بسمه تعالی

دانشکده مهندسی مکانیک

دانشکدگان فنی

دانشگاه تهران

بهینه سازی سیستم‌های مکانیکی

تکلیف شماره سه

محمد مهدی خجسته ۸۱۰۶۹۷۲۸۰

استاد: دکتر شریعت پناهی

بهار ۱۴۰۲

۱- روش‌های عددی چند متغیره نامقید بی‌نیاز از مشتق

هدف این بخش از تمرین مقایسه چند الگوریتم عددی نامقید بی‌نیاز از مشتق می‌باشد. به این منظور تابع هدف به صورت زیر تعریف شده‌است.

$$f(x) = e^{-x} + 2\cos(x+9) + 7\sin^2(8x+2) + 5\sin(4x+2) + e^{\frac{x}{2}}$$

با استفاده از دستورات کتابخانه Scipy زبان برنامه‌نویسی Python و با استفاده از هر یک از روش‌های زیر، نقطه بهینه تابع هدف را بدست آورده و به همراه مقدار تابع در این نقطه و نمودار پیشرفت الگوریتم (مقدار تابع بر حسب شماره تکرار) گزارش کنید. هر یک از حالت‌ها را برای حداقل ۳ نقطه اولیه متفاوت (در بازه$\pm$۳) تکرار کنید (جمعا ۹ سری پاسخ). در انتها بهترین نتیجه بدست آمده با استفاده از هر یک از الگوریتم‌ها را در جدولی آورده و با یکدیگر مقایسه کنید.

a. Method = ‘Powell’ (Link)
b. Method = ‘Nelder-Mead’ (Link)
c. Method = ‘CG’ (Link)

در این قسمت تکلیف می‌بایست با استفاده از الگوریتم‌های عددی نامقید چند متغیره مینیمم تابع زیر را به دست آورده و شرط توقف را چک کرده و مقدار تابع را بر حسب شماره مرحله تکرار (iteration) رسم کنیم. در ابتدا تابع هزینه، دیگر توابع مورد استفاده، کتابخانه‌های مورد نیاز و دیگر پارامترهای مورد استفاده را به شکل زیر تعریف می‌کنیم:

In [14]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize

print("{0:9s}   {1:9.5s}   {2:9.5s}   {3:9.5s}   {4:9.5s}   {5:9.5s}   {6:9.5s}   {7:9.5s}   {8:9.5s}".format('iter',
      'x1','x2','x3','x4','x5','x6','x7','f(x)'))
s=[]
n=[]
def rosen(xs):
    y=0
    for i in range(7):
        y+=(np.exp(-xs[i]))+(2*(np.cos(xs[i]+9)))+(7*(np.sin(8*xs[i]+2)**2))+(5*(np.sin(4*xs[i]+2)))+(np.exp(xs[i]/2))
    return y

def generate_print_callback():
    saved_params = { "iteration_number" : 0 }
    saved_params["iteration_number"] = 1
    def print_callback(xs):
        if saved_params["iteration_number"]%1 == 0:
            print('{0: 4d}   {1: 3.6f}   {2: 3.6f}   {3: 3.6f}   {4: 3.6f}   {5: 3.6f}   {6: 3.6f}   {7: 3.6f}   {8: 3.6f}'
                  .format(saved_params["iteration_number"],xs[0],xs[1],xs[2],xs[3],xs[4],xs[5],xs[6],rosen(xs)))
            saved_params["iteration_number"] += 1
        n.append(saved_params["iteration_number"]-1)
        s.append(rosen(xs))
    return print_callback
```

```
iter        x1          x2          x3          x4          x5          x6          x7          f(x)
```

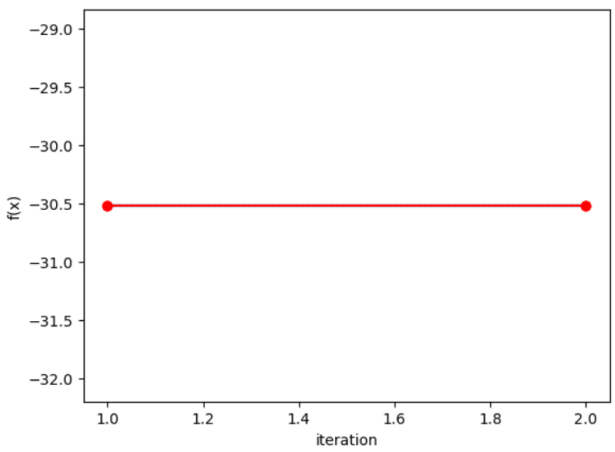
در ابتدا الگوریتم Powell را برای قسمت method قرار می‌دهیم:

- نقطه (0,0,0,0,0,0,0) را به عنوان فرض اولیه (x0) در نظر می‌گیریم و دستورات مربوط به نمودار را می‌نویسیم:

