Chalmers University of Technology

Project in TMA947 - Nonlinear optimization

Planning of Electricity Production and Transmission

Authors:
Nyberg Cecilia
cecnyb@chalmers.se

Sjögren Frida
sjogrenf@chalmers.se

Stiebe Elin
elinsti@chalmers.se

1 november 2023

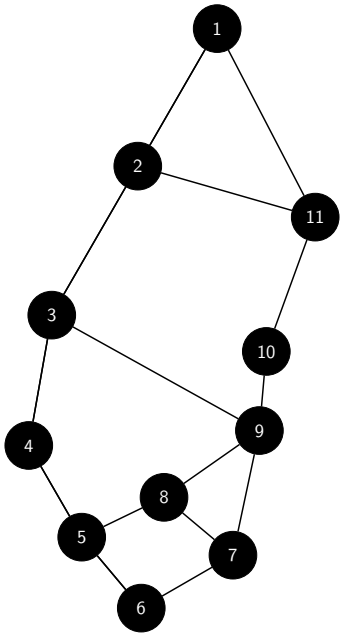
Abstract

This report aims to plan the production and transmission of electric power within a network comprising 11 nodes, each containing energy-producing generators, energy-consuming customers, or both. The primary objective is to develop a nonlinear optimization framework to minimize total energy production costs across the network while ensuring that all power demands are met. To achieve this, power must be routed through edges connecting nodes, with varying energy losses along these edges, and the additional constraint that no generator can exceed its maximum capacity. Two types of power can be generated by the generators, active and reactive. The demand is only for active power, but the reactive has to be considered when balancing the power flow in the network. The findings reveal that the lowest possible cost to satisfy the network's power demand is 186.29 SEK. Given the non-convex nature of the problem, it is not possible to guarantee that this result represents a global optimum but only a local one.

The model shows that five generators have reached their maximum capacity within this solution. By increasing the capacity of one of these generators, a lower optimal solution may be reached.

1 Mathematical Formulation

In this assignment, all problems relate back to minimizing the costs of producing energy using planable energy sources over a network. The given network can be seen in figure 1. The intuition of the 11 nodes are different cities containing generators and, or, consumers. The generators must collectively satisfy the demand from the consumers. There are two types of power; active and reactive. The produced active power can be utilized by consumers while reactive power can be seen as a by-product that can either be generated or absorbed by the generators within the system, to an amplitude of 0.3% of each generator's maximum capacity. It is essential for the energy system, yet it goes unnoticed in the form of cost and usage from consumers. With these complicating factors, the question is how to meet the consumer's needs as cost-effectively as possible.



Figur 1: Map of the network consisting of 11 nodes which all contain generators and, or consumers.

1.1 Properties of the Network

This section presents the given data for the network. Table 1 presents data related to the 9 generators and table 2 presents data related to the 7 consumers.

Generator	**Location node**	**Maximum capacity [pu]**	**Production cost [SEK/pu]**
G1	2	0.02	175
G2	2	0.15	100
G3	2	0.08	150
G4	3	0.07	150
G5	4	0.04	300
G6	5	0.17	350
G7	7	0.17	400
G8	9	0.26	300
G9	9	0.05	200

Tabell 1: Parameters related to the generators

Consumer	**Location node**	**Demand active power [pu]**
C1	1	0.10
C2	4	0.19
C3	6	0.11
C4	8	0.09
C5	9	0.21
C6	10	0.05
C7	11	0.04

Tabell 2: Parameters related to the consumers

For two connected nodes k and l in the given network, the amount of active power that flows from node k to node l is described by the p_{kl} formula and the reactive power that flows from node k to node l is described by the q_{kl} formula, both presented below:

$$p_{kl} = v_k^2 g_{k\ell} - v_k v_\ell g_{k\ell} \cos(\theta_k - \theta_\ell) - v_k v_\ell b_{k\ell} \sin(\theta_k - \theta_\ell)$$

$$q_{k\ell} = -v_k^2 b_{k\ell} + v_k v_\ell b_{k\ell} \cos(\theta_k - \theta_\ell) - v_k v_\ell g_{k\ell} \sin(\theta_k - \theta_\ell)$$

The parameters b_{kl} and q_{kl} describe the edges between the nodes of the network. The values of the parameters are presented in tables 3 and 4.

