicit formula for the solution of a system of linear equations with as many equations as unknowns, valid whenever the system has a unique solution. It expresses the solution in terms of the determinants of the (square) coefficient matrix and of matrices obtained from it by replacing one column by the column vector of right-hand-sides of the equations

Cramer's Rule

$$\begin{aligned} a_1x+b_1y+c_1z&=d_1\\ a_2x+b_2y+c_2z&=d_2\\ a_3x+b_3y+c_3z&=d_3 \end{aligned}$$

$$\text{Let } D=\begin{vmatrix} a_1 & b_1 & c_1\\ a_2 & b_2 & c_2\\ a_3 & b_3 & c_3 \end{vmatrix}$$

$$\text{If } D \neq 0 \text{ then}$$

$$x=\frac{\begin{vmatrix} d_1 & b_1 & c_1\\ d_2 & b_2 & c_2\\ d_3 & b_3 & c_3 \end{vmatrix}}{D} \qquad y=\frac{\begin{vmatrix} a_1 & d_1 & c_1\\ a_2 & d_2 & c_2\\ a_3 & d_3 & c_3 \end{vmatrix}}{D} \qquad z=\frac{\begin{vmatrix} a_1 & b_1 & d_1\\ a_2 & b_2 & d_2\\ a_3 & b_3 & d_3 \end{vmatrix}}{D}$$

Figure 2.1 : Cramer's Rule.

2.2.2 Inverse Matrix

Solving linear system using Inverse matrix, requires the linear system in a form

Ax = C,

Here
A = coefficient matrix
x = variable matrix
C = equation value matrix

where |A| is not equal zero. Then we can solve the system by finding the inverse of the coefficient matrix A, then multiply it with the matrix C.

2.2.3 Gaussian Elimination

Gaussian elimination, also known as row reduction, is an algorithm for solving systems of linear equations. It consists of a sequence of operations performed on the corresponding matrix of coefficients. This method can also be used to compute the rank of a matrix, the determinant of a square matrix, and the inverse of an invertible matrix.

$$\begin{cases} 2x_1 - x_2 + 5x_3 = 10\\ x_1 + x_2 - 3x_3 = -2\\ 2x_1 + 4x_2 + x_3 = 1 \end{cases} \Longrightarrow$$

$$\left(\begin{array}{ccc|c} 2 & -1 & 5 & 10\\ 1 & 1 & -3 & -2\\ 2 & 4 & 1 & 1 \end{array}\right) r_2 \to 2r_2 \left(\begin{array}{ccc|c} 2 & -1 & 5 & 10\\ 2 & 2 & -6 & -4\\ 2 & 4 & 1 & 1 \end{array}\right)$$

$$\begin{array}{l} r_3 \to r_3 - r_2\\ r_2 \to r_2 - r_1 \end{array} \left(\begin{array}{ccc|c} 2 & -1 & 5 & 10\\ 0 & 3 & -11 & -14\\ 0 & 2 & 7 & 5 \end{array}\right) r_3 \to 3r_3 \left(\begin{array}{ccc|c} 2 & -1 & 5 & 10\\ 0 & 3 & -11 & -14\\ 0 & 6 & 21 & 15 \end{array}\right)$$

$$r_3 \to r_3 - 2r_2 \left(\begin{array}{ccc|c} 2 & -1 & 5 & 10\\ 0 & 3 & -11 & -14\\ 0 & 0 & 43 & 43 \end{array}\right) r_3 \to \frac{r_3}{43} \left(\begin{array}{ccc|c} 2 & -1 & 5 & 10\\ 0 & 3 & -11 & -14\\ 0 & 0 & 1 & 1 \end{array}\right).$$

$$\begin{cases} 2x_1 - x_2 + 5x_3 = 10\\ \quad 3x_2 - 11x_3 = -14\\ \quad\quad x_3 = 1 \end{cases}$$

Figure 2.2: Gaussian Elimination

2.2.4 Gauss Jordan Method

Gauss-Jordan elimination is another method for solving systems of equations in matrix form. It is really a continuation of Gaussian elimination.

