Project 1

February 11, 2022

0.1 Project 1 : Unrestrected Optimization

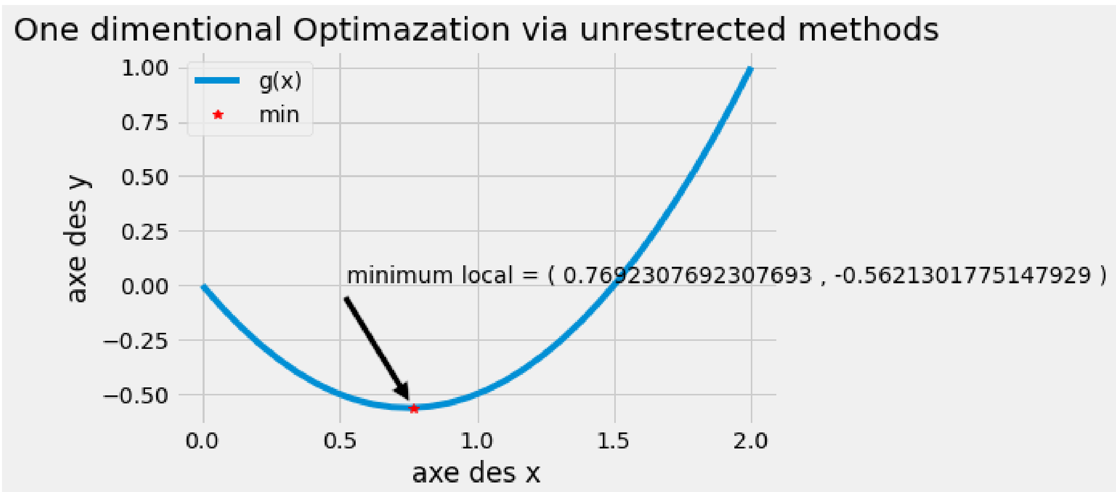
[58]:
```
import time
import numpy as np
import matplotlib.pyplot as plt
```

0.2 I- One dimensional Optimization

[59]:
```
import Unrestrected_Optimization.One_dimentional_optimization
def g(x):
    return x*(x-1.5)
xs=0.0
xf=2.0
n =6
min=Unrestrected_Optimization.One_dimentional_optimization.
 ↪fibonacci_method(g,xs,xf,n)

#plot
x=np.linspace( 0 , 2, 30 )
fig, ax = plt.subplots()
ax.plot(x, g(x), label='g(x)')
ax.grid(True)
ax.set_xlabel('axe des x')
ax.set_ylabel('axe des y')
ax.set_title("One dimentional Optimazation via unrestrected methods")
ax.plot(min,g(min),'r*', label="min")
ax.legend()
ax.annotate ( f" minimum local = ( {min} , {g(min)} ) ", xy=(min, g(min)),␣
 ↪xytext=(0.5, 0.01), arrowprops=dict(facecolor='black', shrink=0.05) )
plt.show()
plt.close()

#fixed_tep_size
t1=time.perf_counter()
min1=Unrestrected_Optimization.One_dimentional_optimization.fixed_Step_Size(g,0.
 ↪0 , 0.05 )
t2=time.perf_counter()
time1=t2-t1
```

```
print("\tla methode a pas fixe donne :\n  min = ",min1,"\n son temps␣
↪d'execussion est : ",time1 , "\n" )

#accelerated step size method
t3=time.perf_counter()
min2=Unrestreted_Optimization.One_dimentional_optimization.
↪accelerated_sep_size(g ,0.0, 0.05)
t4=time.perf_counter()
time2=t4-t3
print("\tla methode a pas accelere donne :\n  min = ",min2,"\n son temps␣
↪d'execussion est : ", time2 , "\n" )

#dichotomous method
t5=time.perf_counter()
min3=Unrestreted_Optimization.One_dimentional_optimization.
↪dichotomous_method(g,0.0,1.0,6)
t6=time.perf_counter()
time3=t6-t5
print("\tla methode dichotomique donne :\n min = ",min3,"\n son temps␣
↪d'execussion est : ", time3 , "\n" )


#interval halving method
t7=time.perf_counter()
min4=Unrestreted_Optimization.One_dimentional_optimization.
↪intervalhalving_method(g, 0.0 , 1.0 , 7 )
t8=time.perf_counter()
time4=t8-t7
print("\tla methode interval halving donne :\n  min = ",min4,"\n son temps␣
↪d'execussion est : ", time4 , "\n" )

#Fibonacci method
t9=time.perf_counter()
min5=Unrestreted_Optimization.One_dimentional_optimization.
↪fibonacci_method(g,0.0,1.0,6)
t10=time.perf_counter()
time5=t10-t9
print("\tla methode de Fibonacci donne :\n  min = ",min5,"\n son temps␣
↪d'execussion est : ", time5, "\n" )

#Golden Section Methode
t11=time.perf_counter()
min6=Unrestreted_Optimization.One_dimentional_optimization.
↪goldensection_method(g,0.0,1.0)
t12=time.perf_counter()
time6=t12-t11
```

```python
print("\tla methode du nombre d'or donne :\n  min = ",min6,"\n son temps␣
↪d'execussion est : ", time6 , "\n" )

#Exhaustive method
t13=time.perf_counter()
min7= Unrestrected_Optimization.One_dimentional_optimization.
↪exhaustive_method(g ,0.0,1.0,9)
t14=time.perf_counter()
time7=t14-t13
print("\tla methode exhaustive donne  :\n  min = ",min7,"\n son temps␣
↪d'execussion est : ", time7 , "\n" )
```




```
        la methode a pas fixe donne :
  min =  0.7750000000000001
 son temps d'execussion est :  4.850002005696297e-05

        la methode a pas accelere donne :
  min =  0.7749999999999988
 son temps d'execussion est :  6.829993799328804e-05

        la methode dichotomique donne :
 min =  0.7575546874999999
 son temps d'execussion est :  4.309997893869877e-05

        la methode interval halving donne :
  min =  0.75
 son temps d'execussion est :  0.00028009992092847824

        la methode de Fibonacci donne :
  min =  0.7692307692307692
```

