Optimization Project

Group 54 - Fanze Meng, Jinyi Zhao

October 2023

1 Introduction

There are several generators and consumers scattered in 11 cities, they are connected by grids and we ignore the distance between two cities. In order to satisfy the demands of consumers and decrease the cost as much as possible, we need to set a certain voltage amplitude and phase in each node and control the generator production. To solve this problem, we will formulate an optimization problem to balance consumption and generation. We will talk about our solution in the following part.

2 Parameters and variables

2.1 Parameters

Parameter	Meaning	Units
$\mathbf{G}$	Location matrix of generators	
$\mathbf{U}$	Location matrix of users	
$\mathbf{A}$	Adjacency matrix of nodes	
$\mathbf{m}$	Maximum capacity vector of generators	
$\mathbf{u}$	Power demand vector of users	
$\mathbf{c}$	Cost vector of generators	
m_i	Maximum capacity of generator i	pu
u_j	Power demand of user j	pu
c_i	Power production cost of generator i	SEK/pu
b_{kl}	Real part of the shunt admittance	
g_{kl}	Imaginary part of the shunt admittance	

2.2 Variables

Variable	Meaning	Units
$\mathbf{P}$	Active power flowing matrix	
$\mathbf{Q}$	Reactive power flowing matrix	
p_{kl}	Active power flows from node k to node l	pu
q_{kl}	Reactive power flows from node k to node l	pu
$\mathbf{g}_a$	Active power vector of generators	
$\mathbf{g}_r$	Reactive power vector of generators	
a_i	Active power produced by generator i	pu
r_i	Reactive power produced by generator i	pu
v_k	Voltage amplitude in node k	vu
θ_k	Voltage phase in node k	radian

2.3 Definitions

2.3.1 Location matrices

We put the locations of generators and users into 9*11 matrix $\mathbf{G}$ and 7*11 matrix $\mathbf{U}$, where columns represent different nodes, rows represent

location within certain node. So we have $\mathbf{G}_{in} = 1$ means generator i is located in node n, and $\mathbf{U}_{jn} = 1$ means user j is located in node n.

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3.2 Adjacency matrix

This 11*11 matrix represents connections between nodes. $\mathbf{A}_{ij} = 1$ means node i and node j is adjacent, whereas $\mathbf{A}_{ij} = 0$ means they don't have connection.

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

2.3.3 Capacity, demand and cost

Here, we have three vectors that represent the maximum capacity of generators, the power demand of users, and the cost of generators. m_i, u_j, c_i represent the i(j)th element in the vector. m_i indicates the maximum capacity of generator i, c_i indicates its energy production cost. u_j equals to the demand of user j.

$$\mathbf{m} = \begin{bmatrix} 0.02 \\ 0.15 \\ 0.08 \\ 0.07 \\ 0.04 \\ 0.17 \\ 0.17 \\ 0.26 \\ 0.05 \end{bmatrix}, \mathbf{u} = \begin{bmatrix} 0.10 \\ 0.19 \\ 0.11 \\ 0.09 \\ 0.21 \\ 0.05 \\ 0.04 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} 175 \\ 100 \\ 150 \\ 150 \\ 300 \\ 350 \\ 400 \\ 300 \\ 200 \end{bmatrix}$$

2.3.4 Power flows

The expressions of p_{kl} and q_{kl} are given as follows.v means the voltage amplitude in certain node, θ is the voltage phase, b_{kl} and g_{kl} are constants.

$$p_{kl} = -v_k^2 g_{kl} + v_k v_l g_{kl} \cos(\theta_k - \theta_l) - v_k v_l b_{kl} \sin(\theta_k - \theta_l) \qquad k, l = 1, 2, 3 \ldots 11$$

$$q_{kl} = -v_k^2 b_{kl} + v_k v_l b_{kl} \cos(\theta_k - \theta_l) - v_k v_l g_{kl} \sin(\theta_k - \theta_l) \qquad k, l = 1, 2, 3 \ldots 11$$

2.3.5 Flow matrices

Form p_{kl} and q_{kl} into 11*11 matrices $\mathbf{P}$ and $\mathbf{Q}$. p_{kl} represents element locates in row k and column l, same as q_{kl}.

$$\mathbf{P} = [p_{kl}], \quad \mathbf{Q} = [q_{kl}] \qquad k, l = 1, 2, 3 \ldots 11$$

2.3.6 Power production

These 9 dimensions vector are built up by active power production a_i and reactive production r_i, i indicates different generator.

$$\mathbf{g}_a = [a_i], \quad \mathbf{g}_r = [r_i] \qquad i = 1, 2, 3 \ldots 9$$

3 Modeling

3.1 Objective Function

The objective function aims to minimize the overall cost of generators that are represented by the amount of power produced by each generator multiply the cost of each generator.

$$\min \ \mathbf{g}_a^{\mathrm{T}} * \mathbf{c}$$

3.2 Constraints

3.2.1 Active power constraint

$$\mathbf{g}_a^{\mathrm{T}} * \mathbf{G} * \mathbf{e}_k - \mathbf{u}^{\mathrm{T}} * \mathbf{U} * \mathbf{e}_k - \mathbf{e}_k^{\mathrm{T}} * \mathbf{P} * \mathbf{A} * \mathbf{e}_k = 0 \qquad k = 1, 2, 3 \ldots 11$$

