بسمه تعالی

دانشکده مهندسی مکانیک

دانشکدگان فنی

دانشگاه تهران

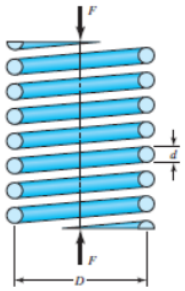
بهینه سازی سیستم‌های مکانیکی

تکلیف شماره چهار

محمد مهدی خجسته ۸۱۰۶۹۷۲۸۰

استاد: دکتر شریعت پناهی

بهار ۱۴۰۲



جدول ۱- پارامترهای طراحی

نماد	اطلاعات
D (mm)	قطر میانگین حلقه‌ها
d (mm)	قطر سیم فنر
N	تعداد حلقه‌های فعال=تعداد کل حلقه‌ها

جدول ۲- کمیت‌های مورد نیاز

مقدار	کمیت
$g = 9.81\ (\mathrm{m/s^2})$	ثابت گرانش
$\gamma = 77500\ (\mathrm{N/m^3})$	چگالی وزنی فولاد فنر
$G = 79.3\ (\mathrm{GPa})$	مدول برشی فولاد فنر
$F = 80\ (\mathrm{N})$	نیروی استاتیکی وارد به فنر

جدول ۳- روابط مورد نیاز

رابطه	کمیت
$W = \frac{\pi^2 d^2 D N \gamma}{4}$	وزن فنر
$\tau = 1.1\frac{8FD}{\pi d^3}$	تنش برشی بیشینه در فنر
$\delta = \frac{8FD^3N}{d^4G}$	تغییر طول فنر

$$\tau_{\max} = \tau_{torsion} + \tau_{pure\ shear} = \frac{8PD}{\pi d^3} + \frac{4P}{\pi d^2} < \tau_a$$

$$k = \frac{Dd^4}{8D^3N}$$

$$\delta = \frac{P}{k} = \frac{8PD^3N}{Gd^4} \geq \Delta$$

$$W = AL_\gamma = \left(\frac{\pi}{4}d^2\right)(\pi DN)(\gamma)$$

$$f = \frac{1}{2}\sqrt{\frac{kg}{W}} \geq \omega_0$$
$$d + D \leq D_0$$

Parameter search range

$5 \leq N \leq 15$

$10 \leq D_0 \leq 40$

$1.5 \leq d \leq 6.5$

The Cost Function

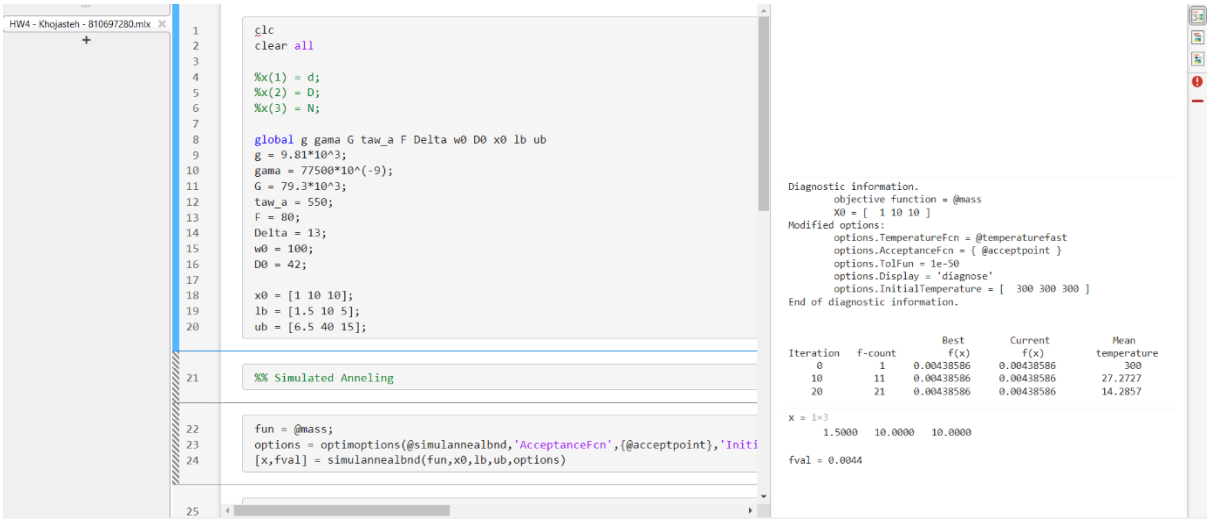
$$Mass(d,\ D,\ N) = \rho\left(\frac{\pi}{4}d^2\right)(\pi DN)$$

