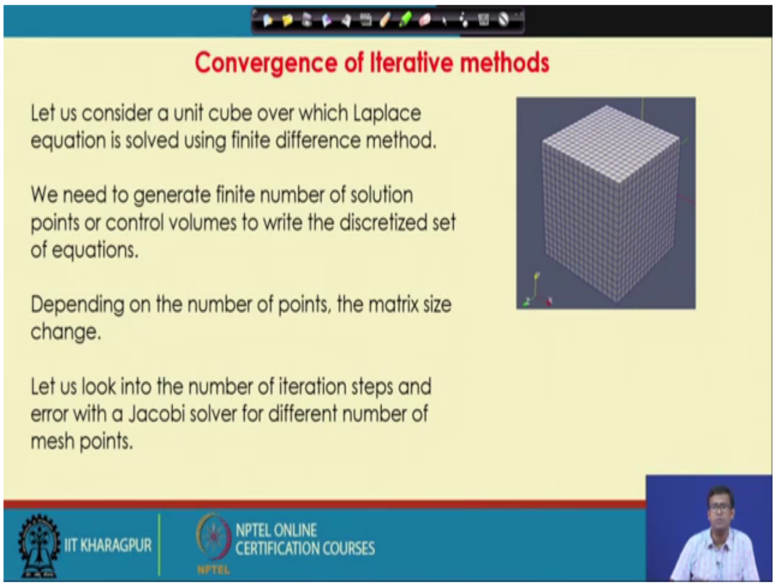
Matrix Solvers
Prof. Somnath Roy
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture - 59
Multigrid Methods – I

Welcome. So, we are continuing our discussion on how to improve convergence rate of Matrix Solvers. That is given a matrix solver for a large matrix. What are the techniques which can be applied over that particular given matrix solver and the convergence rate can be improved. So, that we get faster solution.

So, earlier we have discussed about preconditioning techniques and this session will discuss on Multigrid methods.

(Refer Slide Time: 00:43)



Convergence of Iterative methods

Let us consider a unit cube over which Laplace equation is solved using finite difference method.

We need to generate finite number of solution points or control volumes to write the discretized set of equations.

Depending on the number of points, the matrix size change.

Let us look into the number of iteration steps and error with a Jacobi solver for different number of mesh points.

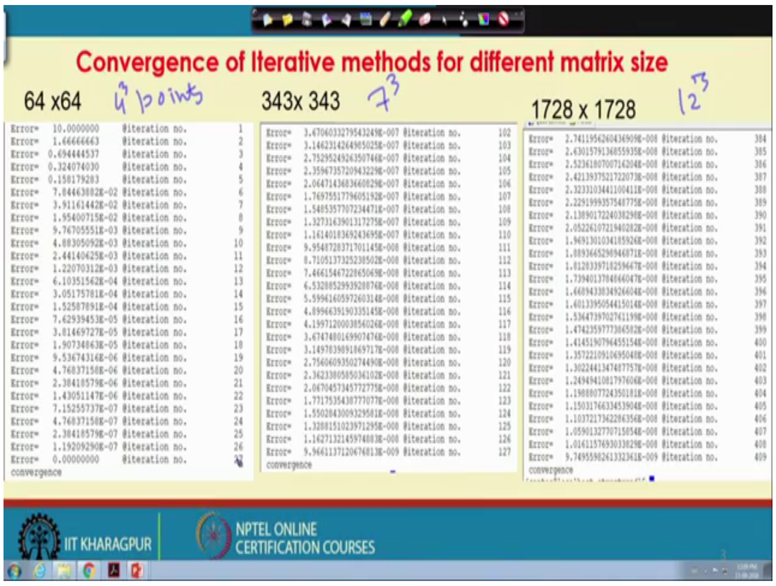
So, will focus again our discussion on what are the issues for convergence of iterative methods. So, let us think of a unit cube over which Laplace equation is solved using finite difference method. So, the geometry is already given and the governing equation is a second order differential linear differential equation which is Laplace equation and that is being used. We have all the known boundary condition we have to solve this using computer methods. So, we will use something like a finite difference method and get a set of equation difference equations which is essentially a matrix equation and we have to solve that.

We need to generate finite number of solution points and or control volumes to write discretize set of equations. So, this is a say this is a unit cube over which we take n points in each direction and gain gate total n cube internal points for which you have to write equations each point will have it is own equation and essentially we will get a matrix equation. Depending on the number of points the matrix size will change true. If I takes for example, if I take n 10 this is 10 by 10 by 10 total 1000 internal points are there. So, the coefficient matrix will be 1000 by 1000. And if I take n is equal to 100 this will be 10 to the power 6 by 10 to the power 6 matrix.

So, the matrix size depends on the number of points and we need to take large number points in order to have better accuracy because we have looked into at least finite difference methods where the accuracy is a function between the of is a function of the spacing between two adjacent grid points or nodes. So, if you increase number of points this distance reduces and the code accuracy increases the solution is more close to the physical solution. So, we often need to take a very large number of points or which will need the large matrix to solve this sets of equation.