3.2.2 Reactive power constraint

$$\mathbf{g}_r^{\mathrm{T}} * \mathbf{G} * \mathbf{e}_k - \mathbf{e}_k^{\mathrm{T}} * \mathbf{Q} * \mathbf{A} * \mathbf{e}_k = 0 \qquad k = 1, 2, 3 \ldots 11$$

3.2.3 Generator capacity constraint

$$0 \leq a_i \leq m_i, \qquad |r_i| \leq 0.3\% * m_i \qquad i = 1, 2, 3 \ldots 9$$

3.2.4 Voltage constraint

$$0.98 \leq v_k \leq 1.02, \qquad -\pi \leq \theta_k \leq \pi \qquad k = 1, 2, 3 \ldots 11$$

4 Results

Optimal solution of objective function is 183.19 SEK

Node	Amplitude/vu	Phase/radians
1	1.0191	0.0026
2	1.0200	0.0072
3	1.0195	0.0034
4	1.0181	-0.0051
5	1.0187	-0.0012
6	1.0181	-0.0059
7	1.0184	-0.0025
8	1.0184	-0.0028
9	1.0192	0.0018
10	1.0190	0.0012
11	1.0192	0.0027

表 1: Amplitude and phase within nodes

Generator	Active power/pu	Reactive power/pu
1	0.02	-0.000023
2	0.15	-0.001178
3	0.08	-0.000354
4	0.07	0.001461
5	0.04	0.001010
6	0.12	0.000781
7	0.00	0.000633
8	0.26	0.000933
9	0.05	0.000035

表 2: Active power and reactive power produced by generators

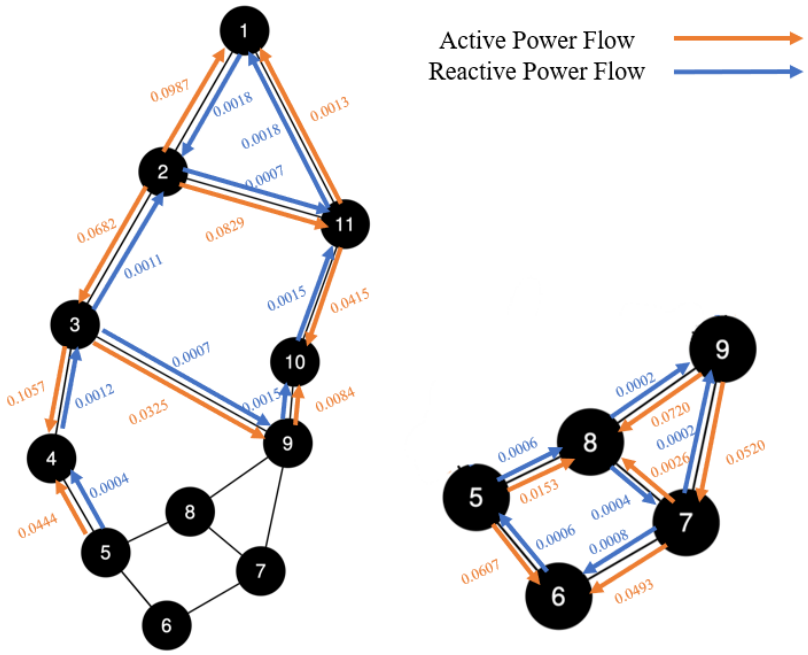

图 1: Power flow within nodes

4.1 Analysis

From the outputs, we can see that the optimal cost is 183.19 SEK. To achieve this optimal solution we need to adjust the voltage amplitude and phase in each node to a certain value as we calculated in Table 1. We also get the exact active power each generator needs to produce in Table 2. According to our results, generator 7 does not need to work at all, which may be due to its high production cost. As we don't count the transition cost, we just neglect the high degree of accessibility of generator 7, which is an advantage in reality. Generators 2 and 8 both have extraordinary workloads. Generator2 is simply because of its lowest cost and large capacity. Generator 8 has a large capacity and relatively low cost when other cheap generators meet their limit and the demand is still not met, so it has a hard workload as well. Here turns a question, is this solution just locally optimal or it can be globally optimal? The objective function is apparently not convex,

which means the IPOPT algorithm we used may find different local minima depending on the initial point. So it is just a locally optimal solution.

Here is another question, what if we want to upgrade the whole system to meet a cheaper cost, but we have limited funds that only have the ability to improve the maximum capacity of one generator by 0.01 pu, which generator we should choose? To answer this question we need to use some outputs which seem useless before. Here are them

Generator	1	2	3	4	5	6	7	8	9
Power	-174.021	-249.021	-199.021	-199.432	-50.418	-1.195	0.000	-49.647	-149.647

表 3: Dual variables corresponding to the maximum capacity constraints

According to the shadow price theorem, $\nabla v(b) = y^*$, here the y^* is in table 4. So aim to deduce z^* which is the primal problem, we need to deduce $w^* = b^T y^* = v(b)$ which is the dual problem. To deduce $v(b)$ as possible as we can, we need to find b_i that has the lowest $\nabla v(b)$. Obviously, the answer is b_2, which means we should upgrade generator 2. At the same time, it matches our intuition. After all, generator 2 has the lowest cost among them, as we don't consider the transition cost.

