PROFESSOR: Finite difference methods for initial value problems that we're coming to the end of and the solving large systems that we're coming to the beginning of was to talk today about matrices, because that language is just useful for everything.

So about the homeworks, Mr. Cho put in a long weekend grading the homeworks that were turned in Friday and preparing code and output to go up on the web page, probably late tonight or tomorrow. So have a look to see and I hope that those codes will be useful for the future too. Right.

So we'll maybe say more about the outputs and about the projects, which by the way, could grow out of that homework or could go in a totally different direction. I'm thinking that the right time to say projects due would be after spring break. So pretty much nearly immediately after the spring break would be -- and we'll talk about it more, but just so you have an idea of what's -- what timetable I had in mind.

So, matrices then. In particular, finite difference matrices. So that's the second difference matrix, k, and I'll frequently use that letter k as I did in 18.085 for that very, very important and useful matrix. So what are its properties? it's tridiagonal. That's a very important property, which we'll see because that means that computations solving linear systems are very fast with a tridiagonal matrix. It's symmetric. All its Eigen values are positive, so I would say it's symmetric positive definite and those Eigen values and Eigen vectors -- so the Eigen vectors for this matrix turn out to be discrete -- not quite discrete exponentials, discrete sines, discrete sine function. If I change the boundary conditions, I can get discrete cosines as the Eigen vectors or I could get discrete exponentials as the Eigen vectors by making it periodic.

Maybe I'll mention how to make it periodic and then I'll erase it again. To make it periodic means that this second different centered at point 1 should look ahead to point 2 and look behind to point 0, but that'll be the same as point n, so I would put a minus 1 in that corner and this one similarly, it looks ahead, which really brings it around again, since we're sort of on a circle, brings it around again here.

So that matrix now with minus 1s added in the corner, I would call c, a circular. So I'll leave the letter k there, but the right letter is c while these minus 1s are here to make it periodic. By the way, they mess up that the tridiagonal. It's no longer tridiagonal. Also, it certainly got -- it's very sparse. Mentioning sparse reminds me, if you're coding large matrices that are sparse, you should let MATLAB know that they're sparse. So MATLAB lab has a whole -- it carries out operations, if you tell it it's a sparse matrix, then it only operates where the non 0s are located and it doesn't waste its time looking at these 0s, these 0s through all the matrix steps.

So using sparse MATLAB is important thing to know about it. It just typically and s or an sp will appear in MATLAB commands. For example, I'll just maybe fill it in here. What's the sparse identity matrix? The normal identity matrix would be i of n and the sparse identity matrix is sp, speye of n. Similarly, we would create k -- we could use 2 times speye of n as the diagonal of k and the two, the upper and lower diagonals -- and we could tell it these two entries. Another thing to say -- what I said I wasn't going to do, I'll do because I hate to see a k up there while it's not right.

Two more points about this matrix. It's singular now. The determinate is 0. Now I'm never going to take the determinate of a giant matrix. That's a bad thing to do. Much better to recognize that there's a vector, x, let's say and it's the vector of all 1s. It's the vector of n 1s. If you imagine multiplying this matrix by the vector of n 1s, what do you get? You get all 0s. So that vector of all 1s is in the null space, I would say, of a matrix. Null space is just the vectors that get wiped out by the matrix. cx is all 0. So that vector, this matrix has some non 0 vectors in its null space. I know right away then, its determinate is 0. So the determinate of that is 0 and now that would tell me something about the Eigen values of the matrix. It tells me about one Eigen value. It's 0.

A matrix, like c, that has cx equals 0 -- I could also say that that cx equals 0 x. That would actually be better, so that both sides are vectors. So I'm realizing that that vector in the null space is an Eigen vector and the corresponding Eigen value is 0. Otherwise, the Eigen values will all still be positive. So this would be a positive semidefinite. I would call that matrix positive semidefinite, the semi telling me that it isn't quite definite, that it gets down and has 0. The matrix is singular, but still it's good to know where all the other n minus 1 Eigen values are. They're all positive.

About the bandwidth -- let me go back to k now. Because the bandwidth, strictly speaking, the bandwidth of c is very large. the bandwidth is, if I'm looking for only one number, that number tells me how many diagonals I have to grow until I reach the last non 0. So the bandwidth here is large and in general, the operation count, the amount of work to do, is going to grow with a bandwidth. Of course, not having full bandwidth isn't quite the full story for this matrix, because it's so sparse. I've got hundreds of 0 diagonals in between and just this one.