Edge(k,l) / Coeff.	(1,2)	(1,11)	(2,3)	(2,11)	(3,4)	(3,9)	(4,5)	(5,6)
b_{kl}	−20.1	−22.3	−16.8	−17.2	−11.7	−19.4	−10.8	−12.3
g_{kl}	4.12	5.67	2.41	2.78	1.98	3.23	1.59	1.71

Tabell 3: b_{kl}andg_{kl} values

Coeff. \ Edge(k,l)	(5, 8)	(6, 7)	(7, 8)	(7, 9)	(8, 9)	(9, 10)	(10, 11)
b_{kl}	−9.2	−13.9	−8.7	−11.3	−14.7	−13.5	−26.7
g_{kl}	1.26	1.11	1.32	2.01	2.41	2.14	5.06

Tabell 4: b_{kl}andg_{kl} values

1.2 Definition of Parameters and Variables

The given parameters for the problem are presented in table 5. Variables used to solve the nonlinear programming problem are defined and stated in table 6.

Parameter	**Definition**		**Unit**
C = $\{c_i\}$	energy production cost for generator i	$i \in$ [1,9]	SEK/pu
d_k	active power demand in node k	$k \in$ [1,11]	pu
$emax_i$	maximum capacity for generator i	$i \in$ [1,9]	pu
g_{kl}	g coefficient for the edge between node k and node l. 0 if there is no edge.	k,$l \in$ [1,11]	Unitless
b_{kl}	b coefficient for the edge between node k and node l. 0 if there is no edge. k,l	k,$l \in$ [1,11]	Unitless
A = $\{a_k^T\}$	a_k is a vector with 9 elements describing the existence of generators in node k. For example, generator 1,2,3 lies in node 2, and then the first, second, and third element in a_2 is 1; else it's 0's.	$k \in$ [1,11]	Unitless

Tabell 5: Parameters

Variable	**Definition**		**Unit**
E = $\{e_i\}$	amount of produced active energy at generator i	$i \in$ [1,9]	pu
R = $\{r_i\}$	amount of reactive power at generator i	$i \in$ [1,9]	pu
v_k	voltage in node k	$k \in$ [1,9]	vu
θ_k	voltage phase angle in node k	$k \in$ [1,11]	radians
$p_{k,l}$	amount of active energy flowing between node k and l	k,$l \in$ [1,11]	pu

Tabell 6: Defined variables

1.3 Nonlinear Programming Problem

The problem is to minimize the cost of production without violating the constraints of the systems. A more in-depth description of each constraint follows after the nonlinear programming system. The objective function (1), also called the cost function, is obtained by summarizing the amount of generated power at each generator multiplied by its associated cost of production.

minimize

$$\sum_{i=1}^{9} c_i e_i. \tag{1}$$

subject to

$$d_k = a_k^T E - \sum_{l=1}^{11} p_{kl}, \quad k = 1, \ldots, 11 \tag{2}$$

$$0 = a_k^T R - \sum_{l=1}^{11} q_{kl}, \quad k = 1, \ldots, 11 \tag{3}$$

$$p_{kl} = v_k^2 g_{k\ell} - v_k v_\ell g_{k\ell} \cos(\theta_k - \theta_\ell) - v_k v_\ell b_{k\ell} \sin(\theta_k - \theta_\ell) \tag{4}$$

$$q_{k\ell} = -v_k^2 b_{k\ell} + v_k v_\ell b_{k\ell} \cos(\theta_k - \theta_\ell) - v_k v_\ell g_{k\ell} \sin(\theta_k - \theta_\ell) \tag{5}$$