```
In [17]: x0 = [0,0,0,0,0,0,0]

minimize(rosen,
         x0,
         args=(),
         method='Powell',
         bounds=None,
         tol=None,
         callback=generate_print_callback(),
         options={'xtol': 0.0000000001,
                  'ftol': 0.000000001,
                  'maxiter': None,
                  'maxfev': None,
                  'disp': True,
                  'direc': None,
                  'return_all': False})

print(s)
print(n)

plt.plot(n,s)
plt.plot(n,s,'o:r')
plt.xlabel("iteration")
plt.ylabel("f(x)")
plt.show()
```

```
   1    0.546324    0.546324    0.546324    0.546324    0.546324    0.546324    0.546324   -30.520551
   2    0.546324    0.546324    0.546324    0.546324    0.546324    0.546324    0.546324   -30.520551
Optimization terminated successfully.
         Current function value: -30.520551
         Iterations: 2
         Function evaluations: 290
```

```
[-30.520550759901134, -30.52055075990115, -30.520550759901134, -30.52055075990115, -30.520550759901134, -30.52055075990115]
[1, 2, 1, 2, 1, 2]
```

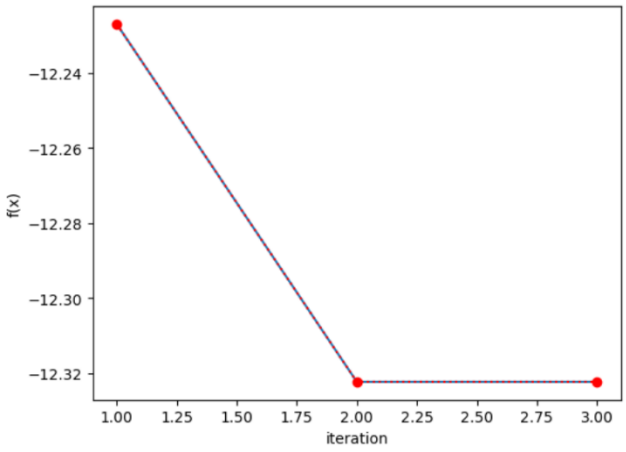


فایل کد پایتون نوشته شده در محیط Jupyter Notebook برای تمامی بخش‌ها در فایل پیوست تقدیم شده است. شایان ذکر است برای تمامی موارد kernel ریست شده و به صورت تکی بررسی شده‌اند.

- نقطه (3, 2,-1,0,1,2 ,3-) را به عنوان فرض اولیه (x0) در نظر می‌گیریم.

In [2]:
```
x0 = [-3,-2,-1,0,1,2 ,3]

minimize(rosen,
         x0,
         args=(),
         method='Powell',
         bounds=None,
         tol=None,
         callback=generate_print_callback(),
         options={'xtol': 0.0001,
                  'ftol': 0.0001,
                  'maxiter': None,
                  'maxfev': None,
                  'disp': True,
                  'direc': None,
                  'return_all': False})

print(s)
print(n)

plt.plot(n,s)
plt.plot(n,s,'o:r')
plt.xlabel("iteration")
plt.ylabel("f(x)")
plt.show()
```

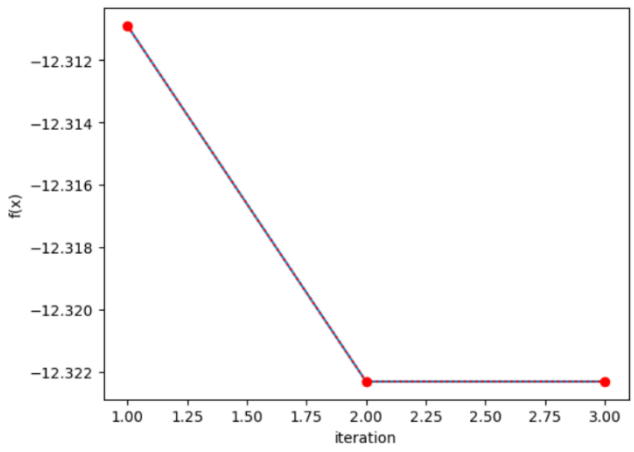
```
   1    0.559786   -0.654719   -1.019373    0.548405    0.548003    2.114289    3.688068   -12.227019
   2    0.546285   -0.657160   -1.019416    0.546334    0.546332    2.113995    3.685218   -12.322313
   3    0.546324   -0.657160   -1.019416    0.546324    0.546324    2.113996    3.685220   -12.322314
Optimization terminated successfully.
         Current function value: -12.322314
         Iterations: 3
         Function evaluations: 321
[-12.227018519667999, -12.322313017914476, -12.322313809491364]
[1, 2, 3]
```



- نقطه (-2,-1,0,1,1,2,3) را به عنوان فرض اولیه (x0) در نظر می‌گیریم.

In [2]:
```
x0 = [-2,-1,0,1,1,2,3]

minimize(rosen,
         x0,
         args=(),
         method='Powell',
         bounds=None,
         tol=None,
         callback=generate_print_callback(),
         options={'xtol': 0.0001,
                  'ftol': 0.0001,
                  'maxiter': None,
                  'maxfev': None,
                  'disp': True,
                  'direc': None,
                  'return_all': False})

print(s)
print(n)

plt.plot(n,s)
plt.plot(n,s,'o:r')
plt.xlabel("iteration")
plt.ylabel("f(x)")
plt.show()
```