هدف این تکلیف طراحی یک فنر مارپیچ فشاری برای تحمل نیروی استاتیکی محوری یاد شده F با کم‌ترین جرم می‌باشد (خواسته شده که جرم فنری را برای بارگذاری تحت نیروی استاتیکی بهینه کنیم) که با توجه به روابط فنرها و قیدهایی که داریم بایستی اینکار را انجام دهیم در ابتدا به انجام این کار در نرم افزار متلب می‌پردازیم و سپس در پایتون نحوه انجام این کار را بررسی می‌کنیم.

ادامه پاسخ تمرین در صفحه بعد...

Matlab

در نرم افزار متلب برای این که قیدها را اعمال کنیم و ناحیه مجاز مشخص شود و نقاطی که در الگوریتم قرار می‌گیرند در ناحیه مجاز باشند از تابع AcceptFcn دستور simulannealbnd استفاده می‌کنیم و تابع قیود را داخل آن اعمال می‌کنیم (شایان ذکر است که برای حد پایین فرکانس طبیعی فنر نیز مقداری دلخواه را به مسئله اضافه شده است). در ادامه صورت کلی کد نوشته شده آمده است.



کد کامل مربوط به این بخش در پیوست این تمرین قدیم شده است.

در کد یاد شده تلاش شده با کامنت گذاری مشخص شود که هر قسمت مربوط به چه کاری می‌باشد.

در ادامه به بررسی تاثیر نقاط اولیه، دمای اولیه و تابع به روزآوری دما بر عملکرد این دستور می‌پردازیم.

ادامه پاسخ تمرین در صفحه بعد...

- X_0 = [1.5 10 10]
- T_0 = [300 300 300]
- temperature function = 'temperatureexp'

نتایج دریافتی این حالت به صورت زیر می‌باشد:

```
%% Simulated Anneling
fun = @mass;
options = optimoptions(@simulannealbnd,'AcceptanceFcn',{@acceptpoint},'InitialTemperature', [300 300 300],...
    'TemperatureFcn','temperatureexp','Display','diagnose','TolFun',1e-50);
[x,fval] = simulannealbnd(fun,x0,lb,ub,options)
```

```
Diagnostic information.
        objective function = @mass
        X0 = [  1.5 10 10 ]
Modified options:
        options.TemperatureFcn = @temperatureexp
        options.AcceptanceFcn = { @acceptpoint }
        options.TolFun = 1e-50
        options.Display = 'diagnose'
        options.InitialTemperature = [  300 300 300 ]
End of diagnostic information.


                          Best        Current          Mean
Iteration   f-count         f(x)         f(x)     temperature
     0          1     0.00438586   0.00438586           300
    10         11     0.00438586   0.00438586        170.64
    20         21     0.00438586   0.00438586       102.168
    30         31     0.00438586   0.00438586        61.172
 x = 1×3
       1.5000   10.0000   10.0000

fval = 0.0044
```

- X_0 = [5 25 4]
- T_0 = [300 300 300]
- temperature function = 'temperatureexp'

نتایج دریافتی این حالت به صورت زیر می‌باشد:

```
%% Simulated Anneling
fun = @mass;
options = optimoptions(@simulannealbnd,'AcceptanceFcn',{@acceptpoint},'InitialTemperature', [300 300 300],...
    'TemperatureFcn','temperatureexp','Display','diagnose','TolFun',1e-50);
[x,fval] = simulannealbnd(fun,x0,lb,ub,options)
```

```
Diagnostic information.
        objective function = @mass
        X0 = [  5 25 4 ]
Modified options:
        options.TemperatureFcn = @temperatureexp
        options.AcceptanceFcn = { @acceptpoint }
        options.TolFun = 1e-50
        options.Display = 'diagnose'
        options.InitialTemperature = [  300 300 300 ]
End of diagnostic information.


                          Best        Current          Mean
Iteration   f-count         f(x)         f(x)     temperature
     0          1     0.0609148    0.0609148           300
    10         11     0.0609148    0.0609148        170.64
    20         21     0.0247741    0.0247741       102.168
    30         31     0.0212144    0.0212144        61.172
 x = 1×3
       1.7805   12.6713    8.4040

fval = 0.0066
```

- X_0 = [6 15 10]
- T_0 = [300 300 300]
- temperature function = ‘temperatureexp’

