Power Network Analysis

Dr. Abheejeet Mohapatra

Department of Electrical Engineering

IIT Kanpur

Week-08

Lecture-38

Hello everyone, welcome to lecture 3 of week 8 of the course Power Network Analysis, in which we continue our discussion on power flow analysis. And in this discussion, we will take up a new technique for solving the power flow equations, which is the Newton-Raphson method, and specifically, this method will have two variants. The first variant is where the voltage phasors are expressed in terms of polar coordinates; that means voltage at bus i has a magnitude as well as a phase angle theta i, which is being measured with respect to a reference. And to be specific, this reference turns out to be the bus also known as the slack bus, swing bus, or angle reference bus, for which the phase angle is usually chosen as 0 degrees or radians for the sake of convenience, and all these phase angles are being measured with respect to this slack bus. So we will assume that our voltages are of the version or that they are in polar coordinates, and accordingly, the power flow equations would turn out to be transcendental equations, which we have seen earlier. How the Newton-Raphson method solves these transcendental equations is what we will see in today's discussion.

In the previous lecture, we extensively discussed the Gauss-Seidel method and how it can solve power flow equations, irrespective of whether voltages are in polar coordinates or rectangular coordinates. We also understood that the Gauss-Seidel method has a slow, or I would say linear, convergence characteristic, which we discussed in length. It might also have convergence issues, specifically for systems or cases that are highly radial, sparse, or for networks that have a non-diagonal dominant Y bus, the bus admittance matrix Y bus, and it also has issues in cases where there is long series compensation, and so on. So, we have discussed all of that in the Gauss-Seidel method.

The major issue is that it has slow and linear convergence, so for a large power network, the time taken to solve the power flow equations in Gauss-Seidel would be much higher. So, is there a better technique to reduce this time? Yes, that is where the Newton-Raphson technique comes in. This slide basically enumerates the discussion that we had in the previous lecture on the previous slide. The Gauss-Seidel method, unlike Newton-

Raphson, is recursive; that means in every step, the voltages of a particular bus are updated, and the updated voltages are used in the next bus update. That is where the recursion part comes in, whereas Newton-Raphson, or the techniques to follow, are iterative techniques where all bus voltages are obtained in a single go.

It has slow linear convergence, has convergence issues when the system is heavily loaded, and a voltage stability problem might arise. For distribution systems where the R by X ratio is high, Gauss-Seidel may have difficulty converging. For highly radial networks with non-dominant diagonal or non-diagonal dominant fibers, which become highly sparse, Ybus will also have difficulties. Basically, why would this non-diagonal dominant Ybus create an issue? If you recollect our Gauss-Seidel method, we had 1 over Yii on the right-hand side term of the Gauss-Seidel update technique. So if Yii becomes a very small number, which is the diagonal loss of diagonal dominance, the ratio of 1 or the value of 1 by Yii may become erroneous, may become difficult to evaluate, and that is where, for non-diagonal dominant networks, this might create a problem.

Again, when the series compensation is high, the particular diagonally dominant value would be very negative. So, overall line reactance may become negative. In that case, the Gauss-Seidel method may also have issues. These problems that exist here might also be true. In fact, they might not be.

They are also true for Newton-Raphson technique. The only motivation for understanding or learning the Newton-Raphson technique is to get rid of this aspect; that is, the Newton-Raphson technique tends to have quadratic convergence, and that is the reason why we would understand Newton-Raphson. So if we go into the Newton-Raphson technique, we will start with a very simple discussion and then complicate the discussion as and when required. Essentially, all our equations that we have seen in power flow analysis are functions of some variables, basically the voltage magnitudes and phase angles in polar coordinates or the real and imaginary parts in rectangular coordinates. So if we consider these variables to be, let's say, x and treat our functions or equations as some function of x, then essentially we are trying to find some value of x for which the power flow equations would satisfy a particular value.

