PROFESSOR: Up on the web now are lots of sections, including this conjugate gradients, so today is the end of the conjugate gradients discussion, and it's, kind of, a difficult alogorithm. It's beautiful and slightly tricky, but if I just see what the point is of conjugate gradients, we don't have to check that the exact formulas that I've listed here -- so that's one step of conjugate gradients: going from k minus 1 to k, and we'll look at each of the five steps in the cycle, without patiently checking all those formulas. We'll just, sort of, see how much work is involved.

But let me begin where we were last time, which was Arnoldi. So Arnoldi was a Gram-Schmidt-like algorithm, not exactly a Gram-Schmidt that produced an orthogonal basis, and that basis is the basis for the Krylov subspace, so that each new basis vector in the Krylov subspace is found by multiplying by another power of a, but that's a terrible basis. So the basis that we get from these guys has to be improved, orthogonalized, and that's what Arnoldi does. And if we look to see what it does matrix-wise, it turns out to be exactly -- have that nice equation, that we're creating a matrix q with the orthogonal vectors, and a matrix h that tells us how they were connected to the column, to the matrix a. OK. So that's the central equation, and the properties of H are important.

So we saw the key property was, in general, we've got -- it's full up in that upper corner, just the way Gram-Schmidt would be. But in case a is a symetric matrix, then we can check that h will be symetric from this formula. And the fact that it's symetric means that those are zeros up there, because we know there are zeros down here. We know there's only one diagonal below the main diagonal. And if it's symetric, then there's only one diagonal above, and that's the great case, the symetric case. So I'm going to take two seconds just to say, here is the best way to find eigenvalues. We haven't spoken one word about computing eigenvalues, but this is the moment to say that word.

If I have a symetric matrix, eigenvalues are way, way easier for symetric matrices. Linear systems are also easier, but eigenvalues are an order of magnitude easier for symetric matrices. And here is a really good way to find the eigenvalues of large matrices. I assume that you don't want all the eigenvalues. I mean, that is very unusual to need many, many eigenvalues of a big matrix. You're looking -- probably those high eigenvalues are not really physically close, good representations of the actual eigenvalues of the continuous problem. So, we have continuous and discrete that are close for the low eigenvalues, because that's where the error is smooth. High eigenvalues are not so reliable. So for the low eigenvalues, here is a good way to do them. Do Arnoldi, and it gets named also now after Lanczos, because he had this eigenvalue idea. Run that for awhile.

Say if you want 10 eigenvalues, maybe go up to j equal to 20. So 20 Arnoldi steps, quite reasonable, and then look at the 20 by 20 sub-matrix of 8, so you're not going to go to the end, to h n. You're going to stop at some point, and you have this nice matrix. It'll be tri-diagonal, symetric tri-diagonal, and that's the kind of matrix we know how to find the eigenvalues of; so this is symmetric and tri-diagonal, and eigenvalues of that class of matrices are very nice, and they're very meaningful, and they're very close to -- in many cases -- I won't try to give a theory here, in many cases, you get very good approximations. The eigenvalues of this submatrix are sometimes called Ritz values, and they use the eigenvalues of this submatrix, and those are often called Ritz values. And so the first 10 eigenvalues of that 20 by 20 matrix, would be in many, many problems very fast, very efficient approximations to the low eigenvalues of the big matrix a that you start with.

OK, the reason is that eigenvalues -- do you all recognize that eigenvalues -- that this is the operation when I bring q inverse over here, that I would call these matrices similar -- two matrices are similar if there's a q that connects them this way. And similar matrices have the same lambda, the same eigenvalue. So the full matrix q, if we went all the way to the end, we could find its eigenvalues, but the whole point of all this month is algorithms that you can stop part way, and you still have something really good. OK, so that's -- I think we can't do everything, so I

won't try to do more with eigenvalues except to mention that. Someday you may want the eigenvalues of a large matrix, and I guess arpack would be the Arnoldi pack, that would be the package to go to for the Arnoldi iteration, and then to use for the eigenvalues. OK, so that's a side point, just since linear algebra is -- 1/2 of it's linear systems and 1/2 of it's eigenvalue problems, I didn't think I could miss the chance to do that second half, the eigenvalue part.

