MIT OpenCourseWare
http://ocw.mit.edu

18.085 Computational Science and Engineering I, Fall 2008
Transcript – Recitation 4

PROFESSOR STRANG: I'm open for questions as always on every aspect, or comments on every aspect of the course. yeah?. The MATLAB problem, yes. I better remember what that MATLAB problem is. So it's solving the equation -u''+Vu' equal some delta function halfway along? And I chose, boundary conditions like those so that this gets replaced by a matrix K/(delta x squared). And this term is V times some center difference. I don't know whether I should use C. We called C earlier in the book, C was a circulant. This is a centered first difference over delta x equal right-hand side. All that multiplies the discrete solution u. I didn't get that very well for the camera. Maybe I'll say it again. We have the second difference that corresponds to -u''. And V times the first difference that corresponds to Vu'.

So the physical problem is, what happens is V gets larger. V grows. How does the solution change as V increases? So it's a very practical problem. The V measures the importance of this convection, the velocity of the fluid. Then n, which is, well maybe delta x is 1/(n+1), maybe. So that we would have n interior points and we know the boundary values. So the question is how do we interpret this increasing n? I guess we expect more and more accuracy, right? As n increases, delta x getting smaller. Our idea is that this should be a better and better approximation to the differential equation.

The question about the eigenvalues of this matrix, I don't know exactly the answer. So the eigenvalues, did you discover that the eigenvalues suddenly, they change their real, for small values of V and n. But then there's some point at which they become complex. And a point that involves both V and delta x actually, V and n. It's kind of interesting. I guess at this moment, interesting is the key word.

So what is the physics going on here? So I have a 1-D flow and it's blocked by these boundary conditions. Are sort of blocking it at both ends. And I'm looking for a steady state. And I'm feeding in this source here. So I'm feeding in a source halfway along. So I think that would be right. Now what's going to happen roughly? This term is, I think, flowing to the right. So it's going to carry, suppose this was smoke or particles or whatever. This is like an environmental differential equation or a chemical engineering equation. Just oh, and more and more applications.

So I think of it as this term is carrying the flow with it. Like, that way. So I'm expecting your solutions to be larger on this side. And how do we get any action at all on the left? That comes from the diffusion. The particles, when they enter, get carried away, and faster and faster as V increases. But at the same time they diffuse, they bounce back and forth. And some bit of it goes this way. But not a lot. I'm guessing that the solution would have a profile that would be kind of small. And

then there's whatever jump there has to be here, and then larger there. Oh, but then I have to get it to zero.

It's a strange situation. And maybe I'll take this chance to mention it. What happens as V gets really, really large? You could say ok, as V gets extremely large, forget that term. This is the important term. Vu' equals that. Which we could easily solve. But this equation with just this term is only first order, right? It only has a first derivative. And how many boundary conditions would we expect to have if I gave you, if this wasn't here and I gave you just a first order equation? One. And we've got two. Which we're imposing.

So this, sort of, limit as V increases, it produces something called a boundary layer. The solution is forced to satisfy these boundary conditions. One of them, it's quite happy about. The other one, it didn't really want to satisfy. It only satisfied them because, I had two originally with this part. But as this takes over, the struggle to satisfy that second boundary condition that it really doesn't want to satisfy is resolved by something, a layer that just makes a, sort of, exponential correction at the boundary to get to where it's supposed to go. So, anyway. This is a model problem that has boundary layers. Those are terrifically important in aerodynamics, you have layers around the actual aircraft. So lots of physics and computational science is hiding in this type of example. So those are a few words. But not an answer to your question.

I guess I'm happy if you, for example, let me know where the eigenvalues change from real to complex. I mean, it happens, but you can probably, if you look at the matrix, you can see what happens at the point where that change takes place. That's some comments. And with the MATLAB, pure MATLAB part, I guess I'm hoping, and the graders are hoping, that by providing a code we'll get a pretty systematic set of answers for the requested V equal, what was it, three and 12 maybe, or something. And the requested n(delta x)'s. So I'm hoping that everybody's graph is going to look pretty similar for those requests. And then if you could do a little exploration yourself and make that a fifth page, or really less than a page, about take V larger. Take n larger. What happens? I'm very happy. Or any direction. I mean, just think of it as a mini-project to do the requested ones and then to do a little experimentation. I'll just be interested about anything you observe. So there's no right answer to that part. Is that any help with the MATLAB?