Those particular values essentially refer to certain injections. We can always transform or shift those injections on the left-hand side of the equation or at the right-hand side of the equation as needed. And eventually, at one end, we will have a value of 0. So, essentially, the purpose is that we want to find a value x star, which is the desired solution. So, a set of functions, or let us say for a simpler case, a single function f of x, should take a value of 0.

So, essentially, we want to find x star for which f becomes 0. So we'll start with a single function. So f of x is a single function, x is also a single variable, and we are in pursuit of that x for which f of x becomes specifically 0. One can always say, "Oh, why only 0?" Okay, fine. Let's say we choose a constant c, where c is not equal to 0.

Then we can again rewrite this equation as f of x minus c equal to 0, wherein we can transform this into g of x equal to 0, where g of x is f of x minus c. That essentially is what I was trying to say when I meant that power flow equations are being solved for particular injections; those injections can be shifted to a particular side, but essentially we have 0 on the other hand. So, if we want to find this x star for which f equals 0, in recursive or iterative techniques, we often tend to start with a good initial choice. We do not actually know what x star is, but through some experience and some mathematical curve-based analysis, one can always guess what the likely value of x star might be. So we choose that value to be x0, which is the initial estimate, and around this initial estimate, we make use of the Taylor series expansion.

So basically, if we use the Taylor series expansion of f of x at x equal to x0, then it would have a constant term followed by a first derivative term, a second derivative term, and this continues to infinity with as many terms as possible that those function values can exist. And for a Taylor series to be written or applicable, the condition for f of x is that it should first of all be continuous and differentiable. If it is not differentiable, then these derivatives cannot be evaluated, and hence Taylor series cannot be written. If f of x is a linear function, then one can easily estimate that for a linear f of x, the Taylor series expansion will not have terms beyond the first order. If f of x is a quadratic function, then the Taylor series expansion will not have terms beyond the second order because the third order derivative of a quadratic function would become a constant value or zero.

So depending on what order of f of x is, a linear function would only have two terms in its Taylor series expansion. One is the constant term. The second term is the first derivative. Whereas in the quadratic form of f of x , the Taylor series will have three terms: term number one, term number two, and the second-order term. So the Taylor series expansion is given in this fashion:

$$f(x) = f(x^0) + \left.\frac{df}{dx}\right|_{x=x^0} (x - x^0) + \frac{1}{2!}\left.\frac{d^2f}{dx^2}\right|_{x=x^0} (x - x^0)^2 + \cdots$$

and if x^0 is a good initial estimate of the actual x^*, then we can make the assumption that since x^0 is a good estimate of x^*, we can truncate the Taylor series after the first order.

No matter whether it is linear or quadratic, we can truncate the Taylor series expansion after the second, I mean on or after the second order term, and we can approximate the above equation since we want f of x to be 0 . So we can express f of x in terms of the two first-order Taylor series forms, and the other higher-order terms are truncated. As if they are neglected because x^0 is a good initial estimate, and through this approximation, we can get our update of Newton-Raphson, which states that for the next iterative value of x based on the initial choice x^0, x should take this right-hand side value:

$$0 = f(x) \approx f(x^0) + \left.\frac{df}{dx}\right|_{x=x^0} (x - x^0)$$
$$\Rightarrow x = x^0 - \left[\left.\frac{df}{dx}\right|_{x=x^0}\right]^{-1} f(x^0)$$

If we observe on the right-hand side, since x^0 is known, this value is known; f of x is differential, so this value is also known at x^0, and f of x^0 is again a function, so f being a function implies that x^0 is also known. So basically, all terms on the right-hand side are dependent known RHS; they are known dependent terms, and the dependency is on x^0, which is that initial estimate.

So with the new x , we can again repeat this or get this equation until this x value tends to be equal to x^*. And what would happen at x^* ? What we would observe at x^* is that, basically, we want to find x^* for which f of x becomes 0 . So as x approaches x^*, this term here would tend toward 0 , and when this tends toward 0 , basically this entire term becomes 0 . So eventually, as the solution progresses to the actual solution x^*, the values will all converge. The differences would narrow down, and that is the reason why it serves as a good convergence check for Newton-Raphson.

