MAE 598 / HW = 4

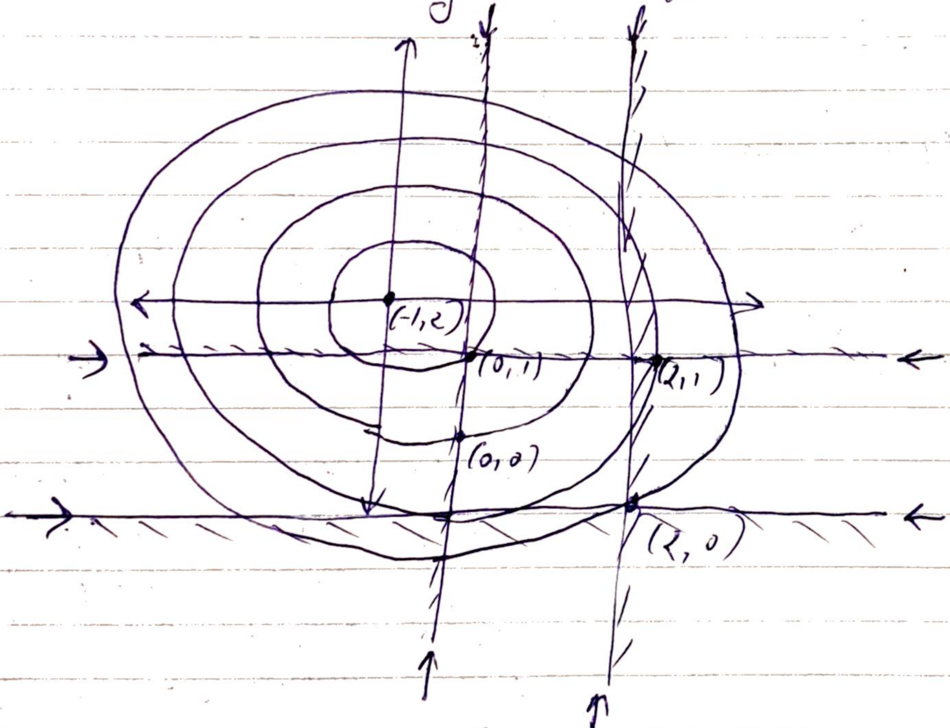
Name :- Dhriam Buch

P1) Sketch graphically the problem :-

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

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

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



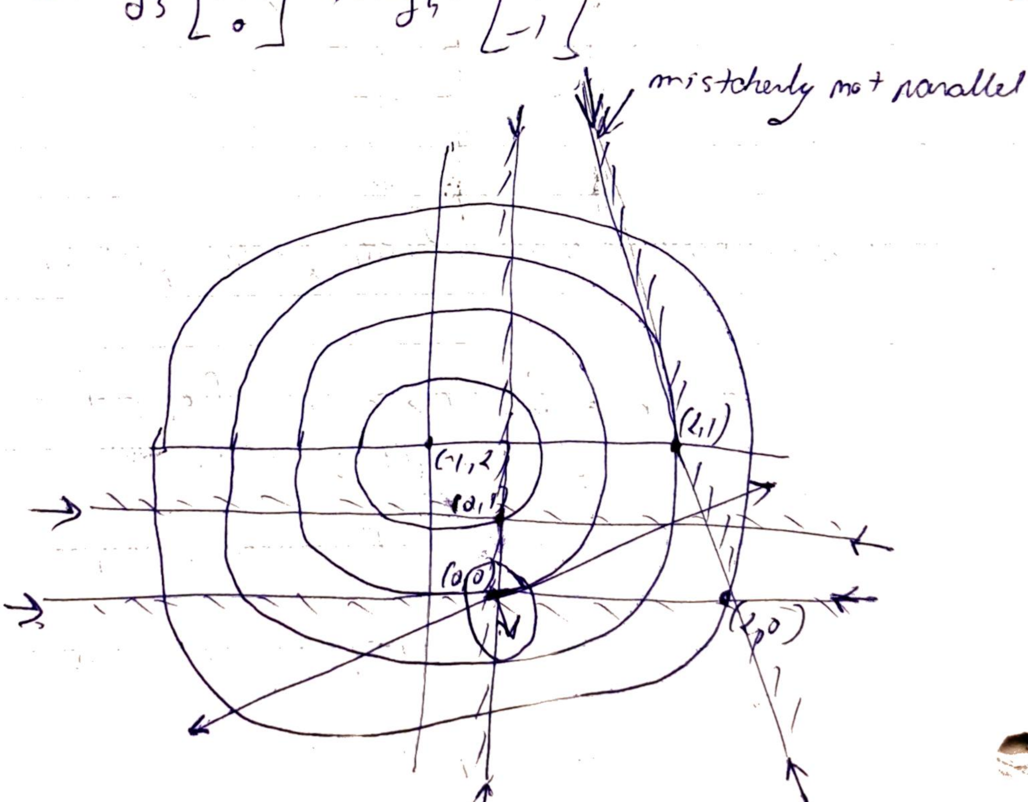
where $L = (x_1+1)^2 + (x_2-2)^2 + \mu_1(x_2-2) + \mu_2(x_1-1) + \mu_3(-x_1) + \mu_4(-x_2)$

