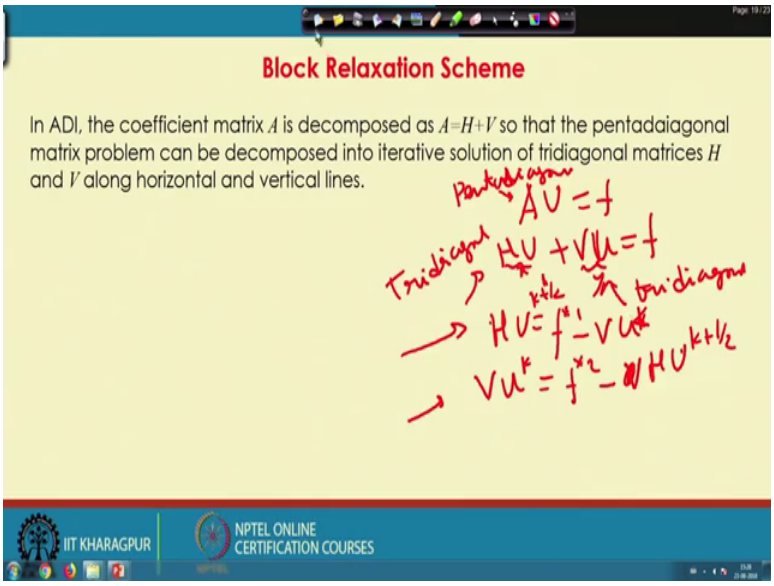
Matrix Solvers
Prof. Somnath Roy
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 54
Block Relaxation Method

(Refer Slide Time: 00:15)



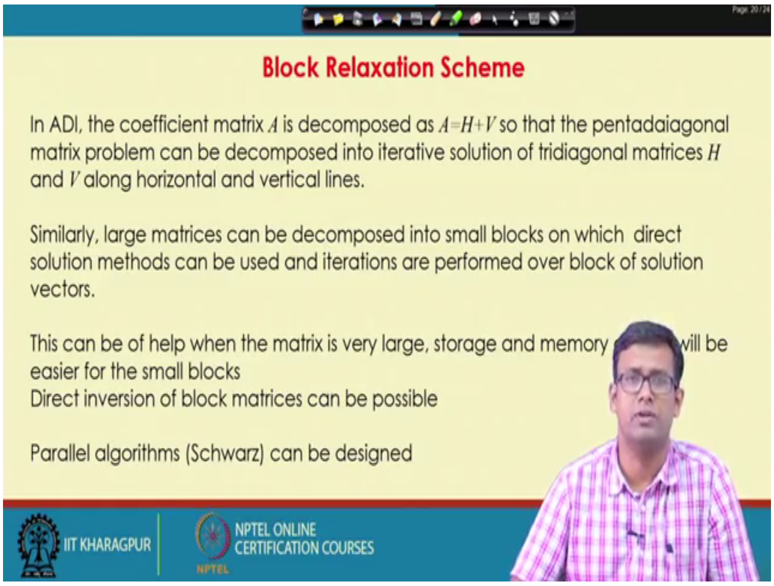
Welcome, we have been discussing about different relaxation schemes line relaxation and Block Relaxation schemes. In an with a focus to solve the matrix equation not in a point wise manner; that means, each element of the x factor will be updated through the iterative schemes, not going in that way rather trying to identify a part of the solution vector and try to update it at one go and then do it for different parts.

So, we have discussed about alternate direction implicit method which I said is a line relaxation scheme. In ADI the coefficient matrix A is decomposed into two parts H plus V; H comes from disiteration in x-direction y comes from this V comes from disiteration of in y-direction.

So, that the pin now the pentadiagonal matrix problem can be decomposed as two problems sort of this is ADI is typically for 2-dimensional Laplace equation problem. So, instead of AU is equal to f we can write HU plus Vu is equal to f and HU is has x-directional disiteration, Vu has y-directional disiteration. So, this is pentadiagonal A x is A is pentadiagonal, H is tridiagonal, V is also tridiagonal.

The idea is first solving HU is equal to f minus Vu star and then solving. So, this is HU k plus half is equal to Vu k rather last updated value and then solving Vu k is equal to f some f star f star 1 and f star 2 some value minus V minus HU k plus half and updating it. So, this is broken in a pentadiagonal matrix equation is broken into two tri diagonal matrix equations and we know the tridiagonal matrix equations are much faster to solve.

(Refer Slide Time: 02:49)



Similarly, large matrices can be decomposed into small blocks on which direct solution methods if not direct solution method some of the iterative solution methods which are easy to implement can be used and iterations are performed over blocks of the solution vectors. So, instead of solving the entire matrix equation we will break the matrix into small blocks and try to solve for each of the small blocks.

And while solving for one particular small block we will assume that the Gauss value is obtained for the vectors which are associated with the other small blocks. Only for the diagonal part we will use the we will solve it for off diagonals will assume the Gauss value for the blocks and then iterations are performed over blocks of solution vectors. This can be a fail when the matrix is very large say a million by million matrix which is very often we encounter million by million matrix.

It will be much difficult to store the matrix we need some storage algorithm to consider the sparsity of the matrix and store the right nonzero values. Because storing zero values will be will add a redundancy in the overall algorithm and also when you are using some storage algorithm that only nonzero values and their right pointers are stored you have to use a memory management methods also.