Goal: turn matrix into reduced row-echelon form $\left[\begin{array}{ccc|c} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \end{array}\right]$

Once it is in this form, we can say $x = a$, $y = b$, and $z = c$ or $(x, y, z) = (a, b, c)$.

3. Description of the Project

There are basically three parts in my project:

- ❖ Matrix Calculator
- ❖ Linear System Solving
- ❖ Encryption & Decryption

3.1. Matrix Calculator

In this part, the main goal is to implement all the functionality used in matrix (matrix Inverse, addition, summation, multiplication, determinant, adjoint of matrix etc.) using pointer. Here I have implemented all of those functionalities in two ways. One for normal matrix and another for fractional matrix. And combined them in a way so that less memory will used and can run faster.

3.2. Linear System Solving

This part represents the solution of any kinds of Linear System given by the user. First it takes the Linear equation, then asked the user in which method it will be solved? User option will be like,

1. Cramer's Rules.
2. Inverse Matrix.
3. Gaussian Elimination.
4. Gauss Jordan Method.
5. Gauss Jacobi Iteration Method.
6. Gauss Seidel Iteration Method.

After taking user choice it start to solve the equations by the methods and storing the solution in file. So that user can use the solution in latter.

3.3. Encryption & Decryption

This part is for "Data Cryptography". In this part I have implemented two different function. First function is for encryption. It will take a plaintext and convert it into a matrix form using ASCII values. Then the matrix will be multiplied by another matrix call "Key”. The result of the multiplication will be the encrypted text which will be not understandable without the Key.

The second function is for decryption. Here it first calculate the inverse matrix of the Key. Then take the encrypted text and convert it into matrix form and multiply with the invers Key matrix to find out the plaintext (actual data) and store the text in a file.

$$\text{ABCD} \rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix} \rightarrow \begin{bmatrix} 65 & 66 \\ 67 & 68 \end{bmatrix} \longrightarrow \begin{bmatrix} 4 & 1 \\ -2 & 3 \end{bmatrix} \times \begin{bmatrix} 65 & 66 \\ 67 & 68 \end{bmatrix} = \begin{bmatrix} 327 & 332 \\ 71 & 72 \end{bmatrix}$$

327, 332, 71, 72 [Encrypted Value]

$$\begin{bmatrix} 4 & 1 \\ -2 & 3 \end{bmatrix}^{-} \times \begin{bmatrix} 327 & 332 \\ 71 & 72 \end{bmatrix} = \begin{bmatrix} 65 & 66 \\ 67 & 68 \end{bmatrix} \rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix} \rightarrow \text{ABCD}$$

4. Implementation and Testing

The main goal of my project is to solve any kinds of linear equation and Encryption-Decryption using matrix. To do that I have create some user define "header file", which contains the prototype of all the function in this project I would use in future.

```
#include<bits/stdc++.h>
using namespace std;

int * input_matrix_from_file(char *fileName, int *row, int *column);
int * input_matrix_from_console(int *row, int *column);
int * matrix_memory_optimization(int * matrix, int row, int column);
void print_matrix(int *matrix, int row, int column);

int * matrix_addition(int *matrix1, int row1, int column1, int *matrix2, int row2, int column2);
int * matrix_subtraction(int *matrix1, int row1, int column1, int *matrix2, int row2, int column2);
int * matrix_multiplication(int *matrix1, int row1, int column1, int *matrix2, int row2, int column2);
int * matrix_multiplication_with_constant(int constant, int *matrix, int row, int column);

void matrix_determinant(int *matrix, int row, int column, int *value);
int matrix_determinant(int *matrix, int row, int column);
void subMatrix(int *matrix, int *newMatrix, int row, int removeRow, int removeColumn);

int * matrix_transpose(int *matrix, int row, int column);
int * matrix_adjoint(int *matrix, int row, int column);
int * matrix_inverse(int *matrix, int row, int column);


int lcm(int x, int y);
long long lcm(long long x, long long y);
int gcd(int x, int y);
long long gcd(long long x, long long y);
char * getNumber(char *p, char sign, int *matrix);
bool is_fractional_matrix(int *matrix);
```

Figure 4.1 : Matrix header file.

