Washington State University
EE 521 Analysis of Power System
Fall 2021
Project 2
Sparse Power Flow

Name: Chufeng Sun
Professor: Dr. Anjan Bose
11/30/2021

Overview

A sparse matrix is one that has "very few" nonzero elements. A sparse system is one in which its mathematical description gives rise to sparse matrices. Any large system that can be described by coupled nodes may give rise to a sparse matrix if the majority of nodes in the system have very few connections. Many systems in engineering and science result in sparse matrix descriptions. Large systems in which each node is connected to only a handful of other nodes include the mesh points in a finite-element analysis, nodes in an electronic circuit, and the busbars in an electric power network. The all data is based on 14 bus system.

In this report, we would describe the following items:

1. Store sparse matrix
2. Read the elements from the table of sparse matrix
3. Use the Court's algorithm to calculate Q matrix
4. Use LU factorization for Q matrix
5. Get the results

Store sparse matrix

In sparse storage methods, only the nonzero elements of the n*n matrix Jacobian are stored in memory, along with the indexing information needed to traverse the matrix from element to element, Thus each element must be stored with its real vale (aij) and its position indices (row and column) in the matrix. The following is the initial Jacobian matrix:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	32.7276	-5.0470	-5.3461	-5.4277	0	0	0	0	0	0	0	0	0	1.7619	1.7777	0	0	0	
2	-5.0470	10.1665	-5.1195	0	0	0	0	0	0	0	0	0	0	2.0058	0	0	0	0	
3	-5.3461	-5.1195	38.7891	-21.5786	0	-4.8895	0	-1.8555	0	0	0	0	0	10.4173	6.8410	-2.9940e...	-1.1362e...	0	
4	-5.4277	0	-21.5786	36.0508	-4.5555	0	0	0	0	0	0	0	0	6.8410	9.4299	0	0	0	
5	0	0	0	-4.5555	18.8644	0	0	0	0	-4.3807	-3.3983	-6.5299	0	0	-2.7894e...	0	0	0	2.0
6	0	0	-4.8895	0	0	20.1675	-6.1879	-9.0901	0	0	0	0	0	-2.9940e...	0	2.4319e-...	-5.5661e...	0	
7	0	0	0	0	0	-6.1879	6.1879	0	0	0	0	0	0	0	0	-3.7890e...	0	0	
8	0	0	-1.8555	0	0	-9.0901	0	24.3400	-10.3654	0	0	0	-3.0291	-1.1362e...	0	-5.5661e...	5.3261	3.9020	
9	0	0	0	0	0	0	0	-10.3654	14.7683	-4.4029	0	0	0	0	0	0	3.9020	5.7829	1.8
10	0	0	0	0	-4.3807	0	0	0	-4.4029	8.7836	0	0	0	0	0	0	0	1.8809	3.6
11	0	0	0	0	-3.3983	0	0	0	0	0	5.6503	-2.2520	0	0	0	0	0	0	
12	0	0	0	0	-6.5299	0	0	0	0	0	-2.2520	11.0969	-2.3150	0	0	0	0	0	
13	0	0	0	0	0	0	0	-3.0291	0	0	0	-2.3150	5.3440	0	0	0	1.4240	0	
14	1.7619	2.0058	-10.6087	6.8410	0	-2.9940e...	0	-1.1362e...	0	0	0	0	0	38.5192	-21.5786	-4.8895	-1.8555	0	
15	1.7777	0	6.8410	-9.7061	-2.7894e...	0	0	0	0	0	0	0	0	-21.5786	35.0165	0	0	0	
16	0	0	-2.9940e...	0	0	2.4319e-...	-3.7890e...	-5.5661e...	0	0	0	0	0	-4.8895	0	18.9305	-9.0901	0	
17	0	0	-1.1362e...	0	0	-5.5661e...	0	-5.3261	3.9020	0	0	0	1.4240	-1.8555	0	-9.0901	23.8450	-10.3654	
18	0	0	0	0	0	0	0	3.9020	-5.7829	1.8809	0	0	0	0	0	0	-10.3654	14.7683	-4.4
19	0	0	0	0	2.0919	0	0	0	1.8809	-3.9728	0	0	0	0	0	0	0	-4.4029	8.2
20	0	0	0	0	1.6328	0	0	0	0	0	-4.1218	2.4890	0	0	0	0	0	0	
21	0	0	0	0	3.3159	0	0	0	0	0	2.4890	-6.9419	1.1370	0	0	0	0	0	
22	0	0	0	0	0	0	0	1.4240	0	0	0	1.1370	-2.5610	0	0	0	-3.0291	0	