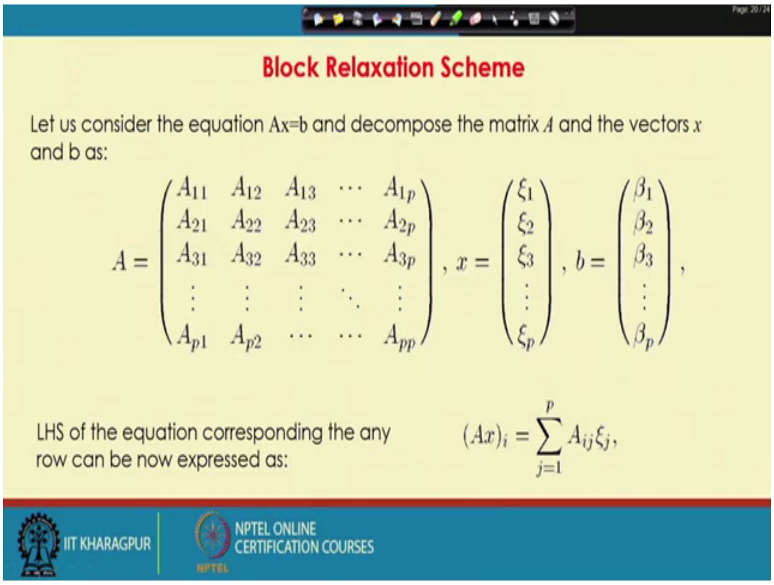
So, for a large matrix the storage and memory access will be much difficult much complex say and it can be easy if we break down into small blocks we will only store the small blocks and doing any memory management will be easy also in the small blocks. So, this is not only memory management and by the computer, but it is also important when we think of that thing that a part of memory management is actually being done by the processors and the computer architecture also.

So, it is accessing memory at the RAM, putting some of it in cache data from the cache, starting with a memory location and going along the pointer to the new memory the new element of that memory all these things this will be easy when you break it into small pieces of blocks. Direct inversion of block matrices can be possible, you have seen the direct inversion is usually a costly method except a TDMA we if we have to use something like ALU or some algorithm like Cholesky etcetera. These are costly methods and they take almost and if consider an n by n matrix they take almost n cube operations.

So, you for a million by million matrix it will be extremely large number of operations. So, you probably cannot do it in any practical purpose, but if we break down into small matrix matrices we will may get 100 by 100 matrices, where n cube operations is still possible. So, direct inversion can be is possible for block matrices. And parallel algorithms like Schwartz method can be designed idea parallel algorithm is that you divide the big matrix intuition equation into number of small matrix blocks and ask different processors in a computer architecture to solve different blocks in parallel.

Nowadays we get computers with dual core 16 core even 32 core processors 32 processors are presented in the computer. So, ask each processor to look into one block. So, that the small 32 small blocks can be done in parallel then you can save a lot of time. Instead of do going from one to million maybe each processor is going 1 to 1000 rows and considering 32 plus 32000 rows are done in the time in which you are supposed to 1000 rows. So, this is the parallel process we will discuss about parallelization in few of the later lectures which also can be done once we think of a block partitioning of the matrix or when you think of block relaxation schemes.

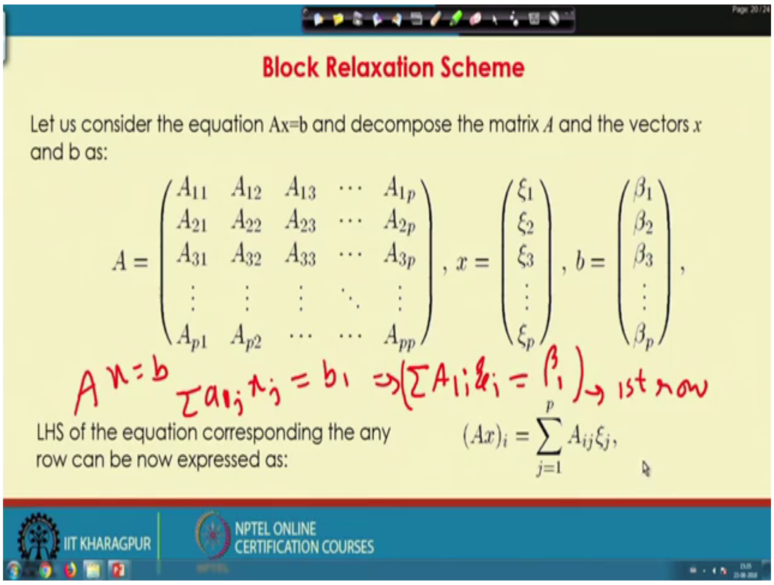
(Refer Slide Time: 06:59)



So, how will it be let us consider an equation Ax is equal to b the matrix A and the vector x and the solution vector and the and the RHS vector b such that a is divided into number of small matrices all A 11, A 12, A 13, A pp all these are matrices. A is decomposed into number of small matrices each is a block matrix and x is decomposed into p blocks xi 1, xi 2, xi 3, xi p excess may be excess ten thousand elements and each of them there are 10 blocks each has 1000 elements.

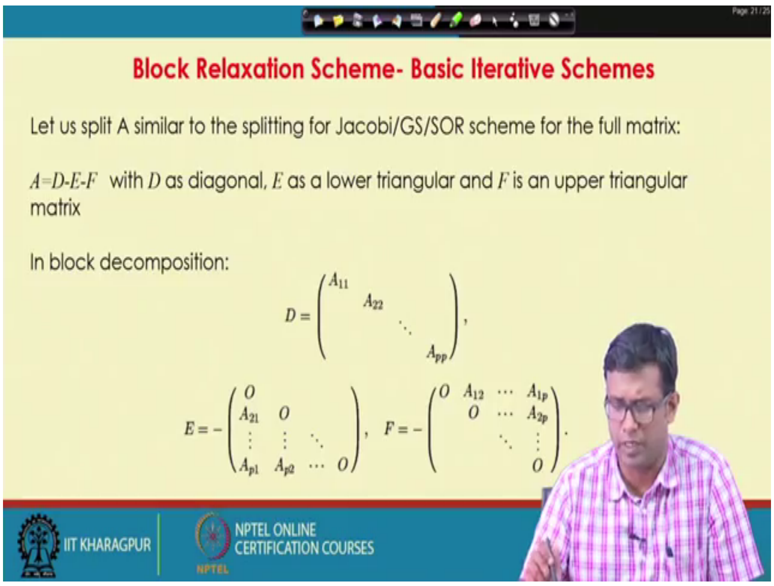
So, all these xi's will be added up to give us x and similarly b is also decomposed into some small blocks. LHS of the equation corresponding to any row now can be expressed as right hand side is summation of A and multiplication of A and x for one particular row will give Ax of that particular row. Ax is nothing, but A ii xi i so, what is the first row of this equation if we go back that.

