Numerical Analysis  
Assignment 1 - Part 2

14 May 2019

# Flowchart

## Gaussian-elimination

|  |  |
| --- | --- |
|  | Describe how gauss elimination method works: |

**Analysis and conclusion for the behavior of Gauss Elimination:**

Gauss Elimination finds the roots of linear equation by eliminating the lower elements in the coefficient matrix then applying backward substitution to get the roots.

Gauss Elimination always converges to the roots of the equation in O(n3).

## LU decomposition

|  |  |
| --- | --- |
|  | Describe how LU decomposition method works: |

**Analysis and conclusion for the behavior of LU Decomposition:**

LU Decomposition finds the roots by splitting the coefficient matrix into lower and upper matrices multiplied together, applying forward substitution on the lower matrix the applying backward substitution on the upper matrix.

LU Decomposition always converges to the roots of the equation in O(n3).

## Gaussian-Jordan

|  |  |
| --- | --- |
|  | Describe how Gaussian Jordan method works:  function [timeElapsed,system\_matrix,steps,lastMatrix,solution] = gauss\_jordan(coeff\_matrix, constants\_matrix, num\_of\_unknowns)  %create\_system\_matrix  for index=1 to length(constants\_matrix)  coeff\_matrix(index, length(constants\_matrix)+1) <-- constants\_matrix(index)  end  system\_matrix <-- coeff\_matrix  result <-- system\_matrix  %forward\_elimination  steps(1) <-- result  for pivot\_index=1 to num\_of\_unknowns  %normalize  pivot <-- result(pivot\_index, pivot\_index)  for col=1 to num\_of\_unknowns+1  result(pivot\_index, col) <-- result(pivot\_index, col)/pivot  end  steps(pivot\_index) <-- result  %apply elimination  for row=1 to num\_of\_unknowns  if row equals pivot\_index  continue  end  row\_pivot <-- result(row,pivot\_index)  for col=pivot\_index to num\_of\_unknowns+1  result(row,col) <-- result(row,col)-row\_pivot\*result(pivot\_index,col)  end  end  end  lastMatrix <-- result  %back\_substitution  solution <-- [0,0]  for i=1 to num\_of\_unknowns  solution(i) <-- lastMatrix(i, num\_of\_unknowns+1)  end |

**Analysis and conclusion for the behavior of Gauss Jordan:**

 Its two main purposes are to solve [system of linear equations](https://brilliant.org/wiki/system-of-linear-equations/) and calculate the [inverse of a matrix](https://brilliant.org/wiki/matrix-inverse/).

**Similar to the Gauss elimination except**

**1. Elimination is applied to all equations (excluding the**

**pivot equation) instead of just the subsequent equations.**

**2. All rows are normalized by dividing them by their pivot**

**elements.**

**3. No back substitution is required.**

## Gauss-Seidel

|  |  |
| --- | --- |
|  | Describe how Gauss Seidel method works:  function [numOfIterations,final,errorArray,answers] = GaussSeidel(numberOfFunctions,coeff\_matrix,constants\_matrix,initialGuess,maxIterations,epsilon)  answers = double.empty  previous = double.empty  for i = 1 to numberOfFunctions  answers <-- [answers;initialGuess(i)]  end  iteration = 1 , error = 100  errorArray = double.empty  final = double.empty  while (iteration less than or equal maxIterations & error greater than epsilon)  for i = 1 to length(answers)  j = 1  previous(i) <-- answers(i)  answers(i) <-- constants\_matrix(i)  for k = 1 to length(answers) - 1  if (j equal i)  j <-- j + 1  end  answers(i) <-- answers(i) - coeff\_matrix(i,j)\*answers(j)  j = j + 1  end  answers(i) <-- answers(i) / coeff\_matrix(i,i)  error <-- abs(((answers(i) - previous(i)) / answers(i)) \* 100)  errorArray(i , iteration) <-- error  final(i,iteration) <-- answers(i)  end  iteration <-- iteration + 1  end  numOfIterations <-- iteration - 1 |

