

**Numerical Method in Civil Engineering**  
**Prof. Arghya Deb**  
**Department of Civil Engineering**  
**Indian Institute of Technology, Kharagpur**

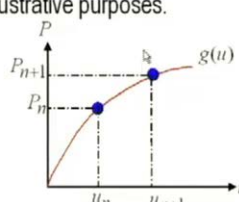
**Lecture - 15**  
**Solving Multi Dimensional Nonlinear Equations – II**

This is the numerical methods in civil engineering, we will continue with our discussion on solving non-linear multi dimensional non-linear equations. We know we discussed last time, I tried to drive home point that the solution of non-linear equations is this multi dimensional non-linear equation, in particular is best understood in a particular physical context.

(Refer Slide Time: 00:30)

### Physical Context

- Knowing  $\mathbf{P}_{n+1}$  we have to find  $\mathbf{u}_{n+1}$  such that they belong to the set of solutions of  $\mathbf{F}(\mathbf{u}, \mathbf{P}) = 0$
- For the purpose of understanding, it often helps if we consider a two-dimensional idealized schematic of the problem where the load vector  $\mathbf{P}$  and the displacement vector  $\mathbf{u}$  are treated as single scalars, purely for illustrative purposes.



Try to if you look at particular equation and look at the particular context that equation arises, is it must easier to understand it than understand in the abstract. We decide to look at the response of our structure, which is in equilibrium than the non-linear response of the structure, which is equilibrium due the action of applied loads. We said that non-linearity might be, because of any reason. May be geometric non-linearity or may be because, of material non-linearity.

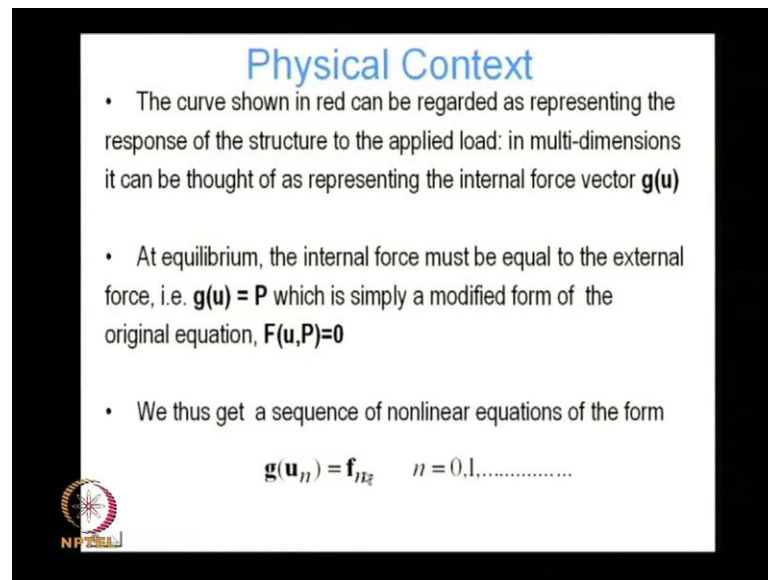
So, basically the problem becomes that we have to solve this non-linear equation, we have to solve it incrementally. So, we have to decide what is driving the solution, what is my input, what is my output. The input in this case we decided is going to be all load

vector. So, we are going to increase the load step by step. Then at every step we are going to iterate. Since, it is a non-linear problem we have to iterate, until we find the displacement solution, which results in an internal force which exactly equilibrates external force. Once it equilibrates converge for that increment, we move to the next increment again we increase the load.

Then again iterate until we reach equilibrium. So, basically knowing  $p_{n+1}$  we have to find  $u_{n+1}$ , such that they belong to the set of solutions of my non-linear equation  $f$ , which is the function of  $u$  and  $p$  where  $u$  and  $p$  are vectors. So, for the purpose of understanding it of help, if we can consider two dimensional idealize schematic of the problem. This is just the schematic it just help to understanding, it is not at all the representation the reality, where the load vector  $p$ . The displacement vector  $u$  are treated as a single scalar purely for the illustrated purpose.


Then we have this blue dot with first blue dot, which represents a the equilibrium state. Where, we know the load  $p_n$ , we know the displace configuration, which results in equilibrium. Then we decide to increase the load from  $p_n$  to  $p_{n+1}$ . We want to find the displace configuration  $u_{n+1}$ , which result in equilibrium. We want to find this blue spot, which is my next equilibrate state, but to go from here to here is not easy you can see. Because, this slope is not a constant, this slope is vary, so if I take the initial step based on the slop here. I am not going to read equilibrium, I have to iterate to I go back to the point on the red curve. The red curve, as I said last time represents the repose of the structure to the plat load.

(Refer Slide Time: 03:46)



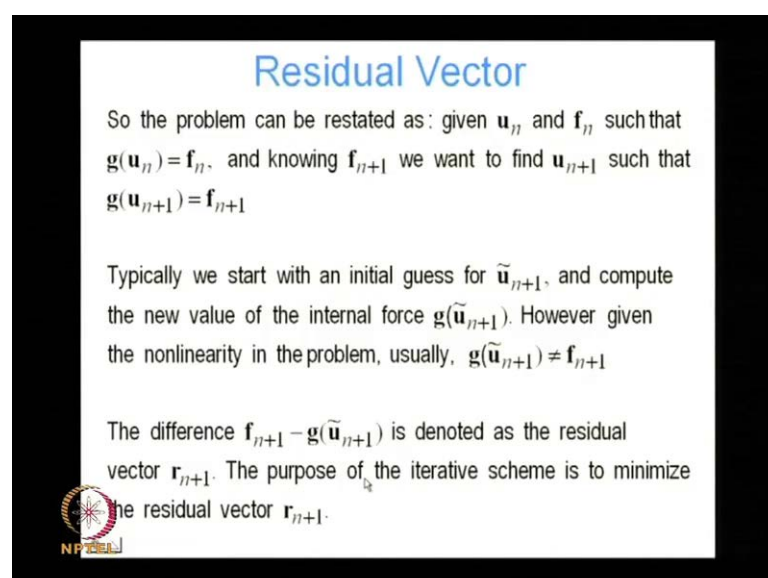
**Physical Context**

- The curve shown in red can be regarded as representing the response of the structure to the applied load: in multi-dimensions it can be thought of as representing the internal force vector  $\mathbf{g}(\mathbf{u})$
- At equilibrium, the internal force must be equal to the external force, i.e.  $\mathbf{g}(\mathbf{u}) = \mathbf{P}$  which is simply a modified form of the original equation,  $\mathbf{F}(\mathbf{u}, \mathbf{P}) = 0$
- We thus get a sequence of nonlinear equations of the form

$$\mathbf{g}(\mathbf{u}_n) = \mathbf{f}_{n+1} \quad n = 0, 1, \dots$$


It can be thought of represent internal force vector  $\mathbf{g}$  of  $\mathbf{u}$ , internal force vector  $\mathbf{g}$ , which is again function of my displace configuration  $\mathbf{u}$  at equilibrium internal force must be equal to the external force. That is  $\mathbf{g}$  of  $\mathbf{u}$  must be equal to  $\mathbf{P}$ , which is simply a modified version of my original equation  $\mathbf{f}$  of  $\mathbf{u}$   $\mathbf{P}$  equal to 0. So, this was my original equation, I am now rewriting it as  $\mathbf{g}$  of  $\mathbf{u}$  equal to  $\mathbf{P}$ . Then we want instead of solving this equation I am going to try to solve this equation in an iterated fashion. If we get the sequence of non-linear equations of the form  $\mathbf{g}$  of  $\mathbf{u}_n$  is equal to  $\mathbf{f}_{n+1}$   $n$  is equal to 0 1 so on for each load steps.

