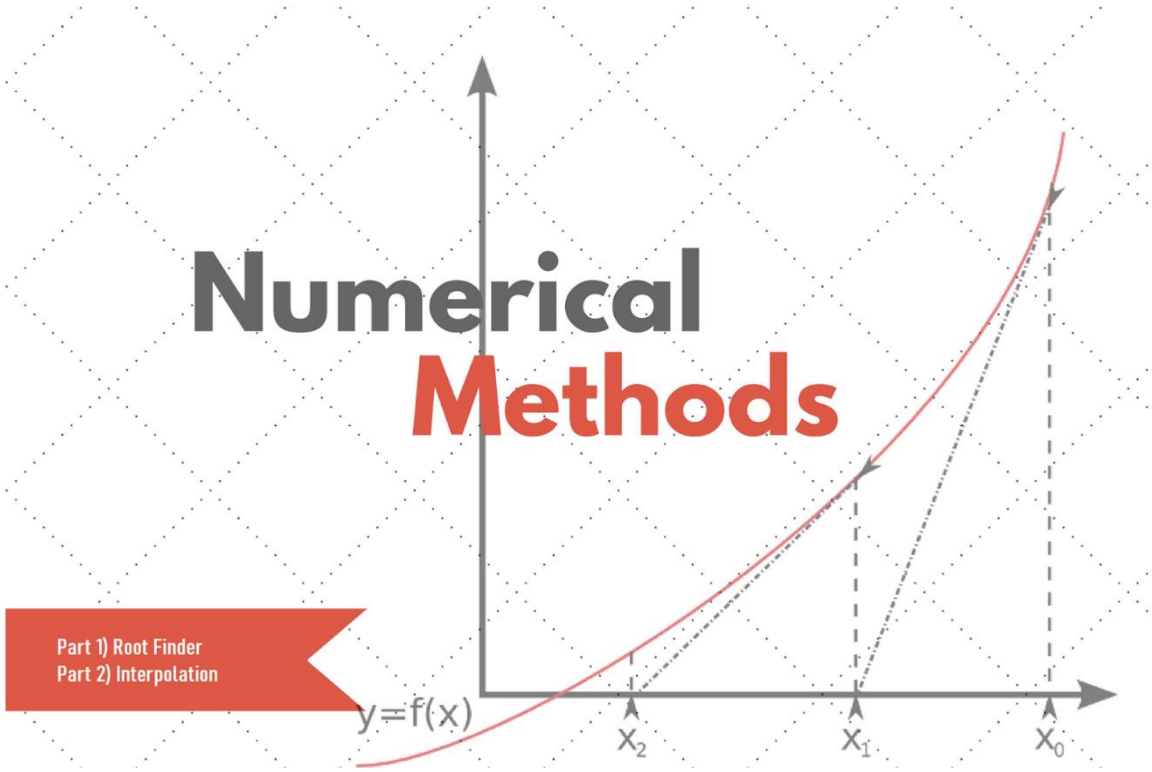
Numerical Methods

Python program to Interpolate a function



Ahmed Ayman Mohamed ID: 03
Ahmed Ali Elsayed Saber ID : 07
Islam Yousry Abdelwhaid ID: 14
Hamza Hassan Mohamed ID: 26
Mahmoud Ebrahim Elsayed ID: 58

1. Pseudo-code for methods used:

- Newton:

Method call() takes a list of x values and the corresponding f(x) to these values and returns an array of b that contains b0, b1, b2…..

```
initialize b and temp to empty list
for i to the size of array x
initialize Top to 0 and delete all elements in list temp
    for j to the size of array f(x)
    add ((fx[j+1] - fx[j])/(x[j+1]-x[Top])) to the temp
    add one to top
    end for
f(x) = temp
add first element in array f(x) to b
end for
```

end call

Method get_value() takes a list of x and a list of b and takes one query and returns a value to that query.

```
initialize value to zero
for i = size of array x - 1 to 0  subtract 1 each iteration.
    value = value + b[i]
    value = value * (query - x[i-1])
end for
value = value + b[0]
```

end get_value

- Lagrange:

Method cal() takes a list of x values and the corresponding f(x) to these values and returns an array of b that contains b0, b1, b2…..

```
for i to size of array x
    initialize mul to one
    for j to size of array x
        If i equal j
            continue
        mul = mul * (x[i] - x[j])
    end for
    b[i] = fx[i] / mul
end for
```

end cal

get_value() method takes a list of x, a list of b and takes one query and returns a value to that query.

```
initialize mul to one
for i to size of array x
    mul = mul * (query - x[i])
end for
initialize value to zero
for i to size of array x
    value = value + (b[i] * (mul / (query - x[i])))
end for
```

end get_value

2. Data Structures Used:

The lists in python are the only data structure used in this part. It is useful to hold the data in the project.

