Introduction to Computational Fluid Dynamics

Prof. M. Ramakrishna
Department of Aerospace Engineering
Indian Institute of Technology - Madras

Lecture - 34
Multigrid method

Okay, so what we were doing in the last class, we are looking at multigrid method right. We looking at series of mechanisms by which we could accelerate convergence of our numerical schemes. So these are in a sense modification to a fundamental scheme, in a fundamental scheme and now you are asking the question is there a way by which I can make this scheme faster? Meaning, from the point that I start the computation to the time that it takes wall clock time.

I looked at the clock and it takes so much time that it takes for me to get the solution okay, I want to reduce that, am I making sense. I am not really interested in other details, I mean there are situations where you can look at right, you can say that the whole is equal to the sum of the parts if I minimize if I optimize the individual parts, then the whole will also get shrink down. But you will see sometime it does not quite work out, when you try to shrink one the other expand and so on.

So we have to keep our eye on the fact that the metric, the measure that we are using to see whether we have made an improvement is we start and we look at the clock, it stops and we look at the clock and we say yes it is taken less time okay. Never lose sight of that, because it is easy to get caught up in all the numbers, CPU times, all of these kinds of stuff, and lose sight of the fact that the overall time is not either reducing, so you are putting an effort but you are not getting anything out of it fine.

In the last class we looked at a particular scheme right, the last part half of the class, we looked at particular scheme which was essentially multigrid methods. And the idea was that we are going to use multiple grids just to recollect multiple grids in order to accelerate convergence fine, and where did that come from? That came from the fact that we said that high frequencies, when we say a signal, so you know today, I will use the little signal processing kind of terminology.

When you say high frequency right, we have already noted from our previous demo that when we say high frequency, we mean high frequency with reference to a grid, high and low are comparative term, the terms that we used to compare 2 things. So you need 2 things and you have to have other right, so the other is grid the underlying grid, what is the structure of your underlying grid?

So on an equally spaced grid a high frequency and low frequency with reference to the grid are defined, and there was schemes that we saw where a high frequencies decayed faster than low frequencies. See now I am careful, there were schemes that we have seen where high frequencies decay faster than low frequency, that means that there are schemes that which that may not necessarily be true right, in which case we need to do something for that, today I will tell you something about that okay.

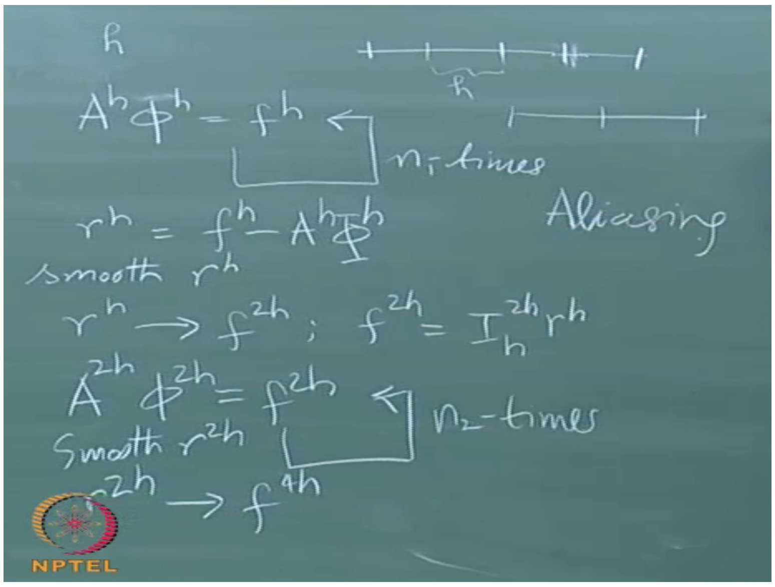
Then so what we basically said was well if the high frequencies decay faster than low frequencies, and then we have the low frequency error with which we are trying to content essentially grid that we have at hand is struggling to decrease the error. Why not transfer the problem to a coarser grid, so that this low frequency error on the fine grid turns out appears to be a high frequency error on the coarse grid okay.

And therefore, it will be decay faster on that coarse that is the basic idea of multigrid okay. Now today, what we will try to do is we will try to write out the whole schemes, so first I will recollect where we left it, then I tried to write out a whole scheme, and then we will see what are the steps that we need to take in order to how should I say make sure that everything is fine, a little patchwork that we need to do right.

There are schemes which may not be inherently you know high frequency is decaying faster than the low frequency, we have to do something for that that is one thing. You still want to transfer the problem from a fine grid to a coarse grid, but we may have to do something for that okay, that is one thing that we have to do. And the other thing is all I have said so far is, it is going to run faster, it is going to decay faster will it?

What is the effort involved in doing this? How much work are we actually doing right? So before you write the program and look at the wall clock time and so on. We are going to ask ourselves the question how much work is actually going to take in order to do this okay, work in the sense that computational work. There is also implementation related work and I will sort of say something about that as we go, is that fine okay.

