LAMPIRAN

1. Struktur Jaringan 33 Bus(22)

Branch No.	From bus	To bus	R (Ω)	X (Ω)
1	1	2	0.0922	0.047
2	2	3	0.493	0.2511
3	3	4	0.366	0.1864
4	4	5	0.3811	0.1941
5	5	6	0.819	0.707
6	6	7	0.1872	0.6188
7	7	8	0.7114	0.2351
8	8	9	1.03	0.74
9	9	10	1.044	0.74
10	10	11	0.1966	0.065
11	11	12	0.3744	0.1238
12	12	13	1.468	1.155
13	13	14	0.5416	0.7129
14	14	15	0.591	0.526
15	15	16	0.7463	0.545
16	16	17	1.289	1.721
17	17	18	0.732	0.574
18	2	19	0.164	0.1565
19	19	20	1.5042	1.3554
20	20	21	0.4095	0.4784
21	21	22	0.7089	0.9373
22	3	23	0.4512	0.3083
23	23	24	0.898	0.7091
24	24	25	0.896	0.7011
25	6	26	0.203	0.1034
26	26	27	0.2842	0.1447
27	27	28	1.059	0.9337
28	28	29	0.8042	0.7006
29	29	30	0.5075	0.2585
30	30	31	0.9744	0.963
31	31	32	0.3105	0.3619
32	32	33	0.341	0.5302

2. Struktur Beban 33 Bus(22)

Bus number	P (kW)	Q(kVar)
1	0	0
2	100	60
3	90	40
4	120	80
5	60	30
6	60	20
7	200	100
8	200	100
9	60	20
10	60	20
11	45	30
12	60	35
13	60	35
14	120	80
15	60	10
16	60	20
17	60	20
18	90	40
19	90	40
20	90	40
21	90	40
22	90	40
23	90	50
24	420	200
25	420	200
26	60	25
27	60	25
28	60	20
29	120	70
30	200	600
31	150	70
32	210	100
33	60	40

3. Struktur Jaringan 69 Bus(22)

Branch No.	From bus	To bus	R (Ω)	X (Ω)	Length (km)
1	1	2	0.0005	0.0012	0.8
2	2	3	0.0005	0.0012	0.7
3	3	4	0.0015	0.0036	0.6
4	4	5	0.0251	0.0294	0.7
5	5	6	0.366	0.1864	0.6
6	6	7	0.3811	0.1941	0.6
7	7	8	0.0922	0.047	0.6
8	8	9	0.0493	0.0251	0.8
9	9	10	0.819	0.2707	0.6
10	10	11	0.1872	0.0619	0.8
11	11	12	0.7114	0.2351	0.8
12	12	13	1.03	0.34	0.7
13	13	14	1.044	0.345	0.7
14	14	15	1.058	0.3496	0.6
15	15	16	0.1966	0.065	0.8
16	16	17	0.3744	0.1238	0.6
17	17	18	0.0047	0.0016	0.6
18	18	19	0.3276	0.1083	0.7
19	19	20	0.2106	0.069	0.6
20	20	21	0.3416	0.1129	0.7
21	21	22	0.014	0.0046	0.6
22	22	23	0.1591	0.0526	0.8
23	23	24	0.3463	0.1145	0.6
24	24	25	0.7488	0.2475	0.6
25	25	26	0.3089	0.1021	0.8
26	26	27	0.1732	0.0572	0.7
27	3	28	0.0044	0.0108	0.6
28	28	29	0.064	0.1565	0.6
29	29	30	0.3978	0.1315	0.8
30	30	31	0.0702	0.0232	0.7
31	31	32	0.351	0.116	0.7
32	32	33	0.839	0.2816	0.7
33	33	34	1.708	0.5646	0.6
34	34	35	1.474	0.4873	0.8
35	3	36	0.0044	0.0108	0.6