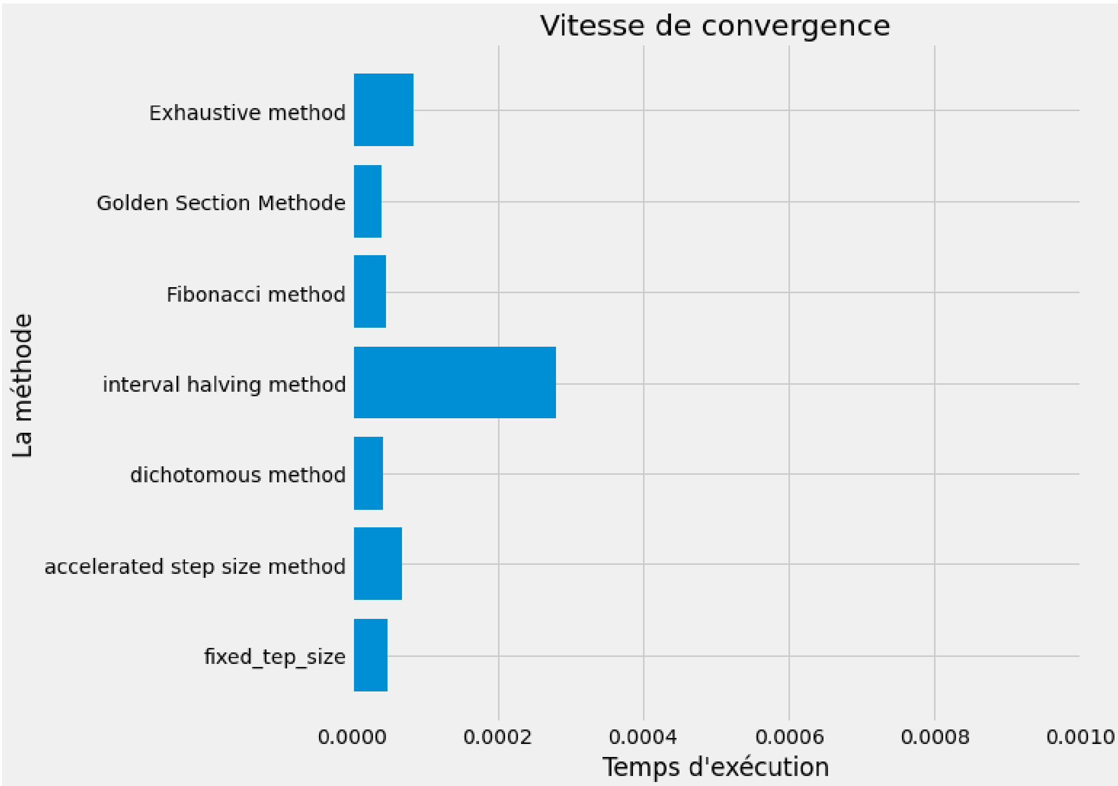
```
son temps d'execussion est :  4.789978265762329e-05

       la methode du nombre d'or donne :
 min =  0.7507644230864321
son temps d'execussion est :  4.0599843487143517e-05

       la methode exhaustive donne  :
 min =  0.75
son temps d'execussion est :  8.559995330870152e-05
```