(Refer Slide Time: 08:09)



We are solving Ax is equal to b the first row will be a ij x j is equal to b i. If we think of a block partitioning this will be this can be also expressed as summation of A 1 i xi i, so, this is first row a 1 j v j is equal to this summation of A 1 i xi i is equal to beta once the entire equations first row of that because this is also a matrix also vector equation beta 1 has number of elements, the first row of this particular element. So, we can say that the left hand side of this equation which is a 1 j x j or a ij for any a particular y can be expressed as A ij xi j into xi j.

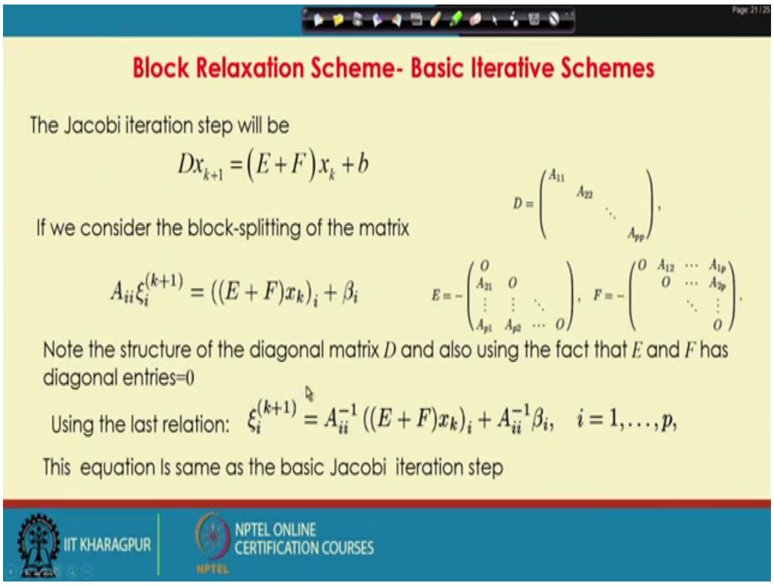
(Refer Slide Time: 09:29)



So, let us split a into A is similar to splitting of Jacobi GS or SOR scheme of the full matrix, that is A is equal to D minus E minus F; D is the diagonal, E is the lower triangular, F is upper triangular matrix. Based on the our scheme whether it is Jacobi or Gauss Seidel we will use will take you multiply the last Gauss value with E F or E E plus F we will say that, but we can revise the discussion on basic iterative methods. This is the standard splitting for Jacobi or Gauss Seidel.

In a block decomposition D will be the blocks containing the diagonal terms. Earlier in a general splitting of a for a basic iterative point relaxation scheme A only contain D only contain the diagonal elements. Here instead of diagonal elements they are containing the diagonal blocks E contain the lower triangular blocks, F contain the upper triangular blocks and E and F both has diagonal element 0. So, instead of splitting it by element we are splitting it by blocks of matrices.

(Refer Slide Time: 10:47)



So, the Jacobi iteration step is Dx k plus 1 E F plus k x k. If we consider the block splitting of the matrix this will be A 11 A 22 etcetera. So, this is one block of matrix is multiplied with the solution value that we have to find out and E plus F will be multiplied with x k which is the last available value. So, this is A ii xi i k plus 1 E plus F x k i beta i. Here we are writing x k not we are writing xi because x k contains the entire solution vector and entire solution vector has to be multiplied with E, E and F and each E and F has diagonal 0. So, that particular vector which is xi k is now being multiplied with E and F.

Note that the structure of the diagonal matrix D and the that has been utilized here that the diagonal matrix is A ii is the structure of diagonal matrix is the diagonal blocks here. This matrix is D is not actually a diagonal matrix rather a block diagonal matrix here and we also use the fact that E and F has diagonal entries 0, so, we can write this. Using the last relation the relation we have developed here we can write xi i k plus 1 which is the new updated block of solution vector. This is not a single solution vector rather this is the block of solution vector a number of x's; number of x's are involved in this vector is equal to A ii inverse E plus F x k plus A ii inverse beta ii is equal to 1 to p.

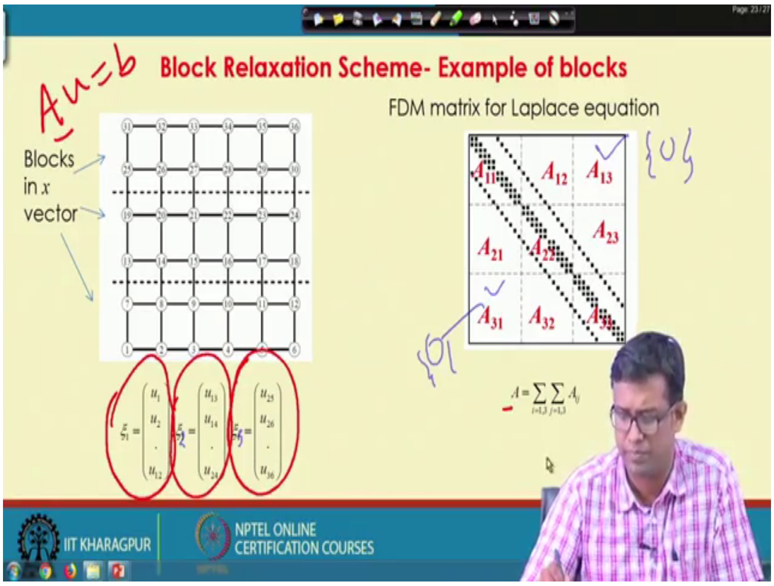
Now, we can see that this is not a direct solver points or relaxation scheme so, that this is D inverse. So, this is a single 1 by the diagonal value this is inverse of a matrix. However, as we think of dividing the matrix into large number of matrices a large number of matrix blocks this matrices are basically smaller matrices and we can easily find out their inverse, it will be not that complex. Complexity is usually in terms of what are the order of operations we need to do to find out in that inverse. Inverse is inverse finding is same like Gauss elimination scheme takes n cube operations.

