|  |  |  |
| --- | --- | --- |
| Numerical Analysis | June 19  2012 | |
| The aim of this assignment is to compare and analyze the behavior of different numerical methods studied in class for solving Simultaneous Linear Equations (Gaussian elimination with and without pivoting, LU Decomposition, the Jacobi and Gauss-Seidel Methods) | | Solving linear system of equations |

Contributors

1. Ahmed Magdy (11)
2. Ahmed Hesham (15)
3. Ashraf Saleh (20)
4. A’amer Mohamed Attia (21)
5. Raed Ahmed Selim (35)

Table of Contents

[1- Problem statement 2](#_Toc327895039)

[2- Pseudo Code 3](#_Toc327895040)

[a- Forward elimination pseudo code 3](#_Toc327895041)

[b- back substitution pseudo code 4](#_Toc327895042)

[c- forward substitution pseudo code 4](#_Toc327895043)

[d- LUdecomposition pseudo code 5](#_Toc327895044)

[e- LU solve pseudo code 6](#_Toc327895045)

[f- Gauss Seidel pseudo code 6](#_Toc327895046)

[g- Jacobi pseudo code 7](#_Toc327895047)

[3- Analysis 8](#_Toc327895048)

[a- Ex1 8](#_Toc327895049)

[b- Ex2 9](#_Toc327895050)

[c- Ex3 10](#_Toc327895051)

[d- Ex4 11](#_Toc327895052)

[e- Ex5 12](#_Toc327895053)

[f- Ex6 13](#_Toc327895054)

[4- Conclusions from analysis 14](#_Toc327895055)

[5- Each method analysis 16](#_Toc327895056)

[6- Data structure used 18](#_Toc327895057)

[7- User Guide 19](#_Toc327895058)

1. Problem statement

The aim of this assignment is to compare and analyze the behavior of different numerical methods studied in class for solving Simultaneous Linear Equations (Gaussian elimination with and without pivoting, LU Decomposition, the Jacobi and Gauss-Seidel Methods)

Required

* Implementation of the above methods
* Analysis of performance for each method
* Analysis of solved examples using each method

1. Pseudo Code
2. Forward elimination pseudo code

Input: augmented matrix A, boolean pivoting

Output: vector x

For k = 1 ... m:

if pivoting = true

//Find pivot for column k

r\_max := k

for r := k+1 …. m do

if abs(A[r,c]) > abs(A[r\_max,c]) then

r\_max := r

end if

end for

//swap rows

swap rows(k, r\_max)

end if

if A[k, k] = 0

error "Matrix is singular!"

//Do for all rows below pivot:

for r = k + 1 ... m:

//Do for all remaining elements in current row:

for c = k ... n:

A[r, c] :=

A[r, c] - A[k, c] \* (A[r, k] / A[k, k])

end for

1. back substitution pseudo code

Input : coefficient matrix A , right hand side vector b

Output : solution vector x

nrow := size(A)

x[nrow] := b[nrow] / A[nrow, nrow]

for i = nrow - 1 to 1

x[i] :=

end for

1. forward substitution pseudo code

Input : coefficient matrix A , right hand side vector b

Output : solution vector x

nrow := size(A)

for i = 1 to nrow

x[i] :=

end for

1. LUdecomposition pseudo code

Input : matrix A

Output : lower matrix L , upper matrix U

//initialize L and U to zeros:

for r = 1…m

for c = 1….n

U[r, c] = 0

if r = c

L[r, c] = 1

else

L[r, c] = 0

end for

end for

for k = 1 ... m

//Find pivot for column k

r\_max := k

for r := k+1 …. m do

if abs(A[r,c]) > abs(A[r\_max,c]) then

r\_max := r

end if

end for

swap rows(k, r\_max)

if A[k, k] = 0

error "Matrix is singular!"

