PROFESSOR: Several pieces of good news. Since Brett is back now, all of chapters, five and six of my notes, that is to say, all that we've covered in class, plus a little more, are on the web page now. So, I was cautious at the beginning of the semester and didn't put that up, and then forgot that it wasn't there. So now you have something to work with, particularly on any experiments, for example, minimum degree ordering. Oh and also there's a movie now. Tim Davis from Florida and [? Perils ?] [? Person ?], my friend here, created a little movie to show the order that elimination occurs. And I had a lot of email with Tim Davis. On that one page handout, it mentions 2d Laplace's equation. It mentions n cubed for the nested dissection, where you cut the region in half, and half, and half, by separators I believe, and I think experiments will show, that minimum degree is even better. But the analysis is extremely weak apparently on minimum degree. So anyway, one possible project either for now or for the final project would be some experiments to see what the actual convergence is like. And maybe a little understanding of minimum degree. So we'll talk more about that. Anyway, there's a movie and a lot more material there on the website. It'll get updated and corrected as I as I do the edits.

Second bit of good news. I made a mistake. That's not normally good news, but this time, I reported a little calculation for multi grid, and I computed the wrong thing. You remember that I computed m. m was the jacobian matrix. So m was the jacobian matrix, the iteration matrix with jacobian which is i minus the inverse a. This is for the 1d second differences. In fact, this was the 5 by 5 matrix that we've been talking about. And now this is multiplied by d inverse which is a half, but then with a weighting factor, that becomes a third. So that's weighted jacobian now. And this is what we would get for ordinary simple iteration. And it's not satisfactory. Remember it had an Eigen value lamda max. The spectra radius was about point 9.

Then I reported about the Eigen values m s, The multi grid matrix sm and I was a

little disappointed in it's Eigen values. The reason is, I should have been doing i minus sn. When I do that, I'm not disappointed anymore.

What are the Eigen values of this? You remember s is the matrix that tells us how much error we're capturing. s is that matrix that came from the error we captured at the end of multi grid, was s times the error that we entered with. And so the error that's left, the remaining error, the new error, is the rest. It's the part we don't get, and that's when I forgot. That's i minus s e. So that's why I should have been using i minus s. So then the step one of multi grid does it smoother. Steps two, three, four leave me i minus s. And step five did another smoother. And this is just with one jacobian step, which you would probably do three.

Anyway the Eigen values of this were, well -- this has two zero Eigen values, and three 1's. So this matrix has Eigen values 0, 0, 1,1,1. Because, why's that?

Well, you remember the Eigen values had to be zero or 1, because s squared equaled s, and i minus s squared equals i minus s. So. We remember that. Now the question is, what are these? And the answer is 1/9 is a triple Eigen value. I was astonished to discover 1/9. Of course, zero, zero survived. The rank is 3. And this is now looking much better. What I reported last time was some more like 7/8, or something, and I thought oh multi grid, it's not showing it true colors. But actually it does. This is a kind Eigen value we expect. Like 1/10, or a multi grid cycle. So much better if we just did 3 m's for example, I would get them down the point 9 q, which is more than point 7, where by doing multi grid instead, we're down to point 1.

OK. So that's the good news. And, of course, if you experiment a little, you'll find different numbers for different sizes and really see what's happening. And, in fact, you could do 2d.

I mentioned all this partly for now, if you're wanting to do it now, or partly for eventually later. Oh, and thinking about projects, just one word to repeat, that the Monday after spring break, I think that's April the 3, no class. So it'll be Wednesday that I'll see you again. But Mr. [? Cho ?] will be available his office hours. So that's April 3, if you wanted to discuss issues with your project with Mr. [? Cho ?] that

would be at class time in his office to 1:30, 1:00. Otherwise taking extra spring break, and I'll see you Wednesday.

OK. So that's various bits of good news. Oh. One other thing that I guess I didn't finish in that lecture, was to identify the two Eigen values. Well, Eigen vectors. Those will be in the notes too. I guess I found one Eigen vectors, 1,2,22,1. And glancing at the matrix, I didn't spot the other one, but it has one oscillation. It's 1,2,0, minus 2 minus 1. Anyway, this is the main thing. A much better result. So that's multi grid, which we now can use as our iteration. Or we can use it as a preconditioner.

