MAE 598: Design Optimization

Homework 4

Tanner Bitz

1. (10 points) Sketch graphically the problem

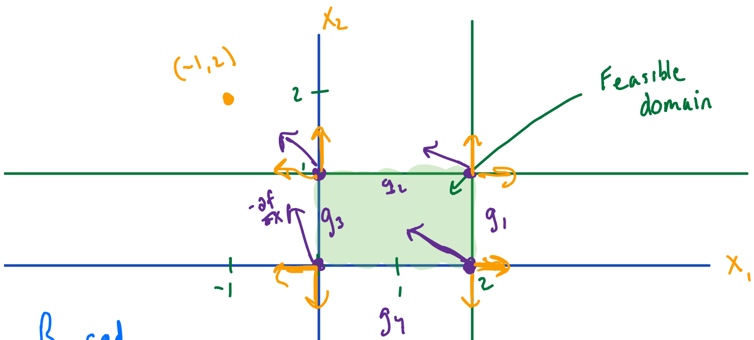
$x_1 \geq 0$
$x_1 \leq 2$

$$\min f(\mathbf{x}) = (x_1+1)^2 + (x_2-2)^2$$
$$\text{subject to } g_1 = x_1 - 2 \leq 0, \quad g_3 = -x_1 \leq 0,$$
$$g_2 = x_2 - 1 \leq 0, \quad g_4 = -x_2 \leq 0.$$

$x_2 \geq 0$
$x_2 \leq 1$

Find the optimum graphically. Determine directions of feasible descent at the corner points of the feasible domain. Show the gradient directions of f and g_is at these points. Verify graphical results analytically using the KKT conditions.

If f were unconstrained the min $= 0$ at $(-1,2)$



Based on the graph, the solution should be $(0,1)$.

$$\hat{g}_1 = x_1 - 2 = 0$$
$$\hat{g}_2 = x_2 - 1 = 0$$
$$\hat{g}_3 = -x_1 = 0$$
$$\hat{g}_4 = -x_2 = 0$$

$$\frac{\partial \hat{g}_1}{\partial x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$\frac{\partial \hat{g}_2}{\partial x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$\frac{\partial \hat{g}_3}{\partial x} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$
$$\frac{\partial \hat{g}_4}{\partial x} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$\frac{\partial f}{\partial x} = \begin{bmatrix} 2(x_1+1) \\ 2(x_2-2) \end{bmatrix}$$

$$-\frac{\partial f}{\partial x}\Big|_{(0,0)} = \begin{bmatrix} -2 \\ 4 \end{bmatrix} \qquad -\frac{\partial f}{\partial x}\Big|_{(2,0)} = \begin{bmatrix} -6 \\ 4 \end{bmatrix}$$

$$-\frac{\partial f}{\partial x}\Big|_{(0,1)} = \begin{bmatrix} -2 \\ 2 \end{bmatrix} \qquad -\frac{\partial f}{\partial x}\Big|_{(2,1)} = \begin{bmatrix} -6 \\ 2 \end{bmatrix}$$

