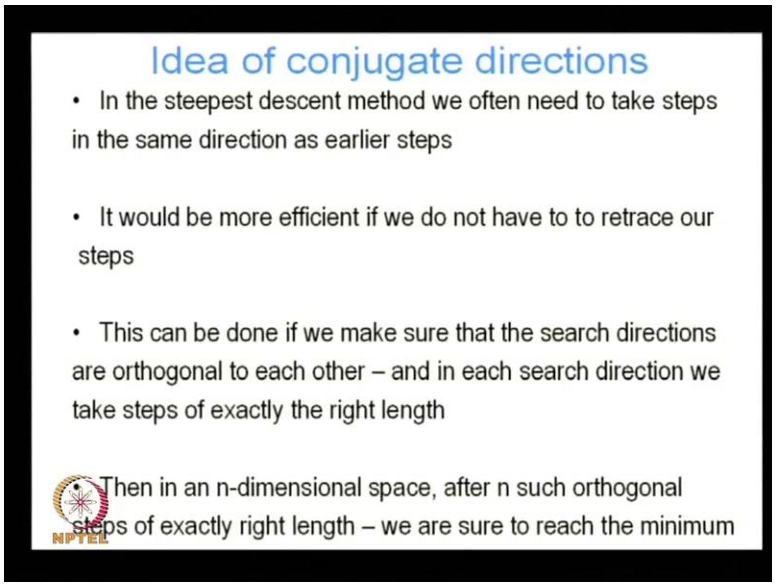
Numerical Methods in Civil Engineering
Prof. Dr. Arghya Deb
Department of Civil Engineering
Indian Institute of Technology, Kharagpur

Lecture - 18
Conjugate Gradient Method

Last time in our this lecture 17; in our series in numerical methods in civil engineering we talked about conjugate directions in general. This time we will complete our direction, we will complete our discussion of conjugate directions right we just started with that basically we will complete that and then we are going to talk about the method of conjugate gradients. We looked at one gradient based method already and the method we looked at in detail was the method of steepest descent, where we go along a particular direction; we try to minimize the function in that direction and then we move along in another direction.

(Refer Slide Time: 01:05)



Idea of conjugate directions

- In the steepest descent method we often need to take steps in the same direction as earlier steps
- It would be more efficient if we do not have to to retrace our steps
- This can be done if we make sure that the search directions are orthogonal to each other – and in each search direction we take steps of exactly the right length
- Then in an n-dimensional space, after n such orthogonal steps of exactly right length – we are sure to reach the minimum

There is no guarantee that the direction in which are moving we are not going to retrace our steps right because we often come back and take the same steps in the same direction this steepest descent method because there is no requirement. Requiring, those steps to be to satisfy any relation with respect to each other. The directions in which we travel; the directions in which we move the steepest descent directions basically the gradient the negative of the gradient direction. It does not there is no restriction, there is no requirement that these those directions satisfy any relationship between each other right.

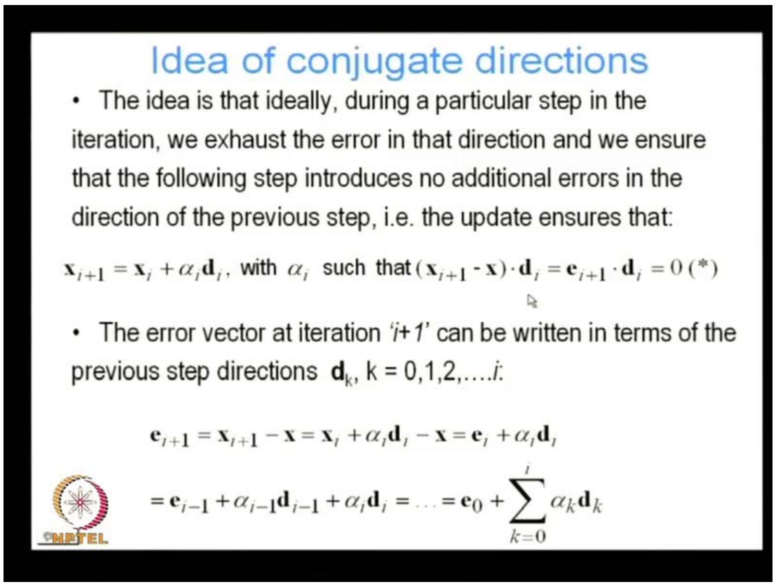
Because of that, we can move in this same directions several times during the course of iteration.

But, this is not very economical or this is not very efficient because we are having to retrace our steps ideally, what we should do is that we should go in one direction and exhaust all our possibilities in that direction; by exhaust all our possibilities I mean that we minimize the function in that direction. If there is an error; if there is a global component of error if you think of global error vector.

So, in that direction we make the component of the error vector fully 0 right and then we move on to another direction. But, when we move to the other direction and we try minimize the function in that direction we do not want errors to be creeping back in a previous direction that we have travelled right. we want those errors to be, the directions to be orthogonal number one and number two we want to make sure that the errors do not grow again in that direction right.

So, both those requirements are fulfilled by some of the methods that we are going to talk about today. Right, this can be done if we make sure the search directions are orthogonal to each other and each in each search direction. we take steps of exactly the right length. That is going to of course, minimize the function in that direction and then in an n dimensional space after n such orthogonal steps of exactly the right length. We are sure to reach the minimum provided we do not the error does not grow back in that direction right; that is a very important criteria.

(Refer Slide Time: 03:19)



Idea of conjugate directions

- The idea is that ideally, during a particular step in the iteration, we exhaust the error in that direction and we ensure that the following step introduces no additional errors in the direction of the previous step, i.e. the update ensures that:

$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i$, with α_i such that $(\mathbf{x}_{i+1} - \mathbf{x}) \cdot \mathbf{d}_i = \mathbf{e}_{i+1} \cdot \mathbf{d}_i = 0 \; (*)$

- The error vector at iteration *'i+1'* can be written in terms of the previous step directions $\mathbf{d}_k$, k = 0,1,2,….*i*:

$$\mathbf{e}_{i+1} = \mathbf{x}_{i+1} - \mathbf{x} = \mathbf{x}_i + \alpha_i \mathbf{d}_i - \mathbf{x} = \mathbf{e}_i + \alpha_i \mathbf{d}_i$$

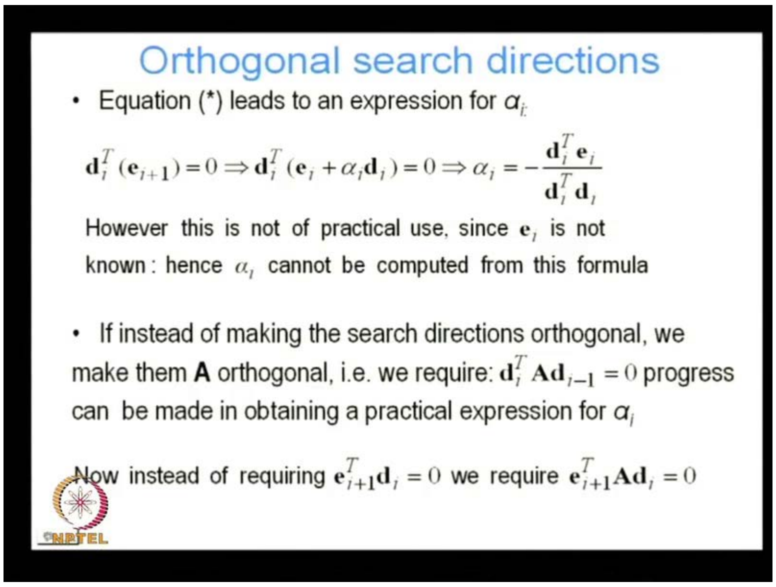
$$= \mathbf{e}_{i-1} + \alpha_{i-1}\mathbf{d}_{i-1} + \alpha_i \mathbf{d}_i = \ldots = \mathbf{e}_0 + \sum_{k=0}^{i} \alpha_k \mathbf{d}_k$$

The idea is that ideally during a particular step in the iteration, we exhaust the error in that direction and we ensure that the following step introduces no additional errors in the direction of the previous step that is the update ensures that if x i plus 1 is equal to x i plus alpha i d i is my update law with alpha i such that alpha i such that x i plus 1 minus x dot d i is equal to e i plus 1 dot d i is equal to 0. So, basically I am saying that the error at the i plus one th iteration the error vector the i plus one th iteration, which is this quantity right x i plus 1 minus x this does not have any component in the d i direction right. This error vector in the next iteration, it has got no component in the d i in the previous direction right.

The error vector at iteration i plus 1 can be written in terms of the previous step directions d k, how can we do that well let us see what we can do? we know that error at i plus 1 e i plus 1 is equal to x i plus 1 minus x right. That is by definition and this x i plus 1 by our update rule is equal to x i plus alpha i d i minus x right. So, x i minus x if i pull it together that becomes e i the error at iteration i plus alpha i d i. So, then if i write e i in terms of e i minus 1 the error at the i minus one th iteration and d minus 1 the direction at the i minus one th iteration then we can do this recursively. Eventually, we will end up with a equation for e or e i plus 1 in terms of my original error vector e 0 and all my directions that I have travelled up to up to that iteration up to the i plus oneth iteration.