(Refer Slide Time: 04:36)



h

$A^h \Phi^h = f^h$ n_1-times

$r^h = f^h - A^h \Phi^h$

Aliasing

Smooth r^h

$r^h \rightarrow f^{2h}$; $f^{2h} = I_h^{2h} r^h$

$A^{2h} \Phi^{2h} = f^{2h}$ n_2-times

Smooth r^{2h}

$r^{2h} \rightarrow f^{4h}$

So first just recollect where we were, we started off remember that the grid size we talked about was h, so this is the typical grid size what that means is that if I have that is supposed to be equally spaced, it is does not quite look equally spaced. If I have equally spaced grids right, the typical grid size is h, and if I have a linear problem and I will tell you what to do when you have the non-linear or quasi-linear problem.

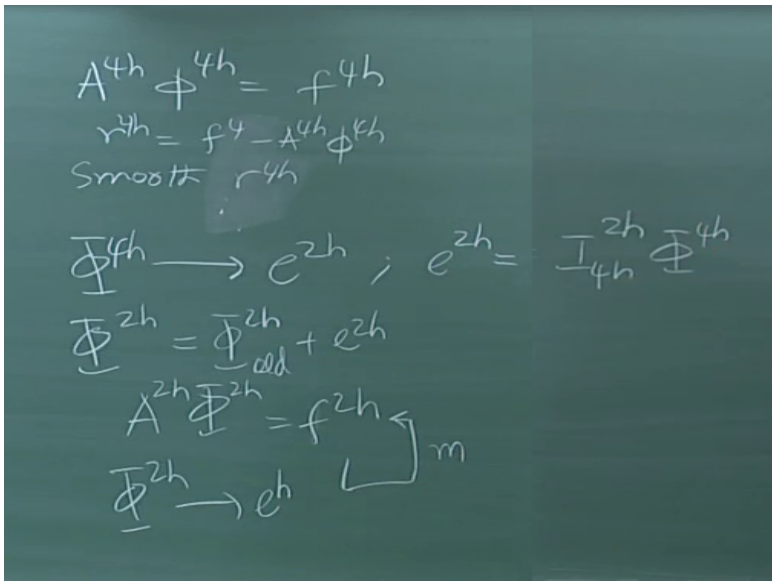
Linear problem I chose Laplace equation as an example, but it could be any system of equations that we talked about Ah phi h=fh, we are trying to solve this. And what I would indicated was that we iterate this a few times or if it is the time marching scheme, we march in time right a few time steps. So we do this what was the simple that are used for the number of iterations? was it alpha? n times, I am going to regret that, but in n times, we going to iterated n times okay.

So then what we do from here remember the algorithm, we get the residue, so we get rh which=fh-Ah phi h where this phi is the intermediate solution capital phi is the intermediate solution, we get the rh, then what? We transfer rh to f2h right, if you want to write it as a equal an equation in books that you may look at they may write something like f2h this is the typical notation right, so you can actually write matrix form of this.

I will recollect for you what each of these things right we will go through that, so you transfer from rh to fh, on the grid which whose size is 2h, you then solve, you then either solve or you iterate a few times or take time steps are appropriately okay. And do this if this was n1 times this could be n2 times, it could be the same it could be different, if you have some magical reason for making them different it is up to you okay.

Then we saw look right if you start with a very fine grid it is possible that the 2h grid is also reasonably fine. So you can then compute after iterating a few times and then you say it is not converging fast enough I am not happy with this, you can actually compute the residue there, which you transfer to f4h, am I making sense okay.

(Refer Slide Time: 08:16)



$$A^{4h}\phi^{4h} = f^{4h}$$
$$r^{4h} = f^{4} - A^{4h}\phi^{4h}$$
Smooth r^{4h}
$$\Phi^{4h} \longrightarrow e^{2h} \; ; \; e^{2h} = I_{4h}^{2h}\,\Phi^{4h}$$
$$\Phi^{2h} = \Phi^{2h}_{old} + e^{2h}$$
$$A^{2h}\Phi^{2h} = f^{2h} \quad m$$
$$\Phi^{2h} \longrightarrow e^{h}$$

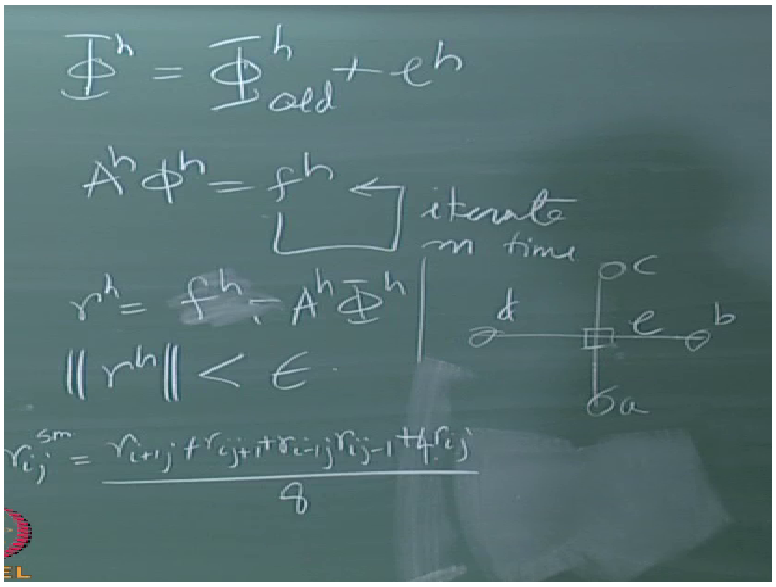
And then consequently we will solve and repeat this process, at this point actually you know the phi is not actually phi the correction, if you think back to algorithm phi was actually a correction. So how does the reverse mechanism work? You take the phi for h that you have got from your iterations let us say iterate, you could go on but I will tell you what happens for phi 4h, and you transfer that to e2h that is the correction at okay.

