

Task# 2

Language : python

Goal : The use of direct methods (one-dimensional methods of exhaustive search, dichotomy, golden section search; multidimensional methods of exhaustive search, Gauss (coordinate descent), Nelder-Mead) in the tasks of unconstrained nonlinear optimization

Problem : Use the one-dimensional methods of exhaustive search, dichotomy and golden section search to find an approximate (with precision ϵ): $f(x) \rightarrow \min$ for the following functions and domains:

I.

Theory:

exhaustive search : In this approach, we generate each element in the problem and then select the ones that satisfy all the constraints, and finally find a desired element

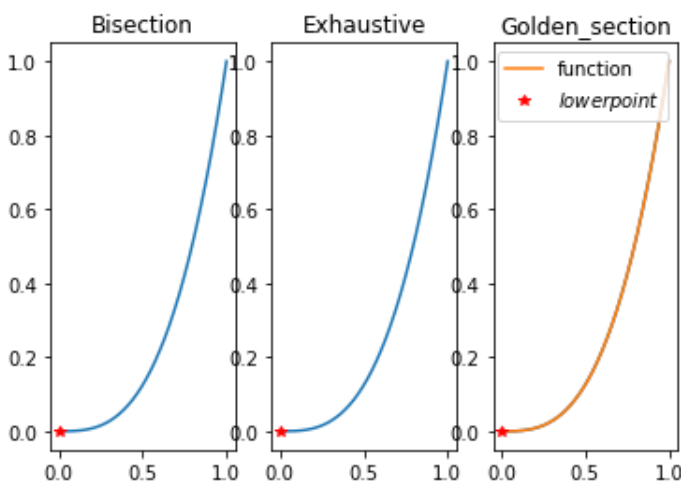
dichotomy search : use bound index to keep finding mid index , and keep trying find the smallest func(index)

golden section search : use golden rate (0.618) and two bound indexes to shrink the range and find the min

Gauss (coordinate descent) : is used to solve non-linear least squares problems, which is equivalent to minimizing a sum of squared function values

Nelder-Mead : is a numerical method used to find the minimum or maximum of an objective function by reflection, expansion, contraction and shrink coefficients in a multidimensional space

1. $f(x)=x^3, x \in [0,1];$



```
iteration is : 10
The value of root is : 0.0010
evaluation of objective function is : 41

boundary is minimum
evaluation of objective function is : 2995
iteration is : 998

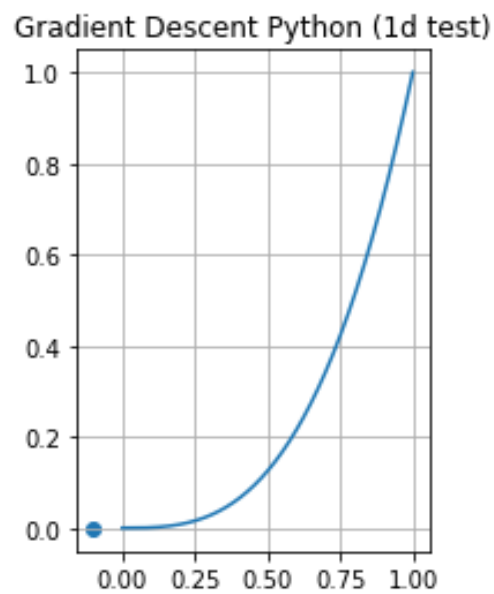
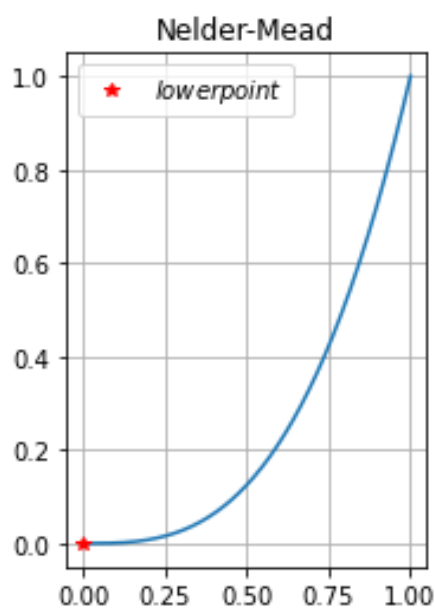
evaluation of objective function is : 32
iteration is : 15
```