4. Struktur Jaringan 69 Bus(22) (Lanjutan)

36	36	37	0.064	0.1565	0.7
37	37	38	0.1053	0.123	0.6
38	38	39	0.0304	0.0355	0.7
39	39	40	0.0018	0.0021	0.8
40	40	41	0.7283	0.8509	0.8
41	41	42	0.31	0.3623	0.6
42	42	43	0.041	0.0478	0.8
43	43	44	0.0092	0.0116	0.7
44	44	45	0.1089	0.1373	0.8
45	45	46	0.0009	0.0012	0.6
46	4	47	0.0034	0.0084	0.6
47	47	48	0.0851	0.2083	0.7
48	48	49	0.2898	0.7091	0.6
49	49	50	0.0822	0.2011	0.8
50	8	51	0.0928	0.0473	0.8
51	51	52	0.3319	0.1114	0.6
52	9	53	0.174	0.0886	0.8
53	53	54	0.203	0.1034	0.8
54	54	55	0.2842	0.1447	0.7
55	55	56	0.2813	0.1433	0.6
56	56	57	1.59	0.5337	0.8
57	57	58	0.7837	0.263	0.6
58	58	59	0.3042	0.1006	0.8
59	59	60	0.3861	0.1172	0.7
60	60	61	0.5075	0.2585	0.6
61	61	62	0.0974	0.0496	0.8
62	62	63	0.145	0.0738	0.6
63	63	64	0.7105	0.3619	0.6
64	64	65	1.041	0.5302	0.7
65	11	66	0.2012	0.0611	0.7
66	66	67	0.0047	0.0014	0.6
67	12	68	0.7394	0.2444	0.6
68	68	69	0.0047	0.0016	0.7

5. Struktur Beban 69 Bus(22)

Bus number	P (kW)	Q(kVar)
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	2.6	2.2
7	40.4	30
8	75	54
9	30	22
10	28	19
11	145	104
12	145	104
13	8	5.5
14	8	5.5
15	0	0
16	45.5	30
17	60	35
18	60	35
19	0	0
20	1	0.6
21	114	81
22	5	3.5
23	0	0
24	28	20
25	0	0
26	14	10
27	14	10
28	26	18.6
29	26	18.6
30	0	0
31	0	0
32	0	0
33	14	10
34	19.5	14
35	6	4
36	26	18.55
37	26	18.55

6. Struktur Beban 69 Bus(22) (Lanjutan)

38	0	0
39	24	17
40	24	17
41	1.2	1
42	0	0
43	6	4.3
44	0	0
45	39.2	26.3
46	39.2	26.3
47	0	0
48	79	56.4
49	384.7	274.5
50	384.7	274.5
51	40.5	28.3
52	3.6	2.7
53	4.35	3.5
54	26.4	19
55	24	17.2
56	0	0
57	0	0
58	0	0
59	100	72
60	0	0
61	1244	888
62	32	23
63	0	0
64	227	162
65	59	42
66	18	13
67	18	13
68	28	20
69	28	20

7. Struktur Jaringan dan Beban 85 Bus(22)

S.No	From	To	R	X	P	Q
1	1	2	0.067384	0.046794	0	0
2	2	3	0.1017	0.06988	0	0
3	3	4	0.135392	0.092965	0.00056	0.000571
4	4	5	0.067384	0.04617	0	0
5	5	6	0.271408	0.18593	0.000353	0.00036
6	6	7	0.169708	0.11605	0	0
7	7	8	0.746839	0.511619	0.000353	0.00036
8	8	9	0.067384	0.04617	0	0
9	9	10	0.373107	0.255809	0	0
10	10	11	0.339415	0.232724	0.00056	0.000571
11	11	12	0.339415	0.232724	0	0
12	12	13	0.373107	0.255809	0	0
13	13	14	0.169708	0.11605	0.000353	0.00036
14	14	15	0.2034	0.139135	0.000353	0.00036
15	2	16	0.454218	0.188425	0.000353	0.00036
16	3	17	0.283886	0.117922	0.00112	0.001143
17	5	18	0.511619	0.212135	0.00056	0.000571
18	18	19	0.39744	0.164716	0.00056	0.000571
19	19	20	0.283886	0.117922	0.000353	0.00036
20	20	21	0.510995	0.212135	0.000353	0.00036
21	21	22	0.965836	0.40056	0.000353	0.00036
22	19	23	0.113554	0.046794	0.00056	0.000571
23	7	24	0.567772	0.235844	0.000353	0.00036
24	8	25	0.283886	0.117922	0.000353	0.00036
25	25	26	0.227109	0.094213	0.00056	0.000571
26	26	27	0.340663	0.141007	0	0
27	27	28	0.170332	0.070504	0.00056	0.000571
28	28	29	0.340663	0.141007	0	0
29	29	30	0.340663	0.141007	0.000353	0.00036
30	30	31	0.170332	0.070504	0.000353	0.00036
31	31	32	0.113554	0.046794	0	0
32	32	33	0.113554	0.046794	0.00014	0.000143
33	33	34	0.510995	0.212135	0	0
34	34	35	0.39744	0.164716	0	0
35	35	36	0.113554	0.046794	0.000353	0.00036
36	26	37	0.227109	0.094213	0.00056	0.000571
37	27	38	0.625173	0.259553	0.00056	0.000571
38	29	39	0.340663	0.141007	0.00056	0.000571