Many of the recommended methods use multi grid as a preconditioner before something even more powerful. Well, I don't know about even more powerful. Multi grid people would say not possible. But some situations we may want to go to a different method but we want a preconditioner, and multi grid excellent.

By multi grid, I include the smoother. So it'd be that. That would be a possible preconditioner for something else.

OK. So where are we now. We're into the new section starting this moment. And what's the section about? It's about things associated with this name Krylov. So, I'm going to use the letter k for the things that are associated, and I'm always solving au equals b.

There's a Krylov matrix that created, exactly as you see here, this is the js the one with j columns. b, Ab, A squared, b up to A to the the j minus 1 b. And the Krylov space is the combinations of those vectors. That's what this word, span means. I could erase span, and say, "all combinations of" maybe that even more familiar. So span by is the same thing is saying all linear combinations of those j vectors. And, of course, that's the same as saying it's the column space of the matrix, because the column space is all combinations.

OK. So Why I am I interested? Why was Krylov interested? Why is everybody interested in these vectors? Because actually, that's what an iteration like Jacobi produces. If I use Jacobi's method, or Gauss-Seidel. Any of those. After one step,

I've got b. After two steps, there's a multiplication by a in there, right? And some combination is taken, depending on the particular method. So after two steps, I've got a combination of b and ab. After three steps, Jacobi produces some combination of b, ab, a squared b. In other words, all of those iterative methods are picking their j'th approximation in this space, kj. I should put a little j on it. So these spaces are growing. They grow by one dimension, by one new basis vector at each iteration. And the point is Jacobi makes a particular choice of xj or uj the approximation after j steps.

But does it make the best choice? Probably not. In fact, pretty definitely not. And so the idea is let's make the best choice. Lets not just use as simple iteration that built in a choice that might not be so terrific. Let's make the best choice. There are a whole lot of Krylov methods, which all choose xj -- since I think this section we use x, I'm going to change that x. They'll all the iterative. They'll all started with xzero as zero say. And then the first guess will be b maybe. Or maybe not. Maybe some multiple of b. Actually a good Krylov method will take the best multiple of b as the first guess, and not necessarily 1 times b. And onwards.

So we have this word, best, coming up. What's the best vector in that space? And there are different methods depending on what I mean by best. Oh, let me tell you the name of the method that I'm going to concentrate on first and most. Will be the will be the conjugate gradient method. CG. The conjugate gradient method. What determines xj? It it chooses xj in kj, the space that we always look for our approximation in. Let me not forget to say these vectors, b, ab, a squared b and so on, they're easy to compute, because each one is just a matrix multiplication from the previous one. And the matrix is, we're assuming we're working with sparce matrixes. And mostly, and especially, sometimes, especially symmetric ones. So just let me put in here, before I even finish that sentence, cg, this is for a transpose equal a symmetric. It's only for those. And positive definite so symmetric, positive definite.

So it's a limited class of problems, but highly, highly important class. And you may say what do we do if the matrix is symmetric, but indefinite? Well, that comes next.

Or what if the matrix is not to symmetric at all. Well, if you're brave, you might try conjugate gradients anyway. But if you're cautious, then you would use one of the other methods like min res. Maybe I'll just mention one more on our eventual list. Actually that tells us probably what choice it makes. Min res chooses the xj to minimize the residual. Minimum r. Remember r is b minus ax. The residual r will always denote the error in the equation. And so it's minimum residual. OK. So that's a natural choice, but you'll see that this is fantastic method. Superior, just quicker than min res for a nice reason so it chooses xj in kj and I think the rule it uses is that I think xj should be orthogonal I'll go and check it in my notes to the residual rj. Let me just be sure if I'm going to write that down, I better be sure I said it correctly. I actually I didn't say it correctly. r j is orthoganal to the whole space kj. It turns out that we can choose xj. When I make a choice of xj its in kj. Now when I compute r, there's a multiplication by a to get the residual, so that moves us up to the next space. And I'm going to make a choice where this is orthogonal to the whole space, kj. You have to do conjugate gradient. And I can't start on that for a good reason. I have to start with something called, arnoldi.