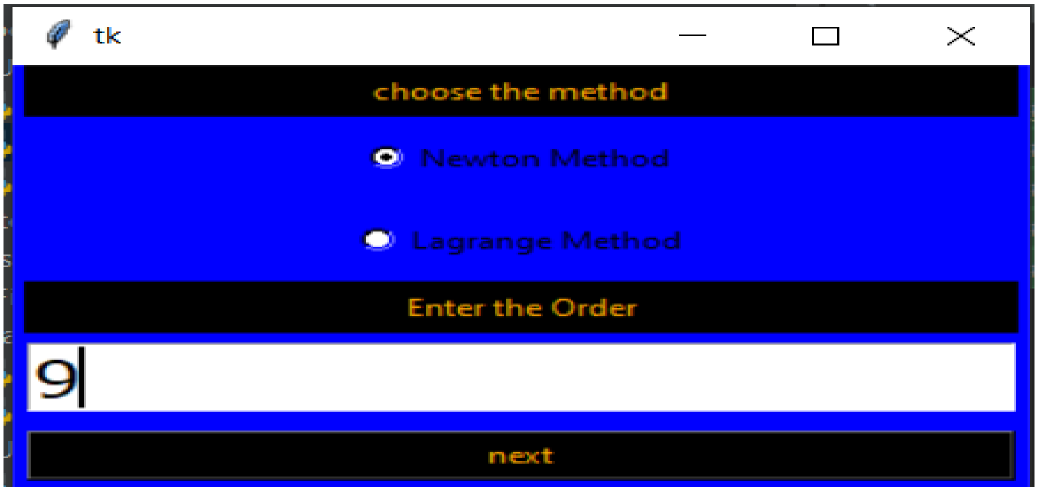
3. Problematic functions and the reason for their misbehavior:

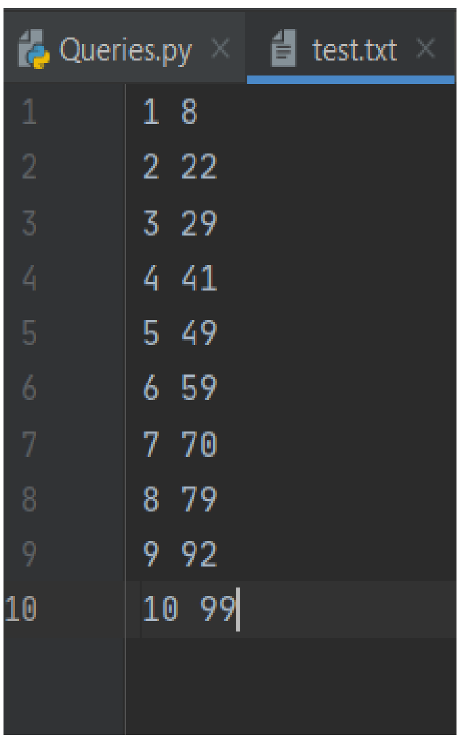
- When the user enters a query out of the range of the points, the resulting value will have a big error. To solve this problem when we have a query which is out of the range we should use regression.

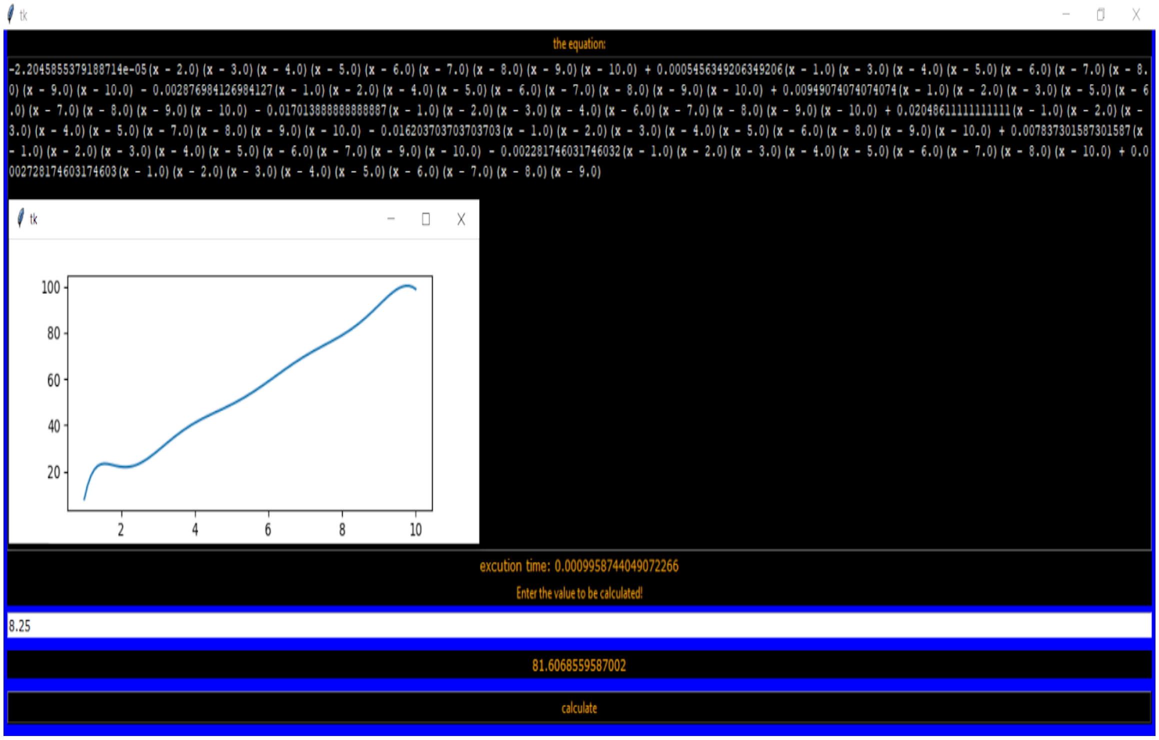
- -They don't provide a polynomial in conventional form: f(x) = a0 + a1*x + a2*x^2 + …. an*x^n. To solve this problem, we should use Gaussian Elim which O(n^3) and this approach results in a highly unstable (ill-conditioned) system of equations.

