Mruganshi Gohel

B20CS014

Lab-2

| import numpy as np from cvxopt import matrix, solvers import random from pandas.core.common import flatten |
| --- |

Question-1

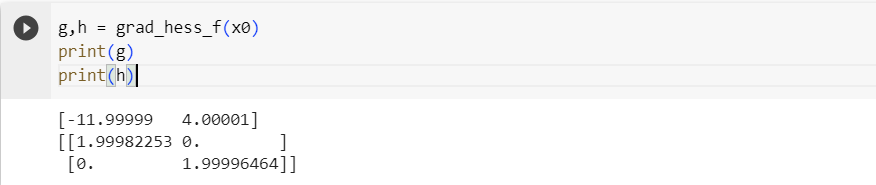
* *Step -1: Function define*

| def f(x):  return (x[0] - 4)\*\*2 + (x[1]- 4)\*\*2 |
| --- |

| x0 = np.array([-0.5 \* 4, 1.5 \* 4]) f(x0) |
| --- |

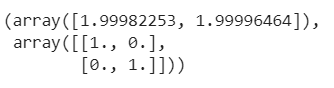
* *Step 2: Find the Gradient and hessian of the given function*

| def grad\_hess\_f(x):  f0=f(x)  n=len(x)  h= 10\*\*(-5)  b=np.zeros(n)  g=np.zeros(n)  H=np.zeros((n,n))  for i in range(0,n):  x1=x.copy()  x1[i]=x1[i]+h  b[i]=f(x1)  g[i]=(f(x1)-f0)/h  for i in range(0,n):  for j in range(0,n):  x2=x.copy()  x2[i]=x2[i]+h  x2[j]=x2[j]+h  H[i][j]=(f(x2)-b[i]-b[j]+f0)/h\*\*2  return g,H |
| --- |



* *Step-3: Find Eigen values of hessian matrix*

| np.linalg.eig(h) |
| --- |

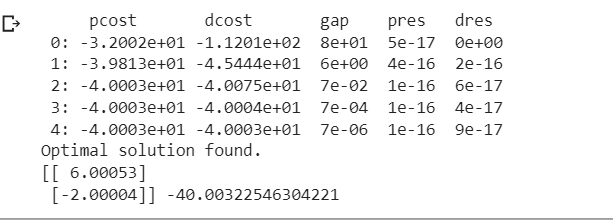


***Eigenvalues are positive and hence it is positive definite and the function is a convex function***

* *Step-4: if Hessian matrix is positive definite then find the optimal solution*

| A=np.vstack((-np.identity(2),np.identity(2))) lb=[-2\*4,-2\*4] ub=[2\*4,2\*4] b=np.append(x0-lb,ub-x0) |
| --- |

| soln= solvers.qp(matrix(h),matrix(g),matrix(A),matrix(b)) print(np.round\_(soln['x'],decimals=5),soln['primal objective']) v1\_11=np.round\_(soln['x'],decimals=5) |
| --- |



*Solution Correctness Check: if* ***dot product of gradient and optimal solution*** *is coming out* ***negative*** *then solution is correct (as said in the lab lecture by sir)*

| v1\_11=list(flatten(v1\_11)) v2\_11=list(g) np.dot(v2\_11,v1\_11) |
| --- |



Question-2

* *Step-1: function defining as per question and taking dimension as 10 in input*

| import numpy as np  #function f1 def f1\_1(x):  n = len(x)  I1 = [i for i in range(2, n+1) if i % 2 == 1]  denominator = len(I1)  result = x[0]  for i in I1:  result += ((x[i-1] - np.sin(6 \* np.pi \* x[0] + (i \* np.pi / n)))\*\*2 )\*(2/(denominator))  return result  #function f2 def f2\_1(x):  n = len(x)  I2 = [i for i in range(2, n+1) if i % 2 == 0]  denominator = len(I2)  result = 1 - np.sqrt(x[0])  for i in I2:  result += ((x[i-1] - np.sin(6 \* np.pi \* x[0] + (i \* np.pi / n)))\*\*2 )\*(2/(denominator))  return result  # taking random 10 values for input in the range given in the question x1=[] x1.append(random.uniform(0.001, 1)) for i in range(9):  x1.append(random.uniform(-1, 1)) print("f1:", f1\_1(x1)) print("f2:", f2\_1(x1)) |
| --- |

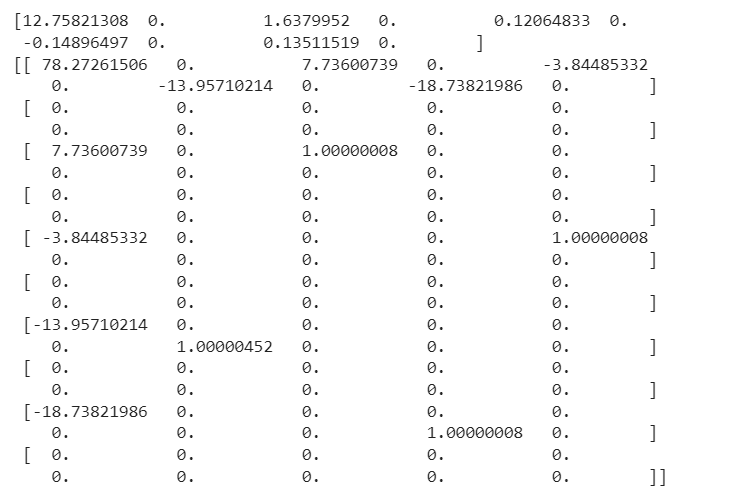


**Optimal Solution Procedure for F1**

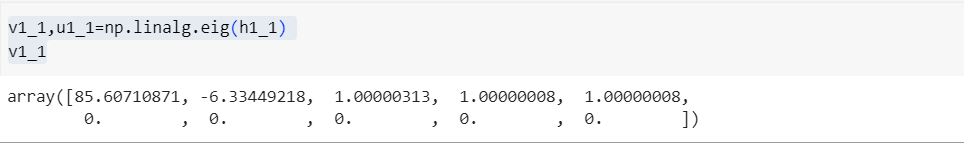
* *Step-2: gradient and hessian find for f1*

