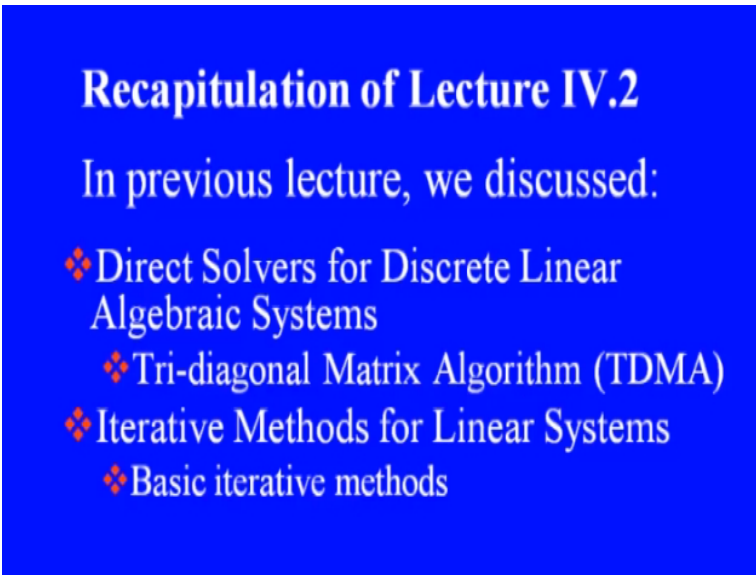


**Computational Fluid Dynamics**  
**Dr. Krishna M. Singh**  
**Department of Mechanical and Industrial Engineering**  
**Indian Institute of Technology – Roorkee**

**Lecture - 24**  
**Accelerated iterative Methods for Linear Systems**

Welcome to the third lecture in Module 4 on solution discrete algebraic systems. We had already seen the features of discrete algebraic systems solution methods for non-linear systems and direct and basic iterative methods for linear systems in previous lectures. Today, we shall focus on accelerated iterative methods for linear systems. So let us have recapitulation of what we did in lecture 2 of this module.

**(Refer Slide Time: 00:57)**



**Recapitulation of Lecture IV.2**

In previous lecture, we discussed:

- ❖ Direct Solvers for Discrete Linear Algebraic Systems
  - ❖ Tri-diagonal Matrix Algorithm (TDMA)
- ❖ Iterative Methods for Linear Systems
  - ❖ Basic iterative methods

We discussed direct solvers for discrete linear system specifically suited for CFT applications. In fact, we had a detailed look at what we call Tri-Diagonal Matrix Algorithm or TDMA which can be used for one dimensional problems and it can also work as a backbone of an iterative scheme, based on alternative direction method. Then we started off with iterative methods for linear systems which are most commonly used in CFDA programming.

And we discussed some basic iterative methods which are generally very slow to converge but nevertheless they very important for accelerated iterative methods, this is what we are going to focus today at accelerated iterative methods for linear systems.

**(Refer Slide Time: 01:56)**

## LECTURE OUTLINE

- ❖ Projection Methods
- ❖ Krylov subspace methods
- ❖ Pre-conditioners
- ❖ Methods for Symmetric Systems
- ❖ Methods for General Systems
- ❖ Multi-grid Methods

So outline, we will look at the category of methods what is known as Projection method and then we will look up a specific type of projection method which we call Krylov subspace methods and we will look at the Pre-conditioners which are required in these methods, the basic characteristics and then we will have a look at one method in detail for Symmetric systems, then we will innumerate the methods for General Systems which are not symmetric.

And in the end we will have a look at a very popular and very powerful class of methods which is called Multi-grid methods. Now what is a projection method? This is again an iterative scheme.

**(Refer Slide Time: 02:39)**

## PROJECTION METHODS

Given an available iterate  $\mathbf{x}_0$ , find the new approximant  $\mathbf{x} \in \mathbf{x}_0 + \mathbf{K}$  such that residual  $(\mathbf{b} - \mathbf{A}\mathbf{x}) \perp \Lambda$  where  $\mathbf{K}$  is the search subspace, and  $\Lambda$  denotes the subspace of constraints (Saad, 2003).

And here we will introduce some terms from the vector algebra that given an available iterate  $\mathbf{x}$ , let us say  $\mathbf{x}_0$ , we have to find a new approximate  $\mathbf{x}$  which will basically be from a set of vectors  $\mathbf{x}_0 + \mathbf{K}$  such that this residual  $\mathbf{b} - \mathbf{A}\mathbf{x}$ , is orthogonal to attain the subspace which is shown as  $\Lambda$  here. Again  $\mathbf{K}$  is called the search space or search subspace that is to say we are going to take out some vectors from this subspace to get our new iterate and your  $\Lambda$  denotes the subspace of constraints.

For a detailed and more definitive definition of projection methods, please have a look at the book by Saad published in 2003, we will give the complete bibliographic detail of this book towards the end of the lecture.

**(Refer Slide Time: 03:42)**

## KRYLOV SUBSPACE METHODS

Krylov subspace method is the projection method for which search subspace  $\mathbf{K}$  is Krylov subspace  $\mathbf{K}_m$  defined by

$$\mathbf{K}_m(\mathbf{A}, \mathbf{r}_0) = \text{span}\{ \mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \dots, \mathbf{A}^{m-1}\mathbf{r}_0 \}$$

Approximate solution  $\mathbf{x}_m$  is of the form

$$\mathbf{x}_m = \mathbf{x}_0 + q_{m-1}(\mathbf{A}) \mathbf{r}_0$$

where  $q_{m-1}$  is a polynomial of degree  $m-1$ .

Now this Krylov subspace methods, they are a specific type of projection method wherein the subspace  $\mathbf{K}$  that is your search subspace is Krylov subspace  $\mathbf{K}_m$  defined by  $\mathbf{K}_m(\mathbf{A}, \mathbf{r}_0)$ ,  $\mathbf{r}_0$  is your residual vector, so span of these vectors  $\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0$  and so on up to  $\mathbf{A}^{m-1}\mathbf{r}_0$  where  $\mathbf{A}$  is our system matrix. And in approximate solution  $\mathbf{x}_m$  is of the form  $\mathbf{x}_m = \mathbf{x}_0 + q_{m-1}(\mathbf{A}) \mathbf{r}_0$  where  $q_{m-1}$  is a polynomial of degree  $m-1$ .

So this is the formal representation of the new iterate, of course we will not actually compute any of these polynomials in real practice in real algorithms, this is just a starting point. Now in Krylov subspace methods we can have different type depending on the subspace of constraints chosen, so different parts of subspace of constraint leads to different Krylov subspace methods.

