MAE 598 / HW - 4

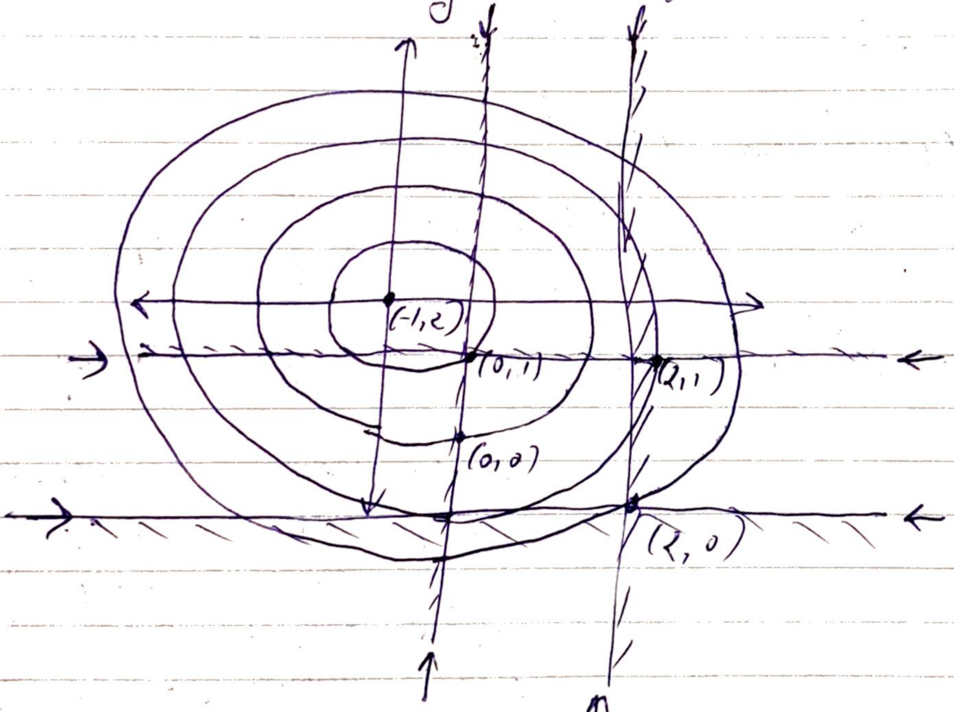
Name :- Dhiram Buch

P1) Sketch graphically the problem :-

$$\min f(x) = (x_1+1)^2 + (x_2-2)^2$$

subject to $g_1 = x_1 - 2 \le 0$, $g_3 = -x_1 < 0$

$\Rightarrow$ $(x_1+1)^2 + (x_2-2)^2$ being a circle equation with center $(-1, 2)$



where $L = (x_1+1)^2 + (x_2-2)^2 + \mu_1(x_2-2) + \mu_2(x_1-1) + \mu_3(-x_1) + \mu_4(-x_2)$

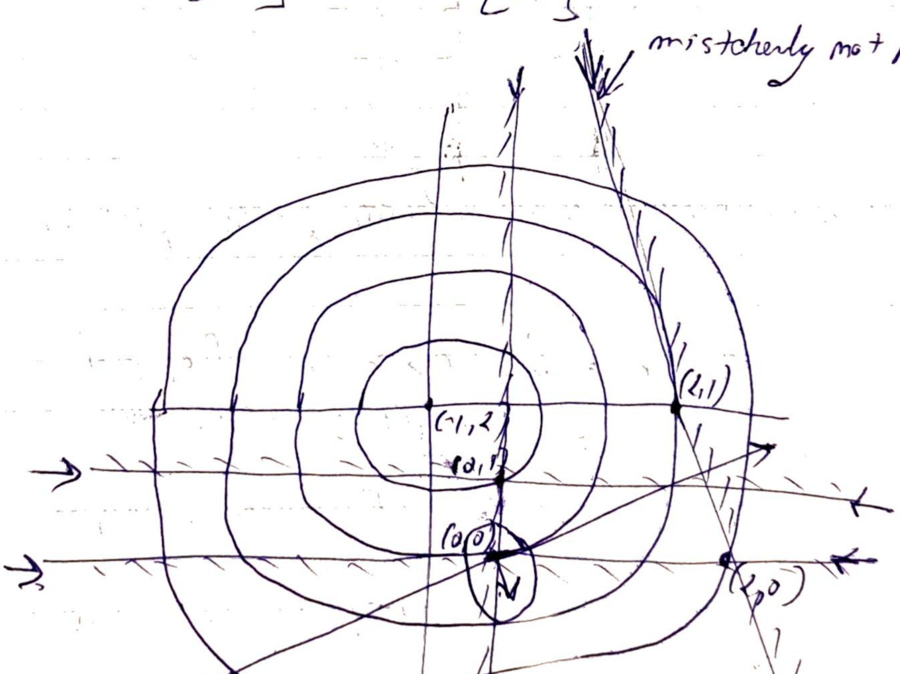
Conditions for μ are as follows:

if $x_2 - 2 > 0$, then $\mu_1 > 0$
if $x_2 - 1 < 0$, then $\mu_2 = 0$
if $x_1 - 1 = 0$, then $\mu_2 > 0$
if $x_1 - 2 < 0$, then $\mu_2 = 0$
if $-x_1 = 0$, then $\mu_3 > 0$
if $-x_1 < 0$, then $\mu_3 = 0$
if $-x_2 = 0$, then $\mu_4 > 0$
if $-x_2 < 0$, then $\mu_4 = 0$

for point $(0,0)$ the active constant are g_3 and g_4

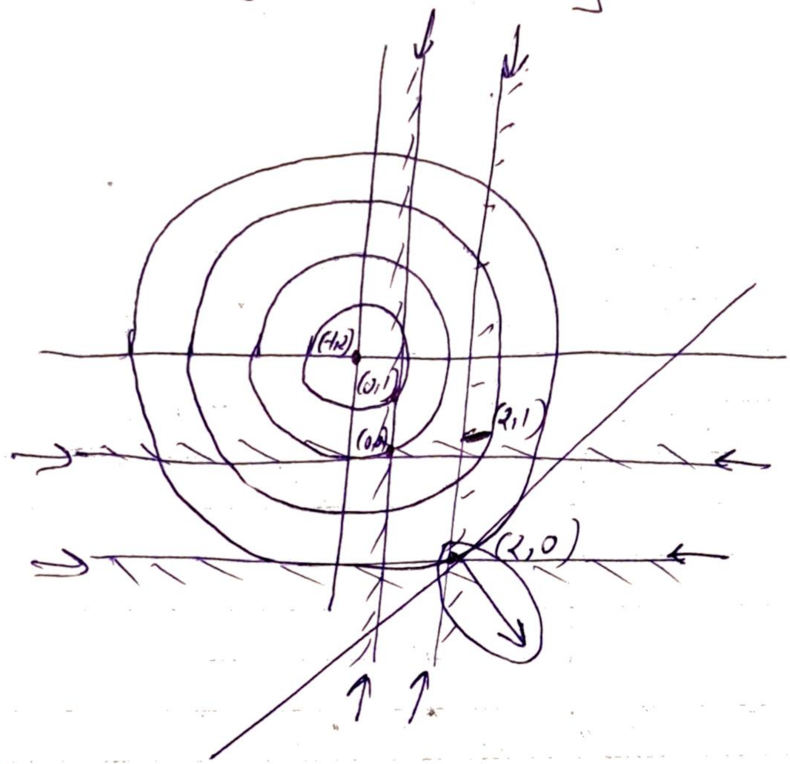
So $\nabla g_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$ & $\nabla g_4 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$