Conditions for μ are as follows:-

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

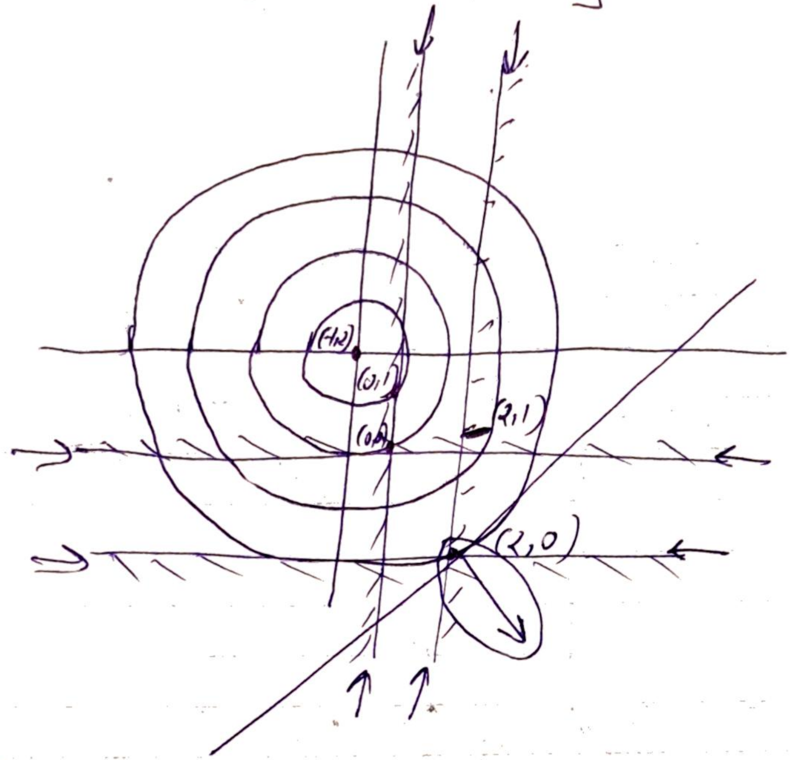
for point $(0,0)$ the active constant are g_3 and g_4

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



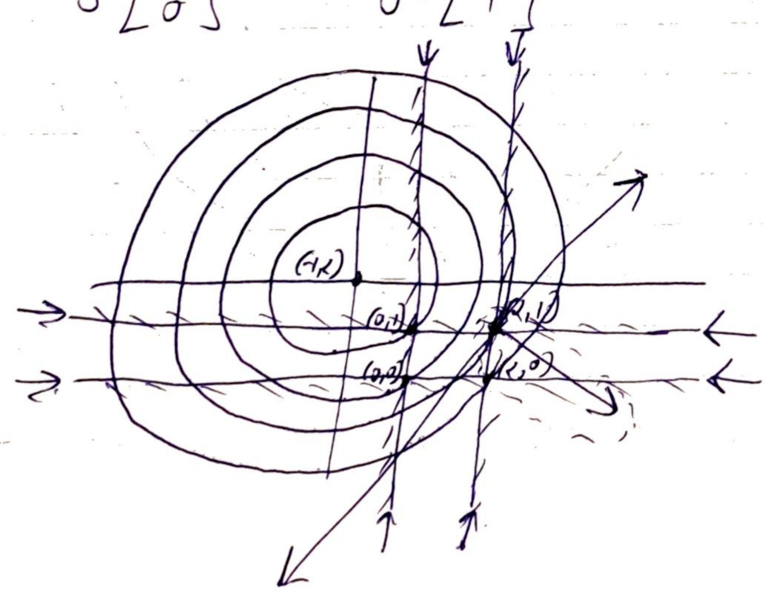
→ for point (2, 0) the active constraints are g_1 and g_4