**Analysis and conclusion for the behavior of Gauss Seidal:**

It is an [iterative method](https://en.wikipedia.org/wiki/Iterative_method) used to solve a [linear system of equations](https://en.wikipedia.org/wiki/Linear_system_of_equations) of n linear equations with unknown x: A x = B

Though it can be applied to any matrix with non-zero elements on the diagonals, convergence is only guaranteed if the matrix is either [diagonally dominant](https://en.wikipedia.org/wiki/Diagonally_dominant_matrix), or [symmetric](https://en.wikipedia.org/wiki/Symmetric_matrix) and [positive definite](https://en.wikipedia.org/wiki/Positive-definite_matrix).

# Problamatic Functions

In Gauss Seidel method :

One class of system of equations always converges: One with a

diagonally dominant coefficient matrix, with non-zero elements on the diagonals.

If a system of linear equations is not diagonally dominant, check to see if

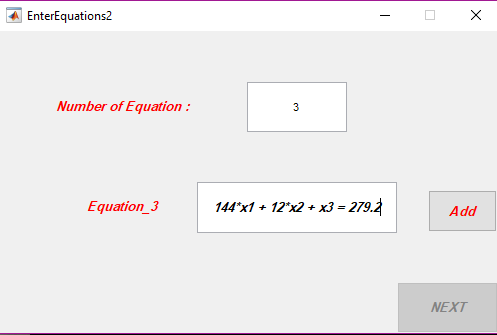
rearranging the equations can form a diagonally dominant matrix.

Not every system of equations can be rearranged to have a

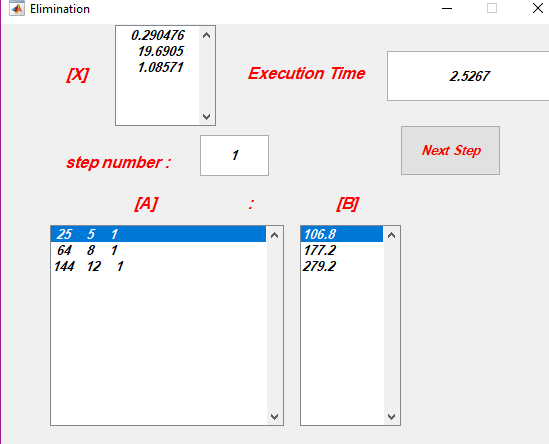
diagonally dominant coefficient matrix.

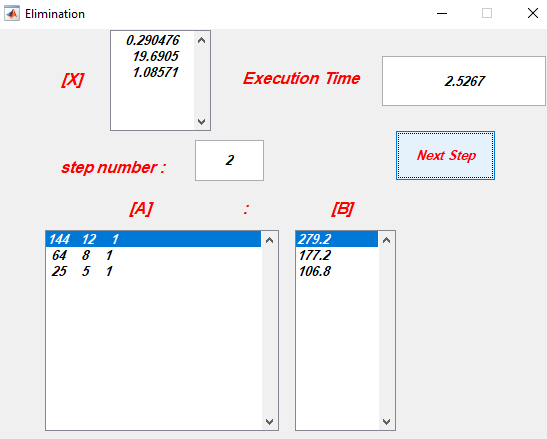
# Sample runs and snapshots

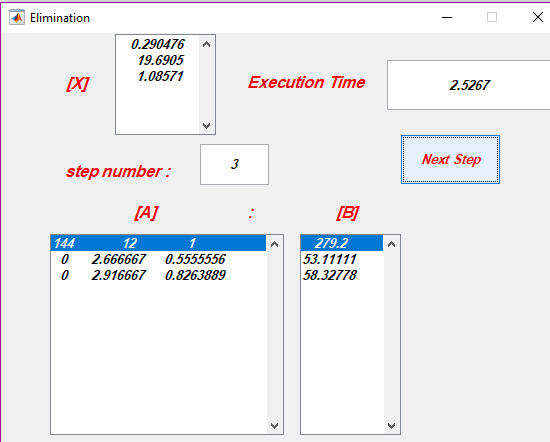
## C:\Users\Blu-Ray\Desktop\60295840_2425374584214984_6649247464614264832_n.pngC:\Users\Blu-Ray\Desktop\60149080_2179888872079912_8487259997175021568_n.png

