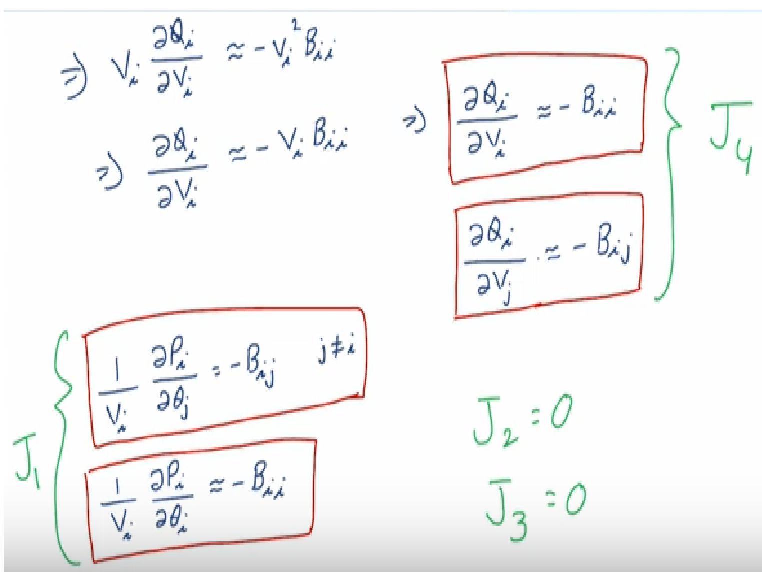
Computer Aided Power System Analysis
Prof. Biswarup Das
Department of Electrical Engineering
Indian Institute of Technology-Roorkee

Lecture - 22
FDLF (Contd..)

Welcome to this lecture of this course on computer aided power system analysis. In the last lecture we have looked into the various elements of this matrices J 1, J 2, J 3 and J 4. We have shown that this matrices J 2 and J 3 are basically the null matrix when we apply some assumptions judiciously. And also we have derived the matrices or rather the elements of this matrices J 1 and J 4.

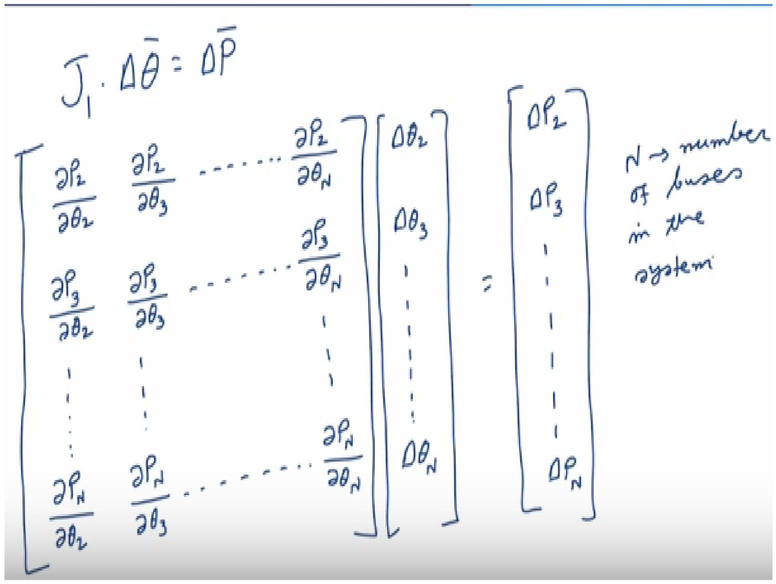
So today we would be again looking into this elements of matrices J 1 and J 4 into more detail and see that how this modified elements can help us to speed up this solution process of the load flow solution. So what we did is that in the last lecture we have shown that this J 2 and J 3 are 0.