**(Refer Slide Time: 04:57)**

## ... KRYLOV SUBSPACE METHODS

Different choices of subspace of constraints  $\Lambda$  leads to different Krylov subspace methods, e.g.

- $\Lambda_m = K_m$  : Conjugate gradient method (for symmetric linear systems)
- $\Lambda_m = AK_m$  : GMRES (for general linear systems)

For instance, if  $\Lambda_m$  is same as  $K_m$  that is our constrained space, subspace is same as the search subspace then we get what we call Conjugate gradient method for symmetric linear systems and in case if you have  $\Lambda_m = AK_m$  we get a powerful method called GMRES for general linear systems. So there would be different possibilities for different types of Krylov subspace methods, for details please refer the book of Saad indicated earlier.

(Refer Slide Time: 05:35)

## ... KRYLOV SUBSPACE METHODS

Performance of a Krylov subspace solver such as PCG or GMRES depends on

- ❖ Pre-conditioner which affects
  - ❖ Convergence of iterations
  - ❖ Overall computing time
- ❖ Matrix-vector products: number of operations required for it affects overall computational efficiency (dependent on sparsity of the matrix)

Now performance of a Krylov subspace solver such as our Conjugate gradient or GMRES depends on two factors. The first factor is the choice of pre-conditioner which we use in these methods and the pre-conditioner primarily affects convergence of iteration and it also affects the

overall computing time because if each call to pre-conditioner is very time consuming that is going to affect our overall computing time as well.

Similarly, if the number of iterations are large then again our computing time would be affected. So ideally we would like to have a pre-conditioner which takes less time per iteration step and it converges very fast. So another factor which will affect the performances is the time or the number of operations which are required to compute matrix vector products.

So in fact this is the most time consuming sequence of operations involved in Krylov subspace methods whether be it our PCG or GMRES, so the time taken by these matrix vector products it affects the overall computational efficiency of a Krylov subspace algorithm and we can easily see that our matrix sparsity pattern would affect the computational operations required in obtaining our matrix vector products.

So this particular thing is dependent on sparsity pattern of the matrix. Fortunately, in CFT applications we will have very sparse matrices, so matrix vector products can be obtained very efficiently and this is one of the reasons why this Krylov subspace solvers are very popular in CFT for large scale problems.

**(Refer Slide Time: 07:23)**

## **PRE-CONDITIONERS**

Pre-conditioner properties to ensure a robust and efficient solution algorithm:

### **❖ Essential properties**

- ❖ Should help cluster the eigenvalues of the preconditioned system.
- ❖ Must be a constant linear operator in all the iterations.

Now let us have a look at some properties of pre-conditioners. So pre-conditioners are required to ensure that our method is a robust and efficient solution algorithm. So what are essential properties of a pre-conditioner? If you look a bit in more detail, which we did not have time today, the iterative solution process, the convergence is linked to the eigenvalues of our matrix. So a pre-conditioner should help cluster the eigenvalues of preconditioned system which is obtained after multiplying pre-conditioner to a system matrix.

So, if they cluster together that ensures faster convergences. The next one is very important restriction which we have got. The majority of Krylov subspace solver, they require that a preconditioner must be a constant linear operator in all the iterations. We cannot have a variable operator or a non-linear operator as our pre-conditioner for Krylov subspace methods.

**(Refer Slide Time: 08:34)**

### ... PRE-CONDITIONERS

#### ❖ Desirable properties

- ❖ Should take as little time as possible
- ❖ Should result in small number of iterations largely independent of problem size.
- ❖ There is a trade-off between preceding two desirable properties.
- ❖ For faster convergence, the preconditioning matrix (operator) must be a close approximation to matrix  $A$ .

Now, there are some Desirable properties, desirable in the sense they may not hold for all choices. So a pre-conditioner should take as little time as possible, that is what we would desire and the second is it should result in small number of iterations and the number of iterations will be independent of the problem size. Now both of these are conflicting requirements if we have a pre-conditioner which takes a very little time for instance if we use Jacobi methods or Gauss-Seidel methods as a pre-conditioner; those methods are very fast.

But the number of iterations required would be must larger with Jacobi as a pre-conditioner. And if we use a better pre-conditioner like multi-grid as a pre-conditioner, the computation time taken by the pre-conditioner would be fairly large but we would get very small number of iterations for convergences and further we would see in one or two numerical examples today that multi-grid used as a pre-conditioner results in the required number iterations which are independent of the system size, whether system size in thousands or millions or billions.

So that is why we say there is a trade-off between the preceding to desirable properties, we choose a pre-conditioner which requires less time, number of iterations would be increase and so on. So we will ultimately the overall computing time might suffer with such schemes. Theoretically, for first convergences the pre-conditioning matrix or operator as the case maybe must be a close approximation to matrix  $A$ , the  $A$  is our system matrix.

Why I have mentioned the word operator here is that we might also use certain sequence of operation and there affect would be what we call the affect of pre-conditioner for instance multi-grid process itself cannot be represented as a matrix, but it is nevertheless form a linear operator and hence can be used as a pre-conditioner with a Krylov subspace solver.

So in all our discussions on Krylov subspace methods wherever we see the algorithms we would put a symbolic form symbolic term  $M$  inverse which would  $M$  inverse multiplied by residual vector that would give us the affect of pre-conditioner, but  $M$  inverse is not--  $M$  is not a matrix per say, it is just  $M$  inverse multiplied by the residual that is the affect which we are looking for with our pre-conditioning operator. So what is the most commonly use pre-conditioners?

**(Refer Slide Time: 11:34)**



## ... PRE-CONDITIONERS

### Commonly used pre-conditioners

- ❖ Jacobi (diagonal) pre-conditioner
  - ❖ Very fast but poor rate of convergence
- ❖ Incomplete LU factorization
  - ❖ Effective but problem of fill-ins
- ❖ Approximate inverse pre-conditioners
  - ❖ Block diagonal pre-conditioners
  - ❖ Preconditioners based on Frobenius norm minimization
- ❖ Multi-grid pre-conditioners

The simplest one use is Jacobi or Diagonal pre-conditioner. But please remember this is a very fast pre-conditioner, but the number of iterations required are pretty large, in fact theoretically, the convergence of a Krylov subspace method would be ensured in number of iterations proportional to the system size, this is not an ideal situation. So Jacobi Pre-conditioner results in very fast, it is very fast per iteration step but results in poor rate of convergences, so our overall computing might have pretty large with Jacobi pre-conditioner.

