Mukul Shingwani

B20AI023

Lab 09

```
from numpy import *
```

Question 1

```
def gradf(fun,x):
  n,h1=len(x),pow(10,-7)
  g=zeros((n,1),dtype=float)
  for i in range(0,n):
    x1,x2=x.copy(),x.copy()
    x1[i],x2[i]=x1[i]+h1,x2[i]-h1
    g[i]=(fun(x1)-fun(x))/(h1)
  return g
```

```
def quasi_newton(fun,con,x0):
  beta1,beta2,r,eps,iter1,n=pow(10,-4),0.9,0.5,pow(10,-5),0,len(x0)
  B0=identity(n,dtype=float)
  f0,g0=fun(x0),gradf(fun,x0)
  alpha=1
  while linalg.norm(g0)>eps and iter1<20000 and alpha>pow(10,-5):
    d0,alpha=-dot(linalg.inv(B0),g0),1

    while max(con(x0+alpha*d0))>-0.000001:
      alpha=alpha*r
    x1=x0+alpha*d0
    f1,g1=fun(x1),gradf(fun,x1)

    while (f1>f0+alpha*beta1*g0.T@d0) and alpha>pow(10,-5):
      alpha=alpha*r
      x1 = x0 + alpha * d0
      f1, g1 = fun(x1),gradf(fun,x1)
    dt1,s1=x1-x0,g1-g0

    if dt1.T@s1>pow(10,-3):
      B0=B0+1/(dt1.T@s1)*s1@s1.T-1/(s1.T@B0@s1)*B0@s1@s1.T@B0

    x0,g0,iter1=x1,g1,iter1+1
```

```
  if iter1>=20000:
    print('maximum iteration attained')

  return x0
```

```
def obj_fun(x):
 return 100*pow(x[0]-1,2)+pow(x[1],2)
```

```
def con_fun(x):
 g=zeros((2,1),dtype=float)
 g[0]=x[0]+6*x[1]-36
 g[1]=-4*x[0]+x[1]
 return g
```

```
def interior_point_solver(obj_fun,con_fun,x0):
  print('-------')
  sigma,opt_cond,iter1=10.0,1.0,0.0
  if max(con_fun(x0)) > -pow(10,-5):
    print('initial point is not strictly feasible. So starting phase 1')
    n = len(x0)
    y0 = zeros((n + 1, 1), dtype=float)
    y0[0:n], y0[n], sigma = x0, max(con_fun(x0) + 1), 10.0
    print(y0)

    def con_fun_phase_1(x):
      n = len(x)
      return con_fun(x[0:n]) - x[-1]

    while max(con_fun(y0[0:n])) > -0.001:
      def barr_phase_1(x):
        return x[-1] - 1 / sigma * sum(log(-con_fun_phase_1(x)))

      y0 = quasi_newton(barr_phase_1, con_fun_phase_1, y0)
      sigma, iter1 = sigma * 10, iter1 + 1

    x0 = y0[0:n]
    print('Phase I complete')
    print('interior point=', x0,'\n','constraint_value=', con_fun(x0))

  else:
    print('initial approximation is an interior point so starting phase II directly')

  sigma=10
  opt_cond=1

  while len(con_fun(x0))/sigma > 0.00000001 and opt_cond >pow(10,-5):
    def barr_fun(x):
      return obj_fun(x) - 1 / sigma * sum(log(-con_fun(x)))
```

```
    x0 = quasi_newton(barr_fun, con_fun, x0)
    opt_cond = linalg.norm(gradf(barr_fun, x0))
    print(opt_cond)
    iter1+=1
    sigma=sigma*5

  print('-------')
  print(sigma)

  if len(con_fun(x0))/sigma <=0.00000001:
    print('maximum iterations attends')
  else:
    print('optimal solution found as norm KKT=',opt_cond,'<10^-7')

  return x0,obj_fun(x0), con_fun(x0),iter1, -10/sigma*1/con_fun(x0)
```

```
x0,fval,con_val,iter1,lagrange_mult=\
interior_point_solver(obj_fun,con_fun,10*ones((2,1),dtype=float))
print('-------')
print('optimalpoint=',x0,'\n')
print('objective value=',fval,'\n')
print('constraint value=',con_val,'\n')
print('no of iterations=',iter1)
print('Lagrange multiplier=',lagrange_mult)
print('-------')
```

```
-------
initial point is not strictly feasible. So starting phase 1
[[10.]
 [10.]
 [35.]]
maximum iteration attained
Phase I complete
interior point= [[ 3698.93834558]
 [-3692.25271244]]
 constraint_value= [[-18490.57792908]
 [-18488.00609477]]
6.602252277357933e-06
-------
50
optimal solution found as norm KKT= 6.602252277357933e-06 <10^-7
-------
optimalpoint= [[ 1.00048285]
 [-0.02097205]]

objective value= [0.00046314]

constraint value= [[-35.12534948]
 [ -4.02290343]]

no of iterations= 2.0
Lagrange multiplier= [[0.00569389]
```

```
 [0.04971534]]
-------
```

