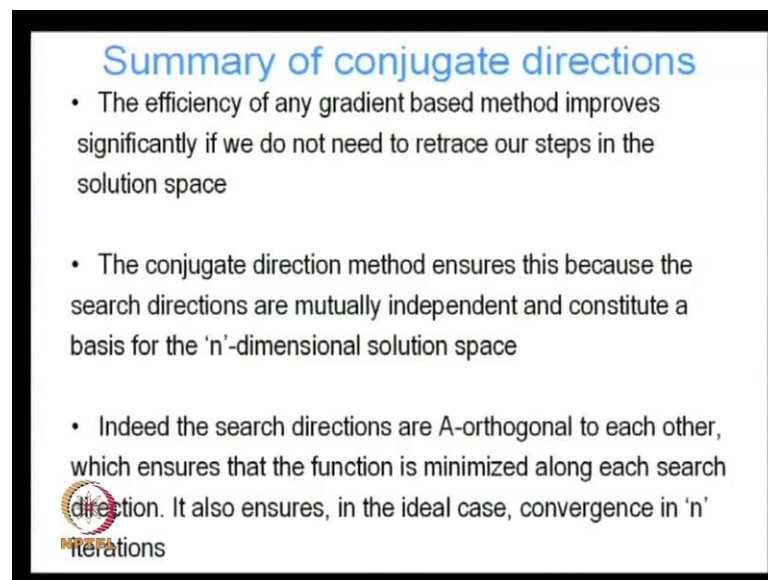


Numerical Methods in Civil Engineering
Prof. Arghya Deb
Department of Civil Engineering
Indian Institute of Technology, Kharagpur

Lecture - 9
Conjugate Gradient Method – II

On numerical methods in civil engineering we will continue with our discussion on conjugate gradient methods. Last time, we looked at the method of conjugate directions and at the end of the lecture; we said that the conjugate gradient method is just a specialisation of the method of conjugate directions.

(Refer Slide Time: 00:39)



Summary of conjugate directions

- The efficiency of any gradient based method improves significantly if we do not need to retrace our steps in the solution space
- The conjugate direction method ensures this because the search directions are mutually independent and constitute a basis for the 'n'-dimensional solution space
- Indeed the search directions are A-orthogonal to each other, which ensures that the function is minimized along each search direction. It also ensures, in the ideal case, convergence in 'n' iterations

So, what was the method of conjugate directions; well, with the why do we need conjugate directions well we started with the postulate that the efficiency of any gradient based method improve significantly if we do not need to retrace our steps in the solution space right. That that was not in that was not a feature of the steepest gradient method; that is steepest descent method that is why we found that we often have to retrace our steps.

But the advantage of the method of conjugate directions was that the steps we took we do not have to take those steps again; we never retrace our steps the steps we take are unique and we take them once only. The conjugate direction method ensures this because the search directions are mutually independent and constitute a basis for the 'n'-


dimensional solution space. Indeed we found that the search directions are A orthogonal to each other, where A is some sort of metric in that solution space right a measure of distance in the solution space; which with which we define the inner product in the solution space and therefore, the norming the solution space.

So, these search directions are A orthogonal to each other which ensures that the function is minimised along each search directions. So, we found that. So, long as soon as we ensure A orthogonality then that automatically ensures that the function is minimised along a search directions and we further found that when we take subsequent step; subsequent search directions; subsequent iterations there are in the in a previously traversed search direction never increases right. So, we systematically keep on reducing the error in each of those search directions right and they never they never recur again and because of this we had we found that in the that the conjugate gradient method it is a conjugate directions method is assured to converge in n iterations.

Right it is sure to converge in n iterations but that is true in the ideal case and the real world nothing is as we plan because of accumulation of round of errors things like that the conjugate directions will no longer be exactly satisfy the orthogonality condition right. Because of that convergence, will be somewhat lesser the convergence rate will be somewhat lesser right we will talk about those things specifically in the context of the conjugate gradient method.

(Refer Slide Time: 03:23)

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!

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 u_i is equal to r_i . First let us take a step back, so when we discussed the conjugate directions method, I pointed out that a major disadvantage of the conjugate directions method is that at each step; at each iteration i need all the previous directions i need all if at a iteration i i need all the directions from 0 to i minus 1. In order to find m_i in order to do my Gram-Schmidt orthogonalization right and because of that we have to store all those directions we have to carry all those directions around and at each iteration we have to come take the projection along those directions.

So, its for have large problem the computational cost is enormous and for the conjugate gradient method we said its great advantage would be that we do not need to carry the old directions around why is that? So, we will see why that right is? so the one the first step the key step in achieving that is to make sure that we choose our search directions from the residual remember earlier in the conjugate direction method we had this set of vectors u_i , which formed linearly independent set in my n dimensional space right and then I use to compute my conjugate directions from those set of vectors u_i but each step doing Gram-Schmidt orthogonalization projecting out the part which is parallel to or which has got which has got any components along the previous directions retaining only the part which is orthogonal to the previous directions right.

That is what we did for the conjugate gradient for the conjugate directions method now the starting point of the conjugate gradient method is saying that well that set u_i is not just any arbitrary set of vectors in my n dimensional space any arbitrary set of linearly independent vectors in my n dimensional space they are a specific set of vectors and what are those specific set of vectors they are my residuals right.

So, the residuals I am going to chose my conjugate directions from my residual vectors right why do we do that well the advantage is that residuals have the property that they are orthogonal to the previous search directions we obtained that at the end of our last lecture we obtained this result that each residual at step at iteration step i is orthogonal to all the previous search directions 0 1 2 3 up to i minus 1 right.

So, since the residuals have the property that they are orthogonal to the previous search directions construction of the search directions from the residual is guaranteed to ensure

that the new search direction is orthogonal is linearly is a linearly independent search direction. It is a new linearly independent search direction right. So, because the residuals have this wonderful property that they are orthogonal to all the previous search directions. So, if I construct my new search direction at step i from my residual it becomes a lot simpler right because of that orthogonality property when will this process breakdown well when the residual becomes 0. Suppose during my iteration, my residual becomes 0 then I cannot construct my new search direction from the residual but then that is not a problem why because the residual is 0; that means, my iteration has converged right. I have reached the true solution So, I do not need the residual any more I do not need any more search directions right.


(Refer Slide Time: 07:13)

Conjugate Gradients

- Since the search directions are constructed from the residuals, the subspace spanned by $\{r_0, r_1, \dots, r_{(i-1)}\}$ is identical to the subspace spanned by $\{d_0, d_1, \dots, d_{(i-1)}\}$

Recall that we showed earlier that residual r_i is orthogonal to all previous search directions d_j i.e. $r_i^T d_j = 0 \quad \forall i > j$.

Hence $r_i^T r_j = 0 \quad \forall i > j$, indeed $r_i^T r_j = 0 \quad \forall i \neq j$, since the subsequent residuals also have to be normal to the space spanned the previous search vectors, i.e. the space spanned by the previous residuals.



So, since the search directions are constructed from the residuals, the subspace spanned by r_0, r_1 through r_{i-1} is identical to the subspace spanned by d_0, d_1 through d_{i-1} right because each of those search directions are constructed from the residuals. So, whatever be the space spanned by the residuals right that is the same as the subspace as the space spanned by the search directions right. Recall that we showed earlier that the residual r_i is orthogonal to all previous search directions d_j that is $r_i^T d_j = 0$ for all $i > j$ right for all $j < i$ right. Hence, $r_i^T r_j = 0$ for all $i > j$, why because r_i is because r_i is orthogonal to all the d_j right. And the space spanned by the d_j is equal to the space spanned by the r like the previous r like r_0 through r_{i-1} right.

