Planning of electricity production and transmission

Nonlinear optimisation (TMA947)

M.Sc. Complex Adaptive Systems, Chalmers University of Technology

12 October 2023

Erik Norlin
norliner@student.chalmers.se

Mattias Wiklund
matwi@student.chalmers.se

Abstract

In this project, electricity production and transmission were analysed. A system with nine generators and seven customers, distributed among eleven nodes, was simulated. The model included both active and reactive power flow, with flow conservation constraints in each node. The objective was to minimise the total cost of active power production, while fulfilling the consumers' demands for active power and simultaneously not violating any other constraints. The simulation, performed with the Ipopt optimiser in Julia, resulted in a minimised total cost of 186.29 kr. Analysing the optimal variable values, five generators produced active power at its maximum capacity and every consumer received its demanded active power. There could be weaknesses with the model which could be analysed further, for example indicated by counter-intuitive behaviour for the flow of the reactive power. Overall, however, the model seemed to result in reasonable values for the different variables and physical quantities.

1 Mathematical formulation

The objective of this problem is to minimise the cost for active power production of all generators such that all consumers' demands for active power are met. Very intuitively, the objective function is

$$f(A_1,\ldots,A_n) = \sum_{i=1}^{n} h_i A_i,$$

where h_i is the cost for producing active power A_i for generator G_i, where $n = 9$ is the number of generators.

To construct a formulation that makes sense and solvable, flow conservation is assumed to hold in each node for both active and reactive power respectively. Generally, in a node k, the flow conservation is expressed as *produced power in k = power going to adjacent nodes + absorbed power in k.* Produced power can be seen as flow into the node, whereas the outgoing power and absorbed power can be seen as flow out from the node. Thus, the flow is conserved with this equation.

In the case of active power, in each node k, we have that $A_k = \sum_l p_{kl} + C_k$, where A_k is the active power generated at the node, p_{kl} is the outgoing active power to adjacent nodes l, and C_k is the consumed active power in k. The following constraints define conservation of flow of active power in each node.

$$0 = p_{1,2} + p_{1,11} + C_1$$

$$\begin{gathered}
A_1 + A_2 + A_3 = p_{2,1} + p_{2,11} + p_{2,3} \\
A_4 = p_{3,2} + p_{3,4} + p_{3,9} \\
A_5 = p_{4,3} + p_{4,5} + C_2 \\
A_6 = p_{5,4} + p_{5,6} + p_{5,8} \\
0 = p_{6,5} + p_{6,7} + C_3 \\
A_7 = p_{7,6} + p_{7,8} + p_{7,9} \\
0 = p_{8,5} + p_{8,7} + p_{8,9} + C_4 \\
A_8 + A_9 = p_{9,3} + p_{9,7} + p_{9,8} + p_{9,10} + C_5 \\
0 = p_{10,9} + p_{10,11} + C_6 \\
0 = p_{11,1} + p_{11,2} + p_{11,10} + C_7
\end{gathered}$$

For the reactive power, in each node k we have that $R_k = \sum_l q_{kl}$, where R_k is the reactive power generated at the node, and q_{kl} is the outgoing reactive power to adjacent nodes l. These two added has to equal to zero to conserve the flow. The following constraints define conservation of flow of reactive power in each node

$$0 = q_{1,2} + q_{1,11}$$

$$\begin{gathered}
R_1 + R_2 + R_3 = q_{2,1} + q_{2,11} + q_{2,3} \\
R_4 = q_{3,2} + q_{3,4} + q_{3,9} \\
R_5 = q_{4,3} + q_{4,5} \\
R_6 = q_{5,4} + q_{5,6} + q_{5,8} \\
0 = q_{6,5} + q_{6,7}
\end{gathered}$$

$$R_7 = q_{7,6} + q_{7,8} + q_{7,9}$$
$$0 = q_{8,5} + q_{8,7} + q_{8,9}$$
$$R_8 + R_9 = q_{9,3} + q_{9,7} + q_{9,8} + q_{9,10}$$
$$0 = q_{10,9} + q_{10,11}$$
$$0 = q_{11,1} + q_{11,2} + q_{11,10}$$

