

# Python 的数值计算

刘德华

2023 年 7 月 16 日

## 1 定积分

### 1.0.1 一元定积分

```
[1]: # 导入积分模块
import scipy.integrate as integrate
# 导入基础计算库
import numpy as np

[2]: result = integrate.quad(lambda x: np.sin(x)+3*x**2-3*x, 0, 4.5)
result

[2]: (61.960795799430784, 6.912907000856692e-13)
```

### 1.0.2 一元定积分，给定参数

```
[1]: # 导入积分模块
import scipy.integrate as integrate
# 导入基础计算库
import numpy as np

[2]: # 定义函数
def integrand(x, a, b):
    return a*x**2 + b

a = 2
b = 1

[3]: I = integrate.quad(integrand, 0, 1, args=(a,b))
I

[3]: (1.6666666666666667, 1.8503717077085944e-14)
```

### 1.0.3 二元重积分

```
[1]: # 导入积分
from scipy.integrate import dblquad
# 导入基础计算库
import numpy as np
# 二重积分
def I(n):
    return dblquad(
        lambda t, x: np.exp(-x*t)/t**n,
        0,
        np.inf,
        lambda x: 1,
        lambda x: np.inf
    )
```

```
[2]: print(I(4))
```

(0.2500000000043577, 1.298303346936809e-08)

```
[3]: print(I(3))
```

(0.33333333325010883, 1.3888461883425516e-08)

```
[4]: print(I(2))
```

(0.4999999999985751, 1.3894083651858995e-08)

```
[5]: area = dblquad(  
    lambda x, y: x*y,  
    0,  
    0.5,  
    lambda x: 0,  
    lambda x: 1-2*x  
)  
area
```

```
[5]: (0.010416666666666668, 4.101620128472366e-16)
```

#### 1.0.4 辛普森法计算积分

```
[1]: # 导入积分模块  
import scipy.integrate as integrate  
# 导入基础计算库  
import numpy as np
```

```
[2]: def f1(x):  
    return x**2  
  
def f2(x):  
    return x**3
```

```
[3]: x = np.array([1,3,4])  
y1 = f1(x)  
# 辛普森数值计算积分  
I1 = integrate.simpson(y1, x)  
print(I1)
```

```
[4]: y2 = f2(x)
      I2 = integrate.simpson(y2, x)
      print(I2)
```

61.5

## 2 最优化

### 2.0.1 多元函数无约束的最优化问题

```
[1]: # 导入基础计算库
      import numpy as np
      # 导入最优化库
      from scipy.optimize import minimize
```

```
[2]: def rosen(x):
      """The Rosenbrock function"""
      return sum(100.0*(x[1:]-x[:-1]**2.0)**2.0 + (1-x[:-1])**2.0)
```

```
[3]: # 初始值
      x0 = np.array([1.3, 0.7, 0.8, 1.9, 1.2])
      res = minimize(
          rosen, x0, method="nelder-mead",
          options={"xatol": 1e-8, "disp": True}
      )
      print(res)
```

Optimization terminated successfully.

Current function value: 0.000000

Iterations: 339

Function evaluations: 571

message: Optimization terminated successfully.

success: True

status: 0

fun: 4.861153433422115e-17

x: [ 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00]

nit: 339

nfev: 571

```
final_simplex: (array([[ 1.000e+00,  1.000e+00, ...,  1.000e+00,
                        1.000e+00],
                       [ 1.000e+00,  1.000e+00, ...,  1.000e+00,
                        1.000e+00],
                       ...,
                       [ 1.000e+00,  1.000e+00, ...,  1.000e+00,
                        1.000e+00],
```

```
[ 1.000e+00,  1.000e+00, ...,  1.000e+00,
  1.000e+00]], array([ 4.861e-17,  7.652e-17,
 8.114e-17,  8.633e-17,
 8.641e-17,  2.179e-16]))
```

```
[4]: # 最小值点
      print(res.x)
```

```
[1.  1.  1.  1.  1.]
```

```
[5]: # 目标函数的梯度
      def rosen_der(x):
          xm = x[1:-1]
          xm_m1 = x[:-2]
          xm_p1 = x[2:]
          der = np.zeros_like(x)
          der[1:-1] = 200*(xm-xm_m1**2) - 400*(xm_p1 - xm**2)*xm - 2*(1-xm)
          der[0] = -400*x[0]*(x[1]-x[0]**2) - 2*(1-x[0])
          der[-1] = 200*(x[-1]-x[-2]**2)
          return der
```

```
[6]: res = minimize(
      rosen, x0, method="BFGS", jac=rosen_der,
      options={"disp": True}
      )
      print(res)
```

Optimization terminated successfully.