Let me just say, since you showed up today, if you have trouble with the MATLAB and you need Peter's help at noon Friday it would be ok to turn in the MATLAB later on Friday. I shouldn't say that. But I just did. So there you are. Right. I mean, this course, as you've begun to see, I know that you have lots of demands on your time. And sometimes you have job interviews. All sorts of stuff. Conferences that you have to go to. We deal with that. So just give me the homeworks as soon as you can. If you're ready Friday at class time, that's perfect. If are stuck on MATLAB and you want to go to Peter's discussion, which follows the Friday class in here, that's fine too. So that's some thoughts about the MATLAB and it's partly open-ended.

What about lots of other things in this course? Homework came in. How was the homework? Maybe I just take this chance to ask. Was it a reasonable length? The homework three, the pset? Not too bad? I will be able to post solutions because several people are contributing typed solutions that could go on on the math.mit.edu/cse website. Or they could also go on our 18.085 website for this semester. Good.

Ok, help me out with some questions. Thanks. So the question is about element matrices. In lecture 8, I guess it was, where we were assembling this, I used the word that's used by finite element people, we were assembling K. And one way to do it was to multiply the three matrices. But that's not how it's actually done. It's actually assembled out of small matrices. Maybe small k would be the right. So this would be a k_element out of smaller element matrices. So for example, what's the element matrix for, one for each spring in this particular problem of springs and masses. So let me draw some springs, some masses, some springs, another mass, more springs, fixed, not fixed, whatever, maybe fixed-free the way I've drawn it there. So here's a spring with spring constant c_2. And we could look at the contribution to the whole matrix coming from this piece of the problem.

This was actually, the finite element method has a wonderful history of people, and it had different names way back, of people seeing the structure as broken in pieces and then connected together. And then what did a typical piece look like? So a typical piece there, well, you let me just write down what this matrix is going to be. The little element matrix is coming from spring two. So this would be, like element two will be, it'll have a c_2 outside. I'll put the c_2 outside. And then inside will be a little this guy. And we can talk more about why it's that one. But just to have the element matrix there on the board. So my claim is that this is a small piece of the big K. So the big K matrix, what's the size of the big K, of K itself? Three by three in this case, yeah, three masses. So it'll be three by three. So I'm thinking that this spring which connects mass one to mass two, so it's only going to be like, a two by two piece, a little local piece, you could say, that that little k fits in this, is assembled into the, I better call it k_2, right? So it's a little k that sits up in this two by two block and doesn't contribute to the rest.

Then let's just draw the rest of the picture. So this would produce an element matrix that looks just the same, that's the beauty of this. That all the elements, apart from change in the spring constant, look the same. So there'd be a little k_3. And where will it go? This is the whole core to the point. It'll go to the lower right. Down here? Overlapping, overlapping. Because this mass is attached to that spring and to that one. So these little element matrices, they overlap. And you just need, if you can imagine the code that's going to do this, you need a list of springs and a list of masses and a list of the connections between them. And it'll sit in here because it's two by two. So that's two by two, that's two by two and in this overlap entry will be a c_2 from the upper box. It'll be a c_2+c_3 as we discovered by direct multiplication. So that's that spring.

Now what about this first spring? So there's a little k_1. Now k_1 should look the same as this. Except what? So it's going to be a difference with this spring because? Because of this fixed. End. There's no mass zero. k_1 would normally sit up here, but actually it's only going to be one by one. So the k_1 little element matrix would look like c_1[1, -1; -1, 1]. And then the boundary conditions, knock those out. And an interesting point if you're writing big code you have to decide. Do I, as I create k_1, this little element matrix, do I pay attention to the boundary conditions? And then k_1 would just be that single number c_1 which would sit there. So up there will be c_1 from the k_1 matrix and c_2 from the k_2 matrix. That's the entry there.