8. Struktur Jaringan dan Beban 85 Bus(22) (Lanjutan)

39	32	40	0.283886	0.117922	0.000353	0.00036
40	40	41	0.625173	0.259553	0	0
41	41	42	0.170332	0.070504	0.000353	0.00036
42	41	43	0.283886	0.117922	0.000353	0.00036
43	34	44	0.625173	0.261425	0.000353	0.00036
44	44	45	0.568396	0.235844	0.000353	0.00036
45	45	46	0.568396	0.235844	0.000353	0.00036
46	46	47	0.340663	0.141007	0.00014	0.000143
47	35	48	0.39744	0.164716	0	0
48	48	49	0.113554	0.046794	0	0
49	49	50	0.227109	0.094213	0.000363	0.00037
50	50	51	0.283886	0.117922	0.00056	0.000571
51	48	52	0.852282	0.353766	0	0
52	52	53	0.283886	0.117922	0.000353	0.00036
53	53	54	0.340663	0.141007	0.00056	0.000571
54	52	55	0.340663	0.141007	0.00056	0.000571
55	49	56	0.340663	0.141007	0.00014	0.000143
56	9	57	0.170332	0.070504	0.00056	0.000571
57	57	58	0.510995	0.212135	0	0
58	58	59	0.113554	0.046794	0.00056	0.000571
59	58	60	0.340663	0.141007	0	0
60	60	61	0.454218	0.188425	0.00056	0.000571
61	61	62	0.625173	0.258929	0.00056	0.000571
62	60	63	0.113554	0.046794	0.00014	0.000143
63	63	64	0.454218	0.188425	0	0
64	64	65	0.113554	0.046794	0	0
65	65	66	0.113554	0.046794	0.00056	0.000571
66	64	67	0.283886	0.117922	0	0
67	67	68	0.567772	0.235844	0	0
68	68	69	0.681326	0.282638	0.00056	0.000571
69	69	70	0.283886	0.117922	0	0
70	70	71	0.340663	0.141007	0.000353	0.00036
71	67	72	0.113554	0.046794	0.00056	0.000571
72	68	73	0.738728	0.306347	0	0
73	73	74	0.170332	0.070504	0.00056	0.000571
74	73	75	0.625173	0.259553	0.000353	0.00036
75	70	76	0.340663	0.141007	0.00056	0.000571
76	65	77	0.056777	0.023085	0.00014	0.000143
77	10	78	0.39744	0.164716	0.00056	0.000571
78	67	79	0.340663	0.141007	0.000353	0.00036
79	12	80	0.454218	0.188425	0.00056	0.000571

9. Struktur Jaringan dan Beban 85 Bus(22) (Lanjutan)

80	80	81	0.227109	0.094213	0	0
81	81	82	0.056777	0.023085	0.00056	0.000571
82	81	83	0.681326	0.282638	0.000353	0.00036
83	83	84	0.625173	0.259553	0.00014	0.000143
84	13	85	0.510995	0.212135	0.000353	0.00036

10. Struktur Jaringan Penyuang 5 GI Bantul

%	No.	Fb	Tb	R(Ohm)	Q(Ohm)
	1	1	2	0.133004	0.004808
	2	2	3	0.016	0.021
	3	3	4	0.03475	0.09755
	4	4	5	0.40257	1.30184
	5	5	6	0.0670	0.18836
	6	6	7	0.0894	0.2511
	7	6	8	0.5478	1.5383
	8	8	9	0.0782	0.21975
	9	9	10	0.20128	0.565
	10	10	11	0.3429	0.910281
	11	11	12	0.0670	0.18836
	12	12	13	0.2907	0.8161
	13	13	14	0.0782	0.2197
	14	14	15	0.7827	2.1975
	15	14	16	0.7817	2.1975
	16	11	17	0.3690	1.036
	17	17	18	0.178	0.5023
	18	17	19	0.0111	0.0313
	19	19	20	0.9281	2.6057
	20	20	21	0.3578	1.0046
	21	21	22	0.4473	1.2257
	22	22	23	0.02236	0.06278
	23	23	24	0.4584	1.2871
	24	22	25	0.0111	0.03139
	25	25	26	2.046	5.7451
	26	26	27	0.27956	0.784
	27	27	28	0.1565	0.4395
	28	28	29	0.2795	0.784
	29	27	30	0.570	1.60
	30	30	31	0.2795	0.784
	31	30	32	0.49203	1.381
	32	32	33	0.0782	0.2197
	33	33	34	0.279	0.784
	34	28	35	0.4137	1.1615
	35	35	36	0.6821	1.915