Anyway, this is a matrix with a large bandwidth, but a little deceptive. Now here's the matrix with a -- back to k again. I call the bandwidth just 1 here. Really, maybe half bandwidth would be a better word. The bandwith is the number of diagonals above or below -- take the maximum of the count above and the count below and in this case, both counts are 1. 1 diagonal above, 1 diagonal below, so really, half bandwidth, I would say, is 1.

Just some convention is needed there. The crucial point is that the bandwidth measures the amount of work to do when you do elimination, as MATLAB will do, of course. One other -- the thing about MATLAB -- so I'm often referring to MATLAB and I'm thinking of its backslash command. The backslash, which solves ax equal b by just a backslash b. So if it doesn't know these matrices are sparse, it will go through all the steps, not taking advantage of the fact that we've got all these 0s here. If it does know that they're sparse, then it's tremendously fast. Let me come back to that point. Maybe actually backslash is smart enough to look to see whether the matrix, whether sparseness is available. So I shouldn't have said -- I think maybe there's a lot engineered into backslash. Actually, backslash, also engineered in there is the least squares solution. If you give it a rectangular problem and, say, too many equations, so that you can't expect to have an exact solution, backslash will pick the least squares solution and much more. That's probably the most used operation.

So I said something about the Eigen values and everybody sees that if I multiply this matrix by the values of u at successive mesh points, I'll get the second difference that corresponds to uxx. Now this would cause this matrix -- so that tells me I'm taking a finite difference. That plus tells me I'm going in the forward direction, so that's 1 at the mesh value to the right minus 1 of the mesh value at the center point. Of course, this has to be divided by delta x and that by delta x squared. So these are matrices, along with the backward difference and the center difference, out of which you build the basic finite difference equation, as you've done.

I want to make a comment here though, about now on this topic of Eigen values. Eigen values for k really tell you the truth about the matrix k. The Eigen values of that matrix k start just a little above 0 and they go to a little before 4. So the Eigen values for k -- so I put that here, the Eigen values of k are between 0 and 1. They come very close to 4 and quite close to 0, depending on the size of the matrix of course. Let me just do -- if I took the one by one case, its Eigen value is 2. If I took the two by two case, its Eigen values are 1 and 3. Notice nice properties there. The Eigen values 1 and 3 are positive, as we said. This matrix k has positive Eigen values, whatever side. What's more, the 1 and 3 kind of interlace the 2. So what I'm saying is, the Eigen value for that is in between the Eigen value for the two by two and the two Eigen values 1 and 3 are in between the three Eigen values that I would get for the three by three case. Maybe I just write those down. Those are useful numbers.

So k 2 as lambda equal 1 and 3. The three by three one, I think the Eigen values are 2 minus root 2, which is smaller than 1, 2 which is in between and 2 plus root 2, which is larger than 3. Of course, they're all between 0 and 4. Just a comment. How do I know they're between 0 and 4? There's a somewhat handy little rule for getting the location of Eigen values, that's just worth knowing as a sort of general principle, but of course it can't tell you exactly where they are. First of all, the fact that the matrix is the metric tells us what about the Eigen values? So we learn a very, very important fact about the Eigen values from just looking at the matrix and observing that it's symmetric. That tells us that the Eigen values are real. They're real numbers.

Actually, it tells us something equally important about the Eigen vectors. The Eigen vectors of the matrix are orthogonal. The symmetric matrix has real Eigen values, orthogonal Eigen vectors. That's a little bit of linear algebra to know. Now why between 0 and 4, though? There's there's this -- what's his name? Gershgorin. Gershgorin pointed out -- and it's a two line proof -- that every Eigen value is in one of his -- one or more of his circles.