$$0 \leq e_i \leq \text{emax}_i \tag{6}$$

$$-0.003 \cdot \text{emax}_i \leq r_i \leq 0.003 \cdot \text{emax}_i \tag{7}$$

$$0.98 \leq v_k \leq 1.02 \tag{8}$$

$$-\pi \leq \theta_k \leq \pi \tag{9}$$

The first constraint (2) ensures that the flow of active power is balanced and that the power demand in each node is satisfied. The demand in a node is set to be equal to the generated power in each generator in that node plus the sum of the power flow from each edge going out of that node. If power is going into the node, it is still viewed as power going out but with a negative sign. The second constraint (3) handles the reactive power by making sure that the amount of reactive power that is produced or generated in all generators in a node minus the sum of the flow of reactive power in connected edges to that node is zero. This makes sure that the reactive power throughout the network is balanced. The third (4) and fourth (5) constraints are equations for the flow of active power (4) and reactive power (4) in the edge between node k and node l. These equations are used in constraint (2) and (3). Constraints (4) and (5) share the variables v and θ and therefore create a connection between the amount of active and reactive power in the network.

Variable constraint (6) ensures that the produced active power in each generator is greater than or equal to zero since it is not possible to produce negative active power in this problem. The upper bound for the produced active power at each generator is given for the problem, corresponding to its maximum capacity. For the generated reactive power at each generator it is given in the problem, that each generator can either generate or absorb reactive power to an amplitude of 0.3% of the generator's maximum capacity (7). Constraints on the variables voltage amplitude (8) and voltage angle (9) are also given for the problem.

2 Results and Analysis

By solving the nonlinear programming problem several results are given. Firstly, the minimum value of the objective function can be seen in table 7.

Minimal Cost
186.29

Tabell 7: Minimal cost in SEK to satisfy the demand in the network.

As presented in the generator data, each generator has a maximum capacity and power production cost. They therefore contribute different amounts of active and reactive power to the network to satisfy the demand of the consumers. Table 8 demonstrates the amount of active power that each generator provides to satisfy the demand.

Generator	**Active Power Produced**
1	0.0049
2	0.15
3	0.080
4	0.070
5	0.040
6	0.14
7	0.0031
8	0.25
9	0.050

Tabell 8: Active power produced by each generator in pu.

As for the reactive power, each generator produces or absorbs the amount shown in table 9. A negative sign indicates that the power is absorbed, while a positive sign indicates produced power. In this problem, all generators produced reactive power and hence the signs are positive.

Generator	**Reactive Power Produced**
1	6.0e-5
2	0.00045
3	0.00024
4	0.00021
5	0.00012
6	0.00051
7	0.00051
8	0.00078
9	0.00015

Tabell 9: Reactive power produced by each generator.

The active and reactive power that flows in the edges depends on the voltage amplitudes and phase angles in the nodes. The specific relation between these two variables and power flow can be seen in constraints (4) and (5) of the nonlinear programming system. The result for voltage amplitudes and phase angles can be seen in table 10