نتایج دریافتی این حالت به صورت زیر می‌باشد:

```
%% Simulated Anneling
fun = @mass;
options = optimoptions(@simulannealbnd,'AcceptanceFcn',{@acceptpoint},'InitialTemperature', [300 300 300],...
    'TemperatureFcn','temperatureexp','Display','diagnose','TolFun',1e-50);
[x,fval] = simulannealbnd(fun,x0,lb,ub,options)
```

```
Diagnostic information.
        objective function = @mass
        X0 = [  6 15 10 ]
Modified options:
        options.TemperatureFcn = @temperatureexp
        options.AcceptanceFcn = { @acceptpoint }
        options.TolFun = 1e-50
        options.Display = 'diagnose'
        options.InitialTemperature = [  300 300 300 ]
End of diagnostic information.


                          Best        Current          Mean
Iteration    f-count        f(x)         f(x)        temperature
     0           1       0.105261     0.105261           300
    10          11       0.105261     0.105261         170.64
    20          21      0.0247984    0.0247984         102.168
    30          31      0.0214407    0.0214407         61.172
 x = 1×3
       1.7351   11.6167    9.8789

fval = 0.0067
```

- X_0 = [1.5 10 10]
- T_0 = [200 200 200]
- temperature function = ‘temperatureexp’

نتایج دریافتی این حالت به صورت زیر می‌باشد:

```
%% Simulated Anneling
fun = @mass;
options = optimoptions(@simulannealbnd,'AcceptanceFcn',{@acceptpoint},'InitialTemperature', [200 200 200],...
    'TemperatureFcn','temperatureexp','Display','diagnose','TolFun',1e-50);
[x,fval] = simulannealbnd(fun,x0,lb,ub,options)
```

```
Diagnostic information.
        objective function = @mass
        X0 = [  1.5 10 10 ]
Modified options:
        options.TemperatureFcn = @temperatureexp
        options.AcceptanceFcn = { @acceptpoint }
        options.TolFun = 1e-50
        options.Display = 'diagnose'
        options.InitialTemperature = [  200 200 200 ]
End of diagnostic information.


                          Best        Current          Mean
Iteration    f-count        f(x)         f(x)        temperature
     0           1     0.00438586   0.00438586           200
    10          11     0.00438586   0.00438586         113.76
    20          21     0.00438586   0.00438586         68.1123
    30          31     0.00438586   0.00438586         40.7814
 x = 1×3
       1.5000   10.0000   10.0000

fval = 0.0044
```

- X_0 = [1.5 10 10]
- T_0 = [1000 1000 1000]
- temperature function = ‘temperatureexp’

نتایج دریافتی این حالت به صورت زیر می‌باشد:

```
%% Simulated Anneling
fun = @mass;
options = optimoptions(@simulannealbnd,'AcceptanceFcn',{@acceptpoint},'InitialTemperature', [1000 1000 1000],...
    'TemperatureFcn','temperatureexp','Display','diagnose','TolFun',1e-50);
[x,fval] = simulannealbnd(fun,x0,lb,ub,options)
```

```
Diagnostic information.
        objective function = @mass
        X0 = [  1.5 10 10 ]
Modified options:
        options.TemperatureFcn = @temperatureexp
        options.AcceptanceFcn = { @acceptpoint }
        options.TolFun = 1e-50
        options.Display = 'diagnose'
        options.InitialTemperature = [  1000 1000 1000 ]
End of diagnostic information.


                          Best        Current          Mean
Iteration   f-count         f(x)         f(x)        temperature
     0          1     0.00438586     0.00438586          1000
    10         11     0.00438586     0.00438586         568.8
    20         21     0.00438586     0.00438586       340.562
    30         31     0.00438586     0.00438586       203.907
 x = 1×3
       1.5000   10.0000   10.0000

fval = 0.0044
```

- X_0 = [1.5 10 10]
- T_0 = [50 50 50]
- temperature function = ‘temperatureexp’