So basically, in terms of iteration, after the kth iteration, I want to find what the next value of x is, which is x^{k+1}, k being the iteration number. I can make use of this Newton-Raphson iterative technique to get the value of x^* for which f is equal to 0 :

$$x^{k+1} = x^k - \left[\left.\frac{df}{dx}\right|_{x=x^k}\right]^{-1} f(x^k)$$

So, as I mentioned, as the difference between x^{k+1} and x^k becomes less than a particular value, which is, let's say, ϵ or tolerance, usually the tolerance is a value close to 0 because these are numerical techniques. So, we cannot expect the ϵ value to be perfectly 0 ; the number of iterations required can become excessively high, and a value close to 0 is good enough. So, usually a value of 0 .

001 times 10 to the power of minus 5 or 10 to the power of minus 6 for ϵ is good enough for the Newton-Raphson technique. And this ϵ check could be on the differences between

the values of x or the function value itself. This double bar infinity norm is basically the infinity norm. which I had explained earlier, which means what the maximum absolute value of the argument present inside this infinity norm is. If x is a scalar, the infinity norm is the same as the value itself.

If x is a vector or set of variables, then the infinity norm would refer to the maximum absolute value of x , if x is a vector, which is the highest magnitude value. For a linear function, the Newton-Raphson method will always converge in one iteration. How is it so? So let's say we assume our f of x to be a linear function, which is Ax minus b ; a could be.
.. Matrix A could be a scalar. So if A is a scalar, x is also a scalar, and b is also a scalar. If A is a matrix, then x becomes a vector and b also becomes a corresponding vector that maps with A into the product of x . We want f of x to be equal to 0 , which means we want to find x for which Ax minus b is equal to 0 . And if A is well-defined, then the straightforward solution for x is A inverse of b .

Where if A is a scalar, then essentially this is b divided by A ; if it is a vector or matrix, then it becomes the inverse of A . So how does the NewtonRaphson method get this solution directly in one iteration? Let's see that. So what we would do in the Newton-Raphson method is start with an initial estimate, let's say. So let me mark that initial estimate of x as x^{in}. And if I apply this iterative update equation on x^{in}, the next iterative update that I would get, so let's say, or maybe I would choose it in this way: let the initial estimate for the linear function f of x be x^k.

Then, as per the Newton-Raphson iterative equation, the next iteration solution, which is x^{k+1}, is equal to x^{k+1} equal to x^k minus the inverse of the first derivative. So let us find, for f(x), that is equal to... Ax minus b for f(x) is equal to ax minus b.

What is the first derivative? The first derivative is A itself, so the value of $\frac{df}{dx}$ at x is equal to x^k, which is the initial estimate. Since A is a constant, it would remain as A . So let us put that: we have $\frac{df}{dx}$ inverse is equal to A inverse multiplied by the function value of f at x^k. The function value of f at x^k for Ax minus b is $Ax^k - b$. So, if we put that over here, Ax minus b , and work it out, then it becomes A inverse times Ax^k plus A inverse of b .

A inverse can exist as per the definition of A itself. So, basically, this becomes $x^k - x^k + A^{-1}b$, which is $A^{-1}b$. x^{k+1} becomes exactly the same as $A^{-1}b$, which is what the function value of x was, and that is the reason why Newton-Raphson for linear functions always converges in one iteration. For non-linear functions, Newton-Raphson typically takes a few iterations; for well-defined functions, this is on the order of a few tens of iterations, depending on the initial choice. If the initial choice x^0 is very far off from x^*,

there might be convergence issues in Newton-Raphson, and the number of iterations required may be much greater depending on the distance between the initial estimate and the actual solution.