(Refer Slide Time: 04:42)




**Residual Vector**

So the problem can be restated as: given  $\mathbf{u}_n$  and  $\mathbf{f}_n$  such that  $\mathbf{g}(\mathbf{u}_n) = \mathbf{f}_n$ , and knowing  $\mathbf{f}_{n+1}$  we want to find  $\mathbf{u}_{n+1}$  such that  $\mathbf{g}(\mathbf{u}_{n+1}) = \mathbf{f}_{n+1}$

Typically we start with an initial guess for  $\tilde{\mathbf{u}}_{n+1}$ , and compute the new value of the internal force  $\mathbf{g}(\tilde{\mathbf{u}}_{n+1})$ . However given the nonlinearity in the problem, usually,  $\mathbf{g}(\tilde{\mathbf{u}}_{n+1}) \neq \mathbf{f}_{n+1}$

The difference  $\mathbf{f}_{n+1} - \mathbf{g}(\tilde{\mathbf{u}}_{n+1})$  is denoted as the residual vector  $\mathbf{r}_{n+1}$ . The purpose of the iterative scheme is to minimize the residual vector  $\mathbf{r}_{n+1}$ .

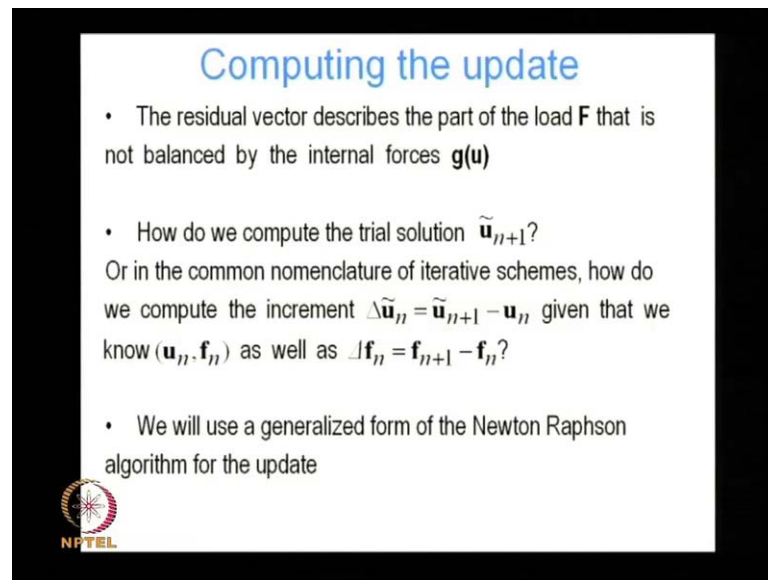


So, given  $u_n$  such that  $g(u_n)$  is equal to  $f_n$ , which correspond to a load equilibrated state and knowing  $f_{n+1}$ . We want to find  $u_{n+1}$  such that  $g(u_{n+1})$  is equal to  $f_{n+1}$ , so to do that we have to start with the initial guess. So, we have to assume some value for my displacement increment. So, starting from here, I assume some value from my displacement increment. How do I assume that value? How do I find that value? Well, I use the tangent here. So, let us see how we do that.

So, you start with the initial guess for  $u_{n+1}$  and given that initial guess we compute the internal force again  $g(u_{n+1})$ . So, I compute my initial guess for my updated configuration. Put my updated configuration in my equation with generate the internal force. I do not want to do the details of those equation, let us assume that. We know as some equation were we put a defect in displacement value, I get back my internal force, let us say that as the black box. So, we get back the internal force, but that internal force is not going to be equal to my external force because of the non-linearity. Because, the slope of the because my  $u_{n+1}$  is not really my converse value.


That is going to result in a disequilibrium in a difference between the external and the internal force. That difference  $f_{n+1} - g(u_{n+1})$  is denoted as the residual. It is the residual vector  $r_{n+1}$  and the purpose of my iterated case is to change  $u_{n+1}$  to  $u_{n+1}^*$ . So, that my residual goes to 0, so that my residual is minimized. So, the purpose of the iterated skin is to minimize the residual vector  $r_{n+1}$ .

(Refer Slide Time: 06:53)



**Computing the update**

- The residual vector describes the part of the load  $\mathbf{F}$  that is not balanced by the internal forces  $\mathbf{g}(\mathbf{u})$
- How do we compute the trial solution  $\tilde{\mathbf{u}}_{n+1}$ ?  
Or in the common nomenclature of iterative schemes, how do we compute the increment  $\Delta\tilde{\mathbf{u}}_n = \tilde{\mathbf{u}}_{n+1} - \mathbf{u}_n$  given that we know  $(\mathbf{u}_n, \mathbf{f}_n)$  as well as  $\Delta\mathbf{f}_n = \mathbf{f}_{n+1} - \mathbf{f}_n$ ?
- We will use a generalized form of the Newton Raphson algorithm for the update



So, what is the physical meaning of the residual vector? Well, the residual vector is nothing but part of the load that is not balanced by the internal forces, that is the out of the balance force. So, that is how we go about it. So, let us take the step back again, how do we compute the trial solution,  $\tilde{\mathbf{u}}$  till the  $n$  plus 1 or in the usual nomenclature of iterative schemes. How do we compute the increment  $\Delta\tilde{\mathbf{u}}$ . So, I am just using some new terminology I am saying  $\tilde{\mathbf{u}}_{n+1}$  the change in the value of  $\tilde{\mathbf{u}}_n$ . That is it is the guess you guess, why do we have tilde.

So, that is my guess to the displacement  $n$  plus 1. My guess minus the value displacement at step  $n$  is the approximate increment  $\tilde{\mathbf{u}}$  and tilde  $n$ . When this become exact the tilde will go away. So, then I will have  $\Delta\tilde{\mathbf{u}}_n$  is equal to  $\tilde{\mathbf{u}}_{n+1} - \tilde{\mathbf{u}}_n$ , but since it is approximate I have the tilde. So, given that  $\tilde{\mathbf{u}}_n$  and  $\mathbf{f}_n$ , as well has  $\Delta\mathbf{f}_n$ , which is equal to  $\mathbf{f}_{n+1} - \mathbf{f}_n$ . This is known the increment in the load is known. So, we will use the generalize form of the Newton Raphson algorithm for the update.

(Refer Slide Time: 08:15)


### Computing the update

- According to Newton Raphson:
 
$$0 = \mathbf{g}(\tilde{\mathbf{u}}_{n+1}) = \mathbf{g}(\mathbf{u}_n) + \mathbf{g}'(\mathbf{u}_n)(\Delta\tilde{\mathbf{u}}_n)$$

$$\therefore \Delta\tilde{\mathbf{u}}_n = -[\mathbf{g}'(\mathbf{u}_n)]^{-1}\mathbf{g}(\mathbf{u}_n)$$

In mechanics nomenclature  $[\mathbf{g}'(\mathbf{u}_n)]$  is the tangent stiffness at the solution point  $(\mathbf{u}_n, \mathbf{f}_n)$  and is denoted as  $\mathbf{K}_n$

- As mentioned earlier, because of the nonlinearity,  $\Delta\tilde{\mathbf{u}}_n$  will not produce an equilibrium solution and thus iterative corrections are necessary to eliminate the residual vector:
 
$$\mathbf{r}_{n+1} = \mathbf{f}_n + \Delta\mathbf{f}_n - \mathbf{g}(\mathbf{u}_n + \Delta\tilde{\mathbf{u}}_n) = \mathbf{f}_n + \Delta\mathbf{f}_n - \mathbf{g}(\tilde{\mathbf{u}}_{n+1})$$



According to Newton Watson according to Newton Watson, we call the Newton Watson. What is all about? It is about linear zing that non-linear equation about the given solutions, I have a certain solution . I know  $u_n f_n$  I approximate might true solution by assuming that the true solution is a straight line, with the slope which is given by the known tangent at  $u_n f_n$ . When that slope, when the straight intersect the x axis the apps is the that gives me my update. So, again are using a Newton Raphson I am saying this is my new iterate right  $g u$  tilde  $n$  plus 1. Since, this is the ordinate is 0, it is point intercut the apps.

So,  $g$  of  $u$  tilde one plus equal to 0 and that is equal to  $g$  of  $u_n$  the starting value plus the tangent times the increment, it is starting value plus the tangent times the increment. So, from this equation I am going to get  $\Delta u$  tilde  $n$  and  $\Delta u_n$  is nothing but  $g$  prime  $u_n$  inverse  $g u_n$ . So, it is just this, so in techniques in lecture  $g$  prime is the tangent stiffness at the solutions point  $u_n f_n$  and is denoted as  $a_n$ . Because, of non-linearity  $\Delta t_n$  will not produce the equilibrated solution. Thus, iterated condition correction are necessary to eliminate the residual vector.