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



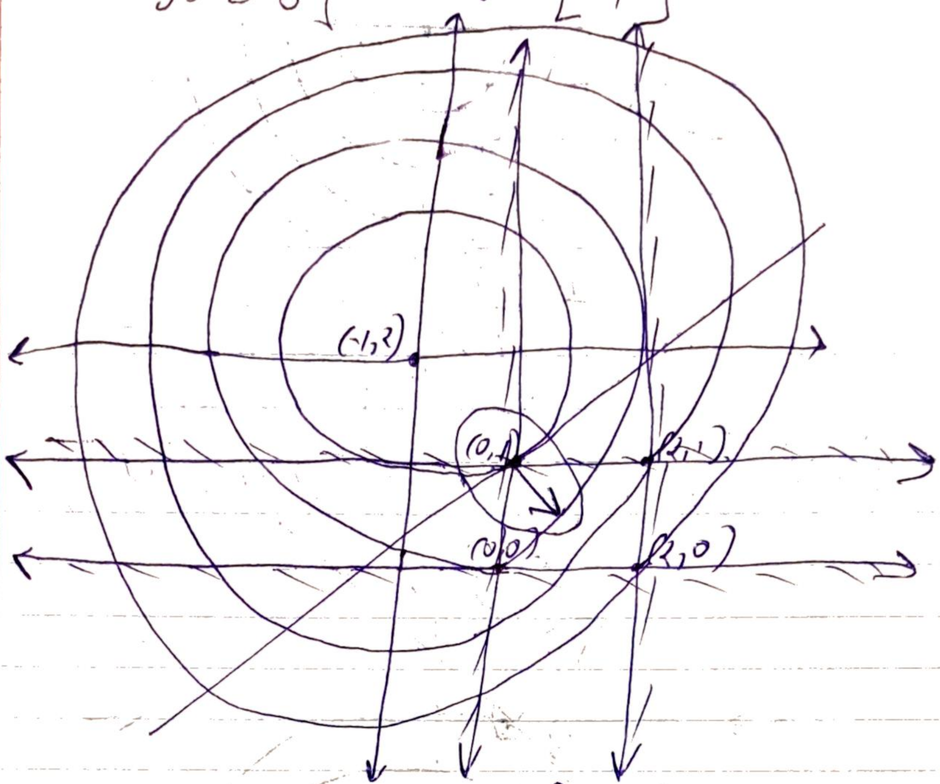
for point (2, 1), the active constraints being g_1 & g_2

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.

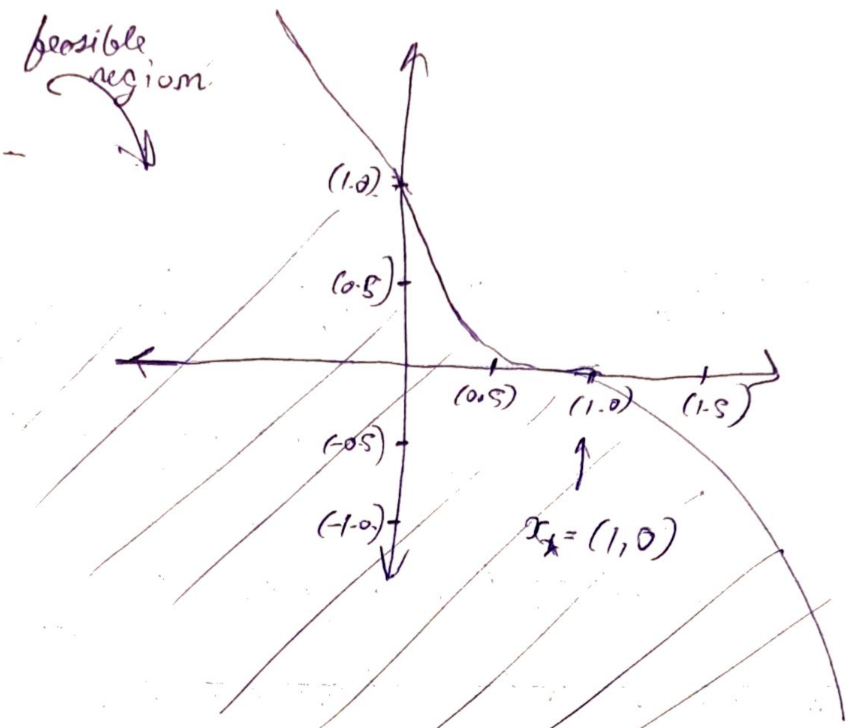
→ Sufficient condition:

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

⇒ Solving it using Lagrangian Method :-

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

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

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

$$\nabla_\lambda L = x_1 + x_2 + x_3 = 0$$

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 $\frac{\partial h}{\partial x} \partial x = 0$.

$$\begin{bmatrix} \frac{\partial h}{\partial x_1}, & \frac{\partial h}{\partial x_2}, & \frac{\partial h}{\partial x_3} \end{bmatrix} \begin{bmatrix} \partial x_1 \\ \partial x_2 \\ \partial x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

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

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

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 than ∂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 dependencies
import numpy as np
from matplotlib import pyplot as plt
import torch
import torch.nn as nn
from torch.autograd import Variable
from torch.autograd.functional import jacobian
from matplotlib import pyplot as plt, rc
from matplotlib.pyplot import figure
```

```
#defining constraints and objective function

Function = lambda X: ((X[0] ** 2) + (X[1] ** 2) + (X[2] ** 2))
Constraint1_h1 = lambda X: (((X[0] ** 2) / 4) + ((X[1] ** 2) / 5) + ((X[2] ** 2) / 25)
Constraint2_h2 = lambda X: (X[0] + X[1] - X[2])
X = Variable(torch.tensor([1.,1.,1.]), requires_grad=True)
eps= 1e-03
```

we have 3 equations and 2 constraints. Hence, n = 3 and m =2. Therefore, the degree of freedom (d.o.f) = n - m = 3 - 2 = 1 Based on the values of d.o.f, m and n, we can conclude that the their is 1 decision variable and 2 state variables. Decision Variable (d) = x1 State Variable (s) = [x2, x3]

```
#reduced gradient
# using jacobian
def Reduced_Gradient_Calc(f, h1, h2, X):

  Jacobian = torch.zeros((3, 3))
  Jacobian[0] = jacobian(f, (X))
  Jacobian[1] = jacobian(h1, (X))
  Jacobian[2] = jacobian(h2, (X))

  df_dd = Jacobian[0, 0]   # del 'f' by del 'd'
  df_ds = Jacobian[0,1:]
  dh_ds = Jacobian[1:,1:]
  dh_dd = Jacobian[1:,0]

  reduced_gradient = df_dd - torch.matmul(torch.matmul(df_ds, torch.pinverse(dh_ds)), c
  return reduced_gradient, df_dd, df_ds, dh_ds, dh_dd
```

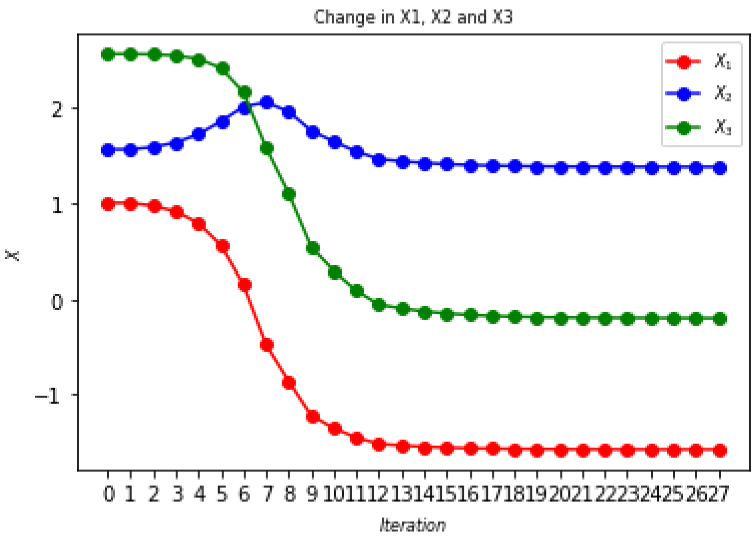
```
#Levenberg-Marquardt and Newtons method o solve constraints
def Constraint_Solver(X):
    Lambda = 1.
    normal_error = torch.norm(torch.tensor([Constraint1_h1(X), Constraint2_h2(X)]))
    while normal_error > 1e-06:
        reduced_gradient, df_dd, df_ds, dh_ds, dh_dd = Reduced_Gradient_Calc(Function,
        with torch.no_grad():
            X[1:] = X[1:] - torch.matmul(
                torch.matmul(torch.pinverse(torch.matmul(dh_ds.T, dh_ds) + Lambda * tor
                torch.tensor([Constraint1_h1(X), Constraint2_h2(X)]))
```

