Electricity Production and Transmission in Sweden, a Simplified Model

Edoardo Mangia
edoardom@student.chalmers.se

Erik Albinsson
erikalbi@student.chalmers.se

Fredrik Nyström
nystromf@student.chalmers.se

November 6, 2024

Abstract

With this report, we would like to present a model that could describe the production and transmission of electricity system within Sweden. This is an exercise, therefore we will consider a simplified electric grid, with approximate geographical locations for the plants and plausible, but not real data.

Contents

1 Mathematical Formulation

In this section, we will first present the problem at hand, followed by a formal definition and discussion of the variables, the objective function, and the constraints that constitute the model. Each component will be carefully detailed to provide a clear understanding of the mathematical framework underlying the model and how it addresses the given problem.

1.1 Problem Presentation

In this project, we have derived a simplified model for planning electricity production and transmission within Sweden. The focus is on power plants where the production of electricity is considered plannable, such as hydro, thermal, and nuclear power plants. The objective of our model is to optimize the total production cost while ensuring that all consumer demands are met.

We represent the electricity grid as a graph (Figure 1), where the nodes correspond to locations containing power plants or consumers, or both. The edges represent transmission lines connecting the nodes. At each node, electricity is either generated, consumed, or both. The model accounts for both active and reactive power, including losses in transmission, voltage amplitudes, and phase angles to better approximate the real-world system.

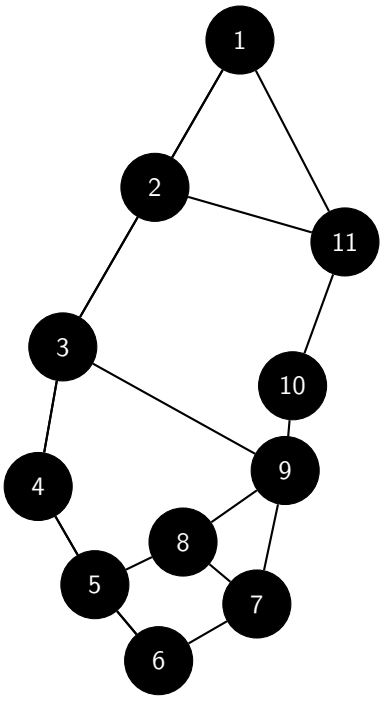

Figure 1: Diagram illustrating the connections between nodes.

The objective of the optimization model is to minimize the total cost of electricity production across the grid. This involves balancing the costs of generation and transmission, while ensuring efficient power flow between nodes.

The constraints of the model make sure that:

1. Power plants do not exceed their maximum capacity for electricity production.
2. Consumer demands at each node are fully met.
3. Voltage amplitudes and phase angles at all nodes remain within specified acceptable intervals.
4. The flow of active and reactive power, inbound and outbound of each node is balanced and equal.

To read in detail what these constraints are for our model view section 1.4.2. By solving this model, we gain insights into the most cost-effective configuration for meeting specific consumer demands while considering power production and transmission costs.

To implement this model, we used nonlinear optimization techniques, which helped us understand how the power plants contribute to the total cost and how power will optimally flow in our simplified network. Although the model is simplified, the approach can be extended to more complex, real-world scenarios.

In particular, to get the numerical results, Julia and the Ipopt and JuMP libraries were used.

1.2 Variables

1.2.1 Active and Reactive Power Production Vectors

In the $\mathbf{w}_a$ and $\mathbf{w}_r$ vectors (9×1) are stored respectively the active and reactive power production variables a_i and r_i, for each generator i.

$$\mathbf{w}_a^T = \begin{bmatrix} a_1 & \cdots & a_9 \end{bmatrix}$$

$$\mathbf{w}_r^T = \begin{bmatrix} r_1 & \cdots & r_9 \end{bmatrix}$$

1.2.2 Active and Reactive Power Flow Formulas

Considering an edge between 2 nodes kl, the flow of active power p_{kl} and reactive power g_{kl} can be described by the following equations:

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

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

- g_{kl} is the conductance coefficient. It's a scalar value that measures how easily electric current can flow through a conductor or circuit. It quantifies the ability of a material to conduct electric current. It's related to the active power flow and is typically associated with the resistance of the transmission lines.