| def grad\_hess\_f1\_1(x):  f0=f1\_1(x)  n=len(x)  h= 10\*\*(-5)  b=np.zeros(n)  g=np.zeros(n)  H=np.zeros((n,n))  for i in range(0,n):  x1=x.copy()  x1[i]=x1[i]+h  b[i]=f1\_1(x1)  g[i]=(f1\_1(x1)-f0)/h  for i in range(0,n):  for j in range(0,n):  x2=x.copy()  x2[i]=x2[i]+h  x2[j]=x2[j]+h  H[i][j]=(f1\_1(x2)-b[i]-b[j]+f0)/h\*\*2  # solver.qp()  return g,H |
| --- |

| g1\_1,h1\_1 = grad\_hess\_f1\_1(x1) print(g1\_1) print(h1\_1) |
| --- |



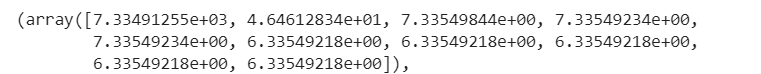
* *Step-3: Eigen value finding for hessian matrix of f1 and check if all are positive or not if positive find optimal solution else make hessian positive and find optimal solution*

**

*Hessian is not positive definite and hence to find optimal solution we have to make it positive definite by following below formula.*

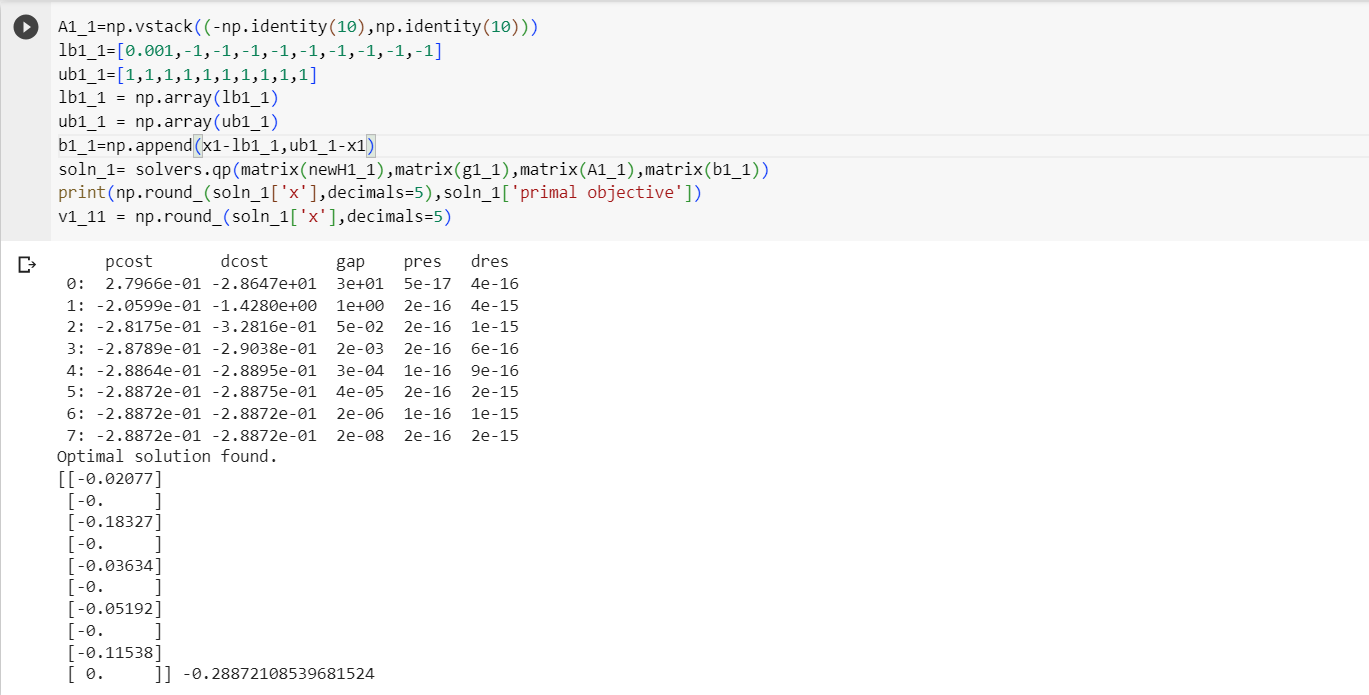
* *Step-4: New Heassian FInding*

| *newH1\_1 = h1\_1@h1\_1 + (0.001-min(v1\_1))\*np.identity(10, dtype = float) np.linalg.eig(newH1\_1)* |
| --- |

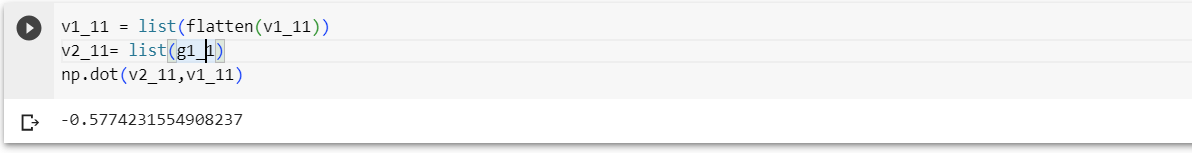
**

*As we can see above for new hessian matrix eigen values are coming out positive and hence we have positive definite hessian matrix and using this we will find optimal solution*