The active power p_{kl} and the reactive power q_{kl} from node k to node l are defined as

$$p_{kl} = v_k^2 g_{kl} - v_k v_l g_{kl} \cos(\theta_k - \theta_l) - v_k v_l b_{kl} \sin(\theta_k - \theta_l),$$

$$q_{kl} = -v_k^2 b_{kl} + v_k v_l b_{kl} \cos(\theta_k - \theta_l) - v_k v_l g_{kl} \sin(\theta_k - \theta_l),$$

where v and θ are the voltage amplitude and phase angle, respectively. The interval for the different variables are

$$\begin{cases} 0 \leq A_i \leq M_i \\ |R_i| \leq 0.003M_i \\ C_j \geq D_j \\ -\pi \leq \theta_k \leq \pi \\ 0.98 \leq v_k \leq 1.02 \end{cases}$$

where M_i is the max capacity of G_i, R_i the reactive power produced or absorbed by G_i, for $i = 1,...,n$. For consumer j, C_j is the active power consumed, D_j is the active power demand, for $j = 1,...,m$, where $m = 7$ is the number of consumers. At node k, θ_k is the phase angle and v_k is the voltage amplitude.

This optimisation problem was implemented in Julia using Ipopt optimiser with a total of 289 variables; 7 variables with only lower bounds (consumers). 40 variables with lower and upper bounds; generated active and reactive power from the generators (9+9), voltage amplitudes and phase angles (11+11). Finally, all elements in two matrices containing p_{kl} and q_{kl} respectively (121+121). 82 equality constraints were used; 30 definitions for p_{kl}, 30 definitions for q_{kl}, 11 flow constraints for p_{kl}, and 11 flow constraints for q_{kl}.

2 Results and analysis

From the simulation, which ran for 23 iterations, a result was obtained. The total cost of power production for all nine generators was 186.29 kr.

The produced active power and produced/absorbed reactive power for each generator are presented in Table 1. It can be noted that G_2, G_3, G_4 G_5 and G_9 produce active power at their maximum capacity. Furthermore, all generators produce more reactive power than they absorb, which is indicated by the positive sign for R in the table.

Table 1: Active power produced and reactive power produced/absorbed for each generator. A minus sign for the reactive power R would indicate a net absorption.

Generator	A [pu]	R [pu]	M [pu]	h [SEK/pu]
1	$4.9 \cdot 10^{-3}$	$6.0 \cdot 10^{-5}$	0.02	175
2	0.15	$4.5 \cdot 10^{-4}$	0.15	100
3	0.08	$2.4 \cdot 10^{-4}$	0.08	150
4	0.07	$2.1 \cdot 10^{-4}$	0.07	150
5	0.04	$1.2 \cdot 10^{-4}$	0.04	300
6	0.14	$5.1 \cdot 10^{-4}$	0.17	350
7	$3.1 \cdot 10^{-3}$	$5.1 \cdot 10^{-4}$	0.17	400
8	0.25	$7.8 \cdot 10^{-4}$	0.26	300
9	0.05	$1.5 \cdot 10^{-4}$	0.05	200

The voltage amplitudes and phase angles are presented in Table 2 below. It can be noted that the amplitudes are close to their maximal value of 1.02 V. The phase angles, on the other hand, are close to zero.

Table 2: Voltage amplitudes and voltage phase angles in each node.

Node	v [vu]	θ [rad]
1	1.019	$1.8 \cdot 10^{-3}$
2	1.020	$6.3 \cdot 10^{-3}$
3	1.019	$3.0 \cdot 10^{-3}$
4	1.018	$-4.8 \cdot 10^{-3}$
5	1.019	$-3.1 \cdot 10^{-4}$
6	1.018	$-5.3 \cdot 10^{-3}$
7	1.019	$-2.2 \cdot 10^{-3}$
8	1.019	$-2.6 \cdot 10^{-3}$
9	1.019	$1.6 \cdot 10^{-3}$
10	1.019	$6.4 \cdot 10^{-4}$
11	1.019	$1.9 \cdot 10^{-3}$

The active and reactive power flowing along the edges (p_{kl} and q_{kl}, respectively) are presented in Table 3. Note that every row contains the flow and the opposite flow, $k \to l$ and $l \to k$. A minus sign indicates a flow in the opposite direction. As can be seen, the losses between in network are relatively small, in general in the order of around 10^{-4} pu.