So where are the Gershgorin circles? A typical Gershgorin circle is centered at the number here, 2, that's on the diagonal and its radius is the sum off the diagonal, but you take absolute values. Gershgorin wasn't doing any of the fine points that really locate the Eigen value. So in this matrix, all the centers are at 2, the diagonals and the radii are 2, also 2, because that and that and absolute value make 2. This first row makes a 1 and actually, that's what -- that by a little careful argument, is what gets the Eigen values not actually touching 0 or touching 4. It takes a little patience. All you could say from here -- all Gershgorin could say what would be, the Eigen values are in a circle, centered at 2, its radius is 2, so they're between 0 and 4. Then it's that first and last row -- either the first or the last row would do to say that they're strictly positive. Of course, the true second difference, I should divide k by delta x squared. So the Eigen values will be divided by delta x squared divided by a small number, so they will really be much larger. The ratio of the biggest to the smallest isn't affected by the delta x squared and that's a key number. Let me put that maybe on the next board for k.

So lambda max is about 5. Lambda min, the smallest Eigen value is close to 0. How close? It's of the order of some constant over n squared. Now I use the word condition number and that -- so let me write that down. Condition number -- say c of k for condition number is the ratio lambda max over lambda min. Of course, I'm using here the fact that k is a symmetric matrix. Symmetric matrices are so beautiful that their Eigen values give you a reliable story here. So 4 divided by this -- the main point is that it's o of -- it's a border 1 over n -- sorry. When I divide by lambda min, that puts the n squared up in the numerator. It's o of n squared, growing like n squared, and that condition number, somehow it measures how close the matrix is to being singular, because it involves this lambda min, which is the smallest Eigen value and would be 0 if it were singular and it's scaled by lambda max. So if I'm multiply the matrix by 100, what happens to the condition number? No change. No change, because I'm doing lambda max over lambda min. Both lambda max and lambda min would be multiplied by 100, but the ratio wouldn't be different

So this is a useful measure and o of n squared, that's a big number if n is large. A big numbers if n is large and that condition number, where does it come in elimination? In elimination, the round off error -- roughly, the rule of thumb is that you would -- if the condition number is 10 to the eighth, you might lose eight significant bits in the back slide. You could. So this condition number measures how sensitive your matrix is to round off.

So that's a few thoughts about matrices and that matrix k in particular. Now what about the other matrix, the first differenec? The point I want to make about that matrix is, what about it's Eigen values? What are the Eigen values of that upper triangular matrix? They are, if you remember linear algebra -- but I can just tell you quickly -- the main point for a triangular matrix, the Eigen values are sitting on the diagonal. So this matrix has the Eigen value minus 1 repeated n time. That true fact is totally misleading, totally misleading. The Eigen values for this triangular matrix don't even notice what I've got above the diagonal and somehow they can't give a reasonable picture of what the matrix is actually doing. So maybe that's my warning here, that for a matrix which is absolutely not symmetric, right? I mean, not at all symmetric. For the center difference, which is -- what's the center difference? I was going to say symmetric, but it's the opposite. Center difference would be -- let's put delta centered down here. Center difference would be I'd have a 1. 0s would go on the point that -- for the central value. 1 would multiply the forward value and minus 1 would multiply that and then I'd have 1/2 and then I'd probably have a delta x. But the main point is, my matrix word look something like this. Minus 1 and 1s on two diagonals. Now we could find the Eigen values of that matrix. Do you know anything about the Eigen values?

This is a chance for me just to speak for a few minutes about useful facts that you can tell about a matrix just by looking at it. So what do I see when I look at that matrix? Is it symmetric? It's the opposite of symmetric, because the symmetric matrix up there, if I transpose it, it doesn't change. If I transpose this one, I'll get some kind of a backward difference. If I transpose this one, then the 1s and the minus 1s will reverse. I'll get the negative. So the rule for this center difference is -- so shall I -- how am I going to call center different? Del centered, for the moment. The transpose of that is minus itself. So what does that tell me?

First, that matrix is -- it's the opposite of symmetric, but it's actually OK. What I mean by OK is, its Eigen vectors -- we're back to orthogonal Eigen vectors. I didn't say anything about the Eigen vectors of del plus, but actually that was the biggest problem. This matrix del plus has one Eigen value repeated n time and it has only one Eigen vector, not -- it doesn't even have a full set of Eigen vectors, much less orthogonal ones.

