MIT OpenCourseWare
http://ocw.mit.edu

18.085 Computational Science and Engineering I, Fall 2008
Transcript – Lecture 25

PROFESSOR STRANG: Well, OK. So I thought I would, as last time, before Quiz 1 and now before Quiz 2, I thought I would just tell you what the topics of the four problems are. Remember that trusses are still to be reviewed, and of course the fun was to find mechanisms, to find solutions, to Au=0 in the case when the truss was unstable. So you can guess that I'll probably pick an unstable truss. And ask for mechanisms. Then maybe the key problem for this third of the course is finite elements. That idea, and finding elements we've really only done in one dimension. So and particularly linear hat function elements. So you see what's not included there is like cubics. I mean, those are really great finite elements, but maybe not so great for an exam. So 1-D is enough; linear elements are enough to do on an exam. And then questions three and four, so this one will be weighted more heavily. Then questions three and four are the topics that we've been doing the last week, two weeks. And I realize we haven't had full time to digest them, to work a whole lot of exercises. So those will be, right now they're too simple, my questions on those topics. Three and four, I'll try to make them a little harder for you. But I might not succeed. So, anyway. There we go. So I just felt we can't talk about a course in applied math without dealing with these topics in vector calculus that lead to Laplace's equation. But my heart is really in solving Laplace's equation or Poisson's equation by finite differences. That'll be today, or by finite elements, that'll be Wednesday and probably Friday. So this is like the core of the computational side but we had to do this preparation. And then you know that Peter kindly changed his weekly review session to Tuesday, I think it's Tuesday at noon; you could look on the website. So Peter will be a Tuesday review and then regular, that's me, on Wednesday. And then the exam, I think, is Wednesday evening. I think, yeah. Let me just do a review.

Oh, Peter sent me a note about a question on an old exam. I think it was 2005, question three. I think it was, 2005 question three. Peter said he didn't understand, at all, the solution to 3c. Nor do I. I think somehow, I think there was a little confusion in that, and the answer in there is the answer to a different question. Forget it. Having looked at this one, a and b are worth looking at in this div grad potential stream function world that we're in. Those are typical exercises in that area. Yes, question.

AUDIENCE: [INAUDIBLE]

PROFESSOR STRANG: Oh, good question. Yes. Thanks for reminding me. So, solutions to the homework that you're practicing with. If anybody has typed up solutions or could, we'll post them right away. So I'll just hope for an email from somebody who solved some or all of those problems. And of course you'll see me

again Wednesday. But that's an invitation for anybody who worked through those homeworks that I'll never collect. The current homework. OK, thanks for that. Other questions about the exam, and then you'll have another chance Wednesday to ask about that, so those are the topics. OK. And before I get to the fast Poisson solvers there's so much about Laplace's equation to say and I think it still remains to write Green's formula on the board. And that's part of the big picture of Laplace's equation is to see this. So we have double integrals now, that's the difference. But we're still integrating Au against w. So Au is the gradient end of u, w is w_1, w_2, and this is now for any u and any w, so I want the inner product. Of the gradient view with any w, so this is like the Auw part, so the gradient view, that would be a du/dx times the w_1. And a du/dy times w_2. dx/dy, that would be. So inner products, if I'm in two dimensions, inner products are integrals, of course, if I'm dealing with function. So this is like the Au against w. Inner product. Au, is those two pieces, w is those, there's the dot product.

And now the key idea behind what Green's formula is about, what we use it for in 1-D is integration by parts. So an integration by parts is going to produce a doubled integral in which, so I'm just sort of thinking through Green's formula. Of course, we could just say OK, remember that formula. But you want to just see it as an integration by parts. See how it evolved. And then we can think. So I plan to take two integration by parts, that x derivative will get taken off of u and put onto w_1. So I'll have u*dw_1. Oh, there'll be a minus right, from the integration by part. u will multiply minus dw_1/dx, so that was this guy integrated by parts. Now this one integrated by parts, a y derivative is coming off of u and onto w_2. So that'll leave me with u times dw_2/dy with the minus sign. That comes also from that integration by parts. And then there's the boundary term. So there's the term which, now the boundary of the region is the curve around it instead of just the two points. And what do we see in the integration by parts? Well, from the boundary term we expect a u and then the key point is it's (w dot n)ds. It's the normal component of of this vector w. Because we're looking, well, that's just what it is.