And and what's arnoldi about? Well. Let me come back to these vectors. Quick to compute. But what's the other property? Everything in applied numerical mathematics, you're choosing basis vectors, and you're looking for two properties. And I guess this is like a general rule, whenever you meet a whole new problem, in the end, you're going to construct some basis vectors if you're going to compute. And what properties are you after? You're after speed. So they have to be quick to compute and work with. Well these are. Multiplying by a is sparce matrix multiplication. You can't beat that. But the other property that you need is some decent independence. And not just barely independent. You want the condition number of the basis somehow to be good. Not in the normal. And best of all, you would like the basis vectors to be orthonormal. That's a condition number of one. That's the best basis you can hope for. And the point is, that this construction produces a lousy basis from the point of view of condition.

So arnoldi's job is orthoganalize the Krylov basis, which is b, ab, and so on, producing orthonormal basis q1, q2, up to qj. It's just beautiful. So arnoldi's taking

like a preliminary step Krylov has to make to get something that numerically reasonable to work with. Then if it's fast, we have to do a multiplication by a, and you'll see one here in the middle of arnoldi. And then, of course, if you look at those ten lines of MATLAB, and we could go through them a little carefully, but let's just take a first look.

The first step is like Gram Schmidt, right? It's sort of a Gram Schmidt idea. You take that first vector b, you accept that direction and the only step remaining is to normalize it. Divide by its length. Then you go on to the next. Then you have some trial vector t, which would in this case be ab, in the direction of ab, which won't be orthoganal to the original b, right. So this won't be orthogonal to that one. So, of course, you've got to compute an inner product between them. And subtract off the right multiple of the previous one from the current t to get the improved t, and then you're going to normalize it again. So you compute it's length and divide by the length.

You see that overall pattern in arnoldi? It's the familiar idea of Gram Schmidt of take a new vector, find it's projections on to the ones that are already set, subtract those components off, you're left with a piece that and you find it's length and normalize that to be a unit factor. It's very familiar. Its exactly the type of algorithm that you would write down immediately. And it just involved the one multiplication by a so that it's not going to cost us us a lot to make the vectors orthonormal.

Well. Wait a minute. Is it going to be expensive or not? That's like the key question. It's certainly going to produce a good result. Producing orthonormal vectors is going to be a good thing to do. But is it expensive or not? That depends. It's certainly not expensive if the new trial vector t has only components in one or two of the already settled directions. In other words, if I only have to subtract off a couple of earlier components, then I'm golden. And that's the case when a is symmetric. So that will be the key point here. And I just make it here. If a is symmetric a transpose equals a, then I only need and it's subtracting off where I'm headed for the new q, I only need to subtract off h j j, multiplying the q j, the q that was just set, and the one before. hj minus 1j. I'll call that a short recurrence. It's short because it only has two

terms. And then there'll be a new h. j plus 1j which is just the length. So there'll be one of these magic things in so many parts of mathematical analysis, a three term recurrence relation will hold -- in fact, I guess the reason we see three term recurrence relations in all the Legendre polynomials, Chebyshev polynomials, all those classical things. The reason is actually the same as it is here. That some thing in the background is symmetric.

I want to put a few numbers on the board so you see what is typical input and output to arnoldi. And you'll see it symmetric and then you'll see its short recurrence, and then we want to see what? OK. So I work out a typical arnoldi example. Not that one. Here. OK. I think this is a good example to look at. So matrix a is not only symmetric, its diagonal. So of course, to find a inverse b is not different. But we're going to do it through conjugate gradients. So that means that we don't figure out a inverse, which, of course, we easily could. We just use a very sparse, here's our right hand side. Here's our Krylov matrix. It has b in its first column. It has a times b in the second column. Multiply by a again to get the third column. And a one more time to get the fourth column. So that's k4, right? And for a 4 by 4 problem, that's the end. So actually here I'm carrying Krylov to the end of it. There's no more to do. Because once I've got this k j, the combinations of those are all of r4 all of four dimensional space. Oh, yeah. That raises an important point about how we're improving on iterations.