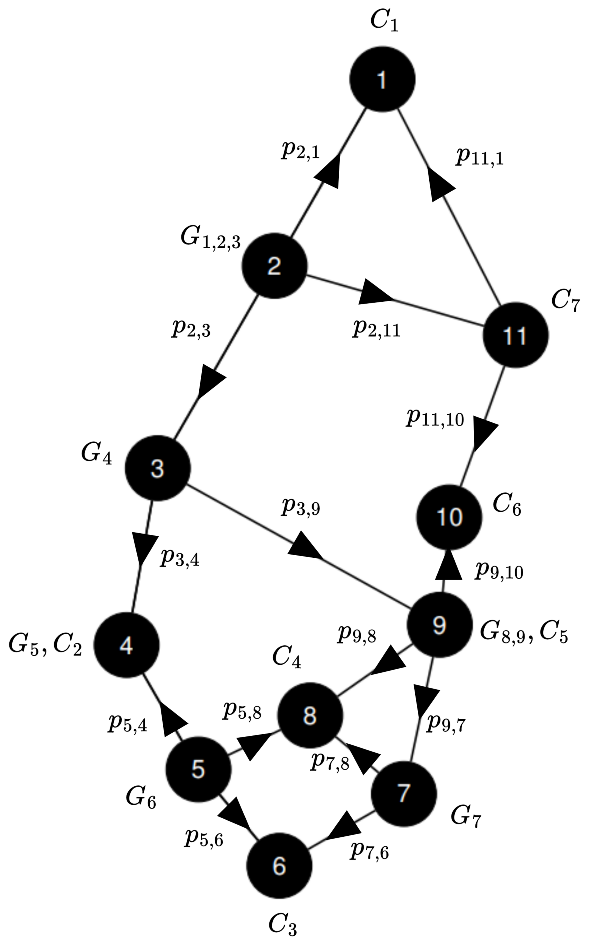
The flow directions are visualised in Figure 1. It can be noted that for the edge $(3, 4)$, both $q_{3,4} > 0$ and $q_{4,3} > 0$, indicating that the reactive power are flowing in both directions. This seems counter-intuitive at first, and could be a weakness with the implemented model. One reason could be that a generator at node 3 or 4 (G_4 or G_5) actually absorbs a net of reactive power, rather than generates (which Table 2 indicates). For example, the flow constraint equation

$$q_{4,3} + q_{4,5} = R_5,$$

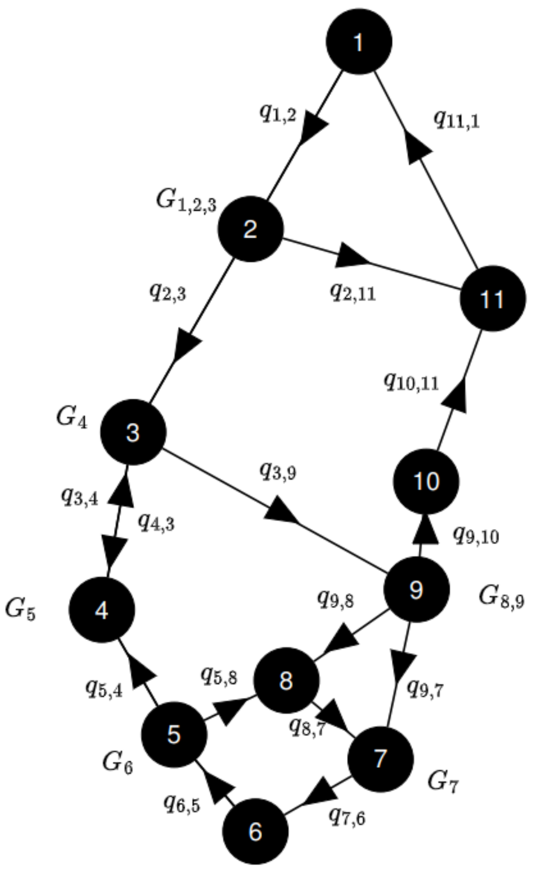
would still hold if the same amount of reactive power is subtracted from each side, making both $q_{4,3} < 0$ and $R_5 < 0$. This could potentially be solved by including opposite-sign constraints for the flows. I.e., if $q_{3,4} > 0$, then $q_{4,3} < 0$. However, since the objective function does not depend on the reactive power, this should not affect the total cost.

Table 3: Active and reactive power flowing along the edges of the network.