Here I declare all the function prototype related to matrix and implement them in other cpp file stored in a separated folder.

```
#include<bits/stdc++.h>
#define EPSILON 1e-9

using namespace std;

void input_equation_from_file(char *fileName, int *row, int *column,
    int **coefficientMat, int **Dmatrix, char **variablesMat);

void write_introduction_part(string title, string equationFileName,string solutionFileName,
        int *coefficientMatrix, int row, int column, int *dMat, char *variables);

int rowElementSmaller(int *coefficientMatrix, int row, int column, int dMat[], int rowNo);
void makeLowerTriangleMatrix(string solutionFileName, int *coefficientMatrix, int row,
                int column, int dMat[]);

int max_length_of_number(int *matrix, int row, int column);

void solution_by_cramersRules(char *equationFileName, char *solutionFileName);
void solution_by_inverseMatrix(char *equationFileName, char *solutionFileName);
void solution_by_GaussianElimination(char *equationFileName, char *solutionFileName);
void solution_by_GaussJordanMethod(char *equationFileName, char *solutionFileName);

void gauss_jacobi_iteration_method(char *equationFileName);
void gauss_seidel_iteration_method(char *equationFileName);
```

Figure 4.2 : Linear System Solving header file.

Here I declare all the function prototype related to linear system solving and implement them in other cpp file stored in a separated folder. And as will as for Encryption & Decryption.

```
#include<bits/stdc++.h>
using namespace std;

int * key(int *row, int *column);
void encryption(string plaintextFile, string encryptedFile);
void decryption(string encryptedFile, string decryptedFile);


char * stringToMatrix(int *matrix, int row, int column, char str[]);
string matrixToString(int *matrix, int row, int column);
short * integerMatrix_to_shortMatrix(int *matrix, int row, int column);
```

Figure 4.3 : Encryption & Decryption header file.