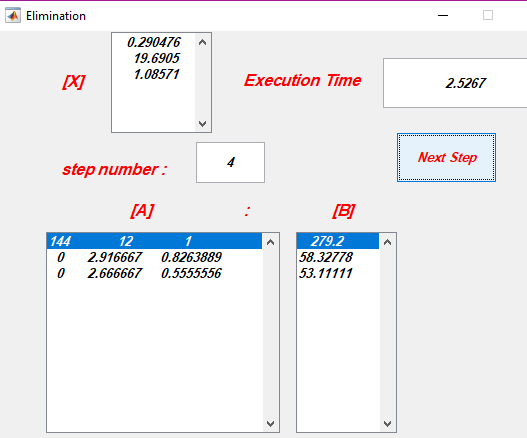


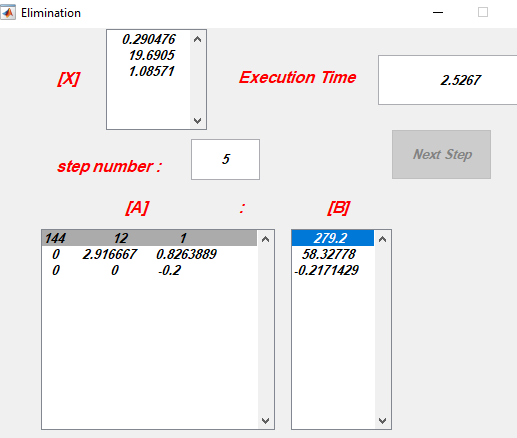
Gaussian-elimination



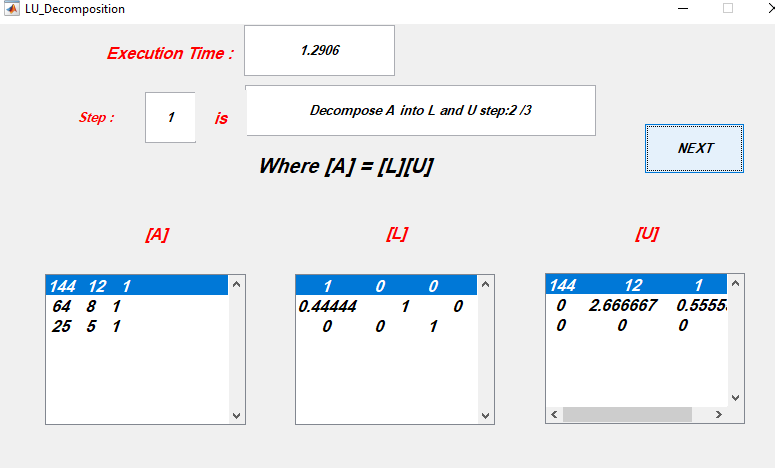
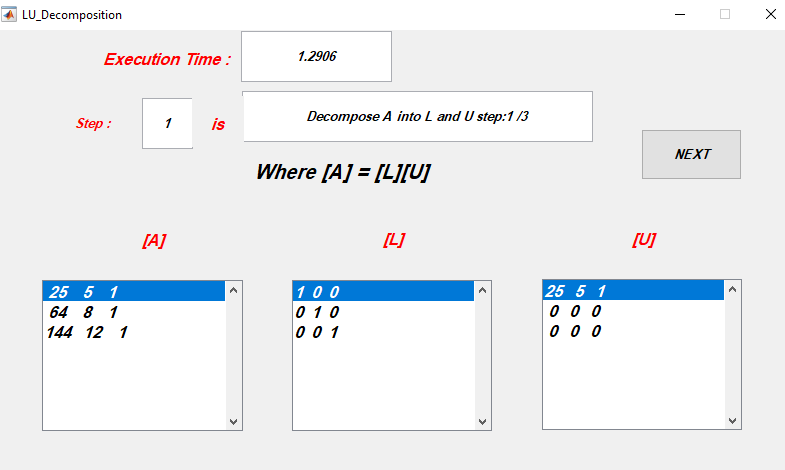