So; that means, each of those residuals must be orthogonal to the previous residuals right. So, each of those residuals are orthogonal to the r for the previous residuals indeed $r_i^T r_j$ is equal to 0 for all $i \neq j$ because the subsequent residuals are also going to be orthogonal to the current residuals. So, basically all the r residuals are going to be orthogonal right. Subsequent residuals, also have to be normal to the space spanned by the previous search vectors meaning the previous such residuals right. So, $r_i^T r_j$ equal to 0 for all $i \neq j$.

(Refer Slide Time: 09:00)


Conjugate Gradients

Recall $r_i = -Ae_i = -A(x_i - x) = -A(x_{i-1} + \alpha_{i-1}d_{i-1} - x)$
 $= -Ae_{i-1} + \alpha_{i-1}Ad_{i-1} = r_{i-1} + \alpha_{i-1}Ad_{i-1}$

Thus each residual is a linear combination of the previous residual and Ad_{i-1} .

Since both r_{i-1} and d_{i-1} belong to subspace D_i , the subspace D_{i+1} is obtained by combining the spaces D_i and AD_i .

Hence, by recursion, D_i is the subspace spanned by the bases $\{d_0, Ad_0, A^2d_0, \dots, A^{i-1}d_0\}$ or equivalently the subspace with bases $\{r_0, Ar_0, A^2r_0, \dots, A^{i-1}r_0\}$



Let us recall that r_i is equal to minus Ae_i so we obtained that result earlier right so; that means; r_i is equal to minus $A(x_i - x)$ the iterate value minus the true solution x , which is equal to minus $A(x_{i-1} + \alpha_{i-1}d_{i-1} - x)$. I have just used the update formula for my x_{i-1} and then I put x_{i-1} and x together that gives me e_{i-1} . So, I have minus $Ae_{i-1} + \alpha_{i-1}Ad_{i-1}$ and again Ae_{i-1} is nothing but minus r_{i-1} .

So, I have $r_{i-1} + \alpha_{i-1}Ad_{i-1}$. So, what does that tell us? That tells us that each residual is a linear combination of the previous residual and Ad_{i-1} . So, each residual is a linear combination of the previous residual and the vector which I get by taking the product of A and the previous search direction d_{i-1} . Now, since both r_{i-1} and d_{i-1} belong to subspace D_i recall what is the subspace D_i , the subspace D_i consists of all the search directions d_0, d_1, d_2, d_3 up to d_{i-1} right

up to d_{i-1} . and since we know that r_{i-1} and d_{i-1} r 's and d 's belong to the same subspace right.


So, both r_{i-1} and d_{i-1} belong to the subspace d_i right the subspace d_{i+1} is obtained by combining the spaces d_i and Ad_i why just look at this equation right r_i is equal to r_{i-1} which we know belongs to the subspace d_i plus A times d_{i-1} . So, it is d_{i-1} again belongs to the space d_i . So, if I have the space d_i and I operate on that space with the A matrix right then I get another space Ad_i and I put those two spaces together I get my new space d_{i+1} to which my r_i as well as my d_i are going to belong right.

So, hence by recursion D_i is the subspace spanned by the. So, I go on doing this right. So, d_{i+1} is nothing but d_i plus Ad_i right d_{i+1} is equal to d_i plus Ad_i . Similarly, we do we doing that? So, eventually we can see that any space d_i is spanned by the basis $d_0, Ad_0, A^2d_0, \dots, A^{i-1}d_0$ because every time we operate with A and D on the previous d_i on d_{i-1} we get d_i right. So, we continue this and by recursion we can see d_i you can write it as like this as spanned by these vectors right or equivalently the subspace with basis r_0, Ar_0, A^2r_0, \dots and So, on because these are the same subspaces.

(Refer Slide Time: 12:26)

Advantage of Krylov subspace

- Such subspaces created by repeatedly applying a matrix to a vector are known as Krylov subspaces.
- Since r_{i+1} is orthogonal to r_i , r_{i+1} is orthogonal to D_{i+1} . But AD_i is included in D_{i+1} . Hence r_{i+1} is orthogonal to AD_i
- This makes finding the new search direction d_{i+1} from r_{i+1} simple since r_{i+1} is orthogonal to AD_i and hence A -orthogonal to $D_i = \{d_0, d_1, \dots, d_{i-1}\}$ i.e. to all previous directions except d_i

 Thus the Gramm –Schmidt procedure need only ensure A -orthogonality with d_i

So, these subspaces which are created by repeatedly applying A matrix to a vector are known as Krylov subspaces. So, I start with a single vector and I operate on that vector

with a matrix and then I operate again with that same matrix and keep on doing it right and I am assured that those made those vectors that I form that I get by operating each time are linearly independent they are a basis right. So, that sort of subspace is known as a Krylov subspace right since r_{i+1} now. So, that that is the just a matter of terminology let us take a step back since r_{i+1} is orthogonal to r_i which we know right from our little result out here right, from my little result out here r_{i+1} is equal to sorry r_{i+1} is orthogonal to r_i ; that means, r_{i+1} must be orthogonal to D_{i+1} why D_{i+1} has what are those what are the vectors which are the basis of D_{i+1} r_0 r_1 through r_{i-1} right.

Those through r_i right D_{i+1} through r_i right. So, since r_{i+1} is orthogonal to r_i . So, r_{i+1} must be orthogonal to D_{i+1} right, but $A d_i$ is included in D_{i+1} . We just found that right because D_{i+1} is nothing but right. So, so hence $A r_{i+1}$ must be orthogonal to $A D_i$ right since r_{i+1} is orthogonal to D_{i+1} D_{i+1} is included $A d_i$ is included in D_{i+1} right hence r_{i+1} must be orthogonal to $A D_i$ what this makes finding a new search direction d_{i+1} from my new search direction d_i plus 1 from the residual at the $i+1$ step r_{i+1} very easy why because r_{i+1} by definition it is orthogonal to $A D_i$ right and hence a orthogonal to all the D_i right what does D_i , D_i consist of D_i spanned by these vectors right d_0 d_1 d_{i-1} .

So, r_{i+1} is orthogonal to $A D_i$; that means, it is A orthogonal to all these previous vectors d_{i-1} right. So, that is that is the key idea right it is because it is a orthogonal to all those d_{i-1} vectors by construction. I do not have to carry all those d_0 d_1 d_{i-1} vectors along right this is automatically my new residual is automatically orthogonal to all those previous vectors right.

So, all I need to do is to ensure orthogonality of r_{i+1} with my with D_i right with r_{i+1} with d_i right. So, so that is the Gram-Schmidt procedure need only ensure A orthogonality with d_i right r_{i+1} has to be orthogonal to all the previous search directions. It is orthogonal it is orthogonal to all the previous search directions d_0 d_1 through d_{i-1} . So, the only thing that that will make r_{i+1} D_{i+1} is to ensure that r_{i+1} is orthogonal to D_i right then I will get r_{i+1} it becomes d_{i+1} right. So, the Gram-Schmidt orthogonalization becomes a lot simpler.