As we have broken down the matrix into small blocks the number of rows in each block is very small. If you think of million by million matrix maybe you have 1000 rows in each block. So, this will be much less time consuming number of steps will be less to find out A ii inverse because this is a small matrix the number n is smaller here. And we have to do it for all the blocks starting from first block xi 1 xi 1, xi 2, xi p if you do it for all the blocks and then again up iterate it this is an iterative method which is discussing.

The this particular equation is very similar as the basic Jacobi iteration step and if we can think this is actually be Jacobi iteration step. Why because for all off diagonals we are using x k which is the last iterative value and A ii inverse instead of A ii we are solving an equation like that a the instead of directly finding inverse we can use something like a Jacobi iteration also. So, this step can be exactly same as Jacobi iteration step if we use Jacobi iteration for finding A ii inverse. But, we can because the A ii matrix is smaller we can use some of the direct solution method or some other method to do it also.

However, if Jacobi iteration converges this step should also converge. So, the iteration is not in A ii inverse explicitly here iteration is on finding out xi k plus 1. This solution is done some in some other method, but iteration is what that fact that xi k plus 1 is found out and this is being it iterated again and again. So, this iteration must converge if the Jacobi iteration converges.

(Refer Slide Time: 15:09)



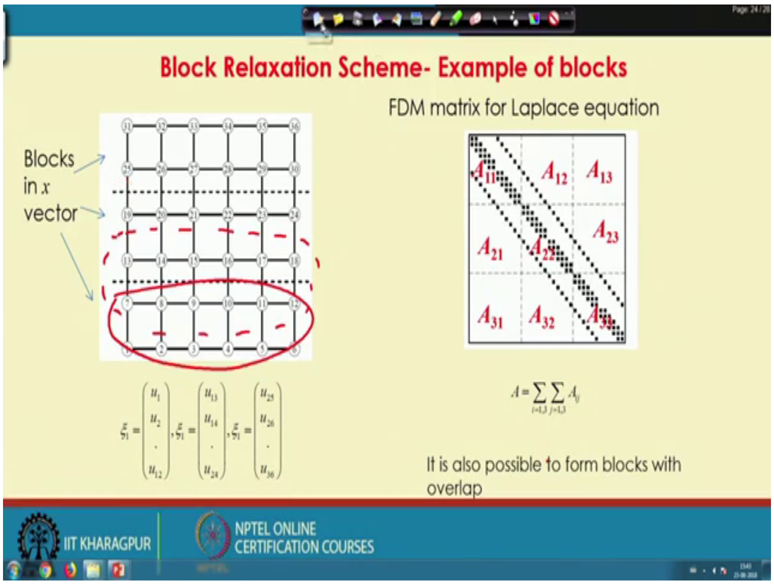
Here we can see an example of block relaxation this is the domain where we have thirty points and a finite difference matrix, of it is a pentadiagonal matrix. So, along any line there are five elements only and they are near that diagonals two super diagonals two sub diagonal and one diagonal line this is the mesh from which we got the finite difference matrix.

Now, we are doing a partitioning here. This is one group one group this is one group this is one group and these are the three diagonal blocks we will get so, block of x vector. So, that that is x 1 2 rather instead of x we should write u we are trying to solve A u vectors we will trying to solve A u is equal to b say.

So, xi 1 u 1 to u 12 is one group of block of vectors, xi 2 u 2 to u 24 is another group I think I should rewrite it this is xi 2, this is xi 3. So, what is that this is one particular block of vector this is one particular block of vector and this is one particular block of vector. So, what the matrix I will get that matrix will be decomposed into several blocks and these three are the diagonal blocks which will be directly multiplied with xi 1, xi 2, xi 3 when we are solving iterating for xi 1, xi 2, xi 3 etcetera.

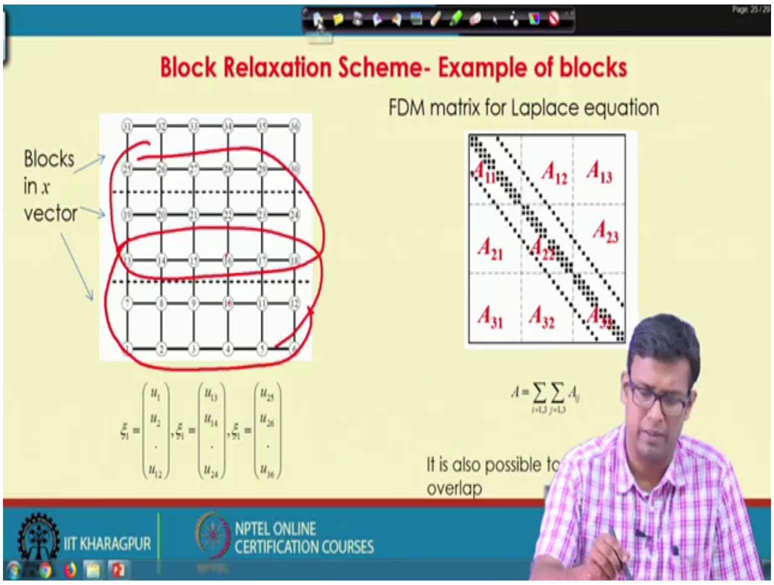
So, these are the A 1 A 11, A 22, A 33 the diagonal blocks and there are few of diagonal blocks it is important to look that these two of diagonal blocks are basically 0 matrices. So, we can eliminate some of the off diagonal blocks also from the calculations. And, A the net the actual matrix A when we because we are solving the matrix equation A u is equal to b, this A matrix is sum of all the small matrix blocks.

(Refer Slide Time: 17:39)



It is also possible to form blocks with the overlap; that means, that maybe this line is being shared by two different blocks. So, you can think of another blocking like this so, this is one block maybe this is another block and or rather let us do it like this.