(Refer Slide Time: 01:28)



And for J 1 these are the elements and for J 4 these are the elements. So now what we have is that, so we have that we have already shown that for this FDLF

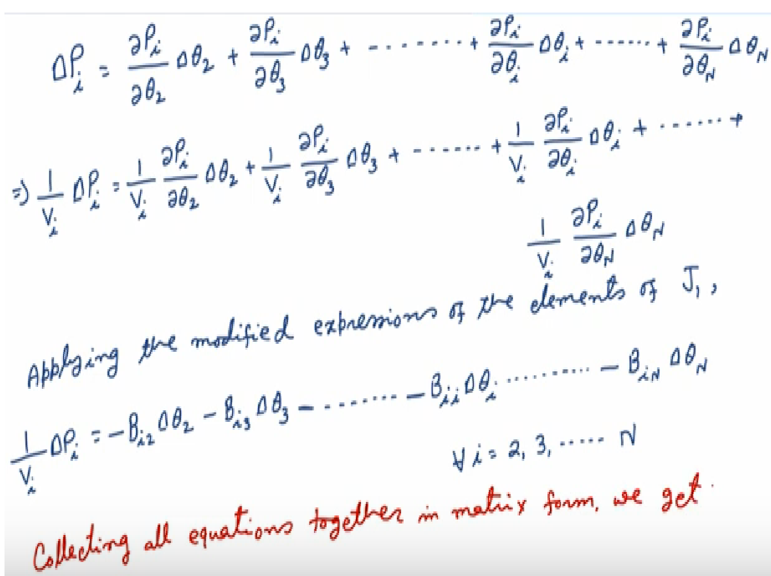
(Refer Slide Time: 01:40)



J 1 into delta theta vector = delta P vector. So if we expand this, so we know that del P 2/del theta 2, del P 2/delta theta 3…del P 2/delta theta N. Then del P 3/delta theta 2, del P 3/del theta 3 then del P 3/del theta N…del P N/del theta 2, del P N/del theta 3…then del P N/del theta N. This is equal to this multiplied by this vector del theta 2 del theta 3…del theta N would be equal to del P 2 del P 3…del P N.

Where N is the number of bus in the system that we have already seen. So N is the number of buses in the system. So now if we do expand this if we now write down the expression of any P i, del P i so what we get?

(Refer Slide Time: 03:54)



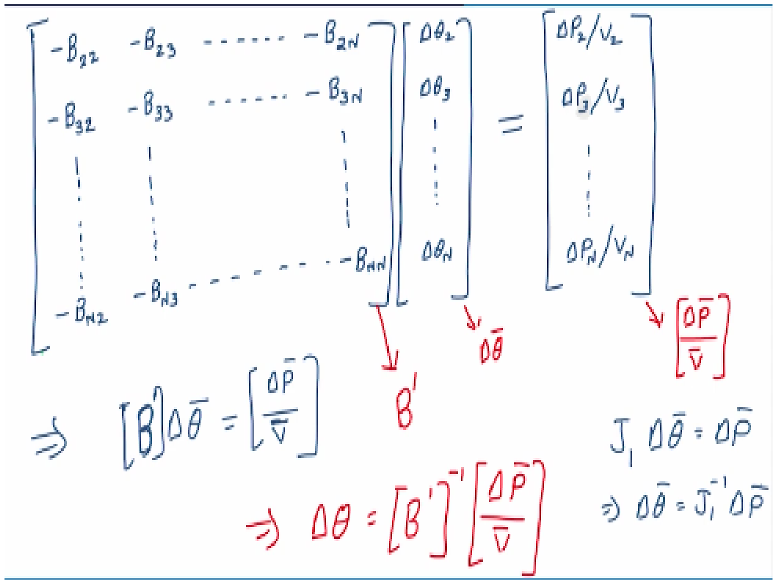
So we get that del P i = del P i/del theta 2 * delta theta 2 + del P i/del theta 3 * delta theta 3 + …del P i/del theta i to delta theta i + …del P i/del theta i * delta N. So this is the expression we get from this matrix if we do expand this matrix equation so then corresponding to the ith row we get this expression. Now what we do is, we do something like this. So if we do if I, so we get something like this.