If f becomes a vector of functions, then x can also become a vector of functions because for N equations or n variables to be solved, we need at least N equations for one-to-one consistency. So basically, f is an n cross 1 vector with n variables, and similarly, f also becomes an N cross 1 vector so that we have consistency in terms of knowns and unknowns and the number of equations involved. If we use the same iterative update equation for these N unknowns and n equations, then instead of having one single derivative, we now have a Jacobian matrix, which is an N by N matrix. And depending on whether this matrix is well-defined, whether it is differentiable with the function values, and whether the Jacobian matrix is a singular matrix. So depending on all those factors, either this iterative update equation may be applicable or may not be.

So for this equation to be true, it is necessary that J be a non-singular matrix. That means its determinant should not be 0 . If a determinant is 0 , it becomes singular, and J inverse cannot be written. And you may wonder how this J became an N by N matrix. So let's work out the order of the equations involved.

So x^{k+1}, x being an n cross 1 vector, we have an N cross 1 vector here, $k + 1$ refers to the k -th plus 1 iterative step. x^k is also an N cross 1 vector; $f(x^k)$, as shown above, is also an N cross 1 vector. Now, if this equation has to be true and the overall product has to be consistent with the term sitting on the left-hand side and right-hand side, then essentially a matrix which is N cross N , when multiplied with an N cross 1 vector, results in an N cross 1 vector again, and that is the reason why J has to be an N cross N matrix. Now how would that J look in practice? J looks like something of this order, where the rows basically refer to the columns, so if I have to mark the row and column index. So, row indices essentially pertain to particular functions; let's say this is function 1 , then we have function 2 all the way to function n .

If a row refers to the functions, which is what the numerator terms here indicate, then the columns essentially indicate the derivatives with respect to each of those n variables sitting inside x . Basically, the row index pertains to the function numbers, and the column index in j pertains to the variable numbers. And this J is a derivative or a partial derivative matrix. I have intently written "partial derivative" instead of "full derivative" because f1 to fn are all functions of all n variables. And for partial derivatives of multiple variable functions, the derivatives are partial derivatives.

So in every Newton-Raphson iterative equation, the Jacobian matrix is evaluated at the latest value of x , which is shown over here or marked over here, and it needs to be factorized. So as the number of equations goes up, if N goes, let's say, in the order of a

few thousand, the size of the Jacobian also grows. And remember, no matter how technologically fast the available processes are, division is always a burdensome exercise and has always been a burdensome exercise. So basically, factorizing or inverting a large N by N matrix, where N is on the order of a few thousand, is a burdensome task. So the Newton-Raphson method has the lacuna of being computationally burdensome for a large number of equations.
f of x is also to be evaluated at the latest x, and this sort of process continues; the convergence criteria remain the same as we saw for one variable, one function, where convergence in this case would be $x^{k+1} - x^k$, the maximum absolute possible value. When it becomes less than the tolerance, then the convergence has happened, or the maximum possible function value
of all possible n functions. If the maximum absolute value becomes less than the tolerance, then convergence has also occurred. So, coming to the Newton-Raphson power flow for polar functions, as I mentioned, for polar coordinates, we assume our voltages to be in polar form, where each voltage has a magnitude as well as a phase. For the sake of convenience, we number our buses from 1 to N .

This capital N can be different from the capital N sitting over here. Bus 1 is being considered as the slack bus or angle reference bus, although any other bus could be chosen as the slack bus, and the power flow equations that need to be solved are in terms of transcendental forms of voltages, magnitudes, and phase angles, and the number of equations shows what order of equations exists. So, basically, for real power equations, we have all buses except the slack bus, whereas for reactive power equations, it is applicable only to PQ buses. The number of unknowns pertains to phase angles for all buses except the slack bus, and the voltage magnitude is unknown only for PQ buses. For PV buses, voltage magnitudes are specified according to the excitation control available at the synchronous generator connected to the PV bus.

The initial estimate or choice is the same as the Gauss-Seidel method. We call this a flat start initial choice where voltage magnitudes typically have to be close to 1 per unit as per the choice of 1 PU . So this is the initial choice similar to Gauss-Seidel, whereas for PV buses, the magnitude is the same as the specified value. Phase angles are all chosen to be zero. The iterative update equation for PQ power flow equations or PQ -based power flow equations remains the same.

