LAB - 5

Comparison of Jacobi's and Gauss-Seidel Methods for solving a system of Linear Equation

9x + y + z = 10

2x + 10y + 3z =19

3x + 4y + 11z =0

In [3]:
```
import numpy as np
import copy
import pandas as pd

#def vector_norm(v):
#    return np.linalg.norm(v, ord=np.inf)


# Jacobi
def jacobi(A, b, x0, tol=1e-5, max_iter=100):
    n = len(A)
    x = x0.copy()
    results = []
    for iter_num in range(max_iter):
        x_new = np.zeros(n)
        for i in range(n):
            s = sum(A[i, j] * x[j] for j in range(n) if j != i)
            x_new[i] = (b[i] - s) / A[i, i]
        results.append((iter_num + 1, *x_new))
        max2 = 0
        for i in range(n):
          if abs(x_new[i] - x[i]) > max2:
            max2 = abs(x_new[i] - x[i])
        if max2 < tol:
          return x_new, results
        x = x_new
    return x, results


# Gauss - Seidel
def gauss_seidel(A, b, x0, tol=1e-5, max_iter=40):
  results=[]
  n = len(A)
  x = []
  for i in range(n):
    x.append(x0[i])

  for k in range(max_iter):
    x_new = []
    for i in range(n):
      x_new.append(x[i])

    for i in range(n):
      sum = 0
      for j in range(n):
        if j<i:
          sum += A[i][j]*x_new[j]
        elif j>i:
          sum += A[i][j]*x[j]
```

```
      # Calculating L(inf) norm for the error
      x_new[i] = (b[i]-sum)/A[i][i]
      max2 = 0
      for i in range(n):
        if abs(x_new[i] - x[i]) > max2:
          max2 = abs(x_new[i] - x[i])
      if max2 < tol:
        return x_new, results
    results.append((k+1,*x_new))
    x = []
    for i in range(n):
      x.append(x_new[i])
  return x,results



A = np.array([[9, 1, 1], [2, 10, 3], [3, 4, 11]])
b = np.array([10, 19, 0])

# Initial guess values
x0 = np.array([0, 0, 0])
x1 = np.array([40, 10, 20])
x2 = np.array([-1, -1, -1])

jacobi_results_0= jacobi(A, b, x0);
jacobi_results_1= jacobi(A, b, x1);
jacobi_results_2= jacobi(A, b, x2);


gauss_seidel_results_0 = gauss_seidel(A, b, x0)
gauss_seidel_results_1 = gauss_seidel(A, b, x1)
gauss_seidel_results_2 = gauss_seidel(A, b, x2)


print("Jacobi (initial guess x=0, y=0, z=0)")
jacobi_table_0 = pd.DataFrame(jacobi_results_0[1], columns=['Iteration', 'x', 'y', 'z'])
print(jacobi_table_0)

print("\nJacobi (initial guess x=40, y=10, z=20)")
jacobi_table_1 = pd.DataFrame(jacobi_results_1[1], columns=['Iteration', 'x', 'y', 'z'])
print(jacobi_table_1)

print("\nJacobi (initial guess x=-1, y=-1, z=-1)")
jacobi_table_2 = pd.DataFrame(jacobi_results_2[1], columns=['Iteration', 'x', 'y', 'z'])
print(jacobi_table_2)

print("\nGauss-Seidel (initial guess x=0, y=0, z=0)")
gauss_seidel_table_0 = pd.DataFrame(gauss_seidel_results_0[1], columns=['Iteration', 'x'
print(gauss_seidel_table_0)

print("\nGauss-Seidel (initial guess x=40, y=10, z=20)")
gauss_seidel_table_1 = pd.DataFrame(gauss_seidel_results_1[1], columns=['Iteration', 'x'
print(gauss_seidel_table_1)

print("\nGauss-Seidel (initial guess x=-1, y=-1, z=-1)")
gauss_seidel_table_2 = pd.DataFrame(gauss_seidel_results_2[1], columns=['Iteration', 'x'
print(gauss_seidel_table_2)

# Convergence Table
iterations_table = []
for result in [jacobi_results_0, jacobi_results_1, jacobi_results_2, gauss_seidel_result
    iterations_table.append([result[1][-1][0]])

iterations_table = pd.DataFrame(iterations_table, columns=['Iterations'],
                                index=['Jacobi (x=0, y=0, z=0)', 'Jacobi (x=40, y=10, z=
                                       'Jacobi (x=-1, y=-1, z=-1)', 'Gauss-Seidel (x=0,
```

```
                                    'Gauss-Seidel (x=40, y=10, z=20)', 'Gauss-Seidel

print("\nNumber of Iterations:")
print(iterations_table)
```

```
Jacobi (initial guess x=0, y=0, z=0)
    Iteration         x         y         z
0           1  1.111111  1.900000  0.000000
1           2  0.900000  1.677778 -0.993939
2           3  1.035129  2.018182 -0.855556
3           4  0.981930  1.949641 -1.016192
4           5  1.007395  2.008472 -0.976760
5           6  0.996476  1.991549 -1.005097
6           7  1.001505  2.002234 -0.995966
7           8  0.999304  1.998489 -1.001223
8           9  1.000304  2.000506 -0.999260
9          10  0.999862  1.999717 -1.000267
10         11  1.000061  2.000108 -0.999859
11         12  0.999972  1.999946 -1.000056
12         13  1.000012  2.000022 -0.999973
13         14  0.999994  1.999989 -1.000011
14         15  1.000002  2.000005 -0.999995
15         16  0.999999  1.999998 -1.000002

Jacobi (initial guess x=40, y=10, z=20)
    Iteration         x          y          z
0           1 -2.222222 -12.100000 -14.545455
1           2  4.071717   6.708081   5.006061
2           3 -0.190460  -0.416162  -3.549770
3           4  1.551770   3.003023   0.203275
4           5  0.754856   1.528663  -1.515218
5           6  1.109617   2.203594  -0.761747
6           7  0.950906   1.906601  -1.103930
7           8  1.021925   2.040998  -0.952647
8           9  0.990183   1.981409  -1.020888
9          10  1.004387   2.008230  -0.990562
10         11  0.998037   1.996291  -1.004189
11         12  1.000878   2.001649  -0.998116
12         13  0.999607   1.999259  -1.000839
13         14  1.000176   2.000330  -0.999624
14         15  0.999921   1.999852  -1.000168
15         16  1.000035   2.000066  -0.999925
16         17  0.999984   1.999970  -1.000034
17         18  1.000007   2.000013  -0.999985
18         19  0.999997   1.999994  -1.000007
19         20  1.000001   2.000003  -0.999997

Jacobi (initial guess x=-1, y=-1, z=-1)
    Iteration         x         y         z
0           1  1.333333  2.400000  0.636364
1           2  0.773737  1.442424 -1.236364
2           3  1.088215  2.116162 -0.735537
3           4  0.957708  1.903018 -1.066299
4           5  1.018142  2.028348 -0.953200
5           6  0.991650  1.982331 -1.015256
6           7  1.003658  2.006247 -0.991298
7           8  0.998339  1.996658 -1.003269
8           9  1.000735  2.001313 -0.998332
9          10  0.999669  1.999353 -1.000678
10         11  1.000147  2.000270 -0.999674
11         12  0.999934  1.999873 -1.000138
12         13  1.000029  2.000055 -0.999936
13         14  0.999987  1.999975 -1.000028
14         15  1.000006  2.000011 -0.999987
15         16  0.999997  1.999995 -1.000006
16         17  1.000001  2.000002 -0.999997
```

