UNIVERSITY OF BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

FET 651 - NUMERICAL METHODS FOR ENGINEERING

Chapter 3 – Assignment
Solve a system of linear equations with GaussSeidel method.

Group Members

ABANDA PACILIA COLETTE	FE20P070	NETWORKING
LENYA HOPE NEMBI	FE19P030	SOFTWARE
FEUTIO WOUTSOP CLIDE STELL	FE19P035	SOFTWARE
EVINA NGALEBA CEDRIC GEDEON	FE20P105	SOFTWARE

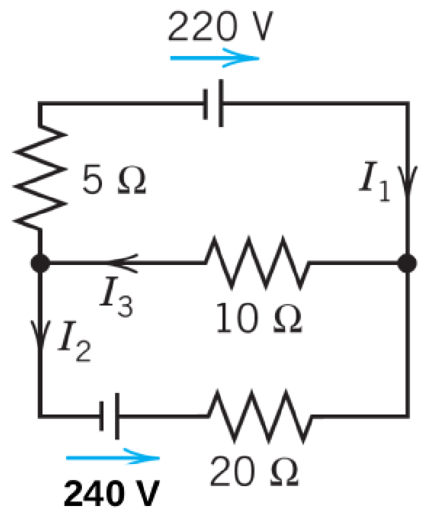
05th February, 2021

Problem Statement

CH.3- System of linear equations

Goal:

Solve a system of linear equations with Gauss-Seidel method.



Problem:

Consider the following circuit.

1. Use Kirchhoff laws of current and voltage, obtain the system of equations that satisfy the currents.
2. Write the matrix equation with nonzeros pivots.
3. Using Matlab or Python, write a code that solves the problem using

(a) The built-in function *linsolve.*

(b) The Gauss-Seidel algorithm built from the lecture notes. Starting from a guess (0,0,0)

- Plot the evolution of relative error on i_2 taking the result in (a) as exact.
- Display the currents after N=20 iterations.

Solution

From the circuit, we can derive the following equations;

$4I_1 - I_2 - 3I_3 = -4$

$-2I_1 + 0I_2 + 3I_3 = 24$

$0I_1 - I_2 + 3I_3 = 44$

From the above equations, we have the following;

$$\begin{pmatrix} 4 & -1 & -3 \\ -2 & 0 & 3 \\ 0 & -1 & 3 \end{pmatrix} \begin{pmatrix} I1 \\ I2 \\ I3 \end{pmatrix} = \begin{pmatrix} -4 \\ 24 \\ 44 \end{pmatrix}$$

$$\begin{pmatrix} 4 & -1 & -3 & -4 \\ -2 & 0 & 3 & 24 \\ 0 & -1 & 3 & 44 \end{pmatrix}$$

$$\begin{pmatrix} -2 & 0 & 3 & 24 \\ 4 & -1 & -3 & -4 \\ 0 & -1 & 3 & 44 \end{pmatrix}$$

$$\begin{pmatrix} -2 & 0 & 3 & 24 \\ 0 & -1 & 3 & 44 \\ 0 & -1 & 3 & 44 \end{pmatrix}$$

Python Code

a. Linsolve Solution

```
from sympy import *
i1, i2, i3 = symbols("i1, i2, i3")
A = Matrix([[4, -1, -3],[-2, 0, 3],[0,-1,3]])
b = Matrix([-4,24,44])
sol= linsolve((A,b),[i1,i2,i3])
i = Array(sol.subs({i3:0}),1)
print(i[0])
```

Result

```
pacilia@pacilia-X751LX:~/Desktop$ python3 ex1.py
FiniteSet((3*i3/2 - 12, 3*i3 - 44, i3))
pacilia@pacilia-X751LX:~/Desktop$
```

NB: From the results, we see that we have a free variable $\mathbf{i_3}$**,** so our initial value of $\mathbf{i_3}$ was 0 but after observation from the results obtained from the Gauss-Seidel method below, from N=15, we observed that the graph was a straight line implying that we have converged to a solution. From the initial assumption of $\mathbf{i_3}$, we obtained $\mathbf{i_1}$ and $\mathbf{i_2}$ and used them for our exact solution to calculate the relative error of $\mathbf{i_2}$. Since the graph shows convergence, we took the value of i_3 from the Gauss-Seidel to calculate the exact value of i_1 and i_2 since they are in terms of i_3, and we saw that i_1 and i_2 gotten were the same as the values gotten from the Gauss-Seidel proving our guess that the algorithm already converged. The value of i_3 gotten was **44/9**

