Appendix

Appendix A

Generating Unit Data

Apparent power base value S_{Base}=100 MVA

Table A.1 Generating unit cost coefficients for Duisburg municipal power system

Units' no. and Identification			Type	P_G^{min} [MW]	P_G^{max} [MW]	α $\left[\frac{CU}{h}\right]$	β $\left[\frac{CU}{MWh}\right]$	γ $\left[\frac{CU}{MW^2h}\right]$
No.	Local	Numeral						
1	HKW1	HKW1	TH	28.8	96.0	80.0	8.0	0.02400
2	HKW2	BLA	TH	21.6	72.0	80.0	8.0	0.02400
3	HKW2	BLB	TH	42.0	140.0	78.0	7.97	0.00482
4	HKW3	DT	TH	42.0	140.0	78.0	7.97	0.00482
5	HKW3	GT	GT	3.80	38.0	120.0	6.00	0.04000

CU fictional currency unit
Realistic cost coefficients α, β, and γ obtained from [24,34,39]
P_G^{min} computed as a percentage of the values given in table 3.2b (see chapter 3)
TH thermal generating unit
GT gas turbine generating unit

Table A.2 Generating unit cost coefficients for 400/230/110 kV transmission system

Units' no. and Identification			Type	P_G^{min} [MW]	P_G^{max} [MW]	α [CU/h]	β [CU/MWh]	γ [CU/MW²h]
No.	Local	Numeral						
1	SOL	B	TH	115.5	385.0	310.0	7.85	0.00194
2	SOL	E	TH	115.5	385.0	310.0	7.85	0.00194
3	MARL	1	TH	19.5	65.0	180.0	9.0	0.06
4	KHER	1	TH	165.0	550.0	561.0	7.92	0.00156
5	RX	N	TH	45.0	150.0	78.0	7.97	0.00482
6	KNEP	C	TH	103.5	345.0	310.0	7.85	0.00194
7	GKW	M7	TH	105.0	350.0	310.0	7.85	0.00194
8	KGW	K2	TH	196.5	655.0	561.0	7.70	0.00156
9	KGW	F2	TH	109.5	365.0	310.0	7.85	0.00194
10	KGW	G2	TH	109.5	365.0	310.0	7.85	0.00194
11	KGW	H2	TH	109.5	365.0	310.0	7.85	0.00194
12	KGW	I2	TH	109.5	365.0	310.0	7.85	0.00194
13	KGW	K1	GT	11.0	110.0	80.0	8.00	0.024
14	KGW	F1	GT	5.5	55.0	80.0	8.00	0.024
15	KGW	G1	GT	5.5	55.0	80.0	8.00	0.024
16	KGW	H1	GT	5.5	55.0	80.0	8.00	0.024
17	KGW	I1	GT	5.5	55.0	80.0	8.00	0.024
18	BKAM	A	TH	224.1	747.0	749.0	6.95	0.00099
19	KWE	C	TH	87.0	290.0	328.1	8.66	0.00525
20	KEM	B2	TH	109.5	365.0	310.0	7.85	0.00194
21	KEM	C2	TH	109.5	365.0	310.0	7.85	0.00194
22	KEM	B1	GT	5.5	55.0	80.0	8.00	0.024
23	KEM	C1	GT	5.5	55.0	80.0	8.00	0.024
24	KKE	1	PWR	405.0	1350.	1285	7.05	0.00073

PWR pressurized water reactor generating unit

Appendix B

Knowledge Base Development

B.1 Applied Rule Syntax

Both forward and backward chained rules consist of heading (conclusion) and condition parts; the latter could be made of one or more conditions connected by conjunctive “and”. The principle of applied rule representation [40] is as depicted in figure B.1. It can be seen that the structure of the conditions part of the rules for both chaining methods is identical, while they differ in the structure and number of arguments of the conclusion part. The arguments are in particular:

- The rule’s name as first argument of the conclusion and each condition for its identification within the inference mechanism, written in lowercase letters.

A forward chained rule is represented as:

rule (name, [sequence rules]):-
condition (name, a_1, b_1,c_1, ci_1, log_1, res_1),
condition (name, a_2, b_2, c_2, ci_2, log_2, res_2),
..,.., .

while a backward chained rule is similarly represented as:

rule (name, a_0, b_0, c_0):-
condition (name, a_1, b_1, c_1, ci_1, log_1, res_1),
condition (name, a_2, b_2, c_2, ci_2, log_2, res__2),
..,.., .