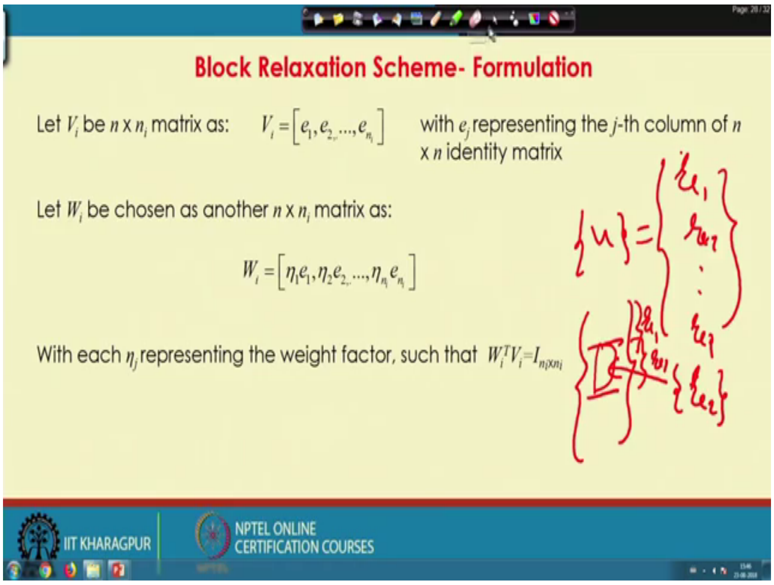
(Refer Slide Time: 18:05)



Maybe this is one block this is one block and we can say that this is another block this blocks have some points in points common to them. And we will see that block overlapping blocks are important just to maintain continuity of the solution as well as to have some of the parallels algorithms can only work with overlaps so, you will see about the overlaps later.

The best thing of overlap is that when you will try to solve in this domain this point will see the full stencil or full deciduous row for if there is an overlap considered we will come into the overlap later. But, it is also possible to form the blocks with overlap that is what is important to keep in mind here that this is a typical blocking with non-overlapping blocking what we are showing here is a non overlapping blocking, but we can also have overlapping blocking.

(Refer Slide Time: 19:11)

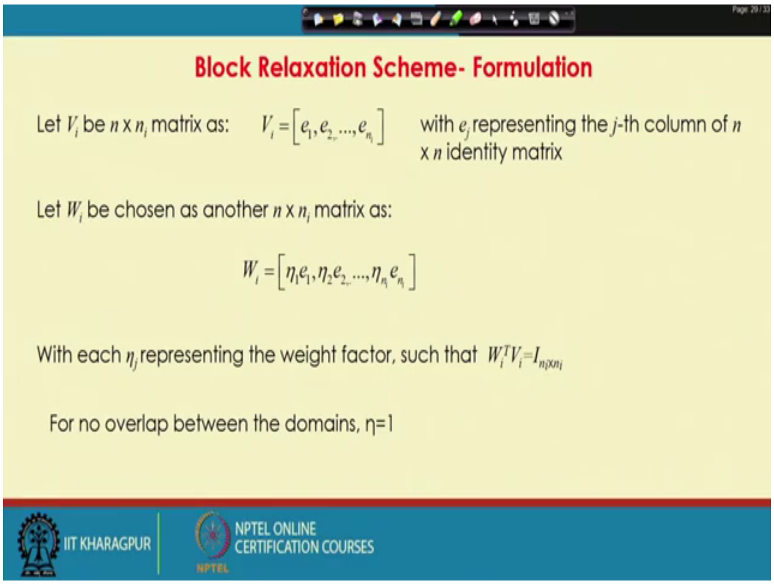


Let V we will see the formulation of a block relaxation scheme let V i be a n into n i matrix and V i is basically the columns of identity matrix of up to n i we will consider in V i. So, it has it will have n row rows and n i columns so, there are n i columns which had e j's and they represent the j-th column of an n into n identity matrix. So, it is V i is a truncated part of an identity matrix.

Where, W i is chosen and another n into n i matrix with eta 1 e 1, eta 2 e 2 etcetera where eta j represents the weight factor such that W i transpose V i is an identity matrix n i into n i. And this is required to map the solutions V i will be required or as well as W i to map the solutions xi to the right x say xi 2 I will take which will be after xi 1 or we can write that x rather the u matrix sorry u matrix will be xi 1, xi 2 so on xi a xi p. So, if I find out xi 2 that will come in one particular location of u matrix.

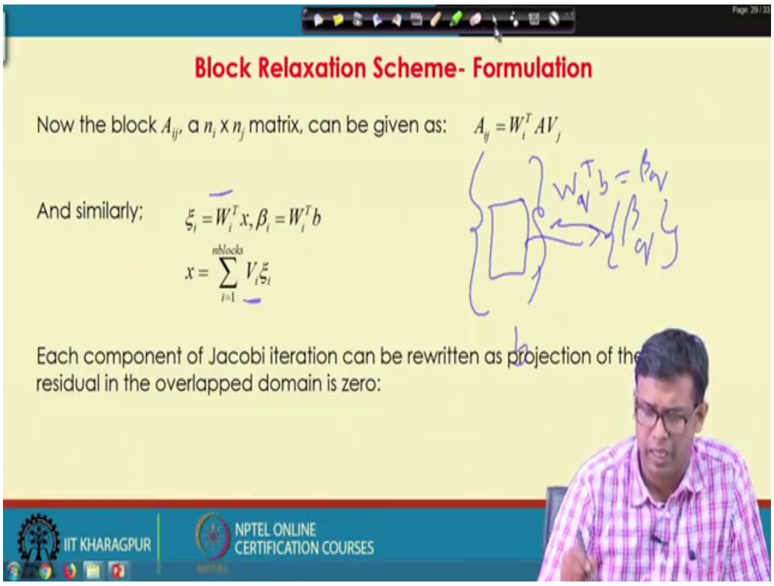
So, I have to multiply something with xi 2 to make it assembly able to u matrix which comes from these terms n 1 e 1, n 2 e 2. If there is or e 1 e 2, if there is over lap then if there is no overlap than e 1 e 2 e 2 so, xi 2 into e 2 will keep xi here if there is no overlap no overlap it will done there. If there is overlap then maybe this is the location of xi 1 and from here we will start the location of xi 2 so, you have to multiply something with e 1.

(Refer Slide Time: 21:25)



For no overlap between the domains eta is equal to 1 so, the V and W are basically same in these cases.