And the new phi 2h= the old phi 2h okay, now you have lots of options, see this is the thing about developing the algorithm you can see there are lots of options. If you want and I would do this, you can also sort of iterate a few times there, m times if you want okay, so you iterate at that level and then you get phi at that level right. The f does not change, if you want right, then you can transfer okay.

This transfer by the way you could write if you wanted to write it as an equality, just so that if you look at books, I want you to make sure that you have the notation, you would write this e2h=along the same lines I4h 2h phi 4h right, and you can actually write it as a, these are vectors you can actually hook up and find out what this matrix is depending on what kind of interpolation or whatever it is that you are doing right.

We will look at I will give you the appropriate terminology, we will go over this blackboard one more time right, is that fine everyone.

(Refer Slide Time: 10:44)



$\Phi^h = \Phi^h_{old} + e^h$
$A^h \phi^h = f^h$
iterate m time
$r^h = f^h - A^h \Phi^h$
$\|r^h\| < \epsilon$
$r^{sm}_{i,j} = \frac{r_{i+1,j} + r_{i,j+1} + r_{i-1,j} + r_{i,j-1} + 4r_{i,j}}{8}$
a
b
c
d
e

And then you would say phi h=, right that it is the correction at the Ah level, maybe m time. And at the finest grid you always ask the question, you compute rh, see I am repeating now, I am repeating the first line now we are back to the first line, but you can ask the question have we converged right, you always iterate on the finest grid a few times, and ask the question how we converged, does that make sense.

You want I mean the reason why we are making this effort is I want a grid on that I want a solution of that fine grid, because that is what, that is the resolution that I want, that is what I am looking for right. I have decided, I have looked at the problem I have looked at I have an idea I have a senses to what the flow features are like, and I have said okay I need a grid that is 1 millimeter in size, let us take actual physical dimension.

You need a grid that is 1 millimeter in size, I want the solution on the grid that is 1 millimeter in size. So having decided that I pick h which is 1 millimeter, and I go through this process right, so I want to make sure that when I finally decide yes I have the solution, I am making the decision on the 1 millimeter grid not at the other grid okay, the question always arises at this point okay. I do not care about the other residue that we have calculated, is that fine, is that okay, are there any questions.

Now we will go over this little detail we will check out fine, we will check out the little of the few things. How do we decide? What is the point here? How many times do we iterate? Is there a mechanism by which we can decide how many times we iterate, is that just an arbitrary parameter should we have the method by which we should decide how often to iterate, think back to our demo? See I said there is the highest frequency that we can represent on a given grid okay.

So if I am going to go from a grid okay, maybe we should say something about grid first okay, so if I am going to go from this grid were really coarse grid it has only one unknown right. I am going to go with 3 interior points to 1 interior point, then I have to make sure that I have iterated enough on this that there is no component of what I am going to transfer the residue that I am going to transfer, which has a frequency higher than what this can represent okay.

If you go back think back to the demo, otherwise what will happen if you will get what the signal our signal processing friends in electrical engineering call aliasing, basically that high frequency error will fall back into the low frequency, am I making sense. And we will start getting a residue here which is spurious, which is not the right residue okay, so if you talk to your friends or who have done signal processing in electrical engineering and so on.

They would call this process either decimation or down sampling, so if you are going to decimation or down sampling the first thing that you have to do is you have to remove the frequency content that you cannot represent on the coarse grid, if you cannot represent it, it will only show up as an error, it will only contaminate the residue that you have on the coarse okay. So one possibility is iterate enough number of times.

So that all that high frequencies right now that is right now that is what we have, all that high frequency error is eliminated one possibility okay. We clearly have to come up with some other way to do this, because remember I said that maybe schemes that do not decay high frequencies, there may be schemes that do not decay high frequencies fast enough right there are all sorts of issues. So but we have to decide, so you iterate enough number of times.

You have to make sure that you have done it enough number of times, that you have only the frequency content in the residue that you are going to represent on the coarse grid, is that fine. Otherwise, anything that you transfer from the fine grid to coarse grid will show up in the wrong place okay right okay, it shows up in the wrong place in our frequency domain, that is if we look at it terms of frequency okay right.