1/V i del P i/del theta 2 * delta theta 2 + 1/V i del P i/del theta 3 * delta theta 3 + …1/V i del P i/delta theta i * delta theta i + …1/V i del P i/del theta N to delta theta N. So this is the equation we get. Now in this equation we now substitute these relations. So 1/V i del P i/del theta j where j is not equal to i is equal to – B ij. So therefore and for and also we have got that 1/V i * del P i/del theta i = - B ii.

So if we apply this, so then therefore applying those equations, so applying the modified elements expressions of the elements of J 1 what we get? We get 1/V i delta P i = minus it would be B i2 * delta theta 2 – B i3 * delta theta 3 + …actually all of them would be minus. So all of them would be minus. So we should write as minus. So it is minus, this is dot then – B ii * delta theta i. Then … - B iN * delta theta N.

So this is true for all i = 2, 3 to N. So then therefore this we can write down for all i where i varies from 2, 3 to N. So then therefore if we collect all these equations together and then write in the matrix form then what we get? So we write that collecting all equations together in matrix form we get, what we get?

(Refer Slide Time: 08:46)



We get is – B 22 – B 23…- B 2N. Then – B 32 –B 33…-B 3N. And then similarly –B N2 –B N3…-B NN * del theta 2 del theta 3…del theta N. That would be equal to del P 2/V 2. Then del P 3/V 3…del P N/V N. So this is the final relation. Now this expression is very interesting. Because I mean we need to note down one very interesting thing is that, that this matrix in common parlance this is called B dash matrix * delta theta vector = del P/V vector. So this is the so B dash matrix B dash * delta theta = del P/V vector.

So where is what is what. So this is nothing but this our original delta theta vector. This we denote as del P/V vector where this is the del P/V vector where every element is nothing but, where this ith element is nothing but del P i/V i and this matrix is called B dash matrix. And this matrix is a constant matrix. So this matrix is a constant matrix and what is this matrix?

This matrix is nothing but the negative part of the elements of the bus admittance matrix. Now because this matrix is a constant matrix and these elements are nothing but the negative or basically nothing but the negative of the imaginary part of the elements of the bus admittance matrix so then therefore what we need to do is we can only form this matrix only once before even our load flow iteration starts and for all iterations this matrix remains constant.

And then, so then therefore how do we solve delta theta? We actually solve delta theta as for example B dash inverse * delta P/V vector. Now because B dash matrix is a constant matrix, so then therefore its inverse is also a constant matrix. So then therefore we can simply pre-calculate this inverse matrix even before our load flow iteration starts. Then at each and every iteration we have to only update this vector delta P and delta V.

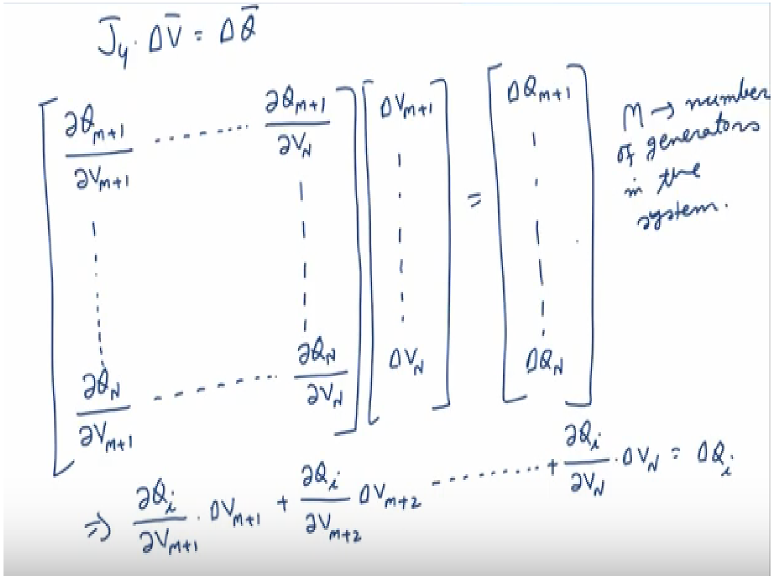
And then we will simply get this correction vector delta theta just by multiplying this matrix which we have already pre-calculated and stored with the updated vector delta P/delta V. Please note in the original even in the decoupled form of equation, what equation we have got? We have got equation as J 1 delta theta = delta P. So then therefore at each iteration in the decoupled form J 1 inverse into delta P.

