ISyE6669 Homework 5 Solution

Fall 2021

1. Compute the gradient $\nabla f(x)$ and Hessian $\nabla^2 f(x)$ of the Rosenbrock function

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

Solution:

$$\begin{aligned} \nabla f(x) &= \begin{bmatrix} 200\,(x_2 - x_1^2)\,(-2x_1) + 2\,(1 - x_1)\,(-1) \\ 200(x_2 - x_1^2) \end{bmatrix} \\ &= \begin{bmatrix} 400\,(x_1^3 - x_1 x_2) + 2\,(x_1 - 1) \\ 200(x_2 - x_1^2) \end{bmatrix} \qquad (1) \\ \nabla^2 f(x) &= \begin{bmatrix} 1200x_1^2 - 400x_2 + 2, & -400\,x_1 \\ -400\,x_1, & 200 \end{bmatrix} \qquad (2) \end{aligned}$$

2. Implement the Newton's Method with line search given in Algorithm 1. Use the Newton's

Algorithm 1 Newton's Method with Line Search

Start with x^0. Set $k = 0$, $\epsilon = 10^{-4}$.
Set $d^0 \leftarrow -(\nabla^2 f(x^0))^{-1} \nabla f(x^0)$
while $\|\nabla f(x^k)\| > \epsilon$ **do**
  Choose $\bar{\alpha} > 0, \rho \in (0, 1), c \in (0, 1)$; Set $\alpha^k \leftarrow \bar{\alpha}$
  while $f(x^k + \alpha^k d^k) > f(x^k) + c\alpha^k \nabla f(x^k)^\top d^k$ **do**
    $\alpha^k \leftarrow \rho\alpha^k$
  end while
  $x^{k+1} \leftarrow x^k + \alpha^k d^k$
  $d^{k+1} \leftarrow -(\nabla^2 f(x^{k+1}))^{-1} \nabla f(x^{k+1})$
  $k \leftarrow k + 1$
end while

Method to minimize the Rosenbrock function in Problem 1. Set the initial stepsize $\bar{\alpha} = 1$. Select your own choice of $\rho \in (0, 1), c \in (0, 1)$. First run the algorithm from the initial point $x^0 = (1.2, 1.2)^\top$, and then try the more difficult starting point $x^0 = (-1.2, 1)^\top$. For each starting point, print out the step length α^k used by the algorithm as well as the point x^k for *every* step k. You should observe that Newton's Method converges very fast.

Note that answers may vary depending on what ρ and c values were used. We use $\rho = 0.9$ and $c = 10^{-5}$. We do not break the inner backtracking while loop in our code, but you can if α falls below 0.2 or 0.3. Such answers will be accepted. Using initial point $x^0 = (1.2, 1.2)^\top$, we get the following output from our code:

```
iter =

     1


alpha =

     1


x =

   1.195918367346939
   1.430204081632654


iter =

     2


alpha =

   0.656100000000000


x =

   1.067803199227371
   1.123784446698436


iter =

     3


alpha =
```

```
   0.900000000000000


x =

   1.053558314632527
   1.108140283036032


iter =

     4


alpha =

   0.900000000000000


x =

   1.018347492953793
   1.035607330512453


iter =

     5


alpha =

   0.900000000000000


x =

   1.005495680368007
   1.010713965562699


iter =

     6
```

```
alpha =

   0.900000000000000


x =

   1.000836216112118
   1.001620661106868


iter =

     7


alpha =

   0.900000000000000


x =

   1.000091437375132
   1.000177081378216


iter =

     8


alpha =

   0.900000000000000


x =

   1.000009239115984
   1.000017891387492


iter =
```

```
     9


alpha =

   0.900000000000000


x =

   1.000000924887601
   1.000001791013948
```

Note that Newton's method converges very fast. Our code converges in 9 iterations to optimal solution $x_* = (1,1)^\top$.