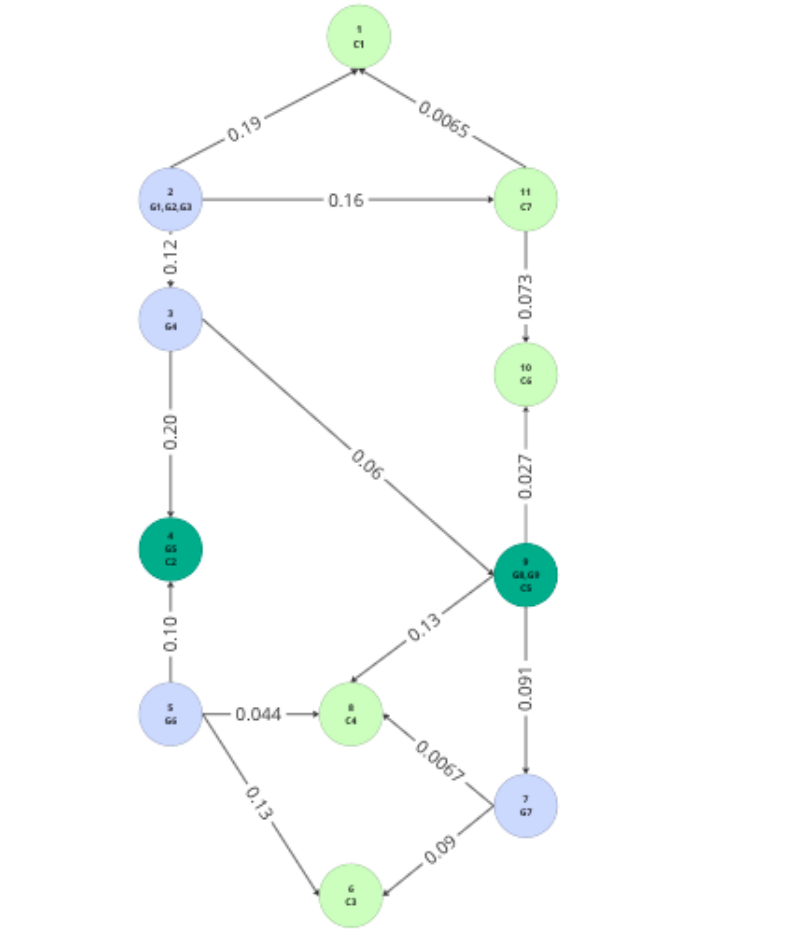
Node	**Voltage Amplitude**	**Phase Angle**
1	1.019	0.0018
2	1.020	0.0063
3	1.019	0.0030
4	1.018	-0.0048
5	1.019	-0.00030
6	1.018	-0.0053
7	1.019	-0.0022
8	1.019	-0.0026
9	1.019	0.0016
10	1.019	0.00065
11	1.019	0.0019

Tabell 10: Each node's voltage amplitude and phase angle.

One can also observe how much power flows in each edge between the connected nodes. Table 11 demonstrates the specifics of the active and reactive power in the edges. Important to observe is that the power going from node k to l is not always the same as the power going from node l to k. An example of this can be seen where k = 7 and l = 9 where there is a 0.01 difference between the two directions. This is due to losses in the network, and the amplitude of the losses is dependent on the properties of the edges. Figure 2 illustrates the direction and amplitude of active power flow between the nodes.

Edge (k,l)	**Active Power k to l**	**Reactive Power k to l**	**Active Power l to k**	**Reactive Power l to k**
(1, 2)	-0.097	0.0015	0.097	-0.0011
(1, 11)	-0.0032	-0.0015	0.0032	0.0015
(2, 3)	0.058	0.00072	-0.058	-0.00054
(2, 11)	0.080	0.0011	-0.080	-0.00078
(3, 4)	0.098	0.00027	-0.098	0.00050
(3, 9)	0.030	0.00047	-0.030	-0.00043
(4, 5)	-0.052	-0.00038	0.052	0.00062
(5, 6)	0.065	-0.00048	-0.065	0.00081
(5, 8)	0.022	0.00038	-0.022	-0.00033
(6, 7)	-0.045	-0.00081	0.045	0.00095
(7, 8)	0.0034	-0.00040	-0.0034	0.00040
(7, 9)	-0.045	-3.6e-5	0.046	0.00021
(8, 9)	-0.065	-7.4e-5	0.065	0.00034
(9, 10)	0.013	0.00081	-0.013	-0.00080
(10, 11)	-0.037	0.00080	0.037	-0.00075

Tabell 11: Each edge's reactive and active power.