- b_{kl} is the susceptance coefficient. It's a scalar that measures how easily reactive power can flow in an AC circuit. It quantifies the ability of a circuit to store and release reactive energy (energy stored in inductors and capacitors). It's related to the reactive power flow in the system and is typically associated with the inductance of the transmission lines.
- v_k represents the voltage amplitude in the node k. It refers to the maximum value of the voltage signal in a waveform. It represents how strong the voltage is at its peak at a given node.
- θ_k is the voltage phase angle in the node k. It describes the phase shift of the voltage waveform relative to a reference waveform. It indicates the position of the waveform in time concerning a specific point in its cycle.

1.2.3 Active and Reactive Power Flow Matrices

The variables p_{kl} and q_{kl} are organized in the **P** and **Q** matrices, both of size (11×11). They represent the power flows of active and reactive power in the edges of the network.

$$\mathbf{P} = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & \cdots & p_{1,11} \\ p_{2,1} & p_{2,2} & p_{2,3} & \cdots & p_{2,11} \\ p_{3,1} & p_{3,2} & p_{3,3} & \cdots & p_{3,11} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{11,1} & p_{11,2} & p_{11,3} & \cdots & p_{11,11} \end{bmatrix} \qquad \mathbf{Q} = \begin{bmatrix} q_{1,1} & q_{1,2} & q_{1,3} & \cdots & q_{1,11} \\ q_{2,1} & q_{2,2} & q_{2,3} & \cdots & q_{2,11} \\ q_{3,1} & q_{3,2} & q_{3,3} & \cdots & q_{3,11} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{11,1} & q_{11,2} & q_{11,3} & \cdots & q_{11,11} \end{bmatrix}$$

1.3 Parameters

1.3.1 Network Nodes Matrices

The **G** (9×11) matrix represents the nodes where a generator is present. In the network there are in fact 9 generators and 11 nodes. In the same fashion, the **D** (7×11) matrix represents whether a node is also a consumer node.

$$\boldsymbol{G} = \begin{bmatrix} g_{1,1} & g_{1,2} & g_{1,3} & \cdots & g_{1,11} \\ g_{2,1} & g_{2,2} & g_{2,3} & \cdots & g_{2,11} \\ g_{3,1} & g_{3,2} & g_{3,3} & \cdots & g_{3,11} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{9,1} & g_{9,2} & g_{9,3} & \cdots & g_{9,11} \end{bmatrix} \qquad \boldsymbol{D} = \begin{bmatrix} d_{1,1} & d_{1,2} & d_{1,3} & \cdots & d_{1,11} \\ d_{2,1} & d_{2,2} & d_{2,3} & \cdots & d_{2,11} \\ d_{3,1} & d_{3,2} & d_{3,3} & \cdots & d_{3,11} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{7,1} & d_{7,2} & d_{7,3} & \cdots & d_{7,11} \end{bmatrix}$$

$$G_{ij} = \begin{cases} 1 & \text{if generator } i \text{ is at node } j, \\ 0 & \text{otherwise.} \end{cases} \qquad D_{ij} = \begin{cases} 1 & \text{if consumer } i \text{ is at node } j, \\ 0 & \text{otherwise.} \end{cases}$$

The **A** (11×11) was build to represent relationship of adjacency between nodes. A node is adjacent to another if they are connected by one edge.

$$\boldsymbol{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,11} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,11} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,11} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{11,1} & a_{11,2} & a_{11,3} & \cdots & a_{11,11} \end{bmatrix} \qquad A_{ij} = \begin{cases} 1 & \text{if node } i \text{ is adjacent to node } j, \\ 0 & \text{otherwise.} \end{cases}$$

1.3.2 Generators Capacities, Costs and Consumer Nodes Demands Vectors

These 3 vectors store unique values for each generator and consumer node in the network.