```
   1   -0.654719   -1.019373    0.548405    0.548003    0.548003    2.114289    3.688068   -12.310896
   2   -0.657160   -1.019416    0.546334    0.546332    0.546332    2.113995    3.685218   -12.322314
   3   -0.657160   -1.019416    0.546324    0.546324    0.546324    2.113996    3.685220   -12.322314
Optimization terminated successfully.
         Current function value: -12.322314
         Iterations: 3
         Function evaluations: 329
[-12.310896228991654, -12.322313707277791, -12.322313809491424]
[1, 2, 3]
```

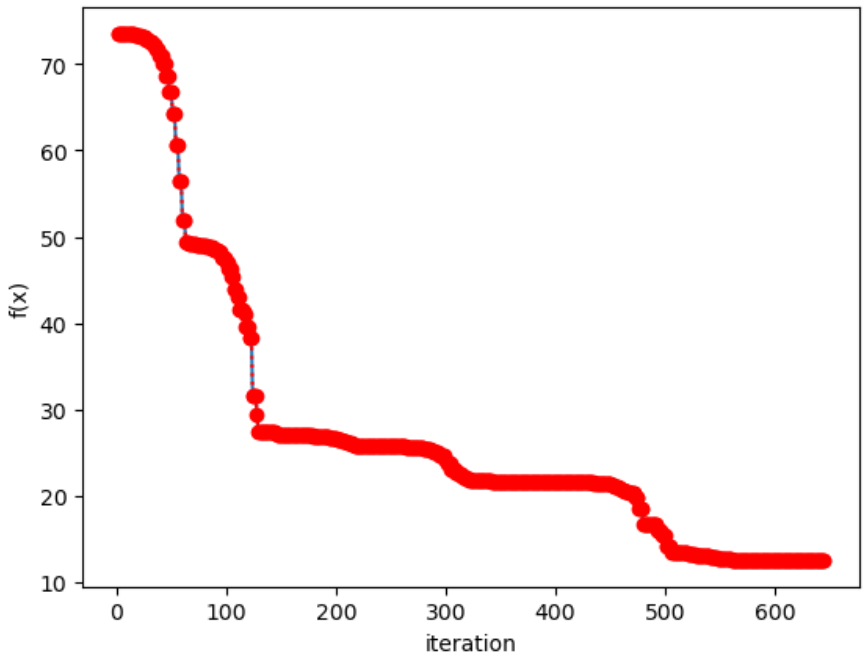


در این حالت Nelder-Mead را در قسمت method قرار می‌دهیم:

- نقطه (0,0,0,0,0,0,0) را به عنوان فرض اولیه (x0) در نظر می‌گیریم.

```
In [2]: x0 = [0,0,0,0,0,0,0]

minimize(rosen,
         x0,
         args=(),
         method='Nelder-Mead',
         bounds=None,
         tol=None,
         callback=generate_print_callback(),
         options={'xtol': 0.0001,
                  'ftol': 0.0001,
                  'maxiter': None,
                  'maxfev': None,
                  'disp': True,
                  'direc': None,
                  'return_all': False})

print(s)
print(n)

plt.plot(n,s)
plt.plot(n,s,'o:r')
plt.xlabel("iteration")
plt.ylabel("f(x)")
plt.show()
```