* *Step5: FInd Optimal Solution*

**

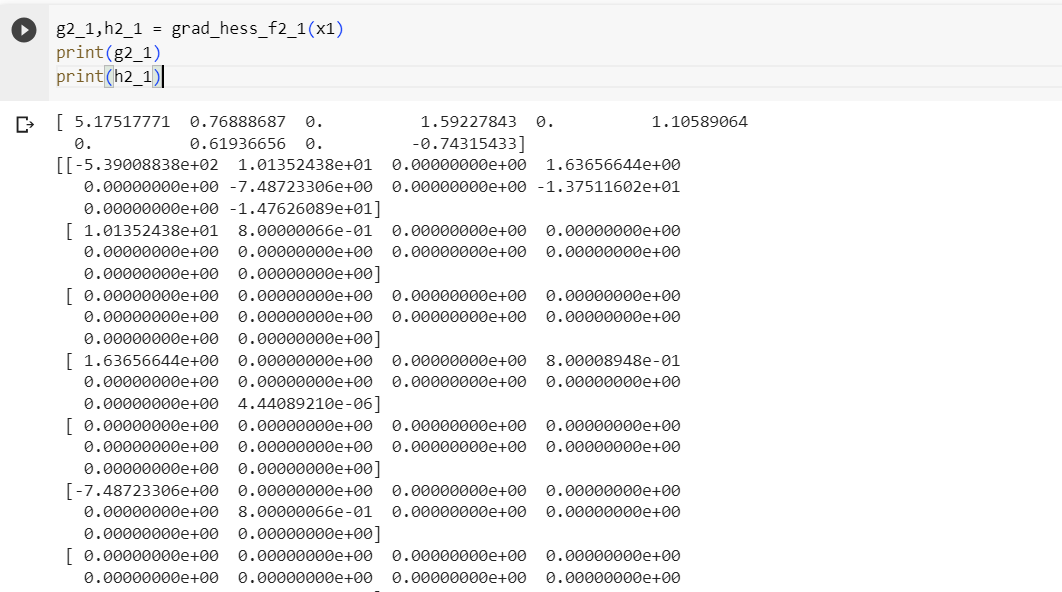
***Solution Correctness check***

******

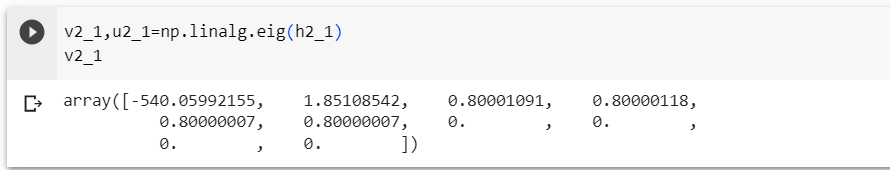
***Optimal Solution Procedure for F2***

* *Step-2: Gradient and Hessian finding*

| *def grad\_hess\_f2\_1(x):  f0=f2\_1(x)  n=len(x)  h= 10\*\*(-5)  b=np.zeros(n)  g=np.zeros(n)  H=np.zeros((n,n))  for i in range(0,n):  x1=x.copy()  x1[i]=x1[i]+h  b[i]=f2\_1(x1)  g[i]=(f2\_1(x1)-f0)/h  for i in range(0,n):  for j in range(0,n):  x2=x.copy()  x2[i]=x2[i]+h  x2[j]=x2[j]+h  H[i][j]=(f2\_1(x2)-b[i]-b[j]+f0)/h\*\*2  # solver.qp()  return g,H* |
| --- |



* *Step-3: Eigen values find*

**

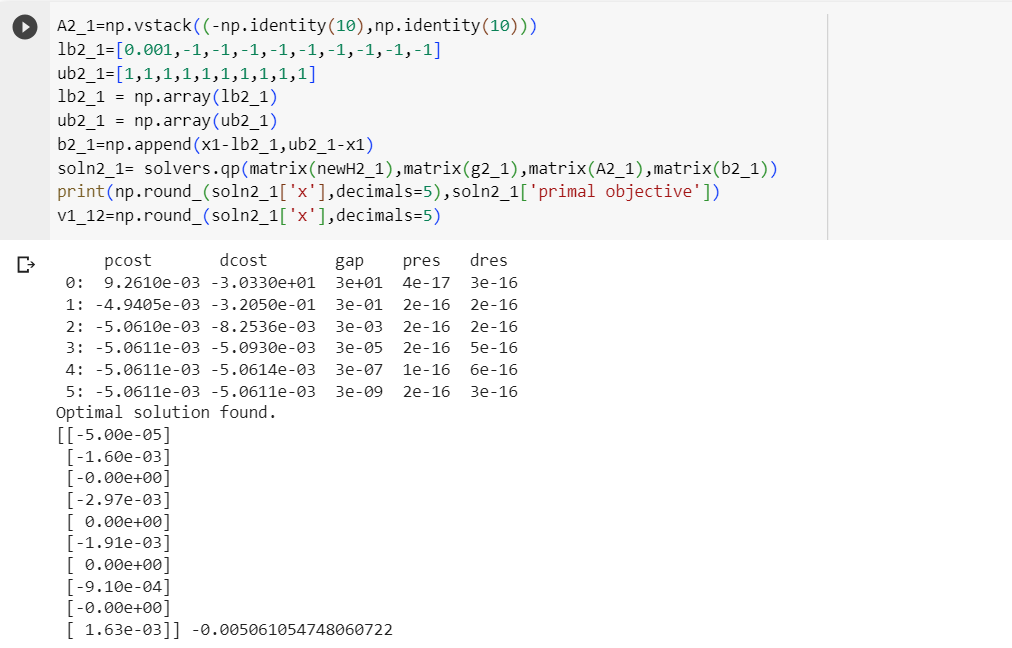
*not positive deifnite matrix*

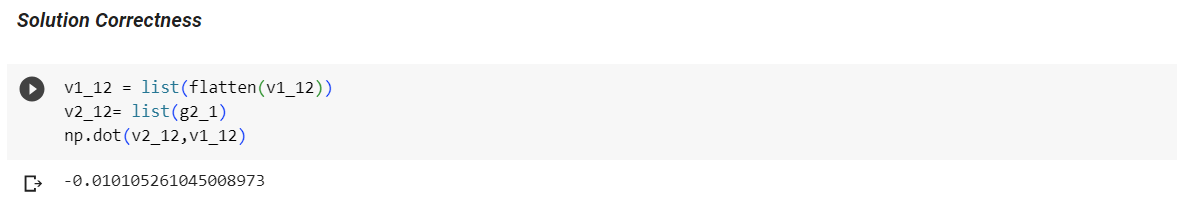
* *Step-4: Find transformed hessian and its eigen values*

| *newH2\_1 = h2\_1@h2\_1 + (0.001-min(v2\_1))\*np.identity(10, dtype = float) np.linalg.eig(newH2\_1)* |
| --- |