11. Struktur beban Penyulang 5 GI Bantul

%No	Bus	P(kW)	Q(kVar)
1	1	136	93
2	2	1371	16772
3	3	1361	16772
4	4	1336	16767
5	5	1136	16273
6	6	1119	15372
7	7	0	0
8	8	890	14840
9	9	857	14765
10	10	774	14569
11	11	633	11041
12	12	4	1603
13	13	2	1602
14	14	1	801
15	15	0	803
16	16	0	803
17	17	541	9230
18	18	0	799
19	19	539	9226
20	20	369	8047
21	21	315	7922
22	22	247	6212
23	23	0	777
24	24	0	777
25	25	246	6210
26	26	46	5175
27	27	20	2819
28	28	2	1151
29	29	0	576
30	30	5	1397
31	31	0	1392
32	32	2	1389
33	33	2	1393
34	34	0	1390
35	35	6.8	574
36	36	0	575

12. Data *Intermitent Photovoltaic*

```
% Data Photovoltaic dari data Gatton Campus mode Instanous[22]

%No  Kapasitas(W)   Tanggal          Waktu
1    0              9-7-2019         06:00
2    0              9-7-2019         06:15
3    0              9-7-2019         06:30
4    0              9-7-2019         06:45
5    0              9-7-2019         07:00
6    60             9-7-2019         07:05
7    180            9-7-2019         07:10
8    280            9-7-2019         07:15
```

9	360	9-7-2019	07:20
10	400	9-7-2019	07:20
11	400	9-7-2019	07:25
12	520	9-7-2019	07:30
13	540	9-7-2019	07:35
14	700	9-7-2019	07:40
15	760	9-7-2019	07:45
16	860	9-7-2019	07:50
17	1040	9-7-2019	07:55
18	1380	9-7-2019	08:00
19	2100	9-7-2019	08:05
20	2740	9-7-2019	08:10
21	2860	9-7-2019	08:15
22	3360	9-7-2019	08:20
23	3940	9-7-2019	08:25
24	4880	9-7-2019	08:30
25	5600	9-7-2019	08:35
26	6420	9-7-2019	08:40
27	6700	9-7-2019	08:45
28	6960	9-7-2019	08:50
29	7360	9-7-2019	08:55
30	7900	9-7-2019	09:00
31	8520	9-7-2019	09:05
32	9140	9-7-2019	09:10
33	9580	9-7-2019	09:15
34	10440	9-7-2019	09:20
35	11080	9-7-2019	09:25
36	11680	9-7-2019	09:30
37	11860	9-7-2019	09:35
38	12180	9-7-2019	09:40
39	12340	9-7-2019	09:45
40	12360	9-7-2019	09:50
41	12820	9-7-2019	09:55
42	13040	9-7-2019	10:00
43	13140	9-7-2019	10:05
44	13620	9-7-2019	10:10
45	13920	9-7-2019	10:15
46	14400	9-7-2019	10:20
47	14680	9-7-2019	10:25
48	14920	9-7-2019	10:30
49	15100	9-7-2019	10:35
50	15240	9-7-2019	10:40
51	15360	9-7-2019	10:45
52	15420	9-7-2019	10:50
53	15620	9-7-2019	10:55
54	15800	9-7-2019	11:00
55	15920	9-7-2019	11:05
56	16100	9-7-2019	11:10
57	16280	9-7-2019	11:15
58	16440	9-7-2019	11:20
59	16400	9-7-2019	11:25
60	16480	9-7-2019	11:30
61	16500	9-7-2019	11:35
62	16540	9-7-2019	11:40