So that is bad that is not good that is not what we want fine. The second thing is so this is this we have to see, is there something else that we can do here right, what else do we have? You transfer from here maybe we will stick with this, maybe I will finish with this first and then let me see where can I write that, I have this, I am already at rh I write it here. So just say you have a schemes, so before I get back to that let us just say right, we will take an aside and look at this let us just say you have a scheme that is not dissipating fast enough right.

Or you decide that you are going to iterate a fixed number of times or 10 times or fixed number of times 5 time steps, 10 time steps right, simply because you remember something I told you earlier when we are doing SOR. I basically said do not wait for it to converge to certain extent, because you do not know how long it is going to take, you are committing unknown amount of CPU time right, unknown number of iterations, always bound the amount of computation that you are willing to do.

So you say 5 times I am willing to iterate 5 times, but that may not eliminate all the high frequency, but we know one operation that eliminates high frequency, we have seen it, when we did our looking at high frequency, when we first encountered high frequencies decays faster than low frequencies, that is what is either it takes heat operator or the Laplace operator. We have seen it that is guaranteed, that really kills right.

So if you have in 2 dimensions, if you have 4 grid points you take the middle fellow and take the average of the 4 right, we have seen that there is a solution to Laplace equation right that sort of decays, and it decays quite fast. I do not want I mean I actually have a value here right, so I do not want to just throw away what I have at that point, so instead what I will do is I will take the average of these 2 average of these 4 that gives me a central value.

I already have a center value, I will take the average of those 2, what do I mean by that? I have values a, b, c, d and at the center I have e okay, this is what we would call laplacian smoothing what I would do is I would say a+ right that is the Laplace operator, we recognize that for an even grid okay, that is like taking omega=1/2 in SOR right, omega=1/2 omega equals what 1.5, 0.5, 0.5 okay, I just throw that just doing around that and see okay fine.

So what do we have? So this of course turns out the way this would work out, if you want to so you can see the Laplace operator here, the way this would work out if you are going to actually apply it to r, so what are we going to apply it to? We are going to apply it to r, I have iterated a few times I have decided say 5 times, I have iterated this equation 5 times, I have had enough doing iterations at that level, I compute the residual I have no clue what is the spectrum.

I have no clue what frequency is RHS, all I want to do is I want to eliminate those high frequencies okay. So what I am going to do is I know this will do it, I know Laplace operator will do it for me, so I am going I am proposing to smooth r, before I do the transfer, am I making sense. I am proposing to smooth r, this is called laplacian smoothing, so you would write r at any grid point as is that okay.

If you work out that fraction that I had earlier, you will see it will be 4 times, so the r smoothed would be basically that okay, so you can do one sweep, you do not feel comfortable do 2 sweeps, I guarantee those high frequencies will be gone. Now transfer the smoothed r okay, so now we are going to add a new, we will add a new step up we have a new improved step that we have. So which is why I strategically left myself a gab here right okay.

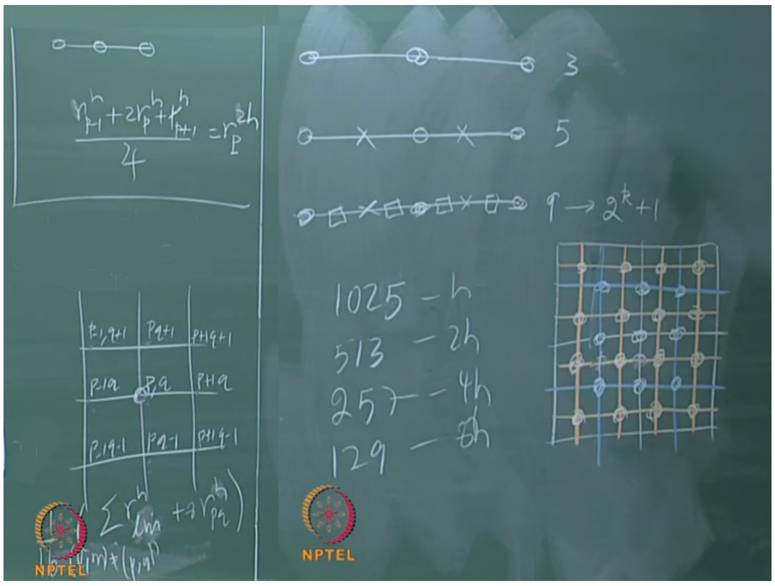
So smooth rh fine, and then do a transfer, is that fine. So at each level you can do that at each level you can smooth rh and then do a transfer. So this f2h now is not going to have any content that it cannot right that grid cannot present, and then as a consequence you can iterate this, and again I would iterate it a few times smooth, again so we will have smooth, and do the transfer okay fine.

So at every point when you are doing a transfer you smooth, you compute the residue find r, that is how it goes that is how it works okay, is that fine everyone okay. So this is as far as making sure that there is no contamination when going from the fine grid to coarse grid okay, so we need to do one more thing we will look at is that fine okay everyone, you do smoothing. Of course you can do a similar kind of smoothing on the way back that is we will get to that.