*All eigen values are positive and so now we have positive definite hessian to find optimal solution*

* *Step-5: Optimal solution*

**

**

Question-3

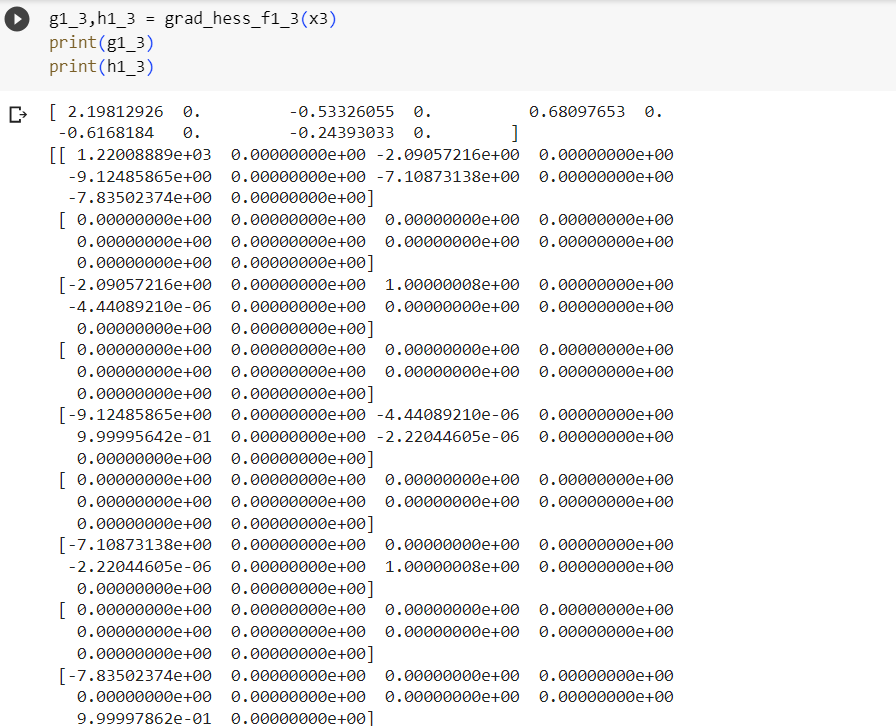
* *Function Definition*

| def yi\_3(i,x):  n=len(x)  if(i%2==1):  return x[i-1]-((0.3\* ((x[0]\*\*2)\*np.cos((24 \* np.pi \* x[0])+(4\*i\*np.pi / n))) + 0.6\*x[0])\*(np.cos((6 \* np.pi \* x[0]) + (i\* np.pi / n))))  else:  return x[i-1]-((0.3\* ((x[0]\*\*2)\*np.cos((24 \* np.pi \* x[0])+(4\*i\*np.pi / n))) + 0.6\*x[0])\*(np.sin((6 \* np.pi \* x[0]) + (i\* np.pi / n))))   def f1\_3(x):  n = len(x)  I1 = [i for i in range(2, n+1) if i % 2 == 1]  denominator = len(I1)  result = x[0]  for i in I1:  result += (yi\_3(i,x)\*\*2)\*(2/(denominator))  return result  def f2\_3(x):  n = len(x)  I2 = [i for i in range(2, n+1) if i % 2 == 0]  denominator = len(I2)  result = 1 - np.sqrt(x[0])  for i in I2:  result += (yi\_3(i,x)\*\*2)\*(2/(denominator))  return result   x3=[] x3.append(random.uniform(0.001, 1)) for i in range(9):  x3.append(random.uniform(-1, 1)) print("f1:", f1\_3(x3)) print("f2:", f2\_3(x3)) |
| --- |

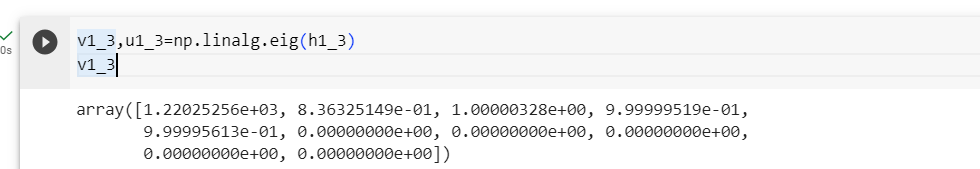
***F1 Solution Finding***

* *Gradient and Hessian Finding*

| *def grad\_hess\_f1\_3(x):  f0=f1\_3(x)  n=len(x)  h= 10\*\*(-5)  b=np.zeros(n)  g=np.zeros(n)  H=np.zeros((n,n))  for i in range(0,n):  x1=x.copy()  x1[i]=x1[i]+h  b[i]=f1\_3(x1)  g[i]=(f1\_3(x1)-f0)/h  for i in range(0,n):  for j in range(0,n):  x2=x.copy()  x2[i]=x2[i]+h  x2[j]=x2[j]+h  H[i][j]=(f1\_3(x2)-b[i]-b[j]+f0)/h\*\*2  # solver.qp()  return g,H* |
| --- |