Then recently, incomplete LU factorization have been developed, LU factorization results in a larger number of fill-ins, so where do we cut-off those fill-ins that would decide the effectiveness of these pre-conditioners. So the word incomplete we use because we will not carry out daily decomposition process in full, it will be stopped when certain allowed number of fill-ins are reached. This regions called LU or incomplete LU factorization pre-conditioner.

There are variety of them available in literature which you can look into (()) (12:51), these are very affective but there is a problem with fill-ins, what do you mean by fill-ins? Our system matrix is very fast but the operator matrix which we get pre-conditioning matrix which we get from incomplete LU factorization might not be as sparse as our original system matrix. So there might be memory trade-offs require with incomplete LU factorizations.

And the matrix products matrix vector products which we require with pre-conditioner that would also be pretty expensive. The next sub category of pre-conditioner of what we call Approximate inverse pre-conditioners, what (13:34) look for pre-conditioning matrix  $A$  is close to our matrix system matrix  $A$  then that would result in a very fast convergences size, so this what approximate inverse pre-conditioners aim at.

And there are few categories like Block diagonal pre-conditioner is one of them. And we also obtain approximate inverse pre-conditioner based on what we call Frobenius norm minimization. For details of this ILU or Block diagonal or Frobenius norm is what we call is fast approximate inverse pre-conditioners, please have a look at the book by Saad.

And then in then I would like to just mention this very powerful pre-conditioner multi-grid pre-conditioners, later on we would see that multi-grid algorithm can be used a very efficient standalone solver. But there are certain situations wherein instead of a solver we can use multi-grid as a pre-conditioner to a Krylov subspace algorithm. And the use of Multi-grid with a Krylov subspace algorithm results in what we call an optimal method for which the number of iterations required is independent the size-- system size.

Now next, we will have a look at a Krylov subspace solver symmetric systems, in fact we would put some more requirements here.

**(Refer Slide Time: 15:13)**

## KRYLOV SYBSPACE SOLVERS: SYMMETRIC SYSTEMS

- ❖ For symmetric and positive definite linear systems, the simplest and most commonly used Krylov subspace method is pre-conditioned conjugate gradient (PCG).
- ❖ Pre-conditioning step ( $\mathbf{z} = \mathbf{M}^{-1}\mathbf{r}$ ) essentially requires solution of a linear system  $\mathbf{M}\mathbf{z} = \mathbf{r}$
- ❖  $\mathbf{M}$  may not be explicitly defined: we only need an operator which is constant in each iteration.

We are looking for symmetric as well as positive definite linear systems. So there are quite a few Krylov subspace methods available for this family but the simplest and most commonly use Krylov subspace method is pre-conditioned conjugate gradient method, or in short we use this term our acronym PCG. We will have a look at the detail of this algorithm, this is very elegant and very simple to program be it on for a serial machine or massively parallel distributed memory machine.

That is the reason if we assured that our system matrix is symmetric and positive definite which is very often the case, the suggestion of Poisson problem on central different grids and a finite element grids. Since such situations PCG can be useful solution of a Poisson problem. Now in the algorithm which we are going to have look in detail is the pre-conditioning would be represented by  $\mathbf{z} = \mathbf{M}^{-1}\mathbf{r}$ , okay.

But we will not attempt to even just form  $\mathbf{M}$  or forget about taking inverse of  $\mathbf{M}$ , all that we mean by this particular step is solve a system of this type  $\mathbf{M}\mathbf{z} = \mathbf{r}$  this particular linear system, solve it  $\mathbf{M}$  get a vector  $\mathbf{Z}$  which is used in this algorithm. And  $\mathbf{M}$  may not even be explicitly defined, we only need an operator which is constant in each iteration, and that operator is represented by the symbol  $\mathbf{M}$ . So now just have a look detailed look at our PCG algorithm.

**(Refer Slide Time: 17:12)**

### Preconditioned Conjugate Gradient Method

- Set a tolerance for convergence, and choose an initial guess  $\vec{x}_0$  (for solution of system  $[A] \{x\} = \{b\}$ ).

begin PCG

- Compute (residual)  $\vec{r}_0 = \vec{b} - \underline{A} \vec{x}_0$ 
  - (i)  $\vec{z}_0 = \underline{M}^{-1} \vec{r}_0$  (effect of pre-conditioning)
  - (ii)  $\vec{p}_0 = \vec{z}_0$
- for ( $j=1, 2, \dots$  until convergence) do
  - $\alpha_j = (\vec{r}_j, \vec{z}_j) / (\underline{A} \vec{p}_j, \vec{p}_j)$

So Pre-conditioned conjugate gradient method. Since it is an iterative procedure so we would require that we start off with an initial guess and we would do our iteration to terminate somewhere so when we should terminate the iteration we should also have to specify some tolerance for the convergences of our iterations. So that is the first step which we need to fix in that we have to set a tolerance for convergences, set a tolerance for convergence check and choose an initial guess.

Let us represent our initial guess by vector  $x$  naught for solution of system equations which we are going to represent as  $Ax=b$ . So now let us write it in a pseudo code form or algorithmic form. So suppose we begin our subroutine being PCG, what we need to first do is, given our initial guess  $x_0$  let us compute what would be the initial residual. So compute residual, what we mean a residual here? Substitute this initial guess  $x_0$  in our system, okay that will not satisfy.

So let us find it for differences residual  $r_0$  would be defined as a vector  $b$ -matrix  $A$  times our initial guess  $x_0$ . We will also do some more computations. So this is one computation compute residual  $r_0$  then, next computation which we would require is to invoke our pre-conditioner, so find out or solve the system to get our initial guess for the vector  $z$ , which we call it  $Z_0$ , so  $M$  inverse  $r_0$ . And we would introduce few vectors here. So we will introduce a vector  $P_0=Z_0$ .

So these are the few things which we do before our iterations start, that is compute the residual vector, solve system equations with our pre-conditioner, so  $M^{-1}r_0$  basically this represents our affect of pre-conditioner. And then we will introduce a vector which is called  $P_0$  which are related search direction  $P_0=Z_0$  that is what we are going to initialize. Now next we will now start our iteration process.

So let us say for  $j=1, 2$  and so on until convergences. We have to repeat certain set of statements. Okay, so now let us write down those set of statement which are to be done at each iteration. We will computer scalar quantities which are required in our algorithm. the first scalar quantity which we will compute is  $\alpha_j$  which is given by  $r_j, z_j/A_{jj}, p_j$ . Now bracket term has got a very specific meaning.