So that's Green's formula written out in detail. And this is the formula that tells us that here, if this was a, a being gradient, on the left side I'm seeing grad u in a product with w, and over here I'm seeing the inner product of u with, what's that? If I look at minus, if I have this quantity there that's in parentheses, what's its name? It's the divergence. It's minus the divergence of w. Plus the boundary term. So this is why I allowed myself this gradient transpose equal minus divergence to see that if A is the gradient then A transpose will be minus the divergence. And this shows how that, just sort of shows you the integration by parts that makes that true. So, you know there's so many integral formulas in this world of vector calculus. But this is like, one to know. And actually, if you want to know, OK what about the details, where did that come from? This actually turns out to be, and the text will make it clear, is the divergence theorem. The divergence theorem gives us this, this, exactly this, so the divergence theorem applied to the vector field u times w. It turns out if I apply the divergence theorem to that, then I get a whole lot of terms from the divergence of that, and they are these. And then I get this for the boundary. To flex out. OK, so. I just didn't want to leave the subject without writing down the central thing. What does this tell us, then? I have a region. In that region I have Laplace's equation or Poisson's equation. Say Poisson. On the boundary. Now, this is the final thing to ask you about. What are suitable boundary conditions for this problem in 2-D? Right? In 1-D, by now we know boundary conditions. That we know the main ones. There's fixed, where u or given. Or there's free where the slope is given. Now, what do we do here?

So this is where we used to be in 1-D. That's a straight line. Not perfectly straight but anyway, straight line. So 1-D there are only two boundary points. The normal vector goes that way. And that way, this is the end in this formula. I think if I just wrote ordinary integration by parts it would look just the same and my boundary reduces to two points. Here my boundary's the whole curve around. So what corresponds to fixed boundary conditions, and what corresponds to free boundary conditions? So I want the one corresponds to fixed, those are the ones that named after Dirichlet. Other names? Or the rest will be free, because the ones named after Neumann, and those are what we call natural boundary conditions. So I haven't, just ready to finish this story here. What are the possibilities? Well, in each we could have part of the boundary could be fixed. Part could be free. So fixed would mean say some other boundary, let me call this part of the boundary fixed. So the temperature, for example is fixed on this part of the boundary. So this would be like, and this might be the whole boundary. It might be fixed on the whole one, but I'm now allowing the possibility that we have here a fixed part and here we have a free part. In fluids, here we have fixed corresponds to, like, there's some obstacle there. Some wall. Free is where the fluid is coming in or flowing out. So there it's a velocity condition.

OK, so fixed conditions, these Dirichlet ones will be like, tell me what u is, u is given. On this part. So on part of the boundary, or possibly all, I could be given u=u_0. It could be zero, as it often was in 1-D. Often just took u to be zero. Now, what I'm asking maybe for the first time, is what's the free boundary conditions? What do we prescribe w=0 on the free part of the boundary? Is w=0 the correct free condition? Or w=w_0? If we wanted to allow it to be non-zero. That would sound right, right? u on one side, w on the other. But now we're in 2-D. u is still a scalar. That's fine. But w is now a vector. So if I was to prescribe w on this part of the boundary, I'm giving you two conditions, not one And that's not good. I can't give you more than one boundary condition. And the clue is there. It's w dot n, it's the outward flow. That you could prescribe. You can't say in advance what the tangential flow should be. So the free conditions would be like w dot n, which is of course, since w is v, we have Laplace's equation here with c=1 in between w and v, so this is the gradient of u dot n, grad u dot n, and it's sometimes written, the derivative of u in the normal direction. And that we can prescribe as some w_0. Oh, I don't to say w_0 because w_0 makes it sound like a vector would be allowed, and that's the whole point here. We give one boundary condition; we either give u or w dot n. So let me give it some different name, like F.

So. I'll just ask one more question to bring this home. And then I'll get onto the fun today of fast Poisson solvers. Suppose what was the deal on in 1-D when we had free-free boundary conditions? What was the deal in one dimension when we had free-free boundary conditions? Both conditions were slope conditions. u'=0. What matrix did we get? What was different about that matrix, the free-free case? I'm always looking back to 1-D because that's the case we really understood. What was the the problem with the free-free matrix? It was singular. What was its name? Was K, was it? You remember the first day of class, the unforgettable four matrices? So this was fixed-fixed, fixed-fixed. This one was, you could tell me better than I can remember. T was free-fixed. And that was still OK. By OK I mean it was still invertable. What was B? Free-free, that's what I'm looking at. And the problem with free-free and then this was periodic, that's Fourier stuff that's the next part of the course. But the point is that this free-free was singular. Right? And it'll be the same here. If the whole boundary is free, then I've got a singular problem. If I've got a