And now let us look into the number of iteration steps required and how the error that is difference l 2 norm of difference between the solution vectors at different iteration level reduces for a Jacobi point solver when in the same geometry we take different number of grid points.

(Refer Slide Time: 03:10)



Convergence of Iterative methods for different matrix size

64 x64 4^3 points

```
Error=   10.0000000      @iteration no.          1
Error=   1.66666663      @iteration no.          2
Error=  0.694444537      @iteration no.          3
Error=  0.324074030      @iteration no.          4
Error=  0.158179283      @iteration no.          5
Error=   7.84463882E-02 @iteration no.          6
Error=   3.91161442E-02 @iteration no.          7
Error=   1.95400715E-02 @iteration no.          8
Error=   9.76705551E-03 @iteration no.          9
Error=   4.88305092E-03 @iteration no.         10
Error=   2.44140625E-03 @iteration no.         11
Error=   1.22070312E-03 @iteration no.         12
Error=   6.10351562E-04 @iteration no.         13
Error=   3.05175781E-04 @iteration no.         14
Error=   1.52587891E-04 @iteration no.         15
Error=   7.62939453E-05 @iteration no.         16
Error=   3.81469727E-05 @iteration no.         17
Error=   1.90734863E-05 @iteration no.         18
Error=   9.53674316E-06 @iteration no.         19
Error=   4.76837158E-06 @iteration no.         20
Error=   2.38418579E-06 @iteration no.         21
Error=   1.43051147E-06 @iteration no.         22
Error=   7.15255737E-07 @iteration no.         23
Error=   4.76837158E-07 @iteration no.         24
Error=   2.38418579E-07 @iteration no.         25
Error=   1.19209290E-07 @iteration no.         26
Error=   0.00000000      @iteration no.         27
convergence
```

343x 343 7^3

```
Error=   3.6706033279543249E-007 @iteration no.        102
Error=   3.1462314264905025E-007 @iteration no.        103
Error=   2.7529524926350746E-007 @iteration no.        104
Error=   2.3596735720943229E-007 @iteration no.        105
Error=   2.0647143683660829E-007 @iteration no.        106
Error=   1.7697551779605192E-007 @iteration no.        107
Error=   1.5485357707234471E-007 @iteration no.        108
Error=   1.3273163901317275E-007 @iteration no.        109
Error=   1.1614018369243695E-007 @iteration no.        110
Error=   9.9548728371701145E-008 @iteration no.        111
Error=   8.7105137325238502E-008 @iteration no.        112
Error=   7.4661546722865069E-008 @iteration no.        113
Error=   6.5328852993928876E-008 @iteration no.        114
Error=   5.5996160597260314E-008 @iteration no.        115
Error=   4.8996639190335145E-008 @iteration no.        116
Error=   4.1997120003856026E-008 @iteration no.        117
Error=   3.6747480169907476E-008 @iteration no.        118
Error=   3.1497839891869717E-008 @iteration no.        119
Error=   2.7560609350274490E-008 @iteration no.        120
Error=   2.3623380585036102E-008 @iteration no.        121
Error=   2.0670457345772775E-008 @iteration no.        122
Error=   1.7717535438777077E-008 @iteration no.        123
Error=   1.5502843009329581E-008 @iteration no.        124
Error=   1.3288151023971295E-008 @iteration no.        125
Error=   1.1627132145974883E-008 @iteration no.        126
Error=   9.9661137120676813E-009 @iteration no.        127
convergence
```

1728 x 1728 12^3

```
Error=   2.7411956260436909E-008 @iteration no.        384
Error=   2.6301579136855935E-008 @iteration no.        385
Error=   2.5236180700716204E-008 @iteration no.        386
Error=   2.4213937521722073E-008 @iteration no.        387
Error=   2.3233103441100411E-008 @iteration no.        388
Error=   2.2291999357548775E-008 @iteration no.        389
Error=   2.1389017224038290E-008 @iteration no.        390
Error=   2.0522610721940282E-008 @iteration no.        391
Error=   1.9691301034185926E-008 @iteration no.        392
Error=   1.8893665298946871E-008 @iteration no.        393
Error=   1.8128339718259667E-008 @iteration no.        394
Error=   1.7394013784866047E-008 @iteration no.        395
Error=   1.6689433834926604E-008 @iteration no.        396
Error=   1.6013395054415014E-008 @iteration no.        397
Error=   1.5364739702761199E-008 @iteration no.        398
Error=   1.4742359777306582E-008 @iteration no.        399
Error=   1.4145190796455154E-008 @iteration no.        400
Error=   1.3572210910695048E-008 @iteration no.        401
Error=   1.3022441347487757E-008 @iteration no.        402
Error=   1.2494941081797606E-008 @iteration no.        403
Error=   1.1988807724350181E-008 @iteration no.        404
Error=   1.1503176633453904E-008 @iteration no.        405
Error=   1.1037217362286356E-008 @iteration no.        406
Error=   1.0590132770715854E-008 @iteration no.        407
Error=   1.0161157693033829E-008 @iteration no.        408
Error=   9.7495598261332361E-009 @iteration no.        409
convergence
```

So, the first one is I have taken four points for internal points and all the outer boundaries are Dirichlet boundaries I am not considering them. So, total 4 cube 64 into 64. Second one we have taken so, total 4 cube points, the second one is where we have taken total 7 cube points and the third one we have taken I guess 12 squared into 12. So, 12 cube points 144 into 12 yeah.