Now at each iteration, this matrix J 1 needs to be calculated and because this matrix J 1 needs to be always calculated at each and every iteration, so then therefore this matrix J 1 will not be constant because this matrix will not be a constant so then therefore its inverse also has to be calculated at each and every iteration. And calculating inverse of a matrix is highly computationally intensive.

So then therefore even in the original decoupled form because we have to invert this matrix J 1 at each and every iteration, still this is a computational intensive job. But here in this case what happens that we have simply reduced everything to a ultimately constant matrix B dash which is a constant matrix and because it is a constant matrix we can simply pre-calculate its inverse and simply store it.

So then therefore at each and every iteration we are simply avoiding the necessity of inverting any matrix. So then therefore because we do not have to invert any matrix at each and every iteration so then therefore every iteration would be quite fast. Now let us go into the second part, for J 4. Now what was J 4?

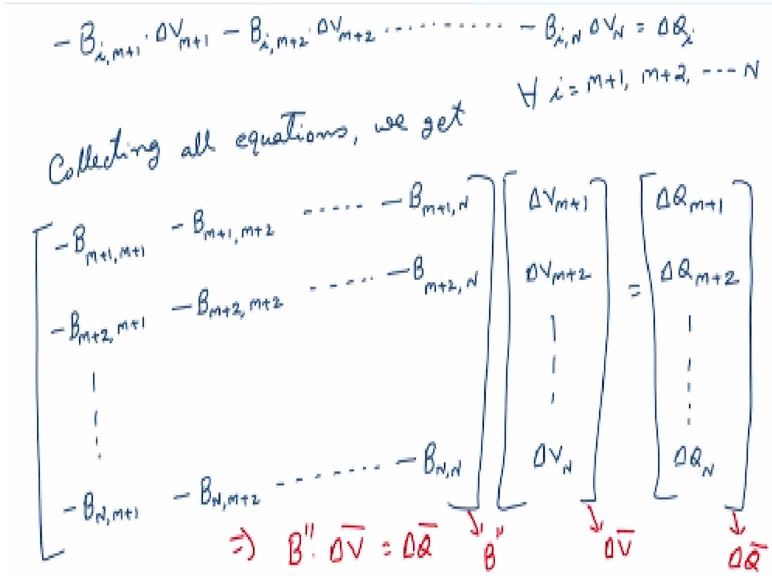
(Refer Slide Time: 15:02)



In the case of J4, for J 4 we have that J 4 * delta V vector = delta Q vector. So it is del Q M + 1/del V M + 1…del Q M +1/del V N and then …del Q N/del V M + 1 and then …del Q N/del V N. It is del V M + 1 then …del V N. It is = del Q M + 1…del Q N. What is M? M is nothing but the number of generators in the system. So again just write it is the number of generators in the system.

So then therefore if we do again expand this ith row so then what do I get? We get that del Q i/del V M + 1 * delta V M + 1 + del Q i/del V M + 2 * delta V M + 2…+ del Q i/del V N * delta V N = delta Q i. So if we do expand as before the ith row for this matrix equation we get this. Now we apply, we apply this relation. We apply this relation for this J 4 matrix. Please note here that here in this case J is not equal to i. So then therefore we apply this relation here. So then what do I get?

