ISYE 6669 – HOMEWORK 5

ANSWER 1 –

The Jacobian –

[

$2(200x^3 - 200xy + x - 1)$
$200(y - x^2)$

]

The Hessian –

$F_{xx} = -400(y - x^2) + 800x^2 + 2$
$F_{xy} = -400x$
$F_{yx} = -400x$
$F_{yy} = 200$

ANSWER 2 –

Here is the code, and the convergence train printed –

```
X, Y:  1.19591836734694 1.43020408163266
Alpha:  1
0.159856101972653
Matrix([[0.399806203197272, -0.00333194502269407]]) -0.0765131314744851


X, Y:  1.17639159275166 1.38350089081809
Alpha:  0.1
0.297082546833743
Matrix([[0.539259453472255, -0.0792577357394748]]) -0.0576895437208930


X, Y:  1.16004780508070 1.34508713086629
Alpha:  0.1
0.387104360072430
Matrix([[0.609541089792856, -0.124755841252295]]) -0.0456260163777355


X, Y:  1.14581824527664 1.31213556955098
Alpha:  0.1
0.435176362892300
Matrix([[0.641744306873193, -0.152776331570522]]) -0.0370067051536298


X, Y:  1.13316893584139 1.28322433863465
```

```
Alpha:  0.1
0.451856293421360
Matrix([[0.650481470666250, -0.169499704251166]]) -0.0304710897530397


X, Y:  1.01930064211089 1.02600781069057
Alpha:  1
35.0813361333537
Matrix([[5.32509737110894, -2.59319766341912]]) -0.0338307150518342


X, Y:  1.01392920309783 1.02802357653754
Alpha:  1
0.00159830026067453
Matrix([[0.0395601051607741, -0.00577047141428011]]) -0.000385985531839
435


X, Y:  1.00007991691008 0.999968037478965
Alpha:  1
0.00738315088333411
Matrix([[0.0768870562967550, -0.0383605455820657]]) -7.36995881832722e-
6
```

X, Y: 1.00000295240061 1.00000589888641 → **Solution**
Alpha: 1
6.98662883061155e-11 → **Objective value**
Done!

The code for it is as follows –

```
def converge(s1,s2):

    f = 100*(y-(x**2))**2 + (1-x)**2
    x0 = s1
    y0 = s2

    j = d1(f)
    h = d2(f)
    j1 = j[0].subs([(x, x0), (y, y0)])
    j2 = j[1].subs([(x, x0), (y, y0)])
    res1 = Matrix([j1,j2])
    h1 = h[0][0].subs([(x, x0), (y, y0)])
    h2 = h[0][1].subs([(x, x0), (y, y0)])
    h3 = h[1][0].subs([(x, x0), (y, y0)])
    h4 = h[1][1].subs([(x, x0), (y, y0)])
    res2 = Matrix([[h1,h2],[h3,h4]])
    res2neg = Matrix([[-h1,-h2],[-h3,-h4]])
    res2neg = res2neg.inv()

    d0 = res2neg.dot(res1)
    k = 0
    e = 10**(-4)

    while (res1.T).dot(res1) > e:

        a = 1
        c = 0.2
```

```
    p = 0.1

    d00 = d0[0]*a
    d01 = d0[1]*a
    m2 = Matrix([d00,d01])
    inner = Matrix([x0,y0])+m2
    func1 = f.subs([(x, inner[0]), (y, inner[1])])
    func2 = f.subs([(x, x0), (y, y0)])
    j1 = j[0].subs([(x, x0), (y, y0)])
    j2 = j[1].subs([(x, x0), (y, y0)])
    newj = Matrix([j1,j2]).T
    final = newj.dot(d0)
    while func1 > (func2 + c*a*final):

        a = p*a

    d00 = d0[0]*a
    d01 = d0[1]*a
    m2 = Matrix([d00,d01])
    inner = Matrix([x0,y0])+m2
    x0 = inner[0]
    y0 = inner[1]
    print("X, Y: ",x0,y0)
    print("Alpha: ",a)

    j1 = j[0].subs([(x, x0), (y, y0)])
    j2 = j[1].subs([(x, x0), (y, y0)])
    res1 = Matrix([j1,j2])
    h1 = h[0][0].subs([(x, x0), (y, y0)])
    h2 = h[0][1].subs([(x, x0), (y, y0)])
    h3 = h[1][0].subs([(x, x0), (y, y0)])
    h4 = h[1][1].subs([(x, x0), (y, y0)])
    res2 = Matrix([[h1,h2],[h3,h4]])
    res2neg = Matrix([[-h1,-h2],[-h3,-h4]])
    res2neg = res2neg.inv()

    d0 = res2neg.dot(res1)

    k += 1
    print((res1.T).dot(res1))

print("Done!")
```

With the alternate starting point with an alpha value of 1.2 and p value of 0.2, it takes around 50 iterations to converge.