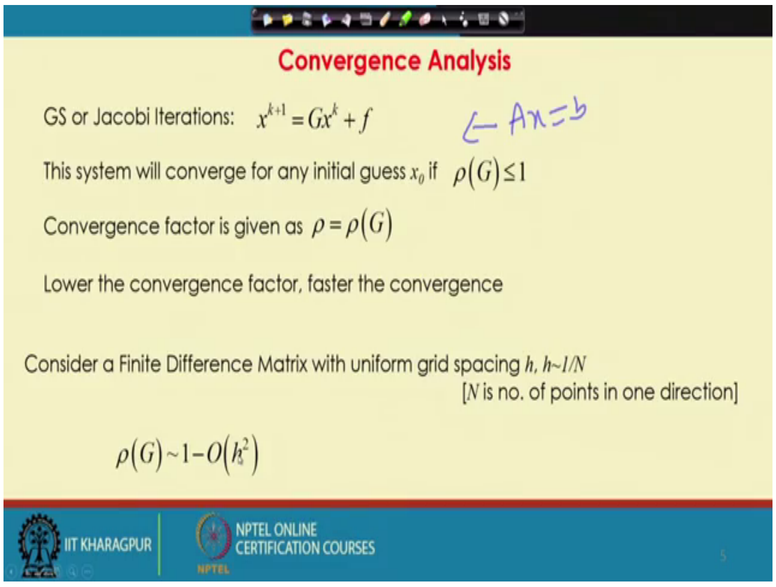
So, how that means, the first one is actually a coarse mesh which is very less number of points and the errors will be large. Second one is little finer mesh which has more number of points. Third one is a finest mesh we have considered.

Now, if we look into the number of iterations. The first one converged at 27 iterations where we have less number of points. Of course, it will be because the number of steps are also a function of the matrix size. So, there will be less number of steps. The second one converged in 127 iterations. The third one converged in 409 iterations a lot more iterations are required for the third one yeah.

And if I see how the error is reducing we will see that in the first one the errors started from 10 to the power minus 2 and then after few steps see 10 into 10 to the power minus 3 10 to the power minus 4. Then came to 10 to the power minus 7 and then converge to the value. In the second one it stayed a long at 10 to the power minus 7 the error is of the order of 10 to the power minus 7. Then it went to 10 to the power minus 8 and so on as the third one in 10 to the power minus 8 itself it stayed for a long time. No number of iterations then it required to converge the error from 10 to 0 in the first one.

So, as we are increasing the matrix size the number of iteration steps increase it almost scales with the matrix size with the solution vector. And as we go to a larger matrix the reduction is error is actually very slow when we go to the final value of the error. The error may be reducing very fast from 10 to 6, but if we have to reduce the error from 10 to the power 1.2 to 10 to into 10 to the power minus 8 to 1.1 to 10 to the power minus 8 that takes a lot of time. So, this is what we can look into the convergence pattern of the solvers.

(Refer Slide Time: 05:55)


Convergence Analysis

GS or Jacobi Iterations: $x^{k+1} = Gx^k + f$ $\leftarrow Ax=b$

This system will converge for any initial guess x_0 if $\rho(G) \leq 1$

Convergence factor is given as $\rho = \rho(G)$

Lower the convergence factor, faster the convergence

Consider a Finite Difference Matrix with uniform grid spacing h, $h \sim 1/N$ [N is no. of points in one direction]

$$\rho(G) \sim 1 - O(h^2)$$

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

And we can essentially look into the convergence issues. The convergence for large matrices when we have lot of grid points the convergence takes large number of steps. When you have increased this is convergence convergence. When we have taken not convergence I will corrected in the uploaded slide.

When we have taken a large matrix the number of steps for convergence is also very high, but we cannot avoid large matrices because if we need a very accurate solution we have to take a lot of internal points which will give us a large matrix. The smaller order errors reduced slowly, that is that when the error is of the order of 10 to the power minus 2 it is probably halved to from say 0.9 into 10 to the power minus 2 to 0.45 into 10 to the power minus 2 in say 10 states. But if I think the error will get half from 0.9 into 10 to the power minus 8 2.45 into 10 to the power minus 8 it will take probably 70 steps.

That means when the value of the error is small the convergence is slow. So, there is something like a fluctuating function which is damped very slowly because there is an asymptotic marching to the right solution the rate of damping of this fluctuation function becomes slow when you go close to the actual solution. But that is again for the large matrices for the small matrices the error reduction even at 10 to the power minus 8 level is much faster than a large matrix. There is another thing to notice. Also during each iteration the number of operations are more in for large matrices. This is how also very important thing because we are dealing with an n by n matrix each iteration probably takes n number of operations in subtraction addition come considering that few n or n square number of operations.