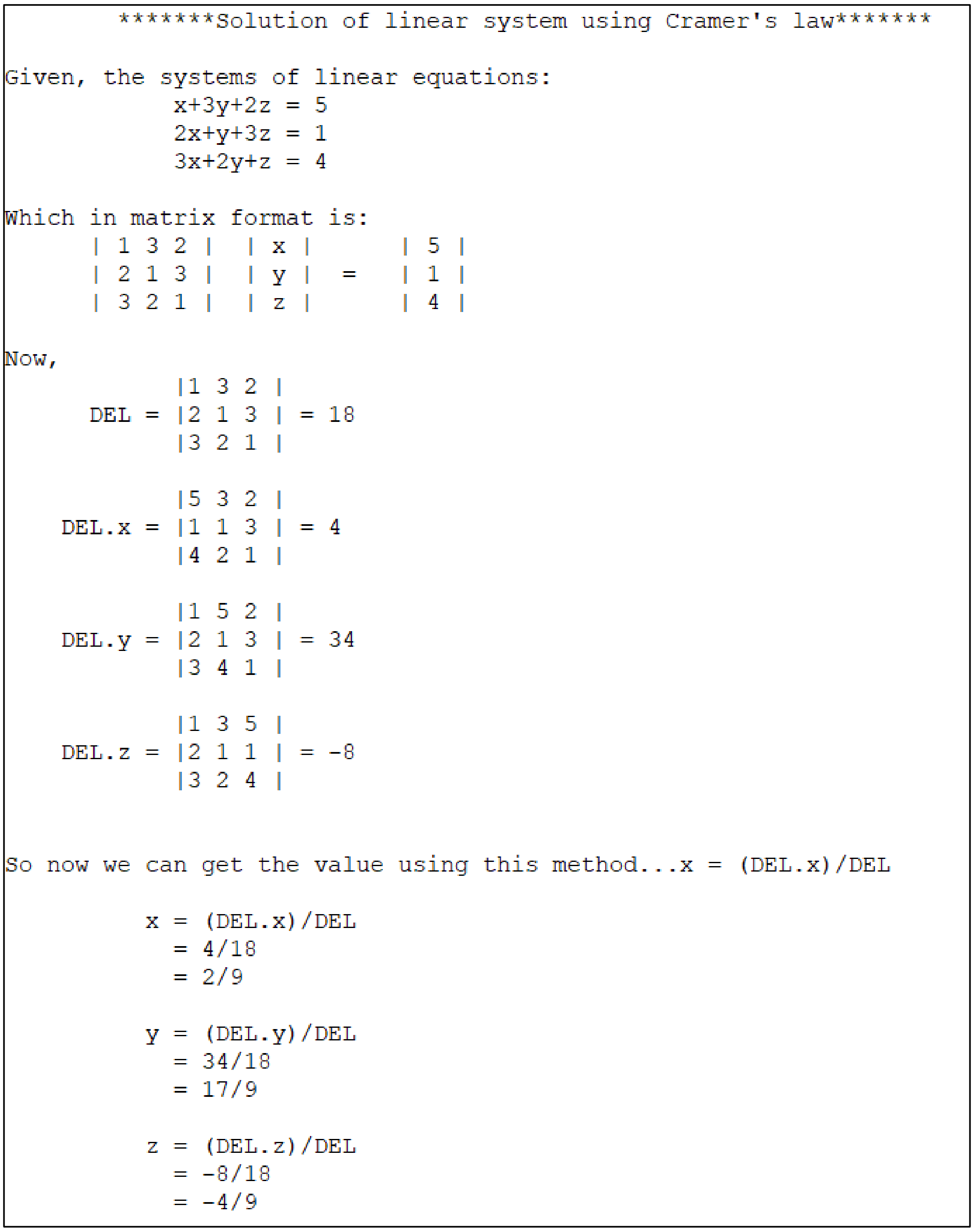
4.1 Matrix Calculation

In matrix calculation part, most significant part was determinant and inverse matrix. I had implemented the determinant function in such a way that it will make copy of the matrix and convert it into an equivalent upper triangular matrix. And then determinant can easily found by multiply the diagonal elements.

In Inverse matrix, first I check that it’s determinant is equal zero or not. If the determinant is not equal zero then I calculate the Adjoint matrix of that matrix and divide it by the determinant to find out the inverse.

4.2 Linear System Solving

To solve a linear system I had implement 6 algorithms into functions. Those functions solve the linear equation step by step and store it in a text file. So that user can read it in future.



```
        *******Solution of linear system using Cramer's law*******

Given, the systems of linear equations:
            x+3y+2z = 5
            2x+y+3z = 1
            3x+2y+z = 4

Which in matrix format is:
      | 1 3 2 |   | x |      | 5 |
      | 2 1 3 |   | y |  =   | 1 |
      | 3 2 1 |   | z |      | 4 |

Now,
            |1  3  2  |
      DEL = |2  1  3  | = 18
            |3  2  1  |

            |5  3  2  |
    DEL.x = |1  1  3  | = 4
            |4  2  1  |

            |1  5  2  |
    DEL.y = |2  1  3  | = 34
            |3  4  1  |

            |1  3  5  |
    DEL.z = |2  1  1  | = -8
            |3  2  4  |


So now we can get the value using this method...x = (DEL.x)/DEL

          x = (DEL.x)/DEL
            = 4/18
            = 2/9

          y = (DEL.y)/DEL
            = 34/18
            = 17/9

          z = (DEL.z)/DEL
            = -8/18
            = -4/9
```