whole boundary that's free, yeah actually I'll just put it here. Suppose I have Laplace's equation, zero, and suppose I make the whole all free. So grad u dot n is zero on the whole boundary. So it's like solving Bu=0. And the point is that that has solutions. You remember, what was in the null space of B, this free-free case? Bu=0, for what vector? Our favorite guilty vector. It was constants. Right, all ones. All constants. u=1, all constants. Right, so that was the free-free case in 1-D. Now, I just want you to tell me the same thing over here.

That equation does not have a unique solution. So it's a singular problem. Here's the equation, something u equals zero, but you can tell me solutions to that pure Neumann problem. That are not zero. So this is a problem where there's a null space. And what is the solution? What's a solution to this? To the Laplace's equation in the inside and slope zero all around the boundary. There's a simple solution to that one, which is just like this guy. And do you see what it is? u equal constant, right. So u equal constant. So I'd like you to see all the whole picture connecting with what we fully understood in the 1-D case for those matrices. We don't expect to be able to have a pure Neumann problem. A totally free problem. Because these, it'll be singular and we'll have to do something special. So the point is that it would be enough for me just to fix that little bit. If I just fix that little bit and prescribe u=u_0 on a little bit, I could make the rest of it free. But I can't make it all free. OK. So those are things which, just to complete the big picture of Laplace's equation. OK, now I'm going to turn to solving. So now I want to solve Laplace's equation and then we'll have, because we had the review sessions coming on Tuesday and Wednesday of the past, now we're looking future. Forward. Ready to look forward. Laplace's equation. On a square. On a square. OK.

So let me draw this square. Laplace's equation or Poisson's equation. In a square. OK, I'm going to make it the boundary conditions, so I'm going to have a mesh. This is going to be finite differences. Finite differences will do fine on a square. As soon as I draw this curved region, I better be thinking about finite elements. So that's what, today is the, like, square day. OK, so square day I could think of finite differences. I just draw a mesh, let's make it small. So this is all known. These are all known values, let me just say u is given. On the boundary. I'll do the Dirichlet problem. So at all these mesh points, I know u. And I want to find u inside. So I've got nine unknowns, right? Nine interior mesh points. Three times three three. So I've got nine unknowns, I want nine equations and I'm thinking here about difference equations. So let me write my equation -u_xx-u_yy=f. Poisson. What finite difference, I want to turn this into finite differences. That'll be my system of nine equations. And the unknowns will be the nine values of u, so these are unknown. Right? Those are the unknown. OK, and I need equations. So let me take a typical mesh point, like this one. I'm looking for the finite difference approximation to the Laplace. If you give me Laplace's equation and I say OK, what's a good finite difference approximation. That I can then put in the computer and solve, I'm ready to do it. So what should we do for this one? For that term? What's the natural, totally natural thing to do to replace the finite difference replacement of minus u_xx? Second difference. Second difference. And with the minus sign there, second differences around this point will be a minus one of there, two of those, a minus one of those. That second difference in the x direction is my replacement. My approximation. And what about in the y direction? Same thing. Second differences, but now vertically. So now I have minus one, two more, that makes this guy a four. And this is minus one. And on the right hand side, I'll take the value of f at the center point. Do you see that molecule? Let me highlight the molecule there. Four in the center minus ones beside it. Our matrix, we're going to produce a KU=f matrix, and what's the K going to be? It's k2D. Let's

give a name. A new name. We've had k forever. Let's have K2D. And u is, so tell me now again just so we got, what size is K2D for this particular problem? What's the shape of it? How many unknowns have I got in u? Nine. Nine unknowns. And notice, of course, near the boundary, if this was the, well let me take the first one. What does the very first row of K look like? If I number nodes like this. So this will be node number one, two, three, four, five, six, seven, eight, nine. So number one is sort of close to a boundary. So I have here a minus one, a two, and this is gone. That's like a column of our matrix. removed. Like a grounded node, or whatever. But then I have the vertical one minus one, two, making this into a four. And this guy will also be removed. So I think if I look at K2D, it's nine by nine.