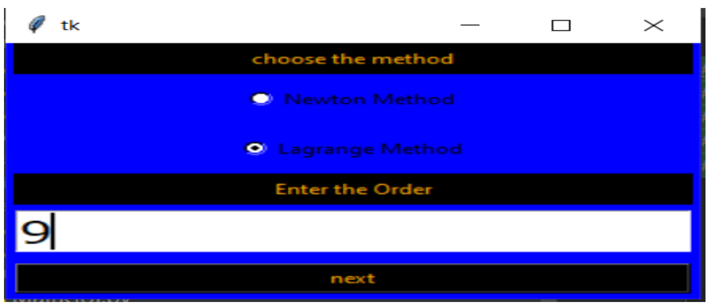
4. Sample runs:

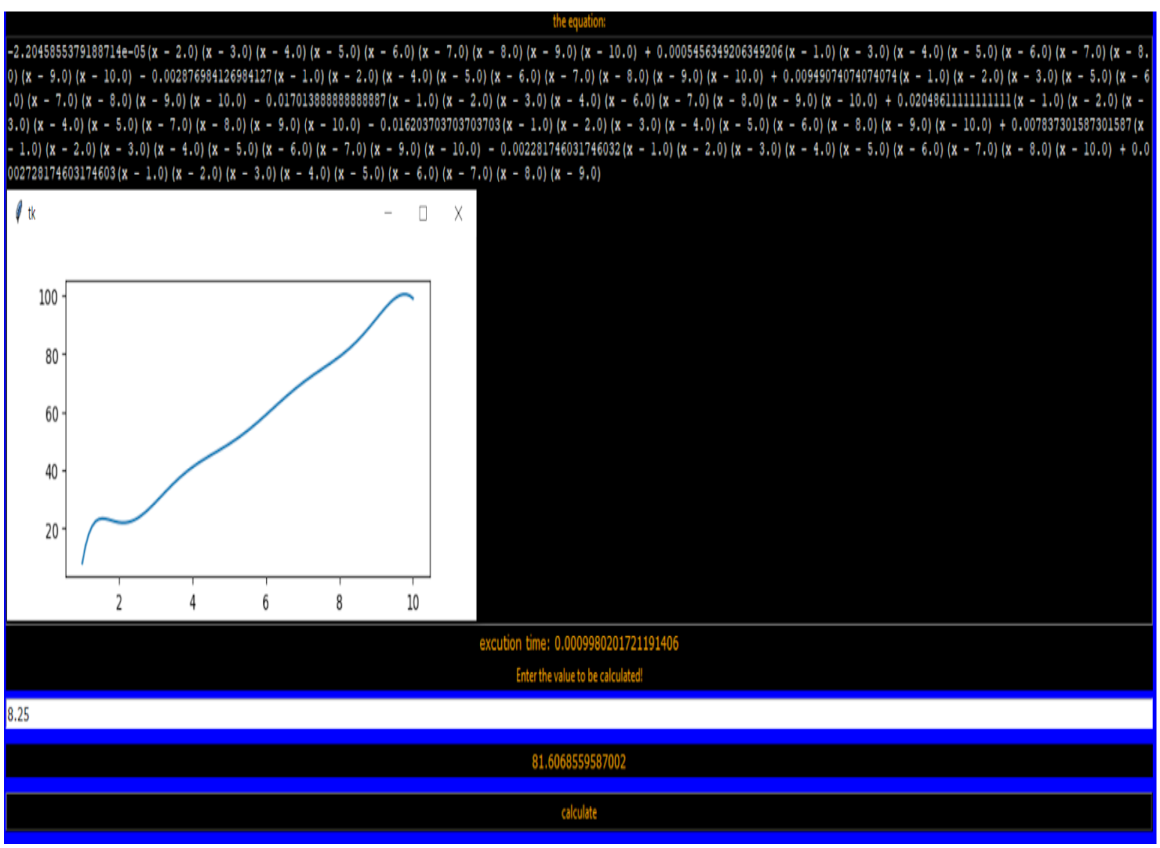
- Example 1



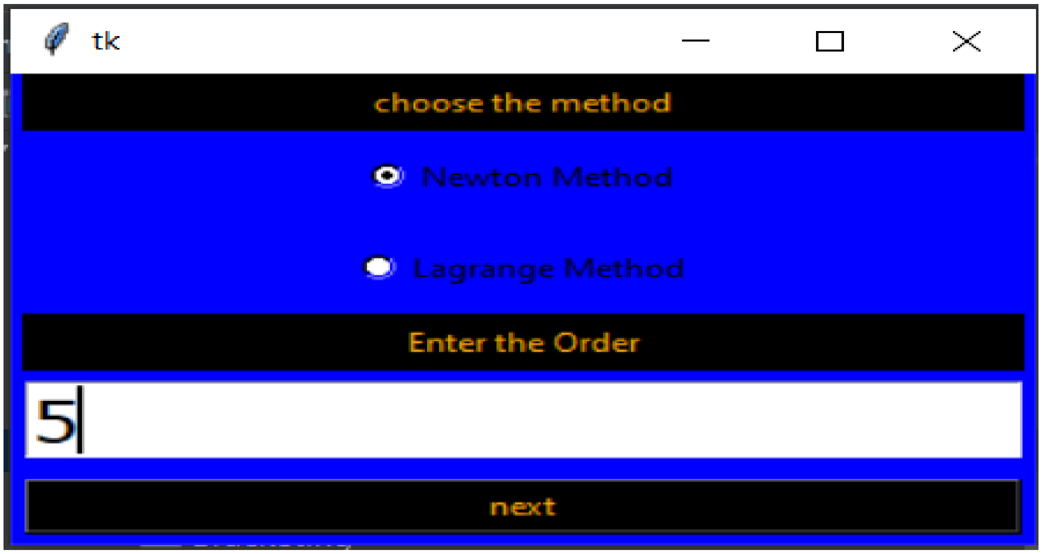


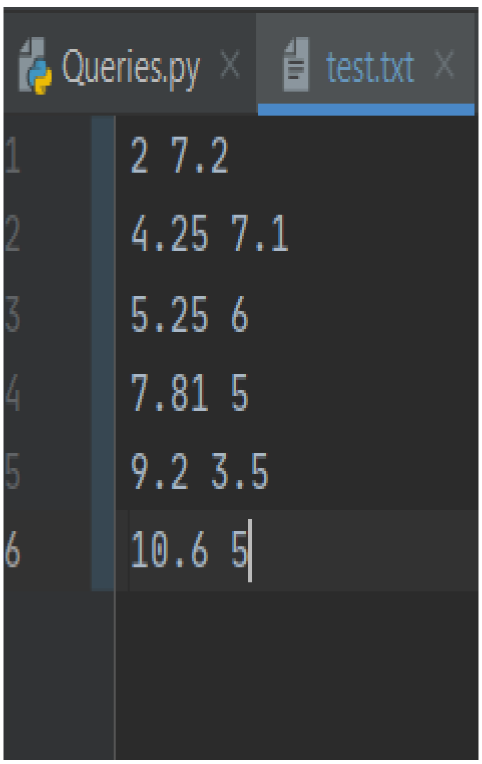

tk
the equation:
-2.2045855379188714e-05(x - 2.0)(x - 3.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 7.0)(x - 8.0)(x - 9.0)(x - 10.0) + 0.0005456349206349206(x - 1.0)(x - 3.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 7.0)(x - 8.
0)(x - 9.0)(x - 10.0) - 0.002876984126984127(x - 1.0)(x - 2.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 7.0)(x - 8.0)(x - 9.0)(x - 10.0) + 0.00949074074074074(x - 1.0)(x - 2.0)(x - 3.0)(x - 5.0)(x - 6
.0)(x - 7.0)(x - 8.0)(x - 9.0)(x - 10.0) - 0.01701388888888887(x - 1.0)(x - 2.0)(x - 3.0)(x - 4.0)(x - 6.0)(x - 7.0)(x - 8.0)(x - 9.0)(x - 10.0) + 0.02048611111111111(x - 1.0)(x - 2.0)(x -
3.0)(x - 4.0)(x - 5.0)(x - 7.0)(x - 8.0)(x - 9.0)(x - 10.0) - 0.016203703703703703(x - 1.0)(x - 2.0)(x - 3.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 8.0)(x - 9.0)(x - 10.0) + 0.007837301587301587(x
- 1.0)(x - 2.0)(x - 3.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 7.0)(x - 9.0)(x - 10.0) - 0.002281746031746032(x - 1.0)(x - 2.0)(x - 3.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 7.0)(x - 8.0)(x - 10.0) + 0.0
002728174603174603(x - 1.0)(x - 2.0)(x - 3.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 7.0)(x - 8.0)(x - 9.0)
tk
100
80
60
40
20
2
4
6
8
10
excution time: 0.0009958744049072266
Enter the value to be calculated!
8.25
81.6068559587002
calculate