Figure B.1 Rule syntax

- Text elements for the transformative grammar in the form of subject (a_y), predicate (b_y), object/attribute (c_y), by combination of which various types of sentences can be formed for user communication if required [51]. Each expression is enclosed in commas as ‘string’. The conclusion part of

backward chained rules also consists of such text elements ‘a_0’, ‘b_0’, and ‘c_0’ which are tried to be matched with those of the condition part of another rule by the inference engine, thus constituting a backward chain.

- The argument [sequence rules] of forward chained rules which consists of a list of zero or more names of rules to be evaluated in the next sequence in the case of positive termination.
- The control instruction (ci_y) which directs the inference mechanism where the required factual information (case data) for the evaluation of the corresponding condition can be obtained. This could take the form of:
  - direct dialog with the system operator or the user indicated with the assignment “#” or
  - the results obtained through pre-evaluation of external routines accessed through the Fortran-interface of the Prolog interpreter. This condensed factual information is usually stored in object files until required by the ES for further application in reasoning. The indication used to direct the inference engine is made with the assignment “prolog”.
- The expected termination value (log_y) which could be positive or negative depending on the logical results of the evaluation of the conditions part of a rule.
- The notation (res_y) which could be any of the form “diagnose”, or “action” which influences the behavior of the inference mechanism in testing of the actual condition [51].

B.2 Rules for the Network State Assessment and Enhancement Scheme

Using the rule syntax described above and the strategy of chapter 6 (see figure 6.1), the rules required for the hybrid system of network state assessment and enhancement are as shown in figure B.2.

```
module saekb.

/*$eject*/
body.
saekb_rule(saekb,[test_1]):-
nl,write_tab(10),write('The network state problem is to be solved with expert
system'),nl.

saekb_rule(test_1,[limit_detection]):-
condition(verify, 'you', 'would','like to enhance the network state by
ES',#,positive ,fix).

saekb_rule(test_1,[standby]):-
condition(verify, 'you','would','like to enhance the network state by ES',#,
negative, fix).

saekb_rule (limit_detection,[overload_detection]):-
condition (limit_detection,'there','is','any limits
violation',prolog,positive,action).

saekb_rule (limit_detection,[exit]):-
condition (limit_detection,'there', 'is','any limits
violation',prolog,negative,action).

saekb_rule(voltage_iteration,[quit]):-
condition(voltage_iteration,'allowable number of voltage
iteration','is','exceeded', prolog, positive, action).

saekb_rule(voltage_iteration,[control_selection]):-
condition(voltage_iteration,'allowable number of voltage
iteration','is','exceeded', prolog, negative, action).

saekb_rule(overload_iteration,[voltage_detection]):-
condition(overload_iteration,'allowable number of overload
iteration','is','exceeded', prolog, positive, action).

saekb_rule(overload_iteration,[branch_overload]):-
condition (overload_iteration,'allowable number of overload
iteration','is','exceeded', prolog, negative, action).

saekb_rule (voltage_detection,[voltage_iteration]):-
condition (voltage_detection,'limits violation','is','voltage
related',prolog,positive, action).
```

Figure B.2 Knowledge base of the hybrid system of network state assessment and enhancement

```
saekb_rule (voltage_detection,[quit]):-
condition (voltage_detection,'limits
volation','is','voltage_related',prolog,negative, action).

saekb_rule (control_selection,[voltage_problem]):-
condition(control_selection,'reactive power
controllers',''are','available',prolog, positive, action).

saekb_rule (control_selection,[quit]):-
condition(control_selection,'reactive power
controllers',''are','available',prolog, negative, action).

saekb_rule (overload_detection,[overload_iteration]):-
condition(overload_detection,'limits violation',''is','overload
related',prolog,positive, action).

saekb_rule (overload_detection,[voltage_detection]):-
condition (overload_detection, 'limits violation','is','overload
related',prolog ,negative, action).

saekb_rule(branch_overload,[limits_detection]):-
condition (branch_overload, 'overload problem','is','being
solved',prolog,positive, action).

saekb_rule(voltage_problem,[limits_detection]):-
condition (voltage_problem, 'voltage problem','is','being
solved',prolog,positive, action).

saekb_rule(exit,[saekb]):-
nl,write_tab(10),write("The system is successfully restored to normal state").

saekb_rule(quit,[saekb]):-
nl,write_tab(10),write("The problem cannot be solved, consider
other measures").

saekb_rule(standby,[saekb]):-
nl,write_tab(10),write("The problem is not yet solved").

endmod /* saekb */.
```