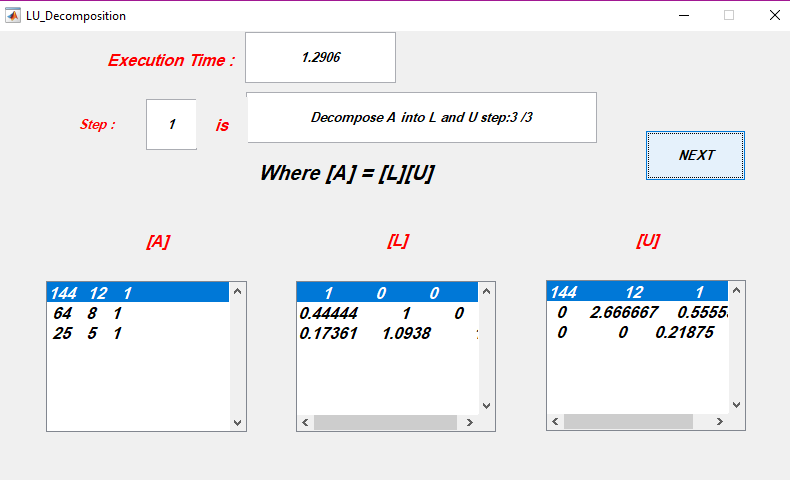


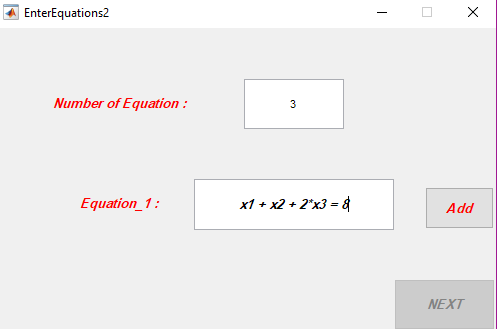


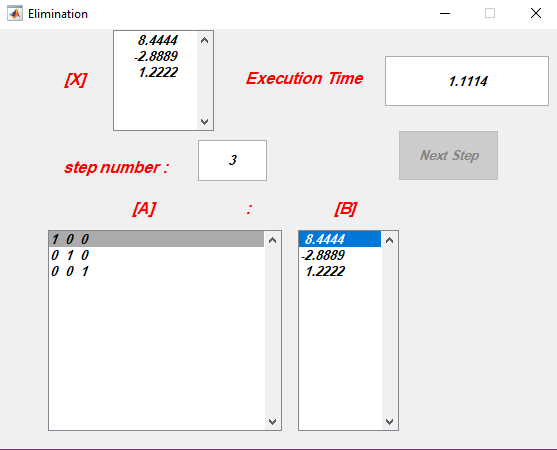
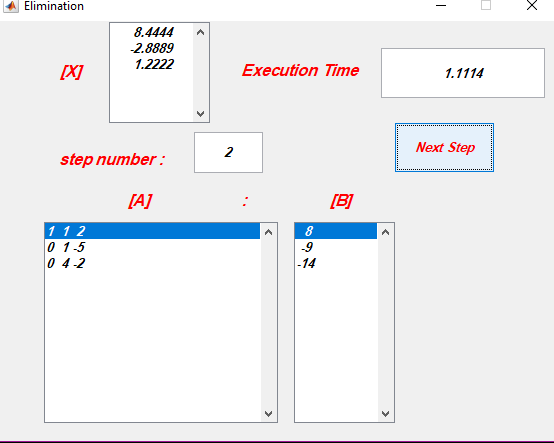
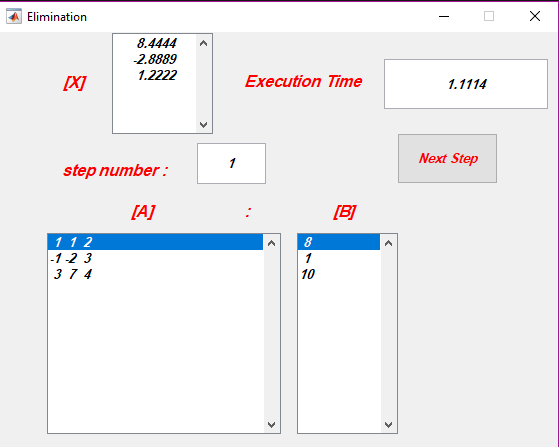
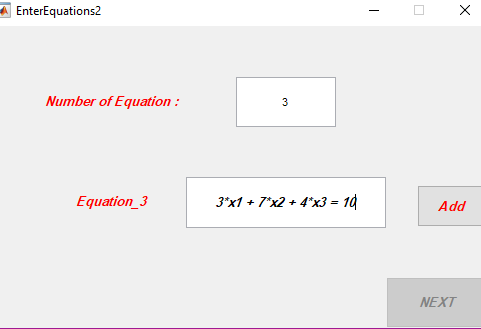
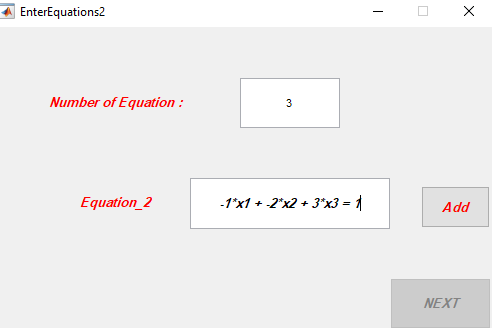