(Refer Slide Time: 16:07)

Modified Gram-Schmidt coefficients

Only one of the Gram-Schmidt coefficients: $\beta_{ij} = -\frac{\mathbf{r}_i^T \mathbf{A} \mathbf{d}_j}{\mathbf{d}_j^T \mathbf{A} \mathbf{d}_j}$


$(j = 0, \dots, i-1)$ need to be evaluated. This can be further seen by taking the dot product of $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A} \mathbf{d}_j$ with \mathbf{r}_i : $\mathbf{r}_i^T \mathbf{r}_{j+1} = \mathbf{r}_i^T \mathbf{r}_j - \alpha_j \mathbf{r}_i^T \mathbf{A} \mathbf{d}_j$

which leads to: $\mathbf{r}_i^T \mathbf{A} \mathbf{d}_j = \frac{1}{\alpha_i} \mathbf{r}_i^T \mathbf{r}_i$ when $i = j$

$$= -\frac{1}{\alpha_{i-1}} \mathbf{r}_i^T \mathbf{r}_i \quad \text{when } i = j + 1$$

$$= 0 \quad \text{otherwise}$$

$\beta_{ij} = \frac{1}{\alpha_{i-1}} \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{d}_{i-1}^T \mathbf{A} \mathbf{d}_{i-1}}$ when $j = i - 1$ and zero for all other j 's



So, only one of the Gram-Schmidt coefficients, which normally are given by this right I can be anything j varies from 0 to i minus 1 by definition need to be evaluated. we can see we can get further verification; we can verify this further by taking the dot product of this expression right, \mathbf{r}_{j+1} equal to \mathbf{r}_j minus $\alpha_j \mathbf{A} \mathbf{d}_j$ which we just obtained last here right did we just obtain here.

So, we take this expression right and take the dot product of this expression with \mathbf{r}_i right. So, what do I get I get $\mathbf{r}_i^T \mathbf{r}_{j+1}$ is equal to $\mathbf{r}_i^T \mathbf{r}_j$ minus $\alpha_j \mathbf{r}_i^T \mathbf{A} \mathbf{d}_j$. So, this gives me $\mathbf{r}_i^T \mathbf{A} \mathbf{d}_j$ equal to this minus this divided by α_j right. Now, we know that this part when i equal to j right this term is going to survive this term is going to survive this term is going to be 0 because i is this is $j + 1$ right. So, this term is going to go to 0 in that case $\mathbf{r}_i^T \mathbf{A} \mathbf{d}_j$ will be given by one by $\alpha_i \mathbf{r}_i^T \mathbf{r}_i$. On the other hand, when i equal to $j + 1$ then this term is going to go to 0 right and this term is going to survive and this term is going to be equal to minus 1 by $\alpha_{i-1} \mathbf{r}_i^T \mathbf{r}_i$ right.

In all other cases, this term is going to be 0 right because if \mathbf{r}_{j+1} is not equal to \mathbf{r}_i or j is not equal to i then neither of these terms are going to survive because of the orthogonality of my residuals right. So, in the other cases this term is going to be 0 right So, what do I get I get that β_{ij} is equal to and we are not interested in β_{ii} right we are only interested in β_{ij} , where j is less than i right where j is goes from 0 to i minus

1 right. So, we are not interested in this term right we are not interested in the first expression, we are only interested in the second expression right and because when we substitute that second expression out here right. For r_i transpose $A d_j$ then i get $\beta_{i,j}$ is equal to one by $\alpha_{i-1} r_i$ transpose r_i divided by d_i transpose $A d_i$ minus 1 and this is this is going to be true, when j is equal to $i-1$ and is going to be 0 for all other j 's right.

(Refer Slide Time: 19:11)


Increased efficiency

Recalling that $\alpha_{i-1} = -\frac{\mathbf{d}_{i-1}^T \mathbf{r}_{i-1}}{\mathbf{d}_{i-1}^T \mathbf{A} \mathbf{d}_{i-1}}$ and substituting in (*), we get :

$$\beta_{i(i-1)} = \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{d}_{i-1}^T \mathbf{r}_{i-1}}. \text{ But we obtained earlier } \mathbf{d}_{i-1}^T \mathbf{r}_{i-1} = \mathbf{r}_{i-1}^T \mathbf{r}_{i-1}. \text{ Hence,}$$

$$\beta_{i(i-1)} = \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{r}_{i-1}^T \mathbf{r}_{i-1}}.$$

Since only $\beta_{i(i-1)}$ is required, it is no longer necessary to store the old search vectors to ensure conjugacy of search directions

 This not only reduces storage but also drastically curtails the number of computations necessary to calculate the new search direction


So, like we will like to simplify a little bit further because by recalling that α_{i-1} is equal to this, which we obtained earlier right substituting this expression for α_{i-1} out here right out here. we get $\beta_{i,i-1}$ is given by this right but we also obtained earlier d_{i-1} transpose r_{i-1} is equal to r_{i-1} transpose r_{i-1} and hence we can get $\beta_{i,i-1}$ in this simple form right.

So, that is the only Gram-Schmidt coefficient which is going to be non zero and it is given by the by the residual and the previous residual right. Since, only $\beta_{i,i-1}$ is required it is no longer necessary to store the old search vectors in order to ensure conjugacy of the search directions. These not only reduces storage, but also drastically curtails the number of computations necessary calculate the new search direction right.

(Refer Slide Time: 20:21)

Algorithm

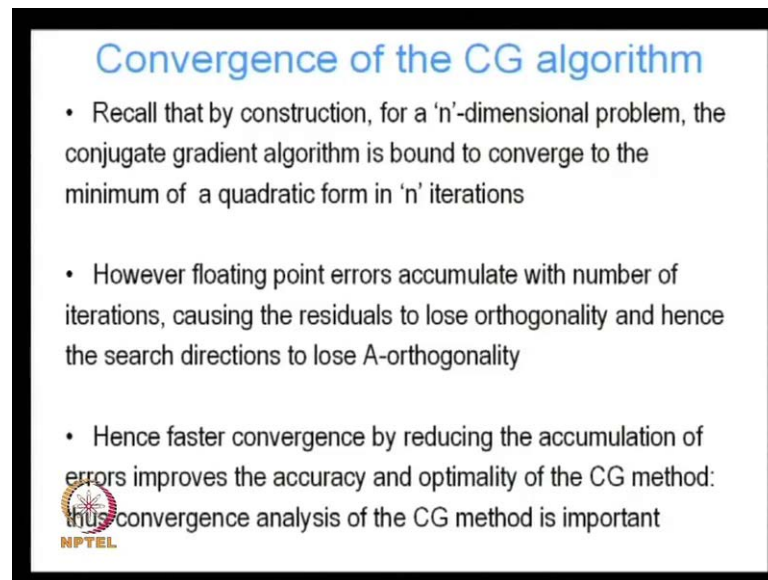
The final form of the conjugate gradient algorithm is as follows :

$$\mathbf{d}_0 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$$
$$\alpha_i = -\frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{d}_i^T \mathbf{A} \mathbf{d}_i}$$
$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i, \quad \mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i \mathbf{A} \mathbf{d}_i$$
$$\beta_{(i+1)(i)} = \frac{\mathbf{r}_{i+1}^T \mathbf{r}_{i+1}}{\mathbf{r}_i^T \mathbf{r}_i}$$
$$\mathbf{d}_{i+1} = \mathbf{r}_{i+1} + \beta_{(i+1)(i)} \mathbf{d}_i$$


So, the final form of the conjugate gradient algorithm is as follows so we have summarised it here. So, we start with a certain initial search direction and what is my initial search direction it is my initial residual and what is my initial residual that is \mathbf{b} minus $\mathbf{A} \mathbf{x}_0$ right then I compute my step size for instance for 0 I compute α_0 equal to minus $\mathbf{r}_0^T \mathbf{r}_0$ divided by $\mathbf{d}_0^T \mathbf{A} \mathbf{d}_0$ all quantities I know right. On the left hand side all the quantities I on the right hand side I know all the quantities right.