And then, as I say, in coding finite elements, which you guys may do at some point, there's a question. What's sufficient? Shall I pay attention to the boundary in creating these element matrices or shall I do it later? And the rule seems to be, do it

later. So what gets created in finite elements is a, in our language, would be a free-free matrix. It's a matrix where even this spring has a two by two piece.

So what's the problem with the free-free matrix? The free-free matrices, those with no supports, those matrices will be singular. Right, singular. Because the vector of all ones, if you multiply the K, the free-free matrix times the vector of all ones, you get zero. The whole thing could shift. And will have other rigid motions in two dimensions. Let's just think ahead here. What are the rigid motions, the null space of K, you could say, for a two-dimensional truss. So I've got a bunch of bars and springs. Think of a mattress. I'm in two dimensions, a mattress is a bunch of springs connected in a 2-D grid. And say it's free at both ends. So what can I do to a mattress? Oh, my gosh! I didn't expect that to be on videotape. So it's in the plane. We're in 2-D here. What can I do if there are no boundary conditions? Well I can shift it to the right. Right? That's our . What else can I do? I can rotate. And I can shift it the other way. So there would be three rigid motions for the 2-D problem. Two translations and a rotation. And when you get up to three dimensions, do you want to guess what's the number in 3-D? Six. Number six. Three translations and rotations around three axes. So those, either one rigid motion or three rigid motions or six rigid motions have to get, the boundary conditions eventually have to remove those, have to knock out rows and columns and remove them.

But my point was just that quite efficient to do it later. The picture is so clear here. So the actual matrix would be four by four, the unreduced matrix. And then when we fix node zero there, that would knock out that part and leave the three by three that we want. So this is just discussion of how to think of this matrix K. So the direct way to think of it was as a product of big matrices. But in reality it's assembled from these element pieces. And of course, our goal in talking about finite elements will be to see that. It'll come up again, of course.

Your good question led me there, but did I answer the question? And maybe the way I mentioned it in class was to notice that these guys, these element matrices can be thought of this way. It is matrix multiplication, but done differently. It's a column of this matrix times the number C here times a row of this. So it's matrix multiplication, columns times rows. So you can multiply AB, columns of A times rows of B and then add over from one to n. So column of A times row of B. So a column, then, is a vector like this. A row is a vector like that. And the result is a full size matrix, but if the column is concentrated at a couple of nodes and the row is, then it will have zeroes everywhere except at that.

This is the element matrix with the C included. That would be the element matrix already blown up to full size by adding zeroes elsewhere. So, I mean the heart of the element matrix is where the action is on that spring. And then, when we assemble it, of course, it doesn't contribute down there. So the technology of finite elements is quite interesting and it fits. It's a beautiful way to see the discrete problem.

Ready for another question of any sort. Thank you, good. Yeah, sorry, it got onto the problem set. And then I thought-- let me write those words down first, singular value decomposition. Well I won't write all those words. That's a wonderful thing. It's a highlight of matrix theory except for the length of its name. So SVD is what everybody calls it. It's only like, every year now people appreciate more and more the importance of this SVD, this singular value decomposition. So shall I say a few words about it now? Just a few. So it's Section 1.7 of the book. And my thought was, hey we've done so much matrix theory including eigenvalues and positive

definiteness, let's get on and use it. And then I can come back to the SVD in a sort of lighter moment. Because I'm not thinking you will be responsible for the SVD. Eigenvalues I hope you're understanding. Positive definite I hope you're understanding. That's heart of the course stuff. But SVD is a key idea in linear algebra, but we can't do everything. But we can say what it is.

What's the first point? The first point is that every matrix, even a rectangular matrix, has got a singular value decomposition. So the matrix A can be m by n. I wouldn't speak about the eigenvalues of a rectangular matrix. Because if I multiply, you remember, the eigenvalue equation wouldn't make sense. Ax=lambda*x is no good if A is rectangular. Right? The input would be of length n and the output would be of length m and this wouldn't work. So eigenvectors are not what I'm after. But somehow the goal of eigenvectors was to diagonalize. The goal of eigenvectors was to find these special directions in which matrix A acted like a number. And then, as we saw today in the part that partly still up here, we could solve equations by eigenvectors by looking for these, following these special guys.