63	16540	9-7-2019	11:45
64	16520	9-7-2019	11:50
65	16600	9-7-2019	11:55
66	16580	9-7-2019	12:00
67	16900	9-7-2019	12:05
68	16620	9-7-2019	12:10
69	16820	9-7-2019	12:15
70	16680	9-7-2019	12:20
71	16520	9-7-2019	12:25
72	16860	9-7-2019	12:30
73	16540	9-7-2019	12:35
74	16600	9-7-2019	12:40
75	16800	9-7-2019	12:45
76	16960	9-7-2019	12:50
77	16360	9-7-2019	12:55
76	16240	9-7-2019	13:00
77	15960	9-7-2019	13:05
78	15900	9-7-2019	13:10
79	15980	9-7-2019	13:15
80	15980	9-7-2019	13:20
81	15380	9-7-2019	13:25
82	3380	9-7-2019	13:30
83	15760	9-7-2019	13:35
84	15280	9-7-2019	13:40
85	14600	9-7-2019	13:45
86	14220	9-7-2019	13:50
87	13960	9-7-2019	13:55
88	13840	9-7-2019	14:00
89	13240	9-7-2019	14:10
90	12620	9-7-2019	14:20
91	12000	9-7-2019	14:30
92	11260	9-7-2019	14:40
93	10620	9-7-2019	14:50
94	9800	9-7-2019	15:00
95	9040	9-7-2019	15:10
96	8360	9-7-2019	15:20
97	7580	9-7-2019	15:30
98	6520	9-7-2019	15:40
99	5300	9-7-2019	15:50
100	4100	9-7-2019	16:00
101	2760	9-7-2019	16:10
102	2060	9-7-2019	16:20
103	1540	9-7-2019	16:30
104	420	9-7-2019	16:40
105	160	9-7-2019	16:50
106	0	9-7-2019	17:00
107	0	9-7-2019	17:10
108	0	9-7-2019	17:20
109	0	9-7-2019	17:30
110	0	9-7-2019	17:40

