Technische Universität Berlin
Fakultät VII - Volkswirtschaftslehre und Wirtschaftsrecht
Fachgebiet Wirtschafts- und Infrastrukturpolitik

Energy Economics - Energy Sector Modeling

Homework 2

Student:	Richard Hauschild	Claudia Günther	Christian Gentsch
Matr #:	358081	576419	349596

Examiner:	Prof.von Hirschhausen	Jens Weibezahn	Richard Weinhold

Berlin, July 3, 2018

Contents

List of Tables

1 Task 1: The Model

(a) The power transfer distribution factors (PTDF) matrix gives the node-line sensitivity of the modelled power network. These sensitivities show how power flows change if power injection between nodes is shifted, resulting in different line flows. Normally, the power flow of an electrical network is determined by a set of nonlinear equations. Since nonlinear equations are more complex to solve and take more time while doing so, the power flow equations can be drastically simplified by making appropriate assumptions. Specifically the following assumptions are made[1]:

- neglecting all reactive power flows
- line resistances are negligible compared to line reactances ($R_L << X_L$), implying that the network is practically lossless
- small differences in neighbouring voltage angles
- flat voltage profile / equal voltage amplitude all nodes

The PTDF matrix is derived from the incidence matrix, describing which nodes are connected to which lines and each line's susceptance. Unfortunately, the equations defining the PTDF matrix are linearly dependent. This results in a non-invertible matrix. Therefore a reduced incidence matrix is created by defining a slack node. This slack node is removed from the incidence matrix and assumed to balance all other grid injections. Its PTDF values are set to zero, meaning that changing its net injection will not affect line flows. The PTDF matrix can then be derived as a product of the Line susceptance matrix and the inverse of the node susceptance matrix.

(b) The objective function maximizes total welfare, which is equal to the sum of consumer (demand) and producer (supply) surplus. Since producer's revenues represent costs to consumers, the maximization problem can be reduced to the difference between gross consumer surplus and total costs. Ehrenmann and Smeers choose symmetric demand and marginal cost function that include a linear (a) and quadratic (b) term. Generators are located at node 1, 2 and 4 and consumers at nodes 3, 5 and 6. Node 6 is defined as slack node.

(c) The problem at hand is a convex quadratic problem. Commercial solvers like *Gurobi Optimizer*[2] are able to solve linear problems as well as mixed integer problems and quadratic problems or quadratically constrained poroblems. Since a free academic version of this solver is available, it is used to solve the quadratic problem formulation. It's also possible to use the *IpoptSolver*[3] as a free source alternative.

[1] EW-MOD lecture, 28.05.2018, slide 8
[2] http://www.gurobi.com/products/gurobi-optimizer
[3] https://github.com/JuliaOpt/Ipopt.jl

(d) The model formulation in the template differentiates between demand and supply quantities for each node and each nodes net injection. Consequently, the first constraint describes the nodal energy balance according to which generation and consumption at each node must equal its net injection. The second constraint relates to the balancing character of the slack node, whose net injection must balance the sum of all other nodes net injections. In constrast, Ehrenmann and Smeers do not explicitly model net injection at each node. Instead, they incorporate only electricity quantities q_i for producer and consumer nodes. The constraint for their slack node simply states that the overall quantity of electricity must be zero, i.e. that generation must equal demand.

Both formulations are equal because nodes are strictly devided into producer and consumer nodes. This means that the quantities consumed/produced at a node equal its (negative) net injection by definition. Though the formulation of Ehrenmann and Smeers might be more intuitive for a microeconomic perspective. Explicitly defining net injections for each node has the advantage that the model can easily be extended to nodes incorporating both consumers and producers. Furthermore, it easily allows to get nodal prices as dual variables on the energy balance of each node.

(e) The market price is defined as the inversed demand function (demand nodes) or as the marginal costs function (supply nodes) depending on the electricity quantity. Demand and supply nodes can be distinguished by indexing the electricity demand either on DEM_NODES or SUP_NODES.

(f) We can derive prices based on marginal costs/utility in each node. This is equal to $a + b \cdot q$ for all nodes. Equivalently, we can derive prices based on the dual values of the energy balace constraint. Resulting line flows can be obtained by multiplying the PTDF matrix with the net injection vector. The following code snippet shows the actual implementation in the template.

```
results = Dict()
results["objective"]    = getobjectivevalue(Ehrenmann)
results["quantity"]     = NamedArray(getvalue(Q).innerArray)
results["prices_dual"]  = NamedArray(getdual(
                                 EnergyBalance).innerArray)
results["prices_marg"]  = [10;15;37.5;42.5;75;80] +
     [0.05;0.05;-0.05;0.025;-0.1;-0.1].*results["quantity"]
results["netinjection"] = getvalue(INJ).innerArray
results["flows"]        = ptdf_array*results["netinjection"]
```

(g) The result corresponds to the one zone setup displayed in "Table 10: One Zone"[1, p.18]. All nodes share the same marginal price except for node four, where the generator is driven out of the market. Interestingly, the dual values from the energy balance constraint differ at node four where the dual value is 35, according to the results in table 1.