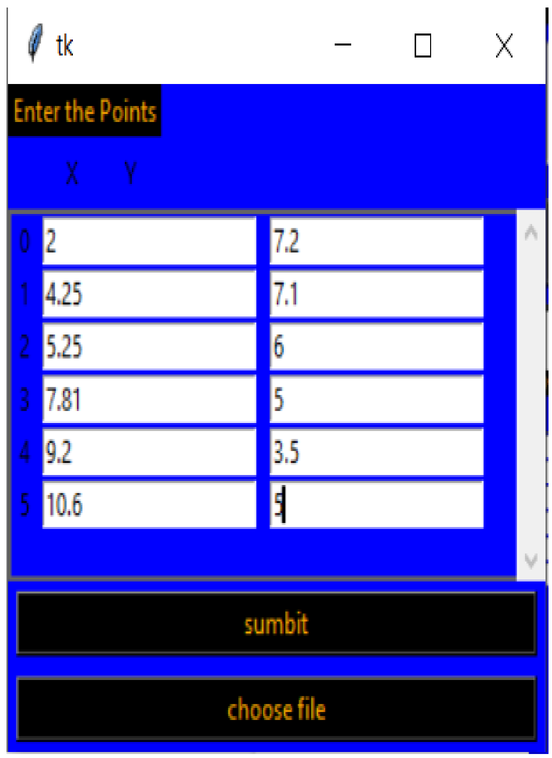

tk
choose the method
Newton Method
Lagrange Method
Enter the Order
9
next

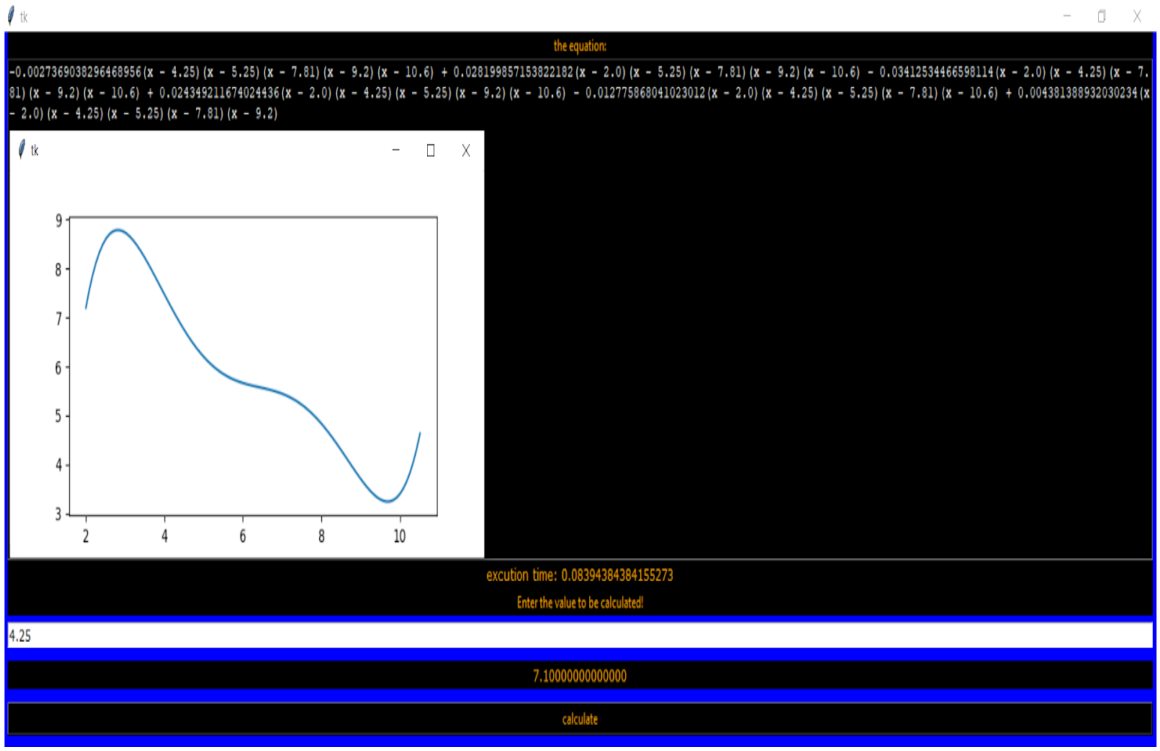

the equation:
-2.2045855379188714e-05(x - 2.0)(x - 3.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 7.0)(x - 8.0)(x - 9.0)(x - 10.0) + 0.0005456349206349206(x - 1.0)(x - 3.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 7.0)(x - 8.
0)(x - 9.0)(x - 10.0) - 0.002876984126984127(x - 1.0)(x - 2.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 7.0)(x - 8.0)(x - 9.0)(x - 10.0) + 0.00949074074074074(x - 1.0)(x - 2.0)(x - 3.0)(x - 5.0)(x - 6
.0)(x - 7.0)(x - 8.0)(x - 9.0)(x - 10.0) - 0.01701388888888887(x - 1.0)(x - 2.0)(x - 3.0)(x - 4.0)(x - 6.0)(x - 7.0)(x - 8.0)(x - 9.0)(x - 10.0) + 0.02048611111111111(x - 1.0)(x - 2.0)(x -
3.0)(x - 4.0)(x - 5.0)(x - 7.0)(x - 8.0)(x - 9.0)(x - 10.0) - 0.016203703703703703(x - 1.0)(x - 2.0)(x - 3.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 8.0)(x - 9.0)(x - 10.0) + 0.007837301587301587(x
- 1.0)(x - 2.0)(x - 3.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 7.0)(x - 9.0)(x - 10.0) - 0.002281746031746032(x - 1.0)(x - 2.0)(x - 3.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 7.0)(x - 8.0)(x - 10.0) + 0.0
002728174603174603(x - 1.0)(x - 2.0)(x - 3.0)(x - 4.0)(x - 5.0)(x - 6.0)(x - 7.0)(x - 8.0)(x - 9.0)
tk
100
80
60
40
20
2
4
6
8
10
excution time: 0.0009980201721191406
Enter the value to be calculated!
8.25
81.6068559587002
calculate