KKT

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

$$\frac{\partial L}{\partial x_1} = 2(x_1+1) + \mu_1 - \mu_3 = 0$$

$$\frac{\partial L}{\partial x_2} = 2(x_2-2) + \mu_2 - \mu_4 = 0$$

if $x_1 - 2 = 0$ the $\mu_1 > 0$
else $x_1 - 2 < 0$ then $\mu_2 = 0$ ①

if $x_2 - 1 = 0$ then $\mu_2 \geq 0$
else $x_2 - 1 < 0$ then $\mu_2 = 0$ ②

if $-x_1 = 0$ then $\mu_3 \geq 0$
else $-x_1 < 0$ then $\mu_3 = 0$ ③

if $-x_2 = 0$ then $\mu_4 \geq 0$
else $-x_2 < 0$ the $\mu_4 = 0$ ④

Case #1

$\mu_1 > 0$ and $\mu_2 > 0$, $\mu_3 = 0$, $\mu_4 = 0$

$x_1 = 2$ $\quad 2(2+1) + \mu_1 - 0 = 0$

$x_2 = 1$ $\quad \mu_1 = -6$ ⇒ not possible

Case #2

$\mu_1 = 0, \quad \mu_2 > 0, \quad \mu_3 > 0, \quad \mu_4 = 0$

$x_1 = 0$

$x_2 = 1$

$$\frac{\partial L}{\partial x_1} = 2(0+1) + 0 - \mu_3 = 0$$

$$\mu_3 = 2 > 0 \quad \checkmark$$

$$\frac{\partial L}{\partial x_2} = 2(1-2) + \mu_2 - 0 = 0$$

$$\mu_2 = 2 > 0 \quad \checkmark$$

possible solution

Check For Convexity

Domain is convex

$$\frac{\partial f}{\partial x} = \begin{bmatrix} 2(x_1+1) \\ 2(x_2-2) \end{bmatrix}$$

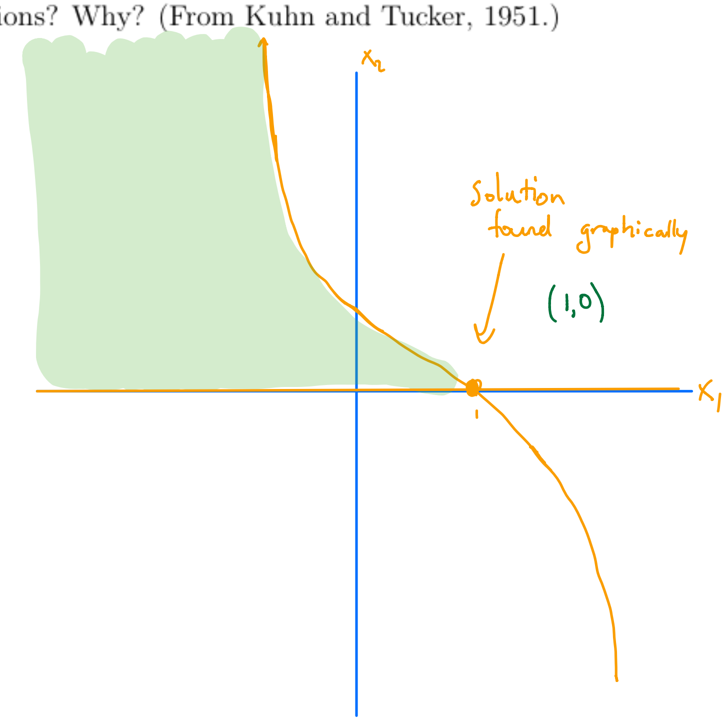
$$\frac{\partial f}{\partial x^2} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \Rightarrow$$

convex therefore our soln found in case 2 is the global solution.

2. (10 points) Graph the problem

$$\min f = -x_1, \text{subject to}$$
$$g_1 = x_2 - (1-x_1)^3 \le 0 \quad \text{and} \quad x_2 \ge 0.$$

Find the solution graphically. Then apply the optimality conditions. Can you find a solution based on the optimality conditions? Why? (From Kuhn and Tucker, 1951.)



$$\min_{\vec{x}} \quad -x_1$$

$$\text{s.t.} \quad g_1 = x_2 - (1-x_1)^3 \le 0$$
$$g_2 = -x_2 \le 0$$

$$\frac{\partial g_1}{\partial \vec{x}} = \begin{bmatrix} 3(1-x_1)^2 \\ 1 \end{bmatrix}$$

$$\frac{\partial g_2}{\partial x} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$\left.\frac{\partial g_1}{\partial x}\right|_{(1,0)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \left.\frac{\partial g_2}{\partial x}\right|_{(1,0)} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$\Rightarrow$ gradients of constraint are linearly dependent so the solution (1,0) is irregular. Therefore, KKT optimality conditions would not find this solution.

3. (30 points) Find a local solution to the problem

$$\max f = x_1x_2 + x_2x_3 + x_1x_3$$
$$\text{subject to } h = x_1 + x_2 + x_3 - 3 = 0.$$

Use three methods: direct elimination, reduced gradient, and Lagrange multipliers. Compare. Is the solution global?

