Homework 3

Name: Naman Makkar

Student ID: nbm49

- Reasoning and work must be shown to gain partial/full credit
- Please include the cover-page on your homework PDF with your name and student ID. Failure of doing so is considered bad citizenship.

In this homework, you are going to learn about optimization problems applied to energy systems. This assignment leverages some of the work you did for Midterm 1. All the data you need to run these studies is provided to you in the datasets folder that contains the following

1. *branches_ieee118_subset.csv*. This file contains information about the edges of the IEEE-118 network. The file contains the following features.
    - **fbus**: The "from" bus ID
    - **tbus**" The "to" bus ID
    - **r**: The resistance in p.u.
    - **x**: The reactance in p.u.
    - **rateA**: The thermal limit (capacity) of the line in MW.
    - **ratio**: transformer off nominal turns ratio
2. *buses_ieee118_subset.csv*. This file contains information about the nodes of the IEEE-118 network. The file contains the following features.
    - **bus_i**: ID of the bus
    - **Pd**: Real power demand in MW
    - **baseKV**: base voltage in kV
3. *generators_ieee118_subset.csv*. This file contains information about the nodes of the IEEE-118 network. The file contains the following features.
    - **bus**: ID of the bus that the generator is connected to
    - **mBase**: Total MVA base of machine
    - **c2,c1,c0**: Coefficients of the generator cost function of the form $f(x) = c_2x^2 + c_1x + c_0$
    - **Pmax,Pmin**: Capacity limits of the generator in MW.

1. (6 points) **DC Optimal Power Flow**: In this problem, we are going to use the DC approximation to run an Optimal Power Flow (OPF) problem and calculate the generator setpoints. Power networks can be denoted as a graph $\mathcal{G} := \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each edge $e := \{i, j\} \in \mathcal{E}$ connects two buses $i, j \in \mathcal{V}$. Also, $N := |\mathcal{V}|$ denotes the number of buses and $E := |\mathcal{E}|$ is the number of deges (the operator $|.|$ denotes the cardinality of a set). The DC OPF is a problem of the form

$$\begin{array}{lll} \min_{\mathbf{p}} & \mathbf{p}^\intercal \mathbf{C}_2 \mathbf{p} + \mathbf{c}_1^\intercal \mathbf{p} + \mathbb{1}^\intercal \mathbf{c}_0 & \\ \text{s.t.} & \mathbf{p} - \mathbf{d} = \mathbf{B}\boldsymbol{\theta} & (\boldsymbol{\lambda}) \\ & \underline{\mathbf{p}} \leq \mathbf{p} \leq \overline{\mathbf{p}} & (\boldsymbol{\beta}) \\ & \underline{\mathbf{f}} \leq \text{diag}(\mathbf{b})\mathbf{M}^\intercal \boldsymbol{\theta} \leq \overline{\mathbf{f}} & (\boldsymbol{\mu}) \end{array} \tag{1}$$

where $\mathbf{p} \in \mathbb{R}^N$ is the vector of power injections, $\mathbf{d}$ is the vector of power demands. The generator cost constants are given by $\mathbf{C}_2 := \text{diag}(\mathbf{c}_2)$ where $\mathbf{c}_2$ as well as $\mathbf{c}_1$ and $\mathbf{c}_0$ are the vectors of constants. Also, $\mathbf{B} \in \mathbb{R}^{N \times N}$ is the susceptance matrix and $\boldsymbol{\theta} \in \mathbb{R}^N$ is the vector of bus angles. $\mathbf{M} \in \{-1, 0, 1\}^{N \times E}$ is the incidence of the graph, $\mathbf{b} := [b_1, \ldots, b_E]^\intercal \in \mathbb{R}^E$ is the vector of line susceptances where $b_e := 1/(\tau_e x_e), \forall e \in \mathcal{E}$ (See Homework 2). Also, $\overline{\mathbf{p}}$, $\underline{\mathbf{p}}$, $\overline{\mathbf{f}}$ and $\underline{\mathbf{f}}$ are the maximum and minimum generator capacity and line capacity limits. It should be noted that $\underline{\mathbf{f}} := -\overline{\mathbf{f}}$. The vectors $\boldsymbol{\lambda}$, $\boldsymbol{\beta}$ and $\boldsymbol{\mu}$ are the dual variables of their respective constraints, and $\mathbb{1}$ is the all-ones vector.

   (a) (**0.5 pts**) Using your solution from the midterm or the solution posted on canvas, rerun the DCOPF using the same data as last time, and provide a list of the transmission capacity constraints that are not binding.
   Solution:

In [1]:

```
!pip install cvxpy
import cvxpy as cp
import numpy as np
import pandas as pd
branches = pd.read_csv('/kaggle/input/energy-systems-hw2/datasets/branches_ieee118_subset.csv')
buses = pd.read_csv('/kaggle/input/energy-systems-hw2/datasets/buses_ieee118_subset.csv')
generators = pd.read_csv('/kaggle/input/energy-systems-hw2/datasets/generators_ieee118_subset.csv')

d = np.array(buses.Pd) / 100
d

c0 = np.array(generators.c0)
C2 = np.diag(generators.c2)
c1 = np.array(generators.c1)

p_max = np.array(generators.Pmax) / 100
p_min = np.array(generators.Pmin) / 100

f_max = np.array(branches.rateA)/100
f_min = -f_max

# Incidence Matrix Calculation
E = len(branches['fbus'])
N = max(branches['tbus'])
M = [[0 for _ in range(N)] for _ in range(E)]
fbus = list(branches['fbus'])
tbus = list(branches['tbus'])

for edge_num in range(E):
    node1 = fbus[edge_num]
    node2 = tbus[edge_num]
    M[edge_num][node1-1] = -1
    M[edge_num][node2-1] = 1
M = np.array(M)
M = M.T

# b_e calculation
transformer_turn_ratios = list(branches['ratio'])
reactance = list(branches['x'])
susceptances = [0 for _ in range(E)]

for edge_num in range(E):
    susceptances[edge_num] = 1/(transformer_turn_ratios[edge_num]*reactance[edge_num])
b = np.array(susceptances)

B = M @ np.diag(b) @ M.T
b_diag = np.diag(b)

Egen = np.zeros((118, 54))
for idx, bus in enumerate(generators.bus):
    Egen[bus-1, idx] = 1
```

