Coding Assignment-3 (Outputs)

Computational Methods & Applications

Name :- Zahir Khan. Roll no:- 112202010 -MTech CaM.

………

Q.1) Output:

1.1)

```
r= RowVectorFloat([1,2,3])

print(r)
```

Result: 1 2 3

1.2)

```
r= RowVectorFloat([1,2,3])
print(len(r))
```

Result: 3

1.3)

```
r= RowVectorFloat([])
print(len(r))
```

Result: 0

1.4)

```
r= RowVectorFloat([1,2,4])
print(r[1])
```

Result: 2

1.5)

```
r= RowVectorFloat([1,2,4])
r[2]=5
print(r)
```

Result: 1 2 5

```
r1 = RowVectorFloat([1, 2 , 4])
r2 = RowVectorFloat([1, 1 , 1])
r3 = 2*r1 + (-3)*r2
print(r3)
```

Result: -1 1 5

………

Q.2) Output:

2.1)

```
s = SquareMatrixFloat(3)
print(s)
```

Result:

```
The matrix is
0 0 0
0 0 0
0 0 0
```

2.2)

```
s = SquareMatrixFloat(4)
s.sampleSymmetric()
print(s)
```

Result:

```
The matrix is
0.1 0.75 0.05 0.21
0.75 1.55 0.32 0.59
0.05 0.32 1.88 0.93
0.21 0.59 0.93 3.66
```

2.3)

```
s = SquareMatrixFloat(4)

s.sampleSymmetric()
print(s)
s.toRowEchelonForm()
print(s)
```

Result:

```
The matrix is
0.1 0.75 0.05 0.21
0.75 1.55 0.32 0.59

The matrix is
1.0 0.23 0.07 0.16
0.0 1.0 0.4 0.29
0.0 0.0 1.0 0.08
0.0 0.0 0.0 1.0
```

2.4)

```
s = SquareMatrixFloat(4)
s.sampleSymmetric()
print(s.isDRDominant())
print(s)
```

Result:

```
False
The matrix is
4.44 0.05 0.39 0.54
0.05 1.22 0.78 0.67
0.39 0.78 1.28 0.94
0.54 0.67 0.94 2.33
```

```
True
The matrix is
2.22 0.22 0.12 0.08
0.22 3.76 0.95 0.36
0.12 0.95 3.57 0.87
0.08 0.36 0.87 1.79
```

2.5)

```
s = SquareMatrixFloat(4)

s.sampleSymmetric()

(e, x) = s.jSolve([1, 2, 3, 4], 10)
print(x)
print(e)
```

Result1:

```
<class 'Exception'>
Not solving because convergence not guranteed
```

Result2:

Solution:

[-0.07919356294984303, 0.24224970957088568, 1.2455888242505428, 0.9211083178601538]

Error:

[4.032955855789316, 2.8984290702885693, 2.08294828660763, 1.5088422466841935, 1.087488435141784, 0.786195796335368, 0.5673826099926563, 0.4098906856203197, 0.2959379386178865, 0.2137397389115902]

2.6)

```
s = SquareMatrixFloat(4)
s.sampleSymmetric()
(err, x) = s.gsSolve([1, 2, 3, 4], 10)
print(x)
print(err)
```

Result:

Solution:

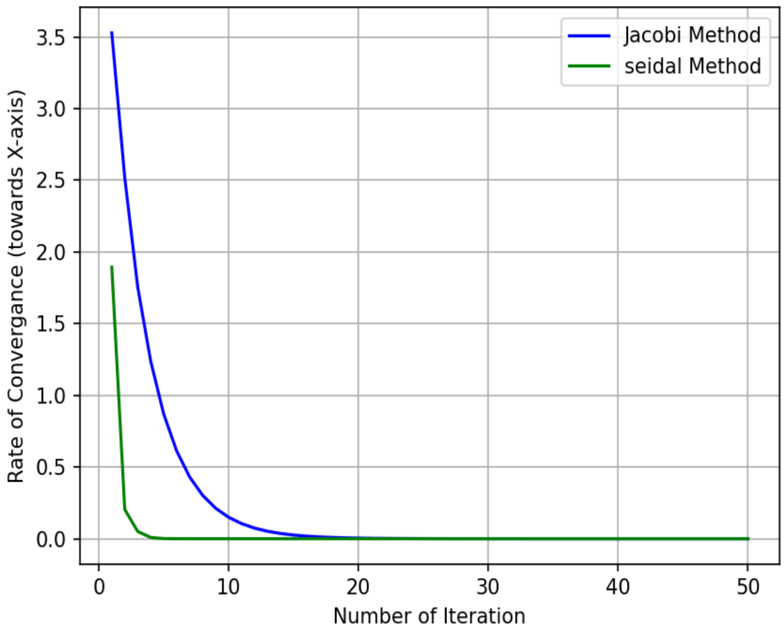
[-1.1767958462323393, 0.8589684399402326, -0.0050721566890308155, 16.573308941502873]