So that matrix is like -- you don't want to trust the Eigen value picture that you get from a matrix like that. Here this anti-symmetric matrix can be trusted. Its Eigen value picture is reliable. It does tell you what's going on. The Eigen vectors are orthogonal. They're complex, actually. Actually, they'll look a lot like our e to the ikxs. So we don't panic when we see complex Eigen values. The Eigen values are -- do you know what the Eigen values looks like for anti-symmetric matrix? They're pure imaginary, just the way that when we took second differences -- maybe I'll just put here the center difference, the centered first difference, when we applied it to -- I want apply to eo to the ikx to find it's what factor comes out. So I get e to the ik plus 1 k plus 1 delta x from the plus side.

The is the matrix I'm doing with 1 and minus 1. So the minus 1 will get me minus e to the ik minus 1 delta x and of course, I factor out of that the e to the ikxs, so I'm left with e to the -- the thing that factors out is e to the i delta x minus e to the e minus i delta x -- and what's that? That multiplies the eo to the ikx, the Eigen vector. This is like the Eigen value and what do I say about that quantity? Of course, it's 2 i sine delta x. Pure imaginary and I should have divided by the 2, which would take away that 2. So it's pure imaginary. In reality and in this Fourier analysis, both are giving this understanding of what i is.

So that when we did this sort of operation, von Neumann's rule of following the exponential, we got something reasonable. When we do it with this one, it's von Neumann that's reliable and the Eigen values that are not reliable. So the Eigen values of this being all minus 1s is nonsense, doesn't tell us what the Fourier difference operator is really doing. But von Neumann tells us what is truly going on. Of course, that would be the same thing in which this minus 1 wouldn't appear and the 2 wouldn't appear.

So what would we get out of von Neumann? So this is for delta plus. I would factor out an e to the i delta x minus 1. That's what would multiply it e to the ikx. Oh, e to the i -- sorry. Yes, that's right. That would multiple e to the ik delta x. Sorry, should've had delta x there, but I wasn't paying attention to that. Here I was paying attention to this and it was pure imaginary. Here I'm paying attention -- here von Neumann at least is paying attention to this and what's he seeing? Not pure imaginary or purely real, off in the complex plane and that's really what the rights growth factor or the right number to associate with frequency k for this forward difference.

So I guess I'm saying that von Neumann does -- did the right thing to come up with these pictures for the growth factors. Eigen values confirm that and really pin it down when they're reliable. Eigen vectors and Eigen values are reliable when Eigen vectors are orthogonal and that's for matrices that are symmetric or anti-symmetric. There's a little bit larger class of matrices that include orthogonal matrices, but beyond that -- actually, there's been a lot of discussion over many years of Eigen values and the -- for problems that are not controlled by symmetric or anti-symmetric matrices. The alternative, the more refined idea of pseudo-Eigen values is now appearing in an important book by Trefethen and Embry, and with many examples -- OK, I won't pursue that. Right. So this is some basic fact about those matrices, all of which are one dimensional.

Now this board prepares the way to get into 2D and 3D. So I just want to ask, what does the -- we didn't really do the heat equation or the wave equation in 2D, but we could have. The von Neumann test would be straightforward, but now I want to think about the matrices. What does the two dimensional second difference matrix look like? What I'm going to do, just to look ahead, I'm going to use MATLAB's operation called kron, short for Kronecker, to create a 2D matrix out of this 1D center difference. So if I think now about center differences, second differences -- so I'm approximating uxx plus uyy -- or really, I'm approximating minus uxx minus uyy, because that k approximates minus the second difference. What do I -- what do my matrices look like? What's their bandwidth? How expensive is it to invert them?

These are the key questions. What's the matrix k 2 d that corresponds to -- gives me second differences in the x direction plus second differences in the y direction. So I'll write k 2 d for that matrix. Let's get some picture of it. First of all, let me imagine that I'm on a squred with a square grid. Delta x in both directions Square grid, let me say n mesh points each way. So n, I don't know whether I'm counting -- right now, I won't worry whether I'm counting the boundary ones or not. Probably not. So n -- I'll say n point, point n, n points -- so n delta x and in this direction n delta x.

So my matrix is a border. It's of size n squared, the number of unknowns. Now I will be a little more careful. Here, let me take the boundary values as given. They're not unknown. So n in this picture is 4. One, two, three, four unknowns there on a typical row. Now I have to give them a new number -- five, six, seven, eight, nine, 10, 11, 12, 13, 14, 15, 16 -- and n being 4, n squared is 16 for that particular square. So my matrix is 16 by 16. But somehow I want to be able to create it out of 4 by 4 matrices like k. So k 1 d, which I'm just going to call k -- kn will be the 4 by 4 one. 2 minus 1 -- so that's the matrix that gives me second differences along a typical row or down a typical column.