In [2]:

```
n = 118
m = 210
p = cp.Variable(54)
theta = cp.Variable(n)

constraints = [
    Egen @ p - d == B @ theta,
    p <= p_max,
    p >= p_min,
    f_min <= b_diag @ M.T @ theta,
    f_max >= b_diag @ M.T @ theta,
]

objective = cp.Minimize(p.T @ C2 @ p + c1 @ p + np.ones((1, len
    (c0))) @ c0)

problem = cp.Problem(objective, constraints)
problem.solve()
```

Out[2]:

```
852.5137715127258
```

In [3]:

```
print(f'Optimal Value: {problem.value}')
print(f'Value of power Injection p is: {p.value}')
print(f'A dual solution corresponding to the 1st constraints is
     : {problem.constraints[0].dual_value}')
print(f'A dual solution corresponding to the 2nd constraints is
     : {problem.constraints[1].dual_value}')
print(f'A dual solution corresponding to the 3rd constraints is
     : {problem.constraints[2].dual_value}')
print(f'A dual solution corresponding to the 4th constraints is
     : {problem.constraints[3].dual_value}')
print(f'A dual solution corresponding to the 5th constraints is
     : {problem.constraints[4].dual_value}')
```

Out[3]:

```
Optimal Value: 852.5137715127258
Value of power Injection p is: [-8.91289139e-15 -8.72964628e-15
    -8.84932198e-15 -8.10221358e-15
  4.40192265e+00  8.31472614e-01 -6.89352239e-15 -6.81768128e-15
 -6.66502576e-15 -4.30869427e-15  2.15192656e+00  3.07144087e+00
 -6.98167114e-15  6.84715782e-02 -6.86237912e-15 -4.29380044e-15
 -4.35584899e-15 -3.00567438e-15 -1.57091787e-15  1.85864502e-01
  1.99564113e+00  4.69577703e-01  1.53339718e-15  1.51971578e-15
  1.51638710e+00  1.56532057e+00  1.53627398e-15  3.82534358e+00
  3.83497831e+00  5.05036317e+00  1.04106081e-15 -1.75771678e-15
  6.07642988e-16  1.99258313e-15  2.94129925e-15  3.77653465e-15
  4.67062640e+00  2.53656817e-15  3.72576680e-02  5.49138081e+00
  2.78366712e-15  5.26398907e-15  8.92495852e-15  6.65695003e-15
  2.49850685e+00  3.96588391e-01  8.42645838e-15  8.57068088e-15
  8.57073570e-15  8.93801481e-15  3.56929550e-01  8.99024666e-15
 -6.93381175e-15  2.16713882e-15]
A dual solution corresponding to the 1st constraints is :
    [-20.19564069 -20.19564064 -20.19564071 -20.1956408  -20.19564082
 -20.19564072 -20.19564068 -20.195641   -20.195641   -20.195641
 -20.19564067 -20.19564062 -20.19564058 -20.19564053 -20.19564028
 -20.19564024 -20.19563942 -20.19563962 -20.19563981 -20.19563517
 -20.1956318  -20.19562796 -20.19562165 -20.19561108 -20.19562969
 -20.19563318 -20.19563057 -20.19563158 -20.19563269 -20.19564136
 -20.19563308 -20.19563025 -20.19564385 -20.19564768 -20.19564779
 -20.19564776 -20.19564794 -20.19564967 -20.19564844 -20.19564872
 -20.19564895 -20.19564958 -20.19564796 -20.19564836 -20.19564851
 -20.19564684 -20.19564303 -20.1956502  -20.19565109 -20.19565269
 -20.19565461 -20.19565508 -20.1956564  -20.19565738 -20.19565783
 -20.19565758 -20.19565553 -20.19565587 -20.19566285 -20.19566481
 -20.19566507 -20.19566449 -20.19566514 -20.19566638 -20.19566975
 -20.19566216 -20.19566324 -20.19567831 -20.19559889 -20.1955669
 -20.19557071 -20.19559004 -20.19557071 -20.19553918 -20.19553067
 -20.195484   -20.19541662 -20.19546483 -20.19555969 -20.1958334
 -20.19573548 -20.19377197 -20.19150512 -20.18799357 -20.18628834
 -20.18628834 -20.18628834 -20.18313575 -20.18093512 -20.18575272
 -20.19193464 -20.20134063 -20.1995952  -20.19808853 -20.19717703
 -20.1960282  -20.19593342 -20.19675941 -20.19759784 -20.1982942
 -20.19960144 -20.20076159 -20.1982942  -20.1982942  -20.19829419
 -20.1982942  -20.19829419 -20.19829419 -20.19829419 -20.19829419
 -20.19829419 -20.19829419 -20.19563824 -20.19563038 -20.1956304
 -20.19567831 -20.19564062 -20.19550877]
A dual solution corresponding to the 2nd constraints is : [0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0.
 0. 0. 0. 0. 0. 0.]
```

Out[3]:

```
A dual solution corresponding to the 3rd constraints is : [19.80435931 19.8043592  19.80435928 19.804359    0.          0.
 19.80435972 19.80436038 19.80436019 19.80438892  0.          0.
 19.80436943  0.         19.80436975 19.80435232 19.80435224 19.80435128
 19.80435042  0.          0.          0.         19.80434217 19.80434242
  0.          0.         19.80433551  0.          0.          0.
 19.8044331  19.80440996 19.80442929 19.80446082 19.804516   19.80458338
  0.         19.81371166  0.          0.         19.81424728 19.80806536
 19.79865937 19.80240216  0.          0.         19.8017058  19.80170581
 19.80170581 19.80170581  0.         19.80170581 19.80436176 19.80432169]
 A dual solution corresponding to the 4th constraints is : [0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.03378118 0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.        ]
```