When we are having a large matrix so, n is very high in a number of our operations during one particular step is add then when the matrix is small. So, smaller matrix will require less number of steps to converge as well as in each steps there number of computational operations is small. So, in total a smaller matrix will require very less number of steps if we consider the number of iterations as well as computational steps in each step in iteration.

However I said we cannot avoid having large matrix things would have been very great for us if we would have been able to solve everything with a large matrix with a coarse grid because the number of computational steps would have been less. But we really cannot operate at small matrices or code strides because the accuracy will be compromised there. So, you have to operate with large matrix which is slow to converge as well as in each step there are a lot of operation. So, we will see how both of these things can be tackled.

Now, if I go back to because we are discussing here with Jacobi iteration. If I go back to Jacobi iteration perspective remember with this discussion will continue based on Jacobi which is the probably most basic iterative solve solver. But similar discussion is also valid for much developed iterative solvers like by conjugate gradient or GMRES, but let us even all the precondition solvers too.

But let us go to the very basic Jacobi solver. And at this lecture I will rather try to give you some key concepts on multi grid methods. As we discussed for pre conditioners the detailed of multi grid methods we will take lot more lectures and lot more deliberations many serious complicated mathematical analysis has to be done also which we are not discussing here we will just discuss over the concepts of multigrid method.

So, you focus our discussion only to Jacobi solvers where these concepts can be at least reviewed with certain result. So, in a Jacobi iteration solver the typical iteration step is given as k plus 1th value of x is equal to G x k plus f. Where this is basically again comes from solving x is equal to b, but G is formed using some splitting of a we have discussed it several times we will discussing a few of this method starting from basic iterative solvers to pre conditioner several times we have discussed this.

So, G is called the iteration matrix. The system will converge for any initial guess x 0 if spectral radius of G or the modulus of the largest eigenvalue of G is less than equal to 1 rather less than 1. The convergence factor is given as rho is equal to rho G or the spectral radius of G. And lower the convergence factor faster the convergence. Convergence rate is actually kind of having an inverse relation with convergence factor. So, if rho G is high convenience factor is high that means, number of iterative steps will be high if rho G is less or convergence factor is small number of iterative steps will be less.

So, we essentially for having a faster solution method rho G should be small, but rho G again depends on the actual matrix a. So, given the matrix A we cannot do anything with spectral radius of G except using preconditioning technique where instead of A x is equal to b we are solving m inverse x is equal to b where. For general case when x is equal to b itself has to be solved we really cannot do anything with G, but if I have two matrices one matrix solves faster than the other matrix. We can check the spectral radius of G for these two matrices and the matrix which will have smaller per spectral radius will converge faster which will have smaller convergence factor will converge faster.

So, we can look into different matrices created out of the same geometry and check their spectral radius of the iteration matrix. And we will see that which matrix will converge faster and why is it. So, lower the convergence factor faster is the convergence.

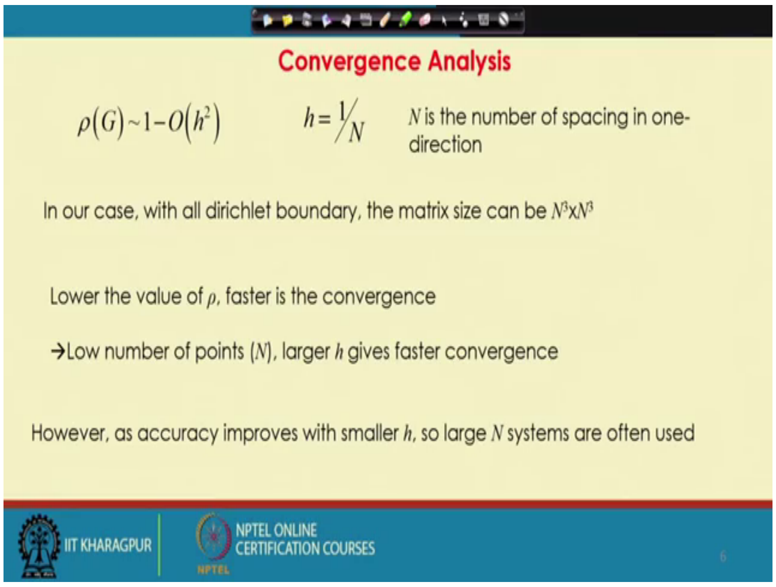
Now, we consider a finite difference matrix came out of using finite difference approximation on a partial differential equation with grid spacing h and h is equal to 1 by n or 1 by n minus 1 by the way you define n. So, it is a around one by n n is the number of points along one particular direction.

And then if I write down the 2 d Laplacian finite difference matrix and split it for a Jacobi iteration. We will see that the spectral radius of the g matrix is of the order of 1 minus order of h square. So, if from Laplace equation if I try to get a finite difference matrix the iteration matrix in Jacobi or even for Guass seidel we will have one minus order of h square spectral radius.