0.3 Comparison

[60]:
```
data={"fixed_tep_size":[min1,time1] ,"accelerated step size method":
 ↪[min2,time2] ,"dichotomous method":[min3,time3]
     ,"interval halving method":[min4,time4] ,"Fibonacci method":[min5,time5]␣
 ↪,"Golden Section Methode":[min6,time6]
     ,"Exhaustive method":[min7,time7]}
method_name=list(data.keys())
method_data=np.array(list(data.values()))
method_time=method_data[:,1]
method_precision=method_data[:,0]
fig,ax=plt.subplots(figsize=(8,8))

#custumize the plot
plt.style.use('fivethirtyeight')
#time plot
print("method_data=",method_data)
print("method_precision=",method_precision)
print("method_time",method_time)
ax.barh(method_name,method_time)
ax.set(xlim=[0.000000, 0.001], xlabel="Temps d'exécution", ylabel='La méthode',
       title='Vitesse de convergence')
```

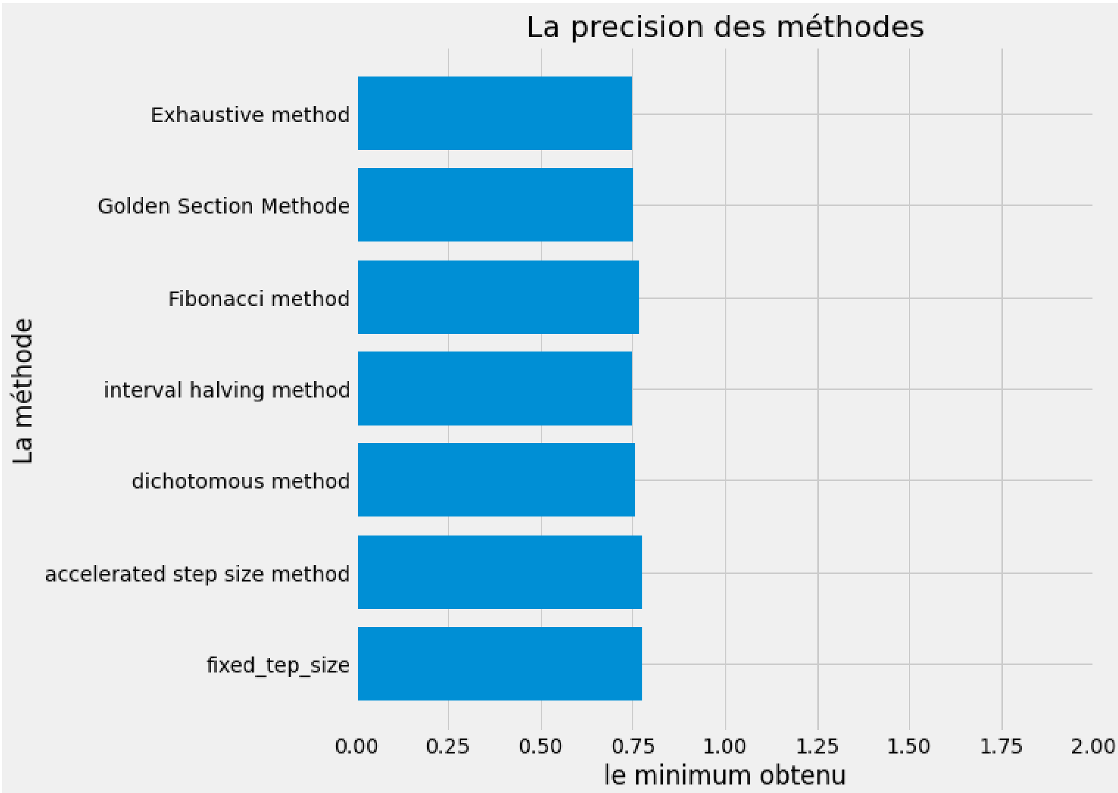
```
method_data= [[7.75000000e-01 4.85000201e-05]
 [7.75000000e-01 6.82999380e-05]
 [7.57554687e-01 4.30999789e-05]
 [7.50000000e-01 2.80099921e-04]
 [7.69230769e-01 4.78997827e-05]
 [7.50764423e-01 4.05998435e-05]
 [7.50000000e-01 8.55999533e-05]]
method_precision= [0.775      0.775      0.75755469 0.75       0.76923077
0.75076442
 0.75      ]
method_time [4.85000201e-05 6.82999380e-05 4.30999789e-05 2.80099921e-04
 4.78997827e-05 4.05998435e-05 8.55999533e-05]
```