So, my residual vector  $r_n$  plus 1 is my new force, which is  $f_n$  plus  $\Delta f_n$   $f_n$  plus 1 minus  $g$  at  $u$  tilde  $n$ , which is  $u_n$  plus  $\Delta u$  tilde the  $n$ . Then represent this  $u_n$  plus  $\Delta u$  tilde  $n$  s  $u$  till plus  $n$ . So, this is my residual, this is residual I start with my guess and this is

my residual. Then I have to minimize that residual, how am I going to minimize that residual? Again, I will linearism, let us again see what we do.

(Refer Slide Time: 10:43)

### Computing the update


- Starting with iteration counter  $i = 0$ , we obtain the corrections by linearizing the residual about the  $i^{\text{th}}$  estimate of the equilibrium state  $n+1$ , defined by:  $(\mathbf{u}_{n+1}^{(i)}, \mathbf{f}_{n+1})$

$$\mathbf{r}_{n+1}^{(i+1)} = \mathbf{r}_n^{(i)} + \left. \frac{\partial \mathbf{g}}{\partial \mathbf{u}} \right|_{\tilde{\mathbf{u}}_{n+1}^{(i)}} \Delta \tilde{\mathbf{u}}_{n+1}^{(i)} = 0$$

$$\therefore \left. \frac{\partial \mathbf{g}}{\partial \mathbf{u}} \right|_{\tilde{\mathbf{u}}_{n+1}^{(i)}} \Delta \tilde{\mathbf{u}}_{n+1}^{(i)} = -\mathbf{r}_n^{(i)}$$

$$\mathbf{K}_n^{(i)} \Delta \tilde{\mathbf{u}}_{n+1}^{(i)} = -\mathbf{r}_n^{(i)}$$

Finally we update  $\mathbf{u}_{n+1}^{(i+1)} = \mathbf{u}_n + \Delta \tilde{\mathbf{u}}_{n+1}^{(i)}$



So, we iterate we start to iterate we start iterating and starting to iteration count  $i$  is equal to 0. We obtain the correction by linearizing the residual about  $i^{\text{th}}$  estimated of the equilibrium state  $n+1$  defined by  $\mathbf{u}_n + \mathbf{f}_n + \mathbf{f}_{n+1}$ . So, basically lets think of like this. I have the residual at the certain at the certain iterate. Suppose, if I start with the iteration 0 were if my iteration is 0 I computed my value del tilde like this. That represent my iteration 0 and for that iteration 0 I have computed my residual like this, that residual is non zero.

So, then I linearized the residual, so again I expand the residual in tailor series about that iterate. I say that  $\mathbf{r}_{n+1}$  at  $i+1$  at  $i+1$ , I know everything at iterate  $i$ . I know the value of  $\mathbf{u}_i$  iterate  $i$ , I know the value of my internal force at iterate  $i$ . I know the value of my tangent iterate  $i$ . So, then I want to find out what increment I must take displacement to make the residual smaller, that is my new problem. So, to do that I expand the residual I expand the residual as a tailor series about the known state, which is at iterate  $i$ . So, I say that  $\mathbf{r}_{n+1}$  at iterated  $i+1$  this equal to  $\mathbf{r}_n$  at  $i$  plus the gradient the gradient at  $i$  times the change in the displacement, increment.

I am saying that, I assume that at again this is like assumption is that at the new item at the new location, new displacement, new value of iterate my residual is 0. That is the

same assumption everywhere for Newton Raphson. So, at the new iterate my residual is 0. So, that again that gives me the say the increment in the displacement increment. This is like I computed some value of displacement increment. Now, I have to change it iteration, we can think of it has change in the increment or increment, whichever convenient.


So, it is the change in the displacement increment  $\Delta u_{i+1}$ . To do that, I removed this right hand side. Since, the known right everything at iteration  $i$  is known I move it to right hand side. Then I invert this is the matrix this is the vector this is the matrix. So, I inverse this and then I find out my new  $\Delta u_{i+1}$ . Once I find this I update I say that I knew knowing my increment at  $i$  at iteration  $i$ . I can find my increment iteration  $n+1$  using this relationship. So, we are going to continue doing this, until what is the this topic criteria residual becomes extremely small. When the residual become sufficiently small, I know that. I can stop my iteration, but usually we use another stopping criteria.

(Refer Slide Time: 14:24)

### Convergence Criteria

- The iteration process continues until the residual becomes near zero.
- Once the residual has become near zero, equilibrium has been reached for load step  $n$  and a new load increment is applied
- The process of the residual becoming zero is known as convergence. The load increment is supposed to have converged if for small values of  $\epsilon$ ,  $\|\mathbf{r}_{n+1}\| < \epsilon \|\mathbf{f}_{n+1}\|$

Sometimes the criterion of small residual is supplemented by the criterion:  $\|\dot{\Delta \mathbf{u}}_{n+1}\| < \epsilon \|\Delta \mathbf{u}_{n+1}\|$



So, I am going to talk about that. So, we continue iterating this we continue the iteration process until the residual becomes 0. Once the residual become  $n$  is 0 equilibrium has been reached for load step  $n$  and new load increment is applied. So, for increment for that increment I have reached equilibrium, my internal force is equal to my external force my residual is 0 the process of the becoming 0 is known as convergence. So, this process



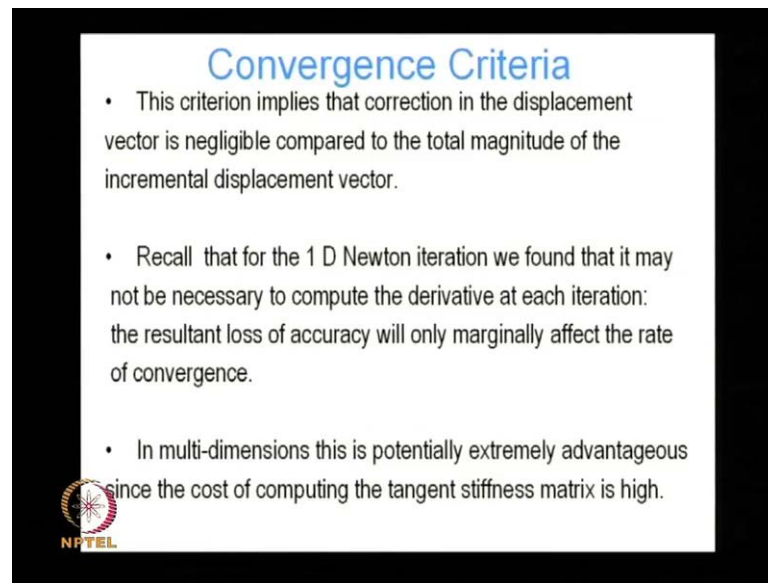
is convergence and the load increment suppose to have converge. Then my residual, that is the residual is a vector.

So, I have to think about some sort of scalar representation of a vector scalar representation of the vector is a norm of that vector. The norm of the residual vector is less than a small fraction of the applied load. So, if  $m f n 1$  is my applied load and my residual as become a epsilon is really a small number. Let us say  $10^{-6}$  for  $10^{-3}$  extend to the power minus 3. Then the residual is 0.001 times my applied load. So, if my applied load is 5 kilo Newton my residual must be 5 less than 5 Newton. So, when the residual is so small, I say that the residual is almost 0, as converge I can move on to next increment.

So, sometimes there is another criteria the small residual criteria is supplement, were another criteria, what is that criteria? That criteria is that the change in the displacement. So, I might, I have my initial guess  $u_{i+1}$  and each at  $i$  at, suppose 0. Then I compute I increment that at iteration 1 2 3 4 and so on. So, at each increment I am adding something to my delta, if I represent by this, if I represent that quantity, which I add by this. I find that the change from my iteration to iteration, my increment is changing the ideally small number. What do I mean, that ideally small number I am saying the norm of the change is 0.001 time the total change till now.


So, from increment, so from increment 5 I, suppose I know the solution at increment 5 I am interested in the solution at increment 6. For that I have to I have taken suppose 40 iteration. That is too big for the timing, let us assume that to go from increment 5 to increment 6, I have taken up till now 40 iterations. My displacement increment to go from increment 5 to increment 6 is suppose 0.1 up till now up till 40 iteration e. Then at the 40 first iteration I find that that displacement increment changes from 0.1 to 0.100 0.00001. So, it changes from 0.1 to 0.1000001, so the change is really small the change of the displacement increment is very small. So, that also tells me that the solution is converge, either the residual criteria or the change in the displacement increment criteria.