(Refer Slide Time: 18:02)

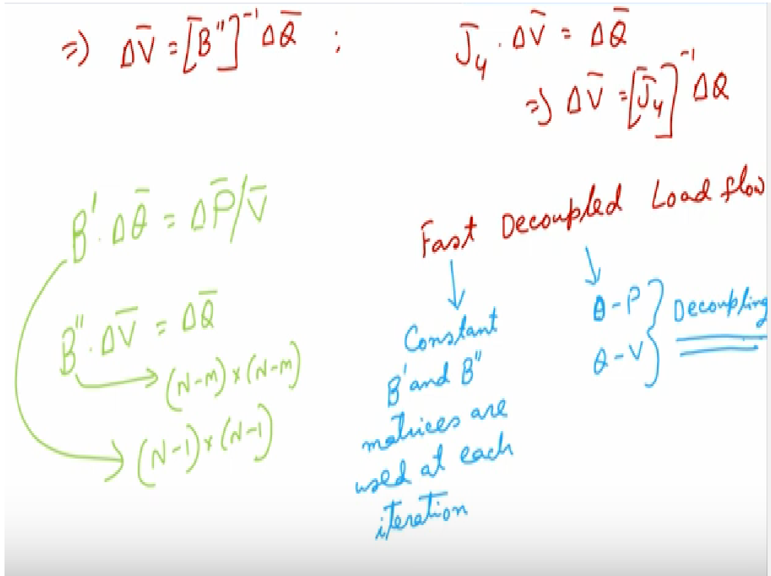


I get –B i M + 1 * delta V M + 1 – B i, M + 2 * delta V M + 2. Then … -B iN * delta V N = delta Q i. What we have done? We have simply substituted these quantities, these expressions into this. So this is true for all i varying from M + 1, M + 2 to N. So then again if we do collecting all the equations, collecting all equations we get, what we get?

We get –B M + 1, M + 1 – B M + 1, M + 2… -B M + 1, N. The –B M + 2, M + 1 –B M + 2, M + 2. Then …-B M + 2, N. Then –B N, M + 1. Then –B N, M + 2. Then …-B N, N. So this = delta V M + 1, delta V M + 2, delta V N. That is = delta Q M + 1, delta Q M + 2 and then delta Q N. So that is the equation. So in power system parlance we write this equation as, this equation as B double dash matrix * delta V vector = delta Q vector.

So this is the B double dash matrix. This is the matrix. This is the delta V vector. This is the delta Q vector. Again, this B double dash matrix is again it is a constant matrix. It is only constituting of the negative of the imaginary part of the elements of the bus admittance matrix. So **because so** then therefore from this equation we can write down

(Refer Slide Time: 21:40)



That delta V = B double dash inverse * delta Q. Now because B double dash matrix is a constant matrix so then therefore we can always pre-calculate this constant matrix even before our algorithm iteration starts and we can also take the inverse of this constant matrix and pre-store it. So then therefore at each and every iteration for updating this vector delta V we really need not perform any kind of matrix inversion.

We can simply use this pre-stored invert matrix which is nothing but the inverse of this B double dash matrix and then simply multiply this inverse matrix with this most updated delta Q matrix. Please note that in the original decoupled form J 4 * delta V = delta Q and here delta V would have been J 4 inverse delta Q. Now because this matrix J 4 has to be re-calculated at each and every iteration.

So then therefore this matrix J 4 will not be a constant, it will be a, it will actually change from iteration to iteration so then therefore we have to take the inverse of this matrix J 4 at each and every iteration and as a result as we know that this inversion process of a matrix is always a very time consuming process, so then therefore here also lot of computation is necessary.

But in this, because this B double dash matrix is a constant matrix so then we can simply pre-calculate its inverse and store it and then use this stored inverse matrix for each and every iteration. So then therefore at each and every iteration we avoid the necessity of inverting any matrix thereby making this each and every iteration quite fast. So then therefore we write this final equations and this final equations are that B dash * delta theta = delta P and B double dash * delta V.

