Iterative Methods for solving the Linear System

Kazi Md Masum Billah
Mary Mackay

December 7, 2015

1 Introduction

Gauss Elimination is a method of solving the system of linear equation directly. The name of this method from Carl Friedrich Gauss(1777-1855). Gauss Elimination method can be used to determine the inverse matrix, determinant and rank of a matrix. It has an important role in the field of computational science and engineering.This method is also known as successive row reduction method. Gauss Elimination uses the following techniques: i) Swap the position of two rows, ii) Multiply a row by nonzero scalar and iii) Add to one row a scalar multiple of another [1]. These steps provide the upper triangular matrix and then by for backward substitution unknowns values can be obtained.To solve the linear system inverse matrix is also helpful. Before do so Gauss Elimination is needed to create the augmented matrix. And then by forward of backward substitution we can obtain the value of unknowns. For larger dimension matrix the significance loss of error is more in this method. This method is only applicable where the matrix is invertible. To solve the bigger system of linear equation LU factorization is a convenient way. It also involves the forward and backward substitution to get the value of unknowns [2]. For this project we have the problem and we solved these by Gauss Elimination, Inverse Matrix, and LU Factorization.

2 Motivation

In linear algebra, the main uniqueness of Gauss Elimination is that we can solve directly. To find out the computational cost interms of time we are comparing three techniques. There are frequently using the Gauss Elimination, Inverse Matrix Method and Gauss Elimination with LU factorization in lots of computational work.

3 Methodology

3.1 Gauss Elimination

Algorithm for Gauss Elimination:

```
for $k = 1$ to $n - 1$
for $i = k + 1$ to $n$
for $j = k + 1$ to $n + 1$
$a_{ij} = a_{ij}\ \ a_{ik}\ / a_{kk}\ * akj$
Compute $x_n = a_{n,n+1}/a_{nn}$
for $k = n - 1$ to 1
sum = 0
for $j = k + 1$ to $n$
sum = sum + $a_{kj}$ * $x_j$
$x_k$= 1/$a_{kk}$ * ($a_{k,n+1} sum$)
stop
```

3.2 Inverse matrix

Inverse Matrix Algorithm

```
for $i = 0$ to number of rows
for $j = 0$ to number of rows
$gauss_{ij}$=$a_{ij}$ for $i = 0$ to number of rows
for $j$ =number of rows to 2 times of number of rows
if($i$+number of rows==$j$)
$gauss_{ij} = 1$
else
$gauss_{ij} = 0$
do gaussian elimination
save inverse matrix.
```

3.3 LU Factorization

LU Factorization refers to the factorization of coefficient Matrix A. i.e. A=LU, Where L is the lower triangular Matrix and U is the Upper Triangular Matrix. The expression of system of linear Expression will be Ax=b.

$$A = \begin{bmatrix} a_{11} & a_{12} & .. & .. & a_{1n} \\ a_{21} & a_{22} & .. & .. & a_{2n} \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ a_{n1} & a_{n2} & .. & .. & a_{nn} \end{bmatrix}$$

$$x^T = [x_1\ x_2\ ..\ ..\ x_n] \quad b^T = [b_1\ b_2\ ..\ ..\ b_n]$$

LU Factorization Algorithm:

```
1  Input the element of coefficient matrix A and Constant matrix b.
2  for $k = 1$ to$n$
3  for $s = 1$ to $k - 1$
4  $L_{kk}$ * $U_{kk}$ = $A_{kk}$ -$L_{sk}$*$U_{sk}$
5  for $j = k + 1$ to $n$
6  for $s = 1$ to $k - 1$
7  $U_{kj} = (A_{kj}\ \ L_{sk} * U_{sk})/L_{kk}$
8  for $i = k + 1$ to $n$
9  for $s = 1$ to $k - 1$
10 $L_{ik} = (A_{ik}\ \ L_{is} * U_{sk})/U_{kk}$ print out of $L_{ij}$ and $U_{ij}$
```

After getting the lower and upper triangular matrix the system will be LUx=b. Now in two steps i.e backward and forward substitution we have to find out the unknowns.
Forward Substitution: From Ly=b system we have to calculate the y vector.

$$\begin{bmatrix} b_1 \\ b_2 \\ .. \\ .. \\ b_n \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & .. & .. & .. & 0 \\ l_{21} & l_{22} & 0 & .. & .. & .. & 0 \\ .. & .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. & .. \\ l_{n1} & l_{n2} & l_{n3} & .. & .. & .. & l_{nn} \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ .. \\ .. \\ y_n \end{bmatrix}$$

Backward Substitution: From Ux=y system we have to calculate the x vector.

$$\begin{bmatrix} y_1 \\ y_2 \\ .. \\ .. \\ y_n \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & u_{13} & .. & .. & .. & u_{1n} \\ 0 & u_{22} & u_{23} & .. & .. & .. & u_2n \\ 0 & 0 & u_{33} & .. & .. & .. & u_{3n} \\ .. & .. & .. & .. & .. & .. & .. \\ 0 & 0 & .. & .. & .. & .. & u_{nn} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ .. \\ .. \\ x_n \end{bmatrix}$$

4 Problem a:

$$Matrix\ A = \begin{bmatrix} 1 & 6 & 0 \\ 2 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \ Matrix\ b = \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}$$

To solve this, we obtained and used the Lower and Upper matrices:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & -0.18 & 1 \end{bmatrix} \ U = \begin{bmatrix} 1 & 6 & 0 \\ 0 & -11 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Alternatively, we obtained and used the inverse matrix:

$$\begin{bmatrix} -0.0909091 & 0.545455 & 0 \\ 0.181818 & -0.0909091 & -0 \\ -0.363636 & 0.181818 & 1 \end{bmatrix}$$