What do we do for a rectangular matrix? What replaces this? So this is now not good. The idea is simply we need two sets of vectors. We need some v's and some u's. So that's the central equation of the SVD. Now what can we get? So now we have more freedom because we're getting a bunch of v's that have, these guys are in our n. They have length n, right to multiply A times v. And the output is, so the n of these, n v's, we're in n dimensional space. So those are called singular vectors. They're called right singular vectors because they're sitting to the right of A. And these guys, these outputs will be in-- these are v's in our n, n dimensions. I have m of these, m u's in m dimensional space. And these things are numbers, of course. And actually, they're all greater or equal to zero. So we can get, by allowing ourselves the freedom of two right singular vectors and left singular vectors, we can get a lot more, and we can get, here's the punch line. We can get the v's to be orthogonal, orthonormal. Just like the eigenvectors of a symmetric matrix, we can get these v's, these singular vectors to be perpendicular to each other and we can get the u's to be perpendicular to each other.

What we can't, what we're not shooting for is the v's to be the same as the u's. They're not even in the same space now, if the matrix A is rectangular. And even if the matrix A is square but not symmetric, we wouldn't get this perpendicularity. But we get it in the SVD by having different v's and u's.

Here's my picture of linear algebra. This is the big picture of linear algebra. This over here is n dimensional space. We have v's. Think of that as n dimensional space. That's kind of a puny picture. But when I multiply by A, I take a vector here, I multiply by A. So let me just do that. I multiply by A, I take a vector v, well, already put v, I take Av and I get something over here. And this will be the space of u's.

Now I have to ask you one thing about linear algebra that I keep hammering away. If I take any vector and multiple by A, What do I get? If I take any vector v, like . Here's A say, [3, 6; 4, 7; 5, 8]. So that's A times v. What can you tell me about A times v that goes a little deeper than just telling me the numbers? It's a linear combination of the columns. These are the outputs, the Av's. So this space is called the column space. It's all combinations of the columns. These are all combinations of the columns. That's the column space. It's a bunch of vectors.

So the point is that these u's, that I have like, a fantastic choice of axes. A linear algebra person would use the word, bases. But geometrically I'm saying they're fantastic axes in these spaces. So that if I choose the right axes in the two spaces, then one axis will go to that one when I multiply by A. The next one will go to that one. The third will go to that one. It just could not be better. And that's the statement.

Now maybe I'll just say a word about where it's used or why it's useful. Oh, in so many applications. Well most of you are not biologists. And I'm not certainly. But we know that there's a lot of interesting math these days in a lot of interesting experiments with genes. So what happens? So we've got about 40 people here. And we measure the, well, we get data. That's what I'm really going to say. Somehow we got a giant amount of data. right?. Probably 40 columns. Everybody here is entitled to be a column of the matrix. And the entries will be like, have you got TB, what height, all this stuff. So you got enormous amount of data. And the question is, what's important in that data. You've got a million entries in a giant matrix and you want to extract the important thing. Well, it's the SVD that does it. You take that giant matrix of data, you find the v's and the sigmas and to u's. And then the largest sigmas are the most important information if things are scaled and statistics is coming into this also.

I could a give sensible, much more mathematical discussion of one word, one name it goes under is principle component analysis. That's a standard tool for statisticians looking at giant amounts of data. Is to find the principal components and those will come from these eigenvectors. Well your question about the SVD set off that discussion.

I'll only add a couple of words. These v's and these u's are actually eigenvectors. But they're not eigenvectors of A. v's happen to be the eigenvectors of A transpose A. And the u's happen to be eigenvectors of A, A transpose. And the linear algebra comes together to give you this key equation. And of course, you and I know that if I'm looking at the eigenvectors v of A transpose*A, A transpose A is a symmetric matrix. In fact, positive definite. So that the eigenvectors will be orthogonal, the eigenvalues will be positive. And then this one is coming from the eigenvectors of A A transpose, which is different.