So, I know my α_0 then I compute my \mathbf{x}_1 is equal to \mathbf{x}_0 plus $\alpha_0 \mathbf{d}_0$ right and once as soon as I compute my \mathbf{x}_1 I can compute my new residual right because my new residual is nothing but $\mathbf{A} \mathbf{x}_1$ plus \mathbf{b} which I can simplify and write it like that \mathbf{r}_1 I can write as \mathbf{r}_0 minus $\alpha_0 \mathbf{A} \mathbf{d}_0$ right. So, I compute my new residual I find out my new Gram-Schmidt coefficient using this expression $\mathbf{r}_1^T \mathbf{r}_1$ divided by $\mathbf{r}_0^T \mathbf{r}_0$ right that gives me my new Gram-Schmidt coefficient β_{10} right β_{10} then I will compute \mathbf{d}_1 how will I compute \mathbf{d}_1 ? Well I compute it from \mathbf{r}_1 plus $\beta_{10} \mathbf{d}_0$ right and continue like this.

(Refer Slide Time: 22:06)



Convergence of the CG algorithm

- Recall that by construction, for a 'n'-dimensional problem, the conjugate gradient algorithm is bound to converge to the minimum of a quadratic form in 'n' iterations
- However floating point errors accumulate with number of iterations, causing the residuals to lose orthogonality and hence the search directions to lose A-orthogonality
- Hence faster convergence by reducing the accumulation of errors improves the accuracy and optimality of the CG method: thus convergence analysis of the CG method is important

 NPTEL

Now, let us look at convergence of the conjugate gradient algorithm we know that by construction for the n dimensional problem the conjugate gradient algorithm is bound to converge in n iterations right because that is an n dimensional space I am traversing in each time in an independent direction and by the time I end up I have spanned the entire space. So, I must and every time I go along a direction I make sure that the error in that direction is goes to 0 right. So, I systematically chop of my errors right.

So, the end of n iterations I am bound to get 0 errors normally; however, floating point errors accumulate with number of iterations causing the residuals to lose orthogonality and hence the search directions to lose A orthogonality right residuals lose orthogonality search directions. Since, search directions are search directions are obtained from the residuals the residuals also the search directions also lose a orthogonality.

So, hence, so that is why it is important to improve convergence. So, one might say why you need to study convergence of this CG algorithm we know that it is sure to converge, but if you converge in fewer number of iterations right instead of taking the full n which is of course, I will talk about that. So, by reducing the number of iterations you can reduce the accumulation of round of error and the less round of error the better is the performance of the algorithm because we are assured of orthogonality of the residuals; residuals will be more orthogonal less round of more orthogonal the search directions will be more a conjugate right.

So, the performance is going to be better right. So, that is why it is important to study the convergence of the conjugate gradient algorithm. Another point is that the conjugate gradient method is typically used for very large problems with very large ends right, where my where my direct solution technique is gauss elimination and variance of that are going to not going to give are going to be extremely expensive right.

So, if I have a 100000 by 100000 dimension matrix that I want to solve then; that means, that I know that it is going to converge in 100000 iterations and that also if there are no round of errors but why go for hundred that also is very expensive right. So, wish we wish to cut down that expense also you do not want to take 100000 iterations we want to converge in a fraction of those iterations right and let us see how we can do that if it is at all possible right.

(Refer Slide Time: 24:59)

Convergence of the CG algorithm


- We saw earlier that at each step of the algorithm, the error \mathbf{e}_i is a linear combination of the vectors \mathbf{e}_0 and $\{\mathbf{d}_0, \dots, \mathbf{d}_{i-1}\}$ i.e. of \mathbf{e}_0 and the subspace D_i

But $D_i = \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \dots, \mathbf{A}^{i-1}\mathbf{r}_0\}$
 $= \text{span}\{\mathbf{A}\mathbf{e}_0, \mathbf{A}^2\mathbf{e}_0, \mathbf{A}^3\mathbf{e}_0, \dots, \mathbf{A}^i\mathbf{e}_0\}$

This allows \mathbf{e}_i to be written in terms of a polynomial in the matrix \mathbf{A} , $P_i(\mathbf{A})$, and the initial error \mathbf{e}_0 :

$\mathbf{e}_i = P_i(\mathbf{A})\mathbf{e}_0$ with $P_0(\mathbf{A}) = 1$

The coefficients of $P_i(\mathbf{A})$ depend on the values of α_i and β_j



We saw earlier that at each step of the algorithm the error \mathbf{e}_i is a linear combination of my original error \mathbf{e}_0 and \mathbf{d}_0 through \mathbf{d}_{i-1} . We saw that last class right we may spend some time in obtaining that expression that what at any iteration \mathbf{e}_i at any iteration I can write the error \mathbf{e}_i as the error initial error plus a linear combination of my search directions up to that point right.

So, \mathbf{e}_i and the subspace D_i why because subspace D_i spans all those directions \mathbf{d}_0 \mathbf{d}_{i-1} but D_i is equal to is this is this space spanned by the vectors \mathbf{r}_0 $\mathbf{A}\mathbf{r}_0$ through $\mathbf{A}^{i-1}\mathbf{r}_0$ \mathbf{e}_0 just showed that little time little while ago right. And it is also this space

spanned by this why because r_0 is equal to a time e_0 right residual is equal to a times the error we know that already right. So, this is the also d_i is the space spanned by a_i is 0, a square e_0 a cube e_0 through $A_i A_0$ right you can see there is an additional a here right

So, it is the space spanned by these vectors because of this I can write e_i as a combination as a combination of these vectors $A_0 A$ square e_0 and. So, on and A_i by $A_i e_0$ right. So, it is a polynomial in the matrix a right. So, e_i is equal to $P_i A$ operating on a_0 . So, where $P_i A$ is the polynomial in a because you see all these A powers of A appearing here right.

So, it is polynomial in a operating on A_0 which satisfies the condition $p_0 A$ equal to 1_i is the iteration number here right I is the iteration number in the initial I mean at the zero th iteration $p_0 A$ has got to be equal to 1 otherwise at this identity is not going to be satisfied right and the coefficients of $P_i A$ depend on the values of my conjugate gradient coefficients basically my step size and my Gramm-Schmidt coefficients right.

(Refer Slide Time: 27:19)

Convergence of the CG algorithm


Expressing e_0 as a linear combination of the n orthonormal eigen vectors v_j of A : $e_0 = \sum_{j=1}^n \xi_j v_j$, we can write e_i as:

$$e_i = P_i(A)e_0 = \sum_{j=1}^n \xi_j P_i(A)v_j = \sum_{j=1}^n \xi_j P_i(\lambda_j)v_j$$

$$Ae_i = \sum_{j=1}^n \xi_j P_i(\lambda_j)\lambda_j v_j$$

Hence, $\|e_i\|_A^2 = \sum_{j=1}^n (\xi_j)^2 (P_i(\lambda_j))^2 \lambda_j$

The Conjugate Gradient method finds the polynomial P_i that minimizes the above expression.