Figure B.2 Continued

Appendix C

Transmission Real Power Losses

Consider a two terminal π equivalent of a transmission line shown in figure C.1, the current at node *l* is given by the summation of the current in the branch connecting node *l* to *m* and shunt charging current

$$I_l = (V_l - V_m) y_{lm} + \frac{y_0}{2} V_l$$

where

$y_{lm} = g_{lm} + jb_{lm}$	branch admittance
$y_0 = g_0 + jb_0$	total shunt charging admittance
g_{lm}	branch conductance
b_{lm}	branch susceptance
g_0	shunt conductance
b_0	shunt susceptance
$V_l = \|V_l\|(\cos\delta_l + j\sin\delta_l)$	complex voltage at node *l*
$V_m = \|V_m\|(\cos\delta_m + j\sin\delta_m)$	complex voltage at node *m*

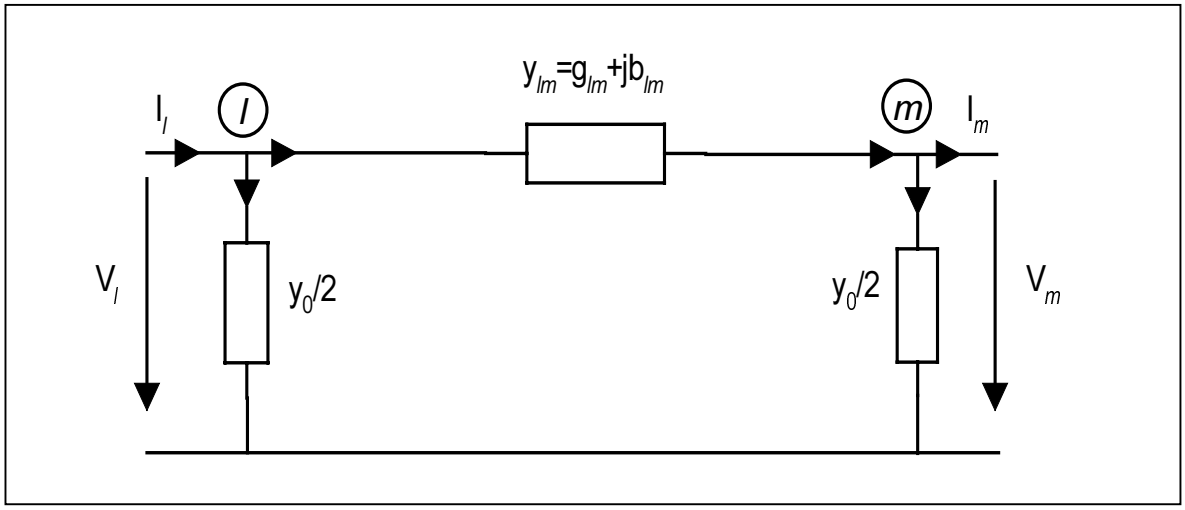

Figure C.1 π-equivalent of a transmission line

The real power injected into the branch at node *l* is given by

$$P_l = \mathrm{Re}\{V_l I_l^*\} = \mathrm{Re}\{V_l^* I_l\}$$
$$= \mathrm{Re}\left\{V_l^* \cdot (V_l - V_m) \cdot y_{lm} + V_l^* \cdot V_l \cdot \frac{y_0}{2}\right\}$$

$$= \mathrm{Re}\left\{\left(\left|V_l\right|^2 - \left|V_l V_m\right| \cdot \left(\left(\cos(\delta_l) - \mathrm{j}\sin(\delta_l)\right)\left(\cos(\delta_m) + \mathrm{j}\sin(\delta_m)\right)\right)\right) \cdot y_{lm} + \left|V_l\right|^2 \frac{y_0}{2}\right\}$$

$$= \mathrm{Re}\left\{\left(\left|V_l\right|^2 - \left|V_l V_m\right| \cdot \left(\cos(\delta_l - \delta_m) - \mathrm{j}\sin(\delta_l - \delta_m)\right)\right) \cdot \left(g_{lm} + \mathrm{j}b_{lm}\right) + \left|V_l\right|^2 \frac{y_0}{2}\right\}$$

$$P_l = \left(\left|V_l\right|^2 - \left|V_l V_m\right| \cdot \cos(\delta_l - \delta_m)\right) \cdot g_{lm} - \left|V_l V_m\right| \cdot \sin(\delta_l - \delta_m) \cdot b_{lm} + \left|V_l\right|^2 \frac{g_0}{2} \qquad \text{(C.1)}$$