(Refer Slide Time: 18:02)



**Convergence Criteria**

- This criterion implies that correction in the displacement vector is negligible compared to the total magnitude of the incremental displacement vector.
- Recall that for the 1 D Newton iteration we found that it may not be necessary to compute the derivative at each iteration: the resultant loss of accuracy will only marginally affect the rate of convergence.
- In multi-dimensions this is potentially extremely advantageous since the cost of computing the tangent stiffness matrix is high.

 NPTEL

This criteria implies the correction in displacement vector is negligible compare to total magnitude of the incremental displacement vector. So, that is it, now let us take another step back. Recall, that for one beam Newton iteration, we found that we may not be necessary to compute the derivative at each iteration. The result in flows of accuracy will only margin the effect of convergence. We saw that if the error in computing the actual function the error computing the function value is more than in the error derivative. Then it is not necessary to compute the derivative accurately.

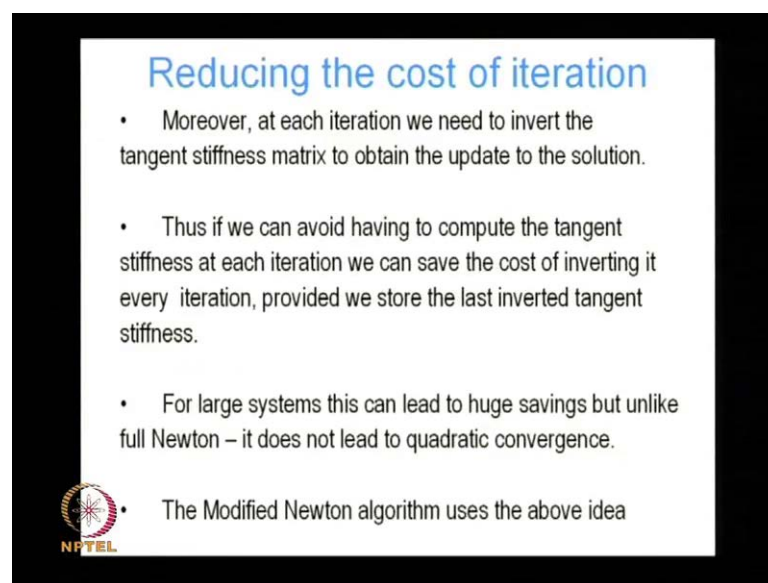
We can save a lot of computational cost by not compute the derivative by taking an approximation of the derivative in multi dimension. This is potentially extremely advantages, since the cost of computing the tangent stiffness is very high. You can see perform an the exact Newton iteration at exact Newton iteration and exact Newton. Then we have to compute not only the tangent matrix at each iteration. We have to invert it also, if I say I give an example. Suppose, I am moving from increment 5 to increment 6 and I have taken 40 increments. So, 40 iterations is move from 5 to 6.

So, what does this mean? That mean at each of those 40 I have to evaluate my tangent stiffness. As soon as I get my it internal force suppose vector at new iteration. I have to evaluate my tangent stiffness and not only have to evaluate the tangent stiffness, I have to inverted it this, in real problem, that particularly a problem, which are common to

industrial application the model is huge, so matrixes of size one hundred thousand by one hundred thousand and not common.


So, if I have to form a matrix size 100000 by 100000, that is it has million terms. Then I have to invert that matrix 40 time to go from one increment to another. So, computational cost is dependent, so in case instead of if we can if instead of evaluating the tangent stiffness every iteration, if we can find an alternative that would save the computational cost significantly.

(Refer Slide Time: 20:42)



**Reducing the cost of iteration**

- Moreover, at each iteration we need to invert the tangent stiffness matrix to obtain the update to the solution.
- Thus if we can avoid having to compute the tangent stiffness at each iteration we can save the cost of inverting it every iteration, provided we store the last inverted tangent stiffness.
- For large systems this can lead to huge savings but unlike full Newton – it does not lead to quadratic convergence.
- The Modified Newton algorithm uses the above idea

 NPTEL

Moreover, at it iteration we need to inverse the tangent stiffness matrix to update the solution, thus if we can avoid having to compute the tangent stiffness at each iteration. We can save the cost of inverting it very iteration provided it showed the last inverted. That is tangent stiffness, if I say that at a certain iteration. I am not going to compute do my Newton iteration. Using my exact tangent stiffness, I am going to use the stored tangent stiffness. So, I computed the tangent stiffness and previous iteration. Hence is that, I do not have to use the exact tangent.

I am going to use that to solve my to update to calculate my increment, but then I meet probably need to store the inverse also. That is always better, if I store the inverse I can directly use the inverse for come substitution. Calculate my update to the increment, for last this can be huge savings, but unlike Newton it will not lead to quadric convergence. Remember that we are only going to get certain quadratic convergence, if we use the


exact tangent. If we use the exact tangent at each iteration, then we are going to get quadratic convergence. Remember, again when do we get quadratic convergence, when we are in the neighborhood of the solution. When we are far away from this solution, we are not going to get the quadratic convergent any way.

So, what is the harm, if we can use the less powerfully approximation to the tangent. Then probably we are going to lose a little bit in our the convergence, but as it is not getting quadratic convergence, but we are saving the computational cost a lot the idea of the modified Newton algorithm is to use an approximation to that tangent. So, the full Newton algorithm uses the full tangent, the modified Newton algorithm, which we are going to talk about, next uses an approximation to the tangent.

(Refer Slide Time: 22:54)

### Modified Newton

- The modified Newton method consists in moving the calculation of the tangent stiffness to outside the iteration loop.
- Then  $\mathbf{K}_n^{(i)} = \mathbf{K}_{n-1} \forall i$  i.e. the tangent stiffness is only computed and factored only once for each load step on the basis of the previous converged state of displacements:
 
$$\mathbf{K}_n^{(i)} = \frac{d\mathbf{r}(\mathbf{u}_{n-1})}{d\mathbf{u}}$$
- In contrast to this, if full Newton iterations are used, then
 
$$\mathbf{K}_n^{(i)} = \frac{d\mathbf{r}(\mathbf{u}_n^{(i)})}{d\mathbf{u}}$$



The modified Newton method consists moving the calculation of tangent stiffness to outside the iteration loop. So, what do we mean by iteration loop? Well, to put it in word very simply it this we are saying that. Suppose, I have my converge solution at increment 5. So, I know this solution at increment 5 I know  $u_5$  I know  $f_5$ . I know  $f_6$  and I am interested in knowing  $u_6$ .

So, one way using full Newton is to compute the tangent at  $u_5$ , compute that tangent at the know converge solution  $u_5$ . Then invert the tangent to get an approximate  $\Delta u$  tilde to get the approximation to the update that  $\Delta u$  tilde, will probably not true

solution. There will be residual, there will be internal force, which will not match the external force go back and again i at slight start iterating.

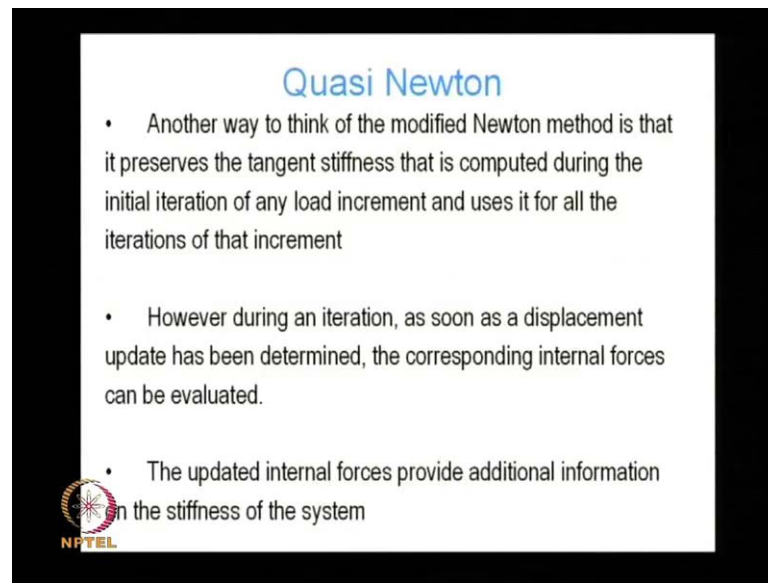
So, starting with 0 iteration I move to the next iteration I again calculate the tangent there, then calculate the update and move on. So, what does modified Newton do modified Newton say that I will compute the tangent at the end of the increment 5, but between increment 5 an increment 6, I am going to use this in tangent. So, between increment 5 and increment 6. We said that we are going to be converge we need 40 iteration, which has made against. Suppose, we need 40 iteration full Newton would say that each iteration.