ANSWER 3 –

For better readability, consider XY as amount of water transferred from X to Y. Hence, formulated LP –

Minimize {

2(AD) +
3(BD) +
2(FD) +
2(BG) +
FG +
CG +
2(DF) +
4(HF) +
4(HF) +
5(CF) +
3(BF) +

4(AE) +
2(HE) +
2(CF) +
2(EH)
}

Subject to the following –

1) Supply Constraints

a) AD + AE ≤ 100 – for source A
b) BD + BF + BG ≤ 100 – for source B
c) CH + CF + CG ≤ 80 – for source C

2) Demand Constraints

a) AD + BD + FD - DF = 50 – for demand D
b) AE + HE - EH - EF = 60 – for demand E
c) DF + EF + HF + BF + CF = 40 – for demand F
d) BG + FG + CG = 30 – for demand G
e) EH + CH - HE - HF = 70 – for demand F

3) Non-Negative Constraint

XY >= 0, for all X,Y in {A, B, C, D, E , F, G, H}

ANSWER 4 –

(a)

Let the variables we solve for be put in the form of theta –

x1 = $\theta_1 - \theta_2$
x2 = $\theta_2 - \theta_3$
x3 = $\theta_6 - \theta_1$
x4 = $\theta_5 - \theta_6$
x5 = $\theta_3 - \theta_4$
x6 = $\theta_4 - \theta_5$

p1 = 11.6(x1) – 10.5(x3)
p2 = 13.7(x5) – 5.9(x2)
p3 = 5.6(x4) – 9.8(x6)

Objective function –

Minimize { 16(p1) + 20(p2) + 8(p3) }

Subject to the following constraints –

1) Supply constraints

- 20 <= p1 <= 200
- 20 <= p2 <= 150
- 10 <= p3 <= 150

2) Demand constraints

- 11.6(x1) – 5.9(x2) = 110
- 5.6(x4) – 10.5(x3) = 95
- 13.7(x5) – 9.8(x6) = 65

3) Flow capacity constraints

- 11.6|x1| <= 100
- 5.9|x2| <= 110
- 10.5|x3| <= 40
- 5.6|x4| <= 60
- 13.7|x5| <= 50
- 9.8|x6| <= 80

(b)

Here is the LP solved using cvxpy –

```
c = np.array([16,20,8])
bp_min = np.array([20,20,10])
bp_max = np.array([200,150,150])
x = cvxpy.Variable(6)
total_cost = c[0]*(11.6*x[0] - 10.6*x[2]) + c[1]*(-5.9*x[1] + 13.7*x[4]) + c[2]*(5.6*x[3] - 9.8*x[5])

objective = cvxpy.Minimize(total_cost)

constraints =  [11.6*x[0] - 10.6*x[2] >= bp_min[0],
                -5.9*x[1] + 13.7*x[4] >= bp_min[1],
                5.6*x[3] - 9.8*x[5] >= bp_min[2],
                11.6*x[0] - 10.6*x[2] <= bp_max[0],
                -5.9*x[1] + 13.7*x[4] <= bp_max[1],
                5.6*x[3] - 9.8*x[5] <= bp_max[2],
                11.6*x[0] - 5.9*x[1] == 110,
                -10.5*x[2] + 5.6*x[3] == 95,
                13.7*x[4] - 9.8*x[5] == 65,
                abs(x[0]*11.6) <= 100,
                abs(x[1]*5.9) <= 110,
                abs(x[2]*10.5) <= 40,
                abs(x[3]*5.6) <= 60,
                abs(x[4]*13.7) <= 50,
                abs(x[5]*9.8) <= 80,
            ]

model = cvxpy.Problem(objective, constraints)
model.solve()
print("\nThe optimal value is", round(model.value,2))
print("x values:",  x.value)
```

And the nodal potential objective values from $\theta_1 - \theta_6$ are as follows for –

```
x values: [ 0.8297948  -5.90468914 -0.50130165  0.36585063  8.21074719
-2.50353853]
rounded x values: [0.83, -5.9, -0.5, 0.37, 8.21, -2.5]
```

And the optimal cost is $3310.29.

```
The optimal value is 3310.29
```

(c)

Using dual optimality, we find that the prices for demands 2, 4 and 6 respectively are $14.4, $17.92, and $9.9.