OK, now I'm going to tackle conjugate gradients, to get the idea. OK, so the idea, well, of course, a is symetric positive definiteness. If it's not, you could try these formulas, but you'd be taking a pretty big chance. If it is, these are just right. OK, so what's the main point about -- here's the rule that decides what xk will be. xk will be the vector in the Krylov, of course, xk is in this Krylov space k. So it'll be -- we can create xk recursively, and we should need, just, if we do it right, should need just one multiplication by A at every step, and then some inner product. OK, so now, what do we know from this? This vector, since it has this, it starts with -- it has an xk that's in the k'th Krylov subspace, but now it's multipled by a, so that's up to the k plus first. And b is in k1, it's in all the Krylov subspaces, so this is in, and therefore rk is in, k plus 1, right? This is in the next space up, because there's a multiplication by a. And therefore, if we choose to determine it by this rule, we learn that this vector -- I mean, we already know from Arnoldi that qk plus 1 is orthogonal to this space. This is the new k plus first vector that's in k plus 1, so this is just where this is to be found, so this vector rk is found in the next Krylov subspace, this one is in the next Krylov subspace. They're both orthogonal to all the previous Krylov subspaces, so one's a multiple of the other. So, and that will mean that automatically, that the r's have the same property that Arnoldi created for the q's they're orthogonal. So the residuals are orthogonal. So, orthogonal residuals. Orthogonal residuals, good.

I better say, by the way, when we do conjugate gradients, we don't also do Arnoldi, so Arnoldi's finding these q's; we're leaving him behind for awhile, because we're going to go directly to the x's and r's here. So, I won't know the q's but I know a lot about the q, so it's natural to write this simple fact: that each residual is a multiple of the new q, and therefore, it's orthogonal to all the old ones. And now I want to -- this

is the other property that determines the new x. So the new x should be -- what do I see here, I see a sort of delta x, the change in x, the correction, which is what I want to compute at step k, that's what I have to find. And I could look at the corrections at the old steps, and notice there's an a in the middle. So the corrections to the x's are orthogonal in the a inner products, and that's where the name conjugate comes from. So this is conjugate directions. Why do I say directions, and why do I say conjugate, conjugate gradients, even, if I could say. Why did I -- conjugate directions, I maybe shouldn't have erased that, but I'll also say conjugate gradients. OK.

So conjugate, all I mean by conjugate, is orthogonal in the natural inner product that goes with the problem, and that's given by a. So the inner product given by the problem is the inner product of x, and y is x transposed ay. And that's a good inner product, and it's the one that comes with the problem and so, natural to look at orthogonality in that sense, and that's what we have. So these are orthogonal in the usual sense, because they have an a already built into them, and these are orthogonal in the a inner product. OK. So I wrote down two requirements on the new x. Well, actually you might say, I wrote down a whole lot of requirements, because I'm saying that this should hold for all the previous i's. And this should hold for all the previous i's. And you see indices are getting in here, and I'm asking for your patience.

What's the point that I want to make here? It's this short recurrence point, this point that we had over here when h was just tri-diagonal, so we only needed, at the new step, the only things we had to find were the -- say at the second step, we just had to find that h and that h, because this one was the same as that. And then the next one, we -- maybe I didn't say that right. I guess when we -- let me start again. At the first step I need to find two numbers. Now because h is symetric, that's already the same as that. So at the next step I need these two numbers, and then that number is already that. So, two numbers at every step and big zeros up here. That's that's the key point when a is symetric. So the same will be true over here. We have two numbers, and they're called alpha and beta, so they're a little different numbers, because we're working with the x's now and not with the q's, so you don't see any

q's written down for conjugate gradients. OK.