The only function or notation here that is different is that f of x now pertains to power flow equations, where there are two variants of power flow equations. The first variant is for real power equations, and the other variant is for reactive power equations; we have these as our equations. The unknowns are phase angles as well as voltage magnitudes. So if we categorize the knowns and unknowns in terms of these four sets and apply the logic of row and column labeling the way we discussed in slide number seven here, then our

Jacobian has four distinct sub-matrices. One refers to the partial derivative of real power equations with respect to phase angles; the other pertains to the partial derivatives of real power equations with respect to voltage magnitudes of PQ, and so on for $J_{Q\theta}$ and J_{QV}.

The rest of the process of finding the Newton-Raphson solution for polarbased power flow equations remains the same as discussed in previous slides. The process or the function values of these four submatrices. So, essentially, what is the order or size of J ? The size of J will remain as N cross N ; sorry, not N cross N , it would essentially depend on the number of equations available for real power and reactive power. So, if we look at these real power equations, we have N minus 1 real power equations, and we have N_{PQ} reactive power equations. Similarly, if we talk about the unknowns, then we have N minus 1 voltage phase angles, which are the ones shown over here, and we have N_{PQ} unknown voltage magnitudes, which are the ones shown over here.

So essentially, the dimension of the J matrix is $(N-1)+N_{PQ}$, where N_{PQ} is the number of PQ buses plus $\mathrm{N}-1$ and N_{PQ} is the number of reactive power equations. So how would these numbers look for the four submatrices? Let's see that in the next slide. $J_{P\theta}$ submatrix is the partial derivative of real power equations with respect to voltage phase angles; the number of real power equations is N minus 1 , and the number of phase angles is N minus 1. So, $J_{P\theta}$ actually is again a square matrix of dimension N minus 1 by N minus 1, and the elements of these $J_{P\theta}$ matrices, since we know what the function order of P and Q is. We can easily evaluate these function values and use those approximations.

I am not going to spend much time on how these function values have to be obtained because, once the expressions are known with respect to θ and voltage magnitude, it is just the partial derivative of P with respect to a different variety of phase angles. The first is applicable when P_i is being derived with respect to θ_j, where j is not the same number as bus i . So, basically, this pertains to non-diagonal terms of the $J_{P\theta}$ matrix, whereas this term refers to the diagonal entries of the $J_{P\theta}$ submatrix:

$$\frac{\partial P_i}{\partial \theta_j} = -|V|_i|V|_j\{G_{ij}\sin\left(\theta_j-\theta_i\right)+B_{ij}\cos\left(\theta_j-\theta_i\right)\}, j\neq i$$

$$\frac{\partial P_i}{\partial \theta_i} = \sum_{\substack{j=1\\ j\neq i}}^{N} |V|_i|V|_j\{G_{ij}\sin\left(\theta_j-\theta_i\right)+B_{ij}\cos\left(\theta_j-\theta_i\right)\} = -Q_i-|V|_i^2B_{ii}.$$

Similarly, if we look at J_{PV}, it is derivative of real power equations with respect to voltage magnitudes; real power is N minus 1 in number, voltage magnitudes are N_{PQ} in number, so the dimension is N minus 1 cross N_D, where N_D is the same as N_{PQ}, although

the notation has been different. And elements of J_{PV} again pertain to certain off-diagonal and diagonal terms.

Similar is the case for the $J_{Q\theta}$ matrix, which is not a square matrix. So basically, J_{PV} is also not a square matrix because its dimensions are different. $J_{Q\theta}$ is also not a usual square matrix. The orders are different, and similarly, we have the corresponding off-diagonal and diagonal terms. For J_{QV}, it turns out to be a square matrix because it is the derivative of reactive power with respect to voltage magnitude:

$$\frac{\partial Q_i}{\partial |V|_j} = \frac{1}{|V|_j}\frac{\partial P_i}{\partial \theta_j}, j \neq i$$
$$\frac{\partial Q_i}{\partial |V|_i} = \frac{Q_i}{|V|_i} - |V|_i B_{ii}.$$

So, corresponding to diagonal and diagonal terms, they can be obtained in this order. Please spend some time understanding how these derivatives have been obtained. I am sure these are very simple functions, and these can easily be cross-checked and verified. The last part or second-to-last part in understanding the Newton-Raphson application is how to handle PV to PQ switchings and PQ to PV switching, specifically handling Q limits. The notion of handling Q limits remains the same; at every iteration of Newton-Raphson power flow for the latest solution that has been obtained, we evaluate the reactive power injection at every such PV bus.