//Do for all rows below pivot:

for r = k + 1 ... m:

//set the U matrix

L[r,k] := A[r, k] / A[k, k]

//Do for all remaining elements in current row

for c = k ... n:

U[r, c] :=

A[r, c] - A[k, c] \* (A[r, k] / A[k, k])



1. LU solve pseudo code

Input : matrix A

output : solution vector x

L,U = LUdecomposition(A)

Z = forward substitution(L, C)

X = backward substitution(U, Z)

1. Gauss Seidel pseudo code

Input: matrix A , int MaxIteration , int maxError

Output: solution vector x

//Rearranging matrix A to be diagonally dominant:

Diagonal = rearrange(A)

If(diagonal = false)

Error “not diagonally dominant”

While( itr < MaxIteration)

For i = 1 to nrow

err = 0

newX[i] :=

err = max(err , (newX[i]-x[i]) / x[i])

newX[i] = x[i]

end for

if(error < maxError)

return

end if

end while

1. Jacobi pseudo code

Input: matrix A , int MaxIteration , int maxError

Output: solution vector x

//Rearranging matrix A to be diagonally dominant:

Diagonal = rearrange(A)

If(diagonal = false)

Error “not diagonally dominant”

While( itr < MaxIteration)

For i = 1 to nrow

err = 0

newX[i] :=

err = max(err , (newX[i]-x[i]) / x[i])

end for

x[1:n] = newX[1:n]

if(error < maxError)

return

end if

end while

1. Analysis

All the analysis done in double precision

R = real solution , C = calculated solution

1. Ex1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | B |  | R |
| 0 | -7 | 0 | 7 | 0 |
| -3 | 2.099 | 6 | 3.901 | -1 |
| 5 | -1 | 5 | 6 | 1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method | X | | | Error = | | Time ms |
| Gaussian Elimination | X1 | Can’t be solved | |  | |  |
| X2 |  | |
| X3 |  | |
| Pivoting | X1 | 0.0 | | 0% | | 0.029434 |
| X2 | -1.00000000000002 | | ≈ 0% | |
| X3 | 1.0 | | 0% | |
| LU decomposition | X1 | 0.0 | | 0% | | 0.878948 |
| X2 | -1.00000000000002 | | ≈ 0% | |
| X3 | 1.0 | | 0% | |
| Jacobi | X1 | ∞ | |  | | 0.028981 |
| X2 | ∞ | |  | |
| X3 | -∞ | |  | |
| after 3 iterations , X0 = [1 , 0 , 0]T , accuracy = 0.01 | | | | | |
| Gauss-seidel | X1 | ∞ | |  | | 0.017208 |
| X2 | ∞ | |  | |
| X3 | NaN | |  | |
| After 2 iterations , X0 = [1 , 0 , 0]T , accuracy = 0.01 | | | | | |
|  | | | | | | |
| LU inverse | -0.052365079 | | -0.1111111112 | | 0.1333333336 | |
| -0.1428571428 | | 0.0 | | 0.0 | |
| 0.02379365079 | | 0.11111111112 | | 0.0666666667 | |
| executed in 0.878948 ms | | | | | |

1. Ex2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | B |  | R |
| 3 | 7 | 13 | 76 | 1 |
| 1 | 5 | 3 | 28 | 3 |
| 12 | 3 | -5 | 1 | 4 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method | X | | | Error = | | Time ms |
| Gaussian Elimination | X1 | 1.0 | | 0% | | 0.017208 |
| X2 | 2.999999999999996 | | % | |
| X3 | 4.0 | | 0% | |
| Pivoting | X1 | 0.999999999999994 | | % | | 0.029434 |
| X2 | 3.000000000000013 | | % | |
| X3 | 3.9999999999999996 | | % | |
| LU decomposition | X1 | 1.0 | | 0% | | 0.187473 |
| X2 | 3.0 | | 0% | |
| X3 | 4.0 | | 0% | |
| Jacobi | X1 | 3.101564760026298 | | 210.156% | | 0.037132 |
| X2 | 6.977100591715977 | | 132.57% | |
| X3 | -4.588934911242604 | | 214.72% | |
| after 5 iterations , X0 = [1 , 0 , 0]T , accuracy = 0.3 | | | | | |
| Gauss-seidel | X1 | 2.6460442691211923 | | 164.6% | | 0.026717 |
| X2 | 7.82787201402586 | | 160.929% | |
| X3 | -4.748710531195739 | | 218.7% | |
| after 4 iterations , X0 = [1 , 0 , 0]T , accuracy = 0.3 | | | | | |
|  | | | | | | |
| LU inverse | 0.06115107913 | | -0.13309352517 | | 0.07913669064 | |
| -0.07374100719 | | 0.307553956834 | | -0.00719424460 | |
| 0.102517985611 | | -0.13489208633 | | -0.01438848920 | |
| executed in 0.187473 ms | | | | | |