So, in case I have a coarse grid where h is large the spectral radius will be small. In case I have a large measures h is small the spectral radius will be large. And precisely that is the reason that large matrices will take lot more steps to converge because this way they have a large spectral radius. Because h is 1 by N if N is high this value is small. I if n is small number of points is small this value is sorry if N is high this value is also high because 1 minus 1 by n square and if n is small this value is small. So, it converges first.

So, this is the precise point that if we have large number of mesh points the iteration matrix will have a large spectral radius therefore, the convergence factor will be high convergence rate will be slow if we have a small coefficient matrix and small number the which will come from considering small number of mesh points in the geometry. The spectral radius of the iteration matrix will be small and therefore, convergence will be faster.

(Refer Slide Time: 14:49)



Convergence Analysis

$\rho(G) \sim 1 - O(h^2)$ $h = 1/N$ N is the number of spacing in one-direction

In our case, with all dirichlet boundary, the matrix size can be N^3xN^3

Lower the value of ρ, faster is the convergence

→Low number of points (N), larger h gives faster convergence

However, as accuracy improves with smaller h, so large N systems are often used

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, rho G is of the order of 1 minus h square h is equal to 1 by N number of spacing in one N is the number of spaces in one direction.

In case with all Dirichlet condition in our case with all Dirichlet condition, but we are considering if there are N internal point the matrix will be N cube by N cube. So, the number of the convergence will depend on the fact on this number N what are the total number of internal points in one direction. If I assume that all points have same number of all direction same number of points.

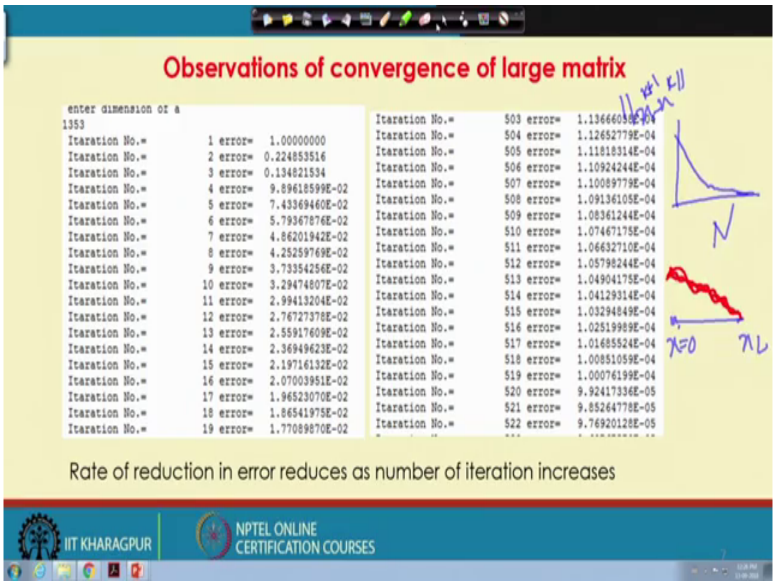
Lower the value of flow faster the convergence that is what we discussed that. If we can reduce this value we will get faster convergence. To reduce this value this should be small that means, this number should be high in order to have a high number here this N should be sorry this should be small. So, this should be rho G small. So, 1 minus h square should be a small number in order to have 1 minus h square small h should be high in order to have high h n should be small we can we can check it out.

So, low number of points N and larger grid spacing will give faster convergence. And that is exactly what you have seen that when you are taking less number of points 4 points in each direction the convergence is in some 27 points. And we are taking 12 points in each direction the convergence is obtained after 500 600 iterations.

However we are asserting this several times as the accuracy improves with smaller h or larger N large N systems are often to be used. So, the question becomes I know that I have to use a large grid framework. Which will give me a slower convergence because h is being small therefore, spectral radius of G matrix is small it will give me a slower convergence. So, what should we do to improve convergence of this? One idea is using preconditioning technique. Is there something else which is incorporated in the method itself how we are looking into the error and the convergence factor.

Something can be done that and one of the technique for doing this is a multigrid method. Where instead of looking into small values of n only we will club number of grid points and get another background mesh which has large which has a large h and solve do some of the iteration on the background mesh reduced residuals and then come back again back to the finer mesh what we are going to discuss later. So, the question is that how we can improve the convergence even with the smaller h.