Using initial point $x^0 = (-1.2, 1)^\top$, we get the following output from our code:

```
iter =

     1


alpha =

     1


x =

  -1.175280898876405
   1.380674157303371


iter =

     2


alpha =

   0.166771816996666
```

```
x =

  -0.852011114246042
   0.620910454427278


iter =

     3


alpha =

   0.900000000000000


x =

  -0.776255621024348
   0.586332645981484


iter =

     4


alpha =

   0.729000000000000


x =

  -0.471434147807713
   0.124932946380622


iter =

     5


alpha =
```

```
   0.900000000000000


x =

  -0.406719192152260
   0.151500754845643


iter =

     6


alpha =

   0.729000000000000


x =

  -0.135706497703422
  -0.058803878313132


iter =

     7


alpha =

   0.900000000000000


x =

  -0.073548007735245
  -0.006176381616507


iter =

     8
```

```
alpha =

   0.810000000000000


x =

   0.188597803134905
  -0.035352576110539


iter =

     9


alpha =

   0.900000000000000


x =

   0.236690898089599
   0.046617464710355


iter =

    10


alpha =

   0.810000000000000


x =

   0.451295330945479
   0.155825440991456


iter =
```

```
    11


alpha =

   0.900000000000000


x =

   0.498022733151194
   0.241058989144317


iter =

    12


alpha =

   0.900000000000000


x =

   0.686772990971616
   0.435333715941717


iter =

    13


alpha =

   0.900000000000000


x =

   0.720882495066941
   0.514875770905670
```

```
iter =

    14


alpha =

   0.900000000000000


x =

   0.849103637916556
   0.704056746370642


iter =

    15


alpha =

   0.900000000000000


x =

   0.880081108112290
   0.771891129041179


iter =

    16


alpha =

   0.900000000000000


x =
```

```
   0.950606625890149
   0.898413945746921


iter =

    17


alpha =

   0.900000000000000


x =

   0.972314794184649
   0.944400913275701


iter =

    18


alpha =

   0.900000000000000


x =

   0.993095511134515
   0.985707341467110


iter =

    19


alpha =

   0.900000000000000
```

```
x =

   0.998712618051308
   0.997342206287941


iter =

    20


alpha =

   0.900000000000000


x =

   0.999851964205533
   0.999694183499308


iter =

    21


alpha =

   0.900000000000000


x =

   0.999984936676745
   0.999968879216080


iter =

    22
```

```
alpha =

   0.900000000000000


x =

   0.999998490972099
   0.999996882326324
```

Using this more difficult initial point, we converge in 22 iterations to optimal solution $x_* = (1,1)^\top$.

Here is the Matlab code (you can use Python alternatively) that generates the above output.

```
clc
clear all
close all
x0=[-1.2;1];
epsilon=10^-4;
alphabar=1;
c=1*10^-5;
rho=0.9;

f=@(x1,x2) 100*(x2-x1^2)^2+(1-x1)^2;
gradf=@(x1,x2) [400*(x1^3-x1*x2)+2*(x1-1);200*(x2-x1^2)];
hessf=@(x1,x2) [1200*x1^2-400*x2+2 -400*x1; -400*x1 200];

d=-inv(hessf(x0(1),x0(2)))*gradf(x0(1),x0(2));

x=x0;
z=x+alphabar*d;
iter=0;
while norm (gradf(x(1),x(2)))>epsilon
    alpha=alphabar;
    while f(z(1),z(2))>f(x(1),x(2))+(c*alpha)*transpose(gradf(x(1),x(2)))*d
        alpha=rho*alpha;
        z=x+alpha*d;
    end
    iter=iter+1
    alpha
    x=x+alpha*d


    d=-inv(hessf(x(1),x(2)))*gradf(x(1),x(2));
```

end

Observe that Newton's method converges in less iterations for initial point $x^0 = (1.2, 1.2)^\top$ than initial point $x^0 = (-1.2, 1)^\top$ because $x^0 = (1.2, 1.2)^\top$ is closer in distance to the optimal solution $x_* = (1, 1)^\top$. Note we also take c to be very small (i.e. 10^{-5}) and ρ to be large (i.e. 0.9) so the condition in the backtracking line search is satisfied faster and so we do not reduce the stepsize as much in this loop. This way, since the stepsize is not too small, we make significant progress towards the optimal solution at each iteration and thus converge in a very small number of iterations.