**(Refer Slide Time: 23:10)**

$$\begin{aligned} \vec{x}_{j+1} &= \vec{x}_j + \alpha_j \vec{p}_j \\ \vec{r}_{j+1} &= \vec{r}_j - \alpha_j A \vec{p}_j \quad \text{(Convergence check)} \\ \vec{z}_{j+1} &= M^{-1} \vec{r}_{j+1} \quad // \text{Preconditioner} \\ \beta_j &= (\vec{r}_{j+1}, \vec{z}_{j+1}) / (\vec{r}_j, \vec{z}_j) \\ \vec{p}_{j+1} &= \vec{z}_{j+1} + \beta_j \vec{p}_j \end{aligned}$$

}

$( , ) \equiv \text{Inner product (dot product) of two vectors}$

\* Used for symmetric and positive-definite systems.

Let us write down some corner, so what we denote this is what we call Inner product, or dot product of two vectors. So this is terminology or all the symbols which we have shorten and have used to represent the dot product. So we have computed our one is scalar quantity  $\alpha_j$ . And once we know  $\alpha_j$  we have already got a computed value of  $P_z$ ; we can get the new iterate in terms of  $\alpha_j$ , so new iterate  $x_j$  that would be given as or rather call  $x_{j+1}$ . So it will be  $x_j + \alpha_j$  times, the vector  $p_j$  which we have calculated earlier.

So  $\alpha_j$  essentially represent a sort of weight factor and  $p_j$  what we call the correction vectors, so correction vectors multiplied by this scalar weight  $\alpha_j$  or to be added together we get an improved guess  $x_{j+1}$ . Next, we will compute our residual or rather, for computing the residual we have do not have to substitute in the system of equation that we do not have to compute  $r$ -A times something. We get fairly some simple formula here.

So  $r_{j+1}$ , so this would be the residual next step. This can be obtained by what the current residual and from this subtract the vector at  $A p_j$  this vector we have obtained earlier multiplied by  $\alpha_j$  subtract it from  $r_j$  and we get an estimate for the residual vector. Now the norm of this residual vector can help us at certain the convergence process. So there are various ways in which we can implement in our convergences check.

We can look at the norm of  $r_j$  or of the relative norm of  $r_j$  that you take the magnitude of  $r_{j+1}$ / magnitude of  $r_0$  which we started off with so that will give us a relative convergence things with our tolerance, if satisfied we can say now our solution is converged. The next thing what we will do is we are now going to compute a new set of vectors  $Z_j$ . So  $z$  of  $Z_{j+1}$  of the  $Z_{j+1}$  would be used in our next iteration to compute this  $\alpha_j$  and so on. So  $Z_{j+1}$  has to be obtained by solving pre-conditioner system.

So we would represent it symbolic as  $M^{-1} r_{j+1}$ , okay. So this is our affect of the pre-conditioner. Okay. Now, we would compute it under scalar quantity  $\beta_j$ ,  $\beta_j$  is given by the Inner product of  $r_{j+1}$ ;  $Z_{j+1}$ ; so dot products of this vector divided by  $r_j$ ,  $z_j$ . Now in this evaluation we would compute only one of the inner products and we will, we do not have to repeatedly compute in same iteration.

We do not have to compute  $r_{j+1}$ ,  $Z_{j+1}$  and  $r_j$ ,  $z_j$ ;  $r_j$   $z_j$  would be available from the previous iteration it is already been computed earlier, so we can save it and use it here. And we can now get an estimate of the correction vector  $P_{j+1}$  this would be given as  $Z$  of  $j+1$  times  $\beta_j$   $P_j$ . So the correction vector which we need to use for the next iteration this  $P_{j+1}$  that is obtained in terms of the correction vector which we get from in the pre-conditioning step plus scalar factor  $\beta_j$  into the correction vector from the previous iteration.

So now this completes our one iteration. And in fact here before we go and compute this  $Z_{j+1}$ , we would also perform our convergence check. So if you want you can do convergence here or you can do convergences check after comparing  $P_{j+1}$  to decide when we need to exit this iteration loop. So you can see some few nice things here that at each step of iteration in that is say in each iterations we require only one call to the pre-conditioner and we also require only two new sets of inner products.

So programming this PCG is very easy you can right, it is almost trivial if you have got, all that we need is we need a routine for computing this matrix vector product. We need one routine to form this scalar or inner product, okay. So this whole algorithm can be coded into few lines of code, in any language in C or FORTRAN whichever is the language of your choice.

So this PCG works very well as we have seen earlier this is useful or in fact it is applicable for symmetric and positive definite systems. But in practice it also works for the systems for which we are not sure, the system is matrix, symmetric but we are not very sure about the positive definiteness, many times this algorithms works pretty effectively. So whenever a system matrix is symmetric among Krylov subspace method our choice is very clear this is-- no algorithm can bit PCG in terms of its efficiency.

The overall computational efficiency of PCG but of course depends on our pre-conditioning step. So if you want to have robust and efficient PCG algorithm to solve a system of equations, we have to choose a suitable pre-conditioner. And you surprise now that this is-- there are lots and lots of pre-conditioner is available in literature and a still considerable amount for research work is going on in developing newer set of pre-conditioners which will give us more robust and efficient version of PCG.

Now let us have a look at list of Krylov subspace solvers for general systems. So the most popular Krylov subspace methods. At a top of the list the priority is what we call Generalized Minimum Residual called GMRES method developed by Saad. So this method is bit more bit difficult to program but it is very efficient method for general linear systems.

(Refer Slide Time: 32:38)

## KRYLOV SYBSPACE SOLVERS: GENERAL SYSTEMS

Popular Krylov subspace methods for general systems are

- ❖ Generalized Minimum Residual (GMRES) Method
- ❖ Biconjugate Gradient Method
- ❖ Conjugate Gradient Squared
- ❖ Biconjugate Gradient Stabilized (BICGSTAB) Method
- ❖ Flexible GMRES Method (Saad, 2003)

Then similar to conjugate gradient we can have what we call a biconjugate Gradient method for our asymmetric or general linear systems, so in this case we require two set of directions that is the reason why we have got biconjugate gradient method. And it would require at each iteration step two calls to pre-conditioner one for our normal system and one for transpose system. So there are few methods available in the literature, which do not require the use of the transpose matrix so conjugate gradient square method is one of them.

And then we have got 8 version of—and the version of biconjugate method which is called biconjugate Gradient Stabilized or BICGSTAB method. This again a very robust method and this method does not require forming the transpose of a matrix. Since it can also be used for non-linear problems wherein  $(A)$   $(33:46)$  matrix by taking finite differences and finding the transpose is very a difficult task.