I don't plan to write all 81 entries. But I could write the first row. The first row comes from the first equation. So the first row has an f_1 on the right hand side, f at this point. And what do you see, what's on the diagonal? Of K2D? Four. And what's the rest of the row? Well there's the next point and it has a minus one. And then this point is not connected to that, so it's a zero. I'd better leave room for nine by nine here. Four, minus one, zero. And now what about number, the rest of the row? So there's for is the center, minus one, zero. And now I come to here, so what's the next entry in my matrix? Minus one. We're multiplying the unknown, the fourth unknown. And what about the rest of the row? This guy is not connected to five, six, seven, eight, or nine. So now I have zero, zero, zero, zero, zero. OK. So that would be a row that would be a row that's sort of different because it corresponds to a mesh point that's way over near the corner. How about the fifth row? What would be the fifth row of K2D corresponding to this guy that's right in the middle? What's on the diagonal, what's the 5, 5 entry of K2D? Four. Good. I'm going to have a diagonal of fours. I'm going to have fours all the way down the diagonal. Because every point the molecule is centered at gives me the four, and I just have a minus one for its neighbor. And this guy has all four neighbors. They're all alive and well, so there's a guy, this is the fifth row. There's somebody right next door on the left. Somebody right next door on the right. And then do the the other minus ones go after that? This is the whole point. I want you to see how this K2D matrix looks. Because it's the one, it's the equation we have to solve. So we've got to know what this matrix looks like. So what's the point here? Four, it's got a neighbor there. This was unknown number five. Its fourth unknown and the sixth unknown, what are the numbers of these unknowns? Two and eight.

So this guy is going to have a minus one in the two position. And then zero. And this guy, it'll also have a minus one, is that nine? Yeah, do you see how it's going? I have a diagonal, I have five diagonals somehow. I have diagonal, the neighbor on the left, the neighbor on the right, the neighbor underneath, and the neighbor above. Across the street you put something. So our matrix, this very important matrix, more attention has gone into how to solve this equation for that matrix than any other single specific example in numerical analysis. It's just a model problem. And what do I notice about this matrix? So now you see it's got five diagonals. But, and what's the big but? The big but is that those five diagonals don't run, it's not just a band of five because you don't get to the one up above until you've gone all the way, you see the bandwidth is big. If my matrix, if this is one, two up to n, and this is one, two up to n, then how many unknowns have I got? Just think big now. We're in 2-D, think 2-D. How many unknowns have I got? n was three in the picture, but now you're kind of visualizing a much finer grid. How many unknowns? n squared unknowns, right? One to n in each direction. So we've got the matrix of size n squared. Way bigger, way bigger. I mean if n was ten, our matrix is of size 100. And of course, that's completely simple. The real question is when n becomes 1,000 and

our matrix is at size a million. And those get solved every day. So, question is how. Actually this, for a million by million matrix, this fast Poisson solver that I want to describe does it great. Actually, so you can deal with a matrix of order a million without turning a hair.

OK now, but can we understand that matrix? So suppose n is 100. Just to have us. What's the story? This matrix, then. What's the size of that matrix? Did I say 100? Or 1,000, do you want to think really big? I guess I said 1,000. And I can get up to a million. OK, so n is 1,000, so we have a 1,000 by 1,000, a million unknowns. My matrix is a million by a million. Let me ask you this. Here's the question that I want to ask you. What's the bandwidth? How far from the main diagonal out to that minus one. In other words, this is really the full band of the matrix. It's got a lot of zeroes inside the band. A whole bunch. Maybe 990 something zeroes in here. And in here, and how far is it from here to here? 1,000, right? It's 1,000. So the bandwidth is 1,000. And what if I do ordinary, so now I want to begin to think how do I solve this equation? How do I work with that matrix? And you have an idea of that matrix? It's a beautiful matrix to work with. It's very like our K matrix; it's just up to 2-D. In fact, it's closely connected to our K2D will be closely connected to K and that's what makes life possible here. But suppose I had to solve this equation. Alright, so I give it to MATLAB just as a system of equations. A million equations, a million unknowns. So MATLAB takes a gulp and then it tries, OK? So ordinary elimination, that's what backslash does.