```
        normal_error = torch.norm(torch.tensor([Constraint1_h1(X), Constraint2_h2(X)]))
    return X

#def line search algo

def Updater(X, alpha):
    new_X = torch.zeros(3)
    reduced_gradient, df_dd, df_ds, dh_ds, dh_dd = Reduced_Gradient_Calc(Function, Cons
    new_X[0] = X[0] - alpha * reduced_gradient
    new_X[1:] = X[1:] + (alpha * (torch.matmul(torch.pinverse(dh_ds), dh_dd)) * reducec
    return new_X

def lineSearch(X):
    t = 0.5
    counter = 25
    reduced_gradient, df_dd, df_ds, dh_ds, dh_dd = Reduced_Gradient_Calc(Function, Cons
    alpha = 1
    i = 0
    func = Function(Updater(X, alpha))
    phi = Function(X) - (t * alpha * (reduced_gradient ** 2))
    while func > phi and i < counter:
        alpha = 0.5 * alpha
        func = Function(Updater(X, alpha))
        phi = Function(X) - (t * alpha * (reduced_gradient ** 2))
        i += 1
    return alpha


#generalized reduced gradient algorithm
def Generalized_Reduced_Gradient(X, eps=1e-03):
    X_val = X.detach().numpy()
    print(f'Initial value of X = {X_val}')
    X = Constraint_Solver(X)
    print(f'\nUsing the Constraint Solver to determine the feasible solution\nX_feasibl
    X_val = np.vstack((X_val, X.detach().numpy()))
    all_obj_func_values = [Function(X).item()]
    alpha_val = [1]
    reduced_gradient, df_dd, df_ds, dh_ds, dh_dd = Reduced_Gradient_Calc(Function, Cons
    error_value = torch.norm(reduced_gradient)
    all_error_values = [error_value]
    iterations = 0
    while error_value > eps:
        alpha = lineSearch(X)  # step 4.1
        reduced_gradient, df_dd, df_ds, dh_ds, dh_dd = Reduced_Gradient_Calc(Function,

        with torch.no_grad():
            X[0] = X[0] - alpha * reduced_gradient  # setp 4.2
            X[1:] = X[1:] + (alpha * np.matmul(torch.pinverse(dh_ds), dh_dd) * reduced_

        X = Constraint_Solver(X)  # step 4.4
        error_value = torch.norm(reduced_gradient)  # step 4.5

        # record values
        X_val = np.vstack((X_val, X.detach().numpy()))
        all_obj_func_values.append(Function(X).item())
        alpha_val.append(alpha)
```

```
        alpha_val.append(alpha)
        all_error_values.append(error_value)
        iterations += 1
    return X_val, all_obj_func_values, alpha_val, all_error_values, iterations


X_val, objFun_val, alpha_val,error_Val,k = Generalized_Reduced_Gradient(X)
```

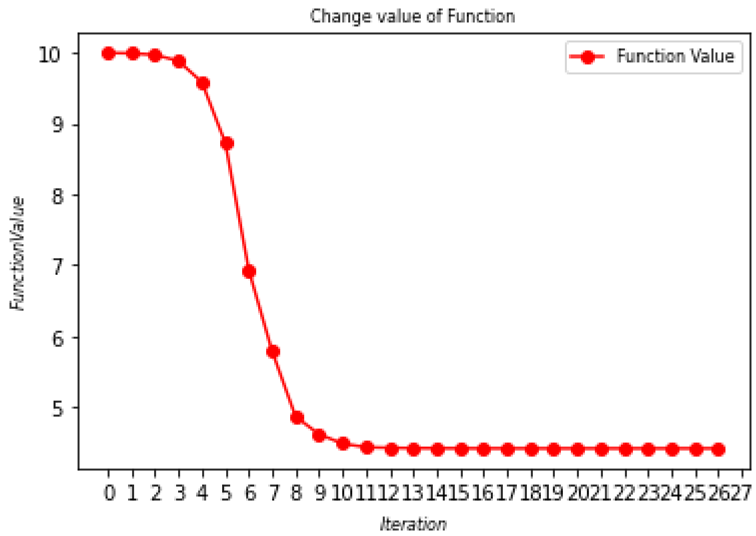
```
Initial value of X = [1. 1. 1.]