(Refer Slide Time: 21:29)



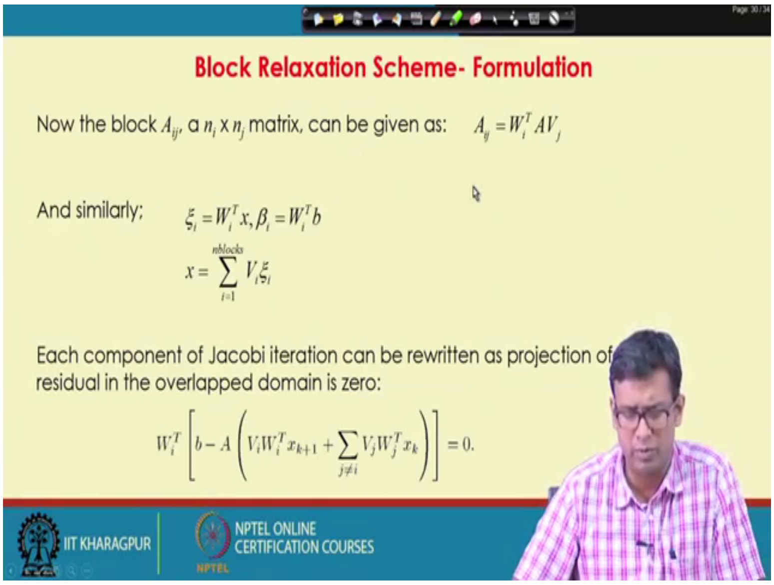
Now, the block A ij of n i n j matrix which is an n i n j matrix can be given as A ij's W i transpose AV j. And similarly xi will be given as W i transpose x i I am I have used x and u mixed x and u in the discussion. Either x or u is fine, because we started with the problem A u is equal to b, but sometimes I confuse myself writing A x is equal to V both are matrix equation be it u be it x is a solution variable.

However, in the present slides that I will upload I will make either x or u uniformly I will make it making by itself. So, nevertheless the xi i will go to the solution vector x we are discussing instead of u using this particular weight matrix W similarly beta will similarly be mapped into b.

So, this is the idea that you have the large matrix b and the part of this the large vector b is say beta q this. So, once you know beta q this will be multiplied with W q transpose b is beta q this will be multiplied like that. So, instead of multiplying W with beta we are doing, but W q is W is transpose multiplied with b that will map b to beta. And, if there are n blocks we can write x is equal to i is equal to one to n V i xi i. So, this is how xi to V x mapping is through V xi 2 x mapping is through V and x to xi mapping is through W they can be same for non overlapping case for overlapping case it will be W will have to carry and weight.

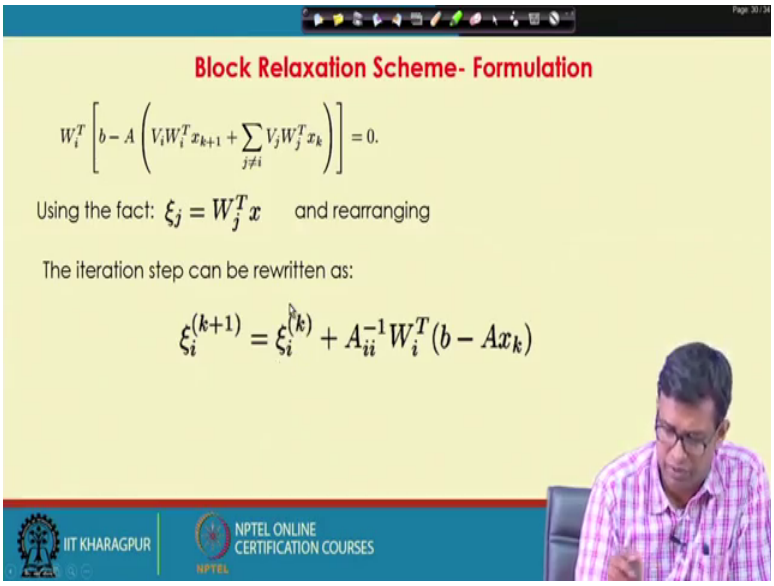
Each component of Jacob iteration can be rewritten as projection of the residual into of the in the overlap domain is equal 0. So, we will get a residual in a in the in a sub domain if there are projections then we will project the if there is overlap then the overlapping of the residual will be obtained. However, some of the residual as projected to the main domain should be 0.

(Refer Slide Time: 24:01)



And that comes as W i transpose b minus V i W i transpose x k plus 1 is if j is not is equal to a n V j W j transpose x k is equal to 0. If there is no overlap V i transpose V i W i transpose V j W j transpose this terms are 1, right. If there is no overlap V transpose W is an is again an identity matrix V and W are same. So, this becomes an identity matrix, but if there is overlap they are not identity matrix. So, this becomes the iteration Jacobi iteration step projecting that the new solutions the residual from the updated solution and non updated solution that residual must be 0 into the overlap domain.

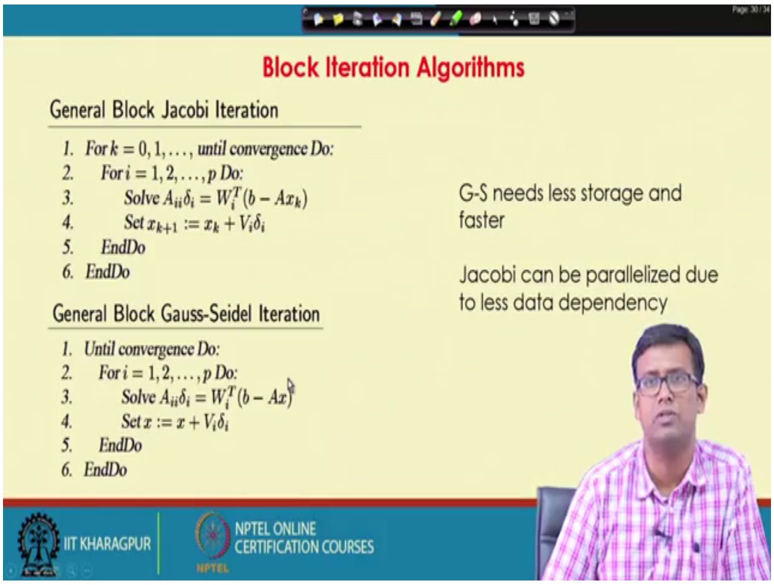
(Refer Slide Time: 24:49)



Now, we use the fact that xi is equal to W j transpose x. So, we multiply W j W i transpose with this and write j is equal to. So, whenever W i transpose x is there this is xi k. So, the new if the iteration step will be written as xi k plus 1 which comes from here and this is equal to W transpose AV is basically A ii so, A ii. So, this will give us A ii xi k plus 1 is equal to A ij x xi k like that.

