Design Optimization\MAE_HW3_Q4.py

```
import numpy as np
import sympy as sp


# Objective function
def objective_function(d,s):
    x_1=d
    x_2=s[0,0]
    x_3=s[1,0]

    return ((x_1**2)+(x_2**2)+(x_3**2))

# We have a function for each matrix that needs to be calculated
def dh_ds(h,s,s_val):
    dh1_ds1=sp.diff(h[0,0],s[0,0])
    dh1_ds1=dh1_ds1.subs(s[0,0],s_val[0,0])
    dh1_ds2=sp.diff(h[0,0],s[1,0])
    dh1_ds2=dh1_ds2.subs(s[1,0],s_val[1,0])
    dh2_ds1=sp.diff(h[1,0],s[0,0])
    dh2_ds1=dh2_ds1.subs(s[0,0],s_val[0,0])
    dh2_ds2=sp.diff(h[1,0],s[1,0])
    dh2_ds2=dh2_ds2.subs(s[1,0],s_val[1,0])
    dh_ds_final=np.array([[dh1_ds1,dh1_ds2],[dh2_ds1,dh2_ds2]])

    return dh_ds_final

def df_ds(f,s,s_val):
    df_ds1=sp.diff(f,s[0,0])
    df_ds1=df_ds1.subs(s[0,0],s_val[0,0])
    df_ds2=sp.diff(f,s[1,0])
    df_ds2=df_ds2.subs(s[1,0],s_val[1,0])
    df_ds_final=np.array([[df_ds1,df_ds2]])

    return df_ds_final

def dh_dd(h,d,d_val):
    dh1_dd=sp.diff(h[0,0],d)
    dh1_dd=dh1_dd.subs(d,d_val)
```

```
    dh2_dd=sp.diff(h[1,0],d)
    dh2_dd=dh2_dd.subs(d,d_val)
    dh_dd_final=np.array([[dh1_dd],[dh2_dd]])

    return dh_dd_final

def df_dd(f,d,d_val):
    df_dd1=sp.diff(f,d)
    df_dd1=df_dd1.subs(d,d_val)

    return df_dd1


# Linesearch algorithm is defined

def linesearch(dfdd,s_val,d_val,h,s,d):
    alp=1
    b=0.5
    t=0.3
    d_val_new=d_val-(alp*dfdd)
    par_dhds=dh_ds(h,s,s_val)
    par_dhdd=dh_dd(h,d,d_val)
    s_val_new=s_val+(alp*(np.matmul(np.linalg.inv(np.float64(par_dhds)),par_dhdd))*dfdd)
    f_alp=objective_function(d_val_new,s_val_new)
    phi_alp=objective_function(d_val,s_val) - (alp*t*(dfdd**2))
    j=0
    while f_alp > phi_alp:
        alp=alp*b
        d_val_new=d_val-(alp*dfdd)
        s_val_new=s_val+(alp*(np.matmul(np.linalg.inv(np.float64(par_dhds)),par_dhdd))*dfdd)
        f_alp=objective_function(d_val_new,s_val_new)
        phi_alp=objective_function(d_val,s_val) - (alp*t*(dfdd**2))
        j=j+1

    return alp


# Newton ralphson algorithm defined

def newton_ralpson(s_val,d_val,h_1,s):
    x1=d_val
```

```
    x2=s_val[0,0]
    x3=s_val[1,0]
    h1=((x1**2)/4)+((x2**2)/5)+((x3**2)/25)-1
    h2=x1+x2-x3
    h=np.array([[h1[0,0]],[h2[0,0]]],dtype=np.float64)
    j=0
    s_old=s_val
    h_norm=np.linalg.norm(h)
    while h_norm > 0.001 or j < 10:
        dhds=dh_ds(h_1,s,s_old)
        s_new=s_old-(np.matmul(np.linalg.inv(np.float64(dhds)),h))
        h= h + (np.matmul(dhds,(s_new-s_old)))
        s_old=s_new
        j=j+1
        h_norm=h[0,0]+ h[1,0]

    return s_new

# Using sympy we create the variables of our function

x1= sp.symbols('x1')
x2= sp.symbols('x2')
x3= sp.symbols('x3')

# Objective function and the constraint equations

f=(x1**2)+(x2**2)+(x3**2)
h1=((x1**2)/4)+((x2**2)/5)+((x3**2)/25)-1
h2=x1+x2-x3

# we create a vector of the constraint equations and the dependant variables

h=np.array([[h1],[h2]])
s=np.array([[x2],[x3]])

# Initial variables have been assigned ensuring that they satisfy the constraints

x0_d=1
x0_s=np.array([[1.56137],[2.56137]])

ephsilon=0.0001
```

```python

# finding the different partial derivatives

par_dhds=dh_ds(h,s,x0_s)
par_dfds=df_ds(f,s,x0_s)
par_dfdd=df_dd(f,x1,x0_d)
par_dhdd=dh_dd(h,x1,x0_d)

df_dd_main=par_dfdd-(np.matmul(np.matmul(par_dfds,np.linalg.inv(np.float64(par_dhds))),par_dhdd))

k=0
while df_dd_main > ephsilon or k < 50:

    alpha = linesearch(df_dd_main,x0_s,x0_d,h,s,x1)
    d_next=x0_d-(alpha*df_dd_main)
    s_next_temp=x0_s+(alpha*(np.matmul(np.linalg.inv(np.float64(par_dhds)),par_dhdd))*df_dd_main)
    s_next_real=newton_ralpson(s_next_temp,d_next,h,s)

    par_dhds=dh_ds(h,s,s_next_real)
    par_dfds=df_ds(f,s,s_next_real)
    par_dfdd=df_dd(f,x1,d_next[0,0])
    par_dhdd=dh_dd(h,x1,d_next[0,0])

    df_dd_main=par_dfdd-(np.matmul(np.matmul(par_dfds,np.linalg.inv(np.float64(par_dhds))),par_dhdd))

    k=k+1

print ('Dependant variable D')
print (d_next)
print ('State variable S')
print (s_next_real)
```

```
(my_packages) PS C:\Users\hbhavnag\Documents\Hussain\ASU\collision detection> python -u "c:\Users\hbhavnag\Documents\Hussain\ASU\collision detection\Design Optimization\MAE_HW3_Q4.py"
Dependant variable D
[[0.985699845719411]]
State variable S
[[1.57342610808005]
 [2.55912595379946]]
(my_packages) PS C:\Users\hbhavnag\Documents\Hussain\ASU\collision detection>
```