So, eventually we are going to get something like e 0 plus sigma k equal to 0 to i alpha k d k because these all these d i s are going to have components right. So, the error at i plus 1 is equal to my original error plus sigma k equal to 0 i alpha k d k. This equation it is very meaning it has a lot of meaning right what it saying is that you think about it. Its saying that the error at some iteration at the end i plus 1 is equal to my original error plus this alpha k d k. So, what these alpha k d k are doing is actually, they are eating away the error. So, e 0 is my original error. So, every time I take this d k this e 0 something is changing in that e 0 right. So, eventually when I take n iterations all that error, all that error in e 0 has been nullified and a n plus 1 is eventually equal to 0 right. So, at the end I get 0 right.

(Refer Slide Time: 06:32)



Orthogonal search directions

- Equation (*) leads to an expression for α_i:

$$\mathbf{d}_i^T(\mathbf{e}_{i+1}) = 0 \Rightarrow \mathbf{d}_i^T(\mathbf{e}_i + \alpha_i \mathbf{d}_i) = 0 \Rightarrow \alpha_i = -\frac{\mathbf{d}_i^T \mathbf{e}_i}{\mathbf{d}_i^T \mathbf{d}_i}$$

However this is not of practical use, since $\mathbf{e}_i$ is not known: hence α_i cannot be computed from this formula

- If instead of making the search directions orthogonal, we make them **A** orthogonal, i.e. we require: $\mathbf{d}_i^T \mathbf{A}\mathbf{d}_{i-1} = 0$ progress can be made in obtaining a practical expression for α_i

Now instead of requiring $\mathbf{e}_{i+1}^T \mathbf{d}_i = 0$ we require $\mathbf{e}_{i+1}^T \mathbf{A}\mathbf{d}_i = 0$

So, let us go back to that equation this equation, equation gives me an expression for alpha i, how does it do that well what it says is that d i dotted with e i plus 1 equal to 0. So, i can write that as d i dot d i transpose e i plus 1 equal to 0. So, d i transpose e i plus alpha i d i equal to 0. From my equation right here right from my equation right here e i plus 1 equal to e i plus alpha i d i. So, e i plus alpha i d i equal to 0. So, that gives me an expression for alpha i but what do you see about this expression. This expression has alpha i on the left hand side and on the right hand side it has d i as well e i right. This is not useful at all why is it not useful because I want to know what step I have to take in the alpha i direction right.

At that point I do not have no idea what will be the result of my step right. So, I have do not know what at I have no idea what the error will be. So, I have no idea what e i will be right i. So, after I take the step alpha i then I update by variable x i plus 1 only then I can calculate the error right. But, it is saying that in order to know how much to travel how much to travel in that direction I have to know what the error will be which impossible right is. So, this is a theoretical this is a good it is a nice equation but it is practically useless right we cannot use it to calculate our step length right.

So, how do we calculate our step length? So, in order to calculate our step length we must be using information which we already know right for instance we know the residual at step i that we can use or we know the residual in the previous steps that we can use. I know the directions in the previous steps that I can use and things like that right. So, I should not be using information, which I do not know yet right that is that does not make any sense. But in order to so, so how do we get that get an expression for alpha i which we can actually calculate well in order to do that we impose a certain, a new notion we actually introduce a new notion of orthogonality right.

We say that we are not going to make the error e i directly orthogonal e i plus 1 directly orthogonal to d i we are not going impose that restriction. We are going to impose this restriction e i is orthogonal, e i plus 1 is orthogonal to d i with respect to this tense this matrix A right. So, this is like some sort of normally when you think of in a inner product right you just think of an inner product is the dot product of two vectors right that is true in a Cartesian space right.

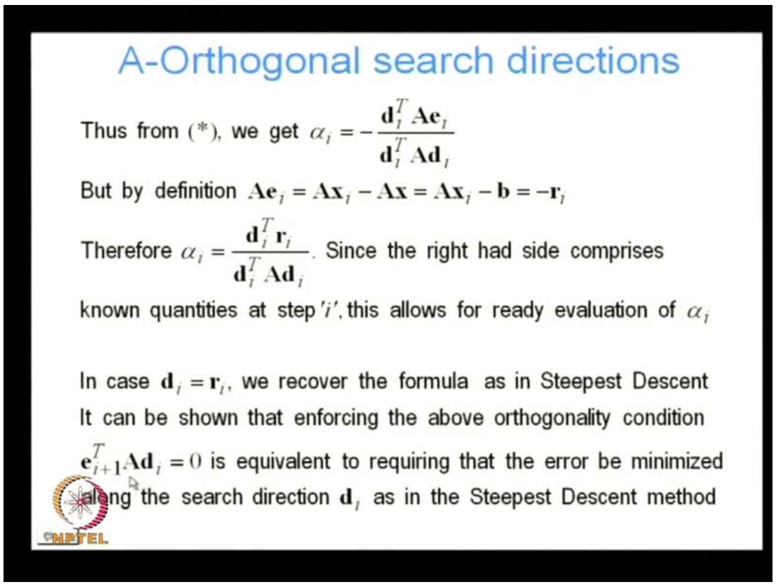
But in a space where there is a different metric and if you are familiar with the notion of metric in the Cartesian space; the metric is the identity tenser it is the identity matrix right that is why we can calculate dot product as e i transpose d i right. Its basically we are actually doing e i plus 1 transpose identity d i right because identity is the metric in that space in the three dimensional Euclidian space in the in a Cartesian coordinate system. But, there are some systems where the metric is not the identity tenser right it is not the identity metrics right.

So, in those in for instance when we are thinking of I can think of structural mechanics applications right when one thinks of shell structures and things like that curve Curvilinear coordinate systems in that case, the notion of length because the inner product the dot product is intimately related to the idea of length right. If you have dot product what does if you take dot product of to a vector with itself that is the square of the length of the vector right. So, the dot product is very related to the idea of norm of length right. So, in a I do not want digress too much, but I am saying this is this is a very common idea in well in lots of fields, basically it started with differential geometry, but there are lots of fields, where this ideas is used right the idea of a metric of a of a tenser which actually defines or a metric or a or a metrics, which actually defines a dot product; which defines a length right.

So, in this case we are saying that here our matrix which defines a dot product is this A matrix. So, any dot products we can calculate are going to be with respect to the A matrix. So, it will be of this form right and when we say that normally when we say that two vectors are in my usual Cartesian space right, if I say that everything is planar, I say dot product is two vectors are orthogonal; if there dot product is equal to 0. So, now, the idea of orthogonality has changed. So, dot product means that those two vectors are their dot product is equal to 0 but the dot product is defined with respect to a particular matrix right which is my metric right.

So, instead of making the search directions orthogonal we make them A orthogonal right. So, this a orthogonal orthogonality with respect to this metric A that is we now require d i transpose A d i minus 1 is equal to 0.So, all the previous direction are orthogonal to my current direction but with respect to a right. So, progress can be made in obtaining a if you do that we can get a practical expression for alpha. So, from start we get So, now, instead of this basically we will go back to this equation right now we will say e i plus 1 dot A e i plus 1 a dotted with d 1 right equal to 0 right.

(Refer Slide Time: 12:53)



A-Orthogonal search directions

Thus from (*), we get $\alpha_i = -\dfrac{\mathbf{d}_i^T \mathbf{A}\mathbf{e}_i}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i}$

But by definition $\mathbf{A}\mathbf{e}_i = \mathbf{A}\mathbf{x}_i - \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}_i - \mathbf{b} = -\mathbf{r}_i$

Therefore $\alpha_i = \dfrac{\mathbf{d}_i^T \mathbf{r}_i}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i}$. Since the right had side comprises known quantities at step 'i', this allows for ready evaluation of α_i

In case $\mathbf{d}_i = \mathbf{r}_i$, we recover the formula as in Steepest Descent

It can be shown that enforcing the above orthogonality condition $\mathbf{e}_{i+1}^T \mathbf{A}\mathbf{d}_i = 0$ is equivalent to requiring that the error be minimized along the search direction $\mathbf{d}_i$ as in the Steepest Descent method

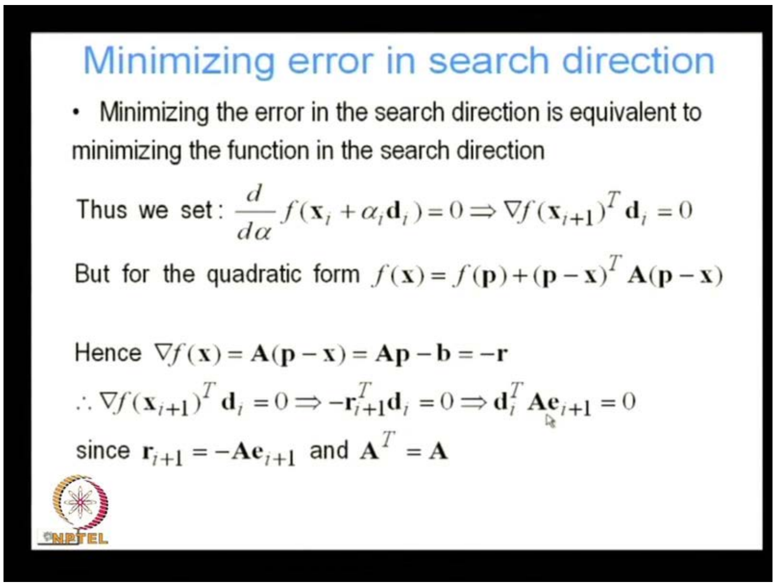
So, if we do that then we get an expression for alpha i which is minus d i transpose A e i d i transpose A d i. So, compare this expression with my earlier expression the only thing that has come is that a right and this changes the whole thing because I know that A e i is r i right because A e i is equal to A x i minus A x, where x is my true solution right and I know a x is equal to b e.

