NARRATOR: The following content provided by MIT OpenCourseWare under a Creative Commons license. Additional information about our license and MITOpenCourseWare in general is available at ocw.mit.edu.

PROFESSOR: So this is the second 086 lecture and it'll be my second and last lecture on solving ordinary differential equations -- u prime equal f of unt. So last time, we got Euler's method, both ordinary forward Euler, which is this method for the particular equation. So I'm using my model problem as always, u prime equal au. That's the model and I'm using small u for the true solution and capital u for the approximation. By whatever method we use, and right now, that method is forward Euler.

We looked at the idea of stability, but now let me follow up right away with the point of stability. Where does stability come in? So I want to do that for this simplest of all methods, because you'll see how we get to an error equation and we'll get to a similar error equation for much better methods. So the accuracy of Euler is minimal, but the question of how stability enters and how we show that it converges will be the same also in partial differential equations, so this is the place to see.

So I'll start with that, but then after that topic, will come better methods than Euler and the first step will be, of course, second order methods because Euler and backward Euler are only first order and we'll see -- actually, that'll be the point. Not only will show when they converge, why they converge, but also how fast they converge. What's the error? This estimate of the error is something that every code is going to somehow use internally to control the step size dela t.

So I'm just going to go with the model problem, u prime equal au, the Euler method. So here is the evaluation -- au is f there, so it's really delta t multiplying f, which is just au in this easy case. Here's the true solution and then here is the new point. When I plug the true solution in to the difference equation, it, of course, doesn't satisfy exactly. There's some error term -- this, I would call the discreditzation error -- making the problem discrete. So you could say the error at a single time step -- it's the new error that's introduced at time step m. Error now is going to be -- e will be u, the true one, minus the approximate, so I just subtract to get an equation. For the error at time step n plus 1, that subtraction gives ena delta t. That subtraction gives en and this is the inhomogeneous term, you could say, the new error.

So the question is -- first, the question is, what size is that, the local error? Then the key question is, when I combine and all these local errors, what's the global error? So first, the local error. You can see what this is, approximately, because the true solution would be e to the at and we all know that e to the at times u 0 -- e to the at -- there's a 1 plus an a -- from a single time step -- sorry, I should have said the true solution would grow by e to the a delta t over one step. It would multiply by e to the a delta t, but instead of multiplying by that exponential e to the a delta t, we're only multiplying by the first two terms in the series. So the error here is going to be something like a half delta t squared -- that's the key point where we see the local error -- times probably a squared -- we needed an a squared and u. So a squared un, that would really be the second derivative. I'll put it in as second derivative, because that's what it would look like in a nonlinear case. So I'm working the linear case for simplicity, but the point is that this is the Euler local error.

That's so typical, that you really want to look at that local error, see what it looks like. There is some constant and that has a certain effect on the quality of the method. There's a power of delta t and the question, what is that exponent has a big effect. So that exponent is the order plus 1. So for me, Euler's method has accuracy p equal 1. Over a single delta t step, the error is delta t squared, but we'll see that when I take 1 over delta t steps to get somewhere, to get to some fixed time like 1, then I have one over delta t of these things and the 2 comes back down to the 1, which I expect for Euler.

And finally, this term is out of our control basically. That's coming from the differential equation. That would tell us how hard or easy the method is, the equation is. So if u double prime is real big, we're going to expect to have to reduce delta t to keep this error under control, so that's a typical term -- a constant delta t to a power of p plus 1 and the p plus first derivative, the second derivative matching that to the solution. So that's -- if I now put this -- think of that as here as the sort of inhomogeneous term or the new source term.

I just want to estimate the end now. So I've done the local part and now I'm interested in putting it all together. How do I look at e -- so I really want somehow the solution -- e at some definite time n is what? I'm asking, really, for the solution to this system, and sort of in words -- so that's a good -- a totally typical error equation.

I think the way to look at it in words is, at each step, some new term, new error is committed, and what happens to that error at later steps? At later steps, that error committed then gets multiplied by 1 plus a delta t at the next step and 1 plus a delta t at the following step. So I think we have something like after n steps, we would have 1 plus a delta t to the n times the error that we did -- our first step error. So that's -- you see what's happening here? Whatever error we commit grows or decays according to this growth factor. Then of course, we made an error at step two and we made an error at step k, so at step k, it will have n minus k steps still to grow. This is de, the error de at step k and all the way through the -- I guess it would end with den. den would be the last thing that hadn't yet started to grow or decay.