Figur 2: Direction and amplitude of active power flow

It is important to consider whether the achieved optimal solution is a global minimum or only a local. The fundamental theorem of global optimality gives that if a problem is convex it guarantees that the found min is global. However, this problem is not convex. This becomes clear by looking at constraints (4) and (5) in the nonlinear programming system. For example, cos(x) and sin(x) are non-convex functions. Further constraint (5) also contains the term -x$\hat{2}$ which makes the constraint nonconvex. The solution could be global, yet this is nothing that can be guaranteed. Therefore it is only possible to say that the solution is a local optimum.

To improve the cost function for the power distribution in the network, a possible modification is to increase one of the generator's maximum capacity. It is therefore relevant to reflect on which increased maximum capacity would decrease the objective function the most. First, only the generators that have reached their maximum capacity should be considered as potential candidates. Increasing the maximum capacity of a generator that is currently not utilizing its full capacity would not yield any effect on the objective function. Some other criteria should be evaluated as well. The objective is to minimize the price, the price of power produced is therefore an important aspect to consider. Another aspect to take into consideration is the generator's position in the network compared to the position of customers with a high power demand. The position is important to consider as power traveling further distances experiences greater losses in the network. This can be seen in the results as generator 1 did not maximize its capacity even though it is the second cheapest generator.

To figure out which generator's capacity to increase to reduce the cost the most, an easy way is

to look at the dual variables since they correspond to the reduced cost. Then the dual variable with the lowest reduced cost should be chosen. The result of the reduced costs is presented below in table 9. It is clear from table 12 that generator 5 corresponds to the lowest reduced cost, therefore this is the generator whose capacity should be increased.

Generator	**Reduced Cost**
1	0
2	-75.0
3	-25.0
4	-111.6654
5	-170.6663
6	-2.5614e-7
7	0
8	-2.2595e-6
9	-100.0

Tabell 12: Reduced cost corresponding to each of the generators.

3 Conclusion

When optimizing energy production over a network, many factors have to be taken into account. Since power cannot be stored, the demand has to be met but there can be no overflow of power. The result showed that the problem is more complex than it may first look. This was because it is not just the cheapest generators that are maximized. Instead, their position in the network is also of importance. Another contributing factor to the complexity of the model is the reactive power which has to be accounted for in the network even though it does not directly influence the cost.

The model shows that for the given network, the generators can supply all consumers with their demand for 186.30 SEK. However, it is important to emphasize that the found optimal value is not guaranteed to be global and hence, it is not possible to ensure that there does not exist a lower cost.

In order to achieve a lower cost for the power production of the network, the maximum capacity for generator 5 should be increased as it has the lowest reduced cost when analysing the dual variables.

A Julia Code

The code was divided into two sections. One for the data and another one for the optimization model. The first file is called data and the code can be seen in the appendix section A.1. The optimization implementation can be seen in the second section of the appendix, A.2.