**

*Eigen values Finding*

**

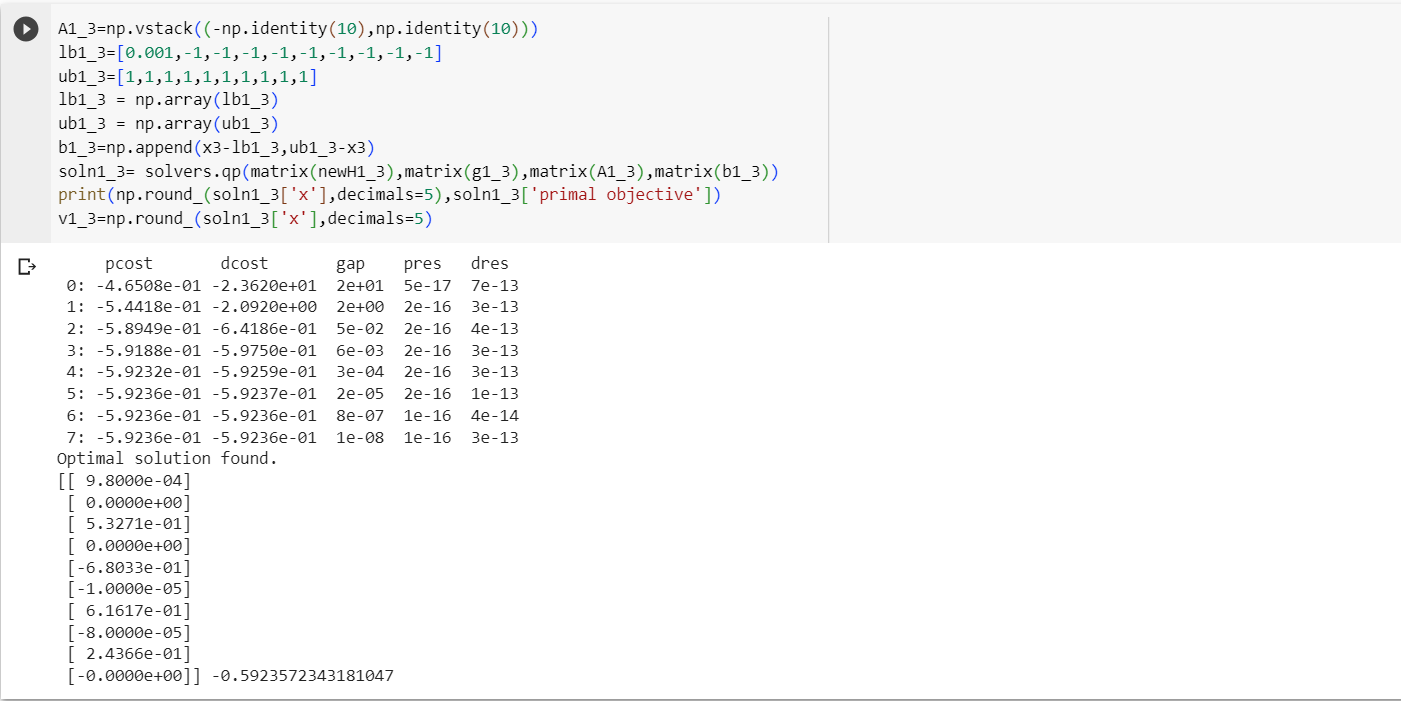
*not a positive deifinite matrix*

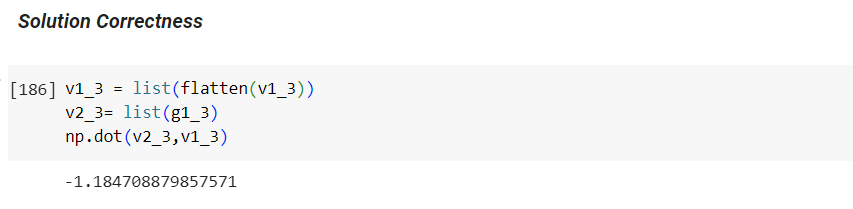
* *New Hessian*

| *newH1\_3 = h1\_3@h1\_3 + (0.001-min(v1\_3))\*np.identity(10, dtype = float) np.linalg.eig(newH1\_3)* |
| --- |