1. Ex3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | B |  | R |
| 12 | 3 | -5 | 1 | 1 |
| 1 | 5 | 3 | 28 | 3 |
| 3 | 7 | 13 | 76 | 4 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method | X | | | Error = | | Time ms |
| Gaussian Elimination | X1 | 0.999999999999994 | | % | | 0.016754 |
| X2 | 3.000000000000013 | | % | |
| X3 | 3.999999999999999 | | % | |
| Pivoting | X1 | 0.999999999999994 | | % | | 0.016755 |
| X2 | 3.000000000000013 | | % | |
| X3 | 3.999999999999996 | | % | |
| LU decomposition | X1 | 1.0 | | 0% | | 0.16936 |
| X2 | 3.0 | | 0% | |
| X3 | 4.0 | | 0% | |
| Jacobi | X1 | 1.010834976988823 | | 1.083% | | 0.039397 |
| X2 | 2.8069822485207103 | | 6.433% | |
| X3 | 3.7172781065088754 | | 7.06% | |
| after 5 iterations , X0 = [1 , 0 , 0]T , accuracy = 0.3 | | | | | |
| Gauss-seidel | X1 | 0.9318101976530846 | | 6.81% | | 0.032604 |
| X2 | 3.0356901022789216 | | 1.189% | |
| X3 | 3.9965183608529458 | | 0.087% | |
| after 5 iterations , X0 = [1 , 0 , 0]T , accuracy = 0.3 | | | | | |
|  | | | | | | |
| LU inverse | 0.07913669064 | | -0.13309352517 | | 0.06115107913 | |
| -0.0071942446 | | 0.307553956834 | | -0.0737410071 | |
| -0.01438848920 | | -0.13489208633 | | 0.1025179856 | |
| executed in 0.16936 ms | | | | | |

1. Ex4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | B |  | R |
| 1 | 1 | 1 | 3 | 1 |
| 2 | 3 | 4 | 9 | 1 |
| 1 | 7 | 1 | 9 | 1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method | X | | | Error = | | Time ms |
| Gaussian Elimination | X1 | 1.0 | | 0% | | 0.016755 |
| X2 | 1.0 | | 0% | |
| X3 | 1.0 | | 0% | |
| Pivoting | X1 | 1.0 | | 0% | | 0.029434 |
| X2 | 1.0 | | 0% | |
| X3 | 1.0 | | 0% | |
| LU decomposition | X1 | 1.0 | | 0% | | 0.128152 |
| X2 | 1.0 | | 0% | |
| X3 | 1.0 | | 0% | |
| Jacobi | X1 | -∞ | |  | | 1.945823 |
| X2 | -∞ | |  | |
| X3 | -∞ | |  | |
| after 561 iterations , X0 = [1 , 0 , 0]T , accuracy = 0.01 | | | | | |
| Gauss-seidel | X1 | NaN | |  | | 0.704155 |
| X2 | NaN | |  | |
| X3 | NaN | |  | |
| after 391 iterations , X0 = [1 , 0 , 0]T , accuracy = 0.01 | | | | | |
|  | | | | | | |
| LU inverse | 2.08333333335 | | -0.5 | | -0.083333331 | |
| -0.1666666669 | | 0.0 | | 0.166666666 | |
| -0.9166666667 | | 0.5 | | -0.083333334 | |
| executed in 0.128152 ms | | | | | |