A.1 Code in Data file

```
emax = [0.02, 0.15, 0.08, 0.07, 0.04, 0.17, 0.17, 0.26, 0.05] # Maximum
    capacity
d = [0.1, 0, 0, 0.19, 0, 0.11, 0, 0.09, 0.21, 0.05, 0.04] # Demand
c = [175, 100, 150, 150, 300, 350, 400, 300, 200] # Costs

e_vars = 9 # Produced active power in generator i, 1 = 1,...,9
r_vars = 9 # Produced reactive power in generator i, 1 = 1,...,9
v_vars = 11 # Voltage
t_vars = 11 # Angle

e_lb = [0, 0, 0, 0, 0, 0, 0, 0, 0]
r_lb = [-0.003 * emax[1], -0.003 * emax[2], -0.003 * emax[3], -0.003 * emax[4],
     -0.003 * emax[5], -0.003 * emax[6], -0.003 * emax[7], -0.003 * emax[8],
    -0.003 * emax[9]]
v_lb = [0.98, 0.98, 0.98, 0.98, 0.98, 0.98, 0.98, 0.98, 0.98, 0.98, 0.98]
t_lb = [-MathConstants.pi, -MathConstants.pi, -MathConstants.pi, -MathConstants
    .pi, -MathConstants.pi, -MathConstants.pi, -MathConstants.pi, -MathConstants
    .pi, -MathConstants.pi, -MathConstants.pi, -MathConstants.pi]

e_ub = [0.02, 0.15, 0.08, 0.07, 0.04, 0.17, 0.17, 0.26, 0.05]
r_ub = [0.003 * emax[1], 0.003 * emax[2], 0.003 * emax[3], 0.003 * emax[4],
    0.003 * emax[5], 0.003 * emax[6], 0.003 * emax[7], 0.003 * emax[8], 0.003 *
    emax[9]]
v_ub = [1.02, 1.02, 1.02, 1.02, 1.02, 1.02, 1.02, 1.02, 1.02, 1.02, 1.02]
t_ub = [MathConstants.pi, MathConstants.pi, MathConstants.pi, MathConstants.pi,
     MathConstants.pi, MathConstants.pi, MathConstants.pi, MathConstants.pi,
    MathConstants.pi, MathConstants.pi, MathConstants.pi]


matrixGenInNode = [[0, 0, 0, 0, 0, 0, 0, 0, 0],
    [1, 1, 1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 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, 0, 0, 0, 0, 0, 0, 1, 1],
    [0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0]]

matrixB = [[0, -20.1, 0, 0, 0, 0, 0, 0, 0, -22.3],
    [-20.1, 0, -16.8, 0, 0, 0, 0, 0, 0, -17.2],
    [0, -16.8, 0, -11.7, 0, 0, 0, 0, -19.4, 0, 0],
    [0, 0, -11.7, 0, -10.8, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, -10.8, 0, -12.3, 0, -9.2, 0, 0, 0],
    [0, 0, 0, 0, -12.3, 0, -13.9, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, -13.9, 0, -8.7, -11.3, 0, 0],
    [0, 0, 0, 0, -9.2, 0, -8.7, 0, -14.7, 0, 0],
    [0, 0, -19.4, 0, 0, 0, -11.3, -14.7, 0, -13.5, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, -13.5, 0, -26.7],
    [-22.3, -17.2, 0, 0, 0, 0, 0, 0, 0, -26.7, 0]]

matrixG = [[0, 4.12, 0, 0, 0, 0, 0, 0, 0, 0, 5.67],
    [4.12, 0, 2.41, 0, 0, 0, 0, 0, 0, 0, 2.78],
```

```
    [0, 2.41, 0, 1.98, 0, 0, 0, 0, 3.23, 0, 0],
    [0, 0, 1.98, 0, 1.59, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1.59, 0, 1.71, 0, 1.26, 0, 0, 0],
    [0, 0, 0, 0, 1.71, 0, 1.11, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1.11, 0, 1.32, 2.01, 0, 0],
    [0, 0, 0, 0, 1.26, 0, 1.32, 0, 2.41, 0, 0],
    [0, 0, 3.23, 0, 0, 0, 2.01, 2.41, 0, 2.14, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 2.14, 0, 5.06],
    [5.67, 2.78, 0, 0, 0, 0, 0, 0, 0, 5.06, 0]]
```