All right, let me just look at the cycle of five steps, so I have to give it a start, so I start with d0 as b and x0 as 0, and I guess, r0 which, of course, is b minus ax0. If that's 0, it's also b. So those are the starts. So I'm ready to take a step. I know the ones with subscript zero, and I'm ready for the next step. I know the ones with subscript k minus 1, and I'm ready for this step, this cycle k r. So, here's a number. So I'm coming into this -- well let me say maybe how I should say it. I'm coming into that cycle with a new d, now what is a d? It's a search direction. It's the way I'm going to go. It's the direction in which this delta x is going to go. This delta x is going to be, probably I'll see it here. Yeah, there is delta x, is a multiple of d, right? If I put this on this side, that's delta x, the change in x is a multiple of d, so d is the direction in dimensional space, and I'm looking for the correction, and alpha is the number, is how far I go along d for the best, for the right, delta x.

So alpha will be determined by, probably by that. Probably by that, yeah, it involves an inner product, product picking off a component. OK? So the first step is to find the number; the next step is to take that update, delta x is the right multiple of the search direction that I'm coming in with. Then correcting r is easy; I'll come back to that; if I know the change in x, I know the change in r right away. So that's given me, I'm now updated. But now I have to look ahead. What direction am I going to travel next? So the next steps are to find a number beta and then the search direction, which isn't r; it's r again with just one correction. So it's a beautiful set-up formula, beautiful. And I could write, you know, because I know x's and r's, I could write the formulas, and there are different expressions for the same alpha and beta, but this is a very satisfactory one.

OK. Have a look at that cycle just to see what how much work is involved. How much work is involved in that cycle? OK, well, there's a multiplication by a, no surprise. It's that multiplication by a that takes us to the next Krylov space, takes us to the next update. So I see a multiplication by a, and also here, but it's the same multiplication. So I see one multiplication by a. So, easy to list, so each cycle there's one multiplication a times the search direction, and because a is a sparse matrix,

the cost there is the number of non-zeros in the matrix. OK. Then, what other computations do you have to do? I see an inner product there, and I'm going to use it again. I guess, and here it is at the next step, so I guess per cycle, I have to do one inner product of r with itself, and I have to do one of these inner products. So I see two inner products, and of course, these are vectors of length 10. so that's 2n multiplications, 2n additions. OK, this was a multiple of n, depending how far sparse a is, this is 2n adds and 2n multiplies for these two inner products. OK. Yeah. I guess you could say take them there.

And then what else do we have to do? Oh, we have these updates, so I have to multiply a vector by that scalar and add, so and again here, I multiply that vector by that scalar and subtract. Oh, and do I have to do it again here? Have I got three? Somehow, I thought I might only have two. Oh, yeah, let me say two or three vector updates. People are much better than I am at spotting the right way to organize those calculations. But that's really a very small price to pay per cycle, and so if it converges quickly, as it normally does, this is going to be a good algorithm. Maybe I'll -- maybe having introduced the question of how quickly does it converge, I should maybe say something about that. So what's the error, after k step? How is it related to the error at the start?

So, I have to first say, what do I mean by this measure of error, and then I have to say what's the -- I hope I see a factor less than 1 to the k power. Yeah, so I'm hoping some factor to the k power will be there, and I sure hope it's less than 1. Let me say what it is. I think -- so this is maybe not the very best estimate, but it shows the main point. It's the square root of lambda max minus the square root of lambda min, divided by the square root of that plus the square root of that. And maybe there's a factor 2 somewhere. So this is one estimate for the convergence factor, and you see what it depends on, I could divide everything by lambda min there, so that I could write that as the square root of the condition number that I'll divide by that minus 1 divided by the square root of the condition number plus 1 to the k power. So the condition number has a part in this error estimate, and I won't derive that, even in the notes, I won't, and other people work out other estimates for the

error, it's an interesting problem. And I do have to say that this norm is the natural inner product, ek, ek squared is the inner product of ek with itself, but again with a in the middle. I suppose that a little part of the message here, and on that middle board is that it's just natural to see the matrix a playing a part in the -- it just comes in naturally.