*All eigen values are positive*

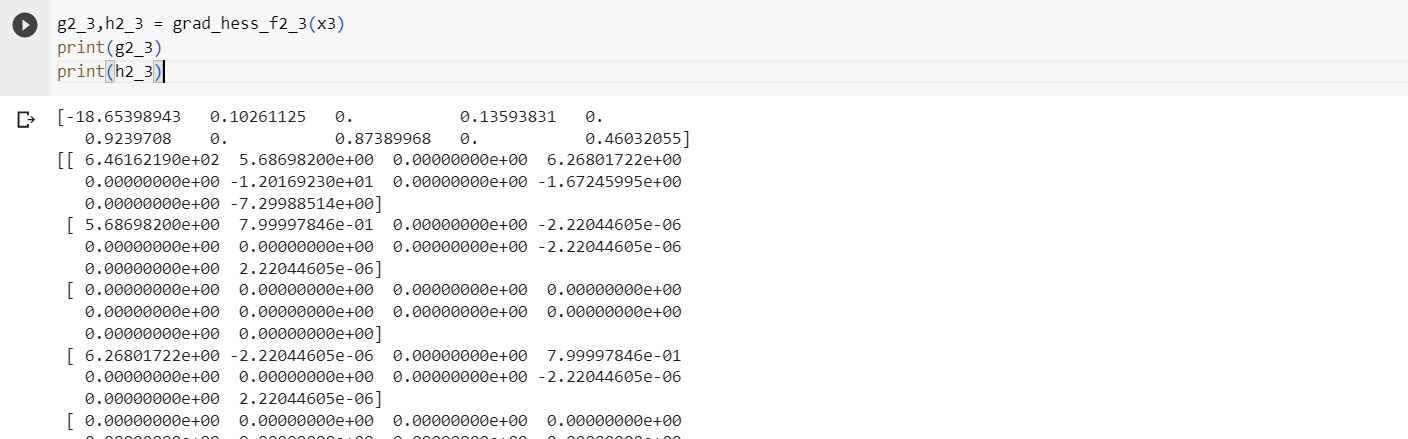
* *Optimal Solution*

**

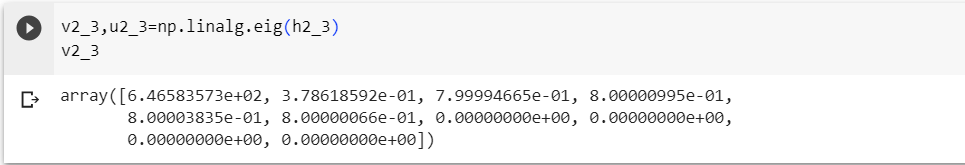
**

***F2 Solution Finding***

| **def grad\_hess\_f2\_3(x):  f0=f2\_3(x)  n=len(x)  h= 10\*\*(-5)  b=np.zeros(n)  g=np.zeros(n)  H=np.zeros((n,n))  for i in range(0,n):  x1=x.copy()  x1[i]=x1[i]+h  b[i]=f2\_3(x1)  g[i]=(f2\_3(x1)-f0)/h  for i in range(0,n):  for j in range(0,n):  x2=x.copy()  x2[i]=x2[i]+h  x2[j]=x2[j]+h  H[i][j]=(f2\_3(x2)-b[i]-b[j]+f0)/h\*\*2  # solver.qp()  return g,H** |
| --- |

******

* *Eigen values Finding for f2*

******

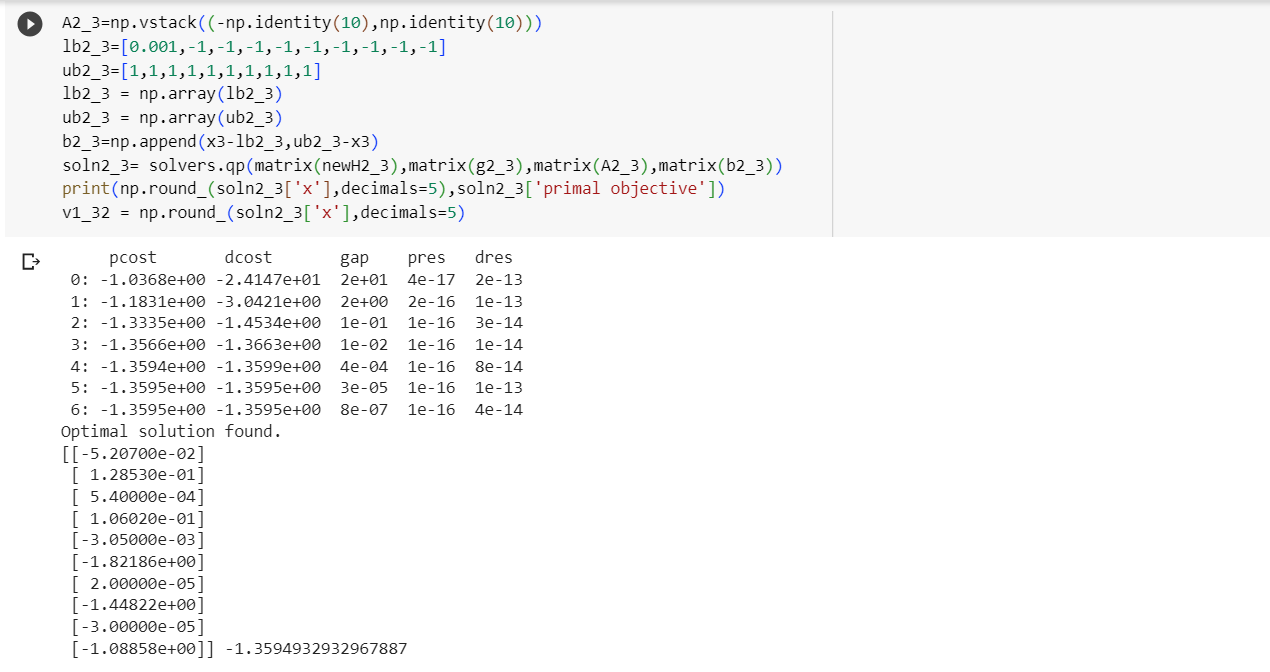
*not positive definite matrix*

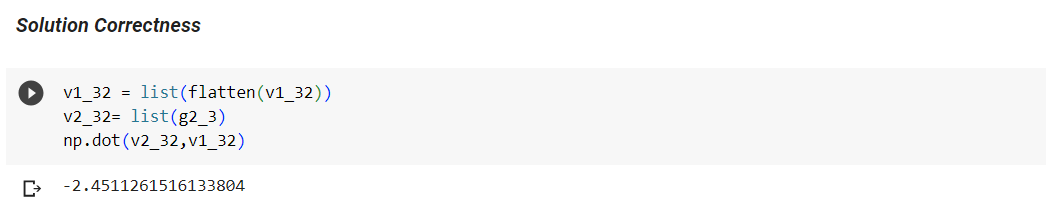
* *New hessian calculation*

| newH2\_3 = h2\_3@h2\_3 + (0.001-min(v2\_3))\*np.identity(10, dtype = float) np.linalg.eig(newH2\_3) |
| --- |