The next thing that we do is we want to look at this what is this operation that we have okay, so how are we going to do this average? I mean how are we going to do this transfer rh to f2h? How do we do the actual transfer? I think I can remove this I need it I will come back right okay. But in order to see the transfer we are going to actually look at what kind of grids? What are these multiple grids that we are going to generate?

We look at how it will there is a underlying structure we look at that okay, and then look at what is the transfer? What is this business of transfer? what are the ways by which you can transfer? So we will start at the coarse level, because that seems to be the, and see if we can get a pattern, that seems to be easiest.

(Refer Slide Time: 23:14)



So the coarse level is 1 grid point interior grid point 3 grid points on the whole. The next level is 5 grid points, is that right, what is the next final level 9, where do you think this is going? So it is going to be basically going as 2 power n+1, I know you know why I did not want any iterations 2 power I have also used m k+1 okay, is that fine right, 2 power k+1, so k=1, k=2, k=3 and so on fine okay.

So you could for instance, if you wanted a fine grid, so I will not start with 1001x1001 right, because that is thinking decimal, I would start with 1025 grids, if I have a 2 dimensional problem 1025x1025, whether they start with 1025 grids. And the next level of grids would be 513, you understand, then 257. now you understand why you need to go multiple levels this is 257 this is still pretty fine, it is just that we want the answer on 1-meter length.

We want the answer on something that says the order of 1 millimetre okay 1025 is close enough okay, so then you would go to 129, and you can go all the way down to 3 right, if you want. So this is the number of levels that you can transfer, so this is h, 2h, 4h, 8h fine, we have tried I mean I think we have gone up to like 7 levels or something of the sort right okay. So you look at this process. So now we know we can pick up grid like this, this is about 1D.

What about 2D we take the Cartesian product of it obviously right, so that could be 3x3 grid, am I making sense, to that you can add maybe I will use the different coloured chalk, that is a 5x5 grid am I making sense and we can go on, you can make it 7x7 grid and so on right. You can make it 7x7 grid, maybe I will choose that is that okay 9x9 grid sorry thank you 9x9 grid doing suddenly 2N-1 instead of 2n+1 9x9 grid that is one way to do it right, so I am considering it up.

The other thing is if you add a 9x9 grid, you can throw things away in what we call the chequerboard pattern or the chessboard pattern okay, so you do not have to you can for instance you can start you know you can retain this grid point, retain that grid point, retain that grid point, retain that grid point okay. Then you can retain time this one, retain this one, retain this one, am I making sense right, it is like the black and white squares of a chess board, is that okay.

So there are obviously once you go to 3 dimensions, 2 dimensions they are in different ways, so this is called coarsening, there are different ways of doing of coarsening okay, is that fine. So what we want is so if we want to go from coarse grid, if you want to go from a fine grid to coarse grid in the multigrid right, it is signal processing they would say you go through the down sampling rotor the multigrid terminology that is called a restriction, you restrict right you restrict.

So what you would basically do is you would now, this is what I was talking about how do I find that? I will write that here, so what are the ways by which we can do it right, so one thing is just pure simple injection as they call it, what you do is you just throw away the points that you do not want right. So if you have okay let us look at this, so you have 5 grids and you want to go to 3 grid, you just throw this away that is one way to do it right.

And another way to do it of course better way to do it, another way to do it a little more expensive, we have looked at in the last class, you take the average okay and a familiar average, average would be if this is p, p-1, p+1, the average would be so we are going from this is residue remember. So r p-1+2r p+ r p+1/4 would give you the value at that point, looks very suspiciously like laplacian smoothing right okay.

These are all at h this will be the rp 2h, the value here would be twice this+ each of those/4 okay. What if you are doing in 2D? How would you do it in 2D? I am not going to do in 3D, because its mean you can work it out but it is a pain to draw the picture, but I will do it in 2D okay. So this would be a full the best transfer and the most expensive transfer most operations, you want to get that, you want to transfer the value of that point.

So you have 1, 2, 3, 4, 5, 6, 7, 8, 8 of those so what are we going to do now? 8 of those+8 of these right /16, am I making sense 1, 2, 3, 4, 5, 6, 7, 8, add up all of those+8 times the central fellow/16, am I making sense. So the logic is now you can see the where it is coming from, so that would be of course I mean you could in theory add more points to it right, you could increase the support so to speak you could add more points to it.

But this is in itself there is certain expense, but you are taking all the points that you are planning to throw away, you are taking data from all the points that you are planning to throw away that is the idea here right. If you are going to take this point its neighbours are not going to be used right, so in this case if you are going to take this point, if you are unless you are doing the chequerboard pattern, unless you are doing this pattern.

If you are just doing the standard right the standard coarsening, this is called the standard coarsening, so if you are going to do only the standard coarsening, then you are going to throw away all the neighbouring point right. So this would be determined really by all the neighbouring points, am I making sense, so you would say summation, what should I say? So in l the surroundings set l0=p, simultaneously l, p !=, I should write that a little more careful l, m, so except for that one +8 times okay, is that fine.