So for this general systems, most effective choices which we call GMRES, GMRES only drawback, that is writing the code is bit involved. BICGSTAB is relatively simple, it is slightly more complicated or PCG algorithm which you have seen. Now please remember the PCG and GMRES, Biconjugate gradient, CGS or this BICGSTAB all these algorithms require a pre-conditioner which must be the same at each iteration.



So take care of this problem, Saad develop one method is called Flexible GMRES method. Now this Flexible GMRES is a variant of GMRES which does not put dis-restriction of consensus of a pre-conditioner at each step. So a pre-conditioner can vary from one iteration to another. So much so that we can have Flexible GMRES method in which GMRES itself can be used as a pre-conditioner. And thereby we get a flexible method which is fair pretty efficient.

Now I would not go into detail of any of these algorithms. I would refer you to the book of Saad which gives you detailed descriptions of the methods along with the algorithm and pseudo code which can be easily programmed. And now we move onto the next category of accelerated methods which we call Multi-grid methods. These methods are also referred to as multilevel methods, why they called multilevel, that would become pretty clear very soon.

**(Refer Slide Time: 35:54)**

## MULTIGRID METHOD

- ❖ Multigrid techniques exploit discretizations with different mesh sizes (or operators) of a given problem to obtain optimal convergence from relaxation techniques.
- ❖ **The basis of multi-grid:** Low frequency components of error become high frequency components on a coarser grid (on which these can be easily reduced using a relaxation or smoothing scheme such as Jacobi or Gauss-Seidel iterations)

So this Multi-grid techniques exploit discretizations with different mesh sizes. In normal finite differential finite volume or for (( )) (36:03) formulation we will, initially have one grid using which we obtain a discrete system equations. But solving there was system just on one Fine grid, we can frequent construct a sequence of coarser grids and use these solutions on those grids to obtain better rate of convergence.

In some cases, it may not be possible for us to form or to come up with a sequence of nested coarse grid, so in that case, we can instead form what we call coarse grid operators algebraically

so that is the reason why I have put here the different mesh sizes or operators. Here by operators I mean is coarse grid operators. And these-- when applied to a given problem obtained optimal convergences based on relaxation techniques.

A relaxation techniques basically a simple iterative schemes or basic iterative schemes seen earlier over Jacobi or Gauss-Seidel methods, those are refer to as relaxation techniques in the context of multi-grid method. And how does this method work? What is the basis of the Multi-grid? If we analyze the errors in our convergences process we can breakdown the errors in two ways. On a given grid, the errors can be classified as high frequency errors which are linked to the fine grid signs and the low frequency errors.

Now if you use a relaxation techniques like Jacobi iteration or Gauss-Seidel iteration, in very few iterations of Jacobi or Gauss-Seidel we can reduce or eliminate high frequency errors. But removing low frequency errors is very difficult. But those low-- this frequency of the errors linked to a grid size, move on to a coarser grid the low frequency error become high frequency errors on a coarser grid.

These grids which can be easily removed using a relaxation or smoothing scheme. We would use this term relaxation or smoothing interchangeably, okay. And the typical methods which we are going to use for relaxation smoothing or Jacobi or Gauss-Seidel iterations. So that is the reason why, if we can combined a sequence of grid, we have got a finest grid on which we would ultimately like to obtain our solution, choose a coarser grid on that coarser grid.

We can eliminate the low frequency errors, obtain a correction at to listed on our finer grid and thereby we would have an iterative technique which converges very fast. So this multi-grid methods they exhibit a convergences rate that is independent of the number of unknowns in the discretized system. We will see it numerically, we will have a look at one or two examples where we have used multi-grid as a pre-conditioner, okay.

**(Refer Slide Time: 39:24)**

## ... MULTIGRID METHOD



- ❖ Multigrid method exhibits a convergence rate that is independent of the number of unknowns in the discretized system. It is, therefore, an *optimal method*.

So in this particular sense this multi-grid methods are refer to as an optimal method and a complexity on all such size that there are almost of order  $N$  that is computational complexity of the method multi-grid methods, is scales linearly with a system size. So that is written the beautiful characteristic of multi-grid methods. Now what are main components of multi-grid algorithm?

(Refer Slide Time: 39:48)

## ... MULTIGRID METHOD

### Main components of multi-grid algorithm:

- ❖ Restriction operator  $R$  which maps (restricts) residuals on a finer grid to coarser grid.
- ❖ Prolongation operator  $P$  which extends (interpolates) solution on a coarser grid to a finer grid.
- ❖ Smoothing procedure,  $\text{SMOOTH}()$

We are dealing with different set of grids with actual physical grids or set of algebraic operator. So we need an operator  $R$  which we call restriction operator which maps or restricts residuals on a finer grid to the coarser grid. So we redistrict them from a large number of grid points to a

smaller number of grid that is why the word used here is restrict and the name of this operator as restriction operator, we normally use symbol capital R for the restriction operators.

So what we will do? We will use this operator to restrict our residual on a finer grid or a coarser grid and we will solve for a correction term on a coarser grid. Once that has been solved we now need to extend or interpolate that suggestion to a finer grid to obtain the solution, or improved iterate. So we need what we call prolongation operator which we will denote by symbol capital P which extends or interpolates solutions on a coarser grid to a finer grid.

And then we talked about relaxation scheme or smoothing procedure so let us say we will symbolically denote by the term SMOOTH within parentheses where we will provide a certain set of arguments. Now this prolongation and the restriction operators  $P$  might they will depend on the underlying differential equation that we will have a differential equation that we will have a differential prolongation operators for let us say Poisson equation and a different one for advection diffusion equation.

So this prolongation and restriction operators they will depend on our partial differential equations. And they will also depend on the Multi-grid strategy which we have chosen, by Multi-grid strategy we will discuss next what we, we can classify our strategy as two times.

**(Refer Slide Time: 42:00)**

## ... MULTIGRID METHOD

### Classification of multigrid method

- ❖ Geometric multi-grid (GMG) which is based on a sequence of nested grids
- ❖ Algebraic multi-grid (AMG) in which multi-grid components are generated based on a purely algebraic procedure.

So first one is what we call the Geometric multi-grid and very often in literature we will come across the acronyms GMG all capital letters. So, Geometric multi-grid is based on a sequence of nested grids. The reason we have used geometric is that geometrically we will actually obtain a sequence of  $(\cdot)$  (42:20) grids using a grid generator. And those set of nested grid would be used in our solution process.

Many a times specifically on unstructured grids it may be very difficult to generate a sequence of nested grid, even if you can generate of sequence of fine coarse and coarser grids calculation of our restriction and prolongation operators could be very complicated. So alternative which has been developed recently in past two decades is what is referred to as Algebraic multi-grid.