Question 2

```
def obj_fun(x):
  return x[0]**2+x[1]**2+2*x[2]**2+x[3]**2-5*x[0]-5*x[1]-21*x[2]+7*x[3]
```

```
def con_fun(x):
  g=zeros((2,1),dtype=float)
  g[0]=x[0]**2+x[1]**2+x[2]**2+x[3]**2+x[0]-x[1]+x[2]-x[3]-8
  g[1]=x[0]**2+2*x[1]**2+x[2]**2+2*x[3]**2-x[0]-x[3]-10
  return g
```

```
x0,fval,con_val,iter1,lagrange_mult=\
interior_point_solver(obj_fun,con_fun,10*ones((4,1),dtype=float))
print('-------')
print('optimalpoint=',x0,'\n')
print('objective value=',fval,'\n')
print('constraint value=',con_val,'\n')
print('no of iterations=',iter1)
print('Lagrange multiplier=',lagrange_mult)
print('-------')
```

```
-------
initial point is not strictly feasible. So starting phase 1
[[ 10.]
 [ 10.]
 [ 10.]
 [ 10.]
 [571.]]
Phase I complete
interior point= [[-0.30105104]
 [ 0.33406226]
 [-0.40052355]
 [ 0.41703144]]
 constraint_value= [[-8.91610463]
 [-9.29390394]]
7.414539833360545e-06
-------
50
optimal solution found as norm KKT= 7.414539833360545e-06 <10^-7
-------
optimalpoint= [[ 0.37284274]
 [ 1.05397994]
 [ 2.06768453]
 [-0.63900034]]
```

```
objective value= [-44.81964542]

constraint value= [[-0.04092616]
 [-2.28112109]]

no of iterations= 2.0
Lagrange multiplier= [[4.88685016]
 [0.08767619]]
-------
```

Question 3

```
def obj_fun(x):
  return exp(x[0])*(4*x[0]**2+2*x[1]**2+4*x[0]*x[1]+2*x[1]+1)
```

```
def con_fun(x):
  g=zeros((2,1),dtype=float)
  g[0]=x[0]+2*x[1]-5
  g[1]=x[0]**2+x[1]**2-25
  return g
```

```
x0,fval,con_val,iter1,lagrange_mult=\
interior_point_solver(obj_fun,con_fun,10*ones((2,1),dtype=float))
print('-------')
print('optimalpoint=',x0,'\n')
print('objective value=',fval,'\n')
print('constraint value=',con_val,'\n')
print('no of iterations=',iter1)
print('Lagrange multiplier=',lagrange_mult)
print('-------')
```

```
-------
initial point is not strictly feasible. So starting phase 1
[[ 10.]
 [ 10.]
 [176.]]
Phase I complete
interior point= [[-1.54805662]
 [-3.09611448]]
 constraint_value= [[-12.74028558]
 [-13.01759581]]
9.630235931125218e-06
-------
50
optimal solution found as norm KKT= 9.630235931125218e-06 <10^-7
-------
optimalpoint= [[-4.6282885 ]
 [ 1.09333122]]
```

```
objective value= [0.69397574]

constraint value= [[-7.44162605]
 [-2.3835724 ]]

no of iterations= 2.0
Lagrange multiplier= [[0.02687585]
 [0.08390767]]
-------
```