So, this is A x i minus b which is nothing but my residual right. So, I already know that right I know already the residual at i right. So, I can use that here. So, I have d i transpose r i d i transpose A d i. So, now, I know everything right I know what the direction that I have travel of course, I know the direction that I have to travel I know the residual the error at the i th iteration.

I have did not it is the residual at their i th iteration and then if I use these two things I can calculate my step alpha i and then can move in the d i direction by that step since the right hand side comprises known quantities at step I this allows for ready evaluation of alpha i right in case d i is equal to r i. So, I do not have this steepest descent formula with me, but if you if you if you go back your to last times lecture you will find that for this steepest descent method only thing that is different was that d i is instead of d i we have r i right. We recover the formula as in steepest descent. So, it can be shown that enforcing the above orthogonality condition e i plus 1 transpose A d i is equivalent to requiring that the error be minimized along the search direction d i as in the steepest descent method.

So, when we impose this condition we are all actually enforcing the condition that the error is actually minimized along the direction d i right. We saw that for steepest descent it turns out that is true for the method of conjugate directions as well.

(Refer Slide Time: 15:21)



Minimizing error in search direction

- Minimizing the error in the search direction is equivalent to minimizing the function in the search direction

Thus we set: $\frac{d}{d\alpha} f(\mathbf{x}_i + \alpha_i \mathbf{d}_i) = 0 \Rightarrow \nabla f(\mathbf{x}_{i+1})^T \mathbf{d}_i = 0$

But for the quadratic form $f(\mathbf{x}) = f(\mathbf{p}) + (\mathbf{p}-\mathbf{x})^T \mathbf{A}(\mathbf{p}-\mathbf{x})$

Hence $\nabla f(\mathbf{x}) = \mathbf{A}(\mathbf{p}-\mathbf{x}) = \mathbf{Ap} - \mathbf{b} = -\mathbf{r}$

$\therefore \nabla f(\mathbf{x}_{i+1})^T \mathbf{d}_i = 0 \Rightarrow -\mathbf{r}_{i+1}^T \mathbf{d}_i = 0 \Rightarrow \mathbf{d}_i^T \mathbf{A}\mathbf{e}_{i+1} = 0$

since $\mathbf{r}_{i+1} = -\mathbf{A}\mathbf{e}_{i+1}$ and $\mathbf{A}^T = \mathbf{A}$

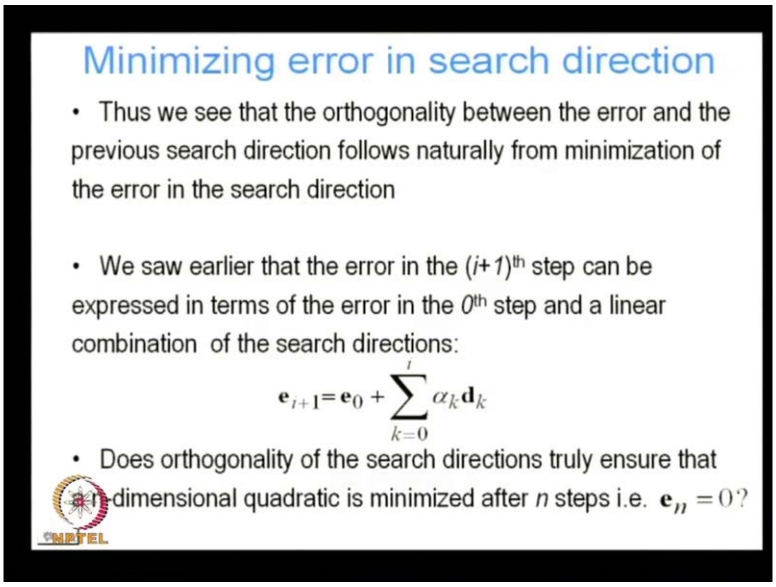
So, minimizing the error in the search direction is equivalent to minimizing the function in the search direction right. So, basically want to minimize the function in order to minimize the error because what we are interested in doing is finding the minimum of the function. So, if you minimize the function in that direction we are also minimizing the error in that direction right because our purpose is to minimize the function f right. So, we want to minimize that function in the d i direction. So, how do we do that? we take the directional derivative right d d alpha d d alpha of f x i plus alpha i d i equal to 0.

So, this if we evaluate this is nothing, but the gradient at x i plus 1 transpose d i right this is actually x i plus 1 right. So, gradient at x i plus 1 transpose times d i that must be equal to 0 right. So, this we I showed you how it comes right we are first taking the gradient of f with respect derivative of f with respect to its argument and then taking the derivative of the argument with respect to alpha right; alpha i and that is how we get this d i right. So, that is the condition I get, but let us see and let us take a look of for the for the equation for a quadratic form which is f x is equal to f p plus p minus x transpose A p minus x right. Therefore gradient of f x is equal to A p minus x A p minus A x A x is equal to b. So, A p minus b equal to minus r right.

So, basically what this condition, what it saying is that the residual this is nothing but the residual a gradient of f at x i plus 1 is equal to the residual at i plus 1, the residual at i plus 1 its actually the negative of the residual at i plus 1 that is orthogonal to d i right that is orthogonal to d i, which basically means that d i dotted with A e i plus 1 equal to 0 because r i plus 1 is equal to A e i plus 1 right, A e i plus one.

So, what do we see from this what this what does this tell us this tells us that if we are minimizing the function in this direction in this direction in that in the d i in the d i direction what we are actually doing what we are actually doing is we are satisfying this condition. We are satisfying the condition that d i is orthogonal to e i plus 1 with respect to the A matrix right with respect to the A matrix. So, d i transpose A e i plus 1 equal to 0 is that clear.

(Refer Slide Time: 18:26)



Minimizing error in search direction

- Thus we see that the orthogonality between the error and the previous search direction follows naturally from minimization of the error in the search direction
- We saw earlier that the error in the $(i+1)^{th}$ step can be expressed in terms of the error in the 0^{th} step and a linear combination of the search directions:

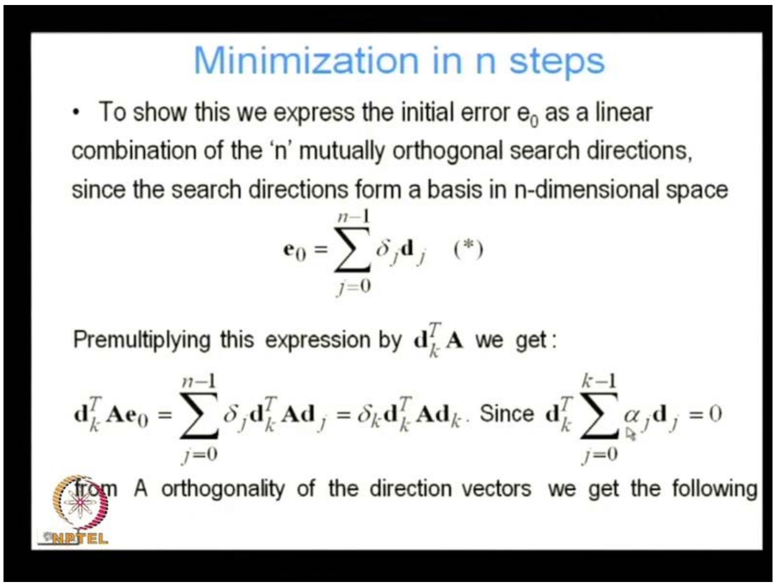
$$\mathbf{e}_{i+1} = \mathbf{e}_0 + \sum_{k=0}^{i} \alpha_k \mathbf{d}_k$$

- Does orthogonality of the search directions truly ensure that n-dimensional quadratic is minimized after n steps i.e. $\mathbf{e}_n = 0$?

Thus we see that the orthogonality between the error and the previous search direction follows naturally from minimization of the error in the search directions. So, we minimize the error in the search direction this automatically leads to the orthogonality of the search direction with a error. we saw earlier that the error in the e plus one th step can be expressed in terms of the error in the zero th step and a linear combination of the search directions we saw that earlier right that error at any i plus 1 is equal to my initial error e 0 plus, some linear combination of my search directions right we saw that already.

So, does orthogonality of the search directions truly ensure that a n dimensional quadratic is minimized after n steps that is n e n equal to 0. So, what we have seen up till now, we have seen that our if we minimize a function along its if you minimize that function along at a particular direction we get orthogonality between the error and the previous search direction that is all we have seen but we but does that does that really lead to the to the minimization of the function in n iterations that is what we want to see right. So, if the if the function is truly minimized after n steps it would mean that e n the error at the n end of n iterations is equal to 0.