Sorry this is delta P/V vector. B double dash * delta V = delta Q vector. So B double dash matrix it has got a size of (N – N) * (N – 1). B double dash matrix has got a size of (N – M) * (N – M). But both these matrices are constant matrix. So then therefore at each and every iteration to update delta theta and delta V what we have to do?

We have to simply multiply the inverse of this B dash and B double dash matrices which are already pre-stored with the most updated value of delta P/V vector and delta Q/V vectors thereby making this entire load flow algorithm quite fast. That is precisely why it is called fast decouple load flow solution. It is fast because, it is fast decoupled load flow.

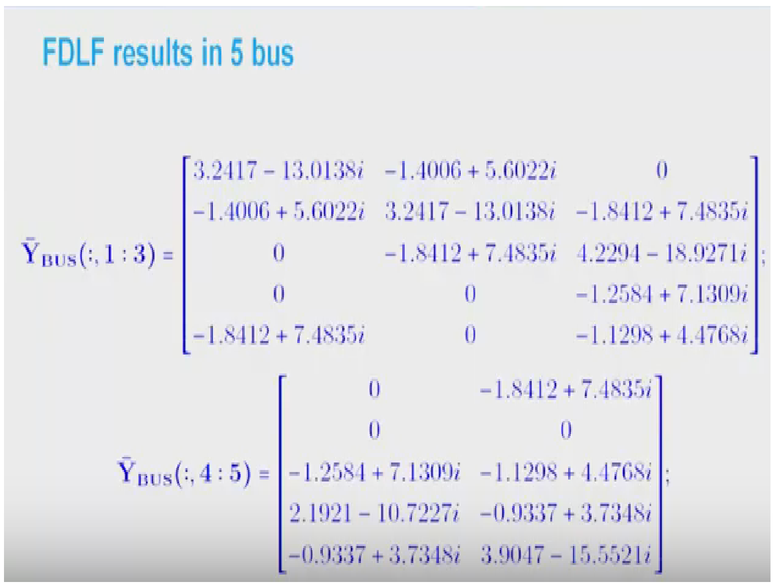
It is fast because constant B dash and B double dash matrices are used at each and every iteration, at each iteration and it is decoupled because it is following theta-P and Q-V decoupling because of theta P and Q-V decoupling. So that is why it is called fast decoupled load flow method. So then therefore in the algorithm what we do? And the algorithm is very simple that we first read the bus data and etc.

And then we form this B dash and B double matrices. Then we take their inverse and we pre-store this inverse matrices. After we pre-store this inverse matrices then we simply proceed in the same way as we do for Newton – Raphson polar method that first we check for all this reactive power violations of all these generators. If any generator violates the reactive power limit, so then that particular bus is changed from PV to PQ.

So accordingly this dimensions of delta Q vector would be changed and this dimension of delta V vector would be changed and accordingly this dimension as well as the element of B double dash matrix will also change. I mean this is the only modification. In case there is no violation is there so then therefore this same B dash and B double dash matrices can be used.

So then utilizing that we update delta theta and delta V and then keep on doing till this convergence is achieved. And we have already discussed how to check for this convergence. So then therefore we are not repeating it here. So now let us look at a very simple example that same example.

(Refer Slide Time: 27:46)



So in this example what we have that we are using the same 5 bus system which whatever we have already used. So this is the Y- Bus matrix. So when I am saying that it is Y-Bus row 1:3 so in the first part only the first 3 columns are shown. And in the second part at the bottom last 2 columns are shown. And we have got already 5 rows here. We can see here.

So then therefore actually this Y- Bus matrix is nothing but a 5 * 5 matrix which it should be because we have got a 5 bus system and we know that the dimension of a Y- Bus matrix is always equal to the number of bus. So from this Y- Bus matrix we form this B dash and B double dash matrix.