mistakenly not parallel



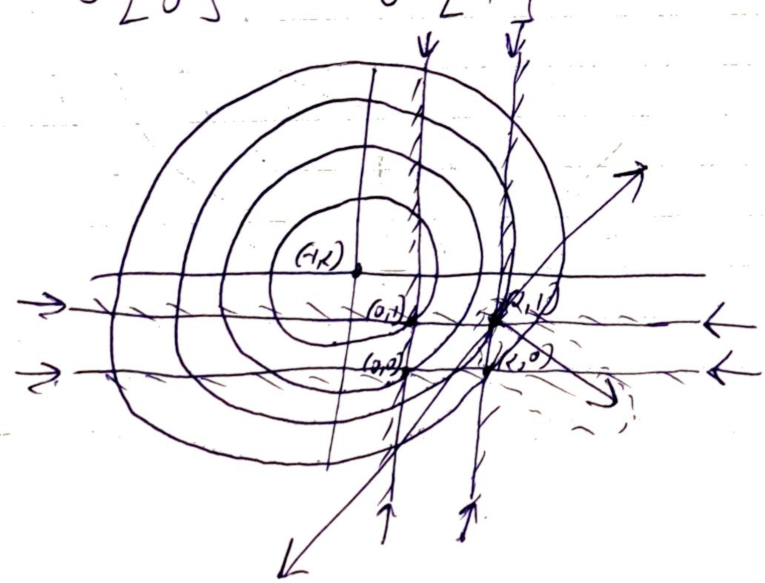
→ for point (2, 0) the active constraints are g_1 and g_4

So $\nabla g_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\nabla g_4 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$



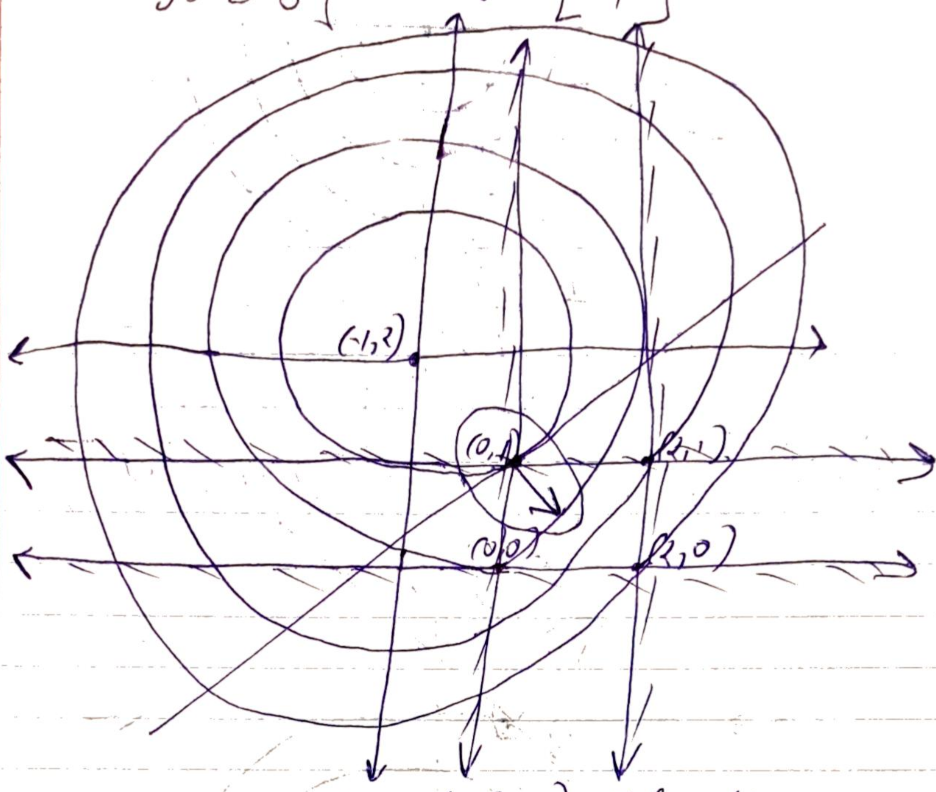
for point (2, 1), the active constraints being g_1 & g_2