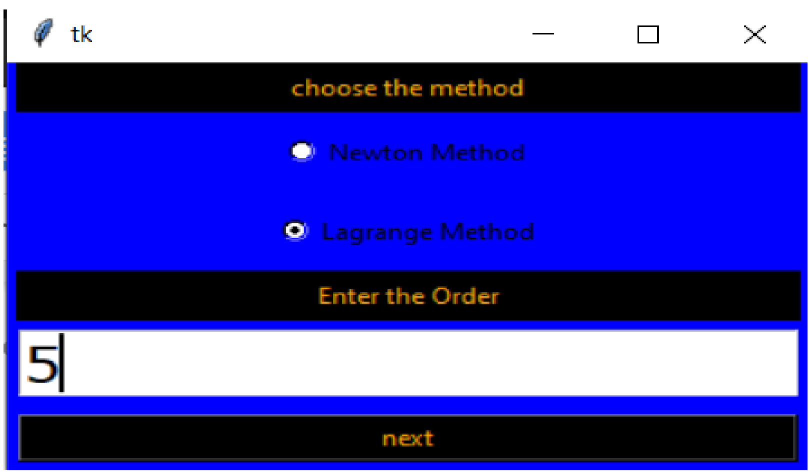
- Example 2

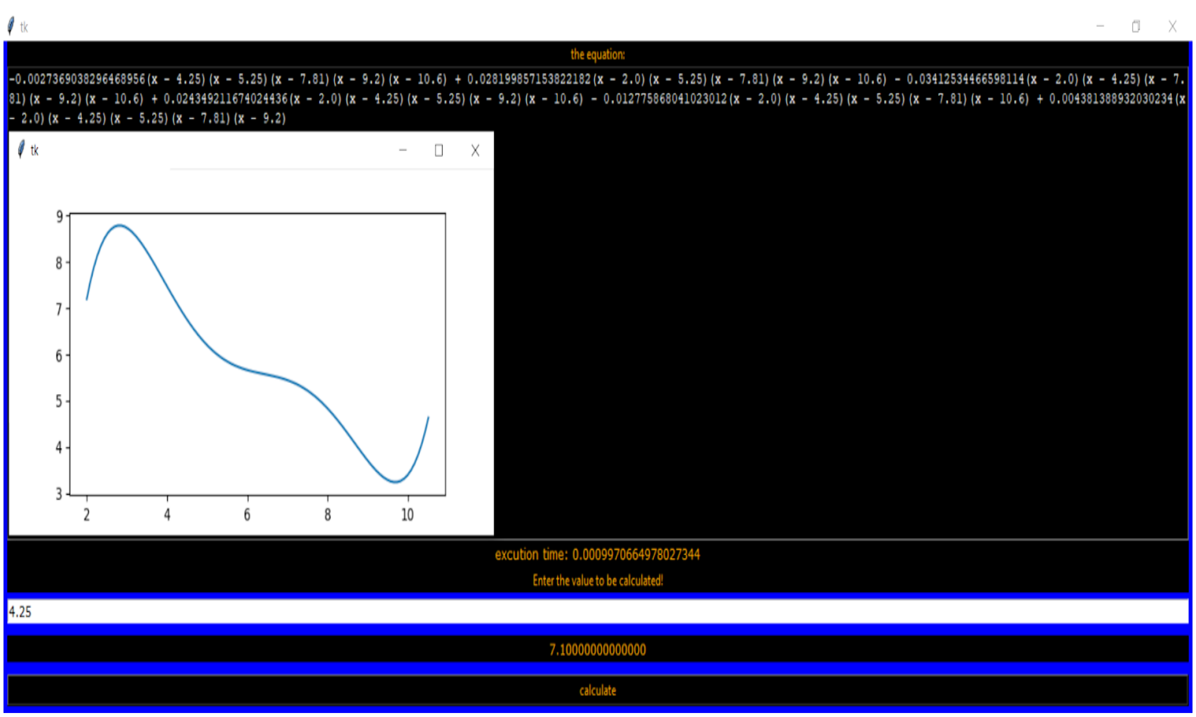

tk
choose the method
Newton Method
Lagrange Method
Enter the Order
5
next


Queries.py
test.txt
2 7.2
4.25 7.1
5.25 6
7.81 5
9.2 3.5
10.6 5

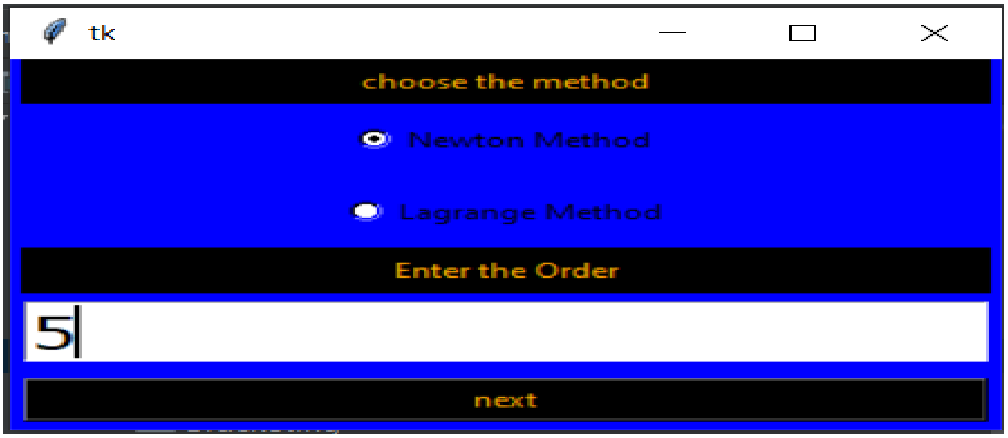

tk
Enter the Points
X Y
0 2 7.2
1 4.25 7.1
2 5.25 6
3 7.81 5
4 9.2 3.5
5 10.6 5
sumbit
choose file