Fig. 1 the initial Jacobian matrix

Then, we used the storage method to represent the Jacobian matrix. The last element of each column and row are linked to a null point. Each object is linked to its adjacent neighbors in the matrix, both by column and by row. In this way, the entire matrix can be traversed in any direction by starting at the first element and following the links until the desired element is reached. If the linked lists of the matrix are not ordered by index, then new elements can be added without transversing the rows or columns. A new element can be inserted in each row or column by inserting them before the first element and updating the first in row and first in column pointers. The following is the table for storaging Jacobian matrix:

	1 Index	2 Value	3 NRow	4 NCol	5 NIR	6 NIC
1	1	32.7276	1	1	2	7
2	2	−5.0470	1	2	3	8
3	3	−5.3461	1	3	4	9
4	4	−5.4277	1	4	5	14
5	5	1.7619	1	14	6	10
6	6	1.7777	1	15	0	18
7	7	−5.0470	2	1	8	11
8	8	10.1665	2	2	9	12
9	9	−5.1195	2	3	10	13
10	10	2.0058	2	14	0	17
11	11	−5.3461	3	1	12	21
12	12	−5.1195	3	2	13	86
13	13	38.7891	3	3	14	22
14	14	−21.5786	3	4	15	23
15	15	−4.8895	3	6	16	37
16	16	−1.8555	3	8	17	39
17	17	10.4173	3	14	18	25
18	18	6.8410	3	15	19	26
19	19	−2.9940e...	3	16	20	41
20	20	−1.1362e...	3	17	0	42
21	21	−5.4277	4	1	22	85
22	22	−21.5786	4	3	23	36
23	23	36.0508	4	4	24	27
24	24	−4.5555	4	5	25	28
25	25	6.8410	4	14	26	40
26	26	9.4299	4	15	0	32
27	27	−4.5555	5	4	28	88
28	28	18.8644	5	5	29	62
29	29	−4.3807	5	10	30	58
30	30	−3.3983	5	11	31	68
31	31	−6.5299	5	12	32	69

	1 FIR	2 FIC
1	1	1
2	7	2
3	11	3
4	21	4
5	27	24
6	36	15
7	43	38
8	46	16
9	56	49
10	62	29
11	67	30
12	72	31
13	79	50
14	85	5
15	95	6
16	101	19
17	108	20
18	118	54
19	124	33
20	129	34
21	134	35
22	141	55

Fig 2. The table of Jacobian matrix

In this table, the total non-zero elements are 146. For this table,

‘NRow’- The row number
‘NCol’- The column number
‘Value’- The value of the element
‘NIR’- The next-in-row links
‘NIC’- The next-in-column
‘FIR’- The first in row
‘FIC’- The first in column

Read the elements from the table of sparse matrix

Function Algorithm

1. Initialize empty arrays for ‘index’, ‘value’, ‘NRow’, ‘NCol’, ‘NIR’, ‘NIC’, ‘FIR’ and ‘FIC’.

2. Using MATLAB default function find( ) to get non-zero elements information from Jacobian matrix , and store as ‘Value’, ‘NRow’, ‘NCol’. Then use find(Value) to get the ‘Index’ pointer array.

3. Using MATLAB default function find() to find the index pointers when ‘NRow’ or ‘NCol’ respectively equals to 1, 2, 3, …, n. Then, store the first elements arrays as ‘FIR’ or ‘FIC’. After that, delete the first elements and store the arrays into ‘NIR’ or ‘NIC’.

4. Construct two tables – ‘TJ’ for ‘index’, ‘value’, ‘NRow’, ‘NCol’, ‘NIR’, ‘NIC’, and ‘TJF’ for ‘FIR’ and ‘FIC’.

Use the Court’s algorithm to calculate Q matrix

Then, we will use scheme 0 to reorder the table.