[60]: [(0.0, 0.001),
 Text(0.5, 0, "Temps d'exécution"),
 Text(0, 0.5, 'La méthode'),
 Text(0.5, 1.0, 'Vitesse de convergence')]



[61]:
```
#precision plot
fig,ax=plt.subplots(figsize=(8,8))
plt.style.use('fivethirtyeight')
ax.barh(method_name,method_precision)
ax.set(xlim=[0.000000,2], xlabel="le minimum obtenu", ylabel='La méthode',
       title='La precision des méthodes')
```

[61]: [(0.0, 2.0),
 Text(0.5, 0, 'le minimum obtenu'),
 Text(0, 0.5, 'La méthode'),
 Text(0.5, 1.0, 'La precision des méthodes')]

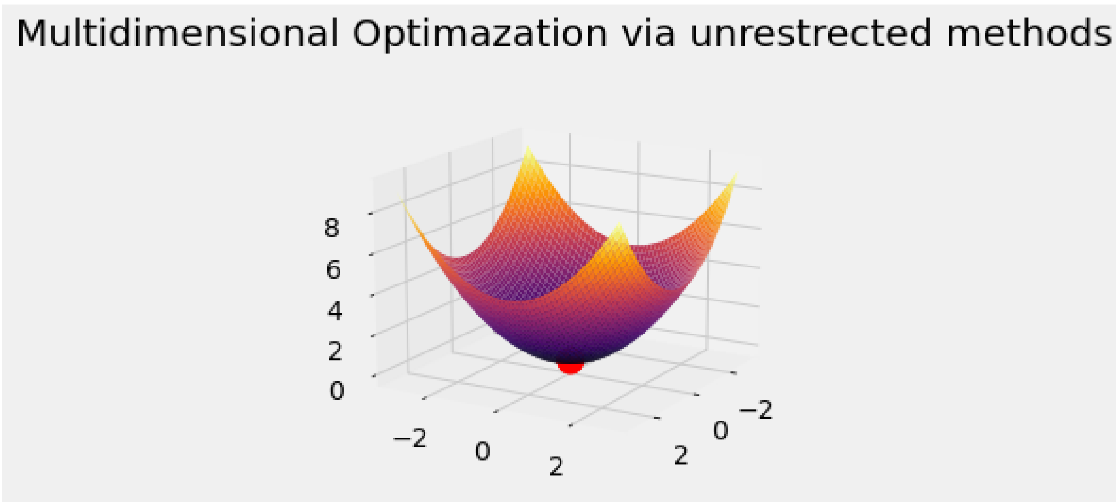


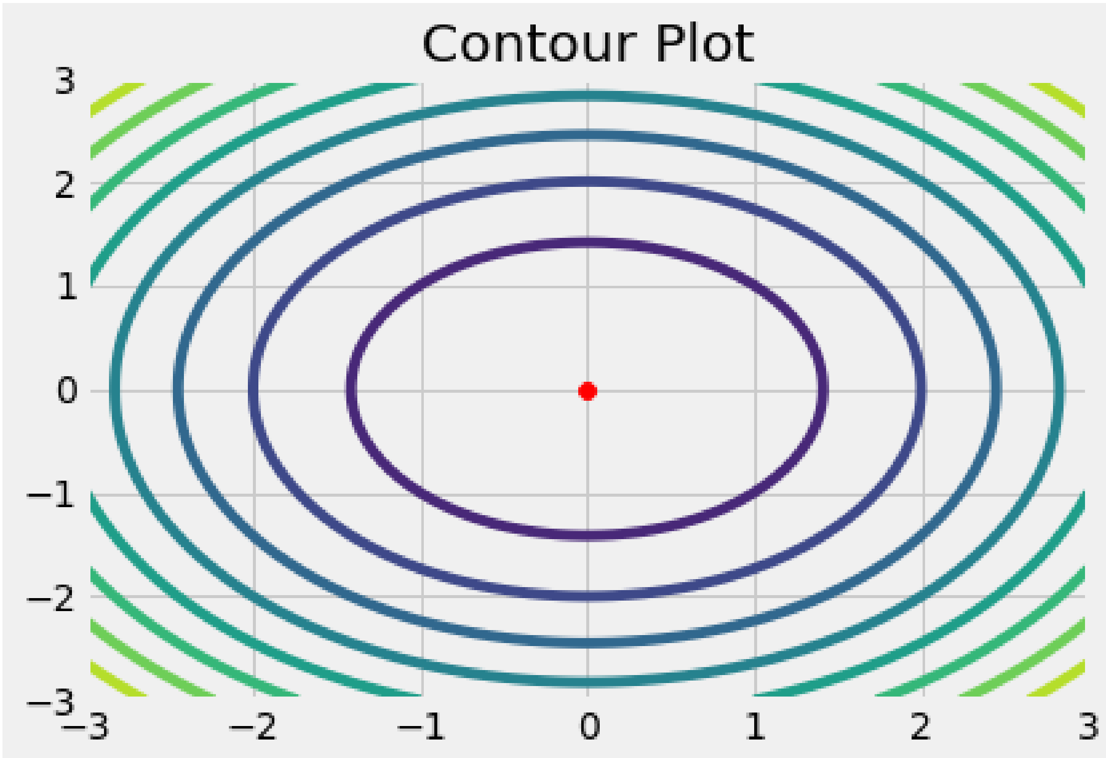
0.4 II- Multidimensional Optimization

[62]:
```
import Unrestrected_Optimization.Multivariable_optimization
f= lambda x:   0.5* ( x[0]**2+x[1]**2 )
x=np.array([2,1])
epsilon =1e-15
min,path=Unrestrected_Optimization.Multivariable_optimization.Gradient_descent␣
 ↪(f,x,epsilon)
print(path)
print("min=",min)
fig,ax =plt.subplots(subplot_kw={"projection": "3d"} )
x_axe=np.linspace(-3,3,100)
y_axe=np.linspace(-3,3,100)
X,Y=np.meshgrid(x_axe,y_axe)
Z=np.array([X,Y])
F=f(Z)

ax.plot_surface(X, Y, F ,antialiased=True,cmap= 'inferno')
ax.scatter( min , min,  f( np.array( [min,min] ) ) ,color="red",s=200)
ax.view_init(azim=30, elev= 15)
ax.set_title("Multidimensional Optimazation via unrestrected methods")
```

```
plt.show()
plt.close()

plt.contour(X,Y,F,10)
plt.scatter(min , f( np.array( [min,min]) ) ,color="red",marker='o')
plt.title("Contour Plot")
plt.show()
plt.close()
print("min=\n",min)
print("f( np.array( min ) ) =",f( np.array( min ) ))
print(" np.array( [min,min] )  =\n",np.array( [min,min] ))
```

```
[array([2, 1]), array([0., 0.])]
min= [0. 0.]
```






```
min=
 [0. 0.]
f( np.array( min ) ) = 0.0
 np.array( [min,min] )  =
 [[0. 0.]
 [0. 0.]]
```