- The $\mathbf{m}$ (9×1) vector stores the maximum capacity in pu of each generator.

$$\mathbf{m}^T = \begin{bmatrix} m_1 & \cdots & m_9 \end{bmatrix} = \begin{bmatrix} 0.02 & 0.15 & 0.08 & 0.07 & 0.04 & 0.17 & 0.17 & 0.26 & 0.05 \end{bmatrix}$$

- The $\mathbf{c}$ (9×1) vector stores the energy production cost in $\frac{SEK}{pu}$ of each generator.

$$\mathbf{c}^T = \begin{bmatrix} c_1 & \cdots & c_9 \end{bmatrix} = \begin{bmatrix} 175 & 100 & 150 & 150 & 300 & 350 & 400 & 300 & 200 \end{bmatrix}$$

- The $\mathbf{d}$ (7×1) vector stores the active power demand in pu of each consumer node.

$$\mathbf{d}^T = \begin{bmatrix} d_1 & \cdots & d_7 \end{bmatrix} = \begin{bmatrix} 0.10 & 0.19 & 0.11 & 0.09 & 0.21 & 0.05 & 0.04 \end{bmatrix}$$

1.4 Modeling

The objective of the optimization problem is to minimize the total cost of energy production in the network, with a few constraints to consider.

1.4.1 Objective Function

The objective function is therefore a function of the active power production for every generator a_i and their costs of production c_i.

$$\min \mathbf{w}_a^T \cdot \mathbf{c}$$

1.4.2 Constraints

Active and Reactive Power Balance Constraint For each node, the inbound and outbound flow of active and reactive power must be equal.

$$\underbrace{\mathbf{w}_a^T \cdot \mathbf{G} \cdot e_k}_{\text{Active power generated at node } k} - \underbrace{\mathbf{d}^T \cdot \mathbf{D} \cdot e_k}_{\text{Consumer demand at node } k} = \underbrace{e_k^T \cdot \mathbf{P} \cdot \mathbf{A} \cdot e_k}_{\text{Net active power flow at node } k}, \quad \forall k = 1, 2, \ldots, 11$$

$$\underbrace{\mathbf{w}_r^T \cdot \mathbf{G} \cdot e_k}_{\text{Reactive power generation at node } k} = \underbrace{e_k^T \cdot \mathbf{Q} \cdot \mathbf{A} \cdot e_k}_{\text{Net reactive power flow at node } k}, \quad \forall k = 1, 2, \ldots, 11$$

Generators Capacity Constraint Each generator can generate a non-negative amount of active power, up to some maximum capacity. The amount of reactive power a generator can generate is at most 3% of the maximum capacity of the generator.

$$0 \leq a_i \leq m_i \quad i = 1, 2, \ldots, 9$$

$$-3\% \cdot m_i \leq r_i \leq 3\% \cdot m_i, \quad i = 1, 2, \ldots, 9$$

Voltage Amplitudes and Phases Constraint The voltage amplitudes must be kept within 0.98 vu and 1.02 vu. The voltage phases are given radians, and must be kept between $-\pi$ and π.

$$0.98 \leq v_k \leq 1.02, \quad k = 1, 2, \ldots, 11$$

$$-\pi \leq \theta_k \leq \pi, \quad k = 1, 2, \ldots, 11$$

2 Results and Analysis

By optimizing this model with Ipopt, a locally optimal solution was found, with a total energy production cost of 183.24 SEK.

2.1 Power Flows Chart

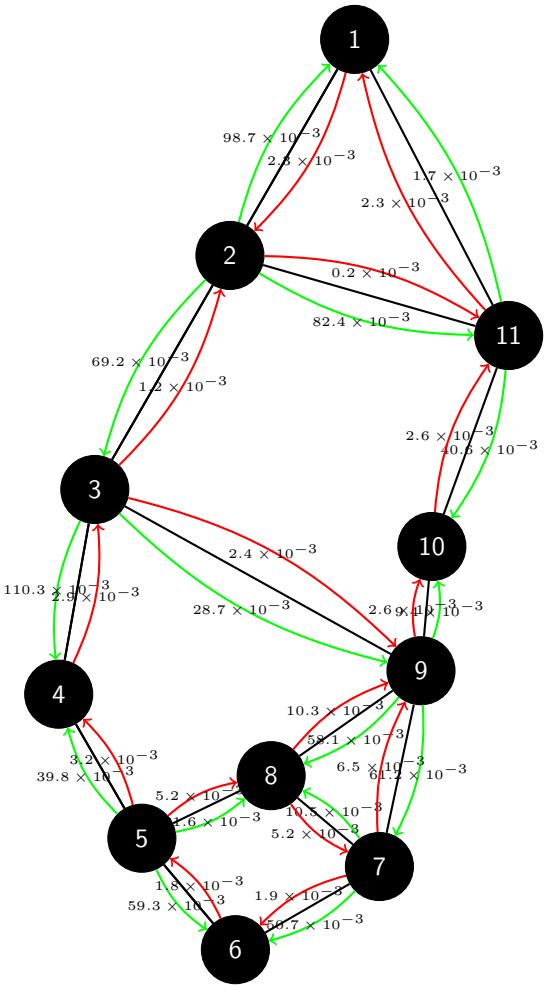

Figure 2: The simplified graph used in the model with the active and reactive power flows in pu.