Out[3]:

```
A dual solution corresponding to the 5th constraints is : [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

In [4]:

```python

# All generators not binding for maximum power
# These are the generators that are not binding for the minimum power constraint
shadow_prices = problem.constraints[2].dual_value.flatten()
idx = np.where(shadow_prices == 0)[0]
idx
```

Out[4]:

```
array([ 4,  5, 10, 11, 13, 19, 20, 21, 24, 25, 27, 28, 29, 36, 38, 39, 44,
       45, 50])
```

Additionally, it can be observed that the following constraints are non-binding: $\overline{\mathbf{f}} \geq \mathrm{diag}(\mathbf{b})\mathbf{M}^{\intercal}\boldsymbol{\theta}$ and therefore all the transmission lines for this constraint are non-binding

In [5]:

```python

# All transmission lines not binding at maximum since the dual for the constraint is 0
# This gives the list of the indexes of the non binding transmission lines for the minimum flow constraint
shadow_prices_flow = problem.constraints[3].dual_value.flatten()
idx = np.where(shadow_prices_flow == 0)[0]
idx
```

Out[5]:

```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,
        12,
            13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,
        25,
            26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,
        38,
            39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,
        51,
            52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,
        64,
            65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,
        77,
            78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,
        90,
            91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102,
        103,
           104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,
        116,
           117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128,
        129,
           130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141,
        142,
           143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154,
        155,
           156, 157, 158, 160, 161, 162, 163, 164, 165, 166, 167, 168,
        169,
           170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
        182,
           183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
        195,
           196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
        208,
           209])
```

(b) (**1.5 pts**) Calculate the electricity prices of the network, and plot these prices on top of the network. You may use the code below to generate the figure that should look like Fig. 8. What do you observe? Is there a lot of variance between nodes? Why do you think this is the case? You answer to this part should be supported by the answer to part (a). (**Note**: The electricity prices are the dual variables of the power flow constraint.)

Solution: The electricity prices can be calculated as the negative of the dual variable $\boldsymbol{\lambda}$. It can be observed that there is very little variance between the electricity price of each node. This is due to the fact that only one of the transmission lines is binding for the minimum transmission capacity constraint and none are binding at the maximum transmission capacity constraint, implying that there exists minimal congestion in the system, leading to a reduced variance in the network prices. In contrast, when we scale down the transmission line capacity and introduce congestion in the system it results in greater variance in the prices of the nodes. An interesting observation can be made that the price is the lowest for node 89 which is the only bus for which the transmission line index 159 is binding for the minimum capacity constraint, meaning thereby that its minimum

capacity is equivalent to the greatest value possible, resulting in minimal congestion when compared to the other lines and buses.

In [6]:

```
import networkx as nx # !pip install networkx
import matplotlib.pyplot as plt

branches = pd.read_csv("../datasets/branches_ieee118_subset.csv")


G = nx.Graph()
G.add_edges_from([(fbus,tbus) for fbus,tbus in zip(branches.fbus,branches.tbus)])

N = len(G.nodes())

node_color = np.random.rand(N,1)*100 # You should provide a list of electricity prices. The order should match that of G.nodes()

plt.figure(figsize=(10,10))
nx.draw(G, pos=nx.kamada_kawai_layout(G), with_labels=True)
nc = nx.draw_networkx_nodes(G, pos=nx.kamada_kawai_layout(G), nodelist=list(G.nodes()), node_color=node_color, cmap=plt.cm.jet)
plt.colorbar(nc,label = "LMP ($/MWh)")
```

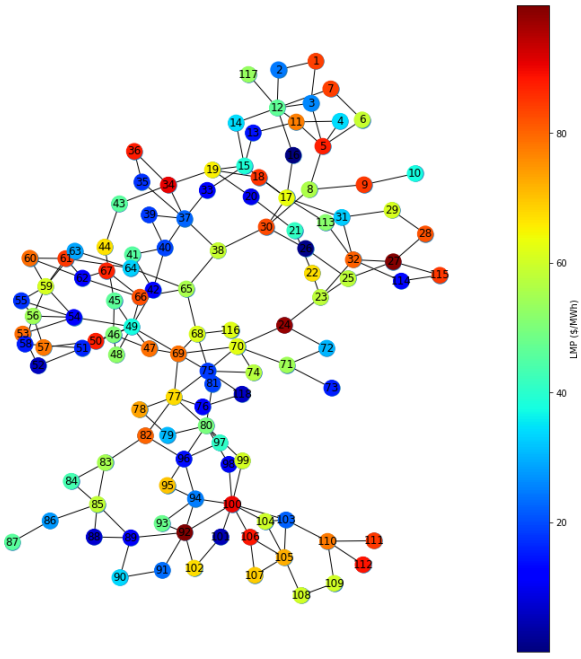

Figure 1: IEEE-118 LMPs (Prices are a random sample)

In [7]:

```
np.var(problem.constraints[0].dual_value)
```

Out[7]:
```
8.47973782315474e-06
```

It can be observed that the total variance in prices between all nodes is **8.47973782315474e-06**. The plot for the network prices can be observed in **Figure 2**

In [8]:
```
!pip install networkx
import matplotlib.pyplot as plt
import networkx as nx # !pip install networkx import matplotlib.pyplot as plt
branches = pd.read_csv("/kaggle/input/energy-systems-hw2/datasets/branches_ieee118_subset.csv")
G = nx.Graph()
G.add_edges_from([(fbus,tbus) for fbus,tbus in zip(branches.fbus,branches.tbus)])
N = len(G.nodes())
#node_color = np.random.rand(N,1)*100 # You should provide a list of electricity prices. The order should match that of G.nodes()
node_color = -problem.constraints[0].dual_value
fig = plt.figure(figsize=(10,10))
plt.title('Plot of the Original Optimal Flow Problems')
nx.draw(G, pos=nx.kamada_kawai_layout(G), with_labels=True)
nc = nx.draw_networkx_nodes(G, pos=nx.kamada_kawai_layout(G), nodelist=list(G.nodes()), node_color=node_color , cmap=plt.cm.jet)
plt.colorbar(nc,label = "LMP (\$/MWh)")
fig.savefig('original_flow.png')
```

(c) (**1.5 pts**) Now repeat parts (a) and (b). Instead, use 1/3 of the transmission capacity you used previously, in other words $\bar{\mathbf{f}}' = -\underline{\mathbf{f}}' := \bar{\mathbf{f}}/3$. Answer the following

1. (**0.5 pts**) Provide the list of binding transmission constraints and plot the network prices.
   Solution:

In [9]:
```
f_max_new = f_max/3
f_min_new = f_min/3

n = 118
m = 210
p = cp.Variable(54)
theta = cp.Variable(n)

constraints = [
    Egen @ p - d == B @ theta,
    p <= p_max,
    p >= p_min,
    f_min_new <= b_diag @ M.T @ theta,
    f_max_new >= b_diag @ M.T @ theta,
]

objective = cp.Minimize(p.T @ C2 @ p + c1 @ p + np.ones((1, len(c0))) @ c0)

problem_2 = cp.Problem(objective, constraints)
problem_2.solve()
```

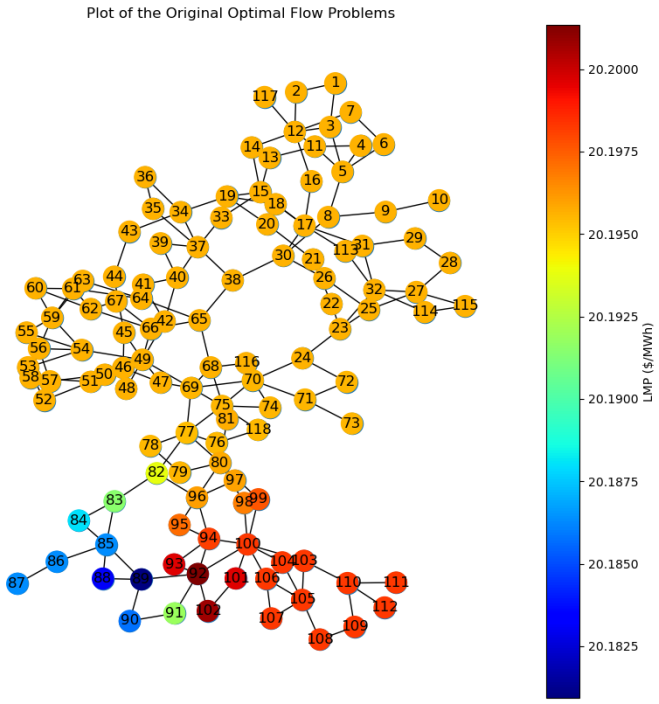

Figure 2: Electricity prices of the network plotted

Out[9]:
```
934.0758104362966
```

In [10]:
```
print(f'Value of power Injection p is: {p.value}')
```

Out[10]:
```
Value of power Injection p is: [ 3.12249619e-07  5.47436730e-02
   3.57328776e-07 -8.24063940e-08
  1.57525159e+00  1.85000020e+00 -2.78100395e-07  9.99999582e-01
  3.84842306e-01 -5.49726259e-07  2.80583295e+00  5.71432617e-01
 -3.28168935e-07  1.06999973e+00 -4.10484446e-07  4.78105630e-07
  7.79535621e-01  6.90535105e-01 -1.04099108e-06  1.09364346e+00
  3.04000023e+00  1.48000015e+00  1.45239321e-07  1.48169399e-07
  2.55000008e+00  2.05862158e+00  3.94487421e-08  2.38476625e+00
  2.34959594e+00  4.67478610e+00 -8.84819708e-07 -7.10687481e-07
 -8.56800408e-07 -1.22641641e-06  1.33385006e-01  1.87663116e-01
  2.51233676e+00  1.10264863e-07  6.66667088e-01  2.34151024e+00
  6.61510567e-01  2.63877793e-07  6.96529219e-07  3.59889497e-07
  3.52000045e+00  1.40000046e+00  4.67494291e-07  4.68781802e-07
  4.68970036e-07  4.88919664e-07  5.83339984e-01  4.89368344e-07
 -5.63036143e-07  1.12663864e-07]
```

In [11]:

```
import matplotlib.pyplot as plt
import networkx as nx # !pip install networkx import
    matplotlib.pyplot as plt
branches = pd.read_csv("/kaggle/input/energy-systems-hw2/
    datasets/branches_ieee118_subset.csv")
G = nx.Graph()
G.add_edges_from([(fbus,tbus) for fbus,tbus in zip(branches.
    fbus,branches.tbus)])
N = len(G.nodes())
#node_color = np.random.rand(N,1)*100 # You should provide a
     list of electricity prices. The order should match that
     of G .nodes()
node_color = -problem_2.constraints[0].dual_value
fig = plt.figure(figsize=(10,10))
plt.title('Plot of the Optimal Power Flow with new
    Constraints')
nx.draw(G, pos=nx.kamada_kawai_layout(G), with_labels=True)
nc = nx.draw_networkx_nodes(G, pos=nx.kamada_kawai_layout(G)
    ,nodelist=list(G.nodes()), node_color=node_color , cmap=
    plt.cm.jet)
plt.colorbar(nc,label = "LMP (\$/MWh)")
fig.savefig('new_flow.png')
```

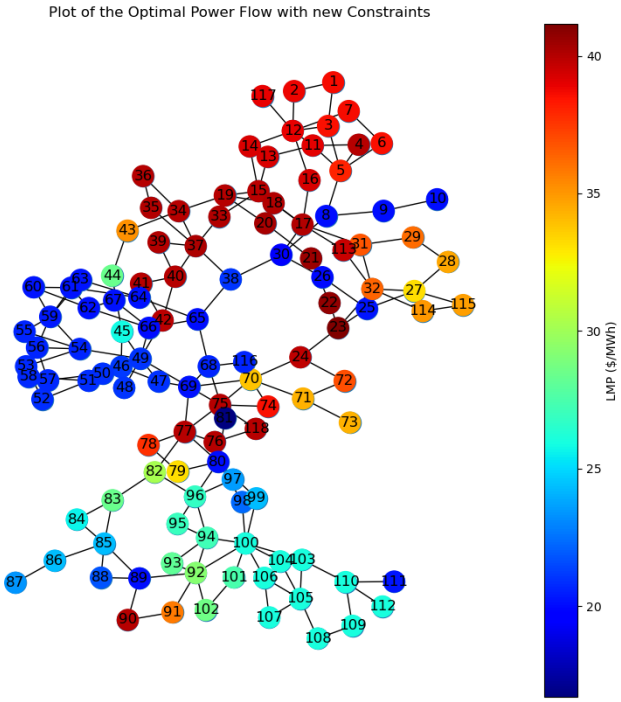

Figure 3: Electricity prices of the network after scaling down the transmission capacity by 1/3rd

2. (**0.5 pts**) Report the objective cost. Did it increase? Why?

   Solution: The objective cost was increased to **934.0758104362966**. The objective increases due to the fact that the decrease in the transmission capacity of transmission lines increases the amount of power that needs to be injected into the network by the generators. An increase in injected power without an equal decrease in the c1, c0 and c2 variables results in an increase in the objective function.

3. (**0.5 pts**) Why did the electricity prices change?

   Solution: The electricity prices change due to the fact that reducing the line capacity causes conjestion leading to an increase in prices.

(d) (**1.5 pts**) In this section, we are going to add a wind farm at node 90. For the time instance we are solving, the wind farm can provide up to 50 MW. The wind farm has a cost of 0, i.e. $c_2 = c_1 = c_0 := 0$, i.e. the wind farm bids at negative prices. In practice, wind farms bid at zero or even negative cost. Answer the following questions.

1. (**0.25 pts**) Resolve the optimization problem and plot the network prices.
   Solution:

In [12]:

```
generators.loc[40.5] = [90, 100, 0, 0, 0, 50, 0]
generators = generators.sort_index().reset_index(drop=True)

c2_new = np.array(generators.c2)
c1_new = np.array(generators.c1)
c0_new = np.array(generators.c0)

p_max_new = np.array(generators.Pmax)/100
p_min_new = np.array(generators.Pmin)/100
C2_new = np.diag(c2_new)

Egen_new = np.zeros((118, 55))
for idx, bus in enumerate(generators.bus):
    Egen_new[bus-1, idx] = 1
Egen_new

n = 118
m = 210
p = cp.Variable(55)
theta = cp.Variable(n)

constraints = [
    Egen_new @ p - d == B @ theta,
    p <= p_max_new,
    p >= p_min_new,
    f_min_new <= b_diag @ M.T @ theta,
    f_max_new >= b_diag @ M.T @ theta,
]

objective = cp.Minimize(p.T @ C2_new @ p + c1_new @ p + np.
    ones((1, len(c0_new))) @ c0_new)

problem_3 = cp.Problem(objective, constraints)
problem_3.solve()
```

Out[12]:

```
914.0716010829331
```

In [13]:
```
import matplotlib.pyplot as plt
import networkx as nx # !pip install networkx import
    matplotlib.pyplot as plt
branches = pd.read_csv("/kaggle/input/energy-systems-hw2/
    datasets/branches_ieee118_subset.csv")
G = nx.Graph()
G.add_edges_from([(fbus,tbus) for fbus,tbus in zip(branches.
    fbus,branches.tbus)])
N = len(G.nodes())
#node_color = np.random.rand(N,1)*100 # You should provide a
     list of electricity prices. The order should match that
     of G .nodes()
node_color = -problem_3.constraints[0].dual_value
fig = plt.figure(figsize=(10,10))
plt.title('Plot of the Optimal Power Flow with Wind Farm
    added to Node 90')
nx.draw(G, pos=nx.kamada_kawai_layout(G), with_labels=True)
nc = nx.draw_networkx_nodes(G, pos=nx.kamada_kawai_layout(G)
    ,nodelist=list(G.nodes()), node_color=node_color , cmap=
    plt.cm.jet)
plt.colorbar(nc,label = "LMP (\$/MWh)")
fig.savefig('new_flow_after_adding_wind_farm_node_90.png')
```

The plot of the electricity prices of the network after adding a wind farm at node 90 can be observed in **Figure 4**

2. (**0.25 pts**) Report the new objective cost and comment your results. Did the objective increase or decrease? Why?

   Solution: The new objective cost is **914.0716010829331** which is a slight decrease from the previous objective cost of **934.0758104362966**. The objective cost decreases since the wind farm is bidding at zero cost in addition to which **it addresses the congestion in the network by providing 50MW of power to the network at 0 cost.**

3. (**0.25 pts**) What happened to the nodal prices at node 90 and its neighbors?

   Solution: The nodal prices at node 90 and node 91 remain the same after the addition of the wind farm due to the fact that the wind farm bids at zero cost.

4. (**0.5 pts**) How much power is the wind farm injecting into the network? Is it at its limit? Why is it (not) at its limit?
   Solution:

In [14]:
```
p.value[41] # Power injected at node 90
```

Out[14]:
```
0.4999999760094896
```

It can be observed that the wind farm is injecting 49.999MW which is roughly its maximum power generation capacity of 50MW into the system. The reason it is performing at its limit in the system is due to the fact that it is bidding at zero cost

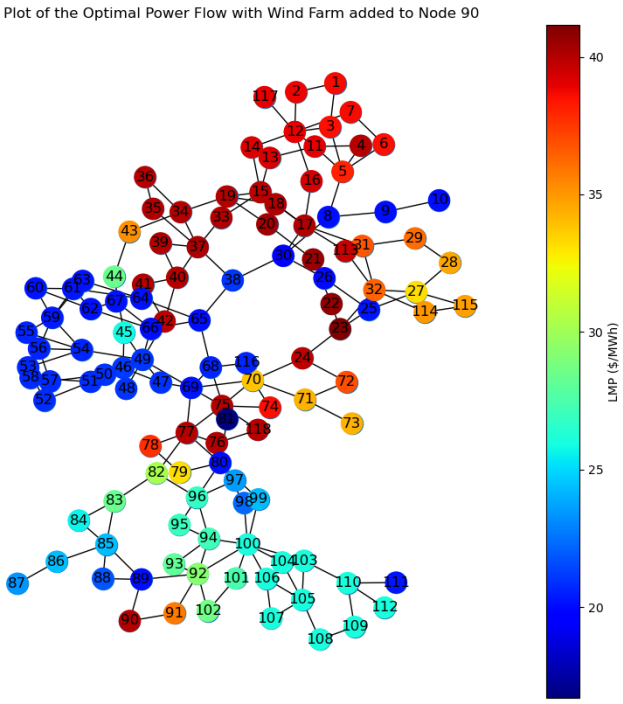

Figure 4: Electricity prices of the network after scaling down the transmission capacity by 1/3rd **and adding a Wind Farm at node 90**

due to which the objective cost remains unchanged for however large power injected by the wind farm. Therefore, since c2, c0 and c1 of the wind farm are minimized to be 0, the wind farm generates power at its maximum limit of 50MW.

5. (**0.25 pts**) Based on your observations through this problem, do you see any limitation to the integration of wind energy? (**Hint**: Transmission is expensive and locations with good wind resources tend to be in remote areas)

   Solution: Wind energy is seasonal and wind farms would rarely provide their maximum power capacity. Additionally, the effectiveness of the wind farm would vary with its location as the power generation capacity of a wind farm is always constrained by the availability of wind resources. Additionally, due to wind farms being located in remote locations, the transmission cost is also greater.

(e) (**1 pts**) Now increase the load at bus 90 by 50 MW. Solve the optimization problem, plot the network prices, and report the objective. What happened to the electricity prices? Comment your results.
Solution:

In [15]:
```
d[89] += 0.5
n = 118
m = 210
p = cp.Variable(55)
theta = cp.Variable(n)

constraints = [
    Egen_new @ p - d == B @ theta,
    p <= p_max_new,
    p >= p_min_new,
    f_min_new <= b_diag @ M.T @ theta,
    f_max_new >= b_diag @ M.T @ theta,
]

objective = cp.Minimize(p.T @ C2_new @ p + c1_new @ p + np.ones
    ((1, len(c0_new))) @ c0_new)

problem_final = cp.Problem(objective, constraints)
problem_final.solve()
```

Out[15]:
```
934.0758354516872
```

It can be observed that the final objective remains the same as it was when we scaled down the transmission capacity. This means that the addition of the wind farm did not help since an increase in power generation by 50MW is met with an equivalent increase of 50MW in the demand at the same node, due to which given constraint 1, both the added power and demand cancel out giving us the exact same network that we obtained in 1c.

In [16]:
```
print(f'Value of power Injection p is: {p.value}')
```

Out[16]:
```
Value of power Injection p is: [ 3.09503707e-07  5.47432795e-02
    3.55339123e-07 -8.38575012e-08
  1.57525181e+00  1.85000020e+00 -2.96140459e-07  9.99999559e-01
  3.84844726e-01 -5.46556317e-07  2.80583281e+00  5.71432824e-01
 -3.39130397e-07  1.06999972e+00 -4.23353644e-07  4.96634804e-07
  7.79532758e-01  6.90536038e-01 -1.07812592e-06  1.09364362e+00
  3.04000024e+00  1.48000016e+00  1.51802131e-07  1.54910476e-07
  2.55000008e+00  2.05862136e+00  3.97857821e-08  2.38476630e+00
  2.34959618e+00  4.67478468e+00 -8.64553144e-07 -6.99371363e-07
 -8.37997658e-07 -1.19667934e-06  1.33383435e-01  1.87665999e-01
  2.51233729e+00  1.07768857e-07  6.66667065e-01  2.34150969e+00
  6.61510516e-01  4.99999978e-01  2.57310586e-07  6.79526802e-07
  3.50481233e-07  3.52000044e+00  1.40000045e+00  4.55584622e-07
  4.56835744e-07  4.57022821e-07  4.76397491e-07  5.83339758e-01
  4.76836523e-07 -5.92856134e-07  1.10525248e-07]
```

In [17]:

```
import matplotlib.pyplot as plt
import networkx as nx # !pip install networkx import matplotlib
    .pyplot as plt
branches = pd.read_csv("/kaggle/input/energy-systems-hw2/
    datasets/branches_ieee118_subset.csv")
G = nx.Graph()
G.add_edges_from([(fbus,tbus) for fbus,tbus in zip(branches.
    fbus,branches.tbus)])
N = len(G.nodes())
#node_color = np.random.rand(N,1)*100 # You should provide a
    list of electricity prices. The order should match that of G
     .nodes()
node_color = -problem_final.constraints[0].dual_value
fig = plt.figure(figsize=(10,10))
plt.title('Plot of the Optimal Power Flow after increasing load
     for Node 90 by 50MW')
nx.draw(G, pos=nx.kamada_kawai_layout(G), with_labels=True)
nc = nx.draw_networkx_nodes(G, pos=nx.kamada_kawai_layout(G),
    nodelist=list(G.nodes()), node_color=node_color , cmap=plt.
    cm.jet)
plt.colorbar(nc,label = "LMP (\$/MWh)")
fig.savefig('new_flow_after_increasing_load_by_50mw_90.png')
```

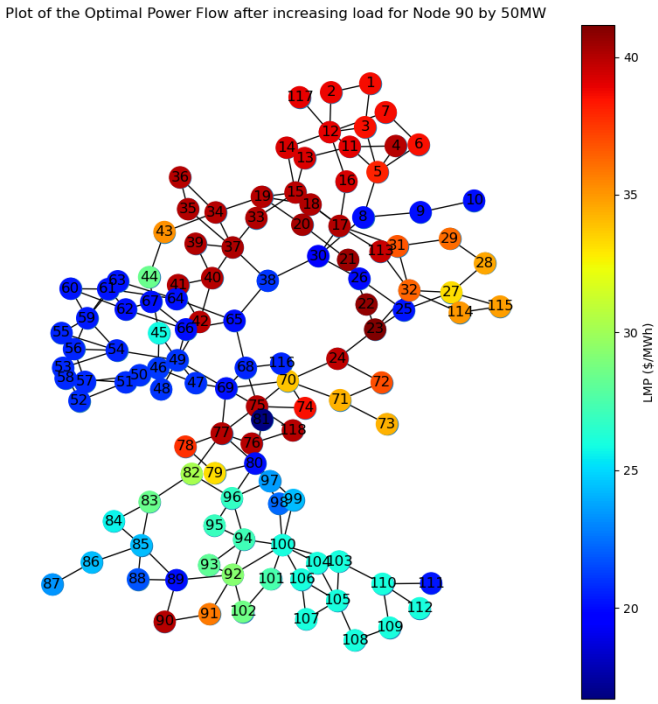

Figure 5: Electricity prices of the network after scaling down the transmission capacity by 1/3rd **and increasing the load by 50MW at node 90 and adding a wind farm at node 90**

2. (2 points) **AC Optimal Power Flow**: In this problem, we will use *pandapower* to run a DC and AC optimal power flow on the IEEE-118 test case. Please note that the network available on pandapower may not perfectly match the network used in the previous problem. Thus, you should not rely on any previous data you have used.

   (a) (**0.75 pts**) What are the differences between an AC and DC optimal power flow? What is similar? What is different? List the assumptions of each model and the variables that are needed in one vs the other.

   Solution: The main difference between AC-OPF and DC-OPF is that the AC-OPF makes use of non-linear AC Powerflow equations whereas the DC-OPF makes use of a simpler model with linear powerflow equations. Assumptions and Variables for AC-OPF -

   1. Voltage magnitudes and phase angles are taken for each bus as decision variables.
   2. Reactive power loads and sources are included in the AC-OPF model.
   3. Therefore, AC-OPF requires the active and reactive power generation at each generator, the voltage magnitudes and phase angles at each bus and the reactive power flow at each bus.

   Assumptions and Variables for DC-OPF -

   1. Voltage magnitudes are taken as constants.
   2. Reactive power sources and loads are not included.
   3. DC-OPF makes use of the following variables - Active power generation at each generator, line powerflow in each transmission line, DC voltage at each bus.

   (b) (**0.25 pts**) Run a DC optimal power flow.
   Solution:

In [18]:
```
!pip install pandapower
import pandapower
import pandapower.networks

net_dc = pandapower.networks.case118()
net_ac = pandapower.networks.case118()

pandapower.rundcopp(net_dc, numba=False, verbose=False)
pandapower.runopp(net_ac, numba=False, verbose=False, ac=True)
```

   (c) (**0.25 pts**) Run an AC optimal power flow.
   Solution:

In [19]:
```
!pip install pandapower
import pandapower
import pandapower.networks

net_dc = pandapower.networks.case118()
net_ac = pandapower.networks.case118()