But now what am I looking for? I'm looking to do both -- second differences in a row and a column. So if I pick a typical mesh point, like number 11. Mesh point 11 -- let me blow up this picture here. It's going to be influenced, mesh point 11, by 10 and 12. The second difference is in the x direction and by 7 and 15, notice those are not so close to 10 or 11. The second differences is in the y direction, so let me blow that up. So here's mesh point number 11 corresponding to row 11 of k 2 d.

So I guess I'm asking what role 11 of k 2 d will look like. So here's mesh point 9, 10 and 12, so I have a second difference in the x direction from uxx but minus uxx -- that means I can put a minus 1 there, a 2 there and a minus 1 there. Now I have the same in the y direction with 15 and 7, so I have a minus 1 in column 15, a minus 1 in column 7, so I have a 4 minus 1 and then two more for the center gives me a 4.

So a typical row will have -- it'll be sparse, of course -- it'll have a minus 1 in positions 7, a minus 1 in position 10, a 4, a minus 1 and a minus 1 over there in position 15. That's a typical row of k 2 d. It adds to 0. If Gershgorin got his hands on this matrix, he would say that since all the rows -- the 4 will be on -- will the 4 be on the diagonal? Yes, the 4 will be on the diagonal all the way. I guess let's see a little bit more clearly what the matrix k 2 d looks like. This is probably the most studied matrix in numerical analysis because his its a model of what stays nice and what gets more difficult as you move into two dimensions. So some things certainly stay nice. Symmetries -- so properties of k 2 d, k to d will be -- it'll be symmetric again. It'll be positive definite again.

So what does k 2 d -- it's 16 by 16 and a typical row looks like that. That's a typical interior row. What does maybe -- if I took the first row, what does the very first row look like if I put row one above it? Let me draw row one. So what's the difference with row one? It's these are boundary values. These are not -- these are going to show up on the right hand side of the equation. They're known numbers. They're not -- they don't involve unknown things, so the only neighbors are 2 and 5. So I'll still have the second difference, minus 12 minus 1 and minus 12 minus 1 -- still this five point molecule with four at the center, but there won't be so many neighbors. There'll be the neighbor at the -- just to the right. Neighbor number two and there'll be neighbor number five a little further along and that's it. It's like the 2 minus 1 boundary row. It's a boundary row and a boundary row hasn't got as many minus 1s because it hasn't got as many neighboring unknowns. That boundary row would have three neighbors. Row two would have a 4 minus 1 minus 1, but not -- nobody there.

So now I'm going to try to draw k 2 d. Let me try to draw k to d. I can do it. k 2 d will have, from the uxx, the second differences along the rows, it will have -- I'll have a row, row one, it'll have another k on row two. It'll have a k on -- a k for each row. These are blocks now. That's of size n by n. So all of those are, so the whole thing is n squared by n squared. Actually, I'll stop here.

If I wanted to create that matrix with the same k on -- it somehow the identity is in there and the matrix k is in there and that's -- let's see. It's one or the other of those. I guess it's the first one. So what's this Kronecker product? Now I'm saying what this construction is. It's very valuable, because it allows you to create matrices. If you created k as a sparse matrix and i as a sparse matrix, then the Kronecker product would be automatically dealt with as a sparse matrix.

So what's the rule for a Kronecker product? You take the first matrix, which is the identity. The rest 0s. So that's i. Then -- so that's 1 d and then you multiple -- each entry in i becomes a block with entry aij times this guy k. This'll be the 0 block, this'll be the k block, this'll be the k block. I take those 1s and multiply k and those 0s and multiple k and that's what I get.

So you see now the Kronecker product is a bigger guy. If matrix a was p by p and matrix b was q by q, then the Kronecker product would be p times q by p times q. It would be square again. It would be symmetric if a and b are symmetric. Actually, it would have various nice properties. Its Eigen values would be the products of the Eigen values of a times the Eigen values of b. It's a very handy construction and here we saw it in a pretty easy case where a was only the identity. Let me do this. What does the Kronecker rule produce here? So I take this first matrix and I put it in here, 2 minus 1 minus 1, 2 minus 1 minus 12. So that's the matrix k and now each of those numbers multiplies the second thing, which here is i. So it's 2 i minus i minus i 2 i minus i minus i minus i minus i 2 i and 2 i. Now that is the second difference matrix that does all the columns. When I add those -- so now I'm going to add that to that. They're both size n squared. They give me k to d. So the neat construction of k 2 d is Kronecker product kron of ik plus kron of ki.