نتایج دریافتی این حالت به صورت زیر می‌باشد:

```
%% Simulated Anneling
fun = @mass;
options = optimoptions(@simulannealbnd,'AcceptanceFcn',{@acceptpoint},'InitialTemperature', [50 50 50],...
    'TemperatureFcn','temperatureexp','Display','diagnose','TolFun',1e-50);
[x,fval] = simulannealbnd(fun,x0,lb,ub,options)
```

```
Diagnostic information.
        objective function = @mass
        X0 = [  1.5 10 10 ]
Modified options:
        options.TemperatureFcn = @temperatureexp
        options.AcceptanceFcn = { @acceptpoint }
        options.TolFun = 1e-50
        options.Display = 'diagnose'
        options.InitialTemperature = [  50 50 50 ]
End of diagnostic information.


                          Best        Current          Mean
Iteration   f-count         f(x)         f(x)        temperature
     0          1     0.00438586     0.00438586           50
    10         11     0.00438586     0.00438586         28.44
    20         21     0.00438586     0.00438586       17.0281
    30         31     0.00438586     0.00438586       10.1953
 x = 1×3
       1.5000   10.0000   10.0000

fval = 0.0044
```

- X_0 = [1.5 10 10]
- T_0 = [500 500 500]
- temperature function = ‘temperatureexp’

نتایج دریافتی این حالت به صورت زیر می‌باشد:

```
%% Simulated Anneling
fun = @mass;
options = optimoptions(@simulannealbnd,'AcceptanceFcn',{@acceptpoint},'InitialTemperature', [500 500 500],...
    'TemperatureFcn','temperatureexp','Display','diagnose','TolFun',1e-50);
[x,fval] = simulannealbnd(fun,x0,lb,ub,options)
```

```
Diagnostic information.
        objective function = @mass
        X0 = [  1.5 10 10 ]
Modified options:
        options.TemperatureFcn = @temperatureexp
        options.AcceptanceFcn = { @acceptpoint }
        options.TolFun = 1e-50
        options.Display = 'diagnose'
        options.InitialTemperature = [  500 500 500 ]
End of diagnostic information.


                          Best        Current          Mean
Iteration    f-count        f(x)         f(x)        temperature
     0           1     0.00438586     0.00438586           500
    10          11     0.00438586     0.00438586         284.4
    20          21     0.00438586     0.00438586       170.281
    30          31     0.00438586     0.00438586       101.953
 x = 1×3
       1.5000   10.0000   10.0000

fval = 0.0044
```

- X_0 = [1.5 10 10]
- T_0 = [500 500 500]
- temperature function = ‘temperaturefast’

نتایج دریافتی این حالت به صورت زیر می‌باشد:

```
%% Simulated Anneling
fun = @mass;
options = optimoptions(@simulannealbnd,'AcceptanceFcn',{@acceptpoint},'InitialTemperature', [500 500 500],...
    'TemperatureFcn','temperaturefast','Display','diagnose','TolFun',1e-50);
[x,fval] = simulannealbnd(fun,x0,lb,ub,options)
```

```
Diagnostic information.
        objective function = @mass
        X0 = [  1.5 10 10 ]
Modified options:
        options.TemperatureFcn = @temperaturefast
        options.AcceptanceFcn = { @acceptpoint }
        options.TolFun = 1e-50
        options.Display = 'diagnose'
        options.InitialTemperature = [  500 500 500 ]
End of diagnostic information.


                          Best        Current          Mean
Iteration    f-count        f(x)         f(x)        temperature
     0           1     0.00438586     0.00438586           500
    10          11     0.00438586     0.00438586       45.4545
    20          21     0.00438586     0.00438586       23.8095
    30          31     0.00438586     0.00438586       16.129
 x = 1×3
       1.5000   10.0000   10.0000

fval = 0.0044
```

- X_0 = [1.5 10 10]
- T_0 = [500 500 500]
- temperature function = ‘temperatureboltz’

نتایج دریافتی این حالت به صورت زیر می‌باشد:

```
%% Simulated Anneling
fun = @mass;
options = optimoptions(@simulannealbnd,'AcceptanceFcn',{@acceptpoint},'InitialTemperature', [500 500 500],...
    'TemperatureFcn','temperatureboltz','Display','diagnose','TolFun',1e-50);
[x,fval] = simulannealbnd(fun,x0,lb,ub,options)
```

```
Diagnostic information.
        objective function = @mass
        X0 = [  1.5 10 10 ]
Modified options:
        options.TemperatureFcn = @temperatureboltz
        options.AcceptanceFcn = { @acceptpoint }
        options.TolFun = 1e-50
        options.Display = 'diagnose'
        options.InitialTemperature = [  500 500 500 ]
End of diagnostic information.


                            Best        Current          Mean
Iteration   f-count          f(x)         f(x)         temperature
     0          1     0.00438586     0.00438586           500
    10         11     0.00438586     0.00438586         208.516
    20         21     0.00438586     0.00438586         164.229
    30         31     0.00438586     0.00438586         145.603
 x = 1×3
       1.5000   10.0000   10.0000

fval = 0.0044
```