1. Ex5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | B |  | R |
| 5 | 2 | 3 | 3 | = -1.0167 |
| 2 | 5 | 3 | 10 | = 1.3167 |
| 2 | 3 | 5 | 11 | = 1.8167 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method | X | | | Error = | | Time ms |
| Gaussian Elimination | X1 | -1.016666666666667 | | 0% | | 0.010868 |
| X2 | 1.3166666666666669 | | 0% | |
| X3 | 1.8166666666666669 | | 0% | |
| Pivoting | X1 | -1.016666666666667 | | 0% | | 0.020378 |
| X2 | 1.3166666666666669 | | 0% | |
| X3 | 1.8166666666666669 | | 0% | |
| LU decomposition | X1 | -1.016666666666668 | | 0% | | 0.129963 |
| X2 | 1.316666666666669 | | 0% | |
| X3 | 1.8166666666666667 | | 0% | |
| Jacobi | X1 | Out of memory | |  | |  |
| X2 | Out of memory | |  | |
| X3 | Out of memory | |  | |
| after … iterations , X0 = [1 , 0 , 0]T , accuracy = 0.01 | | | | | |
| Gauss-seidel | X1 | -1.0183261229154303 | | 0.16% | | 0.009962 |
| X2 | 1.3292976988264855 | | 0.959% | |
| X3 | 1.8097518298702808 | | 0.38% | |
| after 7 iterations , X0 = [1 , 0 , 0]T , accuracy = 0.01 | | | | | |
|  | | | | | | |
| LU inverse | 0.26666666667 | | -0.016666666663 | | -0.15 | |
| -0.066666666667 | | 0.31666666665 | | -0.15 | |
| -0.066666666667 | | -0.18333333338 | | 0.350000003 | |
| executed in 0.129963 ms | | | | | |

1. Ex6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | B |  | R |
| 6 | 2 | 3 | 5 | = -0.8 |
| 2 | 5 | 3 | 10 | = 1.26 |
| 2 | 3 | 5 | 11 | = 1.76 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method | X | | | Error = | | Time ms |
| Gaussian Elimination | X1 | -0.8026315789473685 | | 0% | | 0.016755 |
| X2 | 1.2631578947368423 | | 0% | |
| X3 | 1.763157894736842 | | 0% | |
| Pivoting | X1 | -0.8026315789473685 | | 0% | | 0.020831 |
| X2 | 1.2631578947368423 | | 0% | |
| X3 | 1.763157894736842 | | 0% | |
| LU decomposition | X1 | -0.8026315789473685 | | 0% | | 0.169812 |
| X2 | 1.2631578947368423 | | 0% | |
| X3 | 1.763157894736842 | | 0% | |
| Jacobi | X1 | -0.6896163396177828 | | 14% | | 0.025812 |
| X2 | 1.3920821814269844 | | 10.2% | |
| X3 | 1.8920819971987175 | | 7.3% | |
| after 30 iterations , X0 = [1 , 0 , 0]T , accuracy = 0.2 | | | | | |
| Gauss-seidel | X1 | -0.802033777777778 | | 0.07% | | 0.004528 |
| X2 | 1.3402183111111111 | | 6.1% | |
| X3 | 1.7166825244444446 | | 2.6% | |
| after 5 iterations , X0 = [1 , 0 , 0]T , accuracy = 0.2 | | | | | |
|  | | | | | | |
| LU inverse | 0.21052631578 | | -0.0131578947 | | -0.1184210526 | |
| -0.0526315789 | | 0.31578947368 | | -0.1578947368 | |
| -0.0526315789 | | -0.1842105263 | | 0.34210526315 | |
| executed in 0.169812 ms | | | | | |

1. Conclusions from analysis
2. From ex1
   1. Partial pivoting avoids division by zero
3. From ex2 to ex6
   1. Partial pivoting decreases round off error
   2. Partial pivoting solution nearly equal to forward elimination solution
   3. Forward elimination executes faster than partial pivoting
4. From ex1 to ex6
   1. LU solves all the matrices
   2. All matrices as notice have AA-1 = 1 property and we can get A-1 by using LU decomposition as shown in the examples
   3. LU is slower than forward elimination and partial pivoting but more accurate than both
5. From ex1, ex2, ex4
   1. Gauss and Jacobi failed to converge because the matrices are not strictly diagonal matrix – SDD –
      * SDD for Matrix A means this :
6. From ex3
   1. The matrix is in SDD format so Gauss and Jacobi converge to true solution with reasonable error as
7. From ex5
   1. When the matrix is DD only
      * Jacobi overflows the memory and terminates with no solution
      * Gauss converge with reasonable error
      * Diagonal Dominant matrix DD means that
8. From ex6
   1. When the matrix SDD Gauss and Jacobi converges to true solution with reasonable error
9. So for Gauss and Jacobi when matrix SDD then Gauss and Jacobi tends to converge but if the matric DD then Gauss and Jacobi probably converge or not while if matrix not SDD then Gauss and Jacobi doesn’t converge at all
10. Each method analysis