Current function value: 0.000000

Iterations: 25

Function evaluations: 30

Gradient evaluations: 30

message: Optimization terminated successfully.

success: True

status: 0

fun: 4.0130879949972905e-13

x: [ 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00]

nit: 25

jac: [-5.690e-06 -2.733e-06 -2.545e-06 -7.735e-06 5.781e-06]

hess\_inv: [[ 7.588e-03 1.244e-02 ... 4.615e-02 9.222e-02]

[ 1.244e-02 2.482e-02 ... 9.299e-02 1.857e-01]

...

[ 4.615e-02 9.299e-02 ... 3.738e-01 7.462e-01]

[ 9.222e-02 1.857e-01 ... 7.462e-01 1.494e+00]]

nfev: 30

njev: 30

```
[7]: # 最小值点
      print(res.x)
```

```
[1.00000004 1.00000001 1.00000021 1.00000044 1.00000092]
```

```
[8]: # 目标函数的黑塞矩阵
      def rosen_hess(x):
          x = np.asarray(x)
          H = np.diag(-400*x[:-1],1) - np.diag(400*x[:-1],-1)
          diagonal = np.zeros_like(x)
          diagonal[0] = 1200*x[0]**2-400*x[1]+2
          diagonal[-1] = 200
          diagonal[1:-1] = 202 + 1200*x[1:-1]**2 - 400*x[2:]
          H = H + np.diag(diagonal)
          return H
```

```
[9]: res = minimize(
      rosen, x0, method="Newton-CG",
      jac=rosen_der, hess=rosen_hess,
      options={"xtol": 1e-8, "disp": True}
      )
      print(res)
```

Optimization terminated successfully.

Current function value: 0.000000

Iterations: 24

Function evaluations: 33

Gradient evaluations: 33

Hessian evaluations: 24

message: Optimization terminated successfully.

success: True

status: 0

fun: 3.5306674342205174e-17

x: [ 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00]

nit: 24

jac: [ 2.687e-08 9.267e-08 3.701e-07 1.485e-06 -8.526e-07]

nfev: 33

njev: 33

nhev: 24

```
[10]: # 最小值点
       print(res.x)
```

```
[1.          1.          1.          0.99999999 0.99999999]
```

```
[11]: res = minimize(
    rosen, x0, method="trust-ncg",
    jac=rosen_der, hess=rosen_hess,
    options={"gtol": 1e-8, "disp": True}
)
print(res)
```

Optimization terminated successfully.

Current function value: 0.000000

Iterations: 20

Function evaluations: 21

Gradient evaluations: 20

Hessian evaluations: 19

message: Optimization terminated successfully.

success: True

status: 0

fun: 1.232595164407831e-30

x: [ 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00]

nit: 20

jac: [-0.000e+00 0.000e+00 0.000e+00 4.441e-14 -2.220e-14]

nfev: 21

njev: 20

nhev: 19

hess: [[ 8.020e+02 -4.000e+02 ... 0.000e+00 0.000e+00]

[-4.000e+02 1.002e+03 ... 0.000e+00 0.000e+00]

...

[ 0.000e+00 0.000e+00 ... 1.002e+03 -4.000e+02]

[ 0.000e+00 0.000e+00 ... -4.000e+02 2.000e+02]]

```
[12]: # 最小值点
```

```
print(res.x)
```

```
[1. 1. 1. 1. 1.]
```

```
[13]: res = minimize(
    rosen, x0, method="trust-krylov",
    jac=rosen_der, hess=rosen_hess,
    options={"gtol": 1e-8, "disp": True}
)
print(res)
```

iter	inewton	type	objective	$\hat{a}$	$\hat{g}$	$\hat{a}_{\text{M}} \hat{A}^{-1}$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}^1$	$\hat{I}^2$					
0	0	cg_i	-6.273083e+02	4.029038e+02	0.000000e+00	0.000000e+00		

2.246107e+03 4.021147e+03 2.486853e-04 3.217671e-02

iter	inewton	type	objective	$\hat{a}$	$\hat{g}$	$\hat{a}_{\hat{M}} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I}_{\pm}$			$\hat{I}^2$		
0	0	cg_i	-9.528585e+01	1.478412e+02	0.000000e+00	0.000000e+00		
6.001708e+02	1.890129e+03		5.290645e-04	6.067939e-02				

iter	inewton	type	objective	$\hat{a}$	$\hat{g}$	$\hat{a}_{\hat{M}} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I}_{\pm}$			$\hat{I}^2$		
0	0	cg_i	-8.662599e+00	5.824611e+01	0.000000e+00	0.000000e+00		
1.285783e+02	9.542387e+02		1.047956e-03	2.052100e-01				

iter	inewton	type	objective	$\hat{a}$	$\hat{g}$	$\hat{a}_{\hat{M}} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I}_{\pm}$			$\hat{I}^2$		
0	0	cg_i	-1.798598e+00	3.481545e+01	0.000000e+00	0.000000e+00		
5.168333e+01	7.425689e+02		1.346676e-03	4.537776e-01				