As soon as I compute that as soon as I update I am going to calculate the tangent. So, modified Newton say well no I am not going to calculate the tangent, at each of those iteration. I am going to calculate the tangent at the converged solution at the start of the increment. Then I am going to store that tangent store the inverse. Then use that for all the all the intermediate iteration between 5 and 6 I am going to use the same tangent to compute my update. So, that is what it means by saying that I am going to move the calculation of the tangent stiffness to outside the iteration loop. So, before I start iterating I compute my tangent, but within the iteration loop my tangent does not change it remains constant.

So, then  $k_n$   $k_n$  plus iteration  $i$  is equal to  $k_{n-1}$  for all  $i$  right it remains the value at the end of the previous converged increment, so  $k_{n,i}$  for all iteration  $i$  is equal to  $k_{n-1}$ . That is the tangent stiffness is only computed and factor, what do I mean by factor? Well, we looked at all those factorization scheme also elimination composition all those factorizations, which help in inverting by matrix. So, I am going to keep those factors only once. Basically, what I mean to say simple word is that stored the inverse only once for each load step on the basis of the previous converge state of displacement.


So,  $k_{n,i}$  for each iteration  $i$  this is going to be  $d r u_{n-1} d$ , which is going to be the derivate of the residual with residual is nothing but you can think of this internal force. With respect to  $u$ , in contract in full Newton iterations are used than  $k_{n,i}$  is equal to  $d r u_{n,i} d u$ . So, it is at each iterate and taking the derivative to compute my new tangent. So, that is my modified Newton.

(Refer Slide Time: 26:59)



**Quasi Newton**

- Another way to think of the modified Newton method is that it preserves the tangent stiffness that is computed during the initial iteration of any load increment and uses it for all the iterations of that increment
- However during an iteration, as soon as a displacement update has been determined, the corresponding internal forces can be evaluated.
- The updated internal forces provide additional information on the stiffness of the system

 NPTEL

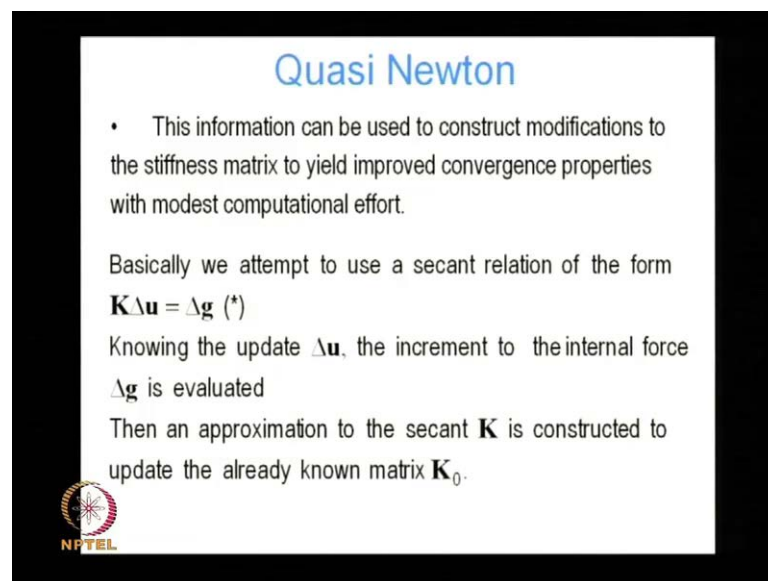
Another way to think of modified Newton is that it preserve the tangent stiffness. That is computed during the initial iteration of an load increment, uses it for all iteration during that increment, however during the iteration. As soon as the displacement update as been determined the corresponding internal forces can be evaluated. So, we are computing the displacement increment at each increment at each iteration, we are updating the displacement. We are calculating the internal force only thing that, we are not doing in modified Newton is that we are not calculating the tangent.

So, that is modified Newton's, so people came up with idea that well we are computing the internal force at that increment. At each iteration can be used that internal force computation to do some simple modification to the tangent stiffness, which an calculated at the beginning of the increment. I am not going to compute whole tangent stiffness at each iteration, but I am going to use my original tangent stiffness. Performs some little changes, some simple changes some little updates, which will not require inverting the whole matrix again. Can I make some small changes to my tangent stiffness? Because, I have some updated information I have some new information, at each iteration I have got the new internal force.

So, will the new internal force I calculating at each iteration can I use that information to update my tangent system to get a better estimate of the tangent stiffness, without acutely computing the full tangent stiffness well. So, that becomes the bases, what is known as

the quasi I Newton method, some till now we are looking at the modified method. Now, we are thinking quasi I Newton modified Newton, this uses the old tangent. Continue it for all the iteration with the increment quasi Newton with for keeps the old tangent. Then it tries to make small step wise modification to that old tangent. Using the information the updated information of my structure of my response, which is my, which is given by the updated internal force.

(Refer Slide Time: 29:33)



**Quasi Newton**


- This information can be used to construct modifications to the stiffness matrix to yield improved convergence properties with modest computational effort.

Basically we attempt to use a secant relation of the form

$$\mathbf{K}\Delta\mathbf{u} = \Delta\mathbf{g} \quad (*)$$

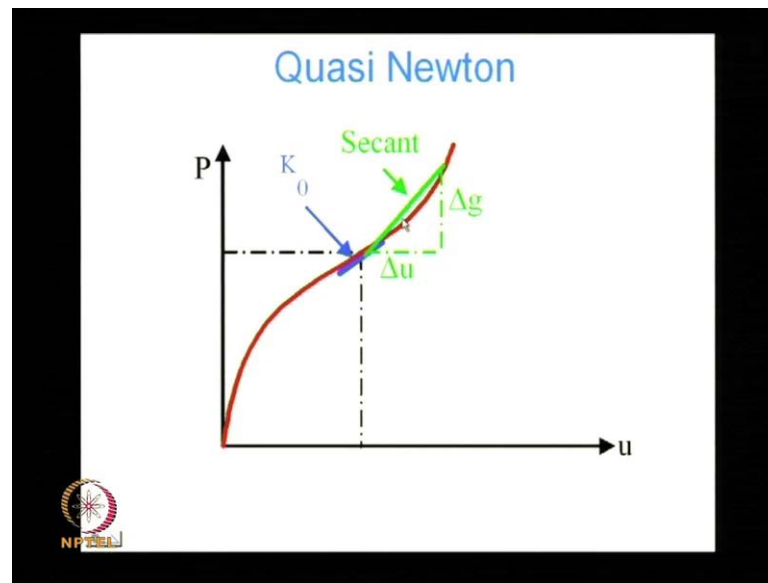
Knowing the update  $\Delta\mathbf{u}$ , the increment to the internal force  $\Delta\mathbf{g}$  is evaluated

Then an approximation to the secant  $\mathbf{K}$  is constructed to update the already known matrix  $\mathbf{K}_0$ .



So, this information this updated internal force modification the information can be used to contract modification to the tangent stiffness, to yield improve convergence properties with modest computational effect. Remember that as soon as use modified Newton we are going to get in terms of convergence, our convergence is not going as good. So, now, the idea is that if we try to improve my stiffness matrix margin at each iteration. Maybe, I am going to get better convergence. So, that is the idea basically we attempt to use a secant relation of this form, I think this is the better explain by this little picture here.

(Refer Slide Time: 30:21)



So, suppose this is my converge state. So, this is my force right and that is my displacement at a certain converge state. So, I have to go from here to the next converge state. So, what I do is that, I go from here to here. Suppose, I gone from here to here using the number of iteration. Then let suppose in one iteration itself gone from here to here in one.