(Refer Slide Time: 17:48)



Observations of convergence of large matrix

```
 enter dimension of a
1353
 Itaration No.=           1 error=   1.00000000
 Itaration No.=           2 error=  0.224853516
 Itaration No.=           3 error=  0.134821534
 Itaration No.=           4 error=   9.89618599E-02
 Itaration No.=           5 error=   7.43369460E-02
 Itaration No.=           6 error=   5.79367876E-02
 Itaration No.=           7 error=   4.86201942E-02
 Itaration No.=           8 error=   4.25259769E-02
 Itaration No.=           9 error=   3.73354256E-02
 Itaration No.=          10 error=   3.29474807E-02
 Itaration No.=          11 error=   2.99413204E-02
 Itaration No.=          12 error=   2.76727378E-02
 Itaration No.=          13 error=   2.55917609E-02
 Itaration No.=          14 error=   2.36949623E-02
 Itaration No.=          15 error=   2.19716132E-02
 Itaration No.=          16 error=   2.07003951E-02
 Itaration No.=          17 error=   1.96523070E-02
 Itaration No.=          18 error=   1.86541975E-02
 Itaration No.=          19 error=   1.77089870E-02
```

```
 Itaration No.=         503 error=   1.13666058E-04
 Itaration No.=         504 error=   1.12652779E-04
 Itaration No.=         505 error=   1.11818314E-04
 Itaration No.=         506 error=   1.10924244E-04
 Itaration No.=         507 error=   1.10089779E-04
 Itaration No.=         508 error=   1.09136105E-04
 Itaration No.=         509 error=   1.08361244E-04
 Itaration No.=         510 error=   1.07467175E-04
 Itaration No.=         511 error=   1.06632710E-04
 Itaration No.=         512 error=   1.05798244E-04
 Itaration No.=         513 error=   1.04904175E-04
 Itaration No.=         514 error=   1.04129314E-04
 Itaration No.=         515 error=   1.03294849E-04
 Itaration No.=         516 error=   1.02519989E-04
 Itaration No.=         517 error=   1.01685524E-04
 Itaration No.=         518 error=   1.00851059E-04
 Itaration No.=         519 error=   1.00076199E-04
 Itaration No.=         520 error=   9.92417336E-05
 Itaration No.=         521 error=   9.85264778E-05
 Itaration No.=         522 error=   9.76920128E-05
```

Rate of reduction in error reduces as number of iteration increases

So, will again look into the convergence pattern for a large matrix it is a matrix of 1353 by 1353 it is the different matrix where is again a large matrix. And we see that initial steps it is from 1 to 0.22 78 percent reduction in error is in one step then even when we are say around 10 to the power minus 2 9.89 to 10 to the power minus 2 to half of it which is maybe 4.25 to 10 to the power minus 2 it to 5 steps.

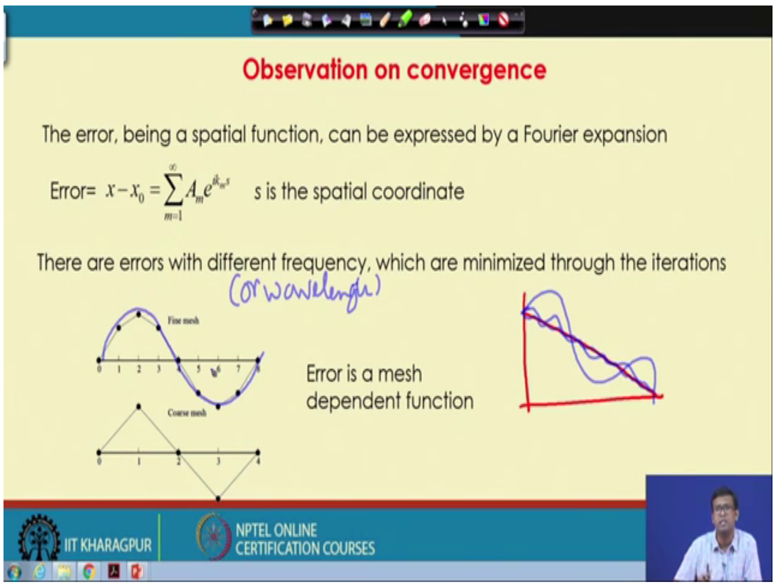
Right and the error is reducing by 4.404. If we go later of the iterations say 51. 1.13 into 10 to the power 1.09 into 10 to the power minus 4 2 1 into 10 to the power minus 4. So, error is reducing at the rate of 10 to the power minus 6 that took 1.09 to 1 that took around 10 steps.

So, the rate of reduction of error reduces as number of iteration increases; that means, that if I look into the number of iteration verses the x minus x k plus 1 minus x k l 2 norm. Finally, this should be 0. And this is initially high and then it goes down very slowly very slowly it goes down there is an asymptotic nature.

So, it is a curve with varying rate. And at each iteration what is happening the error is getting smoother there are some small fluctuations in the value if we go to the previous time step which is the difference from the converge solution. And this difference is coming to 0 and there and the solution is getting smoother in each iteration. However, this smoothing so, maybe I have this geometry say x is equal to 0 to x is equal to l. This will be maybe my final solution. And I will see that the solution as I have given boundary condition is initially something like this. Then it reduces something like this then it further reduces something like this and finally, converges.

So, how this fluctuation of the solution is damp to the final solution that is important. And we are seeing these fluctuations reduce very slowly with time when there are small amplitude fluctuations. And the small amplitude fluctuations because the error is small are also long wavelength fluctuations. So, we will see what is the how are this fluctuation wavelengths.