(Refer Slide Time: 20:05)



Minimization in n steps

- To show this we express the initial error e_0 as a linear combination of the 'n' mutually orthogonal search directions, since the search directions form a basis in n-dimensional space

$$\mathbf{e}_0 = \sum_{j=0}^{n-1} \delta_j \mathbf{d}_j \quad (*)$$

Premultiplying this expression by $\mathbf{d}_k^T \mathbf{A}$ we get:

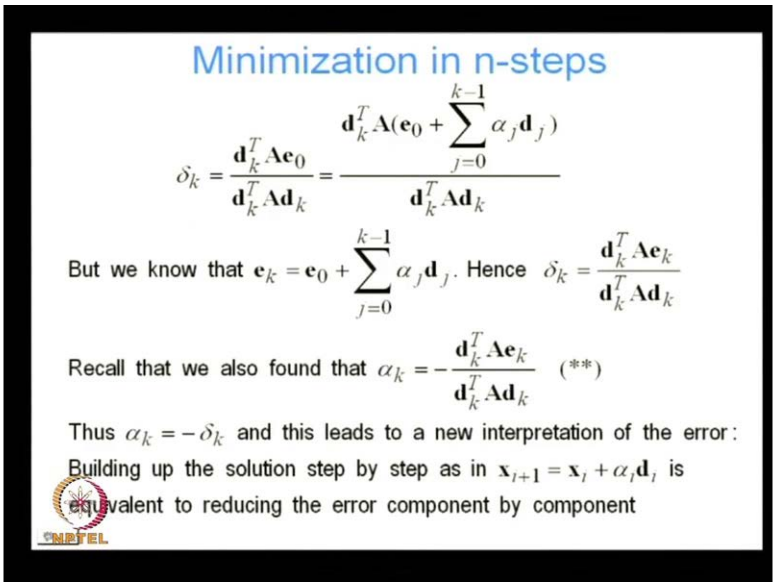
$$\mathbf{d}_k^T \mathbf{A} \mathbf{e}_0 = \sum_{j=0}^{n-1} \delta_j \mathbf{d}_k^T \mathbf{A} \mathbf{d}_j = \delta_k \mathbf{d}_k^T \mathbf{A} \mathbf{d}_k. \text{ Since } \mathbf{d}_k^T \sum_{j=0}^{k-1} \alpha_j \mathbf{d}_j = 0$$

from A orthogonality of the direction vectors we get the following

So, how do us So, to show that to show this we express the initial error e 0 as a linear combination of my search directions, why can I do that? why can I do that the? why how why can I show that my I can express my initial error is this I know I this I have shown already right e i plus 1 is equal to e 0 plus sigma k equal to this I have shown. But now I am saying, that my initial error I can express it as a linear combination of my n such directions why is that well that is simply because my n such search directions are mutually orthogonal. And since they are mutually orthogonal they form a basis in my n dimensional space right. So, any vector I can express as a linear combination of my search directions right. So, I am expressing that the vector e 0 as a linear combination of the search directions and then I premultiply this expression by d k transpose A right.

So, I get d k transpose A e 0 delta j d k transpose A d j you have a summation here right but basically you recall that all my search directions are mutually orthogonal to each other not just not orthogonal but A orthogonal to each other right. So, these all these terms d k transpose A d j are going to give me 0 unless the two indices k and j are identical right, when that happens then I get this term delta k d k transpose A d k right . So, that is true since d k transpose sigma j equal to k k minus 1 alpha j d j is equal to 0 which is basically because they are orthogonal right; they are orthogonal.

(Refer Slide Time: 22:13)



Minimization in n-steps

$$\delta_k = \frac{\mathbf{d}_k^T \mathbf{A}\mathbf{e}_0}{\mathbf{d}_k^T \mathbf{A}\mathbf{d}_k} = \frac{\mathbf{d}_k^T \mathbf{A}(\mathbf{e}_0 + \sum_{j=0}^{k-1} \alpha_j \mathbf{d}_j)}{\mathbf{d}_k^T \mathbf{A}\mathbf{d}_k}$$

But we know that $\mathbf{e}_k = \mathbf{e}_0 + \sum_{j=0}^{k-1} \alpha_j \mathbf{d}_j$. Hence $\delta_k = \frac{\mathbf{d}_k^T \mathbf{A}\mathbf{e}_k}{\mathbf{d}_k^T \mathbf{A}\mathbf{d}_k}$

Recall that we also found that $\alpha_k = -\frac{\mathbf{d}_k^T \mathbf{A}\mathbf{e}_k}{\mathbf{d}_k^T \mathbf{A}\mathbf{d}_k}$ (**)

Thus $\alpha_k = -\delta_k$ and this leads to a new interpretation of the error: Building up the solution step by step as in $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i$ is equivalent to reducing the error component by component

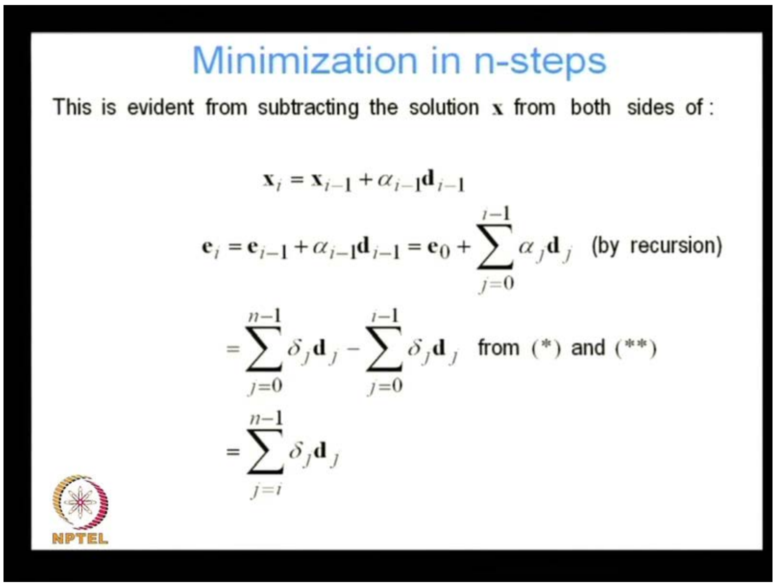
So, what do we have? So, we get we can write delta k these coefficients we can write these coefficients as the quotient of d k transpose A e 0 and d k transpose A d k right. So, I can write that so, then I take out then I look at the numerator I have d k transpose a and then I do a little trick I say I instead of I have e 0 here I write e 0 plus sigma j equal to 0 up to k some over j equal to 0 to k minus 1 alpha j d j why can I do that? because I know that this term with d if I if I take the dot product with d k transpose A that is going to give me 0 because all these directions are A orthogonal to d k right. So, I can write like this divided by d k transpose d k but again then we have already seen that this expression this expression is nothing but e k right. So, what we are saying is that delta k is equal to d k transpose A e k divided by d k transpose A d k right.

So, compare this expression for delta k recall, what is delta k? Delta k is nothing, but the coefficients of the d ks of the d js in my original error vector right. So, e 0 is equal to sigma j equal to 0 to n minus 1 delta j d j. So, delta j i found is this or delta k is this right how does this compare with my expression for my alpha k s which I obtained earlier look at this is my expression for the alpha k s right d i transpose A e i d i transpose A d i. And this is d d i transpose A e i d i transpose A d i that one had a negative sign. So, basically I found that my alpha k and d k are equal in magnitude, but they differ in sign they differ in sign.

So, this is very revealing because this gives me this leads to a new interpretation of the error because what are all the alpha k is doing? The alpha k s are updating my iterates right x k plus 1 is equal to x k plus alpha k d k and what is my delta k doing? delta k is a coefficient in the expansion of the original error. So, what it saying is that every time I take a step I change my trail solution by alpha k in the direction d k but simultaneously I am going to change my because alpha k is equal to delta k.

So, I am changing my error also by the same coefficient right. My error is reducing by the same amount right because alpha k is equal to delta k. So, building up I should not say the error is reducing its little vague statement but the same coefficient right the same coefficient in the error expression right. So, building up the solution by step by step as in x i plus 1 equal to x i plus alpha i d i is equivalent to reducing the error component by component right.

(Refer Slide Time: 25:52)



Minimization in n-steps

This is evident from subtracting the solution $\mathbf{x}$ from both sides of :

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_{i-1}\mathbf{d}_{i-1}$$

$$\mathbf{e}_i = \mathbf{e}_{i-1} + \alpha_{i-1}\mathbf{d}_{i-1} = \mathbf{e}_0 + \sum_{j=0}^{i-1} \alpha_j \mathbf{d}_j \quad \text{(by recursion)}$$

$$= \sum_{j=0}^{n-1} \delta_j \mathbf{d}_j - \sum_{j=0}^{i-1} \delta_j \mathbf{d}_j \quad \text{from (*) and (**)}$$

$$= \sum_{j=i}^{n-1} \delta_j \mathbf{d}_j$$

So, let us this is evident from let us let us see it a in a in a little more detail. So, we have we know this expression x i is equal to x i minus 1 plus alpha i minus 1 d i minus 1 that is always true right. That is yes my update rule and then if I subtract my true solution x from both sides of this equation on the left hand side I have e i on the right hand side I have x i minus 1 minus x, which is e i minus 1 plus alpha i minus 1 d i minus 1 but this I can then again I can expand e i minus 1 in the similar way right. Again, i can write e i minus 1 in terms of d i minus 2 right and. So, eventually I am going to end up with e 0 plus sigma j is equal to 0 to i minus 1 alpha j d j right, just by recursion right continuing this step.

So, I know e i equal to e i minus 1 plus alpha minus 1 d i minus 1 right. So, here I am going to put e i minus 1 is equal to e i minus 2 plus alpha i minus 2 d i minus 2 then again I am going to replace e i minus 2 similarly, right this continue this do recursion right. So, eventually I am going to get e 0 plus this term right. So, this then again e 0 i assume that I could write e 0 I know that I can write I did not assume that I know I can write e 0 in terms of delta j d j right, then n directions because they are orthogonal they form a basis.