(iter , evaluation)

Bisection (10,41)

Exhaustive (998,2995)

golden_sec(15,32)



```

success: True
x: array([1.e-06])
iteration is : 1
evaluation of objective function is : 2

iteration is : 1
lowest point -0.099999,-0.001000
evaluation of objective function is : 1

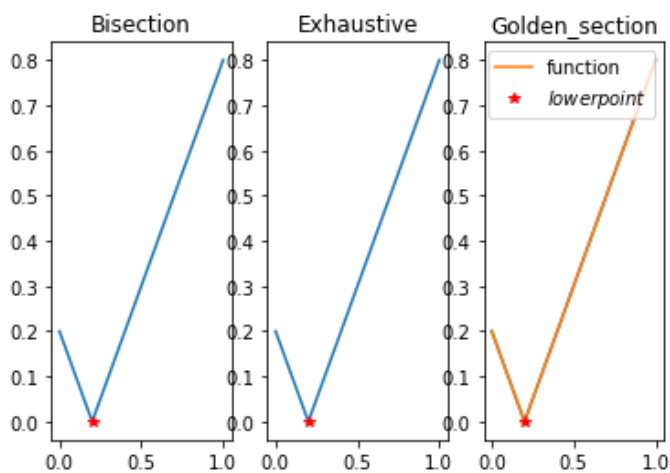
```

(iter , evaluation)

Nelder:(1,2)

Gradient:(1,1)

2. $f(x)=|x-0.2|, x \in [0,1]$;



(iter , evaluation)

Bisection (10,25)

Exhaustive (200,601)

golden_sec(15,18)

```

iteration is : 10
The value of root is : 0.2002
evaluation of objective function is : 25

```

```

The minimum point lies between 0.199000801000000025 & 0.201000799000000026
evaluation of objective function is : 601
iteration is : 200

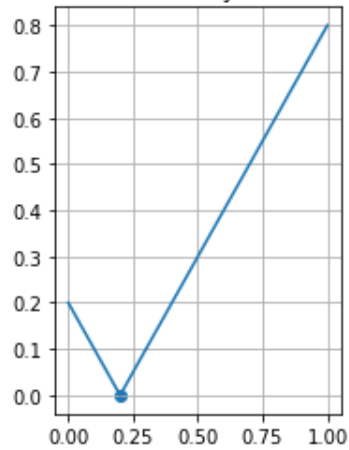
```

```

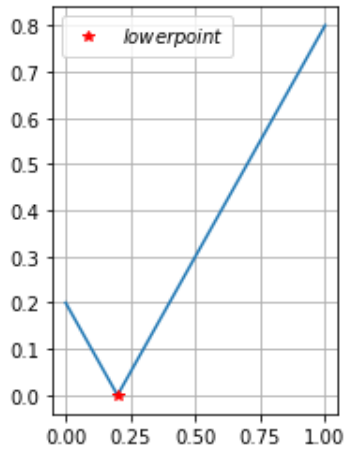
evaluation of objective function is : 18
iteration is : 15

```

Gradient Descent Python (1d test)