iter	inewton	type	objective	$\hat{a}$	$\hat{g}$	$\hat{a}_{\hat{M}} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I}_{\pm}$			$\hat{I}^2$		
1	0	cg_i	-2.721261e+00	1.426076e+01	0.000000e+00	0.000000e+00		
5.002168e+02	9.938183e+02		1.522401e-03	1.677804e-01				

iter	inewton	type	objective	$\hat{a}$	$\hat{g}$	$\hat{a}_{\hat{M}} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I}_{\pm}$			$\hat{I}^2$		
0	0	cg_i	-7.242386e-02	4.575385e+00	0.000000e+00	0.000000e+00		
1.400669e+01	1.354439e+03		7.383128e-04	1.067048e-01				

iter	inewton	type	objective	$\hat{a}$	$\hat{g}$	$\hat{a}_{\hat{M}} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I}_{\pm}$			$\hat{I}^2$		
0	0	cg_i	-8.331939e-03	2.017934e+00	0.000000e+00	0.000000e+00		
4.550681e+00	1.242730e+03		8.046802e-04	1.966351e-01				

iter	inewton	type	objective	$\hat{a}$	$\hat{g}$	$\hat{a}_{\hat{M}} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I}_{\pm}$			$\hat{I}^2$		
0	0	cg_i	-1.639774e-03	1.039910e+00	0.000000e+00	0.000000e+00		
2.017620e+00	1.241266e+03		8.056290e-04	2.656515e-01				

iter	inewton	type	objective	$\hat{a}$	$\hat{g}$	$\hat{a}_{\hat{M}} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I}_{\pm}$			$\hat{I}^2$		
1	0	cg_i	-2.362748e-03	4.063620e-01	0.000000e+00	0.000000e+00		
6.397659e+02	1.077635e+03		1.337094e-03	1.526986e-01				

iter	inewton	type	objective	$\hat{a}$	$\hat{g}$	$\hat{a}_{\hat{M}} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I}_{\pm}$			$\hat{I}^2$		
0	0	cg_i	-1.218928e-04	2.280362e-01	0.000000e+00	0.000000e+00		
4.066296e-01	6.782503e+02		1.474382e-03	3.144920e-01				

iter	inewton	type	objective	$\hat{a}$	$\hat{g}$	$\hat{a}_{\hat{M}} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I}_{\pm}$			$\hat{I}^2$		
1	0	cg_i	-1.746236e-04	3.200387e-01	0.000000e+00	0.000000e+00		
3.803600e+02	7.063791e+02		2.028090e-03	1.969687e+00				



```

2      0  cg_i -3.519759e-04  3.984883e-01  0.000000e+00  0.000000e+00
6.920085e+02  1.259964e+03  3.463074e-03  1.550338e+00

```

```

3      0  cg_i -5.758296e-03  3.569591e+00  0.000000e+00  0.000000e+00
3.595435e+02  4.623626e+02  6.809270e-02  8.024274e+01

```

TR Solving trust region problem, radius 2.500000e-01; starting on first irreducible block

TR Coldstart. Seeking suitable initial  $\hat{I} \gg \hat{a}$ , starting with 0

TR Starting Newton iteration for  $\hat{I} \gg \hat{a}$  with initial choice 0.000000e+00

TR iter  $\hat{I} \gg$   $d\hat{I} \gg$   $\hat{a} \text{ h}\hat{a} (\hat{I} \gg) \hat{a}$  -radius

TR 1 2.943145e-01 2.943145e-01 3.059076e-07

TR 2 2.943152e-01 7.617834e-07 -1.912359e-14

iter	inewton	type	objective	$\hat{I}^3 \hat{a} \mu \hat{c} \hat{a} \hat{a}$	$ \text{h}\hat{a} \mu \hat{c} $	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$					
4	2	hot	-2.875301e-02	0.000000e+00	0.000000e+00	2.943152e-01	
1.315535e+02			1.382362e+03				

Early exit as hotstart with early termination on

iter	inewton	type	objective	$\hat{a} \text{ g}\hat{a} \hat{a} \text{ _M}\hat{a} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I} \pm$	$\hat{I}^2$		
0	0	cg_i	-7.518233e-03	1.522235e+00	0.000000e+00	0.000000e+00
4.090478e+00			1.112762e+03	8.986647e-04	1.384891e-01	

iter	inewton	type	objective	$\hat{a} \text{ g}\hat{a} \hat{a} \text{ _M}\hat{a} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I} \pm$	$\hat{I}^2$		
0	0	cg_i	-1.096841e-03	8.409536e-01	0.000000e+00	0.000000e+00
1.518407e+00			1.051001e+03	9.514742e-04	3.067379e-01	

iter	inewton	type	objective	$\hat{a} \text{ g}\hat{a} \hat{a} \text{ _M}\hat{a} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
1	0	cg_i	-1.570721e-03	3.392978e-01	0.000000e+00	0.000000e+00
5.820853e+02			1.068564e+03	1.340154e-03	1.627863e-01	