Edge	p_{kl} [pu]	p_{lk} [pu]	q_{kl} [pu]	q_{lk} [pu]
(1, 2)	$-9.68 \cdot 10^{-2}$	$9.69 \cdot 10^{-2}$	$1.53 \cdot 10^{-3}$	$-1.10 \cdot 10^{-3}$
(1, 11)	$-3.23 \cdot 10^{-3}$	$3.23 \cdot 10^{-3}$	$-1.53 \cdot 10^{-3}$	$1.53 \cdot 10^{-3}$
(2, 3)	$5.81 \cdot 10^{-2}$	$-5.81 \cdot 10^{-2}$	$7.25 \cdot 10^{-4}$	$-5.35 \cdot 10^{-4}$
(2, 11)	$7.99 \cdot 10^{-2}$	$-7.98 \cdot 10^{-2}$	$1.13 \cdot 10^{-3}$	$-7.80 \cdot 10^{-4}$
(3, 4)	$9.81 \cdot 10^{-2}$	$-9.80 \cdot 10^{-2}$	$2.70 \cdot 10^{-4}$	$5.00 \cdot 10^{-4}$
(3, 9)	$3.00 \cdot 10^{-2}$	$-3.00 \cdot 10^{-2}$	$4.75 \cdot 10^{-4}$	$-4.31 \cdot 10^{-4}$
(4, 5)	$-5.20 \cdot 10^{-2}$	$5.20 \cdot 10^{-2}$	$-3.80 \cdot 10^{-4}$	$6.16 \cdot 10^{-4}$
(5, 6)	$6.49 \cdot 10^{-2}$	$-6.48 \cdot 10^{-2}$	$-4.85 \cdot 10^{-4}$	$8.08 \cdot 10^{-4}$
(5, 8)	$2.20 \cdot 10^{-2}$	$-2.20 \cdot 10^{-2}$	$3.79 \cdot 10^{-4}$	$-3.29 \cdot 10^{-4}$
(6, 7)	$-4.52 \cdot 10^{-2}$	$4.52 \cdot 10^{-2}$	$-8.08 \cdot 10^{-4}$	$9.49 \cdot 10^{-4}$
(7, 8)	$3.36 \cdot 10^{-3}$	$-3.36 \cdot 10^{-3}$	$-4.02 \cdot 10^{-4}$	$4.03 \cdot 10^{-4}$
(7, 9)	$-4.55 \cdot 10^{-2}$	$4.55 \cdot 10^{-2}$	$-3.69 \cdot 10^{-5}$	$2.08 \cdot 10^{-4}$
(8, 9)	$-6.47 \cdot 10^{-2}$	$6.47 \cdot 10^{-2}$	$-7.41 \cdot 10^{-5}$	$3.41 \cdot 10^{-4}$
(9, 10)	$1.34 \cdot 10^{-2}$	$-1.34 \cdot 10^{-2}$	$8.12 \cdot 10^{-4}$	$-8.00 \cdot 10^{-4}$
(10, 11)	$-3.66 \cdot 10^{-2}$	$3.66 \cdot 10^{-2}$	$8.00 \cdot 10^{-4}$	$-7.53 \cdot 10^{-4}$



(a) The resulting flow of active power over the edges of the graph after optimisation.



(b) The resulting flow of reactive power over the edges of the graph after optimisation.

Figure 1: G_i is generator i for $i = 1,...,n$, C_j is consumer j for $j = 1,...,m$, p_{kl} is the flow of active power from node k to l, and q_{kl} is the flow of reactive power from node k to l. The arrows show the direction of the resulting flow for every edge after optimising the model. Edge (3,4) in the flow of reactive power shows arrows pointing in both directions, indicating that the reactive power flows in both directions, which is contradictory.

According to the Ipopt optimiser, the solution is at least locally optimal, and for it to be a globally optimal the problem must be convex. For a problem to be convex the objective function, all inequality constrains must be convex, and all equality constraints must be affine. p_{kl} and q_{kl} are non-linear equality constraints which means that they are not affine, so the problem is not convex. This implies that local optimal solution could be a global optimal, but this is not guaranteed.

Now assume that the maximum capacity of one of the generators is increased with 0.01 pu, without affecting its ability to produce or absorb reactive power. Which one would be most beneficial to choose? We notice that generators 2, 3, 4, 5 and 9 produce at their maximum capacity. It it therefore reasonable to assume that it would be one of them. It is, given the solution, clearly not optimal to let generators 1, 6, 7 and 8 run at their maximum capacity. Out of the five candidates, it is not obvious which one to choose. Generator 2 has the lowest production cost, but the highest maximum capacity. On the other hand, generator 5 has the highest production cost, but the lowest maximum capacity.