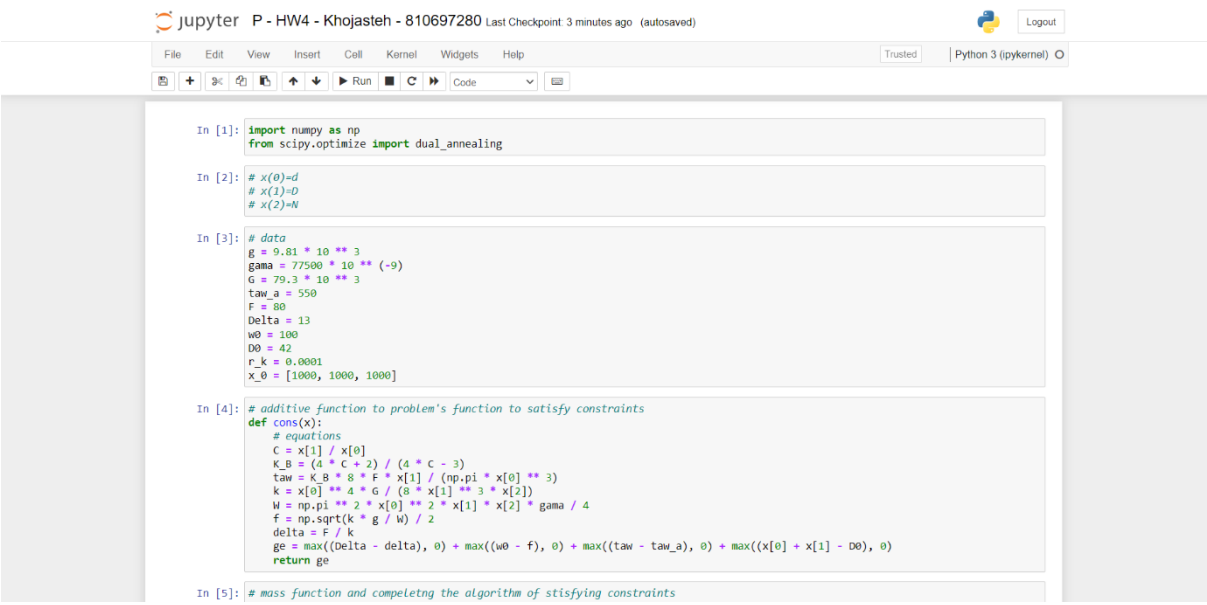
نکته قابل توجه در مورد این دستور در متلب این است که این دستور بعد از یافتن تعدادی نقطه قابل قبول، دما را به دمای اولیه باز می‌گرداند که این موضوع تحت عنوان ReaneelInterval در الگوریتم موجود است.

در متلب، بدیهی است که با انتخاب یک نقطه اولیه دور، الگوریتم تکرارهای بیشتری را انجام می‌دهد، اما مقادیر تابع هدف در هر سه نقطه اولیه مختلف، با دقت خوبی به یکدیگر نزدیک هستند. وقتی مقدار زیادی را برای دمای اولیه انتخاب می کنیم، مقدار نهایی جرم کوچکتر است. بنابراین، الگوریتم به دمای اولیه بسیار حساس است.

ادامه پاسخ تمرین در صفحه بعد...

Python

در پایتون برخلاف متلب تابعی برای یافتن نقاط مجاز که قیدها را ارضا کنند نداریم لذا برای این که هم تابع جرم کمینه شود و هم قیدها ارضا شوند از روش‌های تابع جریمه خارجی استفاده می‌کنیم. در ادامه صورت کلی کد نوشته شده آمده است.



کد کامل مربوط به این بخش در پیوست این تمرین قدیم شده است.

مطابق صورت سوال همان شش حالت قبلی ذکر شده در قسمت متلب را برای مقایسه انتخاب می‌کنیم.

ادامه پاسخ تمرین در صفحه بعد...