Error:

[4.747020872272483, 0.5079934393857451, 0.1895295148079645, 0.049007248043791536, 0.009203303795685801, 0.0012897998457234447, 0.00012069873433267575, 1.9508940875422325e-05, 6.920246185631002e-06, 1.6222793631222403e-06]

....

Q.3) Output:



Observation: From this graph we can say that Gauss Seidal method converges faster than Jacobi method.

....

Q.4) Output:

4.1)

```
p = Polynomial([1, 2, 3])
print(p)
```

Result:

```
Coefficients of the polynomial are:
 1 2 3
```

4.2)

```
p1 = Polynomial([1, 2, 3])
p2 = Polynomial([3, 2, 1])
p3 = p1 + p2
print(p3)
```

Result:

```
Coefficients of the polynomial are:
 4 4 4
```

4.3)

```
p1 = Polynomial([1, 2, 3])
p2 = Polynomial([3, 2, 1])
p3 = p1 - p2
print(p3)
```

Result:

```
Coefficients of the polynomial are:
 -2 0 2
```

4.4)

```
p1 = Polynomial([1, 2, 3])
p2 = (-0.5)*p1
print(p2)
```

Result:

```
Coefficients of the polynomial are:
 -0.5 -1.0 -1.5
```

4.5)

```
p1 = Polynomial([-1, 1])

p2 = Polynomial([1, 1, 1])
p3 = p1 * p2
print(p3)
```

Result:

```
Coefficients of the polynomial are:
 -1 0 0 1
```

4.6)

```
p = Polynomial([1, 2, 3])
print(p[2])
```

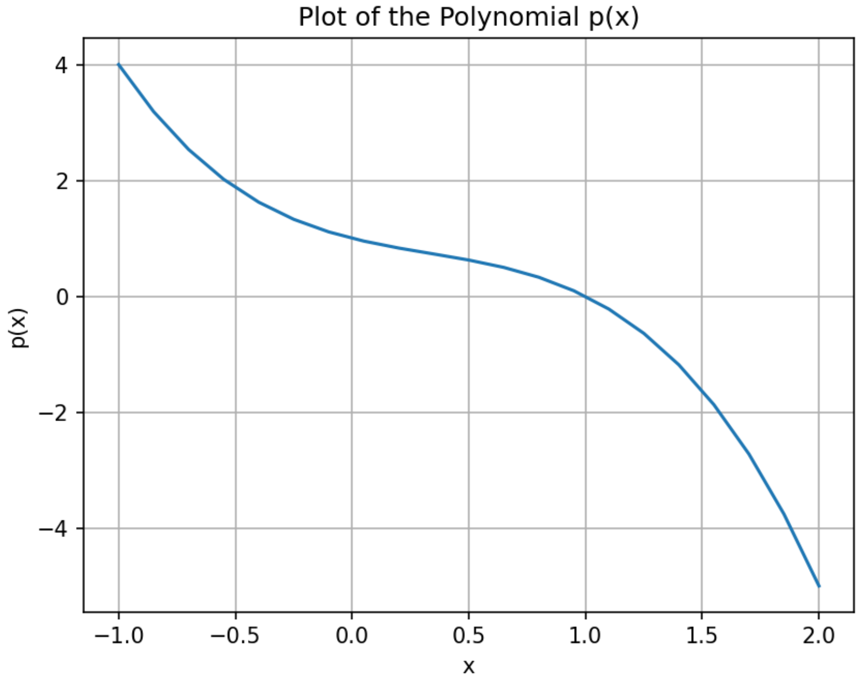
Result:

```
17
```

4.7)

```
p = Polynomial([1, -1, 1, -1])
p.show(-1, 2)
```

Result:



....