ANT LION OPTIMIZER ALGORITHM

```
%_____________________________________________________________________%
%      Ant Lion Optimizer (ALO) source codes demo version 1.0         %
%                                                                     %
%                 Developed in MATLAB R2011b(7.13)                    %
%                                                                     %
%           Author and programmer: Seyedali Mirjalili                 %
%                                                                     %
%                   Edithor by Riki Khomarudin                        %
%              Muhammadiyah University Of Yogyakarta                  %
%                                                                     %
%                  e-Mail:ali.mirjalili@gmail.com                     %
%                seyedali.mirjalili@griffithuni.edu.au                %
%                                                                     %
%   Homepage: http://www.alimirjalili.com                             %
%                                                                     %
%   Main paper:                                                       %
%                                                                     %
%   S. Mirjalili, The Ant Lion Optimizer                              %
%   Advances in Engineering Software , in press,2015                  %
%   DOI: http://dx.doi.org/10.1016/j.advengsoft.2015.01.010           %
%                                                                     %
%_____________________________________________________________________%

% You can simply define your cost in a seperate file and load its handle
to fobj
% The initial parameters that you need are:
%_____________________________________________
% fobj = @YourCostFunction
% dim = number of your variables
% Max_iteration = maximum number of generations
% SearchAgents_no = number of search agents
% lb=[lb1,lb2,...,lbn] where lbn is the lower bound of variable n
% ub=[ub1,ub2,...,ubn] where ubn is the upper bound of variable n
% If all the variables have equal lower bound you can just
% define lb and ub as two single number numbers

% To run ALO:
[Best_score,Best_pos,cg_curve]=ALO(SearchAgents_no,Max_iteration,lb,ub,d
im,fobj)

%function
[Elite_antlion_fitness,Elite_antlion_position,Convergence_curve]=ALO(N,M
ax_iter,lb,ub,dim,fobj)
clear;
%clc;
tic;
global scenario alpha_t CENS ICdg MCdg minVoltage maxVoltage

minVoltage_=[0.90 0.91 0.92 0.93 0.94 0.95 0.96 0.97 0.98 0.99];
minVoltage=minVoltage_(6);
```

```
maxVoltage_=[1.00 1.05];
maxVoltage=maxVoltage_(2);

alpha_t_=[0.7 1 1.05];
alpha_t=alpha_t_(3);

CENS_=[0 8 10 20 30 40 50 60 78 90 100];
CENS=CENS_(4);

 %C_Purc=[1700 1800 1900 2000 2100 2200 2300 2400 2500];
 %CP=C_Purc(4);

MCdg=[26.04 33.00 36.48];
%MCdg=MCdg_(3);

%Ct_=[0.1 0.2 0.3];
%Ct=Ct_(1);

ICdg_=[100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500
1600 1700 1800 1900 2000 2100 2200 2300 2400 2500];
ICdg=ICdg_(6);

%Run base load flow to read data, create distribution structure, and
finevoltage
BaseLoadFlow;
br;
no;
LF;
brlenght=1;
PF=0.85;        % Power factor, option: 0.85; 0.95; 1.
%PF2=0.9;
%PF3=1;
R;
X;
P;
Q;
MVAb;
Vb;
adjcb;
adjb;
oldloss=PL;
%priceENS=CENS;
minVoltage;
maxVoltage;
plot(Voltage,'--rs','LineWidth',2,...
                'MarkerEdgeColor','g',...
                'MarkerFaceColor','g',...
                'MarkerSize',4)
set(gca,'XTick',1:1:no);
title('Voltage profile');
xlabel('Bus');
ylabel('Voltage magnitude (p.u.)');
hold all
```

```
% Dimension of the search variables
%ObjectiveFunction=@func;

populasi_DG=(dgmax-dgmin)*rand()+dgmin;

Elite_antlions_position=populasi_DG;
%iterasi parameter
Max_iter=99;  % Jumlah penrulangan
N=100;    % jumlah populasi

% Initialize the positions of antlions and ants
antlion_position=initialization(N,dimension,ub,lb);
ant_position=initialization(N,dimension,ub,lb);

% Initialize variables to save the position of elite, sorted antlions,
% convergence curve, antlions fitness, and ants fitness
Sorted_antlions=zeros(N,dimension);

Elite_antlion_position=zeros(1,dimension);
Elite_antlion_fitness=inf;

Convergence_curve=zeros(1,Max_iter);

antlions_fitness=zeros(1,N);
ants_fitness=zeros(1,N);

% Calculate the fitness of initial antlions and sort them
for i=1:size(antlion_position,1)
antlions_fitness(1,i)=BaseLoadFlow2(antlion_position(i,:),br,no,brlenght