*All eigen values are positive*

* *Optimal Solution Find*

**

**

Question-4

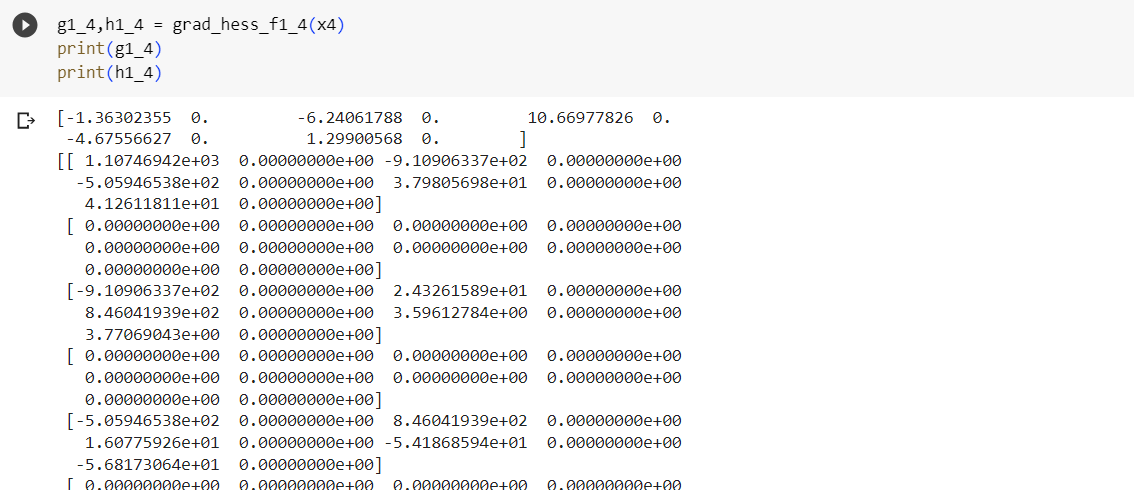
* Function definition

| def yi\_4(i,x):  n=len(x)  return x[i-1] - x[0]\*\*(0.5\*(1 + (3\*((i-2)/(n-2)))))   def f1\_4(x):  n = len(x)  I1 = [i for i in range(2, n+1) if i % 2 == 1]  denominator = len(I1)  result = x[0]  y\_values = [yi\_4(i, x) for i in I1]  sum\_y\_squared = sum([y \*\* 2 for y in y\_values])  cos\_multiplication=1  for i in I1:  cos\_multiplication = cos\_multiplication \* np.cos((20 \* yi\_4(i,x) \* np.pi)/ np.sqrt(i))  result = result + (2/denominator)\*(4\*sum\_y\_squared - 2\*cos\_multiplication + 2)  return result  def f2\_4(x):  n = len(x)  I2 = [i for i in range(2, n+1) if i % 2 == 0]  denominator = len(I2)  result = 1 - np.sqrt(x[0])  y\_values = [yi\_4(i, x) for i in I2]  sum\_y\_squared = sum([y \*\* 2 for y in y\_values])  cos\_multiplication=1  for i in I2:  cos\_multiplication = cos\_multiplication \* np.cos((20 \* yi\_4(i,x) \* np.pi)/ np.sqrt(i))  result = result + (2/denominator)\*(4\*sum\_y\_squared - 2\*cos\_multiplication + 2)  return result  # Example usage x4=[] x4.append(random.uniform(0.001, 1)) for i in range(9):  x4.append(random.uniform(-1, 1)) print("f1:", f1\_4(x4)) print("f2:", f2\_4(x4)) |
| --- |

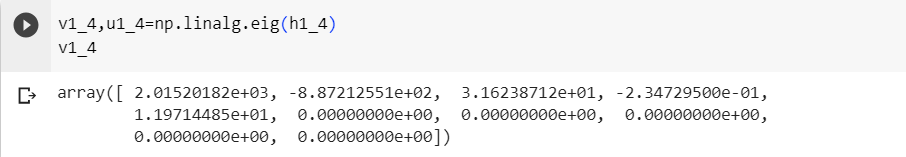
***F1 Solution***

* *Gradient and hessian*

| def grad\_hess\_f1\_4(x):  f0=f1\_4(x)  n=len(x)  h= 10\*\*(-5)  b=np.zeros(n)  g=np.zeros(n)  H=np.zeros((n,n))  for i in range(0,n):  x1=x.copy()  x1[i]=x1[i]+h  b[i]=f1\_4(x1)  g[i]=(f1\_4(x1)-f0)/h  for i in range(0,n):  for j in range(0,n):  x2=x.copy()  x2[i]=x2[i]+h  x2[j]=x2[j]+h  H[i][j]=(f1\_4(x2)-b[i]-b[j]+f0)/h\*\*2  # solver.qp()  return g,H |
| --- |