LU Decomposition



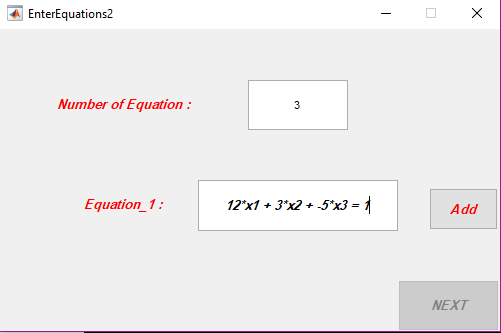


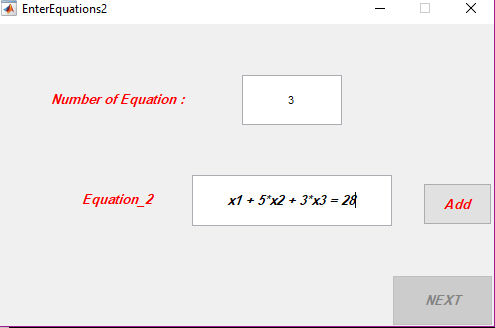
Gauss Jordan

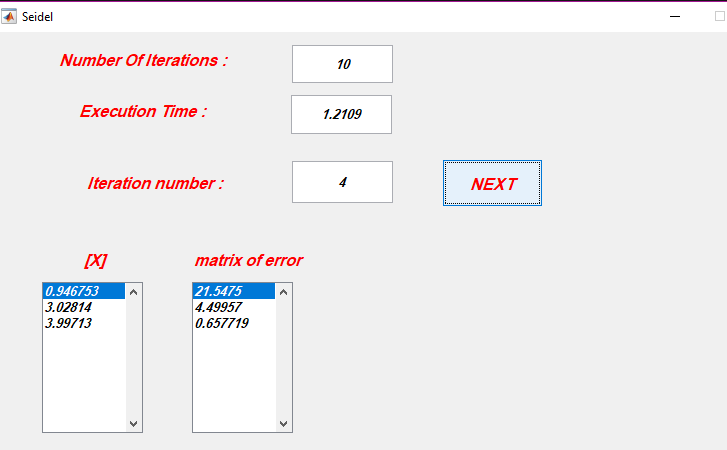
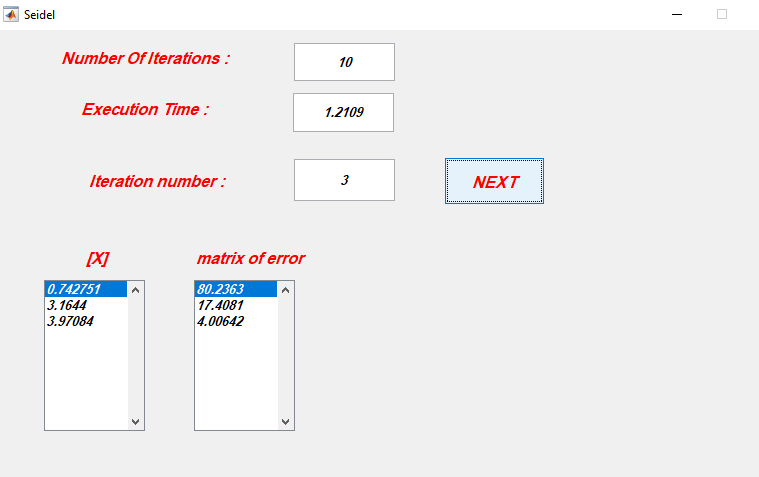