2.2 Generators Power Production and Costs

Generator	Active Power (pu)	Reactive Power (pu)	Contribution (SEK)	Percentage of Total Cost
1	0.02 (0.02)	-4.3e-5	3.5	1.91%
2	0.15 (0.15)	-1.2e-3	15.0	8.19%
3	0.08 (0.08)	-6.4e-4	12.0	6.55%
4	0.07 (0.07)	1.8e-3	10.5	5.73%
5	0.04 (0.04)	1.1e-3	12.0	6.55%
6	0.12 (0.17)	6.3e-4	42.2	23.03%
7	0.00 (0.17)	9.6e-3	0.0	0.0%
8	0.26 (0.26)	1.5e-4	78.0	42.58%
9	0.05 (0.05)	5.7e-6	10.0	5.46%

Table 1: Active and reactive power for each generator, along with their contributions to the total cost.

Active Power Generation Most generators are operating near or at their maximum capacity (values in parentheses), as seen from the active power column in Table 3. Generators such as 2, 3, 4, 5, 8, and 9 are fully utilized. However, Generator 7 is not producing any active power, which could suggest that its operational costs (likely influenced by its high cost of 400 SEK) make it uncompetitive compared to other generators. This implies that cheaper generators are preferred by the optimization model.

Reactive Power Generation The reactive power values are relatively small, as expected in optimal power flow problems where the focus is on meeting active power demands. Some generators (such as Generator 2) have negative reactive power, which means they are absorbing reactive power from the grid rather than producing it. Furthermore, Generator 7 produces no active power but contributes reactive power, while Generator 9 shows the largest contribution to reactive power. This suggests these generators are playing a more significant role in managing voltage stability rather than meeting active power demands.

Cost Contributions Generator 8 contributes to 42.58% of the total cost, indicating it is heavily relied upon. It has the highest capacity (0.26 pu) and a moderate cost (300 SEK), making it a cost-effective but critical part of the network. Generator 6 also contributes significantly to the total cost (23.03%), even though its generation capacity is lower than that of Generator 8. This indicates that the optimization model is using it extensively to meet demand efficiently. Generators 1, 4, and 5 have smaller cost contributions (around 5-7%), indicating they are used for balancing the system but not heavily relied upon, likely due to their limited capacity.

2.3 Voltage Amplitude and Phase Values in the Nodes

Voltage Amplitude and Phase Values The voltage amplitude values are close to 1 pu for all nodes, with some slight variations due to reactive power flows. This suggests the voltage profile is well regulated across the network, as expected in a well-optimized grid. The phase angles are relatively small, indicating that there are minimal power flow imbalances in the network.

Node	Amplitude (vu)	Phase (radians)
1	1.019	0.003
2	1.020	0.007
3	1.019	0.003
4	1.018	-0.005
5	1.018	-0.001
6	1.018	-0.006
7	1.018	-0.002
8	1.018	-0.003
9	1.019	0.002
10	1.019	0.001
11	1.019	0.003

Table 2: Voltage amplitude and phase values for each node.

2.4 Sensitivity Analysis

With JuMP, it was possible to output the Lagrange multipliers for the dual problem (Table 5), to then derive some considerations.

The dual variables (or Lagrange multipliers) in the optimization problem provide insights into the sensitivity of the objective function with respect to the constraints. Below is the interpretation of the dual variables from the results.