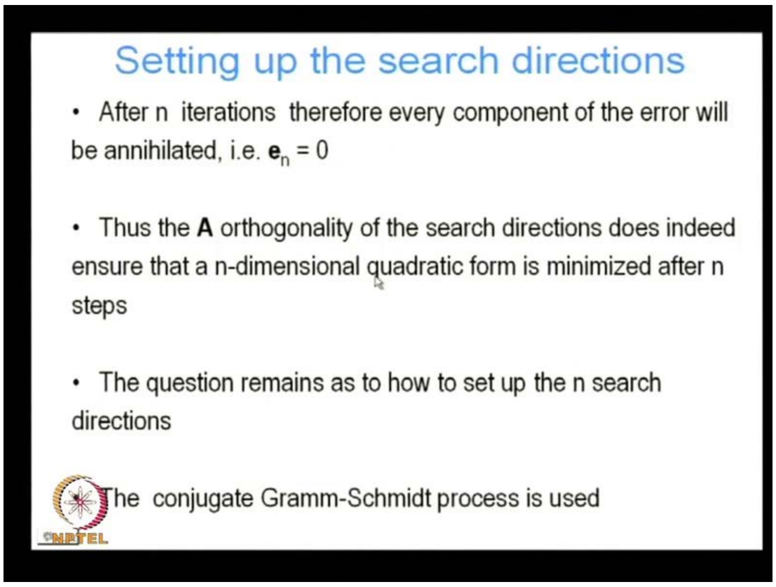
So, then let me replace that expression for e 0 here, see e 0 is equal to sigma j equal to 1 n minus 1 delta j d j. And now I will replace alpha j by minus delta j because already found that alpha i; i is equal to minus delta i. So, I replace alpha j by minus delta j. So, now, what do you have I have j equal to 0 to n minus 1 delta j d j which is my initial error minus this term minus this term.

So, I am left with the summation only from j equal to 1 to n minus 1 delta j d j. So, this is this is very revealing because what it says is that initially my error had components from all, in all the d d directions from j equal to 0 to n minus 1 d 0 d 1 d 2 d n d n minus 1 right. Now, after I steps after i minus 1 steps my error has components only in the directions d i to d i minus 1.

So, from 0 to i minus 1 all those components of e 0 have gone totally to 0 right they have been totally negated because now I can write the error at step i in terms of the basis vector i j is equal to i to n minus 1. So, it no longer has components in the basis vectors 0 to i minus 1 but remember my original e 0 had components in all these directions right from 0 1 2 3 up to n minus 1. So, the components in the directions 0 to i minus 1 have gone away right because of my iteration right.

So, this is this is this is this? So, this is the proof that if I do n iterations if I do n iterations my whole error is going to go away because all my components are going to go to 0 every time I take a step one component is going away right. So, eventually when I take n steps all my components are gone right. So, this is the proof that this conjugate method of conjugate directions is actually going to converge in n iterations right.

(Refer Slide Time: 30:06)



Setting up the search directions

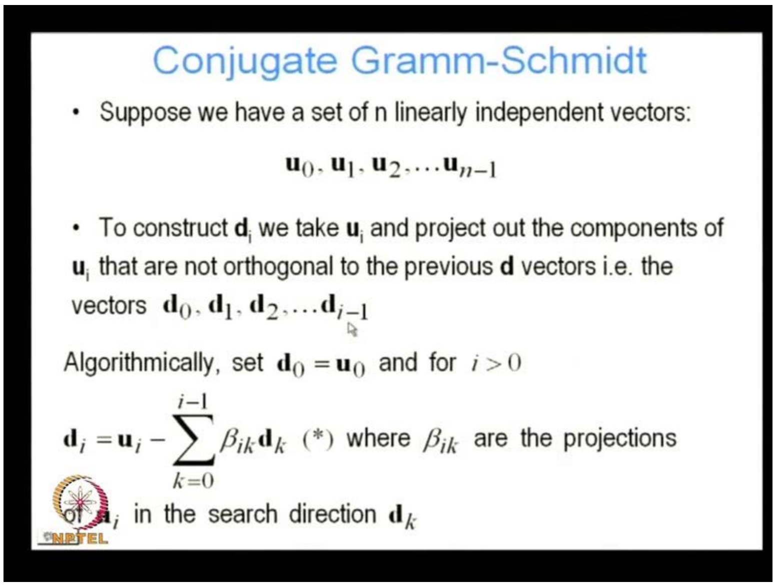
- After n iterations therefore every component of the error will be annihilated, i.e. $\mathbf{e}_n = 0$
- Thus the **A** orthogonality of the search directions does indeed ensure that a n-dimensional quadratic form is minimized after n steps
- The question remains as to how to set up the n search directions

The conjugate Gramm-Schmidt process is used

So, after n iterations therefore, every component of the error will be annihilated that is e n equal to 0 thus the A orthogonality of the search directions does indeed ensure that n dimensional quadratic form is minimized after n steps for. After all this is the only condition that we have used right that the search directions are A orthogonal to each other A orthogonality of the search directions ensures that n dimensional quadratic form is minimized after n steps the question we have not answered till now is that what are my directions right.

I say that there are n n directions they form an orthogonal right they are orthogonal; they form a basis in that space right; and they are also a orthogonal to each other right. So, but then how do I how do I find those search directions right. Well, I find the search directions by a method which is known as Gramm-Schmidt orthogonalization is a very common method in numerical analysis and I think which I have all which we have already encountered earlier in this course.

(Refer Slide Time: 31:24)



Conjugate Gramm-Schmidt

- Suppose we have a set of n linearly independent vectors:

$$\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \ldots \mathbf{u}_{n-1}$$

- To construct $\mathbf{d}_i$ we take $\mathbf{u}_i$ and project out the components of $\mathbf{u}_i$ that are not orthogonal to the previous **d** vectors i.e. the vectors $\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \ldots \mathbf{d}_{i-1}$

Algorithmically, set $\mathbf{d}_0 = \mathbf{u}_0$ and for $i > 0$

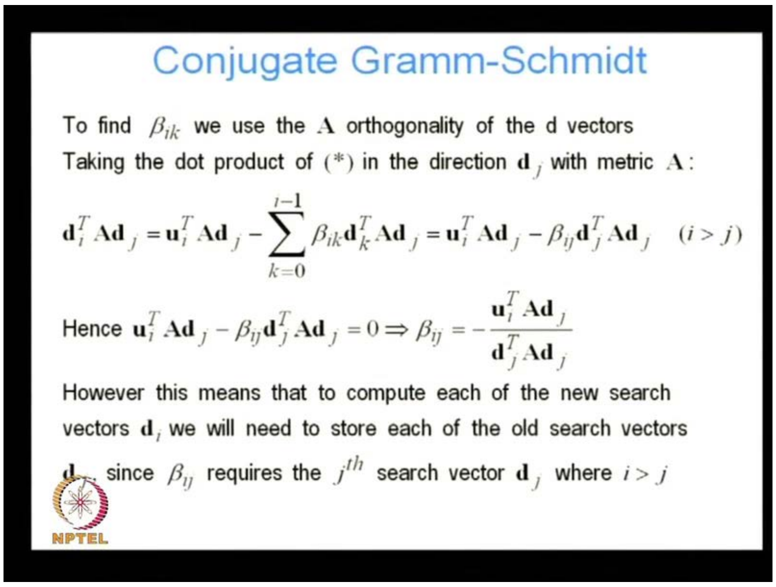
$\mathbf{d}_i = \mathbf{u}_i - \sum_{k=0}^{i-1} \beta_{ik} \mathbf{d}_k$ (*) where β_{ik} are the projections of $\mathbf{u}_i$ in the search direction $\mathbf{d}_k$

So, we use A the conjugate Gramm-Schmidt process. So, what does it say well it says that if I start with any linearly independent vectors note that these vectors need not be orthogonal they need only be linearity independent right then linearly independent to construct d i, to construct any search direction d i we take u i and project out the components of u i in the previous search directions. That is going to result in u i after i remove all those components whatever is left in u i is going to be orthogonal to all my previous search directions and that transformed u i will become a new search direction right.

So, construct d i is which take u i project out the components of u i that are not orthogonal to the previous d d vectors right those which have components along those along the previous d vectors I have removed and the previous d vectors meaning the vectors d 0 d 1 d 2 to d i minus 1 right and whatever is left is assured to be orthogonal to all the these vectors d 0 d 1 d 2 d i minus 1.

So, what is the algorithm well we start with d 0 equal to u 0 because when we I have d 0 there is there are no search directions that is I am just starting my algorithm right. So, I just said u 0 equal to d 0 and then when I move to the next one next iteration when I want to compute my next search direction I make sure that I subtract I start with u one then and I take away from u one the projection of d of d 0 in the u one direction right. So, d i is equal to u i minus sigma k equal to 0 i minus 1 beta i k d k, where beta i k are the projections of the u i in the search directions d k. So, d k k equal to 0 to i minus 1 ,which is very important to note right I am taking the projection with respect to the previous search directions right up to i minus 1, I am taking the projection.