Scheme 0

1. Calculate the degree of all vertices.

2. Choose the node with the lowest degree. Place in the ordering scheme.

3. In case of a tie, choose the node with the lowest natural ordering.

4. Return to step 2.

We can get the initial degree and updated degree:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
5	3	9	5	8	6	2	9	5	4	4	6	5	9	5	6

Fig 3. The initial degree of Jacobian matrix

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
7	2	10	11	19	20	1	4	9	13	15	18	22	6	12	16

Fig 4. The Tinney 0 degree of Jacobian matrix

Then, the following is the updated table of Jacobian matrix:

	1 New_Index	2 New_Value	3 New_NRow	4 New_NCol	5 New_NIR	6 New_NIC
1	1	6.1879	1	1	2	70
2	2	-6.1879	1	14	3	71
3	3	-3.7890e-16	1	16	4	72
4	4	10.1665	2	2	5	28
5	5	-5.0470	2	7	6	29
6	6	-5.1195	2	19	0	32
7	7	2.0058	2	21	8	33
8	8	8.7836	3	3	9	18
9	9	3.6991	3	5	10	19
10	10	-4.4029	3	9	0	20
11	11	1.8809	3	12	12	21
12	12	-4.3807	3	18	13	17
13	13	5.6503	4	4	14	23
14	14	3.9082	4	6	15	24
15	15	-2.2520	4	15	16	25
16	16	2.4890	4	17	17	26
17	17	-3.3983	4	18	18	22
18	18	-3.9728	5	3	19	40
19	19	8.2104	5	5	20	41
20	20	1.8809	5	9	0	42
21	21	-4.4029	5	12	22	43
22	22	2.0919	5	18	23	27
23	23	-4.1218	6	4	24	77
24	24	5.2056	6	6	25	78
25	25	2.4890	6	15	26	48
26	26	-2.2520	6	17	0	49
27	27	1.6328	6	18	28	37
28	28	-5.0470	7	2	29	107
29	29	32.7276	7	7	30	34
30	30	-5.4277	7	8	31	35
31	31	1.7777	7	11	32	36

1 New_FIR	2 New_FIC
1	1
7	4
11	8
21	13
27	9
36	14
43	5
46	30
56	10
62	46
67	31
72	11
79	47
85	2
95	15
101	3
108	16
118	12
124	6
129	44
134	7
141	45

Fig 5. The updated table of Jacobian matrix

When we use Court's algorithm, we write a function to read the table for finding the linked value. We compared the initial Q matrix with Tinney 0 Q matrix in the first iteration.