Generator	Active Power Const	Reactive Power Const	Active Power LB	Active Power UB
1	349.6	2.5e-3	0.0	-173.5
2	349.0	-1.0e-6	0.0	-248.5
3	349.4	1.5e-5	0.0	-198.5
4	350.4	2.1e-4	0.0	-198.9
5	350.0	0.0	0.0	-50.2
6	350.4	1.2e-3	0.0	0.0
7	350.2	0.0	49.9	0.0
8	350.2	4.6e-4	0.0	-49.0
9	349.6	0.0	0.0	-149.0

Table 3: Dual Variables for Active and Reactive Power Constraints

Active Power Constraints Dual Variables These dual variables are associated with the active power balance constraints at each node. Each value represents how much the objective function (total cost) would increase if the active power requirement at that particular node were relaxed by a small amount.

All dual variables are significantly large and positive for the active power constraints. This suggests that the system is heavily constrained, and any increase in active power demand at those nodes will lead to a substantial increase in the total cost.

Reactive Power Constraints Dual Variables These dual variables correspond to the reactive power balance at each node. They indicate how much the objective function would change if the reactive power constraint at a particular node were relaxed slightly.

Most values are close to zero, suggesting that the reactive power balance constraints are either non-binding or only mildly influential on the cost at the optimal solution.

A few small positive values indicate that, for some nodes (such as generators 1, 4, 6 and 8), reactive power constraints are active, but their effect on the total cost is not as pronounced as for active power constraints.

Active Power Lower Bound Dual Variables These dual variables represent how much the objective function would change if the lower bound on active power generation (minimum power output) were relaxed.

For all generators except generator 7, the dual variables are zero, meaning none of these generators are constrained by their lower bound.

Generator 7 has a positive dual value, suggesting that it is producing exactly its minimum allowed power, and relaxing this lower bound would increase the cost by the amount of its dual value.

Active Power Upper Bound Dual Variables These dual variables reflect how much the objective function would change if the upper bound on active power generation (maximum power output) were relaxed.

Several generators have negative values, suggesting that they are producing at their maximum capacity, and increasing their allowed output would reduce the total cost significantly (especially for generators 1, 2, and 9).

Generator 7 has a dual value of zero, indicating that it is not generating at its upper bound, so increasing its generation limit would not impact the total cost.

3 Summary and Conclusions

The Ipopt optimizer used for the computations, found a locally optimal solution for the problem. However, while it could be also globally optimal, it is not guaranteed. The problem non-convex and the reason for this can be traced back to the problem formulation, in particular to the power balance constraints for the nodes (Section 1.4.2). They are in fact non-linear equality constraints, because of the presence of the *cos* and *sin* function in the formulation of the power flows from node to node (Section 1.2.2).

The network relies heavily on a few key generators (such as Generators 8 and 6) to meet the majority of the load. This is due to their higher capacities and moderate costs. Some generators are used primarily for reactive power generation and voltage regulation (e.g., Generators 7 and 9). This highlights the importance of voltage stability in the grid, even if those generators are not contributing significantly to active power demands.

The lack of use for Generator 7 can be attributed to its high cost relative to other generators. The optimization model minimizes total costs, and therefore, it avoids using the most expensive

generators unless absolutely necessary. Increasing the flexibility of Generator 7 or reducing its costs might make it more competitive and reduce strain on other generators.

To reduce reliance on a few large generators like Generator 8, it may be worth exploring system upgrades or integrating cheaper generation sources. Given the reactive power flows and phase angles, optimizing voltage control (through capacitors, for instance) could reduce the need for reactive power generation from generators, leading to further cost savings.

Generators 1, 2, and 9 are likely the most cost-effective generators, as they are operating at their maximum capacity (as indicated by their negative upper bound dual variables). Any relaxation of their upper bounds would reduce the total cost, suggesting that these generators could be further utilized if possible.

The reactive power balance constraints appear to be less critical overall compared to the active power constraints, as indicated by the small or near-zero dual values. The large dual variables for active power constraints reflect the high sensitivity of the total cost to the balance of active power across nodes. This shows the importance of maintaining a precise balance of active power in the network.