```
 643    0.140880    0.086443    0.160761    0.148424   -0.259293    0.625121    0.497942    12.499202
 644    0.140880    0.086443    0.160761    0.148424   -0.259293    0.625121    0.497942    12.499202
 645    0.140880    0.086443    0.160761    0.148424   -0.259293    0.625121    0.497942    12.499202
Optimization terminated successfully.
         Current function value: 12.499202
         Iterations: 645
         Function evaluations: 980
[73.54480758388516, 73.54480758388516, 73.54480758388516, 73.54480758388516, 73.54480758388516, 73.54480758388516, 73.5315260
3452675, 73.53152603452675, 73.53152603452675, 73.5086376842837, 73.5086376842837, 73.48248154283678, 73.48248154283678, 73.4
5083128943088, 73.45083128943088, 73.45083128943088, 73.38182695961625, 73.38182695961625, 73.38182695961625, 73.317462662549
```



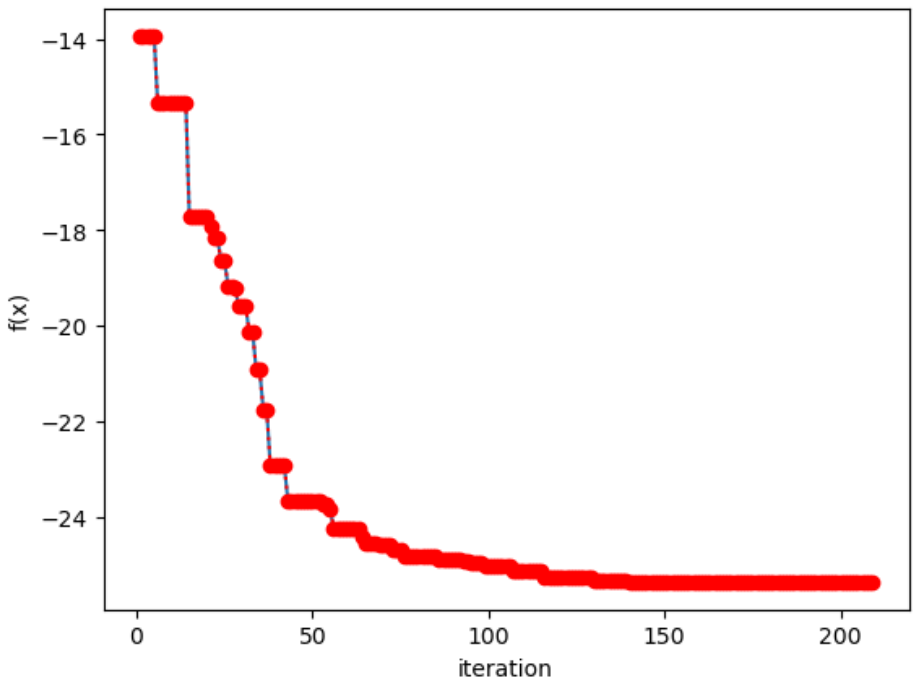
علت توقف در این حالت این است که به حد تخمین و محاسبه تابع رسیدیم با توجه به اینکه تا نقطه کمینه فاصله داریم و کمینه سازی تکمیل نشده این مقدار را افزایش می‌دهیم. برای این کار پارامتر maxfev را 1e5

قرار می‌دهیم (نتایج کد این قسمت تحت عنوان 1-b در کد آورده شده است).

- نقطه (0.5,0.5,0.5,0. 5,0.75,0.75,1) را به عنوان فرض اولیه (x0) در نظر می‌گیریم.

```
In [2]: x0 = [0.5,0.5,0.5,0.5,0.75,0.75,1]

minimize(rosen,
         x0,
         args=(),
         method='Nelder-Mead',
         bounds=None,
         tol=None,
         callback=generate_print_callback(),
         options={'xtol': 0.0001,
                  'ftol': 0.0001,
                  'maxiter': None,
                  'maxfev': None,
                  'disp': True,
                  'direc': None,
                  'return_all': False})

print(s)
print(n)

plt.plot(n,s)
plt.plot(n,s,'o:r')
plt.xlabel("iteration")
plt.ylabel("f(x)")
plt.show()
```

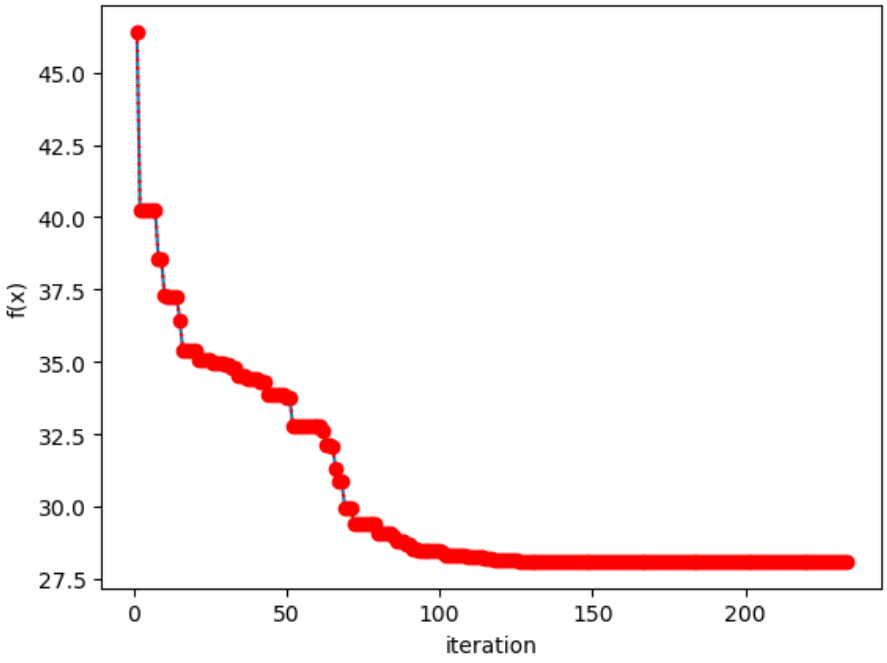
```
 207    0.546312    0.546284    0.546297    0.546278    0.908485    0.908550    0.908550   -25.373686
 208    0.546317    0.546355    0.546279    0.546261    0.908540    0.908523    0.908528   -25.373686
 209    0.546317    0.546355    0.546279    0.546261    0.908540    0.908523    0.908528   -25.373686
Optimization terminated successfully.
         Current function value: -25.373686
         Iterations: 209
         Function evaluations: 323
[-13.945797837836324, -13.945797837836324, -13.945797837836324, -13.945797837836324, -13.945797837836324, -15.34826486716631
8, -15.348264867166318, -15.348264867166318, -15.348264867166318, -15.348264867166318, -15.348264867166318, -15.3482648671663
```



- نقطه (-2.5,-0.5,0.5,1,1.25,2.75,3) را به عنوان فرض اولیه (x0) در نظر می‌گیریم.

In [2]:
```
x0 = [-2.5,-0.5,0.5,1,1.25,2.75,3]