5 Problem b:

$$MatrixA = \begin{bmatrix} -1 & 1 & 0 & -3 \\ 1 & 0 & 3 & 1 \\ 0 & 1 & -1 & -1 \\ 3 & 0 & 1 & 2 \end{bmatrix} \quad Matrix\ b = \begin{bmatrix} 4 \\ 0 \\ 3 \\ 1 \end{bmatrix}$$

To solve this, the Lower and Upper matrices:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -0 & 1 & 1 & 0 \\ -3 & 3 & 2 & 1 \end{bmatrix} \quad U = \begin{bmatrix} -1 & 1 & 0 & -3 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & -4 & 1 \\ 0 & 0 & 0 & -3 \end{bmatrix}$$

And alternatively, the inverse matrix:

$$\begin{bmatrix} 0.416667 & -0.333333 & -0.416667 & 0.583333 \\ -0.583333 & 0.666667 & 1.58333 & -0.416667 \\ 0.0833333 & 0.333333 & -0.0833333 & -0.0833333 \\ -0.666667 & 0.333333 & 0.666667 & -0.333333 \end{bmatrix}$$

6 Problem c:

$$Matrix\ A = \begin{cases} a_{i,j} = 0.01 & if\ 1 \leq i \leq n-1, j = i. \\ a_{i,j} = 1 & if\ 1 \leq i \leq n, j = n. \\ a_{i,j} = -1 & if\ j \geq n. \end{cases}$$

$$Matrix\ b = \begin{cases} b_i = 2.1 - i & Where\ i = 1, 2....., n \\ b_n = 2 - n \end{cases}$$

Lets, n=3; Then

$$MatrixA = \begin{bmatrix} 0.1 & 0 & 1 \\ -1 & 0.1 & 1 \\ -1 & -1 & 1 \end{bmatrix} \quad Matrix\ b = \begin{bmatrix} 2.1 \\ 1.1 \\ -1 \end{bmatrix}$$

To solve this, the Lower and Upper matrices:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -10 & 1 & 0 \\ -10 & -10 & 1 \end{bmatrix} \quad U = \begin{bmatrix} 0.1 & 0 & 1 \\ 0 & 0.1 & 11 \\ 0 & 0 & 121 \end{bmatrix}$$

And alternatively, the inverse matrix:

$$\begin{bmatrix} 0.909091 & -0.826446 & -0.0826446 \\ 0 & 0.909091 & -0.909091 \\ 0.909091 & 0.0826446 & 0.00826446 \end{bmatrix}$$

7 Results and Discussion

Using LU factorization we have the following results:
(a).

$$x^T = \begin{bmatrix} 0.272727 & 0.454545 & 0.0909091 \end{bmatrix}$$

(b).

$$x^T = \begin{bmatrix} 1 & 2 & -0 & -11 \end{bmatrix}$$

(c).

$$x^T = \begin{bmatrix} 1.08264 & 1.90909 & 1.99174 \end{bmatrix}$$

The computation cost for a) 5 flops (5e-06 seconds) b) 7 flops (7e-06 seconds) c) 5 flops (5e-06 seconds).

Using Gauss Elimination with Inverse we have got the following results:
(a).

$$x^T = \begin{bmatrix} 0.272727 & 0.454545 & 0.0909091 \end{bmatrix}$$

(b).

$$x^T = \begin{bmatrix} 1 & 2 & 4.16334e-17 & -1 \end{bmatrix}$$

(c).

$$x^T = \begin{bmatrix} 1.08264 & 1.90909 & 1.99174 \end{bmatrix}$$

The computation cost for a) 6 flops (6e-06 seconds) b) 9 flops (9e-06 seconds) c) 5 flops (5e-06 seconds). So we can conclude that the LU factorization has less computational cost than Gauss elimination with inverse matrix.

8 Conclusion

We learned a great deal in working on this project. Specifically, we had the opportunity to compare the methods of using LU decomposition and computing the inverse to solve a matrix problem. As there seemed to be more steps involved in the process involving LU factorization, we hypothesized that it would be the less efficient method, however, we were wrong. We found this to be very interesting and this would certainly be knowledge that could serve us well in the future. We will likely find ourselves working in further studies involving computational optimization. Lessons like this, in which we should consider other algorithms to potentially improve our code, will be something we will keep in mind.

The greatest challenge involved distinguishing between the two methods as they were similar. For example, in solving using LU decomposition, we initially decomposed the matrix and then used Gaussian elimination to solve for the other arrays. However, we determined later that we should instead have solved by simple substitution.

Another challenge faced was in implementing the Gaussian elimination to solve for the inverse matrix, specifically because the matrix we used was not a square matrix. For example, our code can solve for the inverse matrix of:

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

by constructing the matrix:

$$\begin{bmatrix} 1 & 2 & 3 & 1 & 0 & 0 \\ 4 & 5 & 6 & 0 & 1 & 0 \\ 7 & 8 & 9 & 0 & 0 & 1 \end{bmatrix}$$

and then by using Gaussian elimination. Specifically, we had to cycle only through the first half of the matrix in determining the factors, however, the factors had to be used throughout the entire length of the matrix. This made the loops somewhat complicated, but eventually we were able to get it to work.

We did not find any part of this project to be unnecessary. Even the initial mistake we had made in using Gaussian elimination to solve in the LU decomposition was useful to us later on when we needed that code to compute the inverse matrix.

References

[1] Gauss Elimination, *Wikipedia*.

[2] Joseph F. Grcar, Mathematicians of Gaussian Elimination. American Mathematical Society, VOL- 58, NUM-6, P-(782-792).