Once the injection is evaluated, we find the corresponding generation depending on which demand is connected to the PV bus. If this generation is within its limit, nothing needs to be done; the bus remains a PV bus, the voltage magnitude for the PV bus will not change, and it should not become a variable at all. In case this limit is not met, if it is violated, then Q_g is capped at the corresponding limit it is trying to violate. Once Q_g is limited similar to Gauss-Seidel, the corresponding injection becomes a known quantity; it doesn't remain an unknown quantity. So the PV bus now switches to a PQ bus because the voltage control action has been lost.

Voltage magnitude now becomes free; it is no longer fixed. And since it is now free, we have one more state variable. If we have one more state variable, we need one more equation. That equation comes from the converted PV to PQ bus equation. In case there is a PV bus that was converted in the previous iteration to PQ mode, we always check whether it can return to its PV status.

So what we do is reset its voltage magnitude to the specified value. and we evaluate the corresponding reactive power equation injection, the generation. If this generation at the reset value of voltage magnitude is within its limits, then the corresponding PQ bus can be turned back to PV status. The voltage

magnitude is now fixed, the U equation has been removed, and the switching continues. So basically, this is where the most computationally burdensome task comes in because with the change in PV to PQ switching or PQ to PV switching, the dimension of the Jacobian matrix is changing; it either increases by 1 when PV to PQ conversion happens or reduces by 1 when PQ to PV switching happens, and that creates another problem on its own. If we look at the convergence aspect of the Newton-Raphson technique, as we did for the Gauss-Seidel method, I have just written down those equations.

Let us consider a simple function f of x for which we want to find x such that f of x is zero at x equal to x^k, the kth iteration of Newton-Raphson. We can apply this Newton-Raphson Taylor series expansion and see that, according to the update equation in Newton-Raphson, the x^{k+1} update is this function, which can be rewritten in this particular order. So if we substitute this value of f of x^k on the right-hand side over here, what we see is that we have, so this equation becomes $x^k - x^{k+1}$ multiplied by $\frac{df}{dx}$ at x is equal to x^k, followed by $\frac{df}{dx}$ at x is equal to x^k multiplied by $\frac{x-x^{k+1}}{2!}\frac{d^2f}{dx^2}(x-x^k)^2$ and so on. So this has to be zero. If we rearrange this, then $\frac{df}{dx}$, the first derivative, is common in both these terms.

So x^k and x^k get canceled, and hence we are left with $x - x^{k+1}$. The function multiplied by $\frac{df}{dx} + \frac{1}{2!}\frac{d^2f}{dx^2}(x-x^k)^2$ and so on is equal to 0 . If we denote this as the error in the k -th plus 1 iteration, then we have $\frac{1}{2}$ factorial $\frac{d^2f}{dx^2}$. I missed one term here $\frac{df}{dx}$ where e_{k+1} is $x - x + x_{k+1}$.

We have a square where x is $x - x^k$. with other terms equal to 0 . If we put these right-hand side terms over here, we have e_{k+1} is proportional to e_k^2 with some other terms sitting over here, which means that e_{k+1} is proportional to the square of the error in the previous iterative step, which indicates that Newton-Raphson has quadratic convergence and hence it is a little faster than the Gauss-Seidel method. That is all for this discussion. In the next lecture, we will see an example of the application of the NewtonRaphson method for polar coordinates and take up the discussion on how the Newton-Raphson method would work when voltages are in rectangular coordinates. Thank you.