So, expressing e_0 as a linear combination we can also write e_0 as a linear combination of the n orthonormal Eigen vectors of a right we did that earlier right A is a symmetric matrix its Eigen vectors form an orthonormal basis. So, I can always write the error e_0 as a linear combination of my Eigen vectors of a right. So, $A e_0$ I can write as $\sum_{j=1}^n \xi_j v_j$ right, where v_j are the orthonormal Eigen vectors of a .

Therefore we can write e_i as e_i is equal to $P_i A e_0$, which we just saw right which is equal to replacing e_0 by $\sum x_j v_j$. I can write it like that right and then I have this polynomial in A operating on v_j right, remember what is $P_i A$ it is a polynomial in A . So, it has got multiples of A a square a cube $A^2 A^3 \dots A^n$. So, on A^i right up to A^i right.

So, each of those A^i s operating on v_j is going to give me $\lambda_j^i v_j$ the corresponding Eigen value because v_j is an Eigen vector and if I have a cube operating on v_j what am I going to get? I am going to get $\lambda_j^3 v_j$ right a cube operating on v_j is equal to a square operating on $\lambda_j v_j$ is equal to a square operating on $\lambda_j^2 v_j$ is equal to $\lambda_j^3 v_j$ right.

So, this polynomial in A becomes quickly a polynomial with same polynomial in my Eigen value is λ_j right. So, from A matrix a polynomial in the matrix say this becomes A polynomial in my Eigen values right. So, this is true right. So, let us see what $A e_i$ becomes $A e_i$ becomes $\sum x_j P_i \lambda_j^i v_j$. So, I get $\lambda_j^i v_j$ right, $A e_i$ is equal to this. So, because just A^j operating on v_j that gives me another $\lambda_j^j v_j$ right $\lambda_j^j v_j$ and therefore the norm of e_i by norm I mean $e_i^T A^i e_i$ right because whenever I compute norm I do it with respect to that metric right that metric is given by A right.

So, norm of e_i square is given by $\sum_{i=0}^n x_j^2 P_i \lambda_j^{2i}$ right. So, basically just take the dot product of this with this right of this vector with this vector and we are going to get this vector sorry this expression by this scalar right dot product vector has to be a scalar. So, so this is this is my error expression for the conjugate gradient method.

It says that at any step at any step in my iteration the error the norm of the error right squared is given by this expression right some coefficients right some coefficients of my initial error basically if I take my initial error project it along my Eigen vectors of A right I will get my x_j s right I get my x_j s. So, that is $\sum x_j^2 P_i \lambda_j^{2i}$ right. So, the purpose of the conjugate gradient method is to find that polynomial find that polynomial, which minimises this error right because this is my error. So, when we when we when we saw if the when we are finding the polynomial, which minimises that error right.

(Refer Slide Time: 31:16)

Convergence of the CG algorithm


If we evaluate P_i using the eigenvalue λ of \mathbf{A} that maximizes the magnitude of $(P_i(\lambda))^2$ we can get a bound on the error as follows :

$$\|e_i\|_{\mathbf{A}}^2 \leq \max_{\lambda \in \lambda_j} (P_i(\lambda))^2 \sum_{j=1}^n (\xi_j)^2 \lambda_j$$

The Conjugate Gradient method finds the polynomial P_i that minimizes the above expression. So the polynomial P_i , if found using conjugate gradient, satisfies the condition :

$$\|e_i\|_{\mathbf{A}}^2 \leq \min_{P_i \in \mathcal{P}} [\max_{\lambda \in \lambda_j} (P_i(\lambda))^2] \sum_{j=1}^n (\xi_j)^2 \lambda_j = \min_{P_i \in \mathcal{P}} [\max_{\lambda \in \lambda_j} (P_i(\lambda))^2] \|e_0\|_{\mathbf{A}}^2$$

where \mathcal{P} is the set of all polynomials of order i



So, so the way to do that we calculate let us see so we have this expression $\|e_i\|_{\mathbf{A}}^2 \leq \max_{\lambda \in \lambda_j} (P_i(\lambda))^2 \sum_{j=1}^n (\xi_j)^2 \lambda_j$ square j is equal to one to n , where n are the Eigen values right. So, this is the sum over all the Eigen values of a right. So, now I say that if this is equal to this if I pull this thing this polynomial square out of this summation but make sure that I evaluate it at the Eigen value, which gives the largest value of the polynomial. So, I am I evaluate this polynomial at all the Eigen values right and then I found out the Eigen value for which this polynomial is largest right. So, if I pull this expression, this expression out of the summation then I am guaranteed that this has got to be less than that right.

So, this is what I am doing here, I am pulling out that $\max_{\lambda \in \lambda_j} (P_i(\lambda))^2$ square out of the summation I and I am evaluating it at the value at the Eigen value which maximizes the value of the polynomial right. And if I do that this equality sign changes to the to a lesser than or equal to sign right is that clear because this is a bound right I am taking the largest possible value of the polynomial and I am choosing the Eigen value, which maximizes the value of the polynomial right and then this becomes like this I get a bound on the error right that is what we are interested in getting right we are interested in getting bounds right.

So, the conjugate gradient method finds the polynomial P_i that minimises this above expression right. So, the polynomial P_i if found using conjugate gradient satisfies the following condition what is that condition? norm of \mathbf{A} i square with respect to a is lesser

than or equal to $\min P_i$ belonging to p . So, the conjugate gradient method finds that polynomial P_i belonging to the set of all polynomials of order i if I am considering the iteration at step two at iteration two. Basically, all the all possible all possible quadratics right quadratics all possible quadratics and finds the quadratic finds the quadratic, which minimises this expression right; it looks at all possible quadratics right at a iteration two at iteration two; it looks at all the possible quadratics right quadratics in this there is a P_i right P_i λ right. This and then it finds the quadratic which minimises this expression right.

So, that it finds P_i belonging to the space to the set of all polynomials of order i , which minimises this expression. Basically, which means minimises P_i belonging to p this term remains the same and this term $\sum_i x_i^2$ is nothing but norm of e_0 square right you can show that e_0 is equal to $\sum_j x_j v_j$. So, norm of e_0 square is $\sum_j x_j^2 v_j^T v_j$ operating on x_j . So, this is orthonormality we use orthonormality of the Eigen vectors we get x_j where p . So, so we get that right.

(Refer Slide Time: 35:04)

Convergence of the CG algorithm

- In words: the polynomial P_i satisfies the above condition when $|P_i|^2$ has the smallest possible value for $\lambda = \lambda_{\max}$ where λ_{\max} is the eigen value of \mathbf{A} that maximizes $|P_i|^2$
- Thus for values of $\lambda < \lambda_{\max}$, $|P_i|^2$ must be even smaller. Obviously the best possible polynomial, i.e. the one that would result in the error norm approaching zero, occurs when $|P_i|^2$ is zero at $\lambda = \lambda_{\max}$ and hence zero at all the 'n' eigen values of \mathbf{A}
- This is possible for a polynomial with at least 'n' roots. Since the order of the polynomial depends on the iteration number 'i', it is further confirmation of convergence when $i=n$

So, in words the polynomial P_i satisfies the above condition when the mod of P_i square has the smallest possible value for λ is equal to λ_{\max} right. So, we find the Eigen value So, I look at each polynomial in that space. Suppose, I am at iteration two at iteration two i look at all the quadratics all possible quadratics I evaluate each possible quadratic for each Eigen value for all my Eigen values then for each quadratic I choose

the Eigen value which maximises that quadratic I do that for each of those quadratics right and then among those quadratics, I choose the one which gives the smallest value for its maximum Eigen value right is that is that clear.