Using the Constraint Solver to determine the feasible solution
X_feasible = tensor([1.0000, 1.5614, 2.5614], requires_grad=True)
```

```
# Plots


print("\n")
figure(figsize = (6,4))
plt.plot(X_val[:,0],'ro-')
plt.plot(X_val[:,1],'bo-')
plt.plot(X_val[:,2],'go-')
plt.legend(["$X_1$" ,"$X_2$","$X_3$"],fontsize = 8)
plt.title('Change in X1, X2 and X3', fontsize = 8)
plt.xlabel("$Iteration$",fontsize = 8)
plt.ylabel("$X$",fontsize = 8)
plt.xticks(range(len(X_val[:,0])))
plt.show()
```

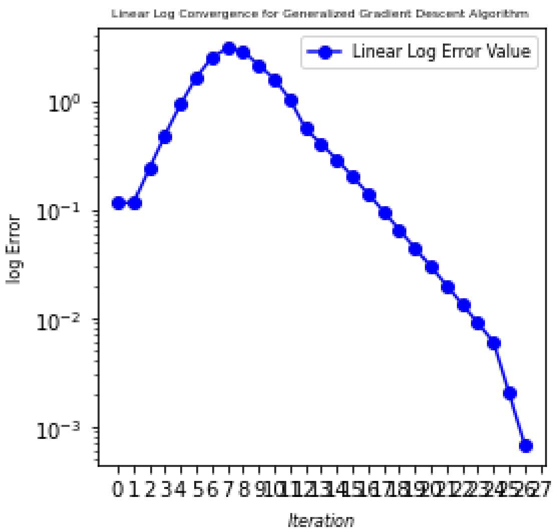


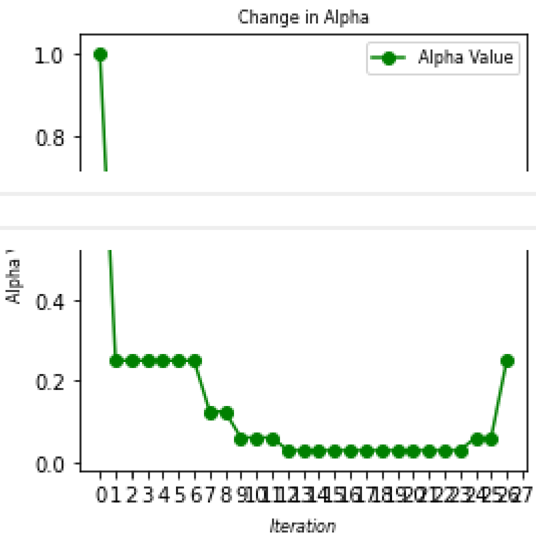

```
print("\n")
figure(figsize = (6,4))
plt.plot(objFun_val,'ro-')
plt.xlabel("$Iteration$",fontsize = 8)
plt.ylabel("$Function Value$",fontsize = 8)
```

```
plt.legend(["Function Value"],fontsize = 8)
plt.title('Change value of Function', fontsize = 8)
plt.xticks(range(len(X_val[:,0])))
plt.show()
```




```
print("\n")
figure(figsize = (4,4))
plt.plot(error_Val,'bo-')
plt.xlabel("$Iteration$",fontsize = 8)
plt.yscale("log")
plt.ylabel(r'log Error',fontsize = 8)
plt.legend(["Linear Log Error Value"],fontsize = 8)
plt.title('Linear Log Convergence for Generalized Gradient Descent Algorithm', fontsize
plt.xticks(range(len(X_val[:,0])))
plt.show()

print("\n")
figure(figsize = (4,4))
plt.plot(alpha_val,'go-')
plt.xlabel("$Iteration$",fontsize = 8)
plt.ylabel(r'Alpha Value',fontsize = 8)
plt.legend(["Alpha Value"],fontsize = 8)
plt.title('Change in Alpha', fontsize = 8)
plt.xticks(range(len(X_val[:,0])))
plt.show()
```


Linear Log Convergence for Generalized Gradient Descent Algorithm
Linear Log Error Value
log Error
Iteration

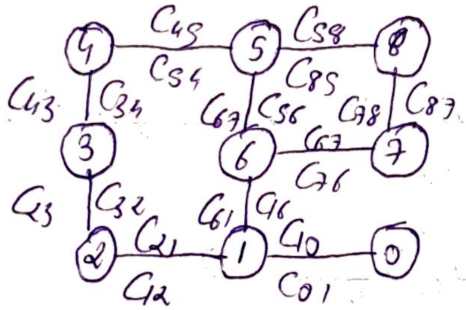

Change in Alpha
Alpha Value
Iteration

15) 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:

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