So, that result in a change in displacement  $\Delta u$  and change in the internal force  $\Delta g$ . So, then I construct a secret a secant stiffness what is the secant stiffness it is  $\Delta g$  by  $\Delta u$ . So, that gives me an approximation to the stiffness, when I use this secant secant stiffness to update my tangent to find tangent here. If I use the full Newton, what will I do? I will go ahead and find the tangent here. I would find the tangent here that I do not do that I am saying I am going to keep my original tangent, but I am going to modify on how am I going to modified? Well, I am going to modify using the information on or my secant.

So, this was my original slope this is was think of it. If we think about it in terms of the slope of this line this was my slope at converge solution. This is probably aligned like this is a slope at the new iteration, but I do not want to evaluate that is too expensive. So, I say that I will use this original tangent plus the information about this secant tangent. I have to compute the update in the internal force, I know the update to the displacement. So, using this and this I am going to get the secant stiffness. Using this secant stiffness at



I am going to construct a approximate tangent here. So, I am going to construct in a approximate to the tangent here, using my original stiffness plus the slope of this secant. So, that is the basic idea of the quasi Newton iteration.

So, basically we attempt use the secant relation of the form  $k \Delta u$  is equal to  $\Delta g$  knowing the updated  $\Delta u$  the increment to the internal force  $\Delta g$  is evaluated. Then the approximate to  $k$  is constructed to update the already known matrix  $k_0$ , to update already known matrix  $k_0$ .

(Refer Slide Time: 33:20)


**Quasi Newton**

Corrections to the stiffness matrix  $K_0$  are introduced in the form of the vectors  $\Delta u$  and  $\Delta g$

The most popular update formula is a symmetric rank two correction, the so-called Broyden-Fletcher-Goldfarb-Shanno (BFGS) update.

In the BFGS update the new stiffness matrix is obtained via two additive corrections :

$$K = K_0 - \frac{(K_0 \Delta u)(K_0 \Delta u)^T}{\Delta u^T K_0 \Delta u} + \frac{\Delta g \Delta g^T}{\Delta g^T \Delta u}$$



So, the correction in the stiffness matrix  $k_0$  are introduced in the form of the vector  $\Delta u$  and  $\Delta g$  these are the things, I know  $\Delta u$  and  $\Delta g$ . So, if I have to construct correction  $k_0$  I have to those known formations the most popular update formula is a symmetric rank two correction. This is so called broaden flexure goal for Shano update, which is popularly known as the BFGS update. Why is it a rank two correction? Well, you can see it form here, this is my original stiffness. These are the changes that I am making to the original stiffness to obtain my updated quasi stiffness. Each of this updates is a rank one matrix, why it is a rank one matrix? Because it has got only one independent row or column.


So, it is the cross product, basically if you think of that product, it is a sensor product of this vector  $k_0 \Delta u$  with itself scaled by this quantity at the bottom, which is the scalar  $\Delta u^T k_0 \Delta u$  is the nothing but the scalar. This is the outer product  $k_0$

delta u is the vector, that vector outer product with that vector itself. Since, it is the outer product with itself its rank one of update there is only one independent vector. So, that matrix has got only one rank. So, that is the one rank one update plus another rank one update. So, it is a rank two correction, one rank one plus rank two rank one plus another rank one.

(Refer Slide Time: 35:14)

### Quasi Newton

- The above update removes the stiffness  $\mathbf{K}_0$  in the direction  $\Delta \mathbf{u}$  and replaces it by an outer product of  $\Delta \mathbf{g}$
- In directions orthogonal to  $\Delta \mathbf{u}$  no change in the stiffness is introduced
- The Sherman-Morrison formula gives the inverse of a matrix where an outer product has been added (i.e. following a rank-one update):

$$\left(\mathbf{A} + \frac{\mathbf{a} \otimes \mathbf{b}}{H}\right)^{-1} = \mathbf{A}^{-1} - \frac{(\mathbf{A}^{-1}\mathbf{a}) \otimes (\mathbf{A}^{-1}\mathbf{b})}{H + \mathbf{b} \cdot \mathbf{A}^{-1}\mathbf{a}}$$


So, what it basically does is that it remove the stiffness  $k_0$  in the direction  $\Delta \mathbf{u}$  and replaces it and outer product of  $\Delta \mathbf{g}$ . So, you can see what it does it takes the original stiffness. Then it takes the projection of that original stiffness in the direction of displacement updates  $k_0 \Delta \mathbf{u}$ . It is projecting the original stiffness and if you think of it very simple in terms of vector. It is taking that original stiffness and projecting in terms in the  $\Delta \mathbf{u}$  direction. That part it is removing from the original stiffness, because there is the negative sign.

So, that part it removes from the original stiffness and it adds and it adds another part, which is nothing but the outer product my increment in the internal force vector. It project how the stiffness in the direction of the displacement in increment. It adds the stiffness replace it, I would say replaces it by this term  $\Delta \mathbf{g} \Delta \mathbf{g}^T$ . So, in direction orthogonal  $\Delta \mathbf{u}$  changes in the stiffness in produced. So, basically in the direction  $\Delta \mathbf{u}$  it projects out something. It add something in direction orthogonal  $\Delta \mathbf{u}$ , there are no changes now. Why update formula, why update formula work? Well, one

might ask, well you are changing your original stiffness is this was the original stiffness. You are changing by this your forming the rank two update.

So, your new stiffness your new tangent stiffness is going to have totally different inverse. The inverse of the new updated tangent stiffness after my quasi my update is going to be totally different from  $k_0$ , that different what is the advantage. Since, the new tangent stiffness is totally different from my  $k_0$  do not I have to factorize it? Do not I have to inverted it again. So, the cost is exactly the same  $n$  square matrix have to again  $n$  square matrix. So, where is the saving, well the saving is because of this very important formula. This is known as the Shermann Morrison formula the Shermann Morrison formula say. If you look at it says that if I have the original matrix  $a$ . Then if I construct a rank, if I update like this.

Basically, I update it like this  $a$  times the product  $v$  by  $8$  by  $i$  update it like this. Then the inverse of my updated matrix can be written in terms of the inverse of my original matrix. So, what is the advance I do not have? I do not have to inverse  $n$  by  $n$  matrixes new  $n$  by  $n$  matrix any more I can use my old inverse. Then operate on that my old inverse on that vector that just a matrix vector multiplication, which is much cheaper than inverting matrix another matrix vector multiplication, which is again very cheap. So, matrix vector multiplication then I get two vector this vector with these two vector. I construct the outer product, I scale it with this scalar and that is my new inverse.

So, this is exact this form Shermann Morrison it exact. So, it says that given the matrix  $a$  and incase I construct the change that matrix like this. Then I do not have to invert that whole matrix. Again, I can construct the exact inverse the new matrix by just taking the old inverse and operating, that on operating with that on the vector and then taking outer product.

So, that is why no point right if after getting this new tangent stiffness. If I have to invert that whole  $n$  by  $n$  matrix again, then this quasi Newton update is no good at all. The quasi Newton update works because of the Shermann Morrison formula. It's says that if you have the rank one update, you do not really need to invert the whole matrix the new matrix to find out the inverse. Well, you can use the inverse to the rank one update to find the new inverse in a simple and a cheap fashion.

(Refer Slide Time: 39:53)

### Quasi Newton

- The BFGS update can be regarded as two successive rank one updates
- The outer product  $\frac{(\mathbf{K}_0 \Delta \mathbf{u}) \otimes (\mathbf{K}_0 \Delta \mathbf{u})}{\Delta \mathbf{u}^T \cdot \mathbf{K}_0 \Delta \mathbf{u}}$  constitutes the first rank one update
- The outer product  $\frac{\Delta \mathbf{g} \otimes \Delta \mathbf{g}}{\Delta \mathbf{g} \cdot \Delta \mathbf{u}}$  constitutes the second rank one update

The resultant inverse has the following form:

$$\mathbf{K}^{-1} = \left( \mathbf{I} - \frac{\Delta \mathbf{u} \otimes \Delta \mathbf{g}}{\Delta \mathbf{g} \cdot \Delta \mathbf{u}} \right) \mathbf{K}_0^{-1} \left( \mathbf{I} - \frac{\Delta \mathbf{g} \otimes \Delta \mathbf{u}}{\Delta \mathbf{g} \cdot \Delta \mathbf{u}} \right) + \frac{\Delta \mathbf{u} \otimes \Delta \mathbf{u}}{\Delta \mathbf{g} \cdot \Delta \mathbf{u}}$$

So, that is the Sherman Morrison formula the B F G S update. Basically, this update this we can regarded as two upsets rank one update the outer product  $\mathbf{K}_0$  tells, the product  $\mathbf{K}_0 \Delta \mathbf{u}$ . That is this part constituent the first rank one update, I have already mentioned that. The second part constituent the second rank one update. So, with these rank one update by new inverse is going to be 5 inverse my Sherman Morrison formula. If I use my Sherman Morrison formula and replace a b with my with these vector. Then this is going to be my new updated positive Newton inverse, is that clear? This is the resultant inverse if I use the quasi Newton update. Then I use my Sherman Morrison formula.