No. of Equations Vs Time (ms)

**Gaussian Elimination**

**Jacobi and Gauss Seidel**

**LU Decomposition**

No. of Equations Vs Time (ms) and number of B’s is the same as the number of equations

**Comment:**

Gaussian Elimination takes more time and to solve higher order of equations takes much time in comparison to other methods. Other methods have a linear order and Gaussian Elimination has a quadratic and that’s the same as the result that was concluded in theoretical analysis.

1. Data structure used

1- Arrays

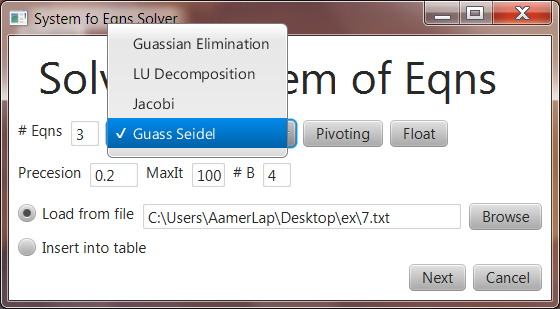
Array gives a better performance and time.

1. **ArrayList**

ArrayList gives a bit better performance and time and it’s extendable.

1. **LinkedList**

LinkedList gives less performance but it’s extendable and we used to save things and not to operate on its content.

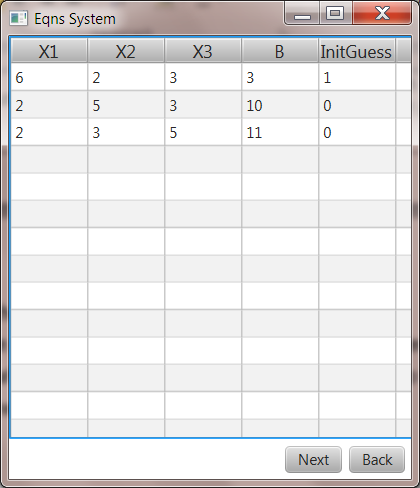
1. User Guide

How to run?!!

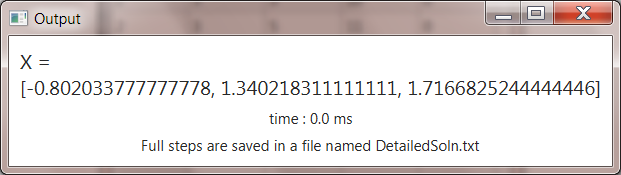
1. Set the # Eqns to the number of equations you have in the system
2. Choose the appropriate method to solve with
3. Set the rest of the fields according to this table

|  |  |
| --- | --- |
| Method | Appropriate fields to set |
| Gaussian Elimination | **Pivoting :** use pivoting or not  **Precession field** : ignore  **MaxIt field** : ignore  **#B** : ignore |
| LU Decomposition | **Pivoting** : ignore  **Precession field** : ignore  **MaxIt field** : ignore  **#B** : you can set number of solutions |
| Jacobi/Gauss-Seidel | **Pivoting** : ignore  **Precession field** : set precession  **MaxIt field :** set max number of iterations  **#B** : ignore |

1. Choose either to Load from file or Insert into table
2. Choose the precision **double is default** can be overridden with **float**
3. Click **next** after setting everything and it will take you to the tabulated data like the figure below



1. Click **next** again to see the solution in a window like the figure below



1. Keep in mind that for Jacobi or Gauss-Seidel there is output file to show the iterations performed