Figure 4.4 : Solution of a linear equation using Cramer's Law

4.3 Data Cryptography

In the Encryption part first plain text is converted into a matrix then multiply with a matrix (Key) and store it into a Binary file. And for decryption to the thing again but this time I used inverse matrix of the Key and store the decrypted data in a text file.

```
// this section for Encryption
char text[tempString.length()], *plaintext;
strcpy(text, tempString.c_str());
plaintext = text;

int row, column, *matrix1, *KEY;
short *matrix2;
KEY = key(&row, &column);
int mat[column][row];

while(*plaintext != '\0'){
    plaintext = stringToMatrix(&mat[0][0], column, row, plaintext);

    matrix1 = matrix_multiplication(KEY, row, column, &mat[0][0], column, row);
    write.write((char *) matrix1, sizeof(int)*row*column);
}
```

Figure 4.5 : Encrypting the Plaintext

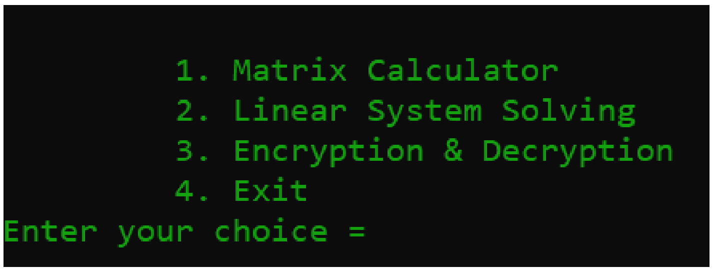
```
matrix1 = matrix_inverse(KEY, row, column);
matrix2 = matrix_multiplication(matrix1, row, column, &mat[0][0], column, row);

write << matrixToString(matrix2, row, column);
free(matrix1);
free(matrix2);
```

Figure 4.6 : Decrypting the Encrypted text

5. User Interface

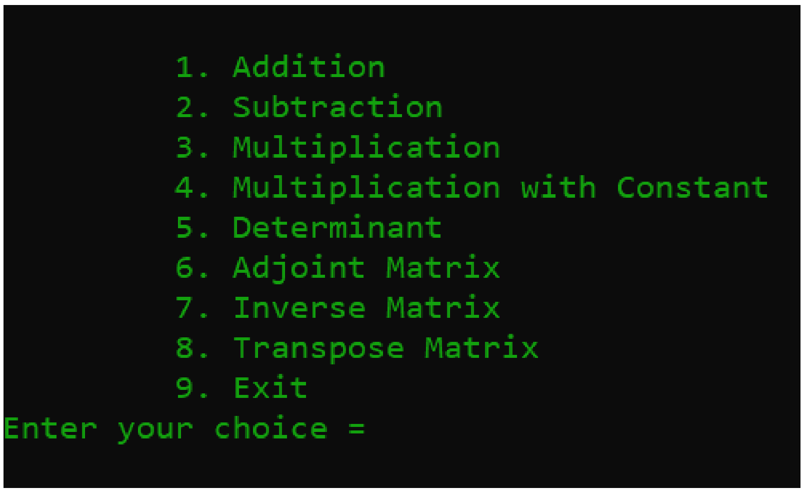
After opening the console window, we get to choose option that which operation of Matrix Calculator, Linear System Solving or Encryption & Decryption.



```
         1. Matrix Calculator
         2. Linear System Solving
         3. Encryption & Decryption
         4. Exit
Enter your choice =
```