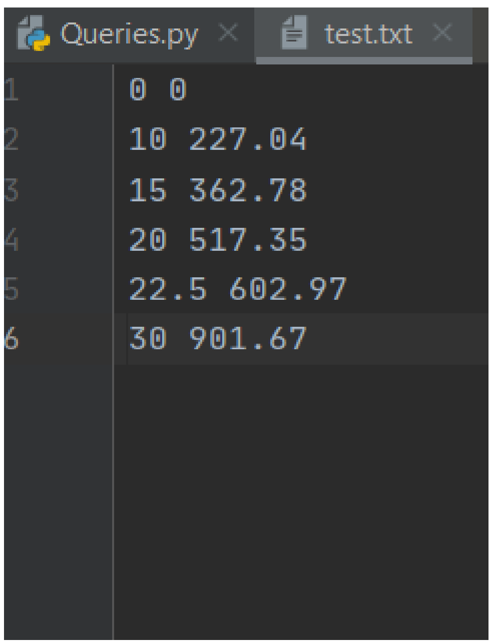

tk
the equation:
-0.0027369038296468956(x - 4.25)(x - 5.25)(x - 7.81)(x - 9.2)(x - 10.6) + 0.028199857153822182(x - 2.0)(x - 5.25)(x - 7.81)(x - 9.2)(x - 10.6) - 0.03412534466598114(x - 2.0)(x - 4.25)(x - 7.81)(x - 9.2)(x - 10.6) + 0.024349211674024436(x - 2.0)(x - 4.25)(x - 5.25)(x - 9.2)(x - 10.6) - 0.012775868041023012(x - 2.0)(x - 4.25)(x - 5.25)(x - 7.81)(x - 10.6) + 0.004381388932030234(x - 2.0)(x - 4.25)(x - 5.25)(x - 7.81)(x - 9.2)
tk
excution time: 0.08394384384155273
Enter the value to be calculated!
4.25
7.10000000000000
calculate

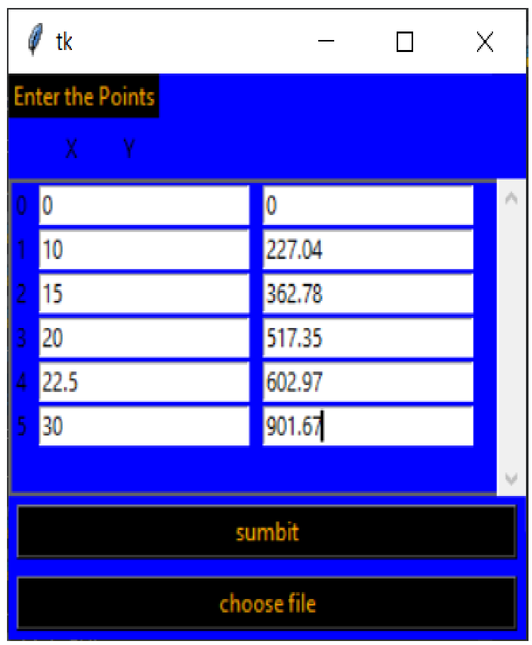

tk
choose the method
Newton Method
Lagrange Method
Enter the Order
5
next

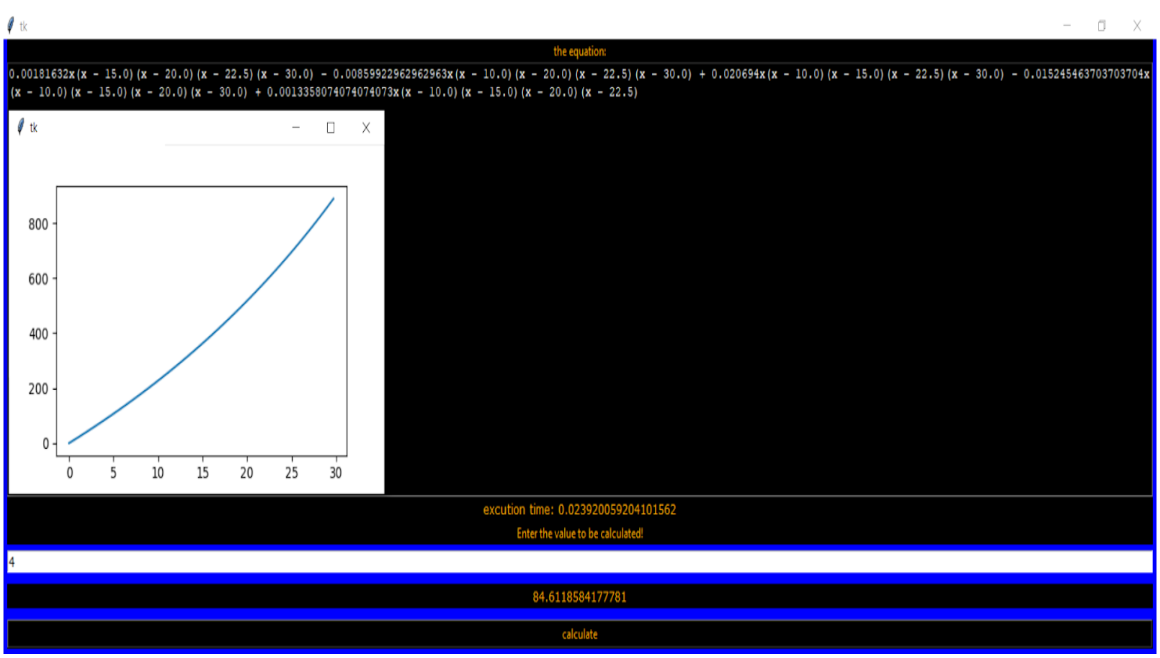

tk
the equation:
-0.0027369038296468956(x - 4.25)(x - 5.25)(x - 7.81)(x - 9.2)(x - 10.6) + 0.028199857153822182(x - 2.0)(x - 5.25)(x - 7.81)(x - 9.2)(x - 10.6) - 0.03412534466598114(x - 2.0)(x - 4.25)(x - 7.81)(x - 9.2)(x - 10.6) + 0.024349211674024436(x - 2.0)(x - 4.25)(x - 5.25)(x - 9.2)(x - 10.6) - 0.012775868041023012(x - 2.0)(x - 4.25)(x - 5.25)(x - 7.81)(x - 10.6) + 0.004381388932030234(x - 2.0)(x - 4.25)(x - 5.25)(x - 7.81)(x - 9.2)
tk
excution time: 0.0009970664978027344
Enter the value to be calculated!
4.25
7.10000000000000
calculate