(Refer Slide Time: 20:56)



Observation on convergence

The error, being a spatial function, can be expressed by a Fourier expansion

Error= $x - x_0 = \sum_{m=1}^{\infty} A_m e^{ik_m s}$ s is the spatial coordinate

There are errors with different frequency, (or wavelength) which are minimized through the iterations

Error is a mesh dependent function

The error being a spatial function can be expressed as a Fourier expression any special function we can expressed as a Fourier expansion in space. Which is x minus x 0 is equal to e to the power A to the power m e to the power i k s s is a spatial coordinate it can be x y z x y z as if we think that x is the solution variable. So, s 1 s 2 s three are three different spatial coordinates.

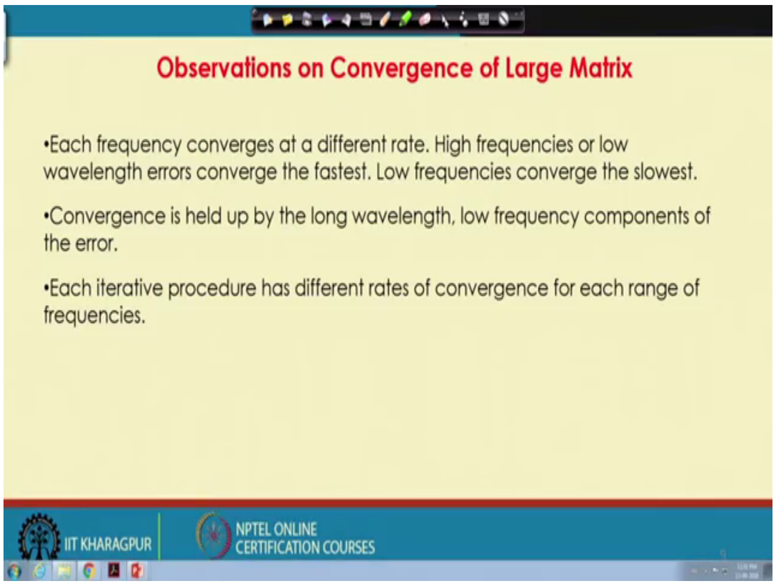
So, it is like a Fourier function the errors with different frequency are minimized through the iteration. So, in a sense we can say that say I have initially the this should be my right solution initially I start with the boundary condition I start with a solution like this. Later the solutions will be something like this. Later these solutions will be something like this and so on.

So, different there are solutions called with different frequencies. There are some high frequency solution there are some low frequency solutions which are getting minimized through these iterations. So, this k is for different wavelength different frequency or wavelength. This k is for different wavelength which is getting minimized to the iterations. If I have a coarse mesh this is how the solution will look like. And I will get the converge solution in each point. If I have a fine mesh then the solution is a much smoother one.

So, what I am doing with each iteration actually smoothing the solution. And this smoothness is done through some the it is the smoothness function has its own wavelength and frequency which are taking care. And it can be seen that the final solution which if I have a large number of mesh points I will get a very fine solution like this. These fits lot of long wavelength solutions because it is trying to fit a very nice continuous and differentiable curve of very high order continuous and differentiable curve over number of points.

So, the final smoothness this smoothness comes through a large wavelength solution. And this large wavelength error and this large wavelength errors are most difficult to minimize.

(Refer Slide Time: 23:38)



Observations on Convergence of Large Matrix

•Each frequency converges at a different rate. High frequencies or low wavelength errors converge the fastest. Low frequencies converge the slowest.

•Convergence is held up by the long wavelength, low frequency components of the error.

•Each iterative procedure has different rates of convergence for each range of frequencies.

Each frequency converges at different rate. Higher frequencies or lower wavelength errors converge the fastest. The solution between two grid points so, it is a very small wavelength at one time step this can converge. Errors low frequency errors with low frequency or large wavelength which is trying to smooth the solution over the entire domain converge in the slowest way.

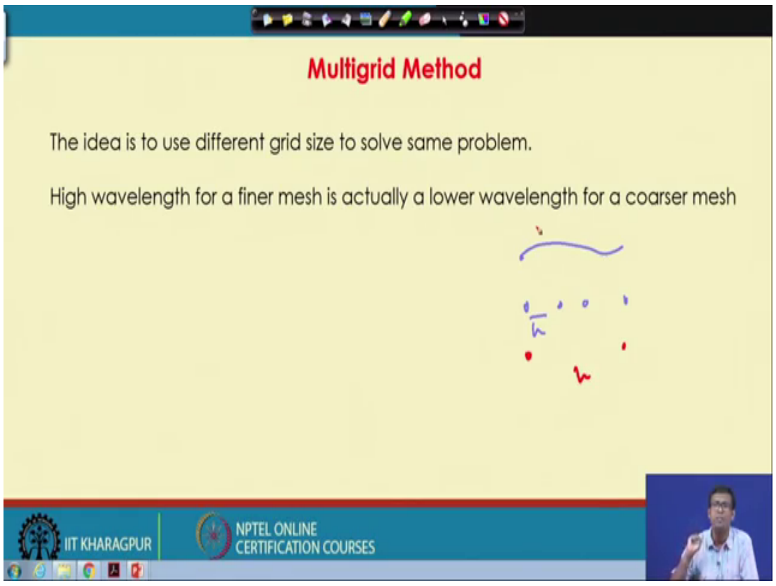
Convergence and therefore, convergence is held up by long wavelength low frequency components of the solution. The solution which is converging among many grid points as a wavelength may of the order of number of grid points that error is reducing in the slowest way.