(Refer Slide Time: 41:03)

### Quasi Newton

- Multiplying the above relation with  $\Delta \mathbf{g}$  we get back  $\Delta \mathbf{u}$ , showing that the update formula is a true secant update of the form given in (\*)
- This update formula can be used successively to modify the inverse of the stiffness matrix based on successive internal force and displacement sub-increments  $(\Delta \mathbf{g}_i, \Delta \mathbf{u}_i)$
- In its original form the BFGS method removes the stiffness associated with the specific displacement increment  $\Delta \mathbf{u}$  and adds a rank one stiffness based on the force increment  $\Delta \mathbf{g}$

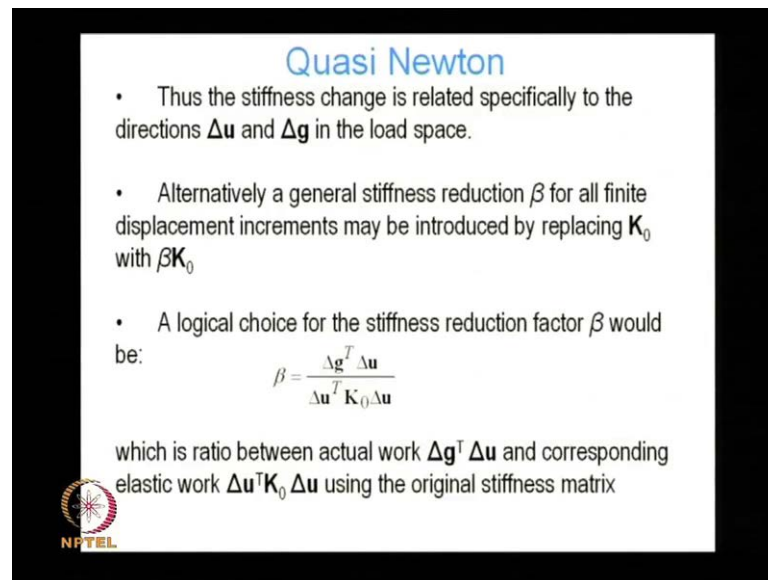
So, we can take that if we multiply this, if we operate with  $k$  inverse on  $\Delta g$ . Then I am going to get back  $\Delta u$ , which shows that update formula is a true secant update of the previous form. What am I going to say? Basically, I am saying that, this is my new updated matrix. So, if I operate matrix on this  $\Delta g$  and the form is such the structure of this matrix is, such that if I take  $k$  inverse operating on  $\Delta g$ . I am going to get back  $\Delta u$ . Remember that was the secant assumption, the entire secret assumption was this that this is  $\Delta g$  by  $\Delta u$ .

So, I have constructed my  $k$  inverse in such a manner with if I operate  $k$  inverse on  $\Delta g$  operate with  $k$  this  $k$  inverse on  $\Delta g$ . I am going to get back  $\Delta u$ , this enough to proved using the wave outer product the tens product vector. This is you can show that very easily. So, we are going to get back the true secant update of the form given in we can use this update formula very iteration. We start with my original converged solution I evaluate the tangent I take one iteration I update it using my rank one. Update using the BFGS update. Then I compute my new inverse using Sherman Morrison formula.

Then I go to the next iteration, again I have the new estimation of  $\Delta g$  I have the new estimate of  $\Delta u$  i use that to compute my update my BFGS update again. Then i again use my Sherman Morrison formula again construct the new inverse. So, this update formula can be used successively to use the modified stiffness matrix based on the stiffness successive internal force and displacement and sub increment. I call it sub increment iterate sub increments.

Now, its original form BFGS methods remove the stiffness associated with the, specific displacement  $\Delta u$  and add rank one stiffness based on the force increment  $\Delta g$ . Let's go back and take a look again. So, it removes the takes the projection  $\Delta u$  removes it and adds the  $\Delta g$ . So, everything all the change is to that  $k_0$  are accruing in the  $\Delta u$  direction.


(Refer Slide Time: 44:06)



**Quasi Newton**

- Thus the stiffness change is related specifically to the directions  $\Delta \mathbf{u}$  and  $\Delta \mathbf{g}$  in the load space.
- Alternatively a general stiffness reduction  $\beta$  for all finite displacement increments may be introduced by replacing  $\mathbf{K}_0$  with  $\beta \mathbf{K}_0$
- A logical choice for the stiffness reduction factor  $\beta$  would be:  
$$\beta = \frac{\Delta \mathbf{g}^T \Delta \mathbf{u}}{\Delta \mathbf{u}^T \mathbf{K}_0 \Delta \mathbf{u}}$$

which is ratio between actual work  $\Delta \mathbf{g}^T \Delta \mathbf{u}$  and corresponding elastic work  $\Delta \mathbf{u}^T \mathbf{K}_0 \Delta \mathbf{u}$  using the original stiffness matrix



The stiffness changes related specifically to the direction  $\Delta \mathbf{u}$  and  $\Delta \mathbf{g}$  in the loads case instead of using the BFGS updates using instead of using BFGS updates. There is another update for another we are updating it. Well, work to another update another way of updating it. Basically, we change the stiffness instead of looking the direction  $\Delta \mathbf{u}$  we change the stiffness in all direction. Alternatively the general stiffness reduction  $\beta$  for all finite displacement in increment, may be introduced by replacing  $\mathbf{K}_0$  with  $\beta \mathbf{K}_0$ .

So, I had a  $\mathbf{K}_0$  I am just going to scale it with the scalar  $\beta$ . The logical choice for the stiffness reduction factor is  $\beta$  could be  $\Delta \mathbf{g}^T \Delta \mathbf{u}$ , divided by  $\Delta \mathbf{u}^T \mathbf{K}_0 \Delta \mathbf{u}$ . That is because,  $\Delta \mathbf{u}^T \mathbf{K}_0 \Delta \mathbf{u}$  is like the elastic work  $\mathbf{K}_0 \Delta \mathbf{u}$ . Suppose, that  $\mathbf{K}_0$  metrical is elastic which is not right which may not be, but it may not be right, that  $\mathbf{K}_0 \Delta \mathbf{u}$  times  $\Delta \mathbf{u}^T$  is like the elastic work,  $\mathbf{K}_0$  is like the force that force times displacement  $\Delta \mathbf{u}$  is like the elastic work material elastic. This would be the elastic work, but the actual work is nothing but  $\Delta \mathbf{g}^T \Delta \mathbf{u}$ . Because, the actual force is  $\Delta \mathbf{g}$  it is not  $\mathbf{K}_0 \Delta \mathbf{u}$ . Because, the response is not linear,  $\Delta \mathbf{g}$  is no longer equal to the  $\mathbf{K}_0 \Delta \mathbf{u}$ .

So, that force the actual force moving through the displacement increment is the actual work the elastic force  $\mathbf{K}_0 \Delta \mathbf{u}$  moving through, the displacement increment is the elastic work. So, we say that the actual work is a fraction of the elastic work. So, we are

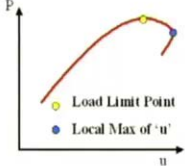
going to change our stiffness by the same fraction. So, it is the beautiful idea, the assumption is that I have  $k_0$ . If I have the  $k_0$  constant represent the elastic matrix.

So, the work done  $k_0 \Delta u$  tangent  $\Delta u$  transpose, but my actual tangent is not that. What is the measure of my actual tangent measure of my actual tangent, is the actual work done. If it was the elastic tangent the total work done would be this the actual work done is  $\Delta g$  transpose  $\Delta u$ . If I take the ratio of these two that is by how much I have to reduce my tangent from my elastic tangent. So, that is the idea, that is another way of doing the quasi Newton iteration.