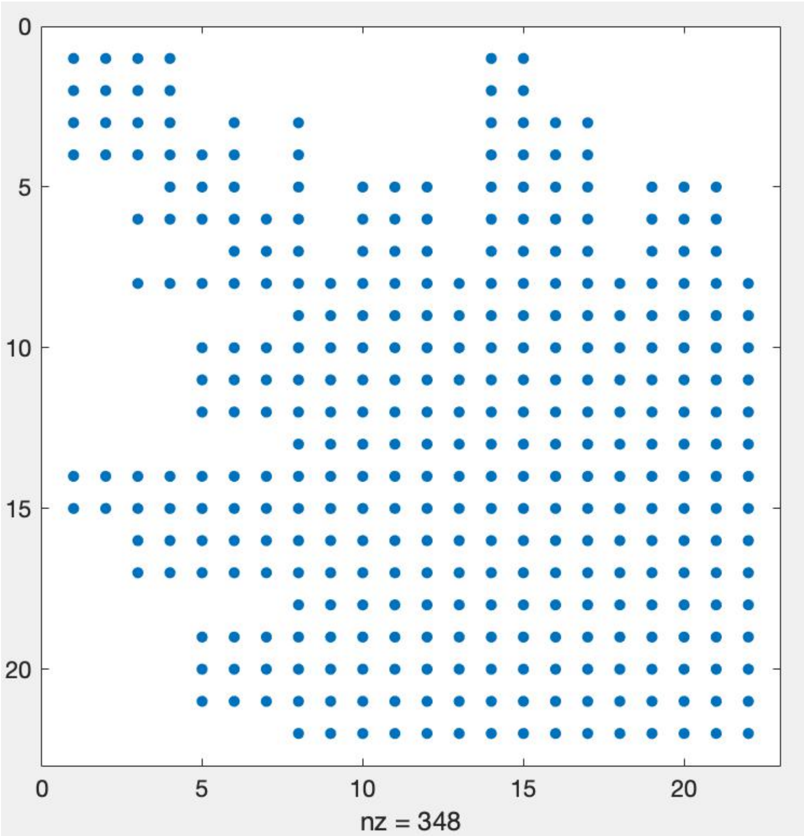

Fig 6. The initial non-zero Q matrix

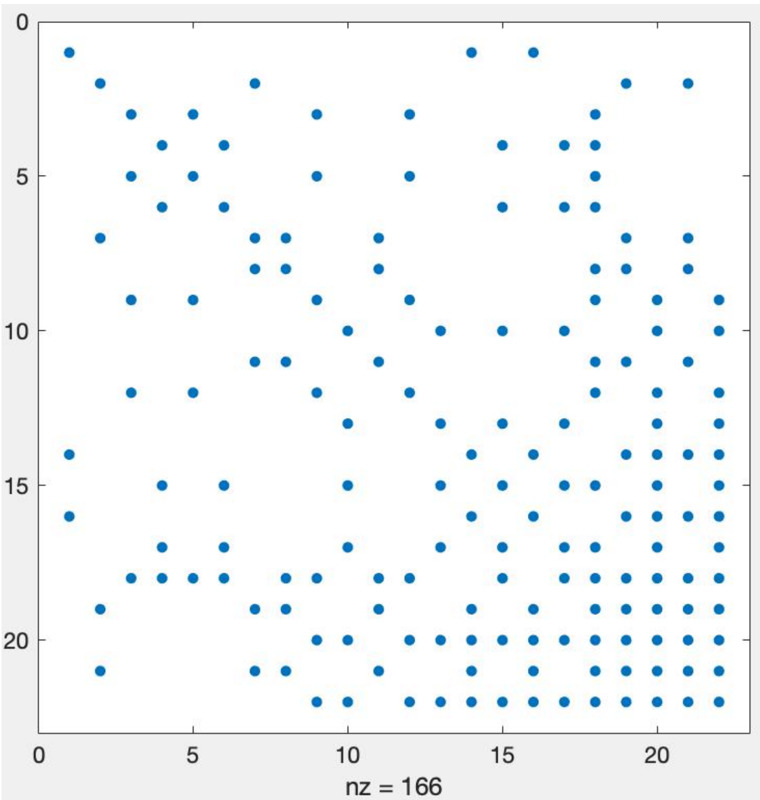

Fig 7. The non-zero Tinney 0 Q matrix

Also, we used the Tinney 1 to get the updated table of Q matrix. The following is the rule of Tinney 1:

1. Calculate the degree of all vertices.

2. Choose the node with the lowest degree. Place in the ordering scheme. Eliminate it and update degrees accordingly.

3. In case of a tie, choose the node with the lowest natural ordering.

4. Return to step 1.

Then, we get the updated table of Jacobian matrix.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	5	3	9	5	8	6	2	9	5	4	4	6	5	9	5	6

Fig 8. The initial degree of Jacobian matrix

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
7	2	1	4	15	10	19	9	18	11	20	12	21	13	22	5

Fig 9. The Tinney 1 degree of Jacobian matrix

We compared the initial Q matrix with Tinney 0 Q matrix in the first iteration.