Similarly, at node *m* the real power injected into the branch is given by

$$P_m = \mathrm{Re}\left\{V_m I_m^*\right\} = \mathrm{Re}\left\{V_m^* I_m\right\}$$

$$= \mathrm{Re}\left\{V_m^* \cdot \left(V_m - V_l\right) \cdot y_{lm} + V_m^* \cdot V_m \cdot \frac{y_0}{2}\right\}$$

$$= \mathrm{Re}\left\{\left(\left|V_m\right|^2 - \left|V_l V_m\right| \cdot \left(\left(\cos(\delta_m) - \mathrm{j}\sin(\delta_m)\right)\left(\cos(\delta_l) + \mathrm{j}\sin(\delta_l)\right)\right)\right) \cdot y_{lm} + \left|V_m\right|^2 \frac{y_0}{2}\right\}$$

$$= \mathrm{Re}\left\{\left(\left|V_m\right|^2 - \left|V_l V_m\right| \cdot \left(\cos(\delta_l - \delta_m) + \mathrm{j}\sin(\delta_l - \delta_m)\right)\right) \cdot \left(g_{lm} + \mathrm{j}b_{lm}\right) + \left|V_m\right|^2 \frac{y_0}{2}\right\}$$

$$P_m = \left(\left|V_m\right|^2 - \left|V_l V_m\right| \cdot \cos(\delta_l - \delta_m)\right) \cdot g_{lm} + \left|V_l V_m\right| \cdot \sin(\delta_l - \delta_m) \cdot b_{lm} + \left|V_m\right|^2 \frac{g_0}{2} \qquad \text{(C.2)}$$

The real power loss in the π-equivalent branch is the algebraic sum of the power flows of equations (C.1) and (C.2) given by

$$\begin{aligned} P_{lm}^{\mathrm{sum}} &= P_l + P_m \\ &= \left(\left|V_l\right|^2 + \left|V_m\right|^2 - 2 \cdot \left|V_l V_m\right| \cdot \cos(\delta_l - \delta_m)\right) \cdot g_{lm} + \left|V_l\right|^2 \frac{g_0}{2} + \left|V_m\right|^2 \frac{g_0}{2} \end{aligned} \qquad \text{(C.3)}$$

Neglecting the shunt losses, the transmission losses P_T for the n_{Br} branches of the system under regard are given by

$$P_T = \sum_{n_{Br}} g_{lm} \left(\left|V_l\right|^2 + \left|V_m\right|^2 - \mathbf{2} \cdot \left|V_l V_m\right| \cdot \cos\delta_{lm}\right) \qquad \text{(C.4)}$$

where

$\delta_{lm} = \delta_l - \delta_m$ voltage angular difference

Appendix D

Linear Mapping between Real Numbers and Binary Strings

To carry out the transformation between strings of a fixed length and a real number $\hat{y}$, the binary string (genotype) is first converted to a base-10 integer $\hat{x}$. This integer is then transformed to a real number using

$$\hat{y} = \hat{m}\hat{x} + \hat{c} \tag{D.1}$$

The values of $\hat{m}$ and $\hat{c}$ depend on the location and width of the parameter space. Their expressions can be derived from the two equations

$$\hat{y}^{min} = \hat{m}\hat{x}^{min} + \hat{c} \tag{D.2}$$

$$\hat{y}^{max} = \hat{m}\hat{x}^{max} + \hat{c} \tag{D.3}$$

where $\hat{y}^{min}$, $\hat{y}^{max}$, $\hat{x}^{min}$ and $\hat{x}^{max}$ are respectively the minimum and maximum possible parameters in real and integer representation. The smallest and largest numbers that can be represented by the binary in base-10 are respectively given by $\hat{x}^{min} = 0$ and $\hat{x}^{max} = 2^{l_c} - 1$, where l_c is the binary string length.
Subtracting equation (D.2) from equation (D.3) and substituting $\hat{x}^{min} = 0$ in equation (D.2), we have

$$\hat{m} = \frac{\hat{y}^{max} - \hat{y}^{min}}{2^{l_c} - 1}, \text{ and } \hat{c} = \hat{y}^{min}.$$

Substituting these in the original equation (D.1), the required transformation is given by

$$\hat{y} = \hat{y}^{min} + \frac{\hat{y}^{max} - \hat{y}^{min}}{2^{l_c} - 1}\hat{x} \tag{D.4}$$

Illustration with an Example

Given a problem where the unknown parameter $\hat{y}$ being sought is known to lie between 3.2 and 4.9, the binary string 11001 is mapped to this space as follows:

string=11001, therefore its base-10 value is $\hat{x}=25$ and $l_c=5$.

By using equation (D.4) the required transformation to the real value is

$$\hat{y} = 3.2 + \frac{4.9 - 3.2}{2^5 - 1} 25 = 4.571$$