So if the matrix A happened to be square, happened to be symmetric, happened to be positive definite, this would just be Ax=lambda*x. I didn't have to cross it out. I'll use K for that. So if the matrix A was one of our favorite matrices, was a K, that would be the case in which the SVD is no different from eigenvalues. The v's are the same as the u's, the sigmas are the same as the lambdas, all fine. This is sort of the way to get the beauty of positive definite symmetric matrices when the matrix itself that you start with isn't even square. Like this one. More than I wanted to say, more than you wanted to hear about the SVD.

What else? Yes, thank you. More about, sure. Let me repeat the question. So this refers to this morning's lecture, lecture 9 about time-dependent problems. And the point was that when I have a differential equation in time there're lots of ways to replace it by difference equations. So let's take the equation du/dt, let's make it first order is some function of u often and t. That would be a first order. Yeah, it's going to look at when I write that much down. What have I got? I've got a first order differential equation. Is it linear? No. I'm allowing some, this function of u could be sin(u). It could be u to the tenth power. It could be e to the u. So this is how

equations really come. The linear ones are the model problems that we can solve. This is how linear equations really come.

Euler thought of, let's give Euler credit here, so forward Euler and backward Euler. And the point will be that this guy is explicit. So can I write that word so I don't forget to write it? Explicit. And that this guy will be implicit. And this is a big distinction. And we'll see it. So this says u_(n+1)-u_n over delta t is the value of the slope at the start of the step. So that's the most important, most basic, first thing you would think of. It replaces this by this. You start with u_0, and from this you get u_1. And then you know u_1 and from this you get u_2. And the point is at every step this equation is telling you what u_1 is from u_0. You just move u_0 over to that side. You've only got stuff you know and then you know u_1.

Contrast that with backward Euler. So that's u_(n+1)-u_n over delta t is f at-- Now what am I going to put there? I'm going to put the end, the point we don't know yet. So is the slope at u_(n + 1) and the time that goes with it, t_(n+1). t_n is just a shorthand for n*delta t. n steps forward in time got us to t_n. This is one more step.

Now why is this called implicit? How do I find u_(n+1) out of this? I've got to solve for it. It's much more expensive. Because it'll be probably a non-linear system of equations to solve. We'll take a little time on that. But of course, there's one outstanding method to solve a system of non-linear equations and that's called Newton's method. So Newton is coming in. Newton's method is sort of the first, the good way, if you can do it, to solve. Well when I say solve, I mean set up an iteration which, after maybe three loops or five loops will be accurate to enough digits that you can say, ok that's u_(n+1). On to the next step. Then the next step will have the same equation but with n up one. So it'd be u_(n+2)-u_(n+1).

So you see the extra work here, but it's more stable. The way this one spiraled out, this one spiraled in in the model problem. I hope you look at that Section 2.2 which was about the model problem that we discussed today. There's more to say. I mean, this is the central problem of time-dependent, evolving initial value problems. You're sort of marching forward. But here, each step of the march takes an inner loop which has to work hard to solve, to find where you march to. Is that a help, to indicate? Because this is a very fundamental difference.

And Chapter 6 of the book develops higher order methods. These are both first order, first order accurate. The error that you're making is of the order of delta t. That's not great, right? Because you would have to take many, many small steps to have a reasonable error. So these higher order methods allow you to get to a great answer with bigger steps. A lot of thinking has gone into that and somehow it's a pretty basic problem. Just to mention for MATLAB, ode45 is maybe the workhorse code to solve this type of a problem. And the method is not Euler. That would not be good enough. It's called Runge-Kutta. Two guys, Runge and Kutta, figured out a formula that got up to fourth order accurate. So that's the thing.

And then if we looked further about this subject we would distinguish some equations that are called stiff. So I'll just write that word down. Some equations, the iteration is particularly difficult. You have two time scales. Maybe you've got things happening on a slow scale and a high scale. Multi-scale computations, that's where the subject is now. And so there would be separate codes with an S in their names suitable for these tougher problems, stiff equation. I guess one thing you sometimes

get in the review session is a look outside the scope of what I can cover and ask homework problems about.