- X_0 = [1.5 10 10]
- T_0 = [300 300 300]

```
w0 = 100
D0 = 42
r_k = 0.0001
x_0 = [1.5, 10, 10]
```

In [8]:
```
# additive function to problem's function to satisfy constraints
def cons(x):
    # equations
    C = x[1] / x[0]
    K_B = (4 * C + 2) / (4 * C - 3)
    taw = K_B * 8 * F * x[1] / (np.pi * x[0] ** 3)
    k = x[0] ** 4 * G / (8 * x[1] ** 3 * x[2])
    W = np.pi ** 2 * x[0] ** 2 * x[1] * x[2] * gama / 4
    f = np.sqrt(k * g / W) / 2
    delta = F / k
    ge = max((Delta - delta), 0) + max((w0 - f), 0) + max((taw - taw_a), 0) + max((x[0] + x[1] - D0), 0)
    return ge
```

In [9]:
```
# mass function and compeletng the algorithm of stisfying constraints
def mass(x):
    global r_k
    W = np.pi ** 2 * x[0] ** 2 * x[1] * x[2] * gama / 4
    r_k = 10 * r_k;
    return W / 9.81 + r_k * cons(x)
```

In [10]:
```
# smiulated annealing
bounds = [[1.5, 6.5], [10, 40], [5, 15]]
dual_annealing(mass, bounds, args=(), maxiter=1000, initial_temp=300, restart_temp_ratio=2e-05,
               visit=2.62, accept=- 5.0, maxfun=10000000.0, seed=None, no_local_search=False, callback=None, x0=x_0)
```

```
F:\Users\Auspicious\anaconda3\lib\site-packages\scipy\optimize\_dual_annealing.py:267: RuntimeWarning: overflow encountered in
double_scalars
  pqv_temp = 1.0 - ((1.0 - self.acceptance_param) *
C:\Users\Auspicious\AppData\Local\Temp\ipykernel_13332\3857302247.py:6: RuntimeWarning: overflow encountered in double_scalars
  return W / 9.81 + r_k * cons(x)
```

Out[10]:
```
 message: ['Maximum number of iteration reached']
 success: True
  status: 0
     fun: 0.007880634308500428
       x: [ 1.979e+00  1.607e+01  6.423e+00]
     nit: 1000
    nfev: 6065
    njev: 16
    nhev: 0
```

ادامه پاسخ تمرین در صفحه بعد...

- X_0 = [5 25 4]
- T_0 = [300 300 300]

```
r_k = 0.0001
x_0 = [5, 25, 4]
```

In [12]:
```
# additive function to problem's function to satisfy constraints
def cons(x):
    # equations
    C = x[1] / x[0]
    K_B = (4 * C + 2) / (4 * C - 3)
    taw = K_B * 8 * F * x[1] / (np.pi * x[0] ** 3)
    k = x[0] ** 4 * G / (8 * x[1] ** 3 * x[2])
    W = np.pi ** 2 * x[0] ** 2 * x[1] * x[2] * gama / 4
    f = np.sqrt(k * g / W) / 2
    delta = F / k
    ge = max((Delta - delta), 0) + max((w0 - f), 0) + max((taw - taw_a), 0) + max((x[0] + x[1] - D0), 0)
    return ge
```

In [13]:
```
# mass function and compeletng the algorithm of stisfying constraints
def mass(x):
    global r_k
    W = np.pi ** 2 * x[0] ** 2 * x[1] * x[2] * gama / 4
    r_k = 10 * r_k;
    return W / 9.81 + r_k * cons(x)
```

In [14]:
```
# smiulated annealing
bounds = [[1.5, 6.5], [10, 40], [5, 15]]
dual_annealing(mass, bounds, args=(), maxiter=1000, initial_temp=300, restart_temp_ratio=2e-05,
               visit=2.62, accept=- 5.0, maxfun=10000000.0, seed=None, no_local_search=False, callback=None, x0=x_0)
```

```
C:\Users\Auspicious\AppData\Local\Temp\ipykernel_13332\3857302247.py:6: RuntimeWarning: overflow encountered in double_scalars
  return W / 9.81 + r_k * cons(x)
```

Out[14]:
```
 message: ['Maximum number of iteration reached']
 success: True
  status: 0
     fun: 0.010459893898266338
       x: [ 2.043e+00  2.022e+01  6.358e+00]
     nit: 1000
    nfev: 6093
    njev: 23
    nhev: 0
```

In [ ]:

ادامه پاسخ تمرین در صفحه بعد...