So $\nabla g_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\nabla g_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$



for point $(0,1)$, the active constraints being g_3 & g_2

where $\nabla g_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$ & $\nabla g_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$



as we can see at point $(0,1)$ all direction are ascent and there is no feasibility for descent direction

Hence the $x_* = (0,1)^T$ is the minimizer
We can also check $(0,1)$ point by Applying the KKT necessary and sufficient condition

→ Necessary conditions:

* The g_3 and g_2 are the active constraints mean μ_3 & $\mu_2 \geq 0$ and μ_1 and μ_4 equal to zero

$$\nabla f - \mu^T \nabla g = 0$$

$$\begin{bmatrix} 2(x_1+1) \\ 2(x_2-2) \end{bmatrix} + \begin{bmatrix} -\mu_3 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 2(0+1) \\ 2(1-2) \end{bmatrix} + \begin{bmatrix} -\mu_3 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 2-\mu_3 \\ -2+\mu_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

we get $\mu_3 = 2$, $\mu_2 = 2$ which satisfies the KKT necessary condition.

→ Sufficient condition:

The Hessian of Lagrangian $= \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, $\lambda_1 = 2$ $\lambda_2 = 2$

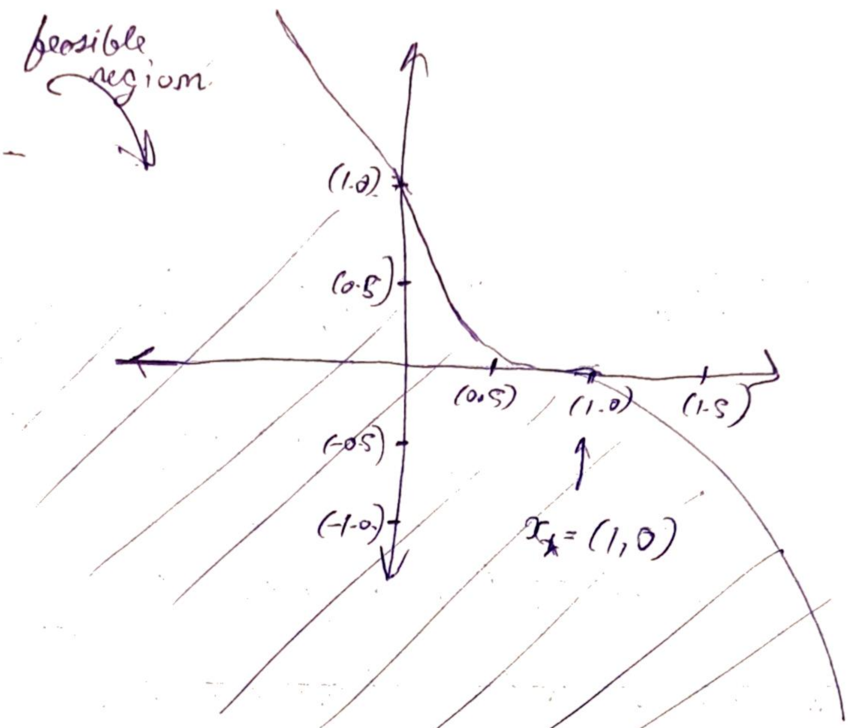
Here, Hessian of Lagrangian is positive definite everywhere. Therefore

$x_* = (0, 1)^T$ is the global minima.

P2) Graph the problems

$$\min f = x_1$$

subject to: $g_1 = x_2 - (1-x_1)^3 \leq 0$ and $x_2 \geq 0$



We can see at point $x_* = (1,0)^T$ is solution

Checking the KKT conditions at $x_* = (1,0)^T$

$$\mathcal{L} = -x_1 + \mu_1(x_2 - (1-x_1)^3) + \mu_2(-x_2)$$

→ Necessity condition :