iter	inewton	type	objective	$\hat{a} \text{ g}\hat{a} \hat{a} \text{ _M}\hat{a} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I} \pm$	$\hat{I}^2$		
0	0	cg_i	-6.479100e-05	1.641071e-01	0.000000e+00	0.000000e+00
3.391135e-01			8.874534e+02	1.126820e-03	2.341879e-01	

iter	inewton	type	objective	$\hat{a} \text{ g}\hat{a} \hat{a} \text{ _M}\hat{a} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
1	0	cg_i	-8.150449e-05	1.044376e-01	0.000000e+00	0.000000e+00
4.294650e+02			1.013502e+03	1.241202e-03	4.050039e-01	

iter	inewton	type	objective	$\hat{a} \text{ g}\hat{a} \hat{a} \text{ _M}\hat{a} \gg \hat{A}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$		$\hat{I}'$	$\hat{I} \pm$	$\hat{I}^2$		

```

0      0  cg_i -6.864955e-06  9.214155e-02  0.000000e+00  0.000000e+00
1.044215e-01  7.941673e+02  1.259181e-03  7.786304e-01

```

```

1      0  cg_i -1.991674e-05  1.301343e-01  0.000000e+00  0.000000e+00
7.007735e+02  9.436083e+02  3.074601e-03  1.994676e+00

```

```

2      0  cg_i -1.195321e-04  3.179941e-01  0.000000e+00  0.000000e+00
4.593539e+02  7.337610e+02  1.176448e-02  5.971103e+00

```

```

3      0  cg_i -4.188868e-04  7.535566e-01  0.000000e+00  0.000000e+00
2.077086e+02  6.764504e+02  5.920769e-03  5.615569e+00

```

```

iter inewton type    objective    â gâ â_Mâ »Â¹    leftmost    Î»
Î³      Î´      Î±      Î²
0      0  cg_i -8.816451e-04  5.184366e-01  0.000000e+00  0.000000e+00
1.324450e+00  9.948272e+02  1.005200e-03  1.532216e-01

```

```

iter inewton type    objective    â gâ â_Mâ »Â¹    leftmost    Î»
Î³      Î´      Î±      Î²
0      0  cg_i -1.332921e-04  2.868284e-01  0.000000e+00  0.000000e+00
5.178932e-01  1.006112e+03  9.939256e-04  3.067355e-01

```

```

1      0  cg_i -1.931659e-04  1.227712e-01  0.000000e+00  0.000000e+00
5.572219e+02  9.956424e+02  1.455536e-03  1.832097e-01

```

```

iter inewton type    objective    â gâ â_Mâ »Â¹    leftmost    Î»
Î³      Î´      Î±      Î²
0      0  cg_i -7.494787e-06  5.036955e-02  0.000000e+00  0.000000e+00
1.227312e-01  1.004896e+03  9.951282e-04  1.684325e-01

```

```

1      0  cg_i -9.038951e-06  2.464230e-02  0.000000e+00  0.000000e+00
4.124145e+02  9.907665e+02  1.217272e-03  2.393460e-01

```

```

2      0  cg_i -9.419876e-06  1.237374e-02  0.000000e+00  0.000000e+00
4.019071e+02  9.936897e+02  1.254603e-03  2.521387e-01

```

```

iter inewton type    objective    â gâ â_Mâ »Â¹    leftmost    Î»
Î³      Î´      Î±      Î²
0      0  cg_i -8.123315e-08  6.017427e-03  0.000000e+00  0.000000e+00
1.237352e-02  9.423732e+02  1.061151e-03  2.365023e-01

```

```

1      0  cg_i -1.141577e-07  5.764425e-03  0.000000e+00  0.000000e+00
4.582903e+02  7.727586e+02  1.818562e-03  9.176781e-01

```

```

2      0  cg_i -1.714596e-07  9.206608e-03  0.000000e+00  0.000000e+00

```

5.267653e+02 7.945608e+02 3.448951e-03 2.550864e+00

3 0 cg\_i -3.626456e-07 1.409674e-02 0.000000e+00 0.000000e+00  
4.630807e+02 9.612789e+02 4.511145e-03 2.344434e+00

iter	inewton	type	objective	$\hat{\alpha}$	$\hat{g}$	$\hat{\alpha}_M \gg \hat{\alpha}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$			$\hat{I}'$	$\hat{I}_{\pm}$	$\hat{I}^2$			
0	0	cg_i	-7.032046e-09	1.474521e-03	0.000000e+00	0.000000e+00		
3.727647e-03			9.880023e+02	1.012143e-03	1.564705e-01			

1 0 cg\_i -8.304899e-09 6.906989e-04 0.000000e+00 0.000000e+00  
3.908177e+02 1.008664e+03 1.170863e-03 2.194196e-01

2 0 cg\_i -8.600911e-09 3.514945e-04 0.000000e+00 0.000000e+00  
4.000658e+02 9.932200e+02 1.240972e-03 2.589760e-01