3. Figure 1 below illustrates the water network of Newvillage. The lines are water piplelines numbered from 1 through 13. The arrows on the lines are possible direction(s) of flow of water in these pipelines. The circles are water sources numbered A, B, C. The rectangles are houses D, E, F, G, H. The maximum possible capacity of the water sources are (the sources can operate at less than the maximum capacity): A: 100 Units, B: 100 Units, C: 80 Units Demands of water in the houses are: D: 50 Units, E: 60 Units, F: 40 Units, G: 30 Units, H: 70 Units Since the houses are at different elevation and the pipes are of different diameter, the cost of transporting water is different in the different pipes. These costs per unit of water are: Pipe 1: \$2, Pipe 2: \$3, Pipe 3: \$4, Pipe 4: \$2, Pipe 5: \$3, Pipe 6: \$2, Pipe 7: \$4, Pipe 8: \$1, Pipe 9: \$2, Pipe 10: \$4, Pipe 11: \$5, Pipe 12: \$1, Pipe 13: \$2. Formulate an LP to minimize the total cost of transporting water so as to meet the water demands of each house.

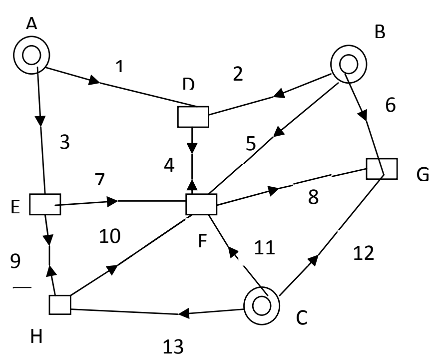

Figure 1: Water Network for Question 2

Solution:
We introduce the following decision variables. x_A, x_B, x_C, respectively, is the amount of water pumped through water source A, B, C, and f_{ij} is the flow of water from location i to location j, for arcs (i, j) in the set of arc $\mathcal{A} := \{(A, D), (A, E), (B, D), (B, F), (B, G), (C, F), (C, G), (C, H), (D, F), (E, F), (E, H), (F, D), (F, G), (H, E), (H, F)\}$.

$$
\begin{aligned}
\min \quad & 2f_{AD} + 4f_{AE} + 3f_{BD} + 3f_{BF} + 2f_{BG} + 5f_{CF} + f_{CG} + 2f_{CH} + 2f_{DF} + 4f_{EF} + 2f_{EH} + 2f_{FD} \\
& + f_{FG} + 2f_{HE} + 4f_{HF} && (3) \\
\text{s.t.} \quad & x_A - f_{AD} - f_{AE} = 0 && (4) \\
& x_B - f_{BD} - f_{BF} - f_{BG} = 0 && (5) \\
& x_C - f_{CF} - f_{CG} - f_{CH} = 0 && (6) \\
& f_{AD} + f_{BD} - f_{DF} + f_{FD} = 50 && (7) \\
& f_{AE} + f_{HE} - f_{EH} - f_{EF} = 60 && (8) \\
& f_{BF} + f_{CF} + f_{DF} + f_{EF} + f_{HF} - f_{FD} - f_{FG} = 40 && (9) \\
& f_{BG} + f_{FG} + f_{CG} = 30 && (10) \\
& f_{CH} + f_{EH} - f_{HE} - f_{HF} = 70 && (11) \\
& x_A \le 100, && (12) \\
& x_B \le 100, && (13) \\
& x_C \le 80, && (14) \\
& x_i \ge 0, \quad \forall i \in \{A, B, C\}, && (15) \\
& f_{ij} \ge 0, \quad \forall (i,j) \in \mathcal{A}. && (16)
\end{aligned}
$$

Note: Edges DF and EH are bidirectional, meaning that there is flow in both directions. Thus we have two variables for each edge: f_{DF} and f_{FD} for DF, and f_{EH} and f_{EH} for HE. Some of the students assumed that arrows pointing to both directions, that may be flow in one direction or the other, but not both at the same time. We will accept that formulation too. Following is the solution in that case.