The g_1 & g_2 are the active constraints mean μ_1 & $\mu_2 > 0$

$$\nabla f - \mu^T g = 0$$

$$\begin{bmatrix} -1 \\ 0 \end{bmatrix} + \mu_1 \begin{bmatrix} 3(1-x_1)^2 \\ 1 \end{bmatrix} + \mu_2 \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

at $x_* = (1,0)$,

$$\begin{bmatrix}-1\\0\end{bmatrix} + \mu_1\begin{bmatrix}3(1-1)^2\\1\end{bmatrix} + \mu_2\begin{bmatrix}0\\-1\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

$$\begin{bmatrix}-1\\0\end{bmatrix} + \mu_1\begin{bmatrix}0\\1\end{bmatrix} + \mu_2\begin{bmatrix}0\\-1\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

$-1 = 0$ contradicts the solution

Hence $\mu_1 - \mu_2 = 0$

So, the points $x_* = (1,0)$ is not a KKT point because this is not a regular point.

P3) Find a local solution to

$$\min f = x_1^2 + x_2^2 + x_3^2$$

subject to $h_1 = x_1^2/4 + x_2^2/5 + x_3^2/25 - 1 = 0$

and $h_2 = x_1 + x_2 - x_3 = 0$

by implementing generalized reduced gradient algorithms

$\Rightarrow$ Solving it using Lagrangian Method :-

$$\mathcal{L} = -f + \lambda h$$

$$\mathcal{L} = -(x_1x_2 + x_1x_3 + x_1x_3) + \lambda(x_1 + x_2 + x_3 - 3)$$

$$\nabla_x \mathcal{L} = \begin{bmatrix}-x_2 - x_3 + \lambda\\ -x_1 - x_2 + \lambda\\ -x_1 - x_3 + \lambda\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$\nabla_\lambda \mathcal{L} = x_1 + x_2 + x_3 = 0$$

we have 4 unknown and 4 equation so by solving the system of linear equation

$x_1 = 1 \quad x_2 = 1 \quad x_3 = 1 \quad \lambda = 2$

→ Check sufficient condition

The hessian ~~language~~ of Lagrangian $L_{xx} = \begin{bmatrix} 0 & -1 & -1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{bmatrix}$

The eigen value of hessian of lagrangian is $\lambda_1 = -2, \lambda_2 = 1 \; \lambda_3 = 1$

we can come to conclusion all the eigen value of Hessian are not positive.

→ But if we check the second order condition which is, $dx^T L_{xx} dx$

where, $dx^T L_{xx} dx$ being second order perturbation

$$dx^T L_{xx} dx = [\partial x_1 \; \partial x_2 \; \partial x_3] \begin{bmatrix} 0 & -1 & -1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \partial x_1 \\ \partial x_2 \\ \partial x_3 \end{bmatrix}$$

$$= -2\partial x_1 \partial x_2 - 2\partial x_1 \partial x_3 - 2\partial x_2 \partial x_3$$

→ We want the ∂x to be feasible so the feasible perturbation is such that $\frac{\partial h}{\partial x} \partial x = 0$.

$$\left[\frac{\partial h}{\partial x_1}, \frac{\partial h}{\partial x_2}, \frac{\partial h}{\partial x_3}\right]\begin{bmatrix}\partial x_1\\ \partial x_2\\ \partial x_3\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$[1, 1, 1]\begin{bmatrix}\partial x_1\\ \partial x_2\\ \partial x_3\end{bmatrix} = 0 \rightarrow \partial x_1 + \partial x_2 + \partial x_3 = 0$$

$$\partial x_1 = -\partial x_2 - \partial x_3$$

So,

$$= -2(-\partial x_2 - \partial x_3)\partial x_2 - 2(-\partial x_2 - \partial x_3)\partial x_3 - 2\partial x_2 \partial x_3$$

$$= 2(\partial x_2^2 + \partial x_2 \partial x_3 + \partial x_3^2)$$

$$= 2\left(\partial x_2 + \frac{1}{2}\partial x_3\right)^2 + \frac{3}{4}\partial x_3^2 > 0$$

→ furthermore $\partial x^T L_{xx} \partial x$ to be 0, ∂x_2 and ∂x_3 must be 0, if so then ∂x_1 is also 0.