5 Summary

In this project, we solved the problem by simplifying the reality problem, formulating an optimization problem, and solving it by julia. Finally, we get a locally optimal solution 183.19 SEK. We weren't concerned about the transition cost of this problem if we added this effect to the objective function, the conclusion may have changed a lot. That is interesting to further thinking.

6 Appendix

Listing 1: **main function.jl**

```
using JuMP
import Ipopt

#include("project_data.jl")

model = Model(Ipopt.Optimizer)

@variable(model, 0 <= act_power[i = 1:generator_num] <= generator_capacity[i])
@variable(model, theta_lb <= vol_angle[1:num_nodes] <= theta_ub)
@variable(model, voltage_lb <= theta[1:num_nodes] <= voltage_ub)
@variable(model, reactivepower_lb[i] <= react_power[i = 1:generator_num] <= reactivepower_ub[i])

@objective(model, Min, sum(act_power[i] * generator_cost[i] for i in 1:generator_num))

active_power_constraints = []
reactive_power_constraints = []
for i in 1 : num_nodes
  constraint = @NLconstraint(model,
    -  sum(
          (theta[i]^2.0 * g_kl[i, j] - theta[i] * theta[j] * g_kl[i, j]*
              cos(vol_angle[i] - vol_angle[j]) - theta[i] * theta[j] *
              b_kl[i, j] * sin(vol_angle[i] - vol_angle[j]))
          for j in 1:num_nodes
          )
    +  sum(
          act_power[j] for j in nodes_with_generators[i]
          )
    -  sum(
          consumerDemand[j] for j in nodes_with_consumers[i]
          )
    == 0)
  push!(active_power_constraints, constraint)
  constraint_1 = @NLconstraint(model,
       sum(
          react_power[k] for k in nodes_with_generators[i]
          )
```

```
35      -  sum(
36              -theta[i]^2.0 * b_kl[i, j] + theta[i] * theta[j] * b_kl[i, j]*
                    cos(vol_angle[i] - vol_angle[j]) - theta[i] * theta[j] *
                    g_kl[i, j] * sin(vol_angle[i] - vol_angle[j])
37              for j in 1:num_nodes
38              )
39      == 0)
40      push!(reactive_power_constraints, constraint_1)
41  end
42
43  optimize!(model)
44
45  println()
46  println("Termination statue: ", JuMP.termination_status(model))
47  println("Optimal(?) objective function value: ", JuMP.objective_value(model)
        )
48
49
50  println("Power generated: ",  round.(JuMP.value.(act_power), digits=6))
51  println("Max power: ",  round.(JuMP.value.(generator_capacity), digits=6))
52  println("Reactive power generated: ",
        round.(JuMP.value.(react_power), digits=6))
53  println("Reactive max +/-: ",
        round.(JuMP.value.(reactivepower_ub), digits=6))
54  println("Voltage amplitude: ",  round.(JuMP.value.(theta), digits=6))
55  println("Voltage angle: ",  round.(JuMP.value.(vol_angle), digits=6))
56  println("\nDual variables/Lagrange multipliers corresponding to some of the
        constraints: ")
57  println("active power constraints",
        round.(JuMP.dual.(active_power_constraints), digits=6))
58  println("reactive power constraints",
        round.(JuMP.dual.(reactive_power_constraints), digits=6))
59  println(JuMP.dual.(JuMP.UpperBoundRef.(act_power)))
60
61  act_power = JuMP.value.(act_power)
62  theta = JuMP.value.(theta)
63  vol_angle = JuMP.value.(vol_angle)
64  flow_active = zeros(num_nodes, num_nodes)
65  flow_reactive = zeros(num_nodes, num_nodes)'
66
```

```
println()
for (k, l) in edges
    flow_active[k, l] = p_kl(theta[k], theta[l], vol_angle[k], vol_angle[l], k, l)
    flow_reactive[k, l] = q_kl(theta[k], theta[l], vol_angle[k], vol_angle[l], k, l)
    flow_active[l, k] = p_kl(theta[l], theta[k], vol_angle[l], vol_angle[k], l, k)
    flow_reactive[l, k] = q_kl(theta[l], theta[k], vol_angle[l], vol_angle[k], l, k)
end

println()
for k in 1 : num_nodes
  for l in 1 : num_nodes
    if (flow_active[k,l] > 0)
      println("Flow of active power from $k to $l: ", round.(flow_active[k,l], digits = 6))
    end
  end
end

println()
for k in 1 : num_nodes
  for l in 1 : num_nodes
    if (flow_reactive[k,l] > 0)
      println("Flow of reactive power from $k to $l: ", round.(flow_reactive[k,l], digits = 6))
    end
  end
end
```