So, this will be xi i k plus 1 is equal to xi i k which will again get here plus A ii inverse W i transpose b minus Ax k. And, this is what we exactly do in a Jacobi iteration step and we get a get the mapping weight function W and then multiply it is transpose with b minus Ax k. And, then multiply to A ii inverse and add with xi k and now using this the W is constructed W and V constructed I can get a general algorithm for Jacobi block relaxation scheme.

(Refer Slide Time: 26:11)



Which is k is equal to 0 to up to convergence do it up to convergence for each block write A ii delta I is equal to W i transpose b minus Ax k solve for delta I and then update x k plus 1 as x k plus V i delta i because delta is a solve for xi k plus 1 minus xi k that is in delta i. So, you have to multiply V i with and update x k plus 1. For better convergence instead of block Jacobi we can think of block Gauss Seidel because each scheme is using the last iteration value of x in block Jacobi in Gauss Seidel instead of using x k it is using the value x which is the last available value in the neighboring blocks rather.

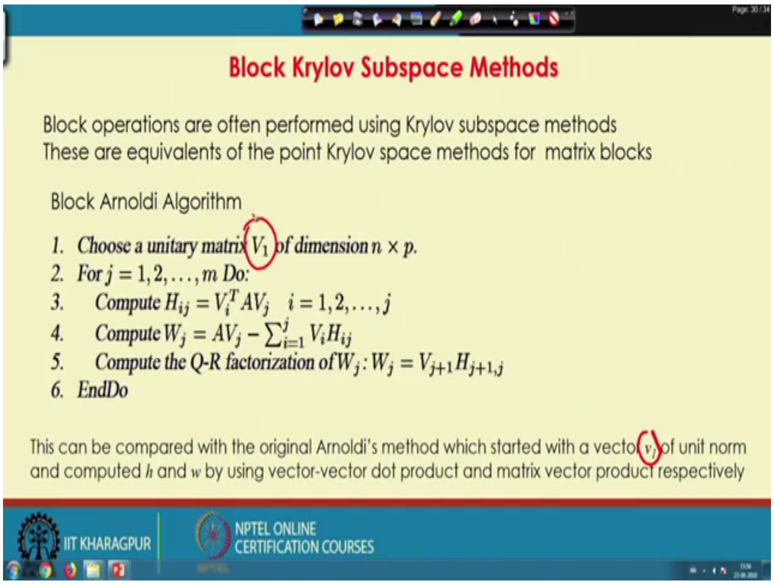
And, you solve not only neighboring blocks in the in the entire domain the last updated value you solved A ii delta i is equal to W i transpose b minus x and update in each block you update x like this. So, this solution can be done using GS or using Jacobi if we want do a full Jacobi in a block Gauss full Gauss Seidel in a block Gauss Seidel, this solution should also be done using Gauss Seidel.

But, however, when using this particular x it is updated from the neighboring values the neighboring values which the neighboring blocks or the last blocks which are above the xi's which are above this particular xi particular block will have the updated value and the other one will have the last iterate value, this very same as Gauss Seidel.

Gauss Seidel iteration needs less storage because we do not need to storage x k plus 1 in x k separate leads needs least storage and also faster because, you are using the last iteration value. However, the Jacobi method can be parallelized due to less data dependency in that sense that once we have the last iteration value we can go to the new Jacobi step. And, if we are doing in different processors none of the processor has to be aware of the fact that what is the updated value in other processors, everybody is using the last iteration value.

Gauss Seidel block Gauss Seidel cannot be parallelized in that way because it is data dependent one particular block has to be aware of what happened to the what is the updated value after the previous blocks in that during that particular iteration the blocks which came before this particular block has been updated. So, it is difficult to paralyze it.

(Refer Slide Time: 28:57)

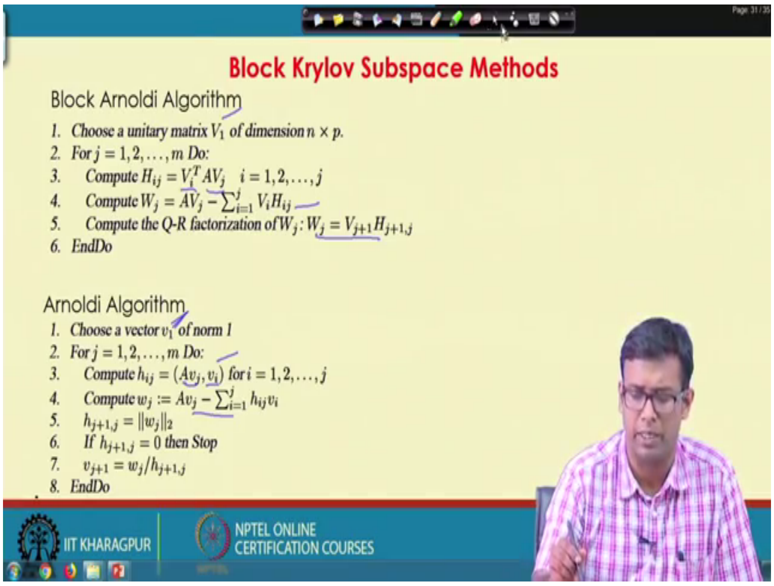


Similarly, Krylov subspace methods can have block operations and they are often performed using block your called block Krylov subspace methods. They are equivalent of Krylov subspace methods, but they are done for matrix blocks and there is a block Arnoldi method which starts with an unitary matrix V 1 dimension n into p computes the Hessenberg block matrix V i j transpose V i transpose AV j.