That's the matrix and let's look at what it actually looks like. We're seeing it block wise here. We saw it row wise here and maybe now we can take one more look at it, assemble it. So k 2 d -- I plan to add that matrix to that matrix to get k 2 d. So it will have -- since they both have 2s on the diagonal, it'll have 4s on the diagonal, so 4s all the way, but it'll be block wise. Up here, this block is k plus 2 i. So that block is -- goes down to 4. Now the k contributed minus 1s next to the diagonal, I guess that's it, right? k plus 2 i has that block as -- that's the one, one block. Why is it -- so we're seeing -- did I get it right? Yes, so we're seeing the neighbors to the right and left, but now let me bring in -- only now comes the neighbor above or below and that comes from this off diagonal block, minus i, which is then minus 1s to minus 1s. These are all 0 blocks. Here will be a minus the identity blocked. Down another block, the diagonal blocks are all the same and another one, another minute the identity.

Now we're getting to interior mesh points, where we see typical rows. So a typical role has the 4s on the diagonal, the minus 1s left and right and the minus 1s far left and far right -- and of course, what I'm going to ask you is, what's the bandwidth?

We're coming to the key point now. What's the bandwidth of this matrix? I only have two non 0s above the diagonal on a typical row, but I have to wait n diagonals before I get to the second one. So the bandwidth is n because I have to wait that long. Then the operation count -- if I just do ordinary elimination on this matrix, the operation will be the size of the matrix times a bandwidth squared. We can easily check that that's the right count of operations for a banded matrix. This is operations on a banded matrix. So what do we get? The size of the matrix is n squared. The bandwidth is n and it gets squared so we get n to the fourth. It's getting up there. If n is 1,000, we've got a serious sized matrix here. Still a very sparse matrix, so it's not like give up, but the question is, does the matrix stay sparse as we do elimination?

That's the center of the next lecture. Can we organize elimination? How closely can we reorganize elimination to preserve all these zillions of 0s that are between here? They're easy to preserve down here, way up there, but in this intermediate, those diagonals tend to fill in and that's not a happy experience. And It's even less happy in 3D.

So let me just do this same calculation it for 3D and then we're done for today. So k 3 d -- you might like to think how k 3 d could be created by the kron operation. Let me just imagine it. So what's k 3 d looking like? I've got three directions, so I have a 6 in the center and 6 minus 1s beside it and so 6 goes on the diagonal of k 3 d. 6 minus 1s go on a typical row and how long do I have to wait until I reach the last one? So I'm again going to do the size -- which is -- the matrix size is going to be like nq and what's the bandwidth going to be like?

Maybe I'll ask you now. What do you think for the bandwidth? How long -- if I just number it in the simplest way - and that'll be the key next time, is there a better numbering? But if I just number along rows until the rows fill up a plane of rows and then I move up to the next, the z direction to the plane above, then I have to wait -- this was the x and y, so this gave me a bandwidth of n, but that's not the bandwidth because I have to wait until I've finish the whole plane, until I go up to the next plane and catch this chap, so the bandwidth is n squared. Then the operations, which are size times bandwidth squared, we're up to n 7 and that is a really horrifying exponent to see into the seventh. That means that even for a moderate -- a problem in 3D with a moderate number, say 1,000 unknowns at each direction, we have -- we're looking, in theory, at a cost that we can't afford.

Of course, there's a lot to do here. So there's a lot to do if I stay with direct methods and that's what I'll do for the next couple of lectures. Then it's really in 3D that iterative methods become essential. I mean, this one, I could do those by direct methods, 2D, by a smarter direct method than any I've tried here, but in 3D, even smarter elimination is facing a serious exponent and loses to iterative methods. So that's what's coming, actually.

So today's lecture, you see, with the simplest possible matrices, what the central questions are.

Thanks and I've got homeworks coming back from Mr. Cho and I'll collect any that are ready to come in and see you Wednesday.