To analyse this in greater depth, the Lagrangian multipliers corresponding to the maximum capacity constraints can be considered. The multipliers μ_i obtained from the Ipopt optimiser are presented in Table 4 below.

Table 4: Lagrangian multipliers, rounded to the closest integer, corresponding to the maximum capacity constraint for each generator.

Generators	μ [SEK/pu]
1	0
2	75
3	25
4	112
5	171
6	0
7	0
8	0
9	100

As expected, the multipliers for generators 1, 6, 7 and 8 are zero. According to sensitivity analysis, we have that $\nabla p(\mathbf{0}^m) = -\mu^*$, where $p(\boldsymbol{u})$ is the perturbation function and μ^* is the Lagrangian dual optimal solution. In this case, $m = 9$ and $\boldsymbol{u}$ consists of eight zero-components, with one component being 0.01. Note that this holds if the perturbed problem has a solution in the neighbourhood of $\boldsymbol{u} = \mathbf{0}^m$. Since the only non-zero component is 0.01, this perturbation can be assumed to be sufficiently small.

From the table, we see that G_5 has the largest multiplier, indicating that the perturbed objective value is decreased the most by increasing the maximum capacity for that generator. In conclusion, generator 5 should be the most beneficial to choose.

3 Summary and conclusions

The minimised cost to provide active power to all consumers such that all their demands is satisfied is 186.29 kr. Active and reactive power generated corresponding to the optimal solution are in congruent with their capacities, all voltage amplitudes and phase angles are in line with what they are allowed to be tuned to. A small percentage of active power transmitting through the lines is lost due to energy losses in the network which can be seen in Table 3 ($p_{kl} \neq -p_{lk}$). The flow of the reactive power behaves similarly as the active power except between two nodes. Between node 3 and 4, the model says that the reactive power is transmitted in both directions. This can be a weakness in the model, which could potentially be solved by including opposite-sign constraints for q_{kl} and q_{lk}. However, this does not affect the objective function.

The optimal solution obtained is locally optimal and could be globally optimal, but this is not guaranteed. This is because convexity cannot be claimed for the problem; the equality constraints that define the active and reactive power are non-linear.

If by any means a single generator would be given an 0.01 increase in their capacity, G_5 would be the most optimal generator to choose. This is motivated by the fact that the largest Lagrangian dual optimal solution, for the maximum capacity constraints, corresponds to the fifth generator. This is when the perturbation function decreases the most.