By the way this matrix is symmetric, and you won't be surprised it's positive definite. Everything is nice about that matrix. In fact more than I've said. OK, what happens if I do elimination on that matrix? How long does it take, how fast is it? Well, MATLAB will take advantage, or any code, will take advantage of all these zeroes, right? In other words, what do I mean by take advantage? And what do I mean by elimination? You remember I'm going to subtract multiples of this row. They'll be a minus one below it on this guy. This row that has some fours and some minus ones, whatever. Elimination, I'm subtracting a multiple of this row from this row to get that minus one to be a zero, right? I'm eliminating this entry in the matrix. This is ordinary, LU, this is ordinary LU of K2D. I'm sort of thinking through LU of k 2-D. And what I'm going to conclude is that there ought to be a better way. It's such a special matrix, and it's going to be messed up by LU. And what do I mean by messed up? I mean that all these great zeroes, 990-something zeroes there and there, inside the band, are going to fill in. That's the message, if you want to talk about solving large linear systems, the big issue is fill in, it's a sparse matrix. That's typical. We have a local operator here, so we expect a very local, sparse matrix. Large sparse matrices is what we're doing now. And the problem is that zeroes here will never get touched because we don't need any elimination, they're already zero. But all this diagonal of minus ones when we do eliminations to make those zero, then some non-zeroes are going to come in and fill in the zeroes.

So the work, how much work would elimination be, actually? We could even think about that. Can I just give the result for how much time would elimination take? Ordinary elimination. So I have n squared rows to work on. n squared rows to work on, and then the distance, that distance was what in terms of n, the width there is? n, right? It was about 1,000. So I get n numbers here and I have to work on n numbers below. To clean up a column I'm working with a vector of length n and I'm subtracting it from n rows below. So cleaning up a column takes me n squared operation. The total then is N^4. N^4, for elimination. And that's not acceptable. Right? If n is 1,000, I'm up to 10^12. So elimination is not a good way to solve.

Elimination, in this ordering, ha. Yeah. So can I tell you two ways, two better ways to deal with this problem? So two better ways to deal with this problem. One is, and MATLAB have a code that would do it, one is to change from this ordering one, two, three, four, five, six, seven, eight, nine, change to a different ordering that had less fill-in. So that's a substantial part of scientific computing. Is to re-order the unknowns. To reduce the number of wasted, fill-in things that you have to fill in, and then you have to eliminate out again. And I can get that N^4 down quite a bit that way.

OK, so that's a topic of importance in scientific computing. Is given a large linear system, put the rows and columns in a good order. OK, and that's something you hope the machine can do. The machine will have a record of where are the non-zeroes. And then so you run the little, you really run the program twice. You run it once without using the numbers. Just the positions of non-zeroes to get a good ordering so that the number of elimination steps will be way down. And then you run it with the numbers in that good order to do the elimination. So that's a topic of, one topic in this. But. Today's topic is using the fact that this is a square. That this matrix has a special form. That we can actually find its eigenvalues and eigenvectors. And that's what makes a fast Poisson solver. So that's the topic of the lecture and the topic of the next section in the book. And I will get started on it now, and then complete it Wednesday. So I want to take this particular K2D for a square and show a different way to solve the equations that uses the fast Fourier transform. When I see, when you see, a regular mesh, regular numbers. Nothing varying, constant diagonals, think fast Fourier transform. The FFT has got a chance when you have matrices like this one. And this I want to show you how it works. I want to show you the underlying idea, and then we'll see the details of the fast Fourier transform in the Fourier section. So, of course, everybody knows the third part of this course, which is coming up next week, we'll be well into it, is the Fourier part. OK. But I can give you, because we've seen the eigenvectors and eigenvalues of K, I can do this now. Ready for the smart idea?

The good way, and it only works because this is a square, because the coefficients are constant, because everything's beautiful, here's the idea. OK. Central idea. The idea is, so I have this matrix K2D and I'm trying to solve a system with that particular matrix. And the fantastic thing about this matrix is that I know what it's eigenvalues and eigenvectors are. So let's suppose we know them. So I know (K2D)y=lambda*y. That's when y is an eigenvector, and remember it's got n squared components. Lambda's a number. And I've got a whole lot of these, y_k, lambda_k, y_k, and I've got k going from one up to n squared. Now I want to use them. I want to use them to solve (K2D)u=f. Let me say, it's totally amazing that I would, in fact this is the only important example I know in scientific computing, in which I would use the eigenvalues and eigenvectors of the matrix to solve a linear system. You see, normally you think of eigenvalue problems as like, those are harder. It's more difficult to solve this problem than this one. Normally, right? That's the way to think about it. But for this special matrix, we can figure out the eigenvalues and eigenvectors by pencil and paper. So we know what these guys are. And that will give us a way to solve a linear system. So can I remember, this is now central message.