minimize(rosen,
         x0,
         args=(),
         method='Nelder-Mead',
         bounds=None,
         tol=None,
         callback=generate_print_callback(),
         options={'xtol': 0.0001,
                  'ftol': 0.0001,
                  'maxiter': None,
                  'maxfev': None,
                  'disp': True,
                  'direc': None,
                  'return_all': False})

print(s)
print(n)

plt.plot(n,s)
plt.plot(n,s,'o:r')
plt.xlabel("iteration")
plt.ylabel("f(x)")
plt.show()
```

```
 230   -2.580558   -0.657203    0.546348    0.908545    1.304959    2.874375    2.874345    28.102704
 231   -2.580558   -0.657203    0.546348    0.908545    1.304959    2.874375    2.874345    28.102704
 232   -2.580528   -0.657140    0.546319    0.908550    1.304879    2.874364    2.874316    28.102701
 233   -2.580528   -0.657140    0.546319    0.908550    1.304879    2.874364    2.874316    28.102701
Optimization terminated successfully.
         Current function value: 28.102701
         Iterations: 233
         Function evaluations: 366
[46.403725558293765, 40.23637512294261, 40.23637512294261, 40.23637512294261, 40.23637512294261, 40.23637512294261, 40.236375
12294261, 38.577318269069295, 38.5584778423823, 37.31190339636453, 37.24429590174949, 37.24429590174949, 37.24429590174949, 3
7.24429590174949, 36.453443499074496, 35.40814156795368, 35.40814156795368, 35.40814156795368, 35.40814156795368, 35.40814156
```



در این حالت CG را در قسمت method قرار می‌دهیم.

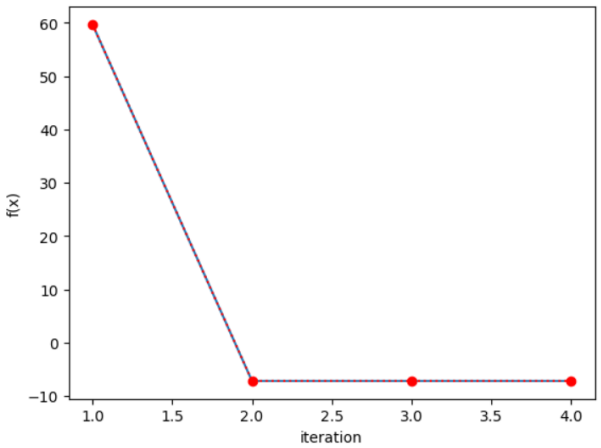
- نقطه (0,0,0,0,0,0,0) را به عنوان فرض اولیه (x0) در نظر می‌گیریم.

```
In [2]: x0 = [0,0,0,0,0,0,0]

minimize(rosen,
         x0,
         args=(),
         method='CG',
         bounds=None,
         tol=None,
         callback=generate_print_callback(),
         options={'xtol': 0.0001,
                  'ftol': 0.0001,
                  'maxiter': None,
                  'maxfev': 1e5,
                  'disp': True,
                  'direc': None,
                  'return_all': False})

print(s)
print(n)

plt.plot(n,s)
plt.plot(n,s,'o:r')
plt.xlabel("iteration")
plt.ylabel("f(x)")
plt.show()
```

```
   1    1.908721    1.908721    1.908721    1.908721    1.908721    1.908721    1.908721    59.698893
   2    2.113924    2.113924    2.113924    2.113925    2.113925    2.113925    2.113924    -7.193190
   3    2.113990    2.113990    2.113990    2.113990    2.113990    2.113990    2.113990    -7.193207
   4    2.113996    2.113996    2.113996    2.113996    2.113996    2.113996    2.113996    -7.193207
Optimization terminated successfully.
         Current function value: -7.193207
         Iterations: 4
         Function evaluations: 120
         Gradient evaluations: 15
```