So, I am choosing this quadratics set of all quadratics I evaluate each quadratic for each of the Eigen values right I look at I find out each Eigen value maximizes that quadratic put that quadratic aside look at the next quadratic do the same thing right and then finally, find out which of those quadratics gives the smallest value for the largest Eigen value corresponding to it right I hope that is clear.

So, the conjugate gradient method does all that right. So, basically it finds the quadratic, which satisfies this condition right not only the quadratic depending on the iteration the cubic the quartic, the quintic and So on and So, forth right thus the values of λ , λ less than λ_{\max} mod of P_i square must be even smaller. we know because P_i is minimised for the largest Eigen value right. So, for smaller Eigen values P_i square must be even smaller; obviously, the best possible polynomial that is the one that would result in the error norm of reaching approaching is 0 will happen when the polynomial becomes 0 at all the n Eigen values of a right because then p when P_i square is 0 at λ equal to λ_{\max} right, if it is 0 at the largest Eigen value then it has to be 0 at all the other Eigen values right.


So, that is ideal situation. So, this is only possible for a polynomial with atleast n roots why because I have n Eigen values right when my matrix A is of size n . So, it must have n Eigen values and if my polynomial is has to be 0 at all those Eigen values; that means, it must have n roots right it must have n roots right a quadratic has two roots. So, for n roots the polynomial must be of order n right.

So, since the polynomial depends on the iteration number i . So, each at each iteration I have polynomial of order i ; that means, when will the polynomial have n roots when I reach n iterations only then it is possible for it to have n roots right. At previous iterations, it cannot have n roots right it cannot have it cannot make all the Eigen values it cannot be 0 at all the Eigen values right. So, it is possible, it is possible only when it has got n roots it is further confirmation of convergence when i is equal to n right.

(Refer Slide Time: 38:38)

Convergence of the CG algorithm

- It has been seen that the convergence of CG is faster when the eigenvalues of \mathbf{A} are clustered together.
- This is likely to happen when the condition number of \mathbf{A} is small
- For example, it has been shown that for a certain class of polynomials known as Chebyshev polynomials the convergence is strongly dependent on the condition number κ as follows:


$$\|\mathbf{e}_i\|_{\mathbf{A}} \leq 2 \left[\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right]^i \|\mathbf{e}_0\|_{\mathbf{A}}$$

So, it has people have done lot of studies on this and people are still working on the if it is after all a relatively young technique may be not more than 40 or 50 years old right, compared to some of the other techniques, which has been around for several 100 years. So, people are still working on that and they found that the convergence of the conjugate gradient method is faster, when the Eigen values are of \mathbf{A} are clustered together which is sort of intuitive right because I need to I my convergence will be better when the polynomial is small at all the Eigen values as small as possible.

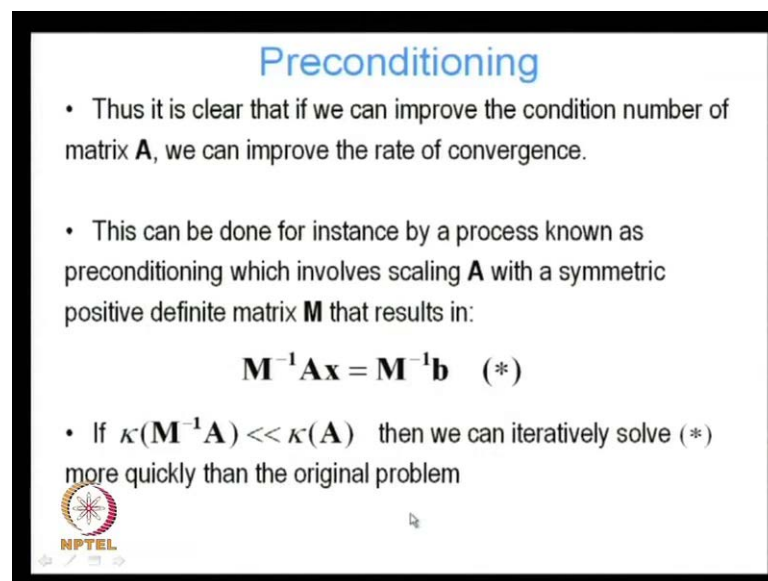
Now, if all the Eigen values are together it becomes easier for me to make the polynomials smaller at all the Eigen values right. Because, if it is small at one Eigen value then all the Eigen values are together since the function is continuous polynomial is continuous it is not going to be to big at a in a neighbourhood of that Eigen value right. So, if all my Eigen values are together it becomes easier to minimise the function right to minimise the function the error function right.

So, when what is going to happen when all the Eigen values are clustered together. Well, my condition number is going to be small, what is my condition number it is my largest Eigen value divided by my smallest Eigen value. So, when the Eigen values are closer together the condition number is going to be small right. So, again we go back to that that old criteria, which we have encountered, time and again the convergence depends on the condition number. For example, it has been shown for a certain class of polynomials

known as Chebyshev polynomials that convergence of the conjugate gradient method is strongly dependent on the condition number.

So, we know that it is going to depend on the condition number what is the exact dependence will depend on the type of polynomials we are choosing. And for a particular type of polynomials I have shown that expression right the dependence on the condition number is like this the it depends on the condition number like this right for a different class of polynomials it will depend on the condition number but the dependence will be of a different form right. For the chebyshev polynomials it is going to depend like this and we can see from this expression that for smaller condition numbers for smaller condition numbers, this term is going to become smaller this term is going to become smaller right.

(Refer Slide Time: 41:13)




Preconditioning

- Thus it is clear that if we can improve the condition number of matrix **A**, we can improve the rate of convergence.
- This can be done for instance by a process known as preconditioning which involves scaling **A** with a symmetric positive definite matrix **M** that results in:

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b} \quad (*)$$

- If $\kappa(\mathbf{M}^{-1}\mathbf{A}) \ll \kappa(\mathbf{A})$ then we can iteratively solve (*) more quickly than the original problem

 NPTEL

And because of that the condition number improve the condition number of the matrix A that is we make it we make the condition number as close to one as possible we can improve the rate of convergence. Well, how can we improve the condition number I have my matrix a which is already given it is a and condition number of a I cannot change my matrix a is defined by the physical problem right I have a physical problem and the values in that physical problem is going to it is going to determine my matrix A. So, the condition number of a is given how can I change the condition number of A. Well, I can change the condition number of a by a process which is known as preconditioning right

preconditioning; which involves scaling A with a symmetric positive definite matrix M that results in this.

So, I had my original equation $Ax = b$, I multiply both sides with M^{-1} right M^{-1} , where M is a symmetric positive definite matrix. Now, if it so turns out that the condition number of my new coefficient matrix $M^{-1}A$ is much smaller than the condition number of A then we can iteratively solve this problem much faster than my original problem right.