→ which mean $\partial x = 0$, which is not a perturbation. Therefore $\partial x^T L_{xx} \partial x > 0$ for any non zero feasible perturbation.

→ So $x_{1*} = x_{2*} = x_{3*} = 1$ is global maxima for original problem

and if we plug this value in the main f so, $f_* = 3$.

```
import numpy as np
import torch
from torch.autograd import Variable
import matplotlib.pyplot as plt
from numpy.linalg import inv


X = Variable(torch.tensor([0.1000, 2.0217, 2.1217]), requires_grad=True)
Xc1 = Variable(torch.tensor([0.1000, 2.0217, 2.1217]), requires_grad=True)
Xc2 = Variable(torch.tensor([0.1000, 2.0217, 2.1217]), requires_grad=True)


X = Variable(torch.tensor([0.1,1.,1.]), requires_grad=True)
Xc1 = Variable(torch.tensor([0.1,1.,1.]), requires_grad=True)
Xc2 = Variable(torch.tensor([0.1,1.,1.]), requires_grad=True)


def objective_function(X):
  obj = (X[0]**2 + X[1]**2 + X[2]**2)
  #obj_sum =  obj.sum()
  return obj

def constraints_1(Xc1):
  c1 = (Xc1[0]**2)/4 + (Xc1[1]**2)/5 + (Xc1[2]**2)/25 - 1
  return c1
def constraints_2(Xc2):
  c2 = Xc2[0] + Xc2[1] - Xc2[2]
  return c2


obj = objective_function(X)
cont1 = constraints_1(Xc1)
cont2 = constraints_2(Xc2)
obj.backward()
cont1.backward()
cont2.backward()


def reduced_gradient():
  gradient = X.grad.numpy()
  gradient_c1 = Xc1.grad.numpy()
  gradient_c2 = Xc2.grad.numpy()

  df_dd = gradient[0]
  df_ds = np.array([gradient[1],gradient[2]])
  dh_ds = np.matrix([[gradient_c1[1],gradient_c1[2]],[gradient_c2[1],gradient_c2[2]]])
  dh_dd = np.array([gradient_c1[0],gradient_c2[0]])

  temp  = np.matmul(inv(dh_ds),dh_dd.transpose())
  temp2 = np.matmul(df_ds,temp.transpose())
  reduced_grad = df_dd - temp2
  return [reduced_grad,df_dd,df_ds,dh_ds,dh_dd]


def Leven_Marqt(X):
```

```
    max_iter = 50
    iters = 0
    Lambda = 1.
    tolerance = 1e-06
    h = torch.tensor([constraints_1(X),constraints_2(X)])
    normal = torch.norm(h)
    while normal > tolerance and iters < max_iter:
        reduced_grad,df_dd,df_ds,dh_ds,dh_dd = reduced_gradient()
        with torch.no_grad():
            # Newton Method
            temp  = np.matmul(dh_ds.transpose(),dh_ds)
            temp2 = temp + Lambda*torch.eye(2).numpy()
            temp3 = np.matmul(inv(temp2),dh_ds.transpose())
            temp4 = np.matmul(temp3,h)
            print(f'\ndelta = {temp4}')
            X[1:] = X[1:] - temp4
            print(f'\nX = {X}')
            iters += 1
        normal = torch.norm(torch.tensor([constraints_1(X),constraints_2(X)]))
    return X
```

```
Leven_Marqt(X)
```

```
X = tensor([0.1000, 6.9643, 6.7591], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 7.1299, 6.9190], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 7.2956, 7.0790], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 7.4613, 7.2390], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 7.6269, 7.3990], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 7.7926, 7.5589], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 7.9583, 7.7189], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 8.1240, 7.8789], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 8.2896, 8.0389], requires_grad=True)
```

```
delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 8.4553, 8.1988], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 8.6210, 8.3588], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 8.7867, 8.5188], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 8.9523, 8.6788], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 9.1180, 8.8387], requires_grad=True)

delta = tensor([-0.1657, -0.1600])

X = tensor([0.1000, 9.2837, 8.9987], requires_grad=True)
```

```
def line_search(X):
  counter = 0
  alpha = 1.
  all_alpha =[1]
  b = 0.5
  t = 0.5
  def check(alpha):
    [reduced_grad,df_dd,df_ds,dh_ds,dh_dd] = reduced_gradient()
    X_c = X.clone().detach()
    X_c = X_c.numpy()

    d_new = X_c[0] - alpha*reduced_grad
    temp  = np.matmul(inv(dh_ds),dh_dd.transpose())
    temp2 = reduced_grad*temp
    s_new = [X_c[1],X_c[2]] + alpha*(temp2)
    X_alpha = [d_new,s_new[0,0],s_new[0,1]]
    f_alpha = objective_function(X_alpha)
    #phi_alpha = objective_function(X_c) - alpha*t*(np.matmul(df_dd,df_dd.transpose()))
    phi_alpha = objective_function(X_c) - alpha*t*(reduced_grad**2)
    return [f_alpha,phi_alpha]

  while check(alpha)[0] > check(alpha)[1] and counter < 25:
    counter += 1
    alpha = b*alpha
    all_alpha.append(alpha)

  print(all_alpha)
  return alpha


line_search(X)
```

```
[1]
1.0
```