Direct Elimination

$$x_1 = -x_2 - x_3 + 3$$

$$\min \quad (-x_2 - x_3 + 3)x_2 + x_2x_3 + (-x_2 - x_3 + 3)x_3 = f$$

$$= -x_2^2 - \cancel{x_2x_3} + 3x_2 + \cancel{x_2x_3} - x_2x_3 - x_3^2 + 3x_3 = f$$

$$= -x_2^2 + 3x_2 - x_2x_3 - x_3^2 + 3x_3 = f$$

$$\frac{\partial f}{\partial x_2} = -2x_2 + 3 - x_3 = 0$$

$$\frac{\partial f}{\partial x_3} = -x_2 - 2x_3 + 3 = 0$$

$$\begin{bmatrix} -2 & -1 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -3 \\ -3 \end{bmatrix}$$

$$x_2 = 1$$
$$x_3 = 1$$
$$x_1 + 1 + 1 - 3 = 0$$
$$x_1 = 1$$

$$\boxed{\vec{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}$$

Check Hessian

$$H = \begin{bmatrix} -2 & -1 \\ -1 & -2 \end{bmatrix}$$

$$\det(\lambda I - H) = 0$$

$$\begin{vmatrix} \lambda+2 & 1 \\ 1 & \lambda+2 \end{vmatrix} = 0$$

$$(\lambda+2)^2 - 1^2 = 0$$

$$\lambda^2 + 4\lambda + 4 - 1 = 0$$

$$\lambda^2 + 4\lambda + 3 = 0$$

$$\boxed{\begin{array}{l} \lambda_1 = -3 \\ \lambda_2 = -1 \quad \rightarrow \text{this is a global maximum} \end{array}}$$

Reduced Gradient

3 variable $n=3$

1 constraint $m=1$

max $f = x_1x_2 + x_2x_3 + x_1x_3$
subject to $h = x_1 + x_2 + x_3 - 3 = 0.$

$$X = \begin{bmatrix} d \\ s \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$d = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$s = x_3$$

$$\frac{\partial z}{\partial d} = \frac{\partial f}{\partial d} - \frac{\partial f}{\partial s}\left(\frac{\partial h}{\partial s}\right)^{-1}\frac{\partial h}{\partial d} = 0$$

$$h = 0$$

$$\frac{\partial f}{\partial d} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} x_2 + x_3 \\ x_1 + x_3 \end{bmatrix}$$

$$\frac{\partial f}{\partial s} = x_1 + x_2$$

$$\frac{\partial h}{\partial s} = 1 \Rightarrow \left(\frac{\partial h}{\partial s}\right)^{-1} = 1$$

$$\frac{\partial h}{\partial d} = \begin{bmatrix} \frac{\partial h}{\partial x_1} \\ \frac{\partial h}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\frac{\partial z}{\partial d} = \frac{\partial f}{\partial d} - \frac{\partial f}{\partial s}\left(\frac{\partial h}{\partial s}\right)^{-1}\frac{\partial h}{\partial d} = 0$$

$$= \begin{bmatrix} x_2 + x_3 \\ x_1 + x_3 \end{bmatrix} - [x_1 + x_2][1]\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x_2 + x_3 - x_1 - x_2 \\ x_1 + x_3 - x_1 - x_2 \end{bmatrix}$$

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

$x_1 + x_1 + x_3 - 3 = 0$

3 eqs, 3 unknowns

$\Rightarrow$ same answer as above

$$\boxed{\vec{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}$$

Lagrange Multiplier

$$f(x_1, x_2, x_3, \lambda) = x_1 x_2 + x_2 x_3 + x_1 x_3 + \lambda(x_1 + x_2 + x_3 - 3)$$

$$\frac{\partial f}{\partial x_1} = x_2 + x_3 + \lambda = 0$$

$$\frac{\partial f}{\partial x_2} = x_1 + x_3 + \lambda = 0$$

$$\frac{\partial f}{\partial x_3} = x_1 + x_2 + \lambda = 0$$

$$\frac{\partial f}{\partial \lambda} = x_1 + x_2 + x_3 - 3 = 0$$

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 3 \end{bmatrix}$$

$$\boxed{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \lambda \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -2 \end{bmatrix}}$$

4. (20 points) Use reduced gradient to find the value(s) of the parameter b for which the point $x_1 = 1$, $x_2 = 2$ is the solution to the problem

$$\begin{aligned} &\max\ f = 2x_1 + bx_2 \\ &\text{subject to } g_1 = x_1^2 + x_2^2 - 5 \le 0 \\ &\text{and } g_2 = x_1 - x_2 - 2 \le 0. \end{aligned}$$