Now in algebraic multi-grid we actually do not form a sequence of nested grids only one grid is good enough that is a finest grid. And multi-grid component that is your prolongation and restriction operators they are generated based on a purely algebraic procedure. Since we are using a purely algebraic procedure there are many types of procedures which have been suggested in literature agglomeration based or a  $(\cdot)$  (43:25).

So since everything is being done, by looking at the coefficients of the matrix which has been generated on the fine grid this set of multi-grids methods are called algebraic multi-grid methods.

**(Refer Slide Time: 43:43)**

## ... MULTIGRID METHOD

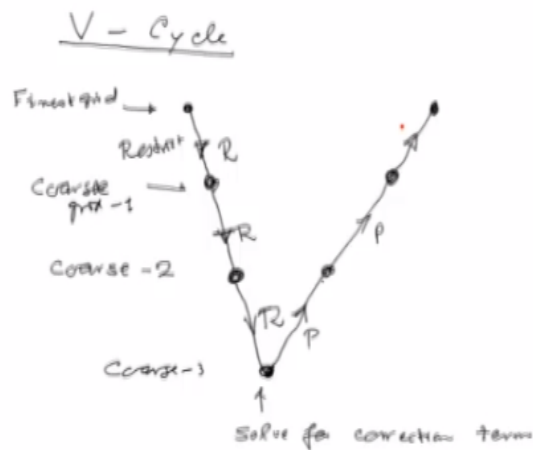
### Multigrid algorithm

- ❖ Basic algorithm is the same for both GMG and AMG.
- ❖ Considerable flexibility in choice of operators and cycle of inter-grid transfer sequence (e.g V or W cycle).
- ❖ Can be used as a **stand-alone solver** or **pre-conditioner** for a Krylov subspace method.
- ❖ For details of multi-grid methods, see Saad (2003) and Trottenberg et al. (2000).

Now as for the algorithm is concerned the basic algorithm or the format of the algorithm is saying both GMG that is your geometric multi-grid and algebraic multi-grid. In addition, we have got considerable flexibility in terms of the taking of the—taking of its prolongation and restriction operators and the cycle of integrate transverse process.

That is how do we, use our restriction in what sequence we use our restriction operators, in what sequence we prolong or interpolate our correction from a coarser grid to finer grid they are very assisted and it is available. Few typical examples V or W cycle, the symbol V or W that distance for the way the restriction and prolongations or employed. Let just have a geometric look at V cycle. And why we use the symbol V.

**(Refer Slide Time: 44:42)**



So suppose a top level, this is our, what we call finest grid. We will assume an initial guess, complete our residual, next restrict that residual, go to the next grid level, this is a coarser or coarse grid let us call it as 1, we want to stop here, further use restriction operator, second use and then obtain the residual on 8 grid which is even coarser than the coarser than coarse grid 1, let us call it coarse grid 2 and so on. We can have many sequences of such levels.

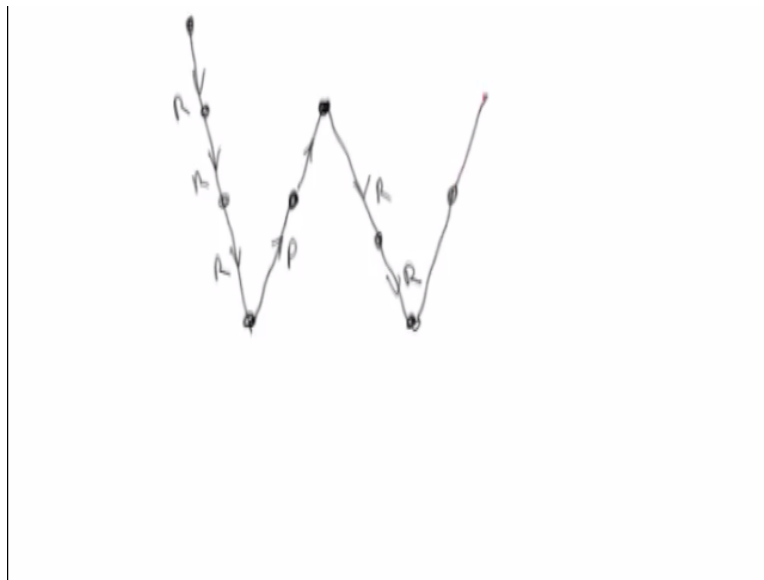
So this is coarse grid 3, so suppose we have selected that okay we are going to go for only 3 set of grids, so use restriction operator once, twice and thrice, so that each level we have used restriction operator, at the coarsest grid level then solve for the correction vector or correction

term and then once we have solved, we have basically solved a linear system which would be much smaller in size compared to our finest grid.

So we can use maybe a direct linear solver or a very accurate or converge Krylov subspace solution or a solution obtained using any Krylov subspace solver which has converged sufficiently and then this correction term would now be prolonged or what we call interpolated. It is prolonged our solution to next grid, to next finer grid to next finer grid, ultimately we would reach the finest grid level whenever our solution is sought.

So we proceed in this way that is what is referred to as V cycle.

**(Refer Slide Time: 47:37)**



Okay, we can also have W cycle, can bend the student same way, we have got a sequence of grids, you go once or apply this restrictions, restrictions, restrictions, restrictions, and obtain a solution here, then prolong it, use the prolongation operators to get a correction term here but instead of proceeding all the way to the top of the hierarchy. We might proceed only to the intermediate level and further we would obtain certain residual here.

We would repeat the process; we will go back to the coarser and coarser grid levels, solve for the correction vector once again and then do the final sort of prolongations to reach to our finest grid levels. So this is the finest grid level. So the way this process has been done, this resembles our

W symbol, that is the reason why it is called as W cycle. So either of these cycles can be used, in fact V cycle looks a bit simpler and is the one which is used most widely.

Now as I mentioned earlier, multi-grid method can be used as standalone solver. That is to say it can be used to solve a problem by itself or it can be used as a pre-conditioner for Krylov subspace method for instance our conjugal gradient or GMRES, so it can be used as a pre-conditioner for PCG or GMRES. There are certain advantages of using it as a pre-conditioner and for details of these different algebraic multi-grid and geometric multi-grid methods please see the books by Saad and Trottenberg et al.

There are a few other multi-grid books but these two are very good, specifically the book by Saad that will give us a compendium of all types of iterative schemes including multi-grid method and their parallel extensions. Before we proceed further let us have a look at a sample multi-grid algorithm.