- X_0 = [6 15 10]
- T_0 = [300 300 300]

```
D0 = 42
r_k = 0.0001
x_0 = [6, 15, 10]
```

In [16]:
```
# additive function to problem's function to satisfy constraints
def cons(x):
    # equations
    C = x[1] / x[0]
    K_B = (4 * C + 2) / (4 * C - 3)
    taw = K_B * 8 * F * x[1] / (np.pi * x[0] ** 3)
    k = x[0] ** 4 * G / (8 * x[1] ** 3 * x[2])
    W = np.pi ** 2 * x[0] ** 2 * x[1] * x[2] * gama / 4
    f = np.sqrt(k * g / W) / 2
    delta = F / k
    ge = max((Delta - delta), 0) + max((w0 - f), 0) + max((taw - taw_a), 0) + max((x[0] + x[1] - D0), 0)
    return ge
```

In [17]:
```
# mass function and compeletng the algorithm of stisfying constraints
def mass(x):
    global r_k
    W = np.pi ** 2 * x[0] ** 2 * x[1] * x[2] * gama / 4
    r_k = 10 * r_k;
    return W / 9.81 + r_k * cons(x)
```

In [18]:
```
# smiulated annealing
bounds = [[1.5, 6.5], [10, 40], [5, 15]]
dual_annealing(mass, bounds, args=(), maxiter=1000, initial_temp=300, restart_temp_ratio=2e-05,
               visit=2.62, accept=- 5.0, maxfun=10000000.0, seed=None, no_local_search=False, callback=None, x0=x_0)
```

```
C:\Users\Auspicious\AppData\Local\Temp\ipykernel_13332\3857302247.py:6: RuntimeWarning: overflow encountered in double_scalars
  return W / 9.81 + r_k * cons(x)
```

Out[18]:
```
 message: ['Maximum number of iteration reached']
 success: True
  status: 0
     fun: 0.009393562725995494
       x: [ 2.163e+00  2.017e+01  5.107e+00]
     nit: 1000
    nfev: 6069
    njev: 17
    nhev: 0
```

In [ ]:

ادامه پاسخ تمرین در صفحه بعد...

- X_0 = [1.5 10 10]
- T_0 = [200 200 200]

```
r_k = 0.0001
x_0 = [1.5, 10, 10]
```

In [20]:
```
# additive function to problem's function to satisfy constraints
def cons(x):
    # equations
    C = x[1] / x[0]
    K_B = (4 * C + 2) / (4 * C - 3)
    taw = K_B * 8 * F * x[1] / (np.pi * x[0] ** 3)
    k = x[0] ** 4 * G / (8 * x[1] ** 3 * x[2])
    W = np.pi ** 2 * x[0] ** 2 * x[1] * x[2] * gama / 4
    f = np.sqrt(k * g / W) / 2
    delta = F / k
    ge = max((Delta - delta), 0) + max((w0 - f), 0) + max((taw - taw_a), 0) + max((x[0] + x[1] - D0), 0)
    return ge
```

In [21]:
```
# mass function and compeletng the algorithm of stisfying constraints
def mass(x):
    global r_k
    W = np.pi ** 2 * x[0] ** 2 * x[1] * x[2] * gama / 4
    r_k = 10 * r_k;
    return W / 9.81 + r_k * cons(x)
```

In [22]:
```
# smiulated annealing
bounds = [[1.5, 6.5], [10, 40], [5, 15]]
dual_annealing(mass, bounds, args=(), maxiter=1000, initial_temp=200, restart_temp_ratio=2e-05,
               visit=2.62, accept=-- 5.0, maxfun=10000000.0, seed=None, no_local_search=False, callback=None, x0=x_0)
```

```
C:\Users\Auspicious\AppData\Local\Temp\ipykernel_13332\3857302247.py:6: RuntimeWarning: overflow encountered in double_scalars
  return W / 9.81 + r_k * cons(x)
```

Out[22]:
```
 message: ['Maximum number of iteration reached']
 success: True
  status: 0
     fun: 0.008757144968948319
       x: [ 2.103e+00  1.880e+01  5.401e+00]
     nit: 1000
    nfev: 6065
    njev: 16
    nhev: 0
```

In [ ]:

ادامه پاسخ تمرین در صفحه بعد..