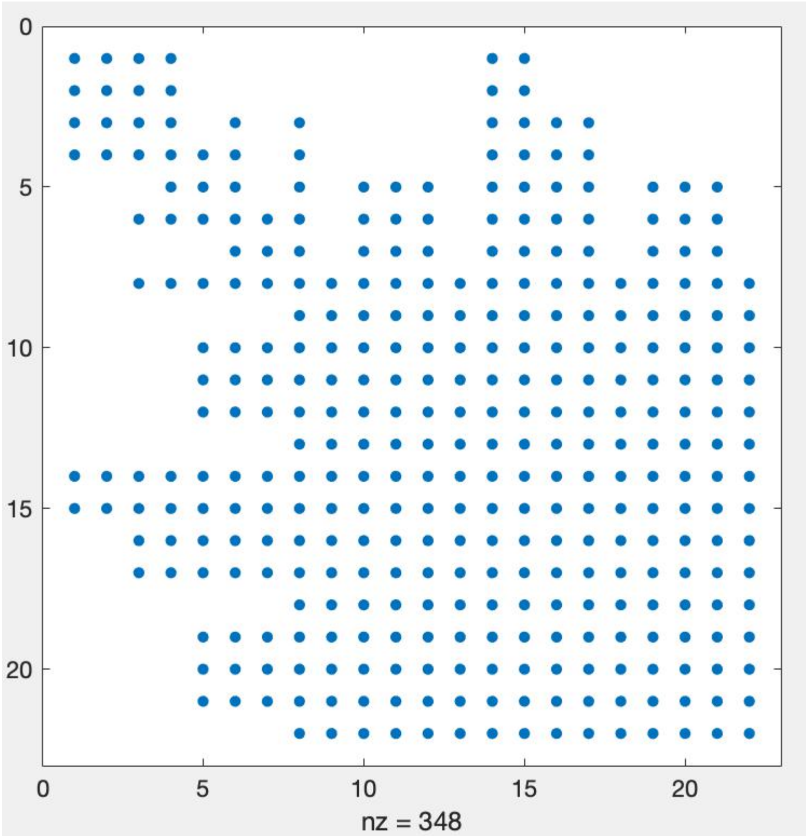

Fig 10. The initial non-zero Q matrix

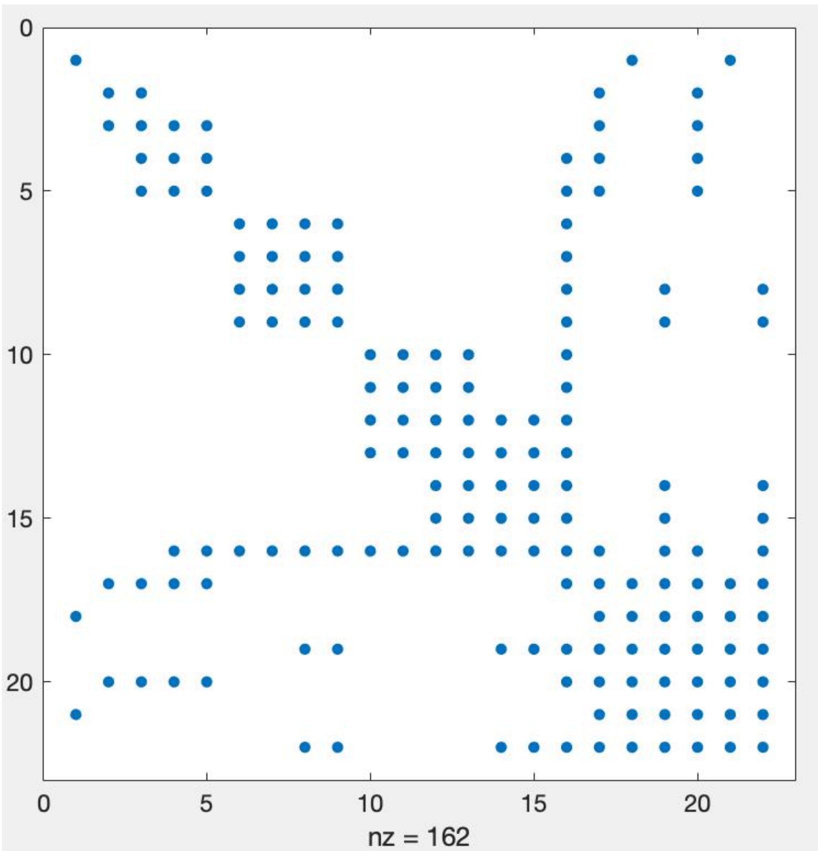

Fig 11. The Tinney 1 non-zero Q matrix

We can see that Tinney 1 has fewer non-zero numbers than Tinney 0, which makes it faster.

Use LU factorization for Q matrix

Then we use LU factorization for Q matrix to get the corresponding value of Y, and then get the value of X through Y.

1
−1.2798e...
−0.0907
0.0116
0.0081
0.0318
0.0298
−0.0131
−2.6267e...
−0.0125
−0.0279
0.0139
0.0015
−0.0185
−2.3061e...
−0.0049
0.0327
0.0325
−0.1891
−0.1028
−0.4136
0.0031
0.0540

Fig 12. The forward substitution

1
−0.5533
−0.3560
−0.6959
−0.7105
0.0429
0.0367
−0.1664
−0.3248
−0.7063
−0.7326
0.0041
0.0322
0.0104
−0.5533
−0.7069
0.0588
0.0346
−0.6588
−0.3626
−0.6559
8.6190e−...
0.0540

Fig 13. The backward substitution

Due to the order is based on Tinney 0, so we need to transform the order into the original order. The following is the code of transform and the original result.

1
−0.1664
−0.3560
−0.3626
−0.3248
−0.6588
−0.5533
−0.5533
−0.6559
−0.7063
−0.6959
−0.7105
−0.7069
−0.7326
8.6190e−...
0.0041
0.0588
0.0540
0.0322
0.0429
0.0367
0.0346
0.0104

Fig 14. The original result for the first iteration

Get the result

Finally, we processed 7 iterations to get the correct result. The following is the final result:

1	1
1.0600	0
1.0450	-0.0850
1.0100	-0.2187
1.0185	-0.1760
1.0204	-0.1494
1.0700	-0.2408
1.0620	-0.2271
1.0900	-0.2271
1.0563	-0.2536
1.0514	-0.2559
1.0572	-0.2504
1.0554	-0.2551
1.0507	-0.2565
1.0360	-0.2719

Fig 15. The final voltage and angle