Figure 5.1 : Main Functions

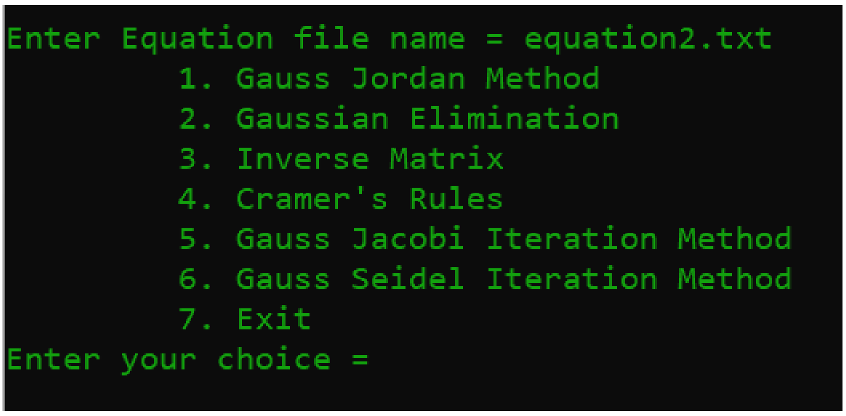
After choosing the option 1, we get another option for all matrix related operation.



```
         1. Addition
         2. Subtraction
         3. Multiplication
         4. Multiplication with Constant
         5. Determinant
         6. Adjoint Matrix
         7. Inverse Matrix
         8. Transpose Matrix
         9. Exit
Enter your choice =
```

Figure 5.2 : Matrix Related Functions

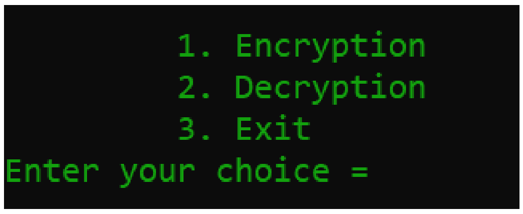
If we choose option 2, we will get option to solve the given linear problem in 6 ways.



```
Enter Equation file name = equation2.txt
         1. Gauss Jordan Method
         2. Gaussian Elimination
         3. Inverse Matrix
         4. Cramer's Rules
         5. Gauss Jacobi Iteration Method
         6. Gauss Seidel Iteration Method
         7. Exit
Enter your choice =
```

Figure 5.3 : Linear System Solving Functions

If we choose option 3, we will get option for Encryption and Decryption



```
         1. Encryption
         2. Decryption
         3. Exit
Enter your choice =
```

Figure 5.4 : Encryption and Decryption Functions

6. Challenges Faced

There are a number of challenges I had to face while implementing the software. A lot of terms and the majority of the tasks were completely new to me that led me towards much confusion and complexity in implementation. Some of the challenges I faced during this implementation is enlisted below-

- Working with header files for the first time.
- Working with multiple source files.
- It was a challenge to process the input file because the written format has no specific structure.
- Had to get clear information about these algorithms
  - Gaussian Elimination.
  - Gauss Jordan Method.
  - Gauss Jacobi Iteration Method.
  - Gauss Seidel Iteration Method.
  - Cramer's Rules.
  - Hill Cipher algorithm (Encryption and Decryption)
- Had to get all functionaliets of Matrix

7. Conclusion

From the project, I have learnt the operations of matrix and Linear Algebra. I have the algorithms, which can help me in future studies. I have learnt how to handle everything using pointer. I have also learnt more details about encryption and decryption.

My future plan is to extend the project. I want to make a mobile app that can solve all linear equation through the function implemented in this project and a Messenger Chat system, in which data will be transfer by being encrypted through my encryption function.

Reference

https://math.libretexts.org/Bookshelves/Applied_Mathematics/Applied_Finite_Mathematics_(Sekhon_and_Bloom)/02%3A_Matrices/2.05%3A_Application_of_Matrices_in_Cryptography 20-05-2022

https://www.ajer.org/papers/v6(06)/ZB0606212217.pdf 26-05-2022

Strang, Gilbert, et al. *Introduction to linear algebra*. Vol. 3. Wellesley, MA: Wellesley-Cambridge Press, 1993.

Poole, David. *Linear algebra: A modern introduction*. Cengage Learning, 2014.

Hogben, Leslie, ed. *Handbook of linear algebra*. CRC press, 2013.

[1] URL, Name of the page, last accessed on dd MMM YYYY

[2] Academic Paper title, author names, conference / journal it was published, year, page number