Nelder-Mead



```
iteration is : 2
evaluation of objective function is : 72
nelder is called : 1

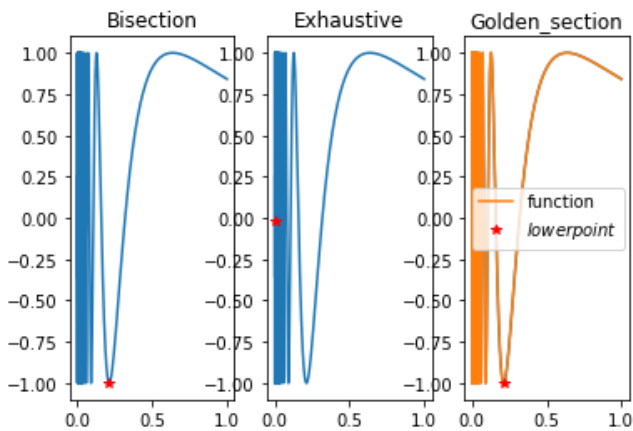
iteration is : 52
lowest point 0.199165,0.000835
evaluation of objective function is : 52
```

(iter , evaluation)

Nelder:(1,2)

Gradient:(52,52)

3. $f(x)=x\sin(1/x), x \in [0.01, 1]$:



```
iteration is : 10
The value of root is : 0.2119
evaluation of objective function is : 17

The minimum point lies between 0.001000998999999998 & 0.003000996999999999
evaluation of objective function is : 7
iteration is : 2

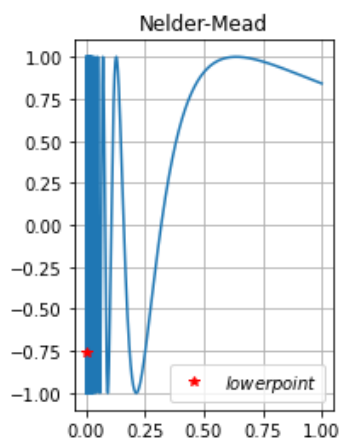
evaluation of objective function is : 18
iteration is : 15
```

(iter , evaluation)

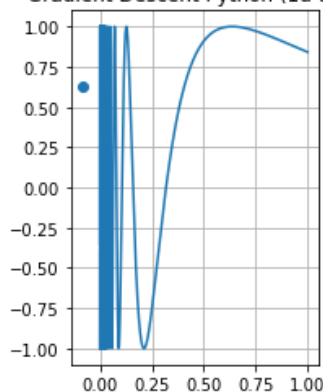
Bisection (10,17)

Exhaustive (2,7)

golden_sec(18,15)



Gradient Descent Python (1d test)



```
x: array([-9.1047800e-001])
iteration is : 1
evaluation of objective function is : 97

iteration is : 1
lowest point -0.084146,0.630563
evaluation of objective function is : 1
```

(iter , evaluation)

Nelder:(1,97)