,PF,R,X,P,Q,MVAb,Vb,adjcb,adjb,oldloss);
end

[sorted_antlion_fitness,sorted_indexes]=sort(antlions_fitness);

for newindex=1:N

Sorted_antlions(newindex,:)=antlion_position(sorted_indexes(newindex),:)
;
end

Elite_antlion_position=Sorted_antlions(1,:);
Elite_antlion_fitness=sorted_antlion_fitness(1);


% Main loop start from the second iteration since the first iteration
% was dedicated to calculating the fitness of antlions
Current_iter=2;
while Current_iter<Max_iter+1

Convergence_curve(1,1)=Elite_antlion_fitness;
```

```
    % This for loop simulate random walks
    for i=1:size(ant_position,1)
        % Select ant lions based on their fitness (the better anlion the
higher chance of catching ant)
        Rolette_index=RouletteWheelSelection(1./sorted_antlion_fitness);
        if Rolette_index==-1
            Rolette_index=1;
        end

        % RA is the random walk around the selected antlion by rolette
wheel
        RA=Random_walk_around_antlion(dimension,Max_iter,lb,ub,
Sorted_antlions(Rolette_index,:),Current_iter);

        % RA is the random walk around the elite (best antlion so far)
        [RE]=Random_walk_around_antlion(dimension,Max_iter,lb,ub,
Elite_antlion_position(1,:),Current_iter);

        ant_position(i,:)= (RA(Current_iter,:)+RE(Current_iter,:))/2; %
Equation (2.13) in the paper
    end

    for i=1:size(ant_position,1)

        % Boundar checking (bring back the antlions of ants inside
search
        % space if they go beyoud the boundaries
        Flag4ub=ant_position(i,:)>ub;
        Flag4lb=ant_position(i,:)<lb;

ant_position(i,:)=(ant_position(i,:).*(~(Flag4ub+Flag4lb)))+ub.*Flag4ub+
lb.*Flag4lb;
ants_fitness(1,i)=BaseLoadFlow2(ant_position(i,:),br,no,brlenght,PF,R,X,
P,Q,MVAb,Vb,adjcb,adjb,oldloss);

    end

    % Update antlion positions and fitnesses based of the ants (if an
ant
    % becomes fitter than an antlion we assume it was cought by the
antlion
    % and the antlion update goes to its position to build the trap)
    double_population=[Sorted_antlions;ant_position];
    double_fitness=[sorted_antlion_fitness ants_fitness];

    [double_fitness_sorted, I]=sort(double_fitness);
    double_sorted_population=double_population(I,:);

    antlions_fitness=double_fitness_sorted(1:N);
    Sorted_antlions=double_sorted_population(1:N,:);
```

```
    % Update the position of elite if any antlinons becomes fitter than
it
    if antlions_fitness(1)<Elite_antlion_fitness
        Elite_antlion_position=Sorted_antlions(1,:);
        Elite_antlion_fitness=antlions_fitness(1);
    end

    % Keep the elite in the population
    Sorted_antlions(1,:)=Elite_antlion_position;
    antlions_fitness(1)=Elite_antlion_fitness;

    % Update the convergence curve
    Convergence_curve(Current_iter)=Elite_antlion_fitness;

    % Display the iteration and best optimum obtained so far
    if mod(Current_iter,50)==0
        display(['At iteration ', num2str(Current_iter), ' the elite
fitness is ', num2str(Elite_antlion_fitness)]);
        if round(Current_iter/100)==Current_iter/100,
        dispElite_antlion_fitness=[round(Elite_antlion_fitness(1))
Elite_antlion_fitness(2)];
         disp(' ');
         disp(['Iteration: ',num2str(Current_iter)]);
         disp(['Best solution: ',num2str(dispElite_antlion_fitness)]);
         disp(['Elite_antlion_fitness:
',num2str(Elite_antlion_position)]);
        end
    end
    %Convergence_curve(Current_iter+1,1)=Elite_antlion_fitness;
    Current_iter=Current_iter+1;
end

% Output/display
dispElite_antlion_position=[round(Elite_antlion_position(1))
(Elite_antlion_position(2))];
disp(' ');
disp(['Total number of evaluations: ',num2str(Max_iter*N)]);
disp(['Best solution= ',num2str(dispElite_antlion_position)]);
disp(['Elite_antlion_fitness: ',num2str(Elite_antlion_fitness)]);

% Run Load Flow with Optimal DG size and Placement

nobus=round(Elite_antlion_position(1,1));
total_unit_PV_based_DG1=Elite_antlion_position(1,2)/PV(nobus,2);

total_PV_based_DG=ceil(total_unit_PV_based_DG1);
vkec=13;

plot(Voltage,'--ks','LineWidth',2,...
                'MarkerEdgeColor','b',...
                'MarkerFaceColor','b',...
                'MarkerSize',3);
                 legend('After DG','Before DG',1);
```

```
                grid on;

figure
plot(Convergence_curve)
set(gca,'XTick',0:10:Max_iter)
title('Convergence curve');
xlabel('Number of iteration');
ylabel('Rugi Daya (kW)');
```