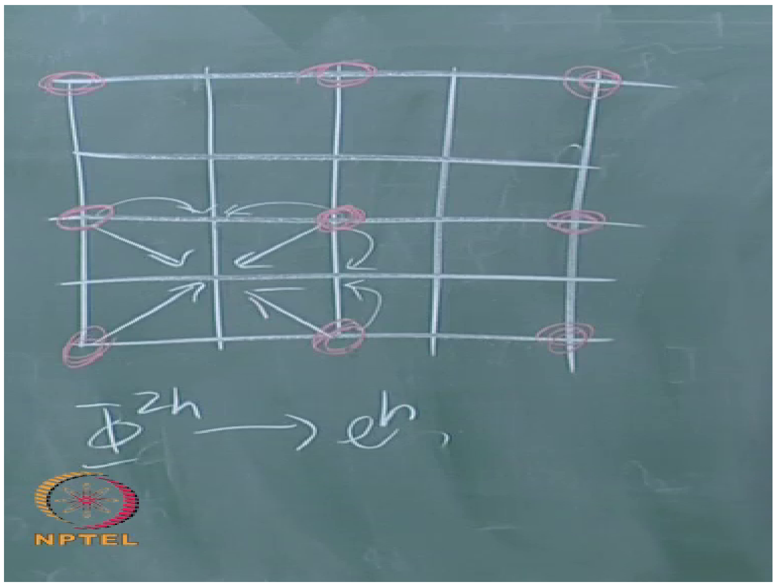
So this how we do the transfer, this is how we are going to do the transfer, so this will allow you if you work it out for a grid I would suggest that you start off with a 1-dimensional grid, you were never really good this is only for the sake of our explaining the algorithm process. You are not going to really you know write this matrix as such, but you can just for the fun of it to see what the structure looks like, try to write it out and see what it looks like okay.

So you do the transfer this way, am I making sense, is that fine right, you do the transfer there are different ways by which you can do it, you can do the averaging depending on 1D, 2D or 3D right, and you can transfer it to the coarser grid, maybe we will come back to this I leave that portion. What else do we have? So we have done the we are now going through this whole process, how do we get back? How do we do this transfer?

How does the reverse transfer take place? So I want to go from coarse grid to a fine grid, so I have to interpolate. So now see there will be cases, so you have to look at the various cases one case is the point is actually in this set okay, so typically the notation that you could write is this is the grid 2h and that is grid h and 2h, but it is okay. What you are basically, so if this belongs to the finer grid, just transfer the data directly.

If it does not belong the coarse from the coarse grid, there is no candidate take the average 1D it is obvious. You go to 2D in 2 dimensions if this grid belongs to the fine grid, transfer it directly, if it does not belong, but it happens to be so in this case this is a funny mesh, let me draw another mesh where can I draw it? I will draw it here, I need to erase something, I will erase this I no longer need this, there we go okay.

(Refer Slide Time: 35:46)



So what you have done is this is the coarse grid, that is a coarse grid okay, I am doing standard coarsening that is the coarse grid, so if it belongs to the same grid line it is on the same grid line, but does not belong to that mesh take the average same thing okay. So you can take in order to find this, you can take the average of these 2, if it does not belong to this same grid line, so this works even here right even in the vertical direction it works.

It does not belong to the same way that is like this, a little more complicated, then you take the average of the 4 okay, is that fine right. So on the side take the average this way, and that side take the average that way, and in the center one take the average of the 4, have I missed anything? I think that covers all possibilities, is that okay right. So that is how you transfer from a coarse grid back to fine grid.

So the correction that is how the correction will be transferred from coarse grid back to a fine grid, is that okay. So if you say that I want phi 2h to go eh this is what you would use fine. So now we come to the critical question, so we have got an algorithm, what we are able to do is we are we have the solution presumably we iterate a few times on the fine grid right, and we can transfer down residues to coarser and coarser grids, transfer back corrections to finer and finer grids.

And we have these nuances residual smoothing and so on, do all of those kinds of things right all that need stuff, we have an algorithm. What does it cost us? Okay, what is the cost? On what grid do we want the solution? So back here on what grid we want the solution, we will pick a big problem 1025 maybe I would say 1025x1025 or 1025 okay, this is the grid on which we want the solution fine.

So if were to take either whatever equation you are solving Laplace equation for example is a good place, and you are iterating, it would take a certain number of iterations to get a solution on the grid of size 1025 the solution that is to your satisfaction right 5000 iterations, 5000 iterations 1025 grid points, am I making sense okay. So if you are too in between transfer the problem to a 513 grid point grid mesh that has 513 points, then 2 iterations here is the same as one iteration there right.