```
[59.698892906602936, -7.193189974853965, -7.193206870211011, -7.193206957189595]
[1, 2, 3, 4]
```

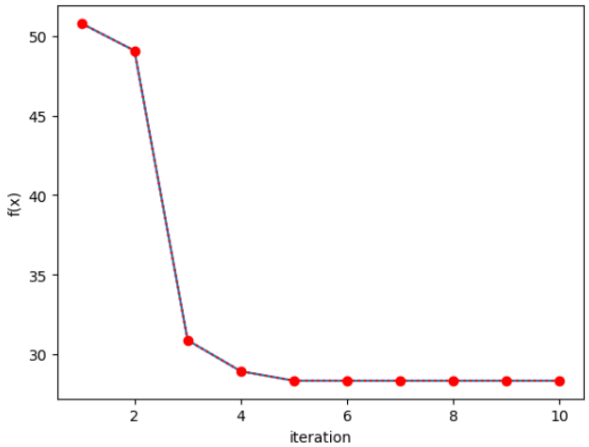
```
C:\Users\Auspicious\AppData\Local\Temp\ipykernel_2592\79533274.py:3: OptimizeWarning: Unknown solver options: xtol, ftol, maxfe
v, direc
  minimize(rosen,
```



- نقطه (-0.75,-0.5,0.25,0,1.25,2.75,3) را به عنوان فرض اولیه (x0) در نظر می‌گیریم.

In [2]:
```
x0 = [-0.75,-0.5,0.25,0,1.25,2.75,3]

minimize(rosen,
         x0,
         args=(),
         method='CG',
         bounds=None,
         tol=None,
         callback=generate_print_callback(),
         options={'xtol': 0.0001,
                  'ftol': 0.0001,
                  'maxiter': None,
                  'maxfev': 1e5,
                  'disp': True,
                  'direc': None,
                  'return_all': False})

print(s)
print(n)

plt.plot(n,s)
plt.plot(n,s,'o:r')
plt.xlabel("iteration")
plt.ylabel("f(x)")
plt.show()
```

```
   1    1.228880   -2.942979   -1.188266    2.135311    2.625601    3.632490    0.486206   50.790907
   2    1.262311   -2.971685   -1.185233    2.152410    2.617399    3.657035    0.475606   49.070621
   3    1.320979   -2.954807   -1.075066    2.096541    2.479214    3.710533    0.580456   30.877170
   4    1.317826   -2.952026   -1.019717    2.102914    2.459923    3.698786    0.569868   28.928600
   5    1.306258   -2.953176   -1.019409    2.114554    2.475568    3.684901    0.546581   28.333914
   6    1.304995   -2.953534   -1.019417    2.113934    2.477022    3.685255    0.546297   28.332000
   7    1.304904   -2.953626   -1.019416    2.114005    2.476983    3.685215    0.546328   28.331989
   8    1.304906   -2.953645   -1.019416    2.113994    2.476982    3.685221    0.546324   28.331989
   9    1.304907   -2.953646   -1.019416    2.113996    2.476982    3.685220    0.546324   28.331989
  10    1.304907   -2.953646   -1.019416    2.113996    2.476982    3.685220    0.546324   28.331989


Optimization terminated successfully.
         Current function value: 28.331989
         Iterations: 10
         Function evaluations: 200
         Gradient evaluations: 25
[50.79090696205635, 49.07062119412317, 30.877170457345066, 28.928600307759403, 28.333913720980945, 28.33199984214818, 28.331989
360450773, 28.331989165293976, 28.331989162641314, 28.33198916264105]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
C:\Users\Auspicious\AppData\Local\Temp\ipykernel_14252\1923779652.py:3: OptimizeWarning: Unknown solver options: xtol, ftol, ma
xfev, direc
  minimize(rosen,
```

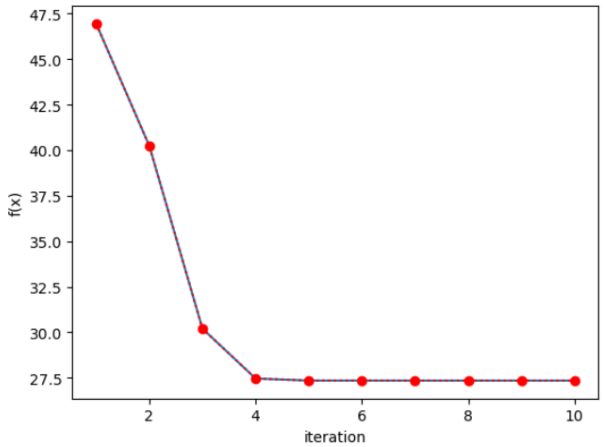


- نقطه (-2.75,-1.5,-0.25,0,1.5,2.25,2.75) را به عنوان فرض اولیه (x0) در نظر می‌گیریم.

In [2]:
```
x0 = [-2.75,-1.5,-0.25,0,1.5,2.25,2.75]

minimize(rosen,
         x0,
         args=(),
         method='CG',
         bounds=None,
         tol=None,
         callback=generate_print_callback(),
         options={'xtol': 0.0001,
                  'ftol': 0.0001,
                  'maxiter': None,
                  'maxfev': 1e5,
                  'disp': True,
                  'direc': None,
                  'return_all': False})

print(s)
print(n)