Conjugate gradients brings it in naturally. These formulas are just created so that this will work. OK. So I suppose I'm thinking there's one word I haven't really justified, and that's the word gradient. Why do we call it conjugate gradient. Here I erased conjugate directions I was quite happy with, the directions or the d's, because this is the direction in which we moved from x, so conjugate directions would be just fine, and those words come into this part of numerical mathematics, but where do gradients come in? OK. So, let me say, gradient of what? Well, what is -- so first of all, gradient of what is in these linear problems where ax equal b, those come from the gradient of the energy, so can I say, d of x for the energy, as 1/2x, transpose ax, minus b transpose x? You might say, where did that come from? Normally, we've been talking about linear systems, everything's been in terms of ax equal b, and now suddenly, I'm changing to a different part of mathematics.

I'm looking at minimizing -- so optimization, the part that's coming in April, is minimizing energy, minimizing function, and what's the link? Well, if I minimize that, the gradient of this is how shall I write, Grad E: the gradient of e, the list of the derivatives of e, with respect to the xi, and if we work that out, well, why not pretend it's scalar for the moment, so these are just numbers, then at this ordinary derivative of 1/2ax squared is ax, and the derivative of bx is b, and that rule continues into the vector case. So what have we got here? We've computed, we've introduced the natural energy, so this is a natural energy for the problem, and the reason it's natural is when we minimize it, set the derivatives to zero, we got the equation ax equal b, so that would be zero at a minimum. This would equal zero at a min, so what I'm doing right now is pretty serious. On the other hand, I'll repeat it again in April when we do minimizations, but this is the -- this quadratic energy has a linear gradient, and its minimum is exactly ax equal b. The minimum point is ax equal b. In other words, solving the linear equation and minimizing the energy are one and the

same, one and the same. And I'm allowed to use the word min, because a is positive definiteness, so the positive definiteness of a is what means that this is really a minimum and not a maximum or a saddle point.

OK. So I was just -- I wanted just to think about how do you minimize a function? I guess any sensible person, given a function and looking for the minimum would figure out the gradient and move that way, move, well, not up the gradient, move down the gradient, so I imagine this minimization problem, I imagine here I'm in x, this is the x1, x2, the x's; here's the e, e of x, it's something like that, maybe it's minimum is there. So the graph of e of x, and I'm trying to find that point, and I make as first guess, somewhere there. Somewhere on that surface. Maybe, so to help your eye, this is like a bowl, and maybe there's a point, so that's on the surface of the bowl. That takes zero, let's say. I'm trying to get to the bottom of the bowl, and as I said, any sensible person would figure out, well, what's the steepest way down.

The steepest way down is the direction of the gradient. So I could call this gradient g, if I want, and then I would go in the direction of minus g, because I want to go down and not up. I mean, we're climbing, I shouldn't say climbing, we're descending a mountain, and trying to get to the -- or descending a valley, sorry. We're descending into a valley and trying to get to the bottom. And I'm assuming that this valley is this nice shape given by just a second degree polynomial. Of course, when we get to optimization, this will be the simplest problem, but not the only one. Here it's our problem. Now so what happens if I move in the direction of -- the direction water would flow. If I tip a flask of water on the ground, it's going to flow in the steepest direction, and that would be the natural direction to go. But it's not the best direction. Maybe the first step is, but so I'm talking now about conjugate gradient seen as a minimization problem, because that's where the word gradient is justified. This is the gradient, and of course, this is just minus the residual, so this is the direction of the residual. So that's the question, should I just go in the direction of the residual all the time? At every step, and this is what the method of steepest descent will do, so let me make the contrast. Steepest descent is the first thing you

would think of, direction is r. That's the gradient direction, or the negative gradient direction. Follow r until, in that direction, you've hit bottom. So you'll go down this, you see what happens, you'll be going down one side, and you'll hit bottom, and then you will come up again on a plane that's cutting through the surface. But you're not going to hit that at first shot.