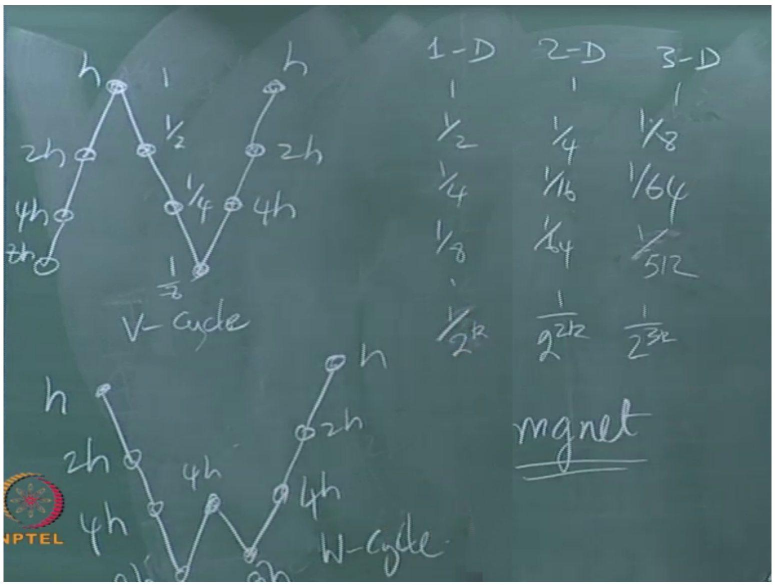
Remember, when we are talking about one of the things that I said when you find out how fast is my computer program running, what did I ask you to do? I told you to find CPU time per grid point per iteration of a time step okay, that gives you a measure of your program what it is up to. So if you right why should I averaging be more expensive just because I am on a coarse grid that does not make sense right.

So I do not expect unless you do something really bad in your programming, I do not expect that 513x513 CPU time per grid point per iteration will be more than this, so the work for the grid point is the same okay, the work for grid point is the same. So now I want to get a measure right, so I will say wait a minute if you did not do multigrid, you would be doing a so much work on 1025 grids.

So I will call one sweep or one-time step or whatever scheme that you receive through this 1025 your finest grid, I will call that one work unit right. We will start off by defining what do I mean by this work? So we will call that 1 work unit, am I making sense sweeping through this your finest grid making 1 iteration or 1-time step, I call the 1 work unit, so if I have 1 work unit here, 2 iterations here will correspond to 1 work unit.

Or basically 1 iteration here would be 1/2 work unit, 1 iteration here would be 1/4 work unit right and so on. 1/8 work unit, am I making sense. So it is better for me to doing 8 iterations here in comparison to 1 iteration there okay, so I will show this process of going from coarse grid to fine grid in a graphical fashion, and this is a standard graphical patient that people use in order to demonstrate multigrid method.

(Refer Slide Time: 41:45)



And because the picture evokes an image meet, I mean there is a letter it gives it a name, so what you do is this is at the level h, I will go from h to 2h, 2h to 4h, 4h to 8h we will go 4 levels, back to 4h, back to 2h, back to h okay. So this represents 1 cycle, because of the way it looks it is called the V-cycle, and of course you know it is called the V-cycle because there is going to be another cycle right a different cycle.

What is the different cycle? I basically look at this and say wait a minute you mean this takes 1 work unit, and this takes only 1/8 work unit I want to spend a lot of time here, after all I am always going ahead in there right, I do not want to know, so I want to spend more time here. So what you do is you go h, 2h, 4h, 8h, 4h, now you have a choice and go back to 8h, 4h, yeah I like that right fine, this is called a W-cycle.

And obviously W-cycles come in lots of favours right, the minute you see the minute you create that innovation lots of things that you could go 8h, 4h, 2h come down or you could go 8h, 4h, 2h, 4h, 2h, 4h, 8h, you know now you can see I want to spend a lot of time down here right. Because it does not cost me anything right, as long as I come back and iterate few times at h, because then I say residue is gone to 0 at h I got the solution, am I making sense right okay, are there any questions.

The work units in 1 dimension go as 1, 1/2, 1/4, 1/8. How about in 2D? It goes down faster right, so in 2D, so in 1 dimension this is work unit per sweep per time step per iteration or whatever, 1/2 power k kind of a thing okay. 2D 64 that is grid, now you know 3D is where you want to do though I have not shown anything, 3D so this savings are enormous 1D you are not you know you try a problem just to do learn how to do this, but you have to do at least 2D right, in order to say that I have got something really great.

So this is going to be 1, and that is going to be 8 right, am I making sense, 32, what is that going to be? 128, 64 2 power this is 2 to power, 2 power 4, 2 power 6 64, so this is going to give me 2 power 8 256, the powers of 2 are worth, now 3k 512 but you have to pick the right power of 2, there are not only worth knowing but you have to make sure you pick the right of power of 2, no use knowing the power if you pick the wrong power of 2 okay fine.