A.2 Code in Main file

```
using JuMP
import Ipopt

# Import data from the data file
include("data.jl")

# Create the model object
the_model = Model(Ipopt.Optimizer)

# Create (one set of) variables, and their lower and upper bounds
@variable(the_model, e_lb[i] <= e[i=1:e_vars] <= e_ub[i]) # Power in generator
@variable(the_model, r_lb[i] <= r[i=1:r_vars] <= r_ub[i]) # Reactive power
@variable(the_model, v_lb[i] <= v[i=1:v_vars] <= v_ub[i]) # Voltage
@variable(the_model, t_lb[i] <= t[i=1:t_vars] <= t_ub[i]) # Angle

# Create the nonlinear objective which we want to minimize
@NLobjective(the_model, Min, sum((e[i] * c[i]) for i in 1:e_vars))

# Constraints p
for node_k in 1:11
    @NLconstraint(the_model, (sum(matrixGenInNode[node_k][i] * e[i] for i in 1:
    e_vars) - (sum(v[node_k]^2 * matrixG[node_k][node_l] -
                    v[node_k] * v[node_l] * matrixG[node_k][node_l] * cos(t[
    node_k] - t[node_l]) -
                    v[node_k] * v[node_l] * matrixB[node_k][node_l] * sin(t[
    node_k] - t[node_l])
                    for node_l in 1:11)) == d[node_k]))
end


# Constraints q
for node_k in 1:11
    @NLconstraint(the_model, ((sum(matrixGenInNode[node_k][i] * r[i] for i in
    1:r_vars) - (sum(-v[node_k]^2 * matrixB[node_k][node_l] +
                     v[node_k] * v[node_l] * matrixB[node_k][node_l] * cos(t[
    node_k] - t[node_l]) -
                     v[node_k] * v[node_l] * matrixG[node_k][node_l] * sin(t[
    node_k] - t[node_l])
                     for node_l in 1:11))) == 0))
end

# Solve the optimization problem
optimize!(the_model)
println(the_model)

```

```
# Printing some of the results for further analysis
println("") # Printing white line after solver output, before printing
println("Termination statue: ", JuMP.termination_status(the_model))
println("Optimal(?) objective function value: ", JuMP.objective_value(the_model))
println("Optimal generated active power: ", JuMP.value.(e))
println("Optimal generated/absorbed reactive power: ", JuMP.value.(r))
println("Optimal voltage amplitudes: ", JuMP.value.(v))
println("Optimal voltage phase angles: ", JuMP.value.(t))
println("Optimal(?) point: ", JuMP.value.(e))
println("Dual variables/Lagrange multipliers corresponding to some of the constraints: ")
#println(JuMP.dual.(SOS_constr))
println(JuMP.dual.(JuMP.UpperBoundRef.(e)))

# Print active power flow
for node_k in 1:11
    for node_l in 1:11
        if matrixG[node_k][node_l] != 0
            println("Optimal active power flow in edge ", node_k, " ", node_l, " is: ", JuMP.value(v[node_k]^2 * matrixG[node_k][node_l] -
                        v[node_k] * v[node_l] * matrixG[node_k][node_l] * cos(t[node_k] - t[node_l]) -
                        v[node_k] * v[node_l] * matrixB[node_k][node_l] * sin(t[node_k] - t[node_l])
            ))
        end
    end
end

# Print active power flow netto
for node_k in 1:11
    for node_l in 1:11
        if matrixG[node_k][node_l] != 0
            println("Netto flow ", node_k, " ", node_l, " is: ", JuMP.value(v[node_k]^2 * matrixG[node_k][node_l] -
                                                                                v[node_k] * v[node_l] * matrixG[node_k][node_l] * cos(t[node_k] - t[node_l]) -
                                                                                v[node_k] * v[node_l] * matrixB[node_k][node_l] * sin(t[node_k] - t[node_l]) - JuMP.value(v[node_l]^2 * matrixG[node_l][node_k] -
                v[node_l] * v[node_k] * matrixG[node_l][node_k] * cos(t[node_l] - t[node_k]) -
                v[node_l] * v[node_k] * matrixB[node_l][node_k] * sin(t[node_l] - t[node_k]))
            ))
        end
    end
end


# Print reactive power flow
for node_k in 1:11
    for node_l in 1:11
        if matrixG[node_k][node_l] != 0
            println("Optimal reactive power flow in edge ", node_k, " ", node_l, " is: ", JuMP.value(-v[node_k]^2 * matrixB[node_k][node_l] +
```

```
85
                            v[node_k] * v[node_l] * matrixB[node_k][node_l] * cos(
    t[node_k] - t[node_l]) -
86
                            v[node_k] * v[node_l] * matrixG[node_k][node_l] * sin(
    t[node_k] - t[node_l])))
87
88          end
89      end
90  end
```