plt.plot(n,s)
plt.plot(n,s,'o:r')
plt.xlabel("iteration")
plt.ylabel("f(x)")
plt.show()
```

1	-2.150527	-0.940913	-0.319281	0.413660	1.381889	1.890747	2.920959	46.949278
2	-2.119308	-0.907686	-0.298285	0.521939	1.322777	1.844745	2.912955	40.236202
3	-2.210330	-0.968921	-0.260505	0.554114	1.302557	1.801542	2.869793	30.196599
4	-2.225108	-1.026753	-0.260521	0.543123	1.305372	1.743532	2.875153	27.459139
5	-2.220322	-1.019499	-0.260518	0.546732	1.304974	1.727943	2.874474	27.346658
6	-2.220306	-1.019038	-0.260518	0.546703	1.304919	1.728054	2.874369	27.346532
7	-2.220825	-1.019456	-0.260518	0.546277	1.304908	1.729899	2.874339	27.344823
8	-2.220774	-1.019417	-0.260518	0.546324	1.304907	1.729937	2.874337	27.344819
9	-2.220773	-1.019416	-0.260518	0.546324	1.304907	1.729938	2.874337	27.344819
10	-2.220773	-1.019416	-0.260518	0.546324	1.304907	1.729938	2.874337	27.344819

```
Optimization terminated successfully.
         Current function value: 27.344819
         Iterations: 10
         Function evaluations: 184
         Gradient evaluations: 23
[46.949278070395465, 40.2362016196209, 30.196598784865778, 27.459138833258894, 27.346658188590684, 27.346531636865713, 27.34482
272719099, 27.34481903538793, 27.34481903482962, 27.34481903482766]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
C:\Users\Auspicious\AppData\Local\Temp\ipykernel_1284\1229317753.py:3: OptimizeWarning: Unknown solver options: xtol, ftol, max
fev, direc
  minimize(rosen,
```



According to the results obtained from Python, the “Nelder-Mead” method convergence rate is very low compared to the other methods. Also “Nelder-Mead” method is not capable of solving the problem with an initial point far from the optimum point showing the warning.

بخش دوم تکلیف

$$f(x_1,x_2) = (x_1 - x_2)^2 + 0.5\cos(x_1)\, e^{(1-\sin(x_2))^2} + 4\sin(x_2)\, e^{(1-\cos(x_1))^2}$$

$$s.t.: 6x_2^2 - 2x_1^4 + 0.1x_1^6 < 0, \qquad -6 < x_1, x_2 < 6$$

طبق خواسته سوال بایستی تابعی نوشته شود که با دریافت تعداد متغیرها و بازه هر یک از آنها، تابع هزینه، قیدها، یک نقطه اولیه، ضریب انعکاس، شرط توقف، دقت ضریب انعکاس، بتواند مختصات نقطه کمینه، مختصات رئوس اولیه و نهایی و تعداد تکرار را به ما نمایش دهد که این تابع به صورت زیر تعریف شده است:

```
[x_cc,f_cc,iters,first_points,final_points] = optimcomplex(cost_func,vars,constraints,ffeasible_point,stop_criterion,alpha,alpha_min)
```

مطابق با توضیحات فوق در صورت سوال، ورودی‌های این تابع به صورت زیر بایستی وارد شوند:

```
clc
clear all

%% input of problem
syms x_1 x_2
vars(1).symbol = x_1;        %first variable's symbol
vars(1).lowerbound = -6;  %lower bound of first variable
vars(1).upperbound = 0;    %upper bound of first variable
vars(2).symbol = x_2;        %second variable's symbol
vars(2).lowerbound = 0; %lower bound of second variable
vars(2).upperbound = 6;    %upper bound of second variable
constraints(1) = (6.*x_2^2) + (-2.*x_1^4) + (0.1.*x_1^6) < 0;
alpha = 1.3;
alpha_min = 0.0001;
stop_criterion = 0.0001;
ffeasible_point=[-2.5,-2.5]; %first feasible point
cost_func = (x_1-x_2)^2 + (0.5.*cos(x_1) * exp((1-sin(x_2))^2)) + (4.*sin(x_2) * exp((1-cos(x_1))^2));

```

همانگونه که مشخص است برای وارد کردن متغیرها و بازه مجاز آن‌ها از structure استفاده می‌شود و برای تغییر ورودی‌ها نیز بایستی از این مورد پیروی کرد. نقطه اولیه مجاز نیز تحت عنوان ffeasible_point بایستی وارد شود که در اینجا برابر ۲/۵- و ۲/۵- برای x_1 و x_2 در نظر گرفته شده است. قیدها نیز به همان شکل اولیه خود تحت عنوان constraints وارد می‌شوند. در نهایت هم تابع ریاضی مورد نظر برای کمینه سازی تعریف شده است. به منظور انجام دستورالعمل گفته شده در صورت پروژه توابع مختلفی تعریف شده است که برای هر یک متناسب با عملکرد آن‌ها عنوانی ذکر شده است و هم نوشته‌ای در بالای هر تابع از جهت توضیح قرار داده شده است. به همین منظور و به دلیل توضیحات در صورت سوال به توضیح هر بخش به صورت جداگانه پرداخته نشده است.

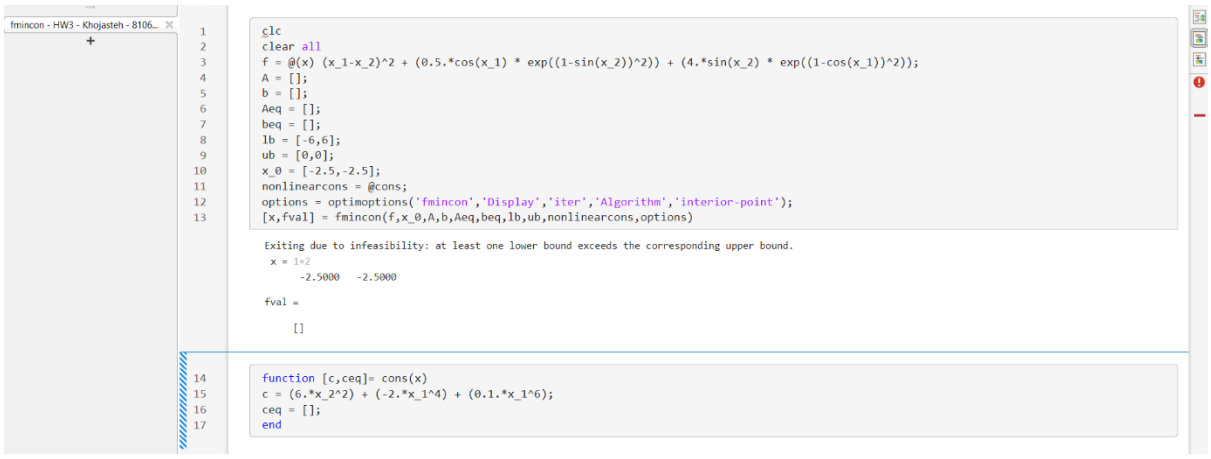
کد کامل در پیوست تقدیم شده است.

اگر این تابع را اجرا کنیم، به نتایجی که ادامه آمده است خواهیم رسید.