And then does an orthogonalization of it; that means, it from AV multiplies V with a a and subtract the V H and then does a Q-R orthogonalization of W to find out W. W is V j plus 1 H j plus 1 j and comes with all the basis functions of W and V. But, these are not basis of the Krylov subspace rather they are basis of the Krylov block subspaces.

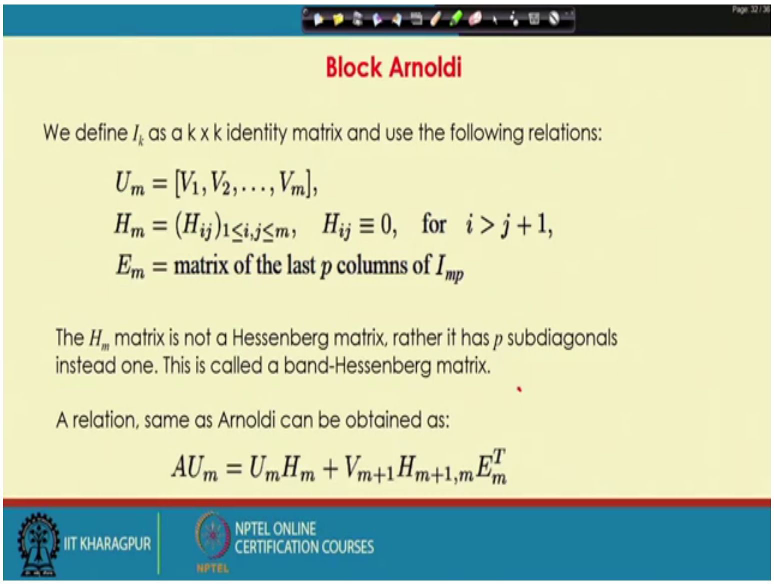
This can be compared with original Arnoldi's method where which started with a vector V 1 instead of starting with an unitary matrix V 1 it started with vector V 1 of unit norm and computed Hessen H and Hessenberg matrix and W by using vector vector dot product. Here with instead of matrix here vector vector dot product is used instead of vector and matrix vector dot product and also a matrix vector product has been used respectively. So, the product products are little the processes are little essentially similar, but instead of starting with a vector v instead of starting with a vector v they are starting with a matrix V 1 that is the difference.

(Refer Slide Time: 30:47)



This is block Arnoldi algorithm and this is if we compare with the Arnoldi method it starts with a vector V 1. So, this is the this these are the difference it starts with a vector. Here it starts with an unitary matrix, here this is compared as a matrix vector product and then it is dot with this. However, there is a matrix product and then it is transpose with H similarly the W j computing is also W j compute is also different it blocks subspace. So, we get the basis vectors for a block of Krylov subspace; Krylov subspace of the block matrices.

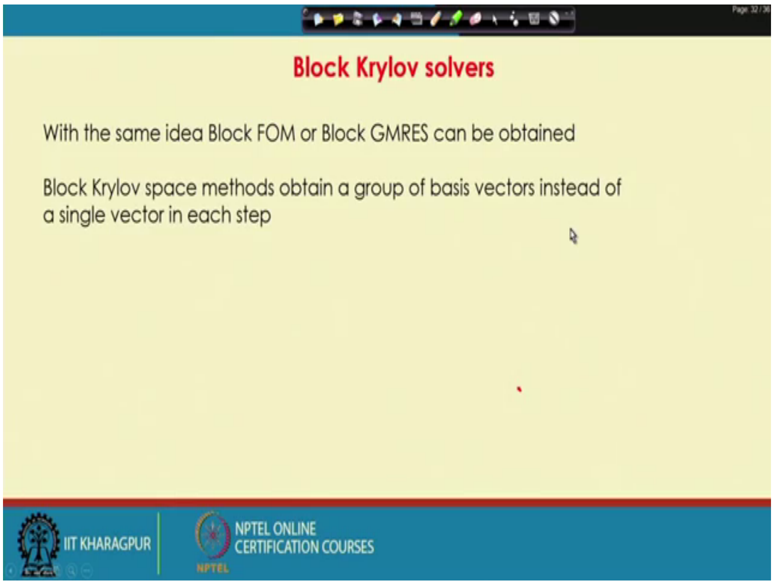
(Refer Slide Time: 31:35)



We defined I k as identity matrix and the and we will see that the following relationships hold same similar type of relationships which we got for simpler or point Arnoldi method. The following relationship shows that Hessenberg matrix H m is H m is not a Hessenberg matrix rather it has p sub diagonals and it is called a banded Hessenberg matrix all these are banded matrices.

And, a relationship like this holds which is very similar to our relationship we obtain the to get the Hessenberg relation between the Hessenberg matrix and the Krylov subspace basis vectors. Now, instead of a basis there is a basis matrix instead of the Hessenberg matrix is a band matrix. But, Krylov subspace type of relations are not were applicable for the solution vector x earlier now it is applicable for blocks of the solution vectors xi's.

(Refer Slide Time: 32:41)



The same idea with the same idea block full orthogonal method or block GMRES can be obtained. Block Krylov space methods obtain a group of basis vectors instead of a single vector in each step. So, instead of finding out basis vector it gets a basis matrix we can say and the iterations can be done between the blocks of the solution vectors. Within each block I can use even a direct solver, that is how this can give a faster solution or this can be paralyzed also.

So, we have discussed about block relaxation schemes also line relaxation schemes in last two sessions. And, we can see that block relaxation schemes has a potential of paralyzing the solution in a sense that the solution task will be distributed into number of computers which will operate together. And Instead of one computer doing all this n square order or n cube order of operations one computer will do less task and all of the computers we work in parallel, so that the computing tasks can be carried away in less time.

Instead of one people carrying a load if two people instead of one person carrying a load if two people carry the same load it can be much faster. That is the idea block that distribute the matrix into several blocks and use different computers to do it in parallel, that is one great advantage of block partitioning can be this can be exploited for parallel computing and we will look into it in the subsequent classes.

Thank you.