- X_0 = [1.5 10 10]
- T_0 = [1000 1000 1000]

```
x_0 = [1.5, 10, 10]
```

In [4]:
```
# additive function to problem's function to satisfy constraints
def cons(x):
    # equations
    C = x[1] / x[0]
    K_B = (4 * C + 2) / (4 * C - 3)
    taw = K_B * 8 * F * x[1] / (np.pi * x[0] ** 3)
    k = x[0] ** 4 * G / (8 * x[1] ** 3 * x[2])
    W = np.pi ** 2 * x[0] ** 2 * x[1] * x[2] * gama / 4
    f = np.sqrt(k * g / W) / 2
    delta = F / k
    ge = max((Delta - delta), 0) + max((w0 - f), 0) + max((taw - taw_a), 0) + max((x[0] + x[1] - D0), 0)
    return ge
```

In [5]:
```
# mass function and compeletng the algorithm of stisfying constraints
def mass(x):
    global r_k
    W = np.pi ** 2 * x[0] ** 2 * x[1] * x[2] * gama / 4
    r_k = 10 * r_k;
    return W / 9.81 + r_k * cons(x)
```

In [6]:
```
# smiulated annealing
bounds = [[1.5, 6.5], [10, 40], [5, 15]]
dual_annealing(mass, bounds, args=(), maxiter=1000, initial_temp=1000, restart_temp_ratio=2e-05,
               visit=2.62, accept=-5.0, maxfun=10000000.0, seed=None, no_local_search=False, callback=None, x0=x_0)
```

```
F:\Users\Auspicious\anaconda3\lib\site-packages\scipy\optimize\_dual_annealing.py:267: RuntimeWarning: overflow encountered in
double_scalars
  pqv_temp = 1.0 - ((1.0 - self.acceptance_param) *
C:\Users\Auspicious\AppData\Local\Temp\ipykernel_9496\3857302247.py:6: RuntimeWarning: overflow encountered in double_scalars
  return W / 9.81 + r_k * cons(x)
```

Out[6]:
```
 message: ['Maximum number of iteration reached']
 success: True
  status: 0
     fun: 0.01159118217069571
       x: [ 2.240e+00  2.230e+01  5.313e+00]
     nit: 1000
    nfev: 6049
    njev: 12
    nhev: 0
```

In [ ]:
```
```

ادامه پاسخ تمرین در صفحه بعد..

- X_0 = [1.5 10 10]
- T_0 = [50 50 50]

```
x_0 = [1.5, 10, 10]
```

In [4]:
```
# additive function to problem's function to satisfy constraints
def cons(x):
    # equations
    C = x[1] / x[0]
    K_B = (4 * C + 2) / (4 * C - 3)
    taw = K_B * 8 * F * x[1] / (np.pi * x[0] ** 3)
    k = x[0] ** 4 * G / (8 * x[1] ** 3 * x[2])
    W = np.pi ** 2 * x[0] ** 2 * x[1] * x[2] * gama / 4
    f = np.sqrt(k * g / W) / 2
    delta = F / k
    ge = max((Delta - delta), 0) + max((w0 - f), 0) + max((taw - taw_a), 0) + max((x[0] + x[1] - D0), 0)
    return ge
```

In [5]:
```
# mass function and compeletng the algorithm of stisfying constraints
def mass(x):
    global r_k
    W = np.pi ** 2 * x[0] ** 2 * x[1] * x[2] * gama / 4
    r_k = 10 * r_k;
    return W / 9.81 + r_k * cons(x)
```

In [6]:
```
# smiulated annealing
bounds = [[1.5, 6.5], [10, 40], [5, 15]]
dual_annealing(mass, bounds, args=(), maxiter=1000, initial_temp=50, restart_temp_ratio=2e-05,
               visit=2.62, accept=- 5.0, maxfun=10000000.0, seed=None, no_local_search=False, callback=None, x0=x_0)
```

```
F:\Users\Auspicious\anaconda3\lib\site-packages\scipy\optimize\_numdiff.py:598: RuntimeWarning: overflow encountered in divide
  J_transposed[i] = df / dx
F:\Users\Auspicious\anaconda3\lib\site-packages\scipy\optimize\_dual_annealing.py:267: RuntimeWarning: overflow encountered in
double_scalars
  pqv_temp = 1.0 - ((1.0 - self.acceptance_param) *
C:\Users\Auspicious\AppData\Local\Temp\ipykernel_12940\3857302247.py:6: RuntimeWarning: overflow encountered in double_scalars
  return W / 9.81 + r_k * cons(x)
```

Out[6]:
```
 message: ['Maximum number of iteration reached']
 success: True
  status: 0
     fun: 0.014076483474719613
       x: [ 2.441e+00  2.383e+01  5.087e+00]
     nit: 1000
    nfev: 6081
    njev: 20
    nhev: 0
```

با توجه به کد نوشته شده، می‌توان بیان کرد که در پایتون به دلیل این که دو الگوریتم عددی همزمان با هم اجرا می‌شوند، جواب‌های متفاوت‌تری نسبت به متلب خواهیم گرفت.