What are the three steps that to use eigenvectors and eigenvalues in solving linear equations, in solving differential equations, in solving difference equations, they're always the same three steps. Can I remember what those three steps are? So step one, so I'm supposing that I know these guys. First of all, that's amazing. To know

them in advance. And the second amazing thing is that they have beautiful structure. And those facts make it possible to do something you would never expect to do. Use eigenvalues for a linear system. So here's the way to do it, three steps. One, expand f as a combination of the eigenvectors. c_1*y_1, up to however many we've got. We've got n squared of them. Gosh, we're loaded with eigenvectors. Do you remember, this a column of size n squared. Our problem has size n squared. So it's big, a million. So I'm going to expand f in terms of the million eigenvectors. OK. So that tells me the right hand side. Now, you remember step two in this one, two, three process?

Step two is the easy one, because I want to get, what am I looking for? I'm aiming for, of course, the answer I'm aiming for is K2D inverse f. Right? That's what I'm shooting for. That's my goal, that's the solution. So here I've got f in terms of the eigenvectors. Now I want to get, how do I get u in terms of the eigenvectors? Well, what's the eigenvalue for the inverse, K to the inverse? Do you remember? It's one over. It's one over the eigenvalue, right? If this is true, then this will be true with lambda_k inverse. It's simple and I won't stop to do it, but we could come back to it. So I know how to get K2D inverse. So that's what I'm going to do. I'm going to divide, so here's the fast step. Divide c_k, each of those c_k's, by lambda_k. What do I have now? I've got c_1/lambda_1*y_1 plus c_n squared, the last guy divided by its eigenvalue times its eigenvector. And that's the answer. Right, that's the correct answer. If this is the right hand side, then this is the solution. Because why? Because when I multiply this solution by K2D it multiplies every eigenvector by eigenvalue, it cancels all the lambdas, and I get x. Do you see, I can't raise that board, I'm sorry. Can you see it at the back? Should I write it up here again, because everything hinges on that. The final, the answer u, the K2D inverse f, is the same combination that gave f. But I divide by lambda_1, c_2 y, the second eigenvector over lambda_2 and so on. Because when I multiply by K2D, multiplying each y, each eigenvector will bring out an eigenvalue, it will cancel at and I'll have the original f. So do you see that those are the steps?

The steps are, you have to do a, you could say it's a transform into eigenspace. I take my vector in physical space, it's got its n squared component. At physical points. I express it in eigenspace. By that I mean it's a combination of the eigenvectors and now the n squared physical values are n squared amplitudes, or amounts of each eigenvector. So that was a, that's the Fourier series idea that's the Fourier transform idea, it just appears everywhere. Express the function in the good basis. We're using these good functions. In these good functions, the answer is simple. I'm just dividing by lambda. Do you see that I'm just dividing by lambda, where if it was a differential equation, this is just what we did for differential equations. What did I do for differential equations? Do you mind if I just ask you? If I was solving dU/dt=(K2D)U. Starting from f, start with U(0) as f. I just want to draw the analogy. How did we solve these equations?

I expanded f in terms of the eigenvector, and then what was step two? For the differential equation, I didn't divide by lambda_k, what did I do? I multiplied by e^(lambda*k*t). You see, it's the same trick. Just get it in eigenspace. In eigenspace it's like one tiny problem at a time. You're just following that normal mode, and so here instead of dividing by lambda, I multiplied by either the lambda*t. And for powers of for K2D to the 100th power, I would multiply by lambda to the hundredth. Here, I have K2D to the minus first power. So I'm multiplying by lambdas to the minus one. OK, and the key point is that these c's can be found fast. And this summing up can be done fast. Now that's what also very exceptional. And

that's where the FFT comes in. The fast Fourier transform is a fast way to get into eigenspace and to get back. When eigenspace happens to be frequency space. You see that's the key. In other words these y's and lambdas that I said we knew, these y's, the eigenvectors, are pure sine vectors. So that this is an expansion is a sine transform. S-I-N-E. A pure sine transform. And that can be done fast, and it can be inverted fast. So you see what makes this fast Poisson solver work, is I have to know eigenvectors and eigenvalues, and they have to be fantastic vectors like sign vectors. OK, so I'll give you more, the final picture the details, Wednesday and then move on to finite elements.