pandapower.rundcopp(net_dc, numba=False, verbose=False)
pandapower.runopp(net_ac, numba=False, verbose=False, ac=True)
```

   (d) (**0.25 pts**) Plot the DC and AC line flows (only active power). How different are the DC and AC solutions? Comment your results.
   Solution:

In [20]:
```
import matplotlib.pyplot as plt
line = np.linspace(0, 172, 173)
fig = plt.figure()
plt.title('Active Power Flow Plots DC vs AC')
plt.plot(line, net_dc.res_line['p_from_mw'], 'r',label='Active
    Power Flow for DC')
plt.plot(line, net_ac.res_line['p_from_mw'], 'b',label='Active
    Power Flow for AC')
plt.ylabel('Power (in) MW')
plt.xlabel('Line Indices')
plt.legend()
fig.savefig('ac_vs_dc_active_power_flow.png')
plt.show()
```

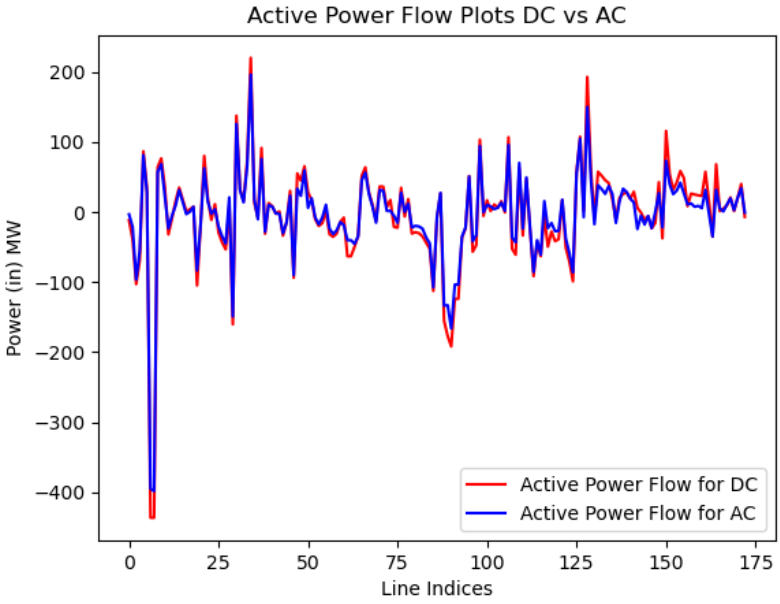

Figure 6: Active Power Flow plot of DC vs AC on the 'from bus'

In [21]:
```
line = np.linspace(0, 172, 173)
fig = plt.figure()
plt.title('Active Power Flow Plots DC vs AC')
plt.plot(line, net_dc.res_line['p_to_mw'], 'r',label='Active
    Power Flow for DC')
plt.plot(line, net_ac.res_line['p_to_mw'], 'b',label='Active
    Power Flow for AC')
plt.ylabel('Power (in) MW')
plt.xlabel('Line Indices')
plt.legend()
fig.savefig('ac_vs_dc_active_power_flow_to_bus.png')
plt.show()
```

It can be observed that the DC-OPF line flow solution achieves slightly higher peaks than the AC-OPF line flow solution while both the solutions are largely similar.

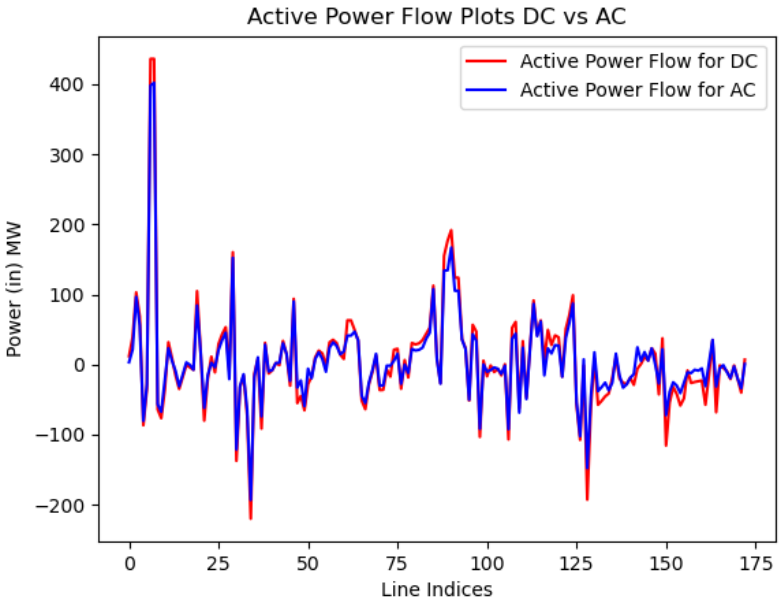

Figure 7: Active Power Flow plot of DC vs AC on the 'to bus'

(e) (**0.25 pts**) Plot the DC and AC active power generation. How different are the DC and AC solutions? Comment your results.
Solution:

In [22]:
```
gen = np.linspace(0, 52, 53)
fig = plt.figure()
plt.title('Active Power Generation Plots DC vs AC')
plt.plot(gen, net_dc.res_gen['p_mw'], 'r',label='Active Power
    Generation for DC')
plt.plot(gen, net_ac.res_gen['p_mw'], 'b',label='Active Power
    Generation for AC')
plt.ylabel('Power (in MW)')
plt.xlabel('Generator Indices')
plt.legend()
fig.savefig('ac_vs_dc_active_power_generation.png')
plt.show()
```

It can be observed that the plot for the DC power generation is slightly smoother than the plot for the AC power generation.

(f) (**0.25 pts**) Based on what you have observed, how do you think AC vs DC modeling could impact the solution of an optimal power flow?

Solution: AC-OPF is a non-convex optimization problem. It considers both the active and reactive power flows and generation in addition to the vector magnitudes and phase angles to each bus. Due to the involvement of these additional variables, the AC-OPF modelling should be more accurate for calculating the solution of Optimal Power Flow.

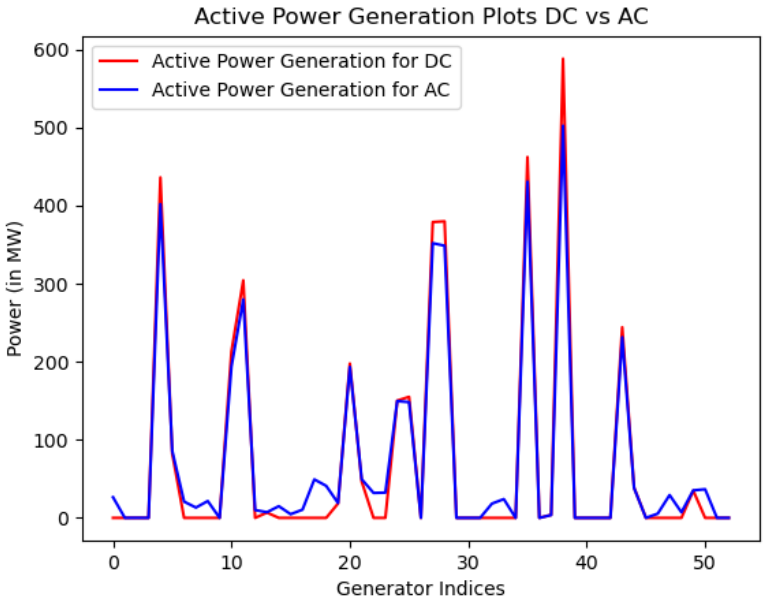

Figure 8: Active Power Generation plot of DC vs AC