(Refer Slide Time: 33:40)



Conjugate Gramm-Schmidt

To find β_{ik} we use the $\mathbf{A}$ orthogonality of the d vectors

Taking the dot product of $(*)$ in the direction $\mathbf{d}_j$ with metric $\mathbf{A}$:

$$\mathbf{d}_i^T\mathbf{A}\mathbf{d}_j = \mathbf{u}_i^T\mathbf{A}\mathbf{d}_j - \sum_{k=0}^{i-1}\beta_{ik}\mathbf{d}_k^T\mathbf{A}\mathbf{d}_j = \mathbf{u}_i^T\mathbf{A}\mathbf{d}_j - \beta_{ij}\mathbf{d}_j^T\mathbf{A}\mathbf{d}_j \quad (i > j)$$

Hence $\mathbf{u}_i^T\mathbf{A}\mathbf{d}_j - \beta_{ij}\mathbf{d}_j^T\mathbf{A}\mathbf{d}_j = 0 \Rightarrow \beta_{ij} = -\dfrac{\mathbf{u}_i^T\mathbf{A}\mathbf{d}_j}{\mathbf{d}_j^T\mathbf{A}\mathbf{d}_j}$

However this means that to compute each of the new search vectors $\mathbf{d}_i$ we will need to store each of the old search vectors $\mathbf{d}_j$ since β_{ij} requires the j^{th} search vector $\mathbf{d}_j$ where $i > j$

To find beta i k we use the A orthogonality of the d vectors. So, take the dot product of star this expression in the direction in the direction d j again with the metric a right. So, we are going to enforce A orthogonality right. So, I take d i transpose, d i transpose A d j. So, that u i transpose A d j minus sigma k equal to 0 i minus 1 beta i k d k transpose A d j right.

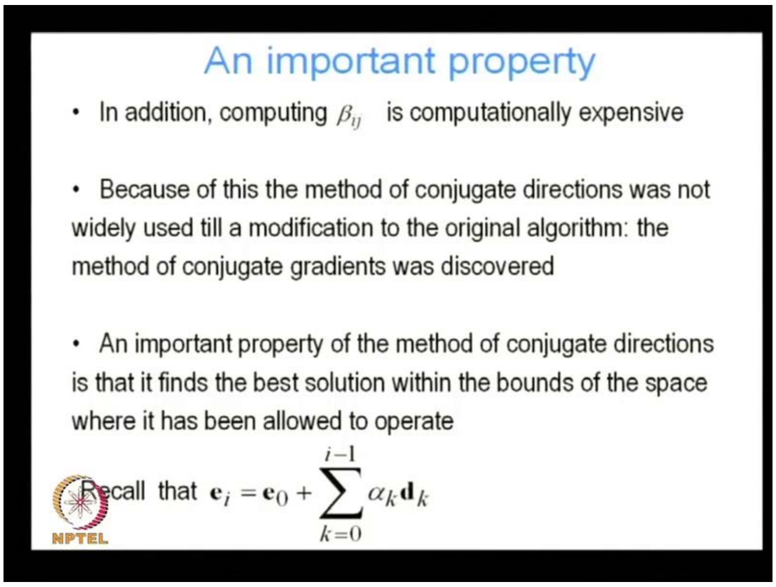
So, this part remains the same but this part I use the A orthogonality of the direction vectors and it and I end up only d j transpose A d j for i greater than j right. Because, the summation is here from k equal to i minus 1 right. So, all these terms are going to go to 0 for i greater than j right, if I if i greater than j since they do not contain they are all less than i minus 1 up to i mean less than i. So, all those terms are going to away only the term that is going to survive is d j transpose A d j right.

So, this gives me that, So, this allows me to calculate beta i j from this expression right from this expression i get beta i j equal to minus u i transpose A d j divided by d j transpose A d j what does it means it means that to compute each of these beta i j s. I have to know all my previous search directions right suppose I am computing my third my third search direction right I am computing my third search direction. So, I know d 0 I know d 1 I want to compute d 2 right. I want to compute d 2. So, let us look at this equation again. So, I will to compute d 2, but then I have to take this sum from k a equal to 0 to 1 right. So, here what do I have I have beta 2 1 d 1 plus minus beta 2 2 d 2 right.

So, not only so but this beta 2 1 and beta 2 2 they involve my vectors d 1 they would we have we want to calculate not only beta 2 1, we also need to calculate beta 2 0 right. So, they involve not they involve my vectors one and 0 right and because this is j this is j right. So, they involve the vectors d 0 as well as d 1. So, at any point after 10000, if I have a huge problem right and I my metrics is of size 20000 by 20000, after 10000 steps after 10000 steps how many directions do I have I have 10000 directions right. So, I have to skip those all those 10000 directions to evaluate my new direction to evaluate to my 10001 one th direction I have to keep all those directions and then I have to perform this operation right I have to perform this operation to calculate my coefficients and then use my equation here right.

So, that is not a good idea means, it can be done, but it is not feasible in any real problem with large problem if you have to store all those vectors and at each step we have to do all those multiplications and all those divisions that is going to be awfully expensive right. So, that is a big disadvantage of the method of conjugate directions; that is a disadvantage of the method of conjugate directions and when we go to the method of conjugate gradients we will see that the beauty of the method of conjugate gradients is that it overcomes this problem. It gets around this problem right. So, it gets around the problem of storing all the previous search directions and having to compute all these coefficients every time we do a Gramm-Schmidt orthogonalization to find the new search direction right.

(Refer Slide Time: 38:32)



An important property

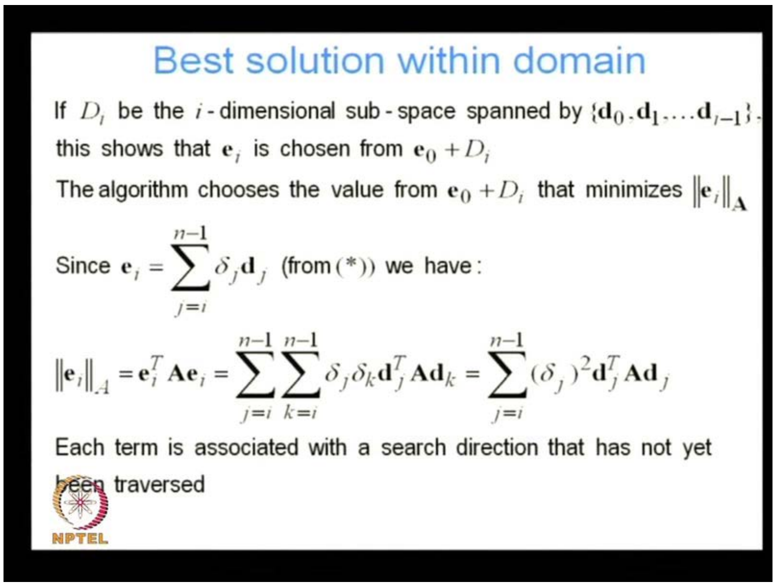
- In addition, computing β_{ij} is computationally expensive
- Because of this the method of conjugate directions was not widely used till a modification to the original algorithm: the method of conjugate gradients was discovered
- An important property of the method of conjugate directions is that it finds the best solution within the bounds of the space where it has been allowed to operate

Recall that $\mathbf{e}_i = \mathbf{e}_0 + \sum_{k=0}^{i-1} \alpha_k \mathbf{d}_k$

So, we have already talked about that computing beta i j is computationally expensive. Because of this the method, of conjugate directions was not widely used till a modification to the to the original algorithm the method of conjugate gradients was discovered right. So, until we discovered the method of conjugate gradients sometime in the early part and the middle part of last century this method of conjugate directions was known for a long time but people did not use it because of this drawback right. So, so we will talk about the method of conjugate gradients which allows us to circumvent this problem

But before doing that I want talk about another important property of the method of conjugate gradients sorry in method of conjugate directions which is that it finds the best solution within the bounds of the space where it has been allowed to operate what do we mean by that? Well, let us see well we know that at any iteration i e i is equal to e 0 plus k sigma k equal to 0 i minus 1 alpha k d k we have already seen that how we got that right.

(Refer Slide Time: 39:47)



Best solution within domain

If D_i be the i-dimensional sub-space spanned by $\{\mathbf{d}_0, \mathbf{d}_1, \ldots \mathbf{d}_{i-1}\}$,
this shows that $\mathbf{e}_i$ is chosen from $\mathbf{e}_0 + D_i$

The algorithm chooses the value from $\mathbf{e}_0 + D_i$ that minimizes $\|\mathbf{e}_i\|_{\mathbf{A}}$

Since $\mathbf{e}_i = \sum_{j=i}^{n-1} \delta_j \mathbf{d}_j$ (from (*)) we have:

$$\|\mathbf{e}_i\|_A = \mathbf{e}_i^T \mathbf{A} \mathbf{e}_i = \sum_{j=i}^{n-1} \sum_{k=i}^{n-1} \delta_j \delta_k \mathbf{d}_j^T \mathbf{A} \mathbf{d}_k = \sum_{j=i}^{n-1} (\delta_j)^2 \mathbf{d}_j^T \mathbf{A} \mathbf{d}_j$$

Each term is associated with a search direction that has not yet been traversed

Now, let us suppose d i be the i dimensional subspace spanned my directions d 0 to d i minus 1. So, I have I am looking at step I now I have already travelled in the direction 0 1 2 up to i minus 1 and I am denoting that subspace that and I know that all these vectors are they form a basis right. So, the subspace that is spanned by these basis vectors right, I am going to denote by d i right and what my equation; my equation here tells me is that the error at step i is formed is a combination can be written as a combination of my basis vectors d 0 to d i minus 1. Basically my subspace, d i and my original error vector e 0 right shows that e i is chosen from e 0 plus d i. Now the algorithm, the algorithm chooses the value from e 0 plus d i that minimizes the norm of the error e i in the when I say norm i define norm with respect to the A; A metric right.