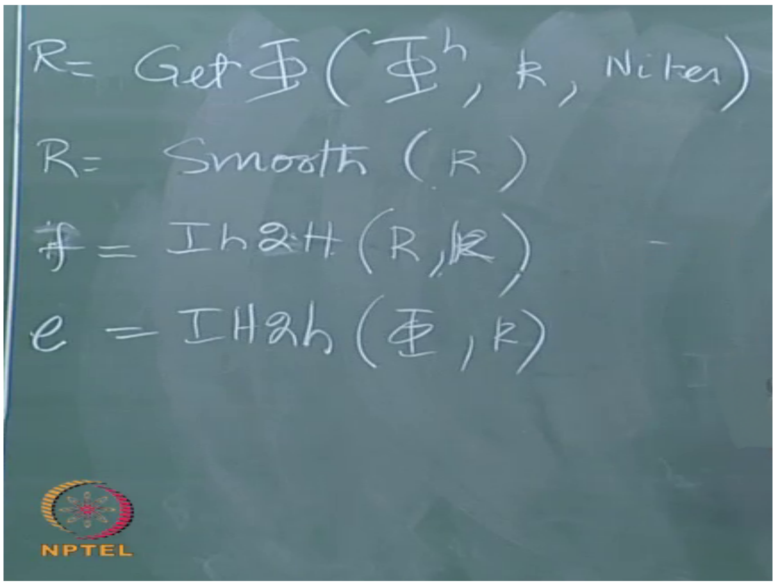
I mean just look at this you just to go down right, you just go down a few levels this is nothing I mean so you take 10 time steps on this, so the 512 time steps that is like taking 1 times step on the finest grid or 1 iteration of the finest grid, why would you not want to hang around? Right, so if you are doing in 3D if you are doing 1025x1025x1025 it is a lot of effort 1 billion more than a billion right. But you come down here, and it is not that bad, it is really not that bad okay.

As long as you make sure that you are always looking for your convergence at the do not forget that okay, there is one last thing you know there is always a little wrinkled that we could throw it, see why started a fine grid right, this is the remember when we talked about initialization we said one way to do it is to start to the coarse grid and transfer yes we can do it here right. So I started, so you basically say why should I start there start here, right start at 8h transfer it to 4h.

That is the way to do it you understand what I am saying, you already have it, so the critical thing is what is it so you this multigrid methods okay lot of people just do not get into multigrid, you do not see multigrid methods used just as often as they should okay, this is what this is my observation. The lot of research being done, lot of enthusiasts if you want to go on the net and you can search for mgnet, enormous amount of resources, huge amount of resources right.

The critical thing that you have to do to make sure that all of this works is something that you have to do anyway, so you have to make sure first that you implement right, if you do this you are set, what are the things that we have here that are different from your standard solution? The standard solution is given a grid take time steps right or do iterations. So first of all your program that you have has to be somehow changed right.

(Refer Slide Time: 48:54)



$R = \text{Get}\Phi(\Phi^h, k, \text{Niter})$

$R = \text{Smooth}(R)$

$f = \text{Ih2H}(R, k)$

$e = \text{IH2h}(\Phi, k)$

So if you are talking about Laplace equation, if you have phi and you have Laplace equation solver right that you are using right now, you just you just have a solver right that says so give me the boundary conditions whatever in some fashion and solves Laplace equation right. What you need to do is you need to convert that into a function that can be called, so you convert it to a function get phi which takes the grid.

It takes the grid or the grid level or whatever it is, which takes the grid level am I making sense right, and possibly an initial guess and all of that kinds of stuff I do not put everything else, takes a grid right and it should return to you, it should return for you the residue, so do 1 iteration come back and return a residue, or do n iterations and come back and return the residue, am I making sense.

So your program that you have right now, that is all the Laplace equation or whatever it is, it has to be changed it has to become a function, because you are now planning to put something piggyback on top of it to make it run faster right. So you are going to say do this on a 1025x1025, and it should come back 1 iteration and give you a residue, are you should be able to say take 10 iterations this is a little better, take 10 iterations with this initial guess okay.

So maybe I will make it a little more with this initial guess at the level k take 10 iterations, I am just making it out take 10 iterations okay. So now you have got this you can just pause it right and take 10 iterations and what it returns as the residue, you can then do you need to have, what is the next thing that you need to have? you have to be able to do laplacian smoothing tool, you have to smooth it right.

So you have to have some smooth which takes the residue right, and it returns the smooth residue am I making sense right, as to whether you are actually implement it in this fashion, whether you actually have these functions you will have to figure it out, what I am saying is these are operations they are atomic operations, you implement each one of these test them make sure that they work. If you got this, then the rest of it is relatively easy.

Then what does the other critical thing that you need, transferring you understand what I am saying. So you have to have something I h to or 2 if you want which I do not know what to call it you know capital H, let us make let us otherwise you will have too many h okay right, fine grid coarse grid, or it takes R and gives back f, whichever way you want to do it, so this gives me R, this gives me R, this also gives me this gives me f okay.

In fact, if this is a generic function it will take 2 arguments, they will take the k, because you need to know not kT, you take the k right, you need to know what level you are, the actual implement as I said the actual implementation will vary, so you will need to know what level you are, you get the f and then what else you need you need the other way around fine to coarse, you need corrections=, which will take you from fine to coarse to back to fine am I making sense. If you do this this is the critical part, if you do this you are set right.

But the idea is to implement this make sure that for any grid it works, do not worry about whether you are solving Euler equations, Navier Stokes equations, Laplace equation, it does not matter. The minute you start worrying about what it done, then that gets to be a problem okay, is that fine okay right. So I guess that is what we have for multigrid methods, you can get a tremendous amount of speed up we will see where it goes okay, let us try it out and see where it goes, thank you.