(Refer Slide Time: 28:34)

FDLF results in 5 bus (contd..)

$$[B'] = \begin{bmatrix} 13.0138 & -7.4835 & 0 & 0 \\ -7.4835 & 18.9271 & -7.1309 & -4.4768 \\ 0 & -7.1309 & 10.7227 & -3.7348 \\ 0 & -4.4768 & -3.7348 & 15.5521 \end{bmatrix};$$

$$[B''] = \begin{bmatrix} 10.7227 & -3.7348 \\ -3.7348 & 15.5521 \end{bmatrix};$$

This is nothing but the negative of these elements. So then for example for this 2, 2 you see you have got this imaginary part is -13.0138. So then therefore the first term is +13.0138. Similarly, all the other terms have been calculated. Please note that in this case this I meanwhile, whenever we are actually forming this B dash matrix, our indices will go from 2 to 5. So then therefore we have to only concentrate on this part.

That is from the second row to fifth row as well as from second column to fifth column. So we have to take that part. So accordingly we have formed B dash matrix and here also we have formed B double dash matrix because we have got 3 generators in this system. Bus 4 and bus 5 are nothing but the PQ bus. So then therefore because as there are only 2 PQ buses, so then therefore this dimension of this B double dash matrix would be nothing but 2 * 2.

And but this same way that so then therefore this matrices would be this, this, this and this with this negative sign and that is precisely what we have got. For example here it is, it is here -10.7227. So in B double dash matrix it will be +10.7227 and similarly other elements are there. And with this then we follow this algorithm.

(Refer Slide Time: 30:10)

Final Results of the 5 bus system with FDLF

Bus no.	Without generator Q limit				With generator Q limit			
	$\lvert V \rvert$ (p.u)	θ (deg)	P_{inj} (p.u)	Q_{inj} (p.u)	$\lvert V \rvert$ (p.u)	θ (deg)	P_{inj} (p.u)	Q_{inj} (p.u)
1	1.0	0	0.56743	0.26505	1.0	0	0.56985	0.34069
2	1.0	1.65757	0.5	-0.18519	1.0	1.69742	0.5	-0.04522
3	1.0	-0.91206	1.0	0.68875	0.98219	-0.63507	1.0	0.49668
4	0.90594	-8.35088	-1.15	-0.6	0.88888	-8.35938	-1.15	-0.6
5	0.94397	-5.02735	-0.85	-0.4	0.93428	-4.9861	-0.85	-0.4
	Total iteration = 19				Total iteration = 20			

And then we have got his result. And if you compare this results with the results obtained earlier with NRLF polar and NRLF rectangular we will find that the results are identically same. Only difference is that this number of iteration has increased. You may wonder that we have been saying that it is quite fast but then here it appears that the number of iterations is still more as compared to the Newton – Raphson polar coordinate.

It is true that this number of iteration is still more as compared to this Newton – Raphson polar coordinate but the time taken for each and every iteration is much less as compared to that taken by NRLF polar technique because in this FDLF technique we do not have take, we simply do not have to compute the matrix inverse at each and every iteration. So then therefore as we have discussed that each and every iteration this process is very fast because we do not have to calculate this inverse of the matrix.

On the other hand in the NRLF polar method we have to calculate the inverse of this entire matrix that is J 1, J 2, J 3, J 4, that is the entire Jacobian matrix. So it is a big matrix especially if my system is becoming larger and larger. So at each and every iteration we have to take the inverse of a very big matrix. So then therefore it will make the iteration, each and every iteration quite slow.

So then therefore as compared to this NRLF polar, although the number of iterations taken by the FDLF is more but the time taken per each iteration by, taken by FDLF is much less thereby making it quite fast. So that is the end of it. So with this we complete the discussion of AC load flow. So from next lecture onwards we would be looking into the other aspects of this course. Thank you.