(Refer Slide Time: 50:06)

V - Cycle Multigrid

$A^{(j)} \Rightarrow$  matrix obtained at  $j^{\text{th}}$  grid level.  
 $R^{(j)} \Rightarrow$  Restriction operator from  $j^{\text{th}} \rightarrow (j-1)^{\text{th}}$   
 $P^{(j)} \Rightarrow$  Prolongation operator from  $j \rightarrow j+1$

$\overline{W}^{(j)} = \text{SMOOTH}(W^{(j)}, A^{(j)}, r^{(j)})$

Multigrid algorithm

$$x_{m+1}^{(j)} = \text{VCYC}(r_m^{(j)}, j, x_m^{(j)})$$

VCYC algorithm: recursive procedure.

- If  $j = 0$  (coarsest grid), use a direct solve to solve  $A^{(0)} x^{(0)} = r^{(0)}$

Okay, so now let us have a look at a typical V cycle multi-grid. Okay, we would use our symbols for instance your system matrix is A, to denote a certain grid level we are going to use the subscripts, so  $A_j$ , this would represent the matrix obtained at  $J^{\text{th}}$  grid level. Similarly,  $P$  superscript  $J$ , this is restriction operation, operator from  $J^{\text{th}}$  grid to  $J-1^{\text{th}}$  grid and this  $P$  superscript  $j$  this would represent our prolongation operator.



Operator from  $J$ th grid to  $J+Y$  and we would represent our smoothing process as let us say we have got vector  $W$ , so  $\bar{W}$ , that over bar would represent the affect of this smoothing process at grid level  $J$ , so this  $\bar{W}_J$  or say  $\bar{W}_J$  this is the effect of the smoothing procedure which could be our Jacobi iteration or Gauss-Seidel iteration used starting with the initial guess  $\bar{W}_J$  for a system  $A_J$  and right hand side given by this. So this is our smoothing procedure.

Now with this notation, we will say that how do we obtain our multi-grid solution. So multi-grid algorithm is a recursive algorithm, so the solution which we would obtain, that is  $X_{m+1}$ , supposing we use  $V$  cycle, so let us call it  $VCYC$  in short form for our algorithm or  $M$  superscript  $j$   $X_m$  subscript  $j$ , so this is our  $V$  cycle algorithm.

Now what is the expanded form of  $V$  cycle algorithm, so  $VCYC$ , it is a recursive process which will depend on our grid level  $J$ , so recursively do the following steps. First let us check the grid level, if we have breached the coarsest grid level, so if  $J=0$  which represents the coarsest grid, so in this case what we will do we will use a direct solver to solve the system  $A_0$ ,  $X_0=R_0$ . So this is what we do at the bottom of the grid level.

(Refer Slide Time: 55:14)

Multi-grid algorithm

$$x_{m+1}^{(j)} = VCYC(\tau_m^{(j)}, j, x_m^{(j)})$$

VCYC algorithm: recursive procedure.

- If  $j = 0$  (coarsest grid), use a direct solver to solve
 
$$A^{(0)} x^{(0)} = r^{(0)}$$
- Else
  - Pre-smoothing  $\bar{x}_m^{(j)} = SMOOTH^{u_1}(x_m^{(j)}, A^{(j)}, r_m^{(j)})$
  - Defect computation
 
$$\bar{r}_m^{(j)} = r_m^{(j)} - A^{(j)} \bar{x}_m^{(j)}$$
  - Restriction
 
$$r_{m-1}^{(j+1)} = R^{(j+1)} \bar{r}_m^{(j)}$$
  - Coarse grid correction
 
$$x_m^{(j)} = x_m^{(j)} + P^{(j)} VCYC(\tau_{m-1}^{(j+1)}, j+1, 0)$$
  - Post smoothing
 
$$x_{m+1}^{(j)} = SMOOTH^{u_2}(x_m^{(j)}, A^{(j)}, r_m^{(j)})$$
- End if

Else we have to do a set of operations and these operations are called, first one we will have to do what we call pre-smoothing. Now pre-smoothing simply means that we have got our solution

vector  $x_m$  at grid level  $J$  obtained, it is a smoothed version by applying a set of or say, a few iterations of our smoothing procedure.

So it is smooth, let us use a superscript new one to denote that number of iterations of our Jacobi or Gauss-Seidel iterations we want to use on starting with the initial guess  $x_m$  subscript  $J$  or system matrix  $A_J$  and this residual vector  $r_m$  of  $J$ . Okay this new one could be as small as 2 or 3, that we might just use 2 or 3 iterations of the Gauss-Seidel or Jacobi iteration for pre-smoothing. Then the next step is what we call defect computation.

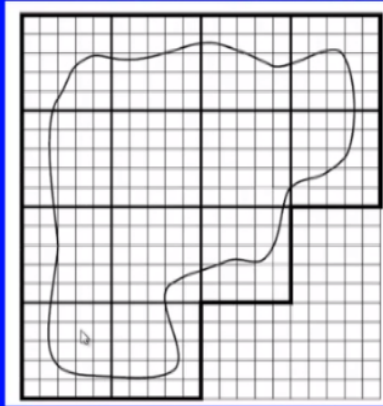
As you compute the residual with the smoothed value of  $x_m$   $J$  which we have computed earlier, so let us call it  $r_m^J = r_m^J - A_J x_m^J$ . And then it has to be restricted to the coarser grid, so restriction process, sorry we will go to coarser grid level  $J-1$ , apply the restriction operator  $R_{J-1}$  to our vector  $r_m^J$  and then once we have got the residual vector on coarse grid.

Then find out coarse grid correction, this is  $x_m^J$  as the previous value of  $x_m$   $J$  + our projection operator at grid level  $J$  multiplied by the result of  $V$  cycle operated on our  $r_m^{J-1}$  at grid level  $J-1$  with the residual of 0. And next is what we call, once we have got this coarse grid correction we have to apply what we call post smoothing, post smoothing simply represents that after post smoothing we will get our new iterate for the next multi-grid iterations  $x_m$  at grid level  $J$  smooth.

This is of course we use new to iterations of Jacobi process  $x_m^J - A_J^{-1} r_m^J$ , so that is it and so if you look carefully here we have got the procedure which calls itself the  $V$  cycle algorithm itself calls it in the middle  $V$  cycle-- so that is why it is a recursive process. So for actual details of algorithm and its implementation please have a look at the two books which I have referred earlier Saad and Trottenberg. Now let us have a quick look at its application as a pre-conditioner.

**(Refer Slide Time: 59:48)**

## MULTIGRID PRECONDITIONER: DOMAIN DECOMPOSITION



Thick outline denotes the fictitious domain.