3 0 cg\_i -8.677805e-09 2.907239e-05 0.000000e+00 0.000000e+00  
4.100794e+02 1.012058e+03 1.244756e-03 6.841074e-03

iter	inewton	type	objective	$\hat{\alpha}$	$\hat{g}$	$\hat{\alpha}_M \gg \hat{\alpha}^1$	leftmost	$\hat{I} \gg$
$\hat{I}^3$			$\hat{I}'$	$\hat{I}_{\pm}$	$\hat{I}^2$			
0	0	cg_i	-1.065816e-17	7.061929e-08	0.000000e+00	0.000000e+00		
1.369755e-07			8.801844e+02	1.136126e-03	2.658036e-01			

1 0 cg\_i -1.488030e-17 6.089564e-08 0.000000e+00 0.000000e+00  
4.537891e+02 8.245442e+02 1.693228e-03 7.435764e-01

2 0 cg\_i -2.014909e-17 8.298704e-08 0.000000e+00 0.000000e+00  
5.092692e+02 7.910571e+02 2.841638e-03 1.857155e+00

3 0 cg\_i -3.327619e-17 1.245258e-07 0.000000e+00 0.000000e+00  
4.795737e+02 9.158651e+02 3.812224e-03 2.251637e+00

Optimization terminated successfully.

Current function value: 0.000000

Iterations: 19

Function evaluations: 20

Gradient evaluations: 20

Hessian evaluations: 18

message: Optimization terminated successfully.

success: True

status: 0

fun: 1.1618442019708214e-28

x: [ 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00]

nit: 19

```

jac: [ 4.396e-14  2.109e-14  6.417e-14  3.499e-13 -1.998e-13]
nfev: 20
njev: 20
nhev: 18
hess: [[ 8.020e+02 -4.000e+02 ...  0.000e+00  0.000e+00]
        [-4.000e+02  1.002e+03 ...  0.000e+00  0.000e+00]
        ...
        [ 0.000e+00  0.000e+00 ...  1.002e+03 -4.000e+02]
        [ 0.000e+00  0.000e+00 ... -4.000e+02  2.000e+02]]

```

```

[14]: # 最小值点
      print(res.x)

```

```
[1. 1. 1. 1. 1.]
```

```

[15]: res = minimize(
        rosen, x0, method="trust-exact",
        jac=rosen_der, hess=rosen_hess,
        options={"gtol": 1e-8, "disp": True}
    )
    print(res)

```

Optimization terminated successfully.

Current function value: 0.000000

Iterations: 13

Function evaluations: 14

Gradient evaluations: 13

Hessian evaluations: 14

message: Optimization terminated successfully.

success: True

status: 0

fun: 1.534303144849083e-22

x: [ 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00]

nit: 13

jac: [ 7.503e-12 2.589e-11 1.048e-10 4.210e-10 -2.398e-10]

nfev: 14

njev: 13

nhev: 14

hess: [[ 8.020e+02 -4.000e+02 ... 0.000e+00 0.000e+00]

[-4.000e+02 1.002e+03 ... 0.000e+00 0.000e+00]

...

[ 0.000e+00 0.000e+00 ... 1.002e+03 -4.000e+02]

[ 0.000e+00 0.000e+00 ... -4.000e+02 2.000e+02]]

```
[16]: # 最小值点
print(res.x)
```

```
[1. 1. 1. 1. 1.]
```

### 2.0.2 多元函数有约束的最优化问题

求解约束问题如下

$$\min_{x_0, x_1} 100(x_1 - x_0^2)^2 + (1 - x_0)^2$$

subject to:

$$x_0 + 2x_1 \leq 1$$

$$x_0^2 + x_1 \leq 1$$

$$x_0^2 - x_1 \leq 1$$

$$2x_0 + x_1 = 1$$

$$0 \leq x_0 \leq 1$$

$$-0.5 \leq x_1 \leq 2$$

```
[1]: import numpy as np
```

```
[2]: from scipy.optimize import Bounds
# 定义变量的边界
bounds = Bounds(
    [0, -0.5], # 下界
    [1.0, 2.0] # 上界
)
```

```
[3]: from scipy.optimize import LinearConstraint
# 定义线性约束
linear_constraint = LinearConstraint(
    [
        [1, 2],
        [2, 1]
    ],
    [-np.inf, 1], # 下界
    [1, 1] # 上界
)
```

```
[4]: # 非线性约束
def cons_f(x):
    return [
        x[0]**2 + x[1],
        x[0]**2 - x[1]
    ]
# 非线性约束的雅可比形式
```

```

def cons_J(x):
    return [
        [2*x[0], 1],
        [2*x[0], -1]
    ]
# 黑塞矩阵的线性组合
def cons_H(x, v):
    return v[0]*np.array(
        [
            [2, 0],
            [0, 0]
        ]
    ) + v[1]*np.array(
        [
            [2, 0],
            [0, 0]
        ]
    )

from scipy.optimize import NonlinearConstraint
# 构造非线性约束
nonlinear_constraint = NonlinearConstraint(
    cons_f,
    -np.inf, # 下界
    1, # 上界
    jac=cons_J,
    hess=cons_H
)