So, we can show that well how can we show that well since e i is equal to sigma j equal to i to delta j d j right I can always right that because my d j my basis d j they form a j equal to i to n minus 1 they form a basis right and we have already shown that right. That is not really we showed that sometime exactly here right we showed that exactly because the errors are all the i, i minus 1 component have been annihilated right. So, because of that i can write this d j i equal j equal to i to n minus 1 form a basis for my e i right I can write the e i as a linear combination of those limited set of d j s right that limited set of this is this is the very important that limited set of d j s right.

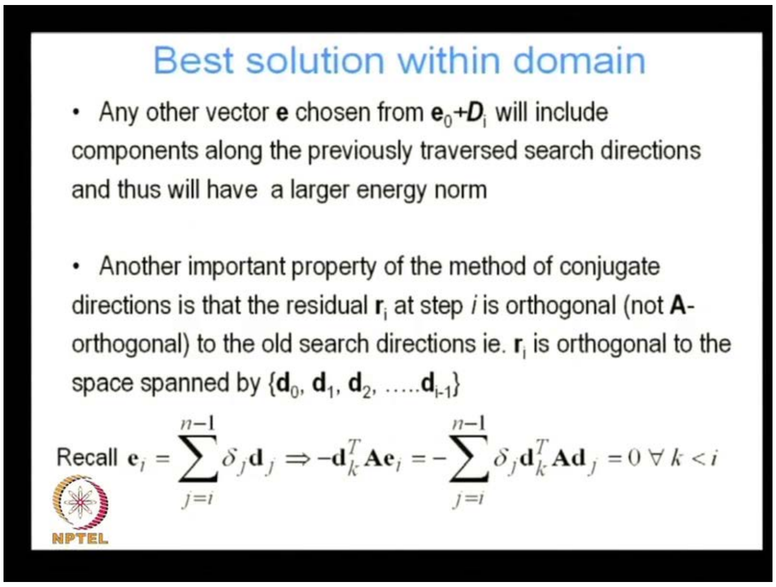
So, every time the basis it is it is a very beautiful thing right everytime I am taking an iteration I had originally my basis was n dimensional. So, every time I am taking iteration my basis is changing. So, basis is again reduced by one. So, my space the space that is spanned by the basis vectors is also getting reduced by one. So, you think of it in simply in 3D. I start with a three dimensional vector it becomes a two dimensional vector first iteration, next iteration it becomes a scalar and then I am done right.

So, basically we can write e i equal to sigma j equal to i to n minus 1 delta j d j from which we have already seen. Then if I take calculate the norm of this error in with respect to the A metric. So, basically I have evaluate this modified inner product, which is my new definition of a dot product right e i transpose A e i. So, I take the dot product of this with itself.

So, I have delta j delta k d j transpose A d k right again, I am going to use the A orthogonality of the direction vectors and I am going to end up with sigma j equal to i to n minus 1 delta j square d j transpose A d j right. So, now, what does it tell me? Each term is associated each term in this expression is associated with the search direction that has not yet been travelled right.

So, all the remaining error all the remaining error after i iterations is associated with the search direction, which is not been travelled right because what is the what is the j index varying from it is varying from i to n minus 1 right each term is associated with the search direction that is not yet been traversed if d i with the i dimensional this I have already talked about right.

(Refer Slide Time: 44:25)



Best solution within domain

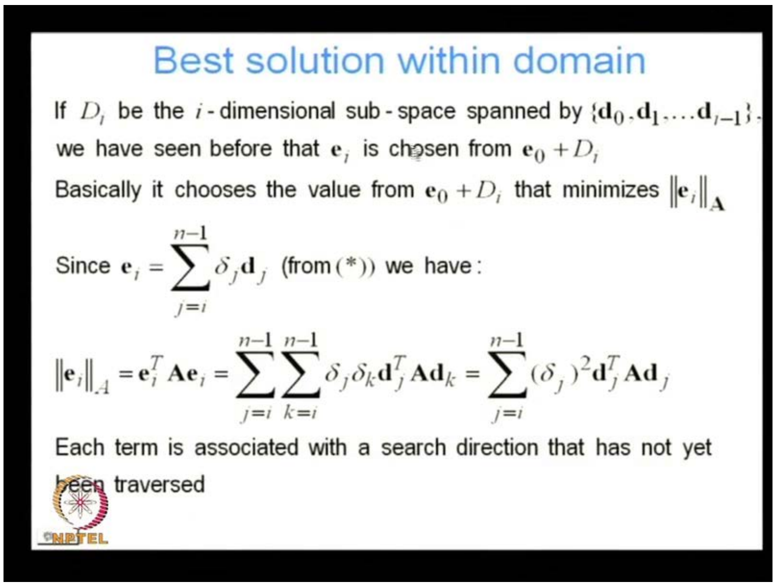
- Any other vector **e** chosen from $\mathbf{e}_0+D_i$ will include components along the previously traversed search directions and thus will have a larger energy norm

- Another important property of the method of conjugate directions is that the residual $\mathbf{r}_i$ at step i is orthogonal (not **A**-orthogonal) to the old search directions ie. $\mathbf{r}_i$ is orthogonal to the space spanned by $\{\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \ldots..\mathbf{d}_{i-1}\}$

Recall $\mathbf{e}_i = \sum_{j=i}^{n-1} \delta_j \mathbf{d}_j \Rightarrow -\mathbf{d}_k^T \mathbf{A}\mathbf{e}_i = -\sum_{j=i}^{n-1} \delta_j \mathbf{d}_k^T \mathbf{A}\mathbf{d}_j = 0 \;\forall\, k < i$

So, any other vector e chosen from e 0 plus d i will include components along the previously traversed search directions and thus will have a larger energy norm why?

(Refer Slide Time: 44:50)



Best solution within domain

If D_i be the i-dimensional sub-space spanned by $\{\mathbf{d}_0, \mathbf{d}_1, \ldots \mathbf{d}_{i-1}\}$, we have seen before that $\mathbf{e}_i$ is chosen from $\mathbf{e}_0 + D_i$

Basically it chooses the value from $\mathbf{e}_0 + D_i$ that minimizes $\|\mathbf{e}_i\|_{\mathbf{A}}$

Since $\mathbf{e}_i = \sum_{j=i}^{n-1} \delta_j \mathbf{d}_j$ (from (*)) we have:

$$\|\mathbf{e}_i\|_A = \mathbf{e}_i^T \mathbf{A}\mathbf{e}_i = \sum_{j=i}^{n-1}\sum_{k=i}^{n-1} \delta_j \delta_k \mathbf{d}_j^T \mathbf{A}\mathbf{d}_k = \sum_{j=i}^{n-1} (\delta_j)^2 \mathbf{d}_j^T \mathbf{A}\mathbf{d}_j$$

Each term is associated with a search direction that has not yet been traversed

e i is chosen from e 0 plus d i right d i has all these components d 0 d 1 d i minus 1 right. But eventually, we see that e i e i has got components it does not have any components in d 0 d one d i minus 1 right it has got only components in the basis from i to n minus 1 right. but if I consider any other vector any other vector e, which is chosen from e 0 plus d i it will include components along the previously traversed search directions right because this d i consists of d 0 d one up to d i minus one.

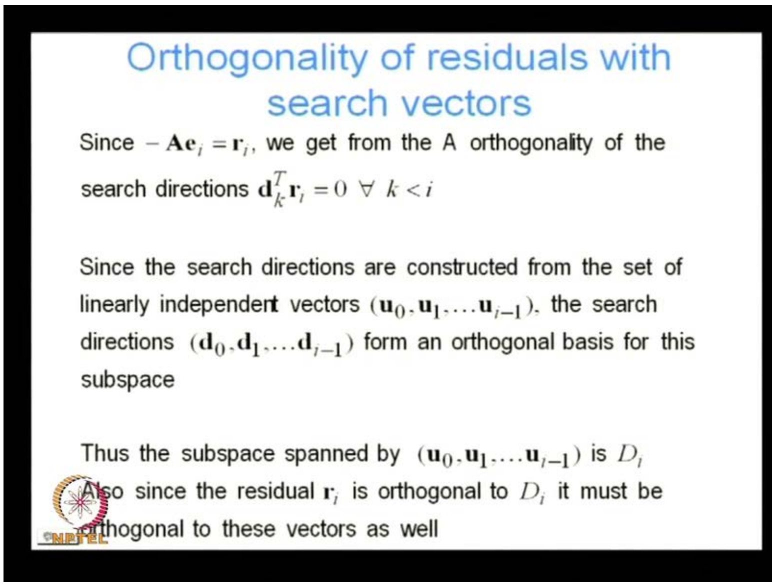
So, if I chose any vector; any arbitrary vector from that search from that space that subspace which is basically d i plus e 0 that may have that is going to have a component along the previous directions right. But, my e i which also belongs to that exact same subspace which I just showed the earlier right e i is chosen from e 0 plus d i e i; however, has got no components along the previous along d i right along in the subspace d i right.

So; that means, that e i, e i is the minimum right because it does not have any components along the d I s right. Any other vector from that subspace is going to have components along those directions right. So, any other vector e chosen from e 0 plus d i will include components along the previously traversed search directions and thus it will have a larger norm why will it have a larger norm, because it will include not only this j equal to i to n minus 1 it will also include terms involving i minus 1, i minus 2 and So on and So forth right. So, this does not include any components of in the subspace spanned by d i right with.