Node	Quantity	Price	Dual	Line	Flow on Line
1	500	35	35	(1-6)	434.38
2	400	35	35	(2-5)	415.63
3	-50	35	35		
4	0	42.5	35		
5	-400	35	35		
6	-450	35	35		

Table 1: Simulation results with no line restricitons

2 Task 1: Equations and Results

(a) As written above, marginal price and dual price differ in the solution at node four. This is due to the nature of the problem and constraint formulation. The marginal prices are calculated based on the cost function coefficients a and b as well as the quantity Q. Supplying node four has the highest cost coefficients and is therefore consequently driven out of the market. Because of the unlimited transfer capacities, nodes one and two can fully supply the whole network demand by themselves. The dual price or local marginal price (LMP) indicates what an increase of 1 p.u. would cost at each node. There are no restrictions in transfer capacities so that the cheapest supply nodes can provide the systems demand and the price at each node is the same at 35.

(b) The additional constraints for upper and lower limits of line flows are implemented using the the @constraintref command which creates an empty container of a specified length, which can be later filled with constraints. That way it is possible to create constraints iteratively using for loops and if/else statements, as the following code documents.

```
@constraintref CapacityConstraints[1:length(keys(cap))*2]
i = 1 # counter for constraint index
for line in LINES
    if line in keys(cap)
        CapacityConstraints[i] = @constraint(Ehrenmann,
            sum(ptdf[line, node]*INJ[node] for node in NODES)
            <= cap[line]
        );
        i = i + 1
        CapacityConstraints[i] = @constraint(Ehrenmann,
            sum(ptdf[line, node]*INJ[node] for node in NODES)
            >= -cap[line]
        );
        i = i + 1
    end
end
```

The solution corresponds to "Table 4: Results for nodal pricing"[1, p.12]. In contrast to the previous results, the marginal and dual price now share the same values. This is due to the fact, that the network capacity constraints enforce the problem to generate nodal prices. The restrictions on line flows for line 1-6 and line 2-5, are forcing supply node four to produce as well because not enough energy can be transferred by the two constrained lines in order to satisfy the demands of nodes five and six.

(c) In order to force equal prices for each market, a new variable and a fourth constraint need to be implemented. But first of all, a dictionary assigning all nodes to their corresponding zones is needed. This is done as the following code segment shows.

```
# dictionary assigning nodes to zones
NODES_IN_ZONE = Dict()
for node in NODES, zone in keys(PRICE_ZONES)
    if node in PRICE_ZONES[zone]
        NODES_IN_ZONE[node] = zone
    end
end
```

The new variable PRICE is indexed for all predefined zones and added to the @variables list as the code below indicates.

```
@variables Ehrenmann begin
    Q[NODES] >= 0
    INJ[NODES]
    FLOWS
    PRICE[keys(PRICE_ZONES)] # new variable for forced prices
end
```

Lastly the constraint, forcing all prices in a zone to be equal is implemented as follows.

```
@constraint(Ehrenmann, ZonalPrice[node=NODES],
    a[node]+b[node].*Q[node] == PRICE[NODES_IN_ZONE[node]]
    );
```

Node	Quantity	Price	Dual	Line	Flow on Line
1	343.75	27.19	24.61	(1-6)	200.00
2	243.75	27.19	29.77	(2-5)	181.25
3	-206.25	27.19	27.19		
4	212.50	47.81	47.81		
5	-271.88	47.81	45.23		
6	-321.88	47.81	50.39	Welfare	22806.64

Table 2: Simulation results for two zones

(d)

Node	Quantity	Price	Dual	Line	Flow on Line
1	322.94	26.15	24.89	(1-6)	200.00
2	298.62	29.93	29.93	(2-5)	194.50
3	-227.06	26.15	27.41		
4	183.49	47.09	47.59		
5	-279.13	47.09	45.07		
6	-298.85	50.11	50.11	Welfare	22943.23

Table 3: Simulation results for four zones

References

[1] Andreas Ehrenmann, Yves Smeers, *Inefficiencies in European Congestion Management Proposals*, Utilities policy journal, Volume 13, Pages 135-152, Pergamon, 2005.