0.4.1 Comparison between Multidimensionel Optimization Methods

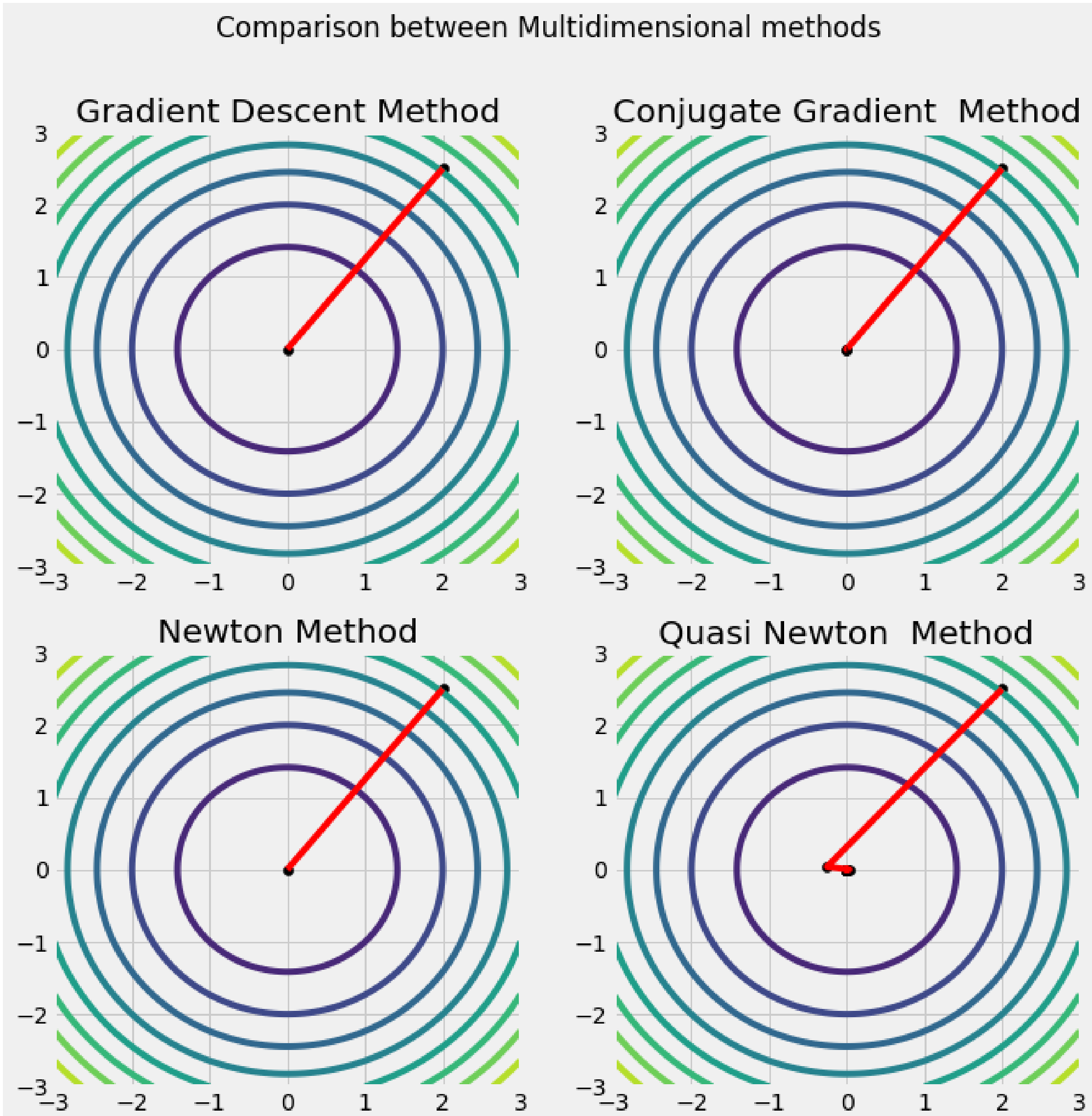
[63]:

```python
#Create 4 subplots
fig,axs=plt.subplots(2,2,figsize=(10,10))
fig.suptitle("Comparison between Multidimensional methods" )

  #For each subplot we create title and contour plot
Methods=["Gradient Descent","Conjugate Gradient ","Newton","Quasi Newton "]
i=0
for row in range(2):
    for col in range(2):
        #axs[row,col].annotate(f'{Methods[i]} Method',(0.5, 0.5),␣
 ↪transform=axs[row, col].transAxes)
        axs[row,col].set_title(f' {Methods[i]} Method ')
        axs[row,col].contour(X,Y,F,10)
        i+=1
```

```python
  #Gradient Descent
min0,path0=Unrestrected_Optimization.Multivariable_optimization.
 ↪Gradient_descent (f,x,epsilon)
print("path0 =" ,path0)
    #make "patth0" an np.array type
path0=np.array(path0)
        #path0[:,0] _example :
""" L=np.array([[1,2],[7,8]])             L=[[1 2]
                                           [7 8]]
    print(L)                         >>>>>  L[:,0] =[1 7]
    print(L[:,0])
"""

F0=[]
for e in path0 :
    F0+=[f(e)]

axs[0,0].scatter(path0[:,0], F0 ,c='black' ,marker='o')
axs[0,0].plot( path0[:,0] , F0 , color = 'red', linestyle = 'solid')

  #Conjugate Gradient
min1,path1=Unrestrected_Optimization.Multivariable_optimization.
 ↪Conjugate_Gradient(f,x )
path1=np.array(path1)
F1=[]
for e in path1:
    F1+=[f(e)]
axs[0,1].scatter( path1[:,0] , F1, c='black' ,marker ='o' )
axs[0,1].plot( path1[:,0] ,F1 , color='red' ,linestyle ='solid' )


   #Newton
min2,path2=Unrestrected_Optimization.Multivariable_optimization.Newton␣
 ↪(f,x,epsilon )
path2=np.array(path2 )
F2=[]
for e in path2 :
    F2+=[f(e)]
axs[1,0].scatter(path2[:,0] , F2 ,c='black',marker='o' )
axs[1,0].plot(path2[:,0], F2, color='red',linestyle='solid')
