Contents

1 Interior Point Methods

1.1 Linear Programming with Inequality Constraints

Solve the following problem:
min $-10x_1 - 9x_2$ subject to:

$$\begin{aligned} 7x_1 + 10x_2 &\leq 6300 \\ 3x_1 + 5x_2 &\leq 3600 \\ 3x_1 + 2x_2 &\leq 2124 \\ 2x_1 + 5x_2 &\leq 2700 \end{aligned}$$

To solve this with our Interior Point Method, we need to convert the inequalities into equalities using surplus variables. We create surplus variables $s_1, \ldots, s_4$ all non-negative, such that:

$$\begin{aligned} -7x_1 - 10x_2 + 6300 - s_1 &= 0 \\ -3x_1 - 5x_2 + 3600 - s_2 &= 0 \\ -3x_1 - 2x_2 + 2124 - s_3 &= 0 \\ -2x_1 - 5x_2 + 2700 - s_4 &= 0 \end{aligned}$$

This means we can create a new vector

$$y = \begin{bmatrix} x_1 & x_2 & s_1 & s_2 & s_3 & s_4 \end{bmatrix}^T$$

and then we have a linear system of equations in 5 variables and we can apply the interior point method

```
import numpy as np
import InteriorPoint as IP

A = np.array([[-7, -10, -1, 0, 0, 0],
              [-3, -5, 0, -1, 0, 0],
              [-3, -2, 0, 0, -1, 0],
              [-2, -5, 0, 0, 0, -1]])
b = np.array([[-6300, -3600, -2124, -2700]]).transpose()
c = np.array([[-10, -9, 0, 0, 0, 0]]).transpose()
Q = np.zeros((6,6))
tol = 1e-5
kmax = 10000
rho = .9
mu0 = 1e4
mumin = 1e-8

x,k = IP.InteriorPointBarrier_EqualityOnly(Q, c, A, b,
                                           tol, kmax, rho,
                                           mu0, mumin)
np.set_printoptions(suppress=True,precision=1)
print("Found optimal :\n" + str(x[0:2]))
print("Num Iterations: " + str(k))
```

1.2 Transportation Problem

We first create a function to handle a general transportation problem and then run it for a particular problem.

Wheat is harvested in the Midwest and stored in grain elevators in three different cities: Kansas City, Omaha, and Des Moines. These grain elevators spply three flour mills, located in Chicago, St. Louis, and Cincinnati. Grain is shipped to the mills in railroad cars, each car capable of holding one ton of wheat. Each grain elevator is able to supply the number of tons of wheat in the following table to the mills on a monthly basis. Each mill demands the number of tons of wheat per month as detailed in the following table.

Grain Elevator	Supply
Kansas City	150
Omaha	175
Des Moines	275
Total	600

Mill	Demand
Chicago	200
St. Louis	100
Cincinnati	300
Total	600

The cost of transporting one ton of wheat from each grain elevator (source) to each mill (destination) differs according to the distance and rail system. These costs are shown in the following table. For example, the cost of shipping one ton of wheat from the grain elevator at Omaha to the mill at Chicago is \$7.

	Chicago	St. Louis	Cincinnati
Kansas City	6	8	10
Omaha	7	11	11
Des Moines	4	5	12

How many tons of wheat should be transported from each grain elevator to each mill on a monthly basis in order to minimize the total cost of transportation?

```
import numpy as np
import InteriorPoint as IP
def TransportationProblem(supplyVec, demandVec, costMatrix):
    m = supplyVec.shape[0]
    n = demandVec.shape[0]

    A = np.zeros((m+n,m*n))
    onevec = np.ones((1,n))
    for i in range(m):
        A[i,i*m:(i+1)*m] = onevec
        A[m:,i*m:(i+1)*m] = np.eye(n)


    c = costMatrix.reshape((-1,1))

    b = np.vstack((supplyVec,demandVec))

    Q = np.zeros((m*n,m*n))

    tol = 1e-5
    kmax = 10000
    rho = .9
    mu0 = 1e4
    mumin = 1e-8

    x,k = IP.InteriorPointBarrier_EqualityOnly(Q, c, A, b,
                                              tol, kmax, rho,
                                              mu0, mumin)
    return x.reshape(costMatrix.shape),k
supplyVec = np.array([[150, 175, 275]]).transpose()
demandVec = np.array([[200, 100, 300]]).transpose()
costMatrix = np.array([[6, 8, 10],[7, 11, 11],[4, 5, 12]])

np.set_printoptions(suppress=True,precision=1)
x, k = TransportationProblem(supplyVec, demandVec, costMatrix)
print(x)
```

1.3 Problem Involving Piecewise Objective Function

Problem: Minimize

$$2.5\,[3x_1 + 2x_2 - 2600]_- + 0.3\,[x_1 + x_2 - 1150]_- + 0.2|x_1 - 400|$$

Subject to:

$$\begin{aligned} 2x_1 + x_2 &\le 1500 \\ x_1 + x_2 &\le 1200 \\ x_1 &\le 500 \end{aligned}$$

Solution:

To solve this, first we need to define $[x]_- = -\min\{x, 0\}$ and $[x]_+ = \max\{x, 0\}$. Then, in this case $u = u_+ - u_-$ and $|u| = u_+ + u_-$. Now, define the following:

$$\begin{aligned} u &= 3x_1 + 2x_2 - 2600 = u_+ - u_- \\ v &= x_1 + x_2 - 1150 = v_+ - v_- \\ w &= x_1 - 400 = w_+ - w_- \end{aligned}$$

Using this notation, we can rewrite the optimization problem as:
Minimize $2.5u_- + 0.3v_- + 0.2(w_+ + w_-)$ subject to:

$$\begin{aligned} 3x_1 + 2x_2 - u_+ + u_- &= 2600 \\ x_1 + x_2 - v_+ + v_- &= 1150 \\ x_1 - w_+ + w_- &= 400 \\ 2x_1 + x_2 &\leq 1500 \\ x_1 + x_2 &\leq 1200 \\ x_1 &\leq 500 \end{aligned}$$

Thus, we can solve this problem using the Interior Point Functions

```
import numpy as np
import InteriorPoint as IP

A = np.zeros((3,8),dtype="float")
A[0,0] = -2.0
A[0,1] = A[1,0] = A[1,1] = A[2,0] = -1.
b = np.array([[-1500, -1200, -500]]).T

C = np.array([[3, 2, -1, 1,  0, 0, 0,  0],
              [1, 1,  0, 0, -1, 1, 0,  0],
              [1, 0,  0, 0,  0, 0, -1, 1]])
d = np.array([[2600, 1150, 400]]).T

c = np.array([[0, 0, 0, 2.5, 0, 0.3, 0.2, 0.2]]).T
Q = np.zeros((8,8))

tol = 1e-8
kmax = 10000
rho = .9
mu0 = 100
mumin = 1e-12

x,k = IP.InteriorPointBarrier_EqualityInequality(Q,c,A,b,
                                                  C,d,tol,
                                                  kmax,rho,
                                                  mu0,mumin)
np.set_printoptions(suppress=True,precision=1)
print("Found optimal :\n" + str(x[0:2]))
print("Num Iterations: " + str(k))
```