Because, I know that the rate of convergence depends on the condition number right so if somehow if I scale it with scale that with a matrix right. And the resultant coefficient matrix the new coefficient matrix is $M^{-1}A$ right and if the if the condition number of the new coefficient matrix is much smaller than the condition number of my original matrix then automatically it is going to converge much better right.

So, that is known as preconditioning. So, almost all implementations of the conjugate gradient method are have a preconditioned right to improve the convergence right. We can we cannot be satisfied for large problems taking n iterations right I have a 100000 by 100000 problem I do not want to take 100000 steps right. So, I want to reduce the number of steps and what is the key to reducing the number of steps reducing the condition number of my coefficient matrix, how am I going to do that? by scaling it right by scaling it.

(Refer Slide Time: 43:44)


Preconditioning

- However $M^{-1}A$ may not be symmetric and positive definite. Then it would not work.
- However if M is symmetric and positive definite, we have seen earlier that it is always possible to use Cholesky decomposition to obtain a lower triangular matrix L such that $LL^T = M$

Then we solve the problem :

$$L^{-1}AL^{-T}\hat{x} = L^{-1}b, \quad \hat{x} = L^T x \text{ instead of } Ax = b$$

The matrix $L^{-1}AL^{-T}$ is symmetric and positive definite by construction.



But, there is a problem well what is a problem? Well if M , $M^{-1}A$ has to be the coefficient matrix has to be symmetric and positive definite for the conjugate gradient method to work if $M^{-1}A$ is not symmetric or positive definite than that is the end of the story. My whole thing breaks down right; just because M is symmetric and positive definite it does not mean that $M^{-1}A$ is going to be symmetric or positive definite right. So, if; however, if M is symmetric and positive definite we have seen earlier several classes earlier that it is always possible to decompose a symmetric positive definite matrix into a product of a lower triangular matrix and its transpose right $L L^T$ transpose is equal to M right.

So, instead of solving that previous problem, in case $M^{-1}A$ is not symmetric or is not symmetric and positive definite; we solve an alternative problem what is that alternative problem that alternative problem is $L^{-1}AL^T x = b$ where we have done it we have done a transformation of variables you have created new variable $x = L^T x$. So, we can see this is identical to this right. Because, $L^{-1}AL^T x = b$ is then equal to $L^{-1}AL^T L^T x = b$ which is $x = b$. So, I have $A x$ and then I have multiplied both sides by L^{-1} right.

So, this problem is identical to this problem right, this problem is identical to this problem right. And now this matrix $L^{-1}AL^T$ is bound to be symmetric and positive definite, why? Evaluate you can see that it is symmetric by construction right $L^{-1}AL^T$ if I take the transpose of that I again get $L^{-1}AL^T$. So, this is symmetric by construction right this is symmetric by construction matrix A of course, is symmetric.

So, this is symmetric by construction and it is also going to be positive definite why because I know the A is positive definite and L is always going to be positive definite right. We have seen that L is going in our if we go back to your notes you will see that L is always positive definite. So, in that case this matrix is always going to be positive definite. So, I can always use this as a preconditioner and why can I use it as a preconditioner well.

(Refer Slide Time: 46:26)

Preconditioning

Also, the spectrum of $L^{-1}AL^{-T}$ is the same as that of $M^{-1}A$.

We can see this as follows: suppose λ is an eigen value of $M^{-1}A$ and v an eigenvector; then λ is also an eigen value of $L^{-1}AL^{-T}$: $(L^{-1}AL^{-T})(L^T v) = (L^T L^{-T})L^{-1}Av = L^T M^{-1}Av = \lambda L^T v$

Thus the condition number of $L^{-1}AL^{-T}$ is the same as $M^{-1}A$: hence $L^{-1}AL^{-T}$ is as effective a preconditioner as $M^{-1}A$

Preconditioning may thus involve a certain extra computational expense. The task thus is to find a preconditioner that improves convergence well enough to compensate for this additional expense

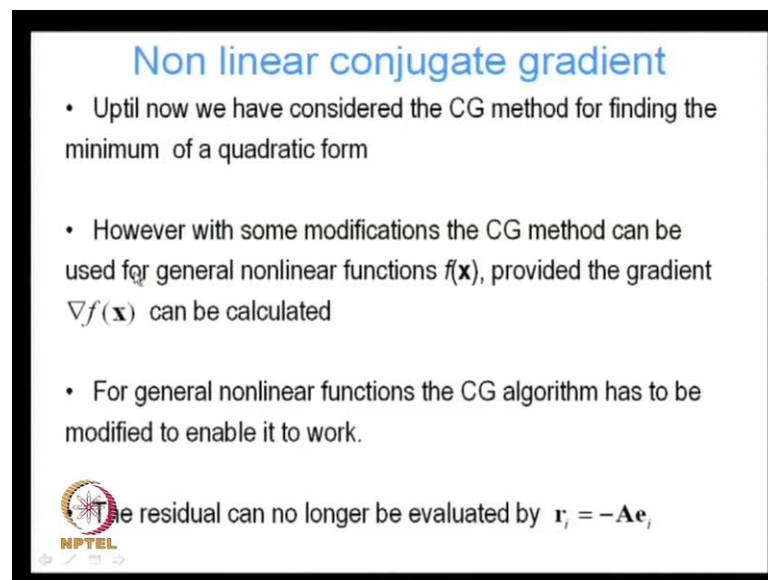
The reason I can use it as a preconditioned is because the spectrum of $L^{-1}AL^{-T}$ is the same as $M^{-1}A$, what good to I mean, well if I have $L^{-1}AL^{-T}$ is $L^{-1}AL^{-T}$ equal to $M^{-1}A$ and I construct a matrix $L^{-1}AL^{-T}$ then the Eigen values of $L^{-1}AL^{-T}$ are going to be identical to the Eigen values of $M^{-1}A$ and if the condition number of $M^{-1}A$ is good; that means, the condition number of $L^{-1}AL^{-T}$ is also will be good right because they are the same Eigen values right.

So, we can see this is follows this is little proof which shows that spectrum of $L^{-1}AL^{-T}$ is the same as the spectrum of $M^{-1}A$. We can show that suppose λ is an Eigen value of $M^{-1}A$ we can show this by little manipulation here right that λ is also an Eigen value of $L^{-1}AL^{-T}$ I do not want to go into this right. Now, because I want to finish this let us continue. So, the condition number of $L^{-1}AL^{-T}$ is the same as $M^{-1}A$ hence $L^{-1}AL^{-T}$ is as effective a preconditioner as $M^{-1}A$ right.

But, we see that preconditioning may involve a certain extra computational expense right because I have got to find that matrix M^{-1} I have got to find L^{-1} and L^{-T} and then I have to invert that and refine this I do this only once right for linear problem I do this only once right and once I do this I can use that that preconditioner every time right for all my iterations. and if I choose my if I

choose my preconditioner sufficiently well. So, that its condition number is much smaller than the condition number of my original matrix I will get much much faster convergence much much faster convergence and this is enough to compensate for the additional expense of calculating the preconditioning right.


(Refer Slide Time: 48:46)



Non linear conjugate gradient

- Uptil now we have considered the CG method for finding the minimum of a quadratic form
- However with some modifications the CG method can be used for general nonlinear functions $f(\mathbf{x})$, provided the gradient $\nabla f(\mathbf{x})$ can be calculated
- For general nonlinear functions the CG algorithm has to be modified to enable it to work.