If I was using Jacobi, or Gauss-Seidel or one of those others, then, well, I won't say for this particular a, but usually, after four steps, I am by no means finished. I've taken four steps, I've gotten closer, but I haven't got the exact answer. But now, in this the world of product methods, after four steps, I will have the exact answer. Why? Because Krylov methods take the best x4 out of the space spanned by those four columns. Well, the whole space, r4 is spanned by those columns and taking the best one must be the answer. So x4 will actually be a inverse b, the answer we're looking for. So these methods are on the one hand, direct methods. They finish after four steps, finish. You've got the answer. Now. That's actually how they were born. As direct methods that gave the answer from an interesting iteration and they

nearly died for that reason. They were born and died, or were on death's door, as direct method. Because as direct methods, there are better ones.

Then some years later, people went back to it. Back to conjugate gradients, and noticed that thought of as just going part way, gave very successful answers. So we're going to think of Krylov. We're planning to stop before j equals n. In this picture, j equals n equal 4 here. But, we plan to stop for j much below n. We're thinking of applications where n is 10 the fifth, and we're going to stop at ten squared. 100 steps maybe. Or ten steps. So that meant a total reconsideration reanalysis of conjugate gradients, and it is now back to a big success.

Well its a rather unusual thing that a method that people have put aside gets picked up again and becomes a favorite as conjugate gradients have. First of all, I was saying that the basis, those basis vectors are not very independent. That's not a good basis. Of course, it's a detractive basis and maybe, do you know the name for a matrix that has this particular form? The columns are constant, then, that's the key column And then it's first powers, second powers, third powers. Do you remember whose name is associated with that matrix? It comes up in interpolation, and it starts with a V? Anybody know? Vandermon, yeah. Vandermon. So I could call it v for vandermonde. And vandermonde matrixes are not very well conditioned. So now a little time out to say, because it's so important, how do you judge the good or poor conditioning of a matrix? Those vectors are linealally independent. The determinant of v is not zero. A matrix is not singular, but it's too close to singular. Suppose you have a basis, as we have here. And you want to know is it good or not? So, of course, you always put your bases vectors, -- I don't want to call them v because -- well I guess I could call them v, they come out of Vandermon. v1, v2, v3, v4. That's our matrix. Essentially I want its condition number. And to find it's condition number, it's not symmetric. So I can't just take the Eigen value of that. You might say, look at Lamda max and Lamda min. The condition number is associated with max divided by min. But when the matrix isn't symmetric, just taking its Eigen values directly is not cool. It's not reliable. I could have a matrix that's badly conditioned, but all it's Eigen values were 1. I mean a matrix was 1s on the diagonal, and zillions up above the diagonal would be have Eigen values of 1, but it

would be badly conditioned. So the right way to take it is v transpose v. Look at v1 transpose v2 transpose, transpose. As always, if a matrix isn't symmetric, if the matrix v is not symmetric, good idea to form v transpose v. That is symmetric. It does have positive Eigen values. And those Eigen values, the Eigen values of v transpose v, are the singular values, or rather the singular values squared of v. So I guess I'm saying, you can't trust the Eigen values of v. It's the singular values you can trust. And the way to find singular values is form v transpose v. That give you a symmetric matrix. Its Eigen values, so the i'th Eigen value would be the i'th singular value squared.

So the condition number, is signma max over cignma min. And, well, it's not enormous for this 4 by 4 matrix, but if I go up to 10 by 10 or 100 by 100. 100 by 100 would just totally wipe me out. 10 by 10 would already be disasterious. Completely disastrous actually. 10 by 10 vandermonde matrix, with 1, 2 3, 4, up to 10 as the points it would have entries if that ended in a 10 and I had 10 rows would that be something like 10 to the ninth power. I mean the dynamics scale would be terrible. So finally just to v transpose v is called the gram matrix. So that guy, Gram is coming back in as giving the good measure. So the point was then, that this measures the dependence or independent of the vs. That ratio. The bigger that ratio is, the more dependency they are. What's the ratio if they're orthonormal? What's the ratio of the q's? what's the condition number of the qs, if we've arnoldi and got a good basis, it's one. What's the gram matrix? If this is q transpose q, with the qs the orthonormal basis in the columns. Then q transpose q is the identity. So a grand matrix. So q transpose q would be the identity, and it's condition number is the best possible one. Lamda max is lamda min is one. The condition number is one. OK. So that's just some comments about why arnoldi gets brought in to fix the situation. OK. So I'll just leave those 10 arnoldi steps on the board. The notes give 10 comments. I'm quite proud my MATLAB is unreliable. Actually you'll find a few errors like after end, I accidentally put a semicolon [INAUDIBLE] But the comments, I'm pleased with, and then the numerical example that runs through one cycle of this with these numbers to see what q2 is. Why do we have a short recurrence? This is a key point here. And in this example, if I work out the hs, I'll discover sure

enough h13 will be zero. h14 will be zero. Here's the key equation.