4 Appendix

4.1 Code

```
using JuMP
import Ipopt
using SparseArrays

# Graph edges representing connections between nodes
edges = [(1, 2), (1, 11), (2, 3), (2, 11), (3, 4), (3, 9), (4, 5),
    ↪ (5, 6), (5, 8), (6, 7), (7, 8), (7, 9), (8, 9), (9, 10), (10,
    ↪ 11)]

# Susceptance and conductance coefficients
susceptance_coefficients = [-20.1, -22.3, -16.8, -17.2, -11.7,
    ↪ -19.4, -10.8, -12.3, -9.2, -13.9, -8.7, -11.3, -7.7, -13.5,
    ↪ -26.7]
conductance_coefficients = [4.12, 5.67, 2.41, 2.78, 1.98, 3.23,
    ↪ 1.59, 1.71,     1.26, 1.11, 1.32, 2.01, 4.41, 2.14, 5.06]

# Generator data
generator_nodes = [2, 2, 2, 3, 4, 5, 7, 9, 9]
generator_capacities = [0.02, 0.15, 0.08, 0.07, 0.04, 0.17,
    ↪ 0.17, 0.26, 0.05]
generator_costs = [175, 100, 150, 150, 300, 350, 400, 300, 200]
consumer_demands = [0.10, 0.19, 0.11, 0.09, 0.21, 0.05, 0.04]
num_generators = length(generator_nodes)
```

```
# Data definitions
generators_at_nodes = [[], [1, 2, 3], [4], [5], [6], [], [7], [],
    ↪ [8, 9], [], []]
consumers_at_nodes = [[1], [], [], [2], [], [3], [], [4], [5], [6],
    ↪ [7]]
total_nodes = 11

# Reactive power limits based on generator capacities
reactive_power_upper_bound = 0.03 .* generator_capacities
reactive_power_lower_bound = -0.03 .* generator_capacities

# Voltage angle constraints
voltage_angle_upper_bound = pi
voltage_angle_lower_bound = -pi

# Voltage magnitude constraints
voltage_magnitude_upper_bound = 1.02
voltage_magnitude_lower_bound = 0.98

# Initialize conductance and susceptance matrices
susceptance_matrix = sparse(zeros(total_nodes, total_nodes))
conductance_matrix = sparse(zeros(total_nodes, total_nodes))

for (i, (node_k, node_l)) in enumerate(edges)
    susceptance_matrix[node_k, node_l] = susceptance_coefficients[i]
    susceptance_matrix[node_l, node_k] = susceptance_coefficients[i]
    conductance_matrix[node_k, node_l] = conductance_coefficients[i]
    conductance_matrix[node_l, node_k] = conductance_coefficients[i]
end

# Define active and reactive power flow functions
function calculate_active_power(voltage_mag_k, voltage_mag_l,
    ↪ voltage_angle_k, voltage_angle_l, k::Int, l::Int)::Float64
    return (voltage_mag_k^2.0 * conductance_matrix[k, l] -
        ↪ voltage_mag_k * voltage_mag_l * conductance_matrix[k, l]
        ↪       * cos(voltage_angle_k - voltage_angle_l) -
        ↪ voltage_mag_k * voltage_mag_l * susceptance_matrix[k, l]*
        ↪ sin(voltage_angle_k - voltage_angle_l))
end

function calculate_reactive_power(voltage_mag_k, voltage_mag_l,
    ↪ voltage_angle_k, voltage_angle_l, k::Int, l::Int)::Float64
    return (-voltage_mag_k^2.0 * susceptance_matrix[k, l] +
        ↪ voltage_mag_k * voltage_mag_l * susceptance_matrix[k, l] *
        ↪  cos(voltage_angle_k - voltage_angle_l) - voltage_mag_k *
```

```
        ↪ voltage_mag_l * conductance_matrix[k, l] * sin(
        ↪ voltage_angle_k - voltage_angle_l))
end



# Create optimization model
model = Model(Ipopt.Optimizer)

# Active power generated by each generator constrained by its
   ↪ capacity
@variable(model, 0 <= active_power[i = 1:num_generators] <=
   ↪ generator_capacities[i])

# Reactive power generated by each generator constrained between
   ↪ reactivepower_lb and reactivepower_ub
@variable(model, reactive_power_lower_bound[i] <= reactive_power[i =
   ↪  1:num_generators] <= reactive_power_upper_bound[i])

# Voltage magnitudes for each node, constrained between voltage_lb
   ↪ and voltage_ub
@variable(model, voltage_magnitude_lower_bound <= voltage_mag[1:
   ↪ total_nodes] <= voltage_magnitude_upper_bound)

# Voltage angles for each node, constrained between theta_lb and
   ↪ theta_ub
@variable(model, voltage_angle_lower_bound <= voltage_angle[1:
   ↪ total_nodes] <= voltage_angle_upper_bound)

