Mathematics for Machine Learning Assignment

Abhin P T

Statement

Comparison of the run time complexity of solving techniques:
- Gauss Jordan
- Gauss Seidel
- Cramers Rule

1 Program Code

1.1 Execution using Cramer's Rule

```
import time

def determinantOfMatrix(mat):

        ans = (mat[0][0] * (mat[1][1] * mat[2][2] -
                            mat[2][1] * mat[1][2]) -
            mat[0][1] * (mat[1][0] * mat[2][2] -
                            mat[1][2] * mat[2][0]) +
            mat[0][2] * (mat[1][0] * mat[2][1] -
                            mat[1][1] * mat[2][0]))
        return ans

def findSolution(coeff):

        d = [[coeff[0][0], coeff[0][1], coeff[0][2]],
            [coeff[1][0], coeff[1][1], coeff[1][2]],
            [coeff[2][0], coeff[2][1], coeff[2][2]]]


        d1 = [[coeff[0][3], coeff[0][1], coeff[0][2]],
            [coeff[1][3], coeff[1][1], coeff[1][2]],
            [coeff[2][3], coeff[2][1], coeff[2][2]]]

        d2 = [[coeff[0][0], coeff[0][3], coeff[0][2]],
            [coeff[1][0], coeff[1][3], coeff[1][2]],
            [coeff[2][0], coeff[2][3], coeff[2][2]]]


        d3 = [[coeff[0][0], coeff[0][1], coeff[0][3]],
            [coeff[1][0], coeff[1][1], coeff[1][3]],
            [coeff[2][0], coeff[2][1], coeff[2][3]]]


        D = determinantOfMatrix(d)
        D1 = determinantOfMatrix(d1)
        D2 = determinantOfMatrix(d2)
        D3 = determinantOfMatrix(d3)

        print("D is : ", D)
```

```
        print("D1 is : ", D1)
        print("D2 is : ", D2)
        print("D3 is : ", D3)


        if (D != 0):

            x = D1 / D
            y = D2 / D

            z = D3 / D

            print("Value of x is : ", x)
            print("Value of y is : ", y)
            print("Value of z is : ", z)

        else:
            if (D1 == 0 and D2 == 0 and
                D3 == 0):
                print("Infinite solutions")
            elif (D1 != 0 or D2 != 0 or
                D3 != 0):
                print("No solutions")
start = time.time()

if __name__ == "__main__":

        coeff = [[4, 1, 2, 4],[3, 5, 1, 7],[1, 1, 3, 3]]

        findSolution(coeff)
end = time.time()
print("\nTime of execution using cramer's rule : ",(end - start)*10**3, "ms")
```

1.2 Gauss-Jordan method

```
import time

M = 10

def PrintMatrix(a, n):
        for i in range(n):
            print(*a[i])

def PerformOperation(a, n):
        i = 0
        j = 0
        k = 0
        c = 0
        flag = 0
        m = 0
        pro = 0

        for i in range(n):
            if (a[i][i] == 0):

                c = 1
                while ((i + c) < n and a[i + c][i] == 0):
```

```
                    c += 1
                if ((i + c) == n):

                    flag = 1
                    break

                j = i
                for k in range(1 + n):

                    temp = a[j][k]
                    a[j][k] = a[j+c][k]
                    a[j+c][k] = temp

            for j in range(n):

                if (i != j):
                    p = a[j][i] / a[i][i]

                    k = 0
                    for k in range(n + 1):
                        a[j][k] = a[j][k] - (a[i][k]) * p

        return flag

def PrintResult(a, n, flag):

        print("Result is : ")

        if (flag == 2):
            print("Infinite Solutions Exists<br>")
        elif (flag == 3):
            print("No Solution Exists<br>")

        else:
            for i in range(n):
                print(a[i][n] / a[i][i], end=" ")

def CheckConsistency(a, n, flag):

        flag = 3
        for i in range(n):
            sum = 0
            for j in range(n):
                sum = sum + a[i][j]
            if (sum == a[i][j]):
                flag = 2

        return flag
start = time.time()

a = [[4, 1, 2, 4],[3, 5, 1, 7],[1, 1, 3, 3]]

n = 3
flag = 0
start = time.time()

flag = PerformOperation(a, n)

if (flag == 1):
        flag = CheckConsistency(a, n, flag)
```

```
#print("Final Augmented Matrix is : ")
PrintMatrix(a, n)
print()

PrintResult(a, n, flag)
end = time.time()
print("\nTime of execution of Gauss-Jordan method : ",(end - start)*10**3, "ms")
```

1.3 Gauss–Seidel method

```
import time
def seidel(a, x ,b):
        n = len(a)
        for j in range(0, n):
            d = b[j]

            for i in range(0, n):
                if(j != i):
                    d-=a[j][i] * x[i]
            x[j] = d / a[j][j]
        return x
start = time.time()
n = 3
a = []
b = []
x = [0, 0, 0]
a = [[4, 1, 2],[3, 5, 1],[1, 1, 3]]
b = [4,7,3]

for i in range(0, 25):
        x = seidel(a, x, b)
print(x)
end = time.time()
print("\nTime of execution of Gauss{Seidel method : ",(end - start)*10**3, "ms")
```

2 Program Output

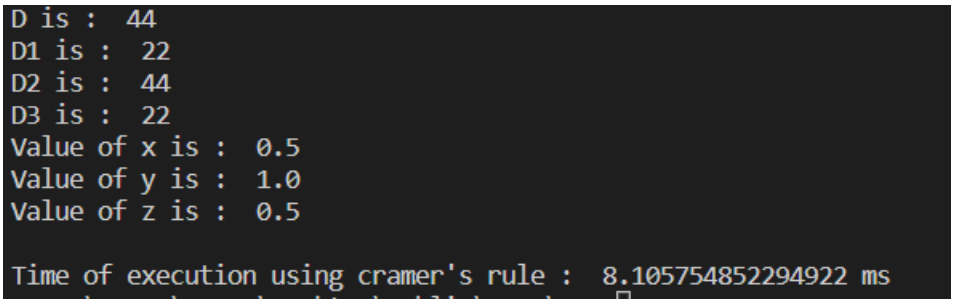

Figure 1: Cramer's rule

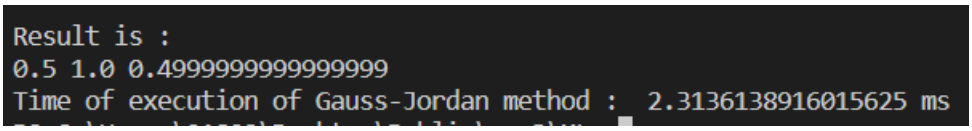

Figure 2: Gauss-Jordan

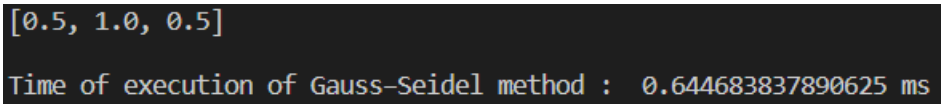

Figure 3: Gauss-Seidel