Gradient:(1,1)

```
Help Variable Explorer Plots Code Analysis

× Console 1/A

-2.15343692e-06, -2.09386409e-06, -2.02855057e-06, -1.95749508e-06,
-1.88069631e-06, -1.79815295e-06, -1.70986369e-06, -1.61582724e-06,
-1.51604228e-06, -1.41050751e-06, -1.29922163e-06, -1.18218331e-06,
-1.05939127e-06, -9.30844190e-07, -7.96540760e-07, -6.56479675e-07,
-5.10659624e-07, -3.59079299e-07, -2.01737389e-07, -3.86325848e-08,
1.30236424e-07, 3.04870949e-07, 4.852272301e-07, 6.71441793e-07,
8.63380735e-07, 1.06109944e-06, 1.26457222e-06, 1.47382740e-06,
1.68885727e-06, 1.90966317e-06, 2.13624639e-06, 2.36860827e-06,
2.60675011e-06, 2.85067322e-06, 3.10037893e-06, 3.35586855e-06,
3.61714340e-06, 3.88420479e-06, 4.15705405e-06, 4.43569249e-06,
4.72012143e-06], 'nfev': 13, 'fjac': array([[ -1.00879539e+01, 9.91355993e-02,
9.1430693e-02, 9.91505416e-02, 9.91580146e-02, 9.91654891e-02,
9.91729643e-02, 9.91804403e-02, 9.91879177e-02, 9.91953959e-02,
9.92028763e-02, 9.92103567e-02, 9.92178386e-02, 9.92253220e-02,
9.92328061e-02, 9.92402917e-02, 9.92477780e-02, 9.92552658e-02,
9.92627544e-02, 9.92702444e-02, 9.92777352e-02, 9.92852275e-02,
9.92927205e-02, 9.93002149e-02, 9.93077109e-02, 9.93152076e-02,
9.93227058e-02, 9.93302040e-02, 9.93377044e-02, 9.93452055e-02,
9.93527081e-02, 9.93602115e-02, 9.93677163e-02, 9.93752219e-02,
9.93827290e-02, 9.93902368e-02, 9.93977454e-02, 9.94052561e-02,
9.94127684e-02, 9.94202806e-02, 9.94277929e-02, 9.94353096e-02,
9.94428248e-02, 9.94503415e-02, 9.94578597e-02, 9.94653769e-02,
9.94728941e-02, 9.94804113e-02, 9.94879285e-02, 9.94954457e-02,
9.95029629e-02, 9.95104801e-02, 9.95180000e-02, 9.95255199e-02,
9.95330398e-02, 9.95405597e-02, 9.95480796e-02, 9.95555995e-02,
9.95631194e-02, 9.95706393e-02, 9.95781592e-02, 9.95856791e-02,
9.95931990e-02, 9.96007189e-02, 9.96082388e-02, 9.96157587e-02,
9.96232786e-02, 9.96307985e-02, 9.96383184e-02, 9.96458383e-02,
9.96533582e-02, 9.96608781e-02, 9.96683980e-02, 9.96759179e-02,
9.96834378e-02, 9.96909577e-02, 9.96984776e-02, 9.97059975e-02,
9.97135174e-02, 9.97210373e-02, 9.97285572e-02, 9.97360771e-02,
9.97435970e-02, 9.97511169e-02, 9.97586368e-02, 9.97661567e-02,
9.97736766e-02, 9.97811965e-02, 9.97887164e-02, 9.97962363e-02,
9.98037562e-02, 9.98112761e-02, 9.98187960e-02, 9.98263159e-02,
9.98338358e-02, 9.98413557e-02, 9.98488756e-02, 9.98563955e-02,
9.98639154e-02, 9.98714353e-02, 9.98789552e-02, 9.98864751e-02,
9.98939950e-02, 9.99015149e-02, 9.99090348e-02, 9.99165547e-02,
9.99240746e-02, 9.99315945e-02, 9.99391144e-02, 9.99466343e-02,
9.99541542e-02, 9.99616741e-02, 9.99691940e-02, 9.99767139e-02,
9.99842338e-02, 9.99917537e-02, 1.00000000e+00], dtype=float64))

9.99917537e-02, 1.00000000e+00], dtype=float64))
```

Conclusion :

In first section , through graphs which I created , we can see that in most of formulas , we can achieve minimum values from each algorithm which I tested . But as we can see I found difficulty to use gauss and Nelder-Mead to find minimum from any function with trigonometric .

Most of time it cost a lot of iterations for exhaustive search to find a min , but somehow in third one with trigonometric , it only cost 2 times of iterations and 7 times of function , probably there is error in my algorithm . But we still can conclude that exhaustive search is not efficient to make calculation , it takes too much time to calculate each possibility .

On the contrary , we can see that Nelder-Mead and gauss show a very efficient way to get to result . But there is little mistake , so I can not get to desired answer .

In the second section , we used linear and rational approximant to try to approximate the data . As we can see that linear one complete with prettier way , and rational goes to wrong side . Maybe it is because the original data that I use to approximate is closer to a line , so rational approximant can not optimize it with better way . Even though we still can see that number of evaluation for rational approximant is higher .

Appendix

<https://github.com/MaChengYuan/task2.git>