* *Eigen values*

**

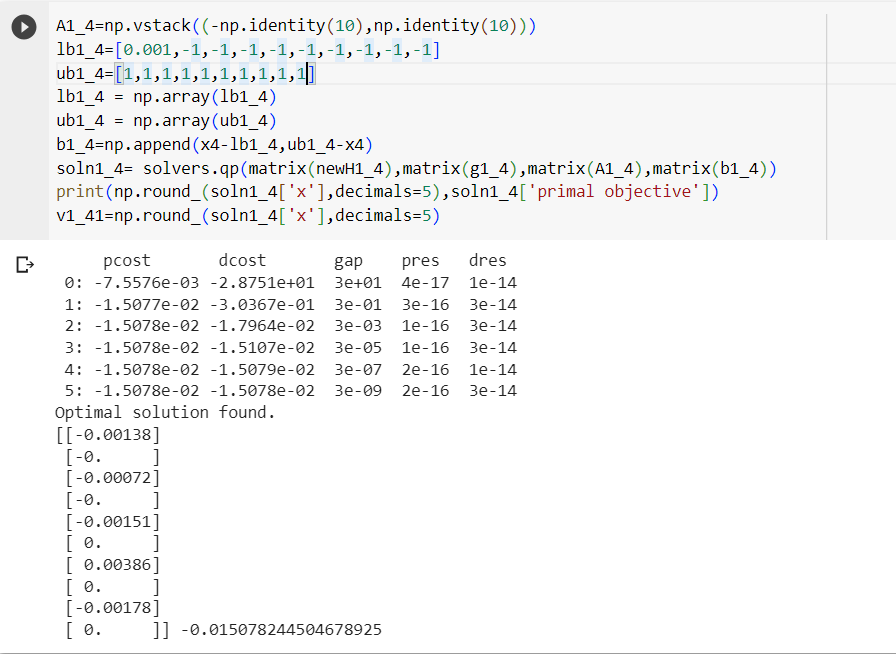
*not positive definite matrix*

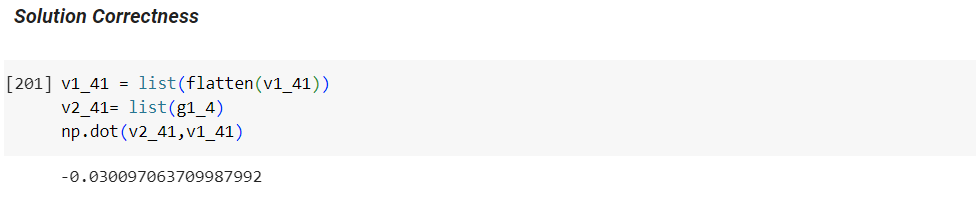
* *New hessian*

| *newH1\_4 = h1\_4@h1\_4 + (0.001-min(v1\_4))\*np.identity(10, dtype = float) np.linalg.eig(newH1\_4)* |
| --- |

*All eigens are positive*

* *Optimal Solution*

**

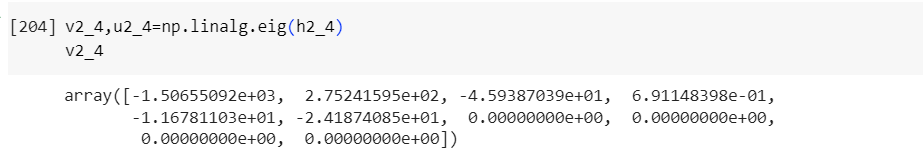
**

***F2 Solution***

* *Gradient and Hessian*

| *def grad\_hess\_f2\_4(x):  f0=f2\_4(x)  n=len(x)  h= 10\*\*(-5)  b=np.zeros(n)  g=np.zeros(n)  H=np.zeros((n,n))  for i in range(0,n):  x1=x.copy()  x1[i]=x1[i]+h  b[i]=f2\_4(x1)  g[i]=(f2\_4(x1)-f0)/h  for i in range(0,n):  for j in range(0,n):  x2=x.copy()  x2[i]=x2[i]+h  x2[j]=x2[j]+h  H[i][j]=(f2\_4(x2)-b[i]-b[j]+f0)/h\*\*2  # solver.qp()  return g,H g2\_4,h2\_4 = grad\_hess\_f2\_4(x4) print(g2\_4) print(h2\_4)* |
| --- |

* *Eigen values*

**

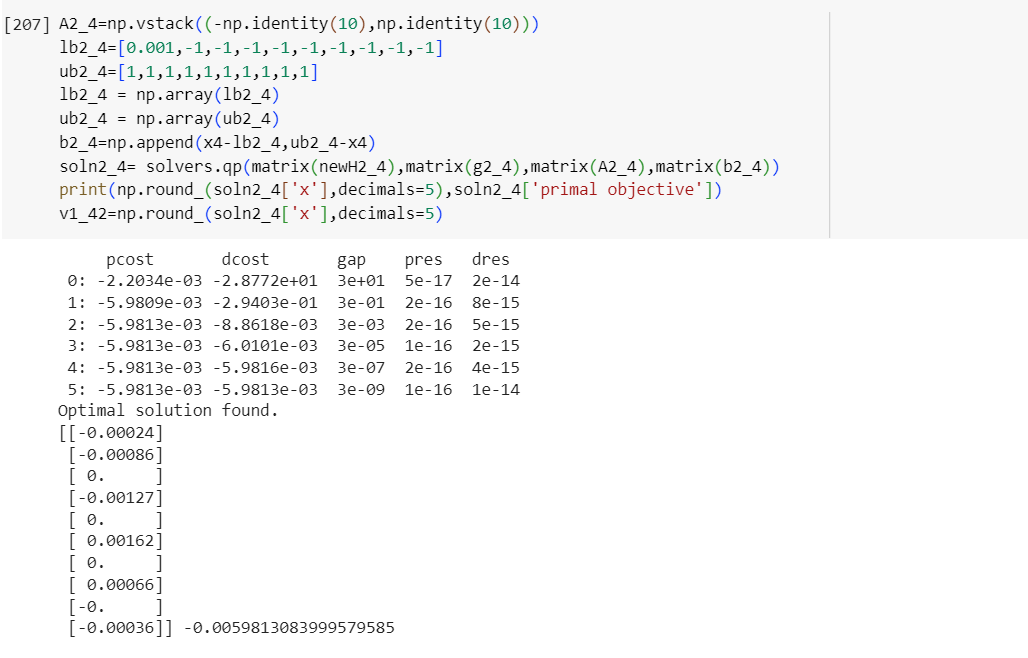
*Not positive definite*

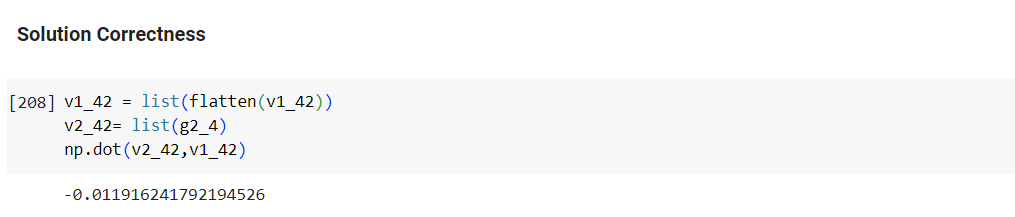
* *FInd new hessian*

| *newH2\_4 = h2\_4@h2\_4 + (0.001-min(v2\_4))\*np.identity(10, dtype = float) np.linalg.eig(newH2\_4)* |
| --- |

*All eigen values are positive and hence find optimal solution*

* *Optimal Solution*

**

**