```

```

[5]: from scipy.optimize import minimize
# 初始值
x0 = np.array([0.5, 0])
# 最优化目标函数
def rosen(x):
    """The Rosenbrock function"""
    return sum(100.0*(x[1:]-x[:-1]**2.0)**2.0 + (1-x[:-1])**2.0)
# 目标函数的导数
def rosen_der(x):
    xm = x[1:-1]
    xm_m1 = x[:-2]
    xm_p1 = x[2:]
    der = np.zeros_like(x)
    der[1:-1] = 200*(xm-xm_m1**2) - 400*(xm_p1 - xm**2)*xm - 2*(1-xm)
    der[0] = -400*x[0]*(x[1]-x[0]**2) - 2*(1-x[0])
    der[-1] = 200*(x[-1]-x[-2]**2)

```

```

    return der
# 目标函数的黑塞矩阵
def rosen_hess(x):
    x = np.asarray(x)
    H = np.diag(-400*x[:-1],1) - np.diag(400*x[:-1],-1)
    diagonal = np.zeros_like(x)
    diagonal[0] = 1200*x[0]**2-400*x[1]+2
    diagonal[-1] = 200
    diagonal[1:-1] = 202 + 1200*x[1:-1]**2 - 400*x[2:]
    H = H + np.diag(diagonal)
    return H

```

```

[6]: res = minimize(
    rosen, # 目标函数
    x0, # 初始值
    method="trust-constr", # 求解算法
    jac=rosen_der, # 梯度
    hess=rosen_hess, # 黑塞矩阵
    constraints=[linear_constraint, nonlinear_constraint], # 约束
    options={"verbose": 1},
    bounds=bounds # 变量范围
)
print(res)

```

```

`gtol` termination condition is satisfied.
Number of iterations: 12, function evaluations: 8, CG iterations: 7, optimality:
2.99e-09, constraint violation: 0.00e+00, execution time: 0.047 s.
    message: `gtol` termination condition is satisfied.
    success: True
    status: 1
        fun: 0.3427175756422305
        x: [ 4.149e-01  1.701e-01]
    nit: 12
    nfev: 8
    njev: 8
    nhev: 8
    cg_niter: 7
    cg_stop_cond: 1
        grad: [-8.265e-01 -4.140e-01]
    lagrangian_grad: [ 1.495e-09 -2.990e-09]
        constr: [array([ 7.552e-01,  1.000e+00]), array([ 3.423e-01,
2.070e-03]), array([ 4.149e-01,  1.701e-01])]
        jac: [array([[ 1.000e+00,  2.000e+00],
[ 2.000e+00,  1.000e+00]]), array([[ 8.299e-01,
1.000e+00],

```

```

            [ 8.299e-01, -1.000e+00]], array([[ 1.000e+00,
0.000e+00],
            [ 0.000e+00,  1.000e+00]]))
    constr_nfev: [0, 8, 0]
    constr_njev: [0, 8, 0]
    constr_nhev: [0, 13, 0]
    v: [array([ 6.536e-04,  4.128e-01]), array([ 2.433e-04,
1.603e-04]), array([-1.121e-04, -1.513e-04])]
    method: tr_interior_point
    optimality: 2.989621706689068e-09
    constr_violation: 0.0
    execution_time: 0.046998023986816406
    tr_radius: 3834.1597660672387
    constr_penalty: 1.0
    barrier_parameter: 0.00016000000000000007
    barrier_tolerance: 0.00016000000000000007
    niter: 12

```

```

[7]: # 最优化的点
      print(res.x)

```

```

[0.41494531 0.17010937]

```

```

[8]: # 定义不等式约束
      ineq_cons = {
          "type": "ineq", # 不等式约束
          "fun" : lambda x: np.array(
              [
                  1 - x[0] - 2*x[1],
                  1 - x[0]**2 - x[1],
                  1 - x[0]**2 + x[1]
              ]
          ),
          "jac" : lambda x: np.array(
              [
                  [-1.0, -2.0],
                  [-2*x[0], -1.0],
                  [-2*x[0], 1.0]
              ]
          )
      }
      # 定义等式约束
      eq_cons = {
          "type": "eq",
          "fun" : lambda x: np.array([2*x[0] + x[1] - 1]),