So, this basically shows us what does it show us? what we intended to show was that not that let us see now where did we state that no that, the method of conjugate directions finds the best solution within the bounds of the space where it has been allowed to operate right. Because, it finds the best solution because it gives me the minimum error so, up till now I have allowed it to operate in this space d i right. It I have allowed it to operate in the space d 0 e 0 which I started with and d i right. So, in that subspace it gave me the best solution right. So, that that is the idea.

So, let us talk about another important property of the method of conjugate directions which is that the residual at step i the r i is orthogonal not a orthogonal in this case it is orthogonal to the old search directions that is r i is orthogonal to the space spanned by d 0 d 1 d 2 to d i minus 1, which I called d i right; which is basically the same thing as d i. So, the at the residual at iteration i is orthogonal to the space spanned by these vectors. Well, how can we show it; well look at this recall we have e i equal to sigma j equal to one n minus 1 delta j d j that is e i has components only in the vectors stretching, it does not have any components in the 0 to i minus 1, which implies if i take d k transpose A e i i get delta j d k transpose A d j and this is going to be 0 for all k less than i right for all k less than i this is going to be 0.

(Refer Slide Time: 49:44)



Orthogonality of residuals with search vectors

Since $-\mathbf{A}\mathbf{e}_i = \mathbf{r}_i$, we get from the A orthogonality of the search directions $\mathbf{d}_k^T \mathbf{r}_i = 0 \ \forall \ k < i$

Since the search directions are constructed from the set of linearly independent vectors $(\mathbf{u}_0, \mathbf{u}_1, \ldots \mathbf{u}_{i-1})$, the search directions $(\mathbf{d}_0, \mathbf{d}_1, \ldots \mathbf{d}_{i-1})$ form an orthogonal basis for this subspace

Thus the subspace spanned by $(\mathbf{u}_0, \mathbf{u}_1 \ldots \mathbf{u}_{i-1})$ is D_i

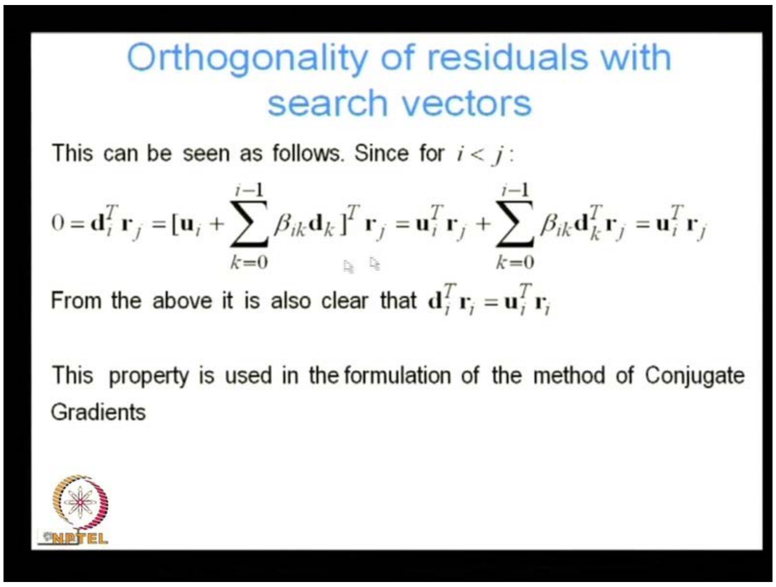
Also since the residual $\mathbf{r}_i$ is orthogonal to D_i it must be orthogonal to these vectors as well

But since a minus A e i is equal to r i we get from the a orthogonality of the search directions d k transpose r i is equal to 0 right. Because this A, e i this A e i is nothing but r i right. So, I get minus d k transpose r i is equal to 0 for all k less than i. So, at residual i at i the residual r i is orthogonal to all the k for which k is less than i; that means, that is orthogonal to all the previous search directions right. So, residual is orthogonal to all the previous search directions then since the search directions are constructed from the set of linearly independent vectors u 0 through u i minus 1 this means that the search directions d 0 d i d i minus 1 form an orthogonal basis for this subspace thus this subspace spanned by u 0 u i u minus 1 is d i.

So, basically I am saying that I got my d 0 through d i minus 1 from my i minus 1 linearly independent vectors u 0 u i t i minus 1 right, but I know that my d 0 d 1 d i minus 1 forms an orthogonal basis right, which forms an orthogonal basis. So, this is an orthogonal basis for the space u 0 u i minus 1 and therefore, the subspace spanned by u 0 u i minus 1 is actually also d i is that clear I do not know if it is clear it is what I am saying is that my d i s i formed from my u 0 u i my u i minus 1 vectors right.

So, the subspace spanned by those mutually independent mutually independent i vectors u 0 u 1 to u i minus 1 must be identical must be identical to the subspace that is spanned by my d vectors right d 0 d 1 d i minus 1 right. So, those two subspaces must be orthogonal and since r i is the orthogonal to the d 0 d i vectors right so; that means, what r i is orthogonal to that subspace spanned by those vectors right r i is orthogonal to any vectors which can be found by linear combination of those d 0 d i d d i minus 1 vectors; that means, that r i must be orthogonal to also my u vectors right. It must be orthogonal to my u vectors because I can always write those u vectors as a combination of those d 0 d d 1 d i minus 1 vectors right.

(Refer Slide Time: 52:38)



Orthogonality of residuals with search vectors

This can be seen as follows. Since for $i < j$:

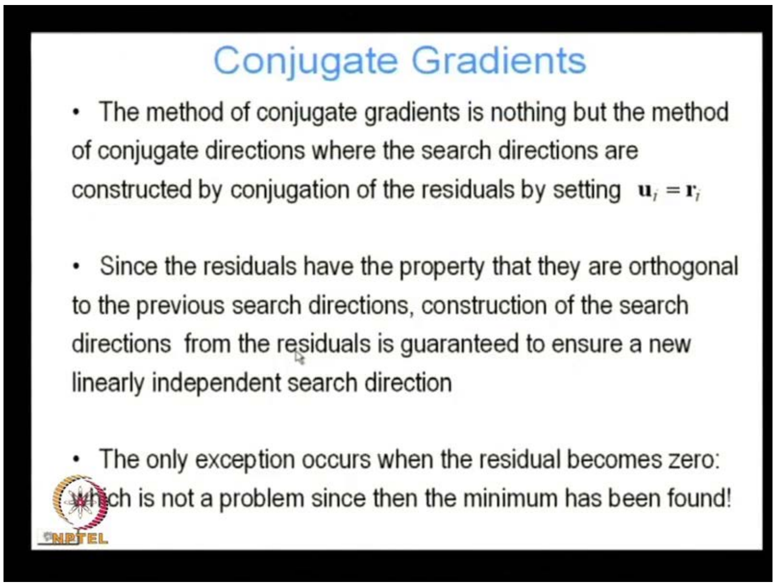
$$0 = \mathbf{d}_i^T \mathbf{r}_j = [\mathbf{u}_i + \sum_{k=0}^{i-1} \beta_{ik} \mathbf{d}_k]^T \mathbf{r}_j = \mathbf{u}_i^T \mathbf{r}_j + \sum_{k=0}^{i-1} \beta_{ik} \mathbf{d}_k^T \mathbf{r}_j = \mathbf{u}_i^T \mathbf{r}_j$$

From the above it is also clear that $\mathbf{d}_i^T \mathbf{r}_i = \mathbf{u}_i^T \mathbf{r}_i$

This property is used in the formulation of the method of Conjugate Gradients

So, r i is orthogonal to those u vectors well we can show that. So, 0 is equal to d i transpose r j which we have already shown right that the residual r j must be orthogonal to all the previous d i s right for i less than j. So, then we write d i in terms of this expression right how did we get that expression with this is, how I calculated my d i from Gramm-Schmidt Orthogonalization right. So, i replaced that d i by this expression I get u i transpose r j u i transpose r j plus this term which is k equal to 0 to i minus 1 beta i k d k transpose r j with this is going to be 0 because k goes from 0 to i minus 1 and j i know is greater than i. So, this has got to be 0. So, this term has go goes to 0. So, I have u i transposed r j. So, from this I get d i transpose r i is equal to u i. So, d i transpose r j is equal to u i transpose r j. So, it also means that d i transpose r i is equal to u i transpose r i right in case right. Then this property is used in the formulation of the method of conjugate gradients.

(Refer Slide Time: 54:04)



Conjugate Gradients

- The method of conjugate gradients is nothing but the method of conjugate directions where the search directions are constructed by conjugation of the residuals by setting $\mathbf{u}_i = \mathbf{r}_i$
- Since the residuals have the property that they are orthogonal to the previous search directions, construction of the search directions from the residuals is guaranteed to ensure a new linearly independent search direction
- The only exception occurs when the residual becomes zero: which is not a problem since then the minimum has been found!

So, now we are ready to talk about the method of conjugate gradients and we will see that the method of conjugate gradients is nothing but the method of conjugate directions. But with one very important difference and that important difference gives the method of conjugate gradients all its power it results in my not having to store all my previous direction vectors. when I compute my new direction vectors leads to huge savings right and it allows the method of conjugate gradients to be used to solve very large problems, problems which cannot be solved by normal direct solver right if I have a one million by one million system. I probably even if I use gauss elimination if I use all the efficiencies I can must a probably will have a hard time solving it right in most computers. But the method of conjugate gradients can be used to solve that system and that is precisely, because that it has overcome the debility with the method of conjugate directions right.

Thank you.