```
clc
clear all

%% input of problem
syms x_1 x_2
vars(1).symbol = x_1;        %first variable's symbol
vars(1).lowerbound = -6;  %lower bound of first variable
vars(1).upperbound = 0;    %upper bound of first variable
vars(2).symbol = x_2;        %second variable's symbol
vars(2).lowerbound = 0; %lower bound of second variable
vars(2).upperbound = 6;    %upper bound of second variable
constraints(1) = (6.*x_2^2) + (-2.*x_1^4) + (0.1.*x_1^6) < 0;
alpha = 1.3;
alpha_min = 0.0001;
stop_criterion = 0.0001;
ffeasible_point=[-2.5,-2.5]; %first feasible point
cost_func = (x_1-x_2)^2 + (0.5.*cos(x_1) * exp((1-sin(x_2))^2)) + (4.*sin(x_2) * exp((1-cos(x_1))^2)

%% making initial points
while feasibility_checking(ffeasible_point,vars,constraints)==false
    for j=1:length(vars)
        ffeasible_point(j)=random('Uniform',vars(j).lowerbound,vars(j).upperbound);
    end
end

%% results
[x_cc,f_cc,iters,first_points,final_points] = optimcomplex(cost_func,vars,constraints,ffeasible_poin

%% optimcomplex
function [x_cc,f_cc,iters,first_points,final_points]=optimcomplex(cost_func,vars,constraints,ffeasib
    n=length(vars);
```

```
x_cc = 1×2
   -0.7590    0.0053

f_cc = 1.583
iters = 15
first_points = 4×2
   -2.0694    0.9757
   -3.6777    1.9829
   -1.6124    1.2423
   -2.4884    1.3429

final_points = 4×2
   -0.7587    0.0030
   -0.7652    0.0088
   -0.7559    0.0034
   -0.7562    0.0059
```

شایان به ذکر است که مقدار نهایی نقطه کمینه به دست آمده از این الگوریتم برای توابع متفاوت به مواردی همچون نقطه مجاز اولیه، ضریب انعکاس و دقت ضریب انعکاس بستگی دارد. در ادامه به بررسی نقطه کمینه به دست آمده با استفاده از الگوریتم fmincon می‌پردازیم که کد آن به صورت زیر نوشته شده است.



```
clc
clear all
f = @(x) (x_1-x_2)^2 + (0.5.*cos(x_1) * exp((1-sin(x_2))^2)) + (4.*sin(x_2) * exp((1-cos(x_1))^2));
A = [];
b = [];
Aeq = [];
beq = [];
lb = [-6,6];
ub = [0,0];
x_0 = [-2.5,-2.5];
nonlinearcons = @cons;
options = optimoptions('fmincon','Display','iter','Algorithm','interior-point');
[x,fval] = fmincon(f,x_0,A,b,Aeq,beq,lb,ub,nonlinearcons,options)
```

```
Exiting due to infeasibility: at least one lower bound exceeds the corresponding upper bound.
 x = 1×2
   -2.5000   -2.5000

fval =

    []
```

```
function [c,ceq]= cons(x)
c = (6.*x_2^2) + (-2.*x_1^4) + (0.1.*x_1^6);
ceq = [];
end
```