```



```
"jac" : lambda x: np.array([2.0, 1.0])
}
```

```
[9]: x0 = np.array([0.5, 0])
res = minimize(
    rosen,
    x0,
    method="SLSQP",
    jac=rosen_der,
    constraints=[eq_cons, ineq_cons],
    options={"ftol": 1e-9, "disp": True},
    bounds=bounds
)
print(res)
```

```
Optimization terminated successfully    (Exit mode 0)
      Current function value: 0.34271757499419825
      Iterations: 4
      Function evaluations: 5
      Gradient evaluations: 4
message: Optimization terminated successfully
success: True
status: 0
      fun: 0.34271757499419825
       x: [ 4.149e-01  1.701e-01]
      nit: 4
     jac: [-8.268e-01 -4.137e-01]
    nfev: 5
    njev: 4
```

```
[10]: # 最优化的点
print(res.x)
```

```
[0.41494475 0.1701105 ]
```

### 2.0.3 带约束的最小二乘优化问题

这种问题在 Python 中计算的一般形式如下

$$\min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^n \rho(f_i(\mathbf{x})^2) \text{ subject to: } \mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub}$$

其中的  $f_i(\mathbf{x})$  表示残差函数，一般是  $y_i - \hat{y}_i$ 。

这里假设  $f_i(\mathbf{x})$  的形式如下

$$f_i(\mathbf{x}) = \frac{x_0(u_i^2 + u_i x_1)}{u_i^2 + u_i x_2 + x_3} - y_i, \quad i = 1, 2, \dots, 11$$

这里的  $x_1, x_2, x_3, x_0$  是要求解的参数，其中的  $u_i, y_i$  是已知的数据。

```
[1]: import numpy as np
      from scipy.optimize import least_squares

[2]: # 定义模型
      def model(x, u):
          return x[0] * (u ** 2 + x[1] * u) / (u ** 2 + x[2] * u + x[3])
      # 定义损失函数  $f_i(x)$ 
      def fun(x, u, y):
          return model(x, u) - y
      # 损失函数的雅可比矩阵
      def jac(x, u, y):
          J = np.empty((u.size, x.size))
          den = u ** 2 + x[2] * u + x[3]
          num = u ** 2 + x[1] * u
          J[:, 0] = num / den
          J[:, 1] = x[0] * u / den
          J[:, 2] = -x[0] * num * u / den ** 2
          J[:, 3] = -x[0] * num / den ** 2
          return J
```

```
[3]: # 数据样本
      u = np.array(
          [
              4.0, 2.0, 1.0, 5.0e-1, 2.5e-1, 1.67e-1, 1.25e-1,
              1.0e-1, 8.33e-2, 7.14e-2, 6.25e-2
          ]
      )
      y = np.array(
          [
              1.957e-1, 1.947e-1, 1.735e-1, 1.6e-1, 8.44e-2, 6.27e-2,
              4.56e-2, 3.42e-2, 3.23e-2, 2.35e-2, 2.46e-2
          ]
      )
      # 初始值
      x0 = np.array([2.5, 3.9, 4.15, 3.9])
      res = least_squares(
          fun, # 损失函数  $f_i(x)$ 
          x0, # 初始值
          jac=jac, # 雅可比矩阵
          bounds=(0, 100), # 变量范围
          args=(u, y), #  $fun$  中的参数
          verbose=1
      )
      print(res)
```

`ftol` termination condition is satisfied.

Function evaluations 131, initial cost 4.4383e+00, final cost 1.5375e-04, first-order optimality 4.52e-08.

```

message: `ftol` termination condition is satisfied.
success: True
status: 2
  fun: [-1.307e-03 -1.876e-03  8.920e-03 -1.111e-02  8.352e-03
        -1.737e-04  2.584e-05  1.263e-03 -3.524e-03  6.168e-04
        -3.889e-03]
    x: [ 1.928e-01  1.913e-01  1.231e-01  1.361e-01]
  cost: 0.00015375280234150847
   jac: [[ 1.008e+00  4.638e-02 -4.676e-02 -1.169e-02]
          [ 1.000e+00  8.800e-02 -8.800e-02 -4.400e-02]
          ...
          [ 1.251e-01  9.180e-02 -1.148e-02 -1.608e-01]
          [ 1.074e-01  8.160e-02 -8.766e-03 -1.403e-01]]
  grad: [-4.526e-10 -1.445e-10  1.552e-09  8.542e-08]
optimality: 4.516919346939224e-08
active_mask: [0 0 0 0]
   nfev: 131
   njev: 104

```

```

[4]: # 最优点
      print(res.x)

```

```

[0.192806  0.19130332 0.12306046 0.13607205]

```

#### 2.0.4 一元函数无约束最优化问题

```

[1]: from scipy.optimize import minimize_scalar
      # 定义函数
      f = lambda x: (x - 2) * (x + 1)**2
      res = minimize_scalar(f, method="brent")
      print(res)

```

```

message:
    Optimization terminated successfully;
    The returned value satisfies the termination criteria
    (using xtol = 1.48e-08 )
success: True
  fun: -4.0
   x: 1.0
  nit: 7
 nfev: 10