We introduce the following decision variables. x_A, x_B, x_C, respectively, is the amount of water pumped through water source A, B, C, and f_{ij} is the flow of water from location i to location j, for arcs (i,j) in the set of arc $\mathcal{A} := \{(A,D),(A,E),(B,D),(B,F),(B,G),(C,F),$ $(C,G),(C,H),(D,F),(E,F),(E,H),(F,G),(H,F)\}$. And let $\mathcal{D} = \{(D,F),(E,H)\} \subset \mathcal{A}$ denotes set of edges for which flow can go in either direction.

$$
\begin{aligned}
\min \quad & 2f_{AD} + 4f_{AE} + 3f_{BD} + 3f_{BF} + 2f_{BG} + 5f_{CF} + f_{CG} + 2f_{CH} + 2|f_{DF}| + 4f_{EF} + 2|f_{EH}| \\
& + f_{FG} + 4f_{HF} \\
\text{s.t.} \quad & x_A - f_{AD} - f_{AE} = 0 \\
& x_B - f_{BD} - f_{BF} - f_{BG} = 0 \\
& x_C - f_{CF} - f_{CG} - f_{CH} = 0 \\
& f_{AD} + f_{BD} - f_{DF} = 50 \\
& f_{AE} + f_{HE} - f_{EH} - f_{EF} = 60 \\
& f_{BF} + f_{CF} + f_{DF} + f_{EF} + f_{HF} - f_{FG} = 40 \\
& f_{BG} + f_{FG} + f_{CG} = 30 \\
& f_{CH} + f_{EH} - f_{HF} = 70 \\
& x_A \leq 100, \\
& x_B \leq 100, \\
& x_C \leq 80, \\
& x_i \geq 0, \quad \forall i \in \{A, B, C\}, \\
& f_{ij} \geq 0, \quad \forall (i, j) \in \mathcal{A} \setminus \mathcal{D}.
\end{aligned}
$$

This is not a linear problem due to two absolute values in the objective function. Hence it will need to be converted into:

$$
\begin{aligned}
\min \quad & 2f_{AD} + 4f_{AE} + 3f_{BD} + 3f_{BF} + 2f_{BG} + 5f_{CF} + f_{CG} + 2f_{CH} + 2z + 4f_{EF} + 2w \\
& + f_{FG} + 4f_{HF} \\
\text{s.t.} \quad & x_A - f_{AD} - f_{AE} = 0 \\
& x_B - f_{BD} - f_{BF} - f_{BG} = 0 \\
& x_C - f_{CF} - f_{CG} - f_{CH} = 0 \\
& f_{AD} + f_{BD} - f_{DF} = 50 \\
& f_{AE} + f_{HE} - f_{EH} - f_{EF} = 60 \\
& f_{BF} + f_{CF} + f_{DF} + f_{EF} + f_{HF} - f_{FG} = 40 \\
& f_{BG} + f_{FG} + f_{CG} = 30 \\
& f_{CH} + f_{EH} - f_{HF} = 70 \\
& x_A \leq 100, \\
& x_B \leq 100, \\
& x_C \leq 80, \\
& x_i \geq 0, \quad \forall i \in \{A, B, C\}, \\
& f_{ij} \geq 0, \quad \forall (i, j) \in \mathcal{A} \setminus \mathcal{D} \\
& -z \leq f_{DF} \leq z \\
& -w \leq f_{EH} \leq w.
\end{aligned}
$$

4. Consider the following electric power network shown in Figure 2. This network is taken from a real-world electric power system. Electricity generators are located at nodes 1, 3, and 5 and producing p_1, p_2, p_3 amounts of electricity, respectively. Electricity loads are located at nodes 2, 4, and 6 and are consuming d_1, d_2, d_3 amounts of electricity, respectively.

   The demand is fixed and given as $d_1 = 110,\ d_2 = 65,\ d_3 = 95$.

   Each generator i's production must be within an upper and a lower bound as $p_i^{min} \leq p_i \leq p_i^{max}$. The bounds are given as $p_1^{\min} = 20,\ p_1^{\max} = 200,\ p_2^{\min} = 20,\ p_2^{\max} = 150,\ p_3^{\min} = 10,\ p_3^{\max} = 150$.

   The flow limits over lines are given as $f_{12}^{\max} = 100, f_{23}^{\max} = 110, f_{34}^{\max} = 50, f_{45}^{\max} = 80, f_{56}^{\max} = 60, f_{61}^{\max} = 40$.

   The line parameters are given as $B_{12} = 11.6,\ B_{23} = 5.9,\ B_{34} = 13.7,\ B_{45} = 9.8,\ B_{56} = 5.6,\ B_{61} = 10.5$. The unit generation costs are given as $c_1 = 16, c_2 = 20, c_3 = 8$.

   (a) Formulate the power system scheduling problem using the model discussed in Lecture 2.

   (b) Implement and solve the model using CVXPY. Write down the optimal solution.