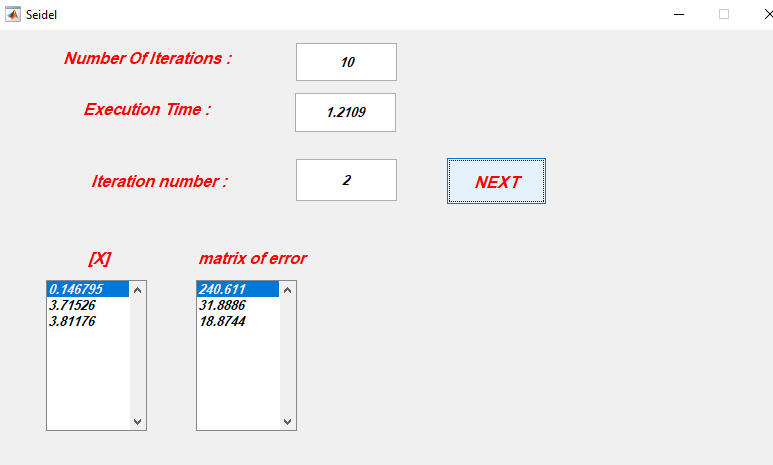
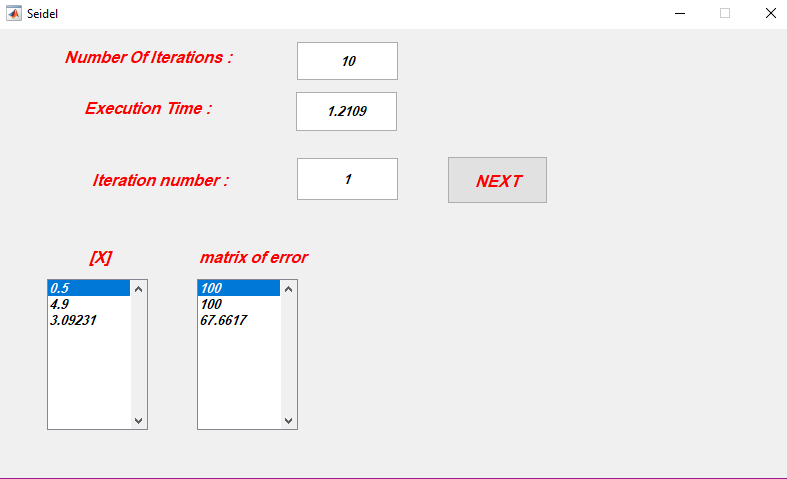
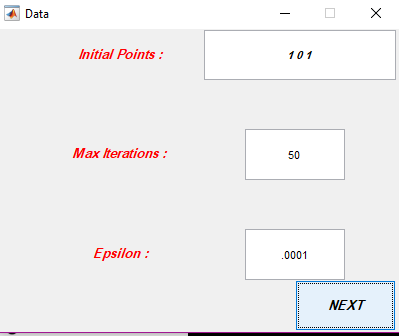
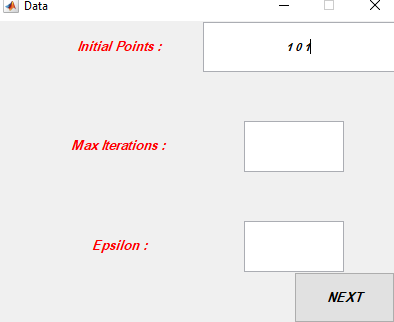
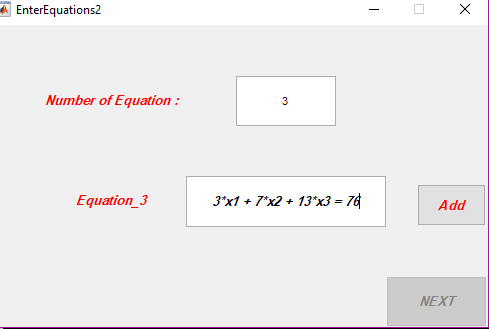