```

```
[2]: # 最优解
      print(res.x)
```

```
1.0
```

### 2.0.5 一元函数有界约束最优化问题

```
[1]: from scipy.optimize import minimize_scalar
      # 定义函数
      f = lambda x: (x - 2) * (x + 1)**2
      res = minimize_scalar(f, bounds=(1,10), method="bounded")
      print(res)
```

```
message: Solution found.
success: True
status: 0
      fun: -3.999999999953027
       x: 1.0000039570068944
      nit: 20
     nfev: 20
```

```
[2]: # 最优解
      print(res.x)
```

```
1.0000039570068944
```

### 2.0.6 一元函数求根

```
[1]: from scipy import optimize
      # 定义目标函数
      def f(x):
          return (x**3 - 1) # only one real root at x = 1
      # 导数
      def fprime(x):
          return 3*x**2
```

```
[2]: sol = optimize.root_scalar(
      f, # 目标函数
      bracket=[0, 3], # 寻根区间
      method="brentq"
      )
      print(sol.root, sol.iterations, sol.function_calls)
```

```
1.0 10 11
```

```
[3]: sol = optimize.root_scalar(
    f,
    x0=0.2, # 初始值
    fprime=fprime, # 导数
    method="newton"
)
print(sol.root, sol.iterations, sol.function_calls)
```

1.0 11 22

### 2.0.7 线性规划

线性规划的数学表达如下

$$\begin{aligned} & \min_x c^T x \\ & \text{subject to:} \\ & A_{ub}x \leq b_{ub} \\ & A_{eq}x \leq b_{eq} \\ & l \leq x \leq u \end{aligned}$$

下面求解一个实际例子

$$\begin{aligned} & \max_{x_1, x_2, x_3, x_4} 29x_1 + 45x_2 \\ & x_1 - x_2 - 3x_3 \leq 5 \\ & 2x_1 - 3x_2 - 7x_3 + 3x_4 \geq 10 \\ & 2x_1 + 8x_2 + x_3 = 60 \\ & 4x_1 + 4x_2 + x_4 = 60 \\ & 0 \leq x_1 \\ & 0 \leq x_2 \leq 5 \\ & x_3 \leq 0.5 \\ & -3 \leq x_4 \end{aligned}$$

```
[1]: import numpy as np
from scipy.optimize import linprog
# 价值系数，最大值需要将 c 的系数变号
c = np.array([-29.0, -45.0, 0.0, 0.0])
A_ub = np.array(
    [
        [1.0, -1.0, -3.0, 0.0],
        [-2.0, 3.0, 7.0, -3.0]
    ]
)
b_ub = np.array([5.0, -10.0])
A_eq = np.array(
    [
```

```

        [2.0, 8.0, 1.0, 0.0],
        [4.0, 4.0, 0.0, 1.0]
    ]
)
b_eq = np.array([60.0, 60.0])
# 变量范围
x0_bounds = (0, None)
x1_bounds = (0, 5.0)
x2_bounds = (-np.inf, 0.5) # +/- np.inf can be used instead of None
x3_bounds = (-3.0, None)
# 构造变量界
bounds = [x0_bounds, x1_bounds, x2_bounds, x3_bounds]
# 线性规划
result = linprog(
    c,
    A_ub=A_ub,
    b_ub=b_ub,
    A_eq=A_eq,
    b_eq=b_eq,
    bounds=bounds
)
print(result)

```

```

message: The problem is infeasible. (HiGHS Status 8: model_status is
Infeasible; primal_status is At lower/fixed bound)
success: False
status: 2
fun: None
x: None
nit: 3
lower: residual: None
      marginals: None
upper: residual: None
      marginals: None
eqlin: residual: None
      marginals: None
ineqlin: residual: None
      marginals: None

```

这个线性规划无最优解，改变一下条件

```

[2]: x1_bounds = (0, 6)
      bounds = [x0_bounds, x1_bounds, x2_bounds, x3_bounds]
      result = linprog(c, A_ub=A_ub, b_ub=b_ub, A_eq=A_eq, b_eq=b_eq, bounds=bounds)
      print(result)

```

```
message: Optimization terminated successfully. (HiGHS Status 7: Optimal)
success: True
status: 0
  fun: -505.974358974359
    x: [ 9.410e+00  5.179e+00 -2.564e-01  1.641e+00]
  nit: 3
lower: residual: [ 9.410e+00  5.179e+00          inf  4.641e+00]
      marginals: [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00]
upper: residual: [          inf  8.205e-01  7.564e-01          inf]
      marginals: [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00]
eqlin: residual: [ 0.000e+00  0.000e+00]
      marginals: [-2.887e+00 -5.415e+00]
ineqlin: residual: [ 0.000e+00  0.000e+00]
      marginals: [-5.174e+00 -1.805e+00]
mip_node_count: 0
mip_dual_bound: 0.0
  mip_gap: 0.0
```

```
[3]: # 最优点
      print(result.x)
      # 最优值
      print(result.fun)
```

```
[ 9.41025641  5.17948718 -0.25641026  1.64102564]
-505.974358974359
```