   (c) Find the electricity prices for demand at nodes 2, 4, 6. To do this, use the command `constraints[0].dual_value` to find the dual variable of `constraints[0]`. Hint: Recall the electricity price at node i is the dual variable for the flow conservation constraint at node i.

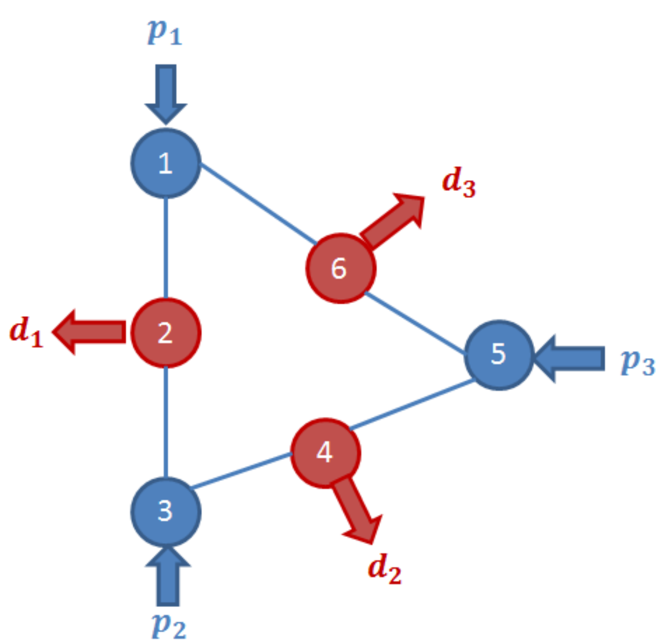

Figure 2: Electric network for Question 3.

(a) The LP can be formulated as follows:

$$
\begin{aligned}
\min \quad & \sum_{i=1..3} c_i p_i \\
\text{s.t.} \quad & \\
& \textit{Flow Conservation:} \\
& f_{12} - f_{61} = p_1, \\
& f_{34} - f_{23} = p_2, \\
& f_{56} - f_{45} = p_3, \\
& f_{23} - f_{12} = -d_1, \\
& f_{45} - f_{34} = -d_2, \\
& f_{61} - f_{56} = -d_3, \\
& \textit{Constraints linking branch flow and nodal potential:} \\
& f_{12} = B_{12}(\theta_1 - \theta_2), \\
& f_{23} = B_{23}(\theta_2 - \theta_3), \\
& f_{34} = B_{34}(\theta_3 - \theta_4), \\
& f_{45} = B_{45}(\theta_4 - \theta_5), \\
& f_{56} = B_{56}(\theta_5 - \theta_6), \\
& f_{61} = B_{61}(\theta_6 - \theta_1), \\
& \textit{Flow limit constraints:} \\
& - f_{12}^{\max} \leq f_{12} \leq f_{12}^{\max}, \\
& - f_{23}^{\max} \leq f_{23} \leq f_{23}^{\max}, \\
& - f_{34}^{\max} \leq f_{34} \leq f_{34}^{\max}, \\
& - f_{45}^{\max} \leq f_{45} \leq f_{45}^{\max}, \\
& - f_{56}^{\max} \leq f_{56} \leq f_{56}^{\max}, \\
& - f_{61}^{\max} \leq f_{61} \leq f_{61}^{\max}, \\
& \textit{Generation constraints:} \\
& p_1^{\min} \leq p_1 \leq p_1^{\max}, \\
& p_2^{\min} \leq p_2 \leq p_2^{\max}, \\
& p_3^{\min} \leq p_3 \leq p_3^{\max}.
\end{aligned}
$$

(b) The optimal objective value is . The quantum of electricity produced are $p_1 = 113.12, p_2 = 20.0, p_3 = 136.88$.
Following is the output from python model
-The optimal value is 3304.96
-The quantum of electricity produced at node 1: p_1= 113.12 , at node 3: p_2= 20.0 , at node 5: p_3= 136.88
-The flows in different lines are f_{12} =78.12 , f_{23}= -31.88 , f_{34}= -11.88, f_{45}= -76.88, f_{56}= 60.0 and f_{61}= -35.0

(c) The electricity price at node 2 is 14.4 , at node 4 is 9.9 and at node 6 is 17.77 .