The residual can no longer be evaluated by $\mathbf{r}_i = -\mathbf{Ae}_i$

 NPTEL

So, up till now we have talked about the linear conjugate gradient method why linear? Because we started with minimising a quadratic form right and if I am minimising a quadratic form we found that eventually we have to solve the equation $\mathbf{A} \mathbf{x} = \mathbf{b}$ where \mathbf{A} is the constant matrix right. So, it is a linear problem $\mathbf{A} \mathbf{x} = \mathbf{b}$ remains constant throughout solve it will converge in an iterations if you do good conditioning you can converge even faster right. So, that is true for quadratic forms but nothing I mean there are lots and lots of problems, where you will encounter non quadratic non-linear equations right.

So, in those equations can we use the conjugate gradient method well the answer is that you can use the conjugate gradient method, but lot of the beautiful things beautiful convergence properties things like that that we saw for linear conjugate gradient no longer holds true right; however, you can use the conjugate gradient method for some with some modifications for general nonlinear equations f of \mathbf{x} provided that gradient can be calculated what do I mean by can be calculated the gradient exists right at the domain of interest I can the gradient exists everywhere.

For general nonlinear functions the CG algorithm has to be modified in order to enable it to work. So, what are the modifications that are necessary well we can no longer calculate the residual by taking the product of the a matrix with my with my error right because there is no a matrix right no constant a matrix right. So, we can no longer do that.

(Refer Slide Time: 50:33)


Modifications to the algorithm

To find the minimum of a general nonlinear function $f(\mathbf{x})$ we need to find a root of the nonlinear function $\nabla f(\mathbf{x}) = 0$

If the minimum corresponds to \mathbf{x} i.e. \mathbf{x} is a root of $\nabla f(\mathbf{x}) = 0$, then at iterate \mathbf{x}_{i+1} , the residual \mathbf{r}_{i+1} can be written as: $\mathbf{r}_{i+1} = \nabla f(\mathbf{x}) - \nabla f(\mathbf{x}_{i+1}) = -\nabla f(\mathbf{x}_{i+1})$

In addition it is no longer possible to write a closed form expression for the residual as in $\alpha_i = \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{d}_i^T \mathbf{A} \mathbf{d}_i}$

Instead we need to compute the step size by solving a one-dimensional minimization problem



So, to find the minimum for general nonlinear function $f(\mathbf{x})$ we need to find a root of nonlinear function $\text{grad of } f(\mathbf{x}) = 0$ right I want to minimise $f(\mathbf{x})$ I want to find the value of f which minimises $f(\mathbf{x})$. So, I have to solve the equation $\text{grad of } f(\mathbf{x}) = 0$ right if the minimum corresponds to \mathbf{x} that is \mathbf{x} is a root of $\text{grad of } f(\mathbf{x}) = 0$ then add iterate $i + 1$ the residual \mathbf{r}_{i+1} can be written as $\mathbf{r}_{i+1} = \text{grad of } f(\mathbf{x}) - \text{grad of } f(\mathbf{x}_{i+1})$ because this is the true solution this is the true solution minus the gradient at $i + 1$. So, this minus this is got to be my residual, but at the true solution this term goes to zero. So, I have minus $\text{grad of } f(\mathbf{x}_{i+1})$ right.

So, that is that is that that has to be true; however, it is no longer possible to write also a closed form expression for the residual this should not be the residual this should be the step size right for the step size α as in this right. So, this I can no longer write because what is my \mathbf{A} ; my \mathbf{A} there is not constant \mathbf{A} . So, this expression is also not going to be true. So, what we need to do is again is at each step I have to find out α which minimises the residual in that direction and I have to do that iteratively right I have to do

a generalized line search to find to minimise the residual along a particular search direction.


(Refer Slide Time: 52:20)

Modifications to the algorithm

This requires choosing α_i in the direction \mathbf{d}_i . Since for a given \mathbf{d}_i , $f(\mathbf{x}_i + \alpha_i \mathbf{d}_i)$ is a general nonlinear function of α_i , this requires a general line search procedure

Also, recall that when the conjugate gradient method is applied to a quadratic form, the Gramm-Schmidt coefficient β_{ij} can be evaluated as $\beta_{ij} = -\frac{\mathbf{r}_i^T \mathbf{A} \mathbf{d}_j}{\mathbf{d}_i^T \mathbf{A} \mathbf{d}_j}$

Taking advantage of the Krylov structure of the sub-space formed by the search directions, this expression simplified to



$$\beta_{ij} = \begin{cases} -\frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{r}_{i-1}^T \mathbf{r}_{i-1}} & \text{if } j = i - 1 \\ 0 & \text{if } j < i - 1 \end{cases}$$

So, this requires choosing alpha i in the direction d i; since for a given d i f of x i plus alpha i d i is a general nonlinear function of alpha i this requires a general line search procedure why because for a given d i i my alpha is an unknown, but this is a general nonlinear functions. so it is a general nonlinear function of alpha it is a basically a polynomial in alpha i right. So, I have to find the value of alpha i, which minimises that polynomial right. So, also let us recall that when conjugate gradient method is applied to a quadratic form we can calculate the Gramm-Schmidt coefficients like this and we found that taking advantage of the Krylov structure the Gramm-Schmidt coefficients become this right.


(Refer Slide Time: 53:17)

Modifications to the algorithm

In the absence of a constant coefficient matrix \mathbf{A} for the general non linear equation, how to choose β_{ij} becomes an open question

Finding the optimum value of β_{ij} for the nonlinear conjugate gradient algorithm is a subject of current research

Two commonly used expressions involve the Fletcher-Reeves formula, which is identical to that used for the quadratic form, and the Polak Ribiere formula which gives $\beta_{ij} = \frac{\mathbf{r}_i^T \mathbf{r}_i - \mathbf{r}_i^T \mathbf{r}_{i-1}}{\mathbf{r}_i^T \mathbf{r}_i}$



So, in the absence of a constant coefficient matrix \mathbf{A} for the general nonlinear equation how to choose those Gram-Schmidt coefficients becomes an open question right becomes an open question. So, finding an optimal value for the Gram-Schmidt coefficients β_{ij} for the nonlinear conjugate gradient algorithm is also a subject of current research. I mean two commonly used expression involve the Fletcher-Reeves formula which is basically identical to that used by the we say that even for the nonlinearly conjugate gradient we are going to calculate β_{ij} using that expression right that is given by what is known as the Fletcher-Reeves formula and as an alternative there is something which is known as the Polak Ribiere formula, which is given by this which is a slight modification of the expression for the in the linear case right.

There is also something known as the Hessian formula which I have not mentioned, but these are all variants of this β_{ij} , which are used for nonlinear conjugate gradient. So, I want to wrap my discussion on nonlinear conjugate gradients in the early part of the next lecture and then we will move on to another second part of this course. Basically, because up till now we have focussed on solution techniques right I have the system of equations either linear or nonlinear how can I solve those equations. So, I have concerned myself with that for the first half of this course.

But the second half of this course I want to show you how I get those equations how I get that system of equations linear or nonlinear equations how do I model my physical

problem to get those equations I eventually I model my physical problem using partial differential equations right. So, from those partial differential equations how do I get this system of this matrix equation which I have to solve to find my solution. So, we will start talking about partial differential equations and numerical techniques for modelling partial differential equations from our next class after we wind up our discussion of the conjugate gradient method.

Thank you.