Let us say we want to apply or we want to solve our problem on a very complicated grid so on which generating a sequence of natural grid would be very difficult, so what we can do is we can use a Cartesian grid but use the geometric multi-grid or generate the set of nested grids only for the rectangular sub blocks. So on each rectangular sub block we can easily generate the sequence of nested grids, obtain the multi-grid correction on that as if pre-conditioner and use that in our Krylov subspace solver.

(Refer Slide Time: 01:00:26)

## ... FICTITIOUS DOMAIN MULTIGRID PRECONDITIONER

- Set  $\bar{r}^l = (r^k, 0)^T$ ,  $\bar{z}_0^l = (z_0^k, 0)^T$ .
- $n$  Multigrid cycles:  

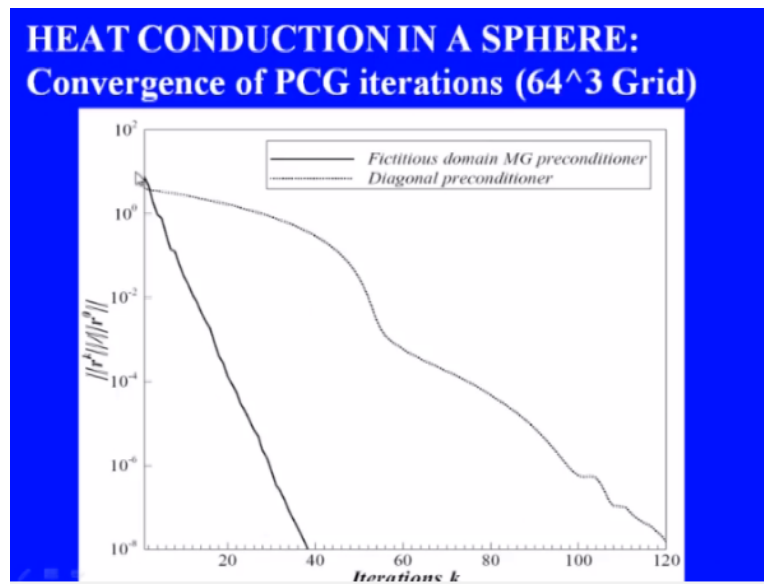
$$\bar{z}_m^l = \text{VCYC}(\bar{r}^l, l, \bar{z}_{m-1}^l), \quad m = 1, \dots, n.$$
- Correction term:  $z^k = \bar{z}_n^l|_{\Omega^h}$

So what we basically do is we have got a residual, now to get the residual at each node which may not be the part of our actual domain like these outer nodes, there we will simply set residual

value to be 0. So 0 extension of our residual vector  $r_k$  and let us say in our PCG, so extended by adding 0s to get this residual term for multi-grid.

Similarly, we will set the initial guess  $z_0$  at  $z_{0k0}$  and then we can use  $N$  number of multi-grid cycles, that  $n$  could be 1, 2 or 3 at the most 3 that is what we will need to generate a solution ZML by using V cycle multi-grid. So V cycle multi-grid will give us a solution of the problem starting off with this our right hand side vector and ZNL that will give us the correction term. And next all that we need to do is just choose those components of ZNL which fall into our complicated domain.

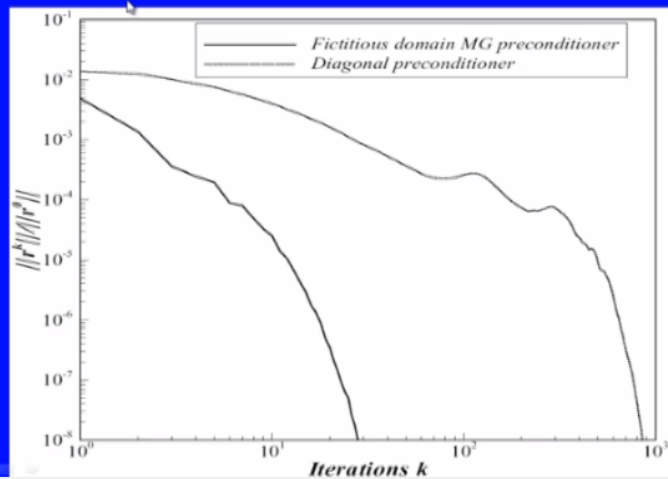
(Refer Slide Time: 01:01:41)



And this is what we applied using a Cartesian grid for heat conduction problem and 64 cube grid which translates to a few 100,000 grid points and this is a comparison of 2 that is our Jacobi preconditioner which takes around 120 iterations and this is our multi-grid pre-conditioner, the solution converges to the tolerance of 10 to the power -8 in 30 to 40 iterations. So this was a grid size of few 100,000s.

(Refer Slide Time: 01:02:14)

### Convergence of PCG iterations ( $N = 7,122,944$ )



Now let us take a very complicated problem and the system size is very large, it is 7 millions. It is a similar problem, Poisson equation for pressure which is similar to a heat conduction equation. So for this problem involving few million, now here I have used algorithmic matrix to indicate number of iterations. This Jacobi we need many thousands of iterations with our Jacobi pre-conditioner, with multi-grid pre-conditioner, the number of iterations is restricted to 30.

Similarly, I have also tried it for a few billions, the number of grid points on few billions and again what we have seen is with multi-grid pre-conditioner, the number of iterations required are of the order of 30 to 40. So that proves our multi-grid method or along with multi-grid pre-conditioner we have obtained what we can call an optimal method which does not depend on the size of the system.

**(Refer Slide Time: 01:03:12)**

## REFERENCES

- ❖ Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (2002). Numerical Recipes in C++. Cambridge University Press, Cambridge.
- ❖ Saad, Y. (2003). Iterative Methods for Sparse Linear Systems, SIAM, Philadelphia.
- ❖ Trottenberg, U., C. W. Oosterlee, C. W. and A. Schüller (2000). Multigrid, Academic Press, London.

## WEB RESOURCES

<http://www.netlib.org>

<http://www.mgnet.org>

Now references, the book by Press gives us Numerical Recipes book gives some algorithms related to the Krylov subspace solvers and their corresponding course in C, C++ FORTRAN, 77 of FORTRAN 90 language. Saad's book is a very comprehensive book on iterative methods including multi-grid systems and this Trottenberg's book which is exclusive to multi-grid method. It discusses both geometric and algebraic multi-grid algorithms for different partial differential equations.

There is some web resources which you can consult, you can get free programs on them and reading materials, netlib.org it contains the collection of different routines or algorithms based on the Krylov subspace methods and this mgnet.org, this particular website is dedicated to the material related to multi-grid methods both geometric and algebraic multi-grid. So I would strongly advise you to have a look at this mgnet.org website to have a look at the further details about the algorithms and programs are related to multi-grid methods.