When you start here, see water would -- the actual steepest direction, the gradient direction, changes as the water flows down to the bottom, but here we've chosen a fixed direction; we're following it as long as we're going down, but then it'll start up again. So we stop there and find the gradient again there. Follow that down, until it goes up again; follow that down, until it goes up again, and the trouble is convergence isn't great. The trouble is that we end up -- even if we're in 100 dimensional space, we end up sort of just, our repeated gradients are not really in good new directions. We find ourselves doing a lot of work to make a little proress, and the reason is that these r's don't have the property that you're always looking for in computations orthogonality. Somehow orthogonality tells you that you're really getting something new. If it's orthogonal to all the old ones. OK. Now, so this, the direction r is not great. The better one is to have some orthogonality. Take a direction d that -- well, here's the point. This is the right direction; this is the direction conjugate gradients choose. This is where -- so it's a gradient, but then it removes the component in the direction we just took. That's what that beta will be. When we compute that beta, and I'm a little surprised that it isn't a minus sign. No, it's OK. It turns out that way, let me just be sure that I wrote the formula down right, at least wrote it down as I have it here, yep. Again, I'm not checking the formulas, but just describing the ideas.

So the idea is that this formula gives us a new direction, which has this property, too, that we're after. That the direction, it's not actually orthogonal, it's a orthogonal to the previous direction. OK, I'm going to draw one picture of what -- woops, not on that -- of these directions, and then that's my best I can do with conjugate gradients. I want to try to draw this search for the bottom of the valley. So I'm going to draw that search by drawing the level sets of the function. I'm going to -- yeah, somehow the contours of my function are, since my function here is quadratic, all the

contours, the level sets will be ellipses. They'll all be, sort of, like similar ellipses, and there is the one I want. That's the bottom of the valley. So this is the contour where energy three, this is the contour energy two, this is the countour energy one, and there's the point where the energy is a minimum. Well, it might be negative, it doesn't, whatever, but it's the smallest. OK. So what does -- can I compare steepest descent with conjugate gradient? Let me draw one more contour, just so I have enough to make this point. OK, so steepest descent starts somewhere, OK, starts there, let's say, or that's my starting point. Steepest descent goes, if I draw it right, it'll go perpendicular, right? It goes perpendicular to the level contour. You carry along until, as long as the contours are going down, and then if I continue here, I'm climbing back up the valley. So I'll stop here. Let's suppose that that happened to be tangent right there, OK.

Now, that was the first step of steepest descent. Now I'm at this point, I look at this contour, and well, if I've drawn it reasonably well, I'm following maybe perpendicular, and now again I'm perpendicular, steepest descent says go perpendicular to the contour, until you've got to the bottom there. Then down, then up, then do you see this frustrating crawl across the valley. It wouldn't happen if these contours were circles. If these contours, if a was the identity matrix, if a was the identity matrix, these would be perfect circles, and you'd go to the center right away. Well that only says that ax equal b is easy to solve, when a is the identity matrix. So this is steepest descent, and it's rate of convergence is slow, because the longer and thinner the valley, the worse it is here, OK.

What about conjugate gradients, well I'm just going to draw a magic -- I'm going to make it look like magic. I'm going to start at the same place, the first step, because I'm starting with zero, the best thing I can do is follow the gradients to there, and now, so that was the first step, and now what? Well, the next direction -- so these are the search directions. That's the search direction d1, d2, d3, d4. Here is d1, and what does d2 look like? May I cheat and go straight to the center. It's a little bit of a cheat, I'll go near the center. The next direction is much more like that and then goes to the bottom, which would be somewhere like there inside that level set, and

then the next direction would probably be very close. So this is not 90 degrees unless you're in the a inner product. So this is a 90 degree angle in the a inner product. And what does the a inner product do? In the a inner product, in which level sets are circles, or spheres or whatever, in whatever dimension we want. Well, I'm not going to be, I don't want to be held to everything here.

If you see this picture, then we've not only made headway with the conjugate gradient method, which is a big deal for solving linear systems, but also we've made headway with the conjugate gradient method for minimizing function. And if the function wasn't quadratic, and our equations weren't linear, the conjugate gradient idea would still be available. Non-linear conjugate gradients would somehow follow the same idea of a orthogonal search directions. So these are not the gradients, they're the search directions here. OK. That's sort of my speech about the conjugate gradients methods. Without having checked all the formulas in the cycle, we know how much work is involved, we know what we get, and we see a bit of the geometry. OK. And we see it as a linear systems solver and also as an energy minimizer, and those are both important ways to think of conjugate gradients. OK.