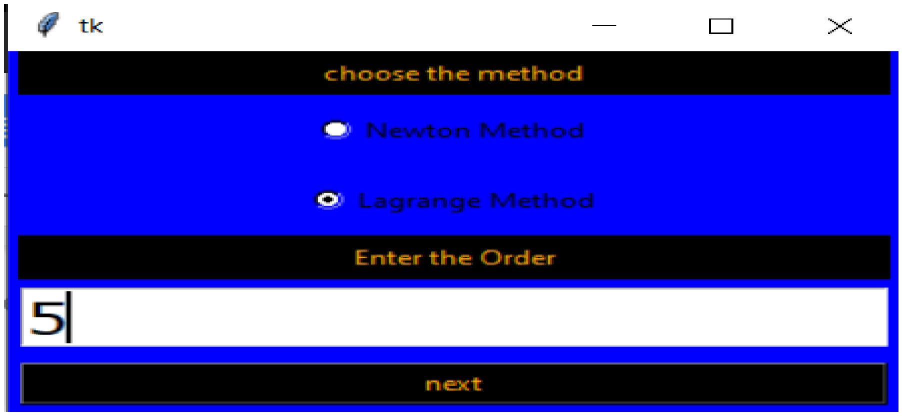
- Example 3

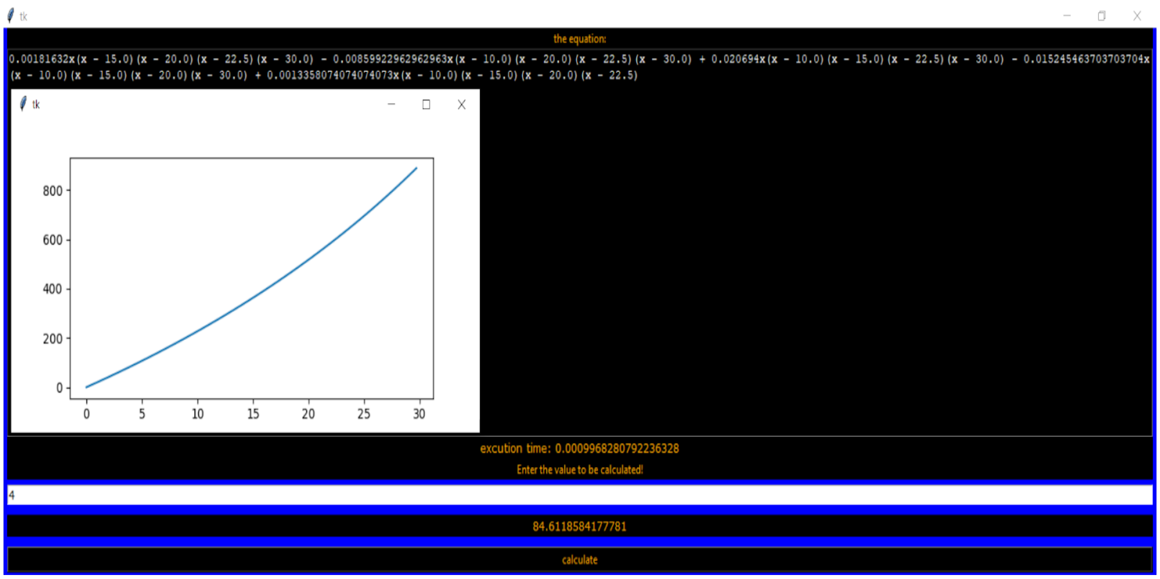

tk
choose the method
Newton Method
Lagrange Method
Enter the Order
5
next


Queries.py
test.txt
0 0
10 227.04
15 362.78
20 517.35
22.5 602.97
30 901.67


tk
Enter the Points
X
Y
0 0 0
1 10 227.04
2 15 362.78
3 20 517.35
4 22.5 602.97
5 30 901.67
sumbit
choose file


tk
the equation:
0.00181632x(x - 15.0) (x - 20.0) (x - 22.5) (x - 30.0) - 0.00859922962962963x(x - 10.0) (x - 20.0) (x - 22.5) (x - 30.0) + 0.020694x(x - 10.0) (x - 15.0) (x - 22.5) (x - 30.0) - 0.0152454637037037O4x
(x - 10.0) (x - 15.0) (x - 20.0) (x - 30.0) + 0.0013358074074074073x(x - 10.0) (x - 15.0) (x - 20.0) (x - 22.5)
tk
800
600
400
200
0
0 5 10 15 20 25 30
excution time: 0.023920059204101562
Enter the value to be calculated!
4
84.6118584177781
calculate





5. Analysis for the behavior of the last examples:

Example No.	Order	Newton execution time seconds	Lagrange execution time seconds
1	9	0.000995874404907	0.0009980201721191
2	5	0.0839438438415527	0.0009997066497802
3	5	0.0239200592041015	0.0009968280792236

- Conclusion :

  Lagrange is almostly faster than Newton method.