# Minimize total cost: sum of each generator's active power
   ↪ multiplied by its cost
@objective(model, Min, sum(active_power[i] * generator_costs[i] for
   ↪ i in 1:num_generators))

# Constraints
active_power_constraints = []
reactive_power_constraints = []

for node in 1:total_nodes
    # Active power balance constraint
    constraint1 = @NLconstraint(model,
    # Sums the active power generated by generator "j" at node "node
       ↪ "
        + sum(
          active_power[j] for j in generators_at_nodes[node]
         )
```

```
    # Sums the active power requested by the consumer "j" at node "
       ↪ node"
        - sum(
          consumer_demands[j] for j in consumers_at_nodes[node]
         )
    ==
        + sum(
          voltage_mag[node]^2.0 * conductance_matrix[node, j] -
             ↪ voltage_mag[node] * voltage_mag[j] *
             ↪ conductance_matrix[node, j] * cos(voltage_angle[node
             ↪ ] - voltage_angle[j]) - voltage_mag[node] *
             ↪ voltage_mag[j] * susceptance_matrix[node, j] * sin(
             ↪ voltage_angle[node] - voltage_angle[j])
          for j in 1:total_nodes
         )
        )
    push!(active_power_constraints, constraint1)

    # Reactive power balance constraint
    constraint2 = @NLconstraint(model,
        # Sums the reactive power generated by generator "j" at node
           ↪  "node"
    + sum(
          reactive_power[k] for k in generators_at_nodes[node]
         )
        ==
        + sum(
          -voltage_mag[node]^2.0 * susceptance_matrix[node, j] +
             ↪ voltage_mag[node] * voltage_mag[j] *
             ↪ susceptance_matrix[node, j] * cos(voltage_angle[node
             ↪ ] - voltage_angle[j]) - voltage_mag[node] *
             ↪ voltage_mag[j] * conductance_matrix[node, j] * sin(
             ↪ voltage_angle[node] - voltage_angle[j])
          for j in 1:total_nodes
         )
    )
    push!(reactive_power_constraints, constraint2)
end



# Optimize the model
optimize!(model)
```

```
# Retrieve solution values
active_power_values = JuMP.value.(active_power)
voltage_magnitude_values = JuMP.value.(voltage_mag)
voltage_angle_values = JuMP.value.(voltage_angle)

# Output results
println("\n--- Optimization Results ---")
println("\nTermination status: ", JuMP.termination_status(model))
println("Objective function value (Total Cost): ", round(JuMP.
    ↪ objective_value(model), digits=6))

# Active power generated
println("\nActive Power Generated by Each Generator:")
for i in 1:num_generators
    println("Generator $i: ", round(active_power_values[i], digits
        ↪ =2), " (Max Capacity: ", round(generator_capacities[i],
        ↪ digits=6), ")")
end

# Reactive power generated
println("\nReactive Power Generated by Each Generator:")
for i in 1:num_generators
    println("Generator $i: ", round(JuMP.value(reactive_power[i]),
        ↪ digits=6), " (Reactive Limits:   ", round(
        ↪ reactive_power_upper_bound[i], digits=6), ")")
end

# Voltage magnitudes and angles
println("\nVoltage Magnitudes and Angles at Each Node:")
for i in 1:total_nodes
    println("Node $i: Voltage Magnitude = ", round(
        ↪ voltage_magnitude_values[i], digits=6), ", Voltage Angle =
        ↪ ", round(voltage_angle_values[i], digits=6))
end

# Dual variables / Lagrange multipliers
println("\nDual Variables / Lagrange Multipliers:")

# Format for Active Power Constraints
println("\nActive Power Constraints:")
for (i, dual_value) in enumerate(round.(JuMP.dual.(
    ↪ active_power_constraints), digits=6))
    println("Generator $i: ", dual_value)
end

# Format for Reactive Power Constraints
```

```
println("\nReactive Power Constraints:")
for (i, dual_value) in enumerate(round.(JuMP.dual.(
    ↪ reactive_power_constraints), digits=6))