I'll just take a few final minutes to say a word about other -- what if our problem is not symetric positive definite. What if we have an un-symetric problem, or a symetric, but not positive, definite problem? Well, in that case these denominators could be zero. I mean, if we don't know that a is positive definite, that could be a zero, it could be anything, and that method is broken. So I want to suggest a safer, but more expensive method now for -- so this would be min res, minimum residual. So the idea is minimize, choose xk, then minimize the norm of rk, that's the residual. So minimize the norm of b minus axk. So that's my choice of xk, and the question is how quickly can I do it? So this is min res, OK. And there are other related methods, there's a whole family of methods.

What do I want to say about min res? Let me just find min res. Well, yeah, I guess, for this I will start with Arnoldi. So for this different algorithm, which comes up with a different xk, I'm going to -- and of course, the xk is going to be in the Krylov space kk, and I'm going to use this good basis. This gives me the great basis of q's. So I

turn my problem, I convert the problem. I convert this minimization -- I look for x as a combination of the of the q's, right? Is that how you see x equals qy? So instead of looking for the vector x, I'm looking for the vector y, which tells me what combination of these columns the q's x is. In other words, I'm working in the q system, the q basis. So I'm working with y. So I would, naturally, write this as a minimum of b minus aqyk. Same problem, same problem, I'm just deciding, OK, I'm going to find this x as a combination of the q's, because those q's are orthogonal, and that will make this calculation a lot nicer. OK, what's going to happen here? Do you know that I'm going to use the property of the q's, that aq is qh?

That was the key point that Arnoldi achieved. And the net result is going to be, of course, I have to do this at level k, so I'm chopping it off, and I have to patiently, and the notes do that, patiently see, OK, what's happening when we chop it off, chop off these matrices. We have to watch where the non-zeros appear. In the end, I get to a minimization problem for y, so we get to a minimum problem for the matrix h that leads me to to the y that I'm after. So it's a minimum problem for a piece, sorry, I should really say hk, some piece of it, like this piece. In fact, exactly that piece. So if I ask -- all right, suppose I have a least squares problem. This is least squares because my norm is just the square, the norm squared. I'm just working with l2 norm, and here's my matrix. It's 3 by 2, so I'm looking at least squares problems, in that matrices of that type -- and they're easy to solve. The notes will give the algorithm that solves the least squares problem. I guess what I'm interested in here is just is it expensive or not? Well, the least squares problem will not be expensive; it's only got one extra row, compared to the number of columns, that's not bad. What's expensive is the fact that rh is not tri-diagonal anymore, these are not zeros anymore. I have to compute all these h's, so the hard work is back in Arnoldi. So it's Arnoldi leading to min res, and we have serious work already in Arnoldi. We don't have short recurrences, we've got all these h's. OK, enough. Because arpack would be a good code to call to do min res. OK that's my shot at the key Krylov algorithms of numerical linear algebra, and you see that they're more subtle than Jacobi. Jacobi and Gauss Seidel were just straight simple iterations; these have recurrence that produces orthogonality, and you pay very little, and you get so much. OK, so

that's Krylov methods, and that section of notes is up on the web, and of course, at some point, numerical comparisons of how fast is conjugate gradients, and is it faster than direct elimination? You know, because that's a big choice you have to make. Suppose you do have a symetric positive definite matrix? Do you do Gauss elimination with re-ordering, minimum degree, or do you make the completely different choice of iterative methods like conjugate gradients? And that's a decision you have to make when you're faced with a large matrix problem, and only experiments can compare two such widely different directions to take. So I like to end up by emphasizing that there's still a big question. Do this or do minimum degree? And only some experiments will show which is the better. OK, so that's good for today, and next time will be direct solution, using the FFT. Good, thanks.