First, check which constraints are active at the solution

$$g_1 = 1^2 + 2^2 - 5 \le 0$$

$$g_1 = 0 \qquad \text{active}$$

$$g_2 = x_1 - x_2 - 2 \le 0$$

$$g_2 = 1 - 2 - 2 \le 0$$

$$g_2 = -3 \qquad \text{not-active}$$

$$\min_{x_1, x_2} \quad 2x_1 + bx_2$$

$$x_1^2 + x_2^2 - 5 = 0$$

$n = 2 \qquad d = x_1$

$m = 1 \qquad s = x_2$

$$\frac{\partial f}{\partial d} = 2 \qquad \frac{\partial h}{\partial d} = 2x_1$$

$$\frac{\partial f}{\partial s} = b \qquad \frac{\partial h}{\partial s} = 2x_2$$

$$\frac{\partial z}{\partial d} = \frac{\partial f}{\partial d} - \frac{\partial f}{\partial s}\left(\frac{\partial h}{\partial s}\right)^{-1}\frac{\partial h}{\partial d}$$

$$= 2 - b\,\frac{1}{2x_2}(2x_1) = 0$$

$$2 - b\,\frac{x_1}{x_2} = 0$$

at solution

$x_1 = 1 \qquad x_2 = 2$

$$2 - b\frac{1}{2} = 0$$

$$-b\frac{1}{2} = -2$$

$$b\frac{1}{2} = 2$$

$$\boxed{b = 4}$$

5. (30 points, MAE 598) Find the solution for

$$\min\ f = x_1^2 + x_2^2 + x_3^2$$
$$\text{subject to } h_1 = x_1^2/4 + x_2^2/5 + x_3^2/25 - 1 = 0$$
$$\text{and } h_2 = x_1 + x_2 - x_3 = 0,$$

by implementing the generalized reduced gradient method (e.g., using MATLAB). See template code here.

$z = x_1^2 + x_2^2 + x_3^2$ $\quad s = \begin{bmatrix} x_2 \\ x_3 \end{bmatrix}$ $\quad d = [x_1]$

$$h = \begin{bmatrix} \frac{x_1^2}{4} + \frac{x_2^2}{5} + \frac{x_3^2}{25} - 1 \\ x_1 + x_2 - x_3 \end{bmatrix}$$

We are solving:

$$d_{k+1} = d_k - \alpha \left(\frac{\partial z}{\partial d}\right)_k^T$$

$$s'_{k+1} = s_{k+1} + \alpha_k \left(\frac{\partial h}{\partial s}\right)_k^{-1} \left(\frac{\partial h}{\partial d}\right)_k \left(\frac{\partial z}{\partial d}\right)_k^T$$

$$[s_{k+1}]_{j+1} = \left[s_{k+1} - \left(\frac{\partial h}{\partial s}\right)_{k+1}^{-1} h(d_{k+1}, s_{k+1})\right]_j$$

$$\frac{\partial z}{\partial d} = \frac{\partial f}{\partial d} - \frac{\partial f}{\partial s}\left(\frac{\partial h}{\partial s}\right)^{-1}\frac{\partial h}{\partial s}$$

We need functions for:

$\frac{\partial f}{\partial d}$, $\frac{\partial f}{\partial s}$, $\frac{\partial h}{\partial d}$, $\frac{\partial h}{\partial s}$, h (constraints)

These will be inputs for:

$\frac{\partial z}{\partial d}$, d_{k+1}, s'_{k+1}, s_{k+1}^{j+1}

Evaluate $\frac{\partial f}{\partial d}, \frac{\partial f}{\partial d}, \frac{\partial h}{\partial d}, \frac{\partial h}{\partial s}$;

$$\frac{\partial z}{\partial d} = \frac{\partial f}{\partial d} + \frac{\partial f}{\partial s}\left(\frac{\partial h}{\partial s}\right)^{-1}\frac{\partial h}{\partial d}$$