   #Quasi Newton
min3,path3=Unrestrected_Optimization.Multivariable_optimization.
 ↪Quasi_Newton(f,x,epsilon)
path3=np.array(path3)
```

```
F3=[]
for e in path3 :
    F3+=[f(e)]
axs[1,1].scatter(path3[:,0], F3 , c='black',marker='o' )
axs[1,1].plot(path3[:,0] , F3 ,color='red' , linestyle='solid')

plt.show()
plt.close()

print("****************************************")
print("L 'enregistrement des positions pour les differentes méthodes :")
print("Gradient Descent :\n" ,path0 )
print("Conjugate Gradient :\n" ,path1 )
print("Newton :\n" ,path2 )
print("Quasi Newton:\n" ,path3 )
print("****************************************")
```

```
path0 = [array([2, 1]), array([0., 0.])]
```




```
****************************************
L 'enregistrement des positions pour les differentes méthodes :
Gradient Descent :
 [[2. 1.]
 [0. 0.]]
Conjugate Gradient :
 [[2.00000000e+00 1.00000000e+00]
 [4.44089210e-16 2.22044605e-16]
 [4.44089210e-15 2.22044605e-15]]
Newton :
 [[2.00000000e+00 1.00000000e+00]
 [4.44089210e-16 6.66133815e-16]]
Quasi Newton:
```

```
[[ 2.00000000e+00  1.00000000e+00]
 [-2.51799814e-01 -1.25899907e-01]
 [ 3.17015731e-02  1.58507865e-02]
 [-3.99122510e-03 -1.99561255e-03]
 [ 5.02494868e-04  2.51247434e-04]
 [-6.32640571e-05 -3.16320285e-05]
 [ 7.96493889e-06  3.98246945e-06]
 [-1.00278506e-06 -5.01392532e-07]
 [ 1.26250546e-07  6.31252731e-08]
 [-1.58949320e-08 -7.94746600e-09]
 [ 2.00117046e-09  1.00058523e-09]
 [-2.51947174e-10 -1.25973587e-10]
 [ 3.17201258e-11  1.58600629e-11]
 [-3.99356157e-12 -1.99678073e-12]
 [ 5.02788917e-13  2.51394370e-13]
 [-6.33010985e-14 -3.16505604e-14]
 [ 7.96959024e-15  3.98479658e-15]
 [-1.00337557e-15 -5.01691198e-16]
 [ 1.26321959e-16  6.31552906e-17]]
****************************************
```

```
[ ]:
```