I'm sure hoping that you're assembling the elements of computational science here. First, what are the questions? What are some answers? What are the issues? What do you have to balance to make a good decision?

Time for another question if we like. Yeah, thank you. Stability, yeah, right. This here? First, if we want a matrix then this has to be a vector of unknowns. I'm thinking now of a system of, this is n equations. I've got u is a vector with n components. I've got n equations here. The notation, I can put a little arrow over it just to remind myself. And if I want to get to a matrix I better do the linear case. So I'll do the linear case. The big picture is that the new values come from the old values. There has to be a matrix multiplier. Anytime we have a linear process. So the new values come from old values by some linear map. A matrix is doing it. So there's a sort of growth matrix. Can I just put down some letters here? I won't be able to give you a complete answer. But this'll do it. So u_(n+1) is some matrix times u_n. That's what I wrote down this morning for a special case. It's gotta look like that for a linear problem. And let's suppose that we have this nice situation as today where it's the same G at every step. So the problem is not changing in time, it's linear, it's all good.

What is the solution after n times steps compared to the start? So give me a simple formula for the solution to this step, step, step equation. If I started at initial value u_0 and I looked to see, well what matrix gets me over n steps, what do I write there? G to the n, right, G to the n. So now comes the question. The stability question is whether do the powers of G to the nth grow? And here's the point. That the continuous problem that this came from, it's got its own growth or decay or oscillation. This guy, the discrete one, has got its. They're going to be close for a step or two. For one or two steps I'm expecting that this is a reasonable consistent approximation to my problem. But after I take a thousand steps, this one could still be close or it could have exploded. And that's the stability thing. The choice of the difference method can be be stable or not. And it's going to be the eigenvalues of G that are the best guide. So in the end people compute the eigenvalues of the growth matrix and look to see are they bigger than one or not.

Let's close with that example because that's a good example. So it fits this, but not quite, because it wasn't completely forward or completely backward. And let's just write down what it was in that model problem and find the matrix. So can you remind me what I wrote today? So u, was it u first? U_(n+1) was U_n+delta t*V_n. And then the new velocity was the old velocity and it should be minus delta t times the u. Because our equation is, these are representing the equation. u'=v is the first one. v'=-u is my second equation. So and then the point was I could take that because I know it already, I could take it there. Where is the matrix here? I want to find the growth matrix that tells me now-- My growth matrix, of course, this is and this is . And this is a two by two matrix. And I hope we can read it off. Or find it anyway. Because that's the key.

How could we get a matrix out of these two equations? Let me just ask you to look at them and think what are we going to do. That's a good question. What shall I do? I would like to know the new U's, U, V from the old. What'll I do? Bring stuff onto, a U_(n+1) on this side and n on this side. This guy I want to get over here. Can I do that with erasing? So it's going to come over with a plus sign. So I guess I see here

an implicit matrix acting on the left and an explicit matrix acting on the right. So I see a . And I see my explicit matrix. Shall I call it E? That's the right sides. I see a one and a one and a delta t. And now my left side of my equation, the n+1 stuff. What's my implicit matrix on the left sides of the equation? Well, a one and a one and a plus delta t is here, right? And let's call that I for implicit. Are we close? I think that looks pretty good.

What's G? What's the matrix G now? You just have to begin to develop a little faith that yeah, I can deal with matrices, I can move them around, they're under my control. I wanted this picture. I want to move everything to the right. What do I do to move everything to the right-hand side? I want that to be over there. It's an inverse. It's the inverse. So G, the matrix G there is, I bring I over here. It's the inverse of I times the E. And I can figure out what that matrix is. And I can find its eigenvalues. And it'll be interesting. So actually that's the perfect problem. I mean, if you want to see what's going on, well the book will be a good help I think, but that's the growth matrix for leapfrog. And I'll tell you the result. The eigenvalues are right of magnitude one, as you hope, up to a certain value of delta t. And then when delta t passes that stability limit the eigenvalues take off. So that this method is great provided you're not too ambitious and take too large a delta t.

Thanks for that last question, thanks for coming.