Final Project
Math 5370

Iterative Methods for Solving Linear System

Kazi Md Masum Billah
Mary Mackay

The University of Texas at El Paso (UTEP)
El Paso, Texas 79968, USA

November 24, 2015

- Gauss Elimination: It uses the successive row reduction.
    - Forward substitution.
    - Backward substitution
- Inverse matrix method can be implemented to solve the system of linear equation.
- LU factorization provides the more convenient way to solve the larger system of linear equation. It also involves the forward and backward substitution after decomposing the coefficient matrix in to lower and upper triangular matrix.
- Algorithm for Gauss Elimination is following:

```
for k = 1 to n − 1 do
    for i=k + 1 to n do
        for j = k + 1 to n + 1 do
            a[i][j] = a[i][j]−a[i][k]/a[k][k] ∗ a[k][j];
            Compute x[n] = a[n][n + 1]/a[n][n];
        end
    end
end
for k = n − 1 to 1 do
    sum = 0;
end
for j = k + 1 to n do
    sum = sum +a[k][j] ∗ x[j];
    x[k] = 1/a[k][k] ∗ (a[k][n + 1]− sum);
end
```

- Inverse Matrix: To find the inverse matrix at first an augmented matrix created. Then it uses the Gauss elimination algorithm.
- Algorithm to find the inverse matrix:
  - do for i=0 to number of rows
    do for j=0 to number of rows
    Gauss[i][j]=a[i][j]
  - do for i=0 to number of rows
    do for j=number of rows to 2 times of number of rows
    if(i+number of rows==j)
    Gauss[i][j]=1
    else
    Gauss[i][j]=0
  - do Gaussian elimination
    Save inverse matrix.

- 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.

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

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

```
Input elements of coefficient matrix A and Constant matrix b.
 for k = 1 to n do
     for s = 1 to k - 1 do
         L[k][k] * U[k][k] = A[k][k] - L[s][k] * U[s][k];
     end
end
for j = k + 1 to n do
     for s = 1 to k - 1 do
         U[k][j] = (A[k][j]-L[s][k] * U[s][k])/L[k][k];
     end
end
for i = k + 1 to n do
     for s = 1 to k - 1 do
         L[i][k] = (A[i][k]-L[i][s] * U[s][k])/U[k][k];
     end
end
```

- LUx=b
- Compute Lower triangular Matrix and Upper triangular Matrix.

$$\text{Lower Triangular Matrix}, L = \begin{bmatrix} l_{11} & 0 & 0 & .. & .. & .. & 0 \\ l_{21} & l_{22} & 0 & .. & .. & .. & 0 \\ .. & .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. & .. \\ l_{n1} & l_{n2} & l_{n3} & .. & .. & .. & l_{nn} \end{bmatrix}$$

$$\text{Upper Triangular Matrix}, U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & .. & .. & .. & u_{1n} \\ 0 & u_{22} & u_{23} & .. & .. & .. & u_{2n} \\ 0 & 0 & u_{33} & .. & .. & .. & u_{3n} \\ .. & .. & .. & .. & .. & .. & .. \\ 0 & 0 & .. & .. & .. & .. & u_{nn} \end{bmatrix}$$

- Now solve Ly=b for y vector.

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

- Now solve Ly=b for y vector.

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

- Problem a.

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

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:

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

- Problem b.

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

To solve this, the Lower and Upper matrices:

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

And alternatively, the inverse matrix:

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

- Problem c.

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

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

- Lets, n=3; Then

$$\text{Matrix } A = \begin{bmatrix} 0.1 & 0 & 1 \\ -1 & 0.1 & 1 \\ -1 & -1 & 1 \end{bmatrix} \text{ Matrix } \; b = \begin{bmatrix} 2.1 \\ 1.1 \\ -1 \end{bmatrix}$$

To solve this, the Lower and Upper matrices:

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

And alternatively, the inverse matrix:

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

- Results(LU factorization)
  - Problem (a).

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

Using LU decomposition, it took: 5 flops (5e-06 seconds) for the calculation.

  - Problem (b).

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

Using LU decomposition, it took: 7 flops (7e-06 seconds) for the calculation.

  - Problem (c).

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

Using LU decomposition, it took: 5 flops (5e-06 seconds) for the calculation.

- Result (Gauss Elimination with Inverse)
  - Problem (a).

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

    Using the inverse of A to solve, it took: 6 flops (6e-06 seconds) for the calculation.
  - Problem (b).

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

    Using the inverse of A to solve, it took: 9 flops (9e-06 seconds) for the calculation.
  - Problem (c).

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

    Using the inverse of A to solve, it took: 5 flops (5e-06 seconds) for the calculation.

```
void Matrix::computeLU(){
double factor;
for (int i = 0; i <A.rows; i++) do
    for (int j = 0; j<A.cols; j++) do
        if i == j then
            L.mat[i][j] = 1;
        end
        else
            L.mat[i][j] = 0;
        end
        U.mat[i][j]=A.mat[i][j];
    end
end
```

```
for (int index= 0; index<A.cols−1; index++) do
    for (int i =index+1; i <A.rows; i + +) do
        factor = −U.mat[i][index]/U.mat[index][index];
        for (int j = 0; j <A.cols; j + +) do
            U.mat[i][j]=factor*U.mat[index][j]+U.mat[i][j];
            L.mat[i][index]=-factor;;
        end
    end
end
}
```

```
void Matrix::SolveLU(){
y.cols = 1;
x.cols = 1;
for (int i = 0; i <L.rows; i++) do
    y.mat[i][0]=b.mat[i][0];
end
for (int j = 0; j < i; j++) do
    y.mat[i][0]- =L.mat[i][j] * y.mat[j][0];
    y.mat[i][0]/ =L.mat[i][i];
end
```

```
for (int i =U.rows−1; i >= 0; i − −) do
    x.mat[i][0]=y.mat[i][0];
end
for (int j =U.rows−1; j > i; j − −) do
    x.mat[i][0]− = U.mat[i][j] ∗ x.mat[j][0];
    x.mat[i][0]/ =U.mat[i][i];
end
}
```

```
void Matrix::SolveInverse(){
findinverse();
double tempsum = 0;
y.cols= 1;
for (int i = 0; i <A.rows; i + +) do
    for (int j = 0; j <A.cols; j + +) do
        tempsum += Gauss.mat[i][j+A.cols]*b.mat[j][0];
    end
    y.mat[i][0]=tempsum;
    tempsum = 0;
end
}
```

- Our Instructor Dr. Sharma
  The unlimited support, teaching, and guidance.
- Collegues: for moral support and team work

Thank You ALL!

Question ?????