$$\frac{\partial f}{\partial d} = \frac{\partial f}{\partial x_1} = 2x_1$$

$$\frac{\partial f}{\partial s} = \begin{bmatrix} \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 2x_2 & 2x_3 \end{bmatrix}$$

$$\frac{\partial h}{\partial s} = \begin{bmatrix} \frac{\partial h_1}{\partial x_2} & \frac{\partial h_1}{\partial x_3} \\ \frac{\partial h_2}{\partial x_2} & \frac{\partial h_2}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \frac{2x_2}{5} & \frac{2x_3}{25} \\ 1 & -1 \end{bmatrix}$$

$$\frac{\partial h}{\partial d} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} \\ \frac{\partial h_2}{\partial x_1} \end{bmatrix} = \begin{bmatrix} \frac{x_1}{2} \\ 1 \end{bmatrix}$$

$$\frac{\partial z}{\partial d} = \begin{bmatrix} 2x_1 \end{bmatrix} + \begin{bmatrix} 2x_2 & 2x_3 \end{bmatrix} \begin{bmatrix} 2x_2 & 2x_3 \\ 1 & -1 \end{bmatrix}^{-1} \begin{bmatrix} \frac{x_1}{2} \\ 1 \end{bmatrix}$$

GeneralizedReducedGradient.py is the function file which contains all the GRG algorithms and hw4_desoptṗy contains the functions specific to this problem. To run it, put both python files into the same folder and run hw4_desopt.py.

The code in GeneralizedReducedGradient.py is:

```
import numpy as np

def dzdd(sk, dk, dfdd, dfds, dhds, dhdd):
    # This function calculates and returns the partial derivates of the reduced
    # gradient fucntion w.r.t. the decision variables at the current iteration

    ########## INPUTS ###############
    # sk - state variables at iteration k
    # dk - decision variables at iteration k
    # dfdd - function to calculate dfdd (with dk as input)
    # dfds - funciton to calculate dfds (with sk as input)
    # dhds - function to calculate dhds (with sk as input)
    # dhdd - function to calculate dhdd (with dk as input)
    dfdd_k = dfdd(dk)
    dfds_k = dfds(sk)
    dhds_k = dhds(sk)
    dhdd_k = dhdd(dk)
    return dfdd_k + dfds_k.dot(np.linalg.inv(dhds_k)).dot(dhdd_k)


def dk1(sk, dk, alphak, dfdd, dfds, dhds, dhdd):
    # This function iterates the decision variables and returns d_k+1

    ########## INPUTS ###############
    # sk - state variables at iteration k
    # dk - decision variables at iteration k
    # alphak - step size coefficient
    # dfdd - function to calculate dfdd (with dk as input)
    # dfds - funciton to calculate dfds (with sk as input)
    # dhds - function to calculate dhds (with sk as input)
    # dhdd - function to calculate dhdd (with dk as input)
    global dzdd
    dzdd_k = dzdd(sk, dk, dfdd, dfds, dhds, dhdd)
    return dk - alphak*dzdd_k.transpose()

def skp1(sk, dk, alphak, dfdd, dfds, dhds, dhdd):
    # This function calculates s_prime_k+1

    ########## INPUTS ###############
    # sk - state variables at iteration k
    # dk - decision variables at iteration k
    # alphak - step size coefficient
    # dfdd - function to calculate dfdd (with dk as input)
    # dfds - funciton to calculate dfds (with sk as input)
    # dhds - function to calculate dhds (with sk as input)
    # dhdd - function to calculate dhdd (with dk as input)
    global dzdd
```

```
    dhds_k = dhds(sk)
    dhdd_k = dhdd(dk)
    dzdd_k = dzdd(sk, dk, dfdd, dfds, dhds, dhdd)
    return sk + alphak*np.linalg.inv(dhds_k).dot(dhdd_k).dot(dzdd_k.transpose())

def iteratesk(sk, dk, dhds, h):
    # This function iterates sk during the convergence step of sk+1, after dk+1 has been ca

    ########## INPUTS ###############
    # sk - state variables at iteration k
    # dk - decision variables at iteration k
    # dhds - function to calculate dhds (with sk as input)
    # h - function to calculate the constraint values at the current iteration
    dhds_k = dhds(sk)
    hsd = h(dk, sk)
    return sk - np.linalg.inv(dhds_k).dot(hsd)

def sk1(sk, dk, alphak, dfdd, dfds, dhds, dhdd, h, epsilon):
    # This function iterates the state variables and returns s_k+1

    ########## INPUTS ###############
    # sk - state variables at iteration k
    # dk - decision variables at iteration k
    # alphak - step size coefficient
    # dfdd - function to calculate dfdd (with dk as input)
    # dfds - funciton to calculate dfds (with sk as input)
    # dhds - function to calculate dhds (with sk as input)
    # dhdd - function to calculate dhdd (with dk as input)
    # h - function to calculate the constraint values at the current iteration
    # epsilon - limit of convergence criteria
    global dzdd
    sk = skp1(sk, dk, alphak, dfdd, dfds, dhds, dhdd)
    h_k = h(dk, sk)
    while (np.linalg.norm(h_k) > epsilon):
        sk = iteratesk(sk, dk, dhds, h)
        h_k = h(dk, sk)
    return sk


def GRG(d0, s0, epsilon, alphak, dfdd, dfds, dhds, dhdd, h):
    # This function calculates the generalized reduced gradient

    ########## INPUTS ###############
    # s0 - initial state variables
    # d0 - initial decision variables
    # epsilon - limit of convergence criteria
    # alphak - step size coefficient
    # dfdd - function to calculate dfdd (with dk as input)
    # dfds - funciton to calculate dfds (with sk as input)
    # dhds - function to calculate dhds (with sk as input)
    # dhdd - function to calculate dhdd (with dk as input)
    # h - function to calculate the constraint values at the current iteration
    global dzdd
    dk = d0
```

```
    sk = s0
    dzdd_k = dzdd(sk, dk, dfdd, dfds, dhds, dhdd)
    cnt = 0
    itermax = 10000
    while (np.linalg.norm(dzdd_k) > epsilon or cnt == itermax):
        dk = dk1(sk, dk, alphak, dfdd, dfds, dhds, dhdd)
        sk = sk1(sk, dk, alphak, dfdd, dfds, dhds, dhdd, h, epsilon)
        dzdd_k = dzdd(sk, dk, dfdd, dfds, dhds, dhdd)
        cnt += 1
    if (cnt == itermax):
        print("A solution wasn't reached in {} iterations".format(cnt))
    else:
        print("A solution was reached in {} iterations".format(cnt))
        print("d: {}".format(dk))
        print("s: {}".format(sk))
```