Analisis ALiran Daya Backward/Forward Sweep

```
m=load('bus33.m');
l=load('branch33.m');

%m=xlsread('bus69.xls');
%l=xlsread('branch69.xls');

br=length(l); % Number of branch
no=length(m); % Number of node

% Base values
MVAb=100;                   % MVA base
KVb=20;                  % KV base
Zb=(KVb^2)/MVAb;            % Z base
LF=1;                       % Load factor

% Per unit values
% R = Resistance in p.u.
% X = Reactance in p.u.
% P = Active power in p.u.
% Q = Reactive power in p.u.

R(:,1)=(l(:,4))/Zb;
X(:,1)=(l(:,5))/Zb;

P(:,1)=(LF*(m(:,2))/(1000*MVAb));
Q(:,1)=(LF*(m(:,3))/(1000*MVAb));

%=========================================================================
==
% Load flow analysis for radial system
C=zeros(br,no);
% C = Radial distribution system matrix
% a = bus from
% b = bus to
for i=1:br
    a=l(i,2);
    b=l(i,3);
    for j=1:no
        if a==j
            C(i,j)=-1; % Diagonal matrix and sending bus (-1)
```

```
        end
        if b==j
            C(i,j)=1; % Upper one diagonal matrix
        end
    end
end
% Identifying end node of branches
e=1;
for i=1:no
    k=0;
    for j=1:br
        if C(j,i)==-1
            k=1;
        end
    end
    if k==0
        endnode(e,1)=i;
        e=e+1;
    end
end
h=length(endnode);
for j=1:h
    e=2;

    f=endnode(j,1);
   % while (f~=1)
   for s=1:no
     if (f~=1)
       k=1;
       for i=1:br
           if ((C(i,f)==1)&&(k==1)) % Find sending bus of endnode, store
to f
                f=i;
                k=2;
           end
       end
       k=1;
       for i=1:no
           if ((C(f,i)==-1)&&(k==1)); % Find node at f with value -
1(Check if node in endnode?) if no, store to f and g(j,e)
                f=i;
                g(j,e)=i; % Matrix g is the main feeder, lateral going
to root bus
                e=e+1;
                k=3;
           end
       end
     end
   end
end
for i=1:h
    g(i,1)=endnode(i,1); % Insert endnode to first column of matrix g
end
g;
```

```
w=length(g(1,:));
for i=1:h
    j=1;
    for k=1:no
        for t=1:w
            if g(i,t)==k
                g(i,t)=g(i,j); % Rearrange root bus in firts column and
endnode at last column of matrix g
                g(i,j)=k;
                j=j+1;
             end
         end
    end
end
g;
for k=1:br
    e=1;
    for i=1:h
        for j=1:w-1
            if (g(i,j)==k)
                if g(i,j+1)~=0
                    adjb(k,e)=g(i,j+1);
                    e=e+1;
                else
                    adjb(k,1)=0;
                end
             end
        end
    end
end
adjb; % Like Transpose of matrix g with first column is feeder and
lateral without root bus at respective feeder or lateral
for i=1:br-1
    for j=h:-1:1
        for k=j:-1:2
            if adjb(i,j)==adjb(i,k-1)
                adjb(i,j)=0;
            end
        end
    end
end
adjb; % Make second column of adjb is transition/curve from feeder to
lateral
x=length(adjb(:,1));
ab=length(adjb(1,:));
for i=1:x
    for j=1:ab
        if adjb(i,j)==0 && j~=ab
            if adjb(i,j+1)~=0
                adjb(i,j)=adjb(i,j+1);
                adjb(i,j+1)=0;
            end
        end
        if adjb(i,j)~=0
```

```
            adjb(i,j)=adjb(i,j)-1;
        end
    end
end
adjb; % insert root bus at adjb sequence
for i=1:x-1
    for j=1:ab
        adjcb(i,j)=adjb(i+1,j); % Like adjb, but root bus is deleted
    end
end
b=length(adjcb);


% Voltage current program
for i=1:no
    Vb(i,1)=1;
end

for s=1:10
for i=1:no
    nlc(i,1)=conj(complex(P(i,1),Q(i,1)))/(Vb(i,1)); % Find Ibr with
root bus I=S*/V (At each branch)
end
nlc;
for i=1:br
    Ibr(i,1)=nlc(i+1,1); % Deleting root bus, and add to "Ibr"
end
Ibr;
xy=length(adjcb(1,:));
for i=br-1:-1:1
    for k=1:xy
        if adjcb(i,k)~=0
            u=adjcb(i,k);
            %Ibr(i,1)=nlc(i+1,1)+Ibr(k,1);
            Ibr(i,1)=Ibr(i,1)+Ibr(u,1); % Backward sweep, increment of I
from end branch
        end
    end
end
Ibr; % True value of I at respectively branch (After increment)
for i=2:no
      g=0;
      for a=1:b
          if xy>1
            if adjcb(a,2)==i-1
                u=adjcb(a,1);
                Vb(i,1)=((Vb(u,1))-((Ibr(i-1,1))*(complex((R(i-
1,1)),X(i-1,1))))); % Backward sweep of voltage with first lateral
connection
                g=1;
            end
            if adjcb(a,3)==i-1
                u=adjcb(a,1);
```

```
                Vb(i,1)=((Vb(u,1))-((Ibr(i-1,1))*(complex((R(i-
1,1)),X(i-1,1))))); % Backward sweep of voltage with second lateral
connection
                g=1;
            end
          end
        end
        if g==0
            Vb(i,1)=((Vb(i-1,1))-((Ibr(i-1,1))*(complex((R(i-1,1)),X(i-
1,1))))); % Backward sweep of voltage drop in normal case
        end
end
s=s+1;
end
nlc;
Ibr;
Vbp=[abs(Vb) angle(Vb)*180/pi];

for i=1:no
    Va(i,2:3)=Vbp(i,1:2);
end
for i=1:no
    Va(i,1)=i;
end
Va;

Ibrp=[abs(Ibr) angle(Ibr)*180/pi];
PL(1,1)=0;
QL(1,1)=0;

% Losses
for f=1:br
    Pl(f,1)=(Ibrp(f,1)^2)*R(f,1);
    Ql(f,1)=X(f,1)*(Ibrp(f,1)^2);
    PL(1,1)=PL(1,1)+Pl(f,1);
    QL(1,1)=QL(1,1)+Ql(f,1);
end

Plosskw=(Pl)*100000;
Qlosskw=(Ql)*100000;
PL=(PL)*100000;
QL=(QL)*100000;

Voltage = Vbp(:,1);
Angle = Vbp(:,2)*(pi/180);


plot(Voltage,'--rs','LineWidth',2,...
                'MarkerEdgeColor','b',...
                'MarkerFaceColor','b',...
                'MarkerSize',4)
set(gca,'XTick',1:1:no);
title('Voltage profile');
```

```
xlabel('Bus');
ylabel('Voltage magnitude (p.u.)');
hold all
```