(Refer Slide Time: 47:11)

**Possible convergence problems**

- In many structural and solid mechanics problems the load displacement curve changes slope. This occurs typically at limit points.



- The equilibrium path following algorithm adopted so far moves from one equilibrium state to the next based on the tangent stiffness

 NPT

So, that is say about various Newton iteration scheme, but the fact is that Newton iteration do not work in all situations. They need to divergent the solution, do not converge and this circum circumstances, what are those certain circumstances? When the Newton iteration does not converge, well in many structure and solve mechanic problem the load displacement curve changes So, this limit at certain point at limit point or for instant point, when we have instability in the solution the slope become negative. If I draw my load displacement curve the slope up till here. This load displacement has the positive slope reaches the limit point the slope becomes 0.

Then the slope has changed the sign in situations like this the Newton's Raphson method cannot converge. Because, you can see this tangent is predicting something like this well the actual slope is like this. When this become to huge, it takes the increment like this it

goes somewhere here. Then it cannot find there is no load displacement curve as that level. Let us assume the slope takes like this right the slope takes like this and it projects it out, takes the displacement increment. Then it tries to find load displacement curve to find the new to find the new equilibrium equilibrium configuration, but the load displacement curve simply does not exist. Because, it is change slope, because it is totally different.

So, it has the lot of difficulty in finding the new high equilibrated solution. In such situations the typically we have problems. This typically a curves at the limit points at load limit points the equilibrium path following algorithm adopted. So, far moves from one equilibrium state to the next based on the tangent stiffness.

(Refer Slide Time: 49:36)

**Possible convergence problems**

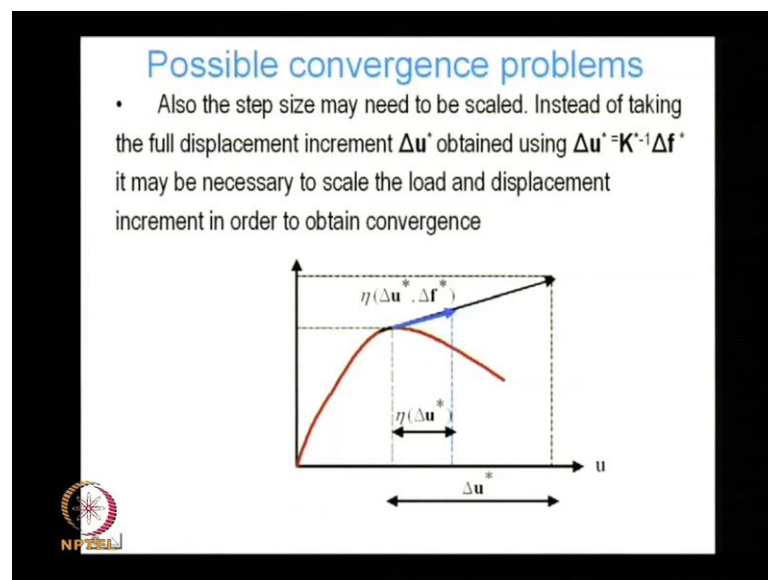
- If the slope undergoes large changes along the equilibrium path, it is necessary that (a) the orientation of the path be correct (b) step sizes along the path be controlled
- Once a limit point has been exceeded, there may be a need to change the direction of the load incrementation

If the slope under goes the large changes along the equilibrium path it is necessary that the orientation of the path correct and the steps size be conclude. Let's look at example, where we have the limit path, where the load displacement curve is changing slope here its changing slope here. So, suppose I went from here to here, this was my previous converge solution. I got another converge solution here then I construct my tangent. I construct my tangent to the converge solution. It turns out that my tangent is like this, its tangent is like this is my tangent. At this point I construct my tangent and my tangent is like this, but this is pointing in the totally wrong direction.



It is pointing exactly opposite direction in which my load displacement load equilibrium solution lays my equilibrium, some were my this red curve, but this if I compute the tangent here. My tangent says that take the direction that is the direction, which is going to give my that is the slope at that point. Then I am going to go totally in the wrong direction, I am going to get lost am not going to converge anymore, so the right solution the actually the second orientation. The orientation is this the actual that orientation is that, if I take the displacement increment in this direction. I take the increment as this direction as the totally lost once the limit point exceeded there, maybe the need to change the direction of the load incrimination.

(Refer Slide Time: 51:28)



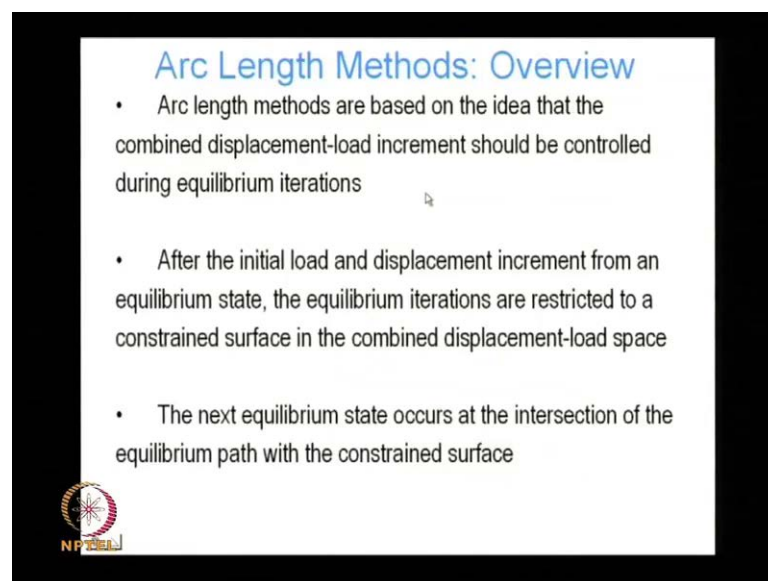
What is the analyses situation? Well, here we have a again we are looking at a maximum. My load has reached maximum my load has reached maximum. The projection is down, suppose I calculate the tangent here. I will calculate the tangent here and I take the loads step like this. I have already mention this step were I take a step delta u star. Then from here I tried to find for this delta u star try to find what is my internal force, but for this delta u star there is actually no internal force at all, because my load displacement curve is only up to here.

So, there is no internal force at all we are not going to get any were. So, what do you have to do? Well, in that case I have to cut back I have to cut back my displacement increment, what does that mean? That mean instead of taking my full delta u star, how

did I calculate my  $\Delta u^*$ ? Well, I knew my tangent here I knew my force increment here from here. I inverted that tangent matrix I multiplied it by that force increment that gave me  $\Delta u^*$ . If I take the  $\Delta u^*$  because there is no internal force vector curve here.


So, I have to cut back whatever estimate have got from my from my solution I have to reduce the increment scale, this with this factor  $\theta$ . The step size may need to be scaled instead of taking full displacement increment  $\Delta u^*$   $\Delta u^*$  is equal to  $k^*$  inverse into  $\Delta f$ . Where,  $\Delta f^*$  is this little part right is this little part and  $k^*$  inverse is the  $k^*$  is the slope here. Then I got  $\Delta u^*$ , it may be scale the load and displacement increment. In order to obtain convergence look at that I said I said scale the load as well as the displacement increment not does the displacement increment. Also, the load and those are the basis what are known as arc length methods.

(Refer Slide Time: 53:47)



**Arc Length Methods: Overview**

- Arc length methods are based on the idea that the combined displacement-load increment should be controlled during equilibrium iterations
- After the initial load and displacement increment from an equilibrium state, the equilibrium iterations are restricted to a constrained surface in the combined displacement-load space
- The next equilibrium state occurs at the intersection of the equilibrium path with the constrained surface



We are going to talk about, we have various arc length method for instance the wings method and things like that, which are important for doing finite any advance finite element code. You will have this methods, if they are interest the computing things like post buck solution. I do not want talk too much about solid mechanics any particular application, but these arc length methods are necessary. When my solution curve can counter this the limit points, which result in my Newton hydration.

Next class we are going to talk about arc length methods. Then we are going to talk about gradient method particular iterated method, which with depend on calculated gradients for instant the step is method and the configuration radiant method.

Thank you.