A Code

```julia
using JuMP
import Ipopt

function is_connection(target, matrix)
    for row in eachrow(matrix)
        if row == target
            return true
        end
    end
    return false
end

n_generators = 9
n_customers = 7
n_nodes = 11

b_matrix = zeros(Float64, n_nodes, n_nodes);
g_matrix = zeros(Float64, n_nodes, n_nodes);
b = [-20.1, -22.3, -16.8, -17.2, -11.7, -19.4, -10.8, -12.3, -9.2,
    -13.9, -8.7, -11.3, -14.7, -13.5, -26.7]
g = [4.12, 5.67, 2.41, 2.78, 1.98, 3.23, 1.59, 1.71, 1.26, 1.11, 1.32,
    2.01, 2.41, 2.14, 5.06]

lb_generators = [0,0,0,0,0,0,0,0,0]
ub_generators = [0.02, 0.15, 0.08, 0.07, 0.04, 0.17, 0.17, 0.26, 0.05] #
     Maximum capacity
lb_customers = [0.10, 0.19, 0.11, 0.09, 0.21, 0.05, 0.04] # Customer
    demand
b_reactive = 0.003*ub_generators
cost = [175, 100, 150, 150, 300, 350, 400, 300, 200]
connections = [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]
connections2 = [connections[:,2] connections[:,1]]
connections_all = [connections; connections2]

k = 0;
for i in 1:n_nodes
    for j in i+1:n_nodes
        if is_connection([i,j],connections_all) == true
            k += 1
            b_matrix[i,j] = b[k]
            b_matrix[j,i] = b[k]
            g_matrix[i,j] = g[k]
            g_matrix[j,i] = g[k]
        end
    end
end

the_model = Model(Ipopt.Optimizer)

@variable(the_model, lb_generators[i] <= A[i = 1:n_generators] <=
    ub_generators[i])
@variable(the_model, C[i = 1:n_customers] >= lb_customers[i])
@variable(the_model, -b_reactive[i] <= R[i = 1:n_generators] <=
    b_reactive[i])
@variable(the_model, p[i = 1:n_nodes, j = 1:n_nodes])
@variable(the_model, q[i = 1:n_nodes, j = 1:n_nodes])
@variable(the_model, 0.98 <= v[i = 1:n_nodes] <= 1.02)
@variable(the_model, -pi <= theta[i = 1:n_nodes] <= pi)
```

```

@objective(the_model, Min, sum(A[i]*cost[i] for i in 1:n_generators))

for i in 1:n_nodes
    for j in 1:n_nodes
        if is_connection([i,j],connections_all) == true
            @NLconstraint(the_model,p[i,j]==v[i]^2*g_matrix[i,j]-v[i]*v[j]*g_matrix[i,j]*cos(theta[i]-theta[j])-v[i]*v[j]*b_matrix[i,j]*sin(theta[i]-theta[j]))
            @NLconstraint(the_model,q[i,j]==-v[i]^2*b_matrix[i,j]+v[i]*v[j]*b_matrix[i,j]*cos(theta[i]-theta[j])-v[i]*v[j]*g_matrix[i,j]*sin(theta[i]-theta[j]))
        end
    end
end

@constraint(the_model, pflow1, p[1,2] + p[1,11] + C[1] == 0)
@constraint(the_model, pflow2, p[2,1] + p[2,3] + p[2,11] == A[1] + A[2] + A[3])
@constraint(the_model, pflow3, p[3,2] + p[3,4] + p[3,9] == A[4])
@constraint(the_model, pflow4, p[4,3] + p[4,5] + C[2] == A[5])
@constraint(the_model, pflow5, p[5,4] + p[5,6] + p[5,8] == A[6])
@constraint(the_model, pflow6, p[6,5] + p[6,7] + C[3] == 0)
@constraint(the_model, pflow7, p[7,6] + p[7,8] + p[7,9] == A[7])
@constraint(the_model, pflow8, p[8,5] + p[8,7] + p[8,9] + C[4] == 0)
@constraint(the_model, pflow9, p[9,3] + p[9,7] + p[9,8] + p[9,10] + C[5] == A[8] + A[9])
@constraint(the_model, pflow10, p[10,9] + p[10,11] + C[6] == 0)
@constraint(the_model, pflow11, p[11,1] + p[11,2] + p[11,10] + C[7] == 0)

@constraint(the_model, qflow1, q[1,2] + q[1,11] == 0)
@constraint(the_model, qflow2, q[2,1] + q[2,3] + q[2,11] == R[1] + R[2] + R[3])
@constraint(the_model, qflow3, q[3,2] + q[3,4] + q[3,9] == R[4])
@constraint(the_model, qflow4, q[4,3] + q[4,5] == R[5])
@constraint(the_model, qflow5, q[5,4] + q[5,6] + q[5,8] == R[6])
@constraint(the_model, qflow6, q[6,5] + q[6,7] == 0)
@constraint(the_model, qflow7, q[7,6] + q[7,8] + q[7,9] == R[7])
@constraint(the_model, qflow8, q[8,5] + q[8,7] + q[8,9] == 0)
@constraint(the_model, qflow9, q[9,3] + q[9,7] + q[9,8] + q[9,10] == R[8] + R[9])
@constraint(the_model, qflow10, q[10,9] + q[10,11] == 0)
@constraint(the_model, qflow11, q[11,1] + q[11,2] + q[11,10] == 0)

println(the_model)

optimize!(the_model)

println("")
println("Termination statue: ", JuMP.termination_status(the_model))
println("Optimal(?) objective function value: ", JuMP.objective_value(the_model))
println("Optimal(?) point: ", JuMP.value.(A))
println("Optimal(?) point: ", JuMP.value.(C))
println("Optimal(?) point: ", JuMP.value.(R))
println("Optimal(?) point: ", JuMP.value.(p))
println("Optimal(?) point: ", JuMP.value.(q))
println("Optimal(?) point: ", JuMP.value.(v))
println("Optimal(?) point: ", JuMP.value.(theta))

println("Dual variables/Lagrange multipliers corresponding to maximum
    capacity constraints: ")
println(JuMP.dual.(JuMP.UpperBoundRef.(A)))
```