Here's arnoldi in matrix language. Let me see if I can remember arnoldi in matrix language. Yeah. So arnoldi in matrix language is going to be this. It's going to aq equals qh. I can't write out all of q. So that's the big equation. Its a very important equation that we now have all the pieces for. So a is our original matrix that we were given symmetric, let's say. q is our basis out of arnoldi and h is the multipliers that gave that basis. So this qh is a little bit like gram schmidt. Do you remember gram schmidt is described by q times r. q again is orthonormal. So its an orthogonal matrix. In Gram schmidt r is upper triangular. Here it's not.

Here it's hessenberg so h stands for hessenberg. I'll write down what the actual h is for these for these numbers. I won't write down the q. I'll just write down what h turned out to be for those numbers if I did it right. Five halves. Oh, interesting. The lengths all turned out to be five halves. And this turned out to be root 5 on 2. This turned out to be the square root of 4 over 5, and this turned out to be the square root of 9 over 20. And the point is from here, this one is just below the diagonal. And it will show up as symmetric. Root 5 over 2, root 4/5 and root 9/20. OK.

So, what am I seeing from that particular h which somehow can't be an accident? It must be that it's built in. It's the fact that h is symmetric and tridiagonal. And what does that tridiagonal tell us? It tells us that we have short recurrence. It's the 3 term recurrence relation is what I'm seeing here in matrix language, because there were 3 non zeros in the columns of h. All right. I was going to write out, I going to try to understand this one by looking at the first column. What if I take the first column of both sides? That'll be a times q1. q1 is the first column vector in the first basis vector.

And what do I have here when I take qs times hs? Do you do matrix multiplication a column at a time? You should. OK. So this says take 5 halves of the first column. And this says that factor times the second column. And I could track back and see, yes that's what arnoldi has produced. And then the second one, the next one, would be an aq2, the next would an aq3. Well, look, here it is. I want to show that h is

symmetric when a is. Could you do that? We know what the property of q is here. We know that q is, because arnoldi made it that way, as q transpose q equal i. And I can, in one step, show that h is symmetric if a is symmetric. How do you do that? I guess we need a formula for h, so I just multiply by q inverse. So that's good. And even better is to recognize what q inverse is. So what is q inverse here? Its q transpose. Anytime we see q, that's my letter. o is for an orthogonal matrix. So this is q transposed, aq. And now what? the argument's finished. We're here.

If a is symmetric, what can you say about that combination? If a is a symmetric matrix? It is symmetric. Right? That's how you get the symmetric matrixes. You start with one. You multiply on one side by a matrix, and on the other side by its transpose. The thing has to be symmetric, because if I transfer this whole thing, what will happen? To transpose things their transposes come in the opposite order. So q, it's transpose comes first. a, it's transpose comes in the middle, but what's that? The transpose of a is a. We're assuming a to be symmetric. And then the transpose of q transpose is q.

So we got this back again when we transpose so it's symmetric. So h is symmetric. So the conclusion is, h equals h transposed. And then we know immediately that its tridiagonal, because every h from arnoldi is hessenberg. We know that these zeros are here. The arnoldi cycle ended produced h i j in column j but it's stopped one below the diagonal. So we know these are zero, but now, if we know the matrix is symmetric then we know these are zero. And, of course, it works out that way.

So the conclusion is that we can orthogonalize the kylov basis, quickly, easily. And work with that basis either explicitly by computing it or by implicitly by keeping things orthogonal and that's what conjugate gradients will do.

So next time, I'm going to make an effort to describe the conjugate gradients method. I'll pick the highlights of it. It has fantastic properties, and to verify those properties in full detail is often more confusing then not. If you see today's lecture, you're seeing the important points; the role of symmetry of a and the arnoldi algorithm.

OK. so that's our first lecture on the krylov ideas. And next time, we'll probably complete that topic, and it will be on the web. So I'll see you Wednesday for the end of krylov. Thanks. Good.