The code in hw4_desopt.py is:

```
#!/usr/bin/python3
import numpy as np
from GeneralizedReducedGradient import GRG

#  THESE WOULD BE CHANGED FOR A DIFFERENT PROBLEM

def dfdd(d):
    # Function to calculate the partial derivative of the objective function w.r.t.
    # the decision variables, d
    # d - np array with shape (n-m, 1)
    return 2*d

def dfds(s):
    # Function to calculate the partial derivates of the objective function w.r.t.
    # the state variables, s
    # s - np array with shape (m, 1)
    return np.array([[2*s[0, 0], 2*s[1, 0]]])

def dhds(s):
    # Function to calculate the partial derivates of the constraints w.r.t.
    # the state variables, s
    # s - np array with shape (m, 1)
    return np.array([[2/5*s[0, 0], 2/25*s[1, 0]], [1, -1]])

def dhdd(d):
```

```
    # Function to calculate the partial derivates of the constraints w.r.t.
    # the decision variables, d
    # d - np array with shape (n-m, 1)
    return np.array([[d[0, 0]/2],[1]])

def constraints(d, s):
    # Function to calculate the value of the constraints at each iteration k
    # d - np array with shape (n-m, 1)
    return np.array([[1/4*d[0, 0]**2 + 1/5*s[0, 0]**2 + 1/25*s[1, 0]**2 - 1],
                     [d[0, 0] + s[0, 0] - s[1, 0]]])


def main():
    d0 = np.array([[1]])
    s0 = np.array([[1], [1]])
    epsilon = 1e-6 # this is used in the convergence critera
    alphak = 1e-2  # coefficient of step size
    GRG(d0, s0, epsilon, alphak, dfdd, dfds, dhds, dhdd, constraints)

main()
```

Solution

$[0.4667, 1.899, 2.366]^T$