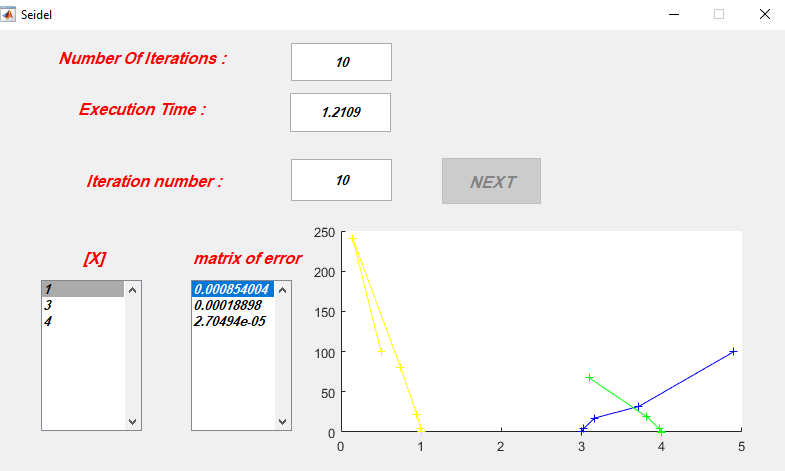
Gauss Seidel Method

Which converges :



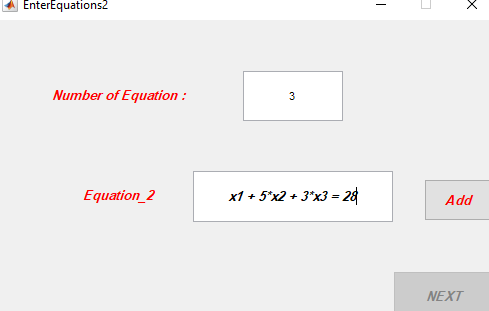
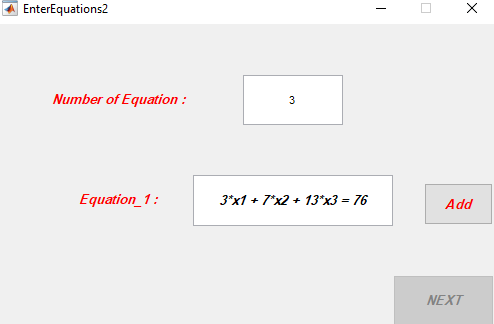