So, the large wavelength low frequency component error is the bottleneck thing in getting the solution. And that is what creating very slow convergence pattern at that 10 to the power minus 8 level because it is trying to just smooth the solution in between a large number of grid points and trying to fit a very less amplitude, but large wavelength and also low frequency large wavelength a smoothness over the error or trying to correct a large wavelength low frequency error.

Each iterative procedure has different rates of convergence for each range of frequencies of. We will show something for Jacobi you will see something you will see from Gauss Seidel something you will see from conjugate gradient, but essentially the if we have large number of grid points the final smoothening will take a lot of time. Each point will come to close to some of the ballpark value, but the solution will show lot of roughness and wavy nature because it has not been smoothened before we iterate and converge for the long wavelength solutions. And this long wavelength solutions take lot of time for all type of all these solvers.

So, essentially we need to find out some methodology to converge the long wavelength solvers. And this one methodology is called a multigrid Method.

(Refer Slide Time: 25:49)



Multigrid Method

The idea is to use different grid size to solve same problem.

High wavelength for a finer mesh is actually a lower wavelength for a coarser mesh

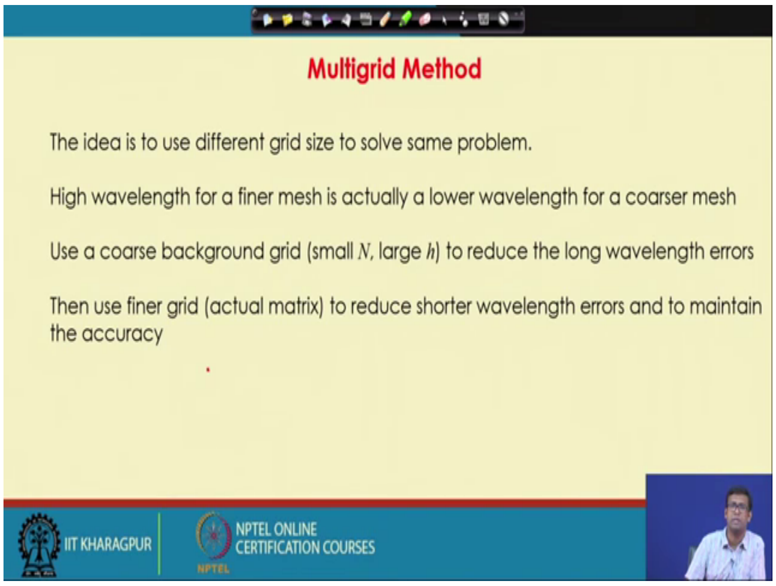
That is the idea is to use different grid size to solve same problem. Why? Because what is high wavelength say I have 3 4 points. And there is a smoothness function there is a smoothness function like this which is a high wavelength. So, this is h if we compare with h this is a very high wavelength. And it will take a lot of time to smooth it.

But if I use a different mesh size 1 and 2 this is and now this is my h the wavelength is actually of the order of h it is some this the wavelength was 3 h here the wavelength is h. So, a high wavelength solution which is a high wavelength solution for a finer mesh is actually a lower wavelength solution for a coarser mesh. And that is the trick to handle large matrices using Multigrid Method.

Because there is a geometric method you already know the geometry over which you are solving the equations generate a large grid spacing background mesh over it. Solve the equations on that. So, that faster you can very fast you can solve the large wavelength solutions.

And then go back to the final mesh and correct the smaller wavelength solutions errors which are faster to converge.

(Refer Slide Time: 27:13)


Multigrid Method

The idea is to use different grid size to solve same problem.

High wavelength for a finer mesh is actually a lower wavelength for a coarser mesh

Use a coarse background grid (small N, large h) to reduce the long wavelength errors

Then use finer grid (actual matrix) to reduce shorter wavelength errors and to maintain the accuracy

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

And that is the trick, use a coarse background grids to reduce the long wavelength solution first use a coarse background grid where the lord long wavelength small N and large h. Which is long wavelength for a finer mesh will be a short wavelength for coarse mesh. So, use a coarse background grid to solve the long wavelength errors.

And then use finer grid to reduce shorter wavelength errors and to maintain the accuracy. So, basically this goes into using 2 3 different multiple layers of grid different grids to solve same problem. You have a same partial differential equation you write a matrix equation with different grid size and. First solve it in coarser mesh and then come to finer mesh and get better accuracy. And then do some iteration between the mesh levels. So, that the final solution is smoother as well as converges to the right result at the finest grid level. And that is why we called it a Multigrid Method. So, we look into the implementation of Multigrid method in the next session.

Thank you.