b. Gauss-Seidel Solution

```
from sympy import *
import matplotlib.pylab as plt
import numpy as np
ITERATION_LIMIT = 20

i1, i2, i3 = symbols("i1, i2, i3")
A = Matrix([[4, -1, -3],[0,-1,3],[-2, 0, 3]])
b = Matrix([-4,44,24])
sol= linsolve((A,b),[i1,i2,i3])

# exact solution
print("Solution from linsolve: " + str(sol))
I_exact = Array(sol.subs({i3:44/9}),3)

A = np.array(A).astype(np.float64)
b = np.array(b).astype(np.float64)
print(A)
print(b)
print(I_exact)
I_ini = np.zeros_like(b)
I2 = np.zeros(ITERATION_LIMIT)
for i in range(0, ITERATION_LIMIT):
    I_new = np.zeros_like(I_ini)
    print("Iteration {0}: {1} \n\n".format(i, I_ini))
    for j in range(A.shape[0]):
        print("for i =",j)
        print(A[j, :j])
        print(I_new[:j])
        s1 = np.dot(A[j, :j], I_new[:j])
        s2 = np.dot(A[j, j + 1:], I_ini[j + 1:])
        I_new[j] = (b[j] - s1 - s2) / A[j, j]
    I2[i] = I_new[1]
    I_ini = I_new

print("Solution N = {1}: {0}".format(I_ini, ITERATION_LIMIT))
rel_error_i2 = lambda x:np.absolute((x - I_exact[1])/I_exact[1])

X = np.arange(ITERATION_LIMIT)
re_i2 = rel_error_i2(I2)

plt.plot(X, re_i2,label = "re")
plt.xlabel('N (Iteration)')
plt.ylabel('RE_I2 (Relative error I2)')
plt.legend()
plt.show()
```

Results

Solution at the 20th iteration:

Solution N = 20: [[-4.66667366]

[-29.33330536]

[4.88888423]]

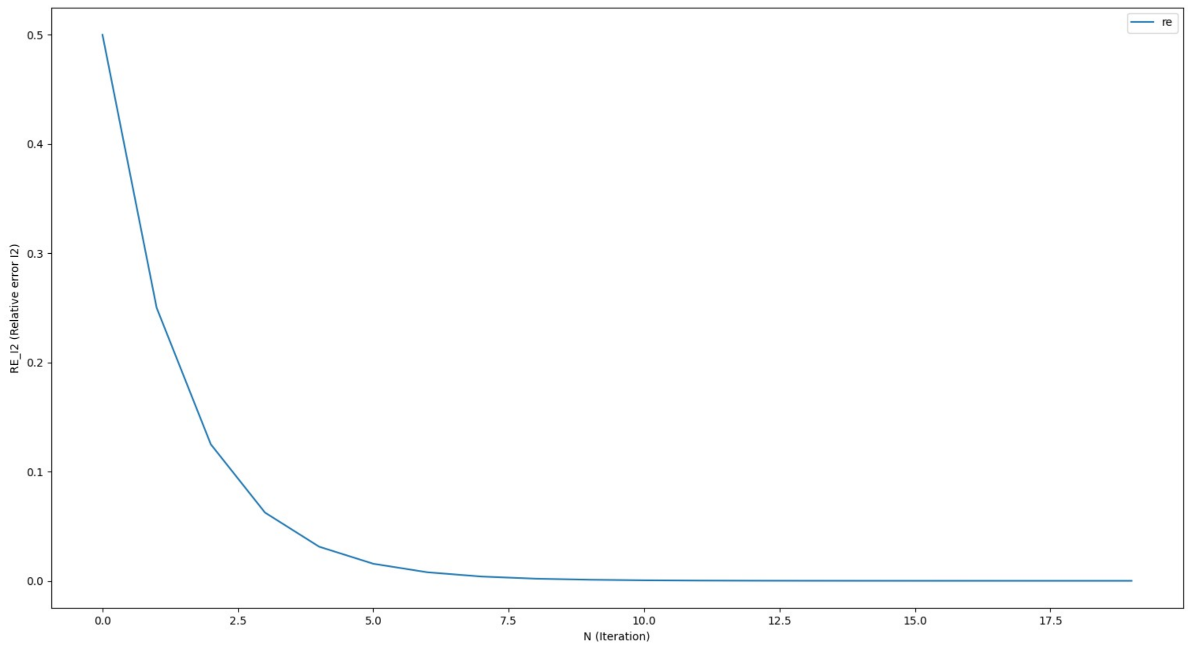


Figure 1: Relative error of i_2