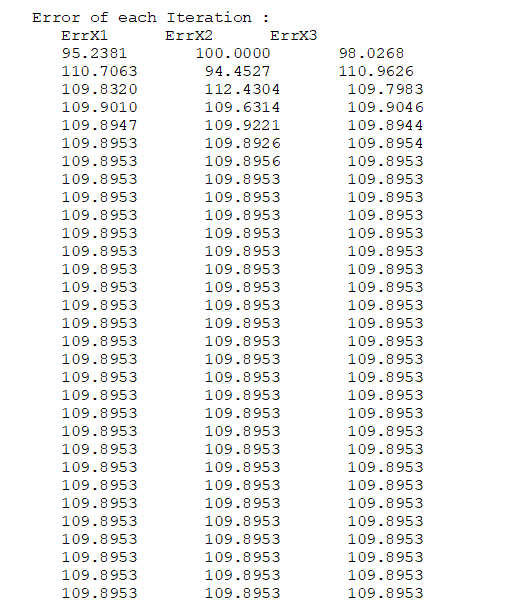
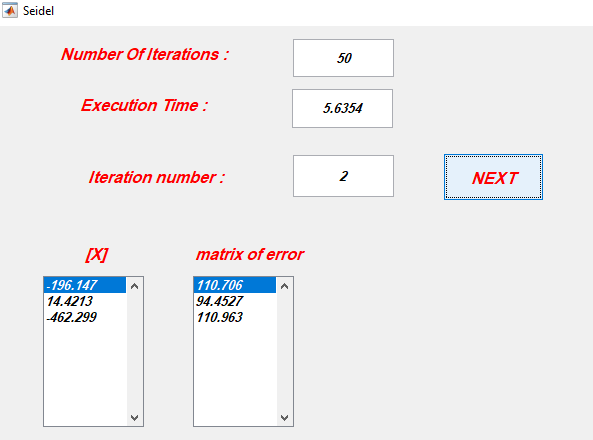
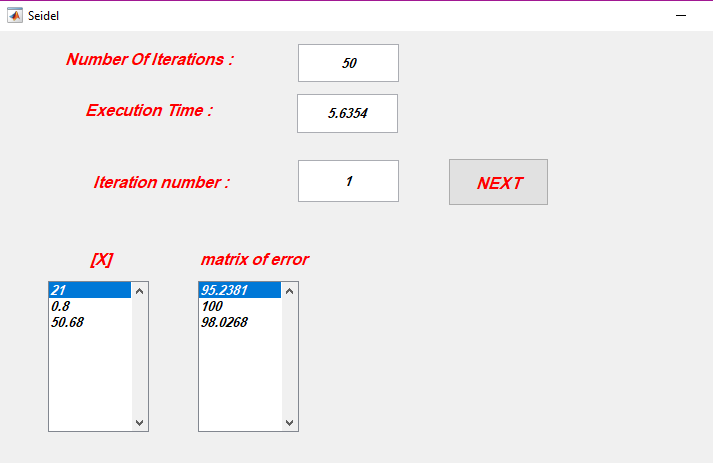
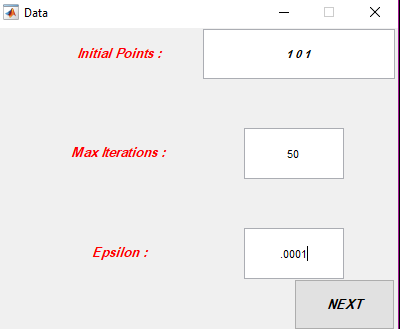
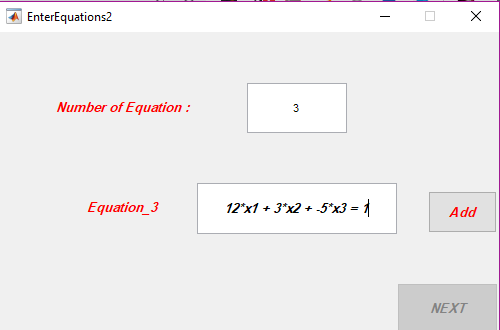




Gauss Seidel Method

Which diverges :





# Team

|  |  |
| --- | --- |
| Name | ID |
| ایمان رفیق عبد القادر محمد على | 11 |
| تقى علاء احمد الجندى | 14 |
| ميرنا محمد مصطفى اسماعيل مصطفى | 53 |
| ندى سلامھ محمد على | 55 |
| یمنى جمال الدین محمود السید | 60 |