Gauss-Seidel (initial guess x=0, y=0, z=0)

	Iteration	x	y	z
0	1	1.111111	1.677778	-0.913131
1	2	1.026150	1.968709	-0.995753
2	3	1.003005	1.998125	-1.000138
3	4	1.000224	1.999997	-1.000060
4	5	1.000007	2.000017	-1.000008

Gauss-Seidel (initial guess x=40, y=10, z=20)

	Iteration	x	y	z
0	1	-2.222222	-3.655556	1.935354
1	2	1.302245	1.058945	-0.740229
2	3	1.075698	1.906929	-0.986801
3	4	1.008875	1.994265	-1.000335
4	5	1.000674	1.999966	-1.000171
5	6	1.000023	2.000047	-1.000023
6	7	0.999997	2.000008	-1.000002

Gauss-Seidel (initial guess x=-1, y=-1, z=-1)

	Iteration	x	y	z
0	1	1.333333	1.933333	-1.066667
1	2	1.014815	2.017037	-1.010236
2	3	0.999244	2.003222	-1.000965
3	4	0.999749	2.000340	-1.000055
4	5	0.999968	2.000023	-1.000000
5	6	0.999997	2.000000	-0.999999

Number of Iterations:

	Iterations
Jacobi (x=0, y=0, z=0)	16
Jacobi (x=40, y=10, z=20)	20
Jacobi (x=-1, y=-1, z=-1)	17
Gauss-Seidel (x=0, y=0, z=0)	5
Gauss-Seidel (x=40, y=10, z=20)	7
Gauss-Seidel (x=-1, y=-1, z=-1)	6