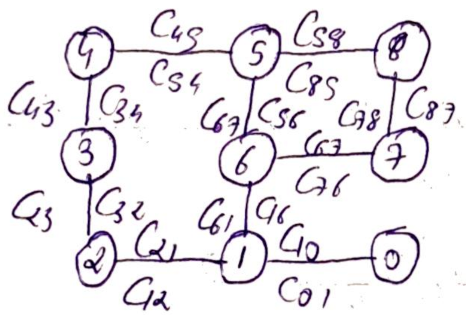
```
max_iter = 10
epsilon = 1e-3
for i in range(0,max_iter):
  Xc1 = X
  Xc2 = X
  obj = objective_function(X)
  cont1 = constraints_1(Xc1)
  cont2 = constraints_2(Xc2)
  obj.backward()
  cont1.backward()
  cont2.backward()
  if reduced_gradient()[0] > epsilon:
    alpha = line_search(X)
```

i'5) The problem formulation is :-

$$\min \sum_{ij}^{N} (x_{ij} \, C_{ij})$$

where, x_{ij} is movement from node $i \to j$

Below diagram better explains all the parameters involved in problem :-



for forward movement $x_{ij} = \begin{cases} 1 & \text{if } i \text{ connect with } j \\ 0 & \text{if } i \text{ not connect with } j \end{cases}$

backward movement $x_{ji} = \begin{cases} 1 & \text{if } i \text{ connect with } j \\ 0 & \text{if } i \text{ not connect with } j \end{cases}$

where c_{ij} is cost of moving from node i to j

cost of forward move is :- $x_{ij} = \begin{cases} C_{ji} & \text{if } i \text{ connect with } j \\ \infty & \text{if } i \text{ not connect with } j \end{cases}$

cost of backward move is :- $x_{ji} = \begin{cases} C_{ji} & \text{if } i \text{ connect with } j \\ \infty & \text{if } i \text{ not connect with } j \end{cases}$

→ The constraints of objective function as follows:

$\sum x_{ij} \geq N$: The truck needs to visit all the nodes where N is the number of Nodes.

→ Traffic control: $\sum x_{ij} = \sum x_{ji}$

as time in = times out

There must be a connection between starting to atleast one neighbour node.

for starting $\sum x_{0j} \geq 1 \ \forall j$;; for ending $\sum x_{j0} \geq 1 \ \forall j$

Hence the final problem is:

$$\boxed{\begin{aligned} &\min_{\{s.t\}} \sum_{ij}^{N} (x_{ij} c_{ij}) \\ &\sum x_{ij} \geq N \\ &\sum x_{ij} = \sum x_{ji} \\ &\sum x_{0j} \geq 1, \forall j \\ &\sum x_{j0} \geq 1, \forall j \end{aligned}}$$