    println("Generator $i: ", dual_value)
end

# Dual values for active power bounds (lower and upper constraints)
println("\nActive Power Lower Bound Dual Variables:")
for i in 1:num_generators
    println("Generator $i: ", round(JuMP.dual(JuMP.LowerBoundRef(
        ↪ active_power[i])), digits=6))
end

println("\nActive Power Upper Bound Dual Variables:")
for i in 1:num_generators
    println("Generator $i: ", round(JuMP.dual(JuMP.UpperBoundRef(
        ↪ active_power[i])), digits=6))
end

# Voltage Magnitude Lower Bound Dual Variables
println("\nVoltage Magnitude Lower Bound Dual Variables:")
for i in 1:total_nodes
    println("Node $i: ", round(JuMP.dual(JuMP.LowerBoundRef(
        ↪ voltage_mag[i])), digits=6))
end

# Voltage Magnitude Upper Bound Dual Variables
println("\nVoltage Magnitude Upper Bound Dual Variables:")
for i in 1:total_nodes
    println("Node $i: ", round(JuMP.dual(JuMP.UpperBoundRef(
        ↪ voltage_mag[i])), digits=6))
end

# Voltage Angle Lower Bound Dual Variables
println("\nVoltage Angle Lower Bound Dual Variables:")
for i in 1:total_nodes
    println("Node $i: ", round(JuMP.dual(JuMP.LowerBoundRef(
        ↪ voltage_angle[i])), digits=6))
end

# Voltage Angle Upper Bound Dual Variables
println("\nVoltage Angle Upper Bound Dual Variables:")
for i in 1:total_nodes
    println("Node $i: ", round(JuMP.dual(JuMP.UpperBoundRef(
        ↪ voltage_angle[i])), digits=6))
end
```

```
# Calculate and display power flows
active_power_flows = zeros(total_nodes, total_nodes)
reactive_power_flows = zeros(total_nodes, total_nodes)

for (k, l) in edges
    active_power_flows[k, l] = calculate_active_power(
       ↪ voltage_magnitude_values[k], voltage_magnitude_values[l],
       ↪ voltage_angle_values[k], voltage_angle_values[l], k, l)
    reactive_power_flows[k, l] = calculate_reactive_power(
       ↪ voltage_magnitude_values[k], voltage_magnitude_values[l],
       ↪ voltage_angle_values[k], voltage_angle_values[l], k, l)
        active_power_flows[l, k] = calculate_active_power(
           ↪ voltage_magnitude_values[l], voltage_magnitude_values[
           ↪ k], voltage_angle_values[l], voltage_angle_values[k],
           ↪ l, k)
    reactive_power_flows[l, k] = calculate_reactive_power(
       ↪ voltage_magnitude_values[l], voltage_magnitude_values[k],
       ↪ voltage_angle_values[l], voltage_angle_values[k], l, k)
end

println("\nActive␣Power␣Flows:")
for k in 1:total_nodes
    for l in 1:total_nodes
        if active_power_flows[k, l] > 0
            println("Active␣power␣flow␣from␣node␣$k␣to␣node␣$l:␣",
               ↪ round(active_power_flows[k, l], digits=6))
        end
    end
end

println("\nReactive␣Power␣Flows:")
for k in 1:total_nodes
    for l in 1:total_nodes
        if reactive_power_flows[k, l] > 0
            println("Reactive␣power␣flow␣from␣node␣$k␣to␣node␣$l:␣",
               ↪  round(reactive_power_flows[k, l], digits=6))
        end
    end
end

# After optimizing the model, calculate cost contributions
active_power_values = JuMP.value.(active_power)
total_cost = round(JuMP.objective_value(model), digits=6)

# Calculate the cost contribution of each generator
```

```
cost_contributions = [active_power_values[i] * generator_costs[i]
    ↪ for i in 1:num_generators]
total_cost_contribution = sum(cost_contributions)

# Normalize contributions to get percentages
cost_percentages = [contribution / total_cost_contribution * 100 for
    ↪  contribution in cost_contributions]

# Output the contributions
println("\nCost␣Contributions")
for i in 1:num_generators
    println("Generator␣$i:␣Contribution␣=␣SEK␣", round(
        ↪ cost_contributions[i], digits=2), "␣(", round(
        ↪ cost_percentages[i], digits=2), "%␣of␣total␣cost)")
end
```