Ok so that's our formula. It looks a little messy, but it shows everything. It shows the crucial important of stability and how it enters. Stability controls -- so there is the error that we made way at the beginning and then this is the growth factor that happened n time, so you see that if 1 plus a delta t has magnitude bigger than 1, what's going to happen? That will explode and the computer will very quickly show totally out of bounds output. But if it's stable, if this thing is less than 1, less or equal to 1, then that stability means that this error is not growing and so we get -- so let me put, if we have this stability -- 1 plus a delta t, smaller equal 1, then we get the bound that we want. So can you see what that looks like? We've got n terms, right? To get to point n, we've made n -- capital n errrors -- and then each term, that gives a 1. That's the critical role of stability and the error is of this order, so I'm getting something like a half delta t squared over 2 times -- let's say some maximum. I'll use u double prime for a maximum value, let's say, of the second derivative. That's a typical good satisfactory error estimate.

So what's up here? So n times delta t times 1 delta t is the time that we reached, right? We took n steps, delta t each step. So this is the same as -- this is what I'm wanting and it's going to fit on this board -- there's a one half t -- I should make it t over 2, right? n delta t is some time -- capital t. We have the half. We still have 1 delta t. That's the key. We have this u double prime, which is fixed. Anyway, we have first order convergence. So the error after n steps goes down like delta t. If I cut the time step in half, I cut the error. So that's not a great rate of convergence. That's, like, the minimal rate of convergence -- just 1 power delta t and that's why we call Euler a first order method.

I think, I hope you've seen and can kind of -- because we'll -- this expressions for the solution is messy but meaningful. That's a little bit of pure algebra or something that shows how stability is used, where it enters.

So now I'm ready if you are to move to second order methods. Now we're actually going to get methods that people would really use. So all those methods are, I think, worth looking at. Can I say that the notes that are on the web -- you may have spotted chapter five, section one, for example, which is this material -- so I'm -- after the lecture, I always think back, what did I say and improve those notes, so possibly later this afternoon a revised, improved version of this material will go up and then next week starts the real thing, what I think of as the real thing, the wave equation and the heat equation -- partial differential equation -- but it's worth -- here we saw how stability works.

Here we're going to see how accuracy can be increased and in different ways. In this way -- so what do I notice about the trapezoidal method? Sometimes it's called Crank-Nickolson. In PDs, the same idea was proposed by Crank and Nickolson. So what's their idea or what's the trapezoidal idea in the first place? Basically, we've centered -- we've recentered the method at what time position? You can go ahead. So where -- instead of Euler, which was, like, started at time n or tn or backward Euler that was sort of focused on the new time tn plus 1? This combination is at time n plus a half delta t, right? Somehow we've symmetrized everything around the half and the point is that around the midpoint, the accuracy jumps up to 2. Let me try always u prime equal au. So f is au. Then what does this become? Can I, as you would have to do in coding it, separate the implicit part? This is implicit, of course. That stands for this: fn plus 1 means f of the new unknown u at the new time. So it's implicit.

So let me bring that over to the left side, because it's sort of part of the unknown. I'll multiply through by delta t and I'll take f to be au. So I just want to rewrite that for our model problem. So it will be 1 minus a half when I bring that over minus a delta t over 2 un plus one. Is that what I get when I multiply through by delta t and bring the implicit part over and now I bring the explicit part to that side, so it's 1 plus a half delta t a delta tun. So that's what we actually solve at every step.

So first of all, I see that it's pretty centered. I see that there's something has to be inverted. That's because the method's implicit. If this was a matrix, a system of equations, which is very typical -- could be a large system of equations with the matrix, capital a, then u would be a vector and we would be solving this system: matrix times vector equals known right hand side from time n. But here it's a scale or it's just a division, so what is stability going to depend on? What does stability require? At every step, we don't want to grow. So if we take a step -- that's 1 plus a over 2 delta t divided by 1 minus a over 2 delta t, that's the growth factor, the decay factor that multiplies un to give you n plus 1, so the stability question is, is this smaller or equal to 1? That's stability for this trapezoidal method.

What's the story on that? Suppose a is very negative. That's what killed forward Euler, right? If a delta t -- that's where we look for instability. If a delta t becomes too negative, we lost stability in forward Euler and we'll lose stability in other, better methods too. But here, if a delta t is negative, that denominator is good. It's bigger than the numerator, right? So we have a -- you'll see it. If a delta t is less than 0, if it's very much less than 0, then this is approaching maybe minus 1, but it's below 1.

So this message is a, stable. Absolutely stable. Stable for all a less than 0. We're fine. Delta t can be in principle as large as we like. So we might ask, why don't I just jump in one step to the time that I want to reach and not bother taking a lot of small steps? What's the -- in that little paradox? It would be stable, but what's no good. So if I jumped in one giant step, this thing would be about minus 1 or something. So u here would be about minus 1 times u 0 and of course that's not the answer. So why do we still need a delta t? Because of the discretization error. Because this isn't the exact differential equation, so we still need -- we don't have problem with stability, but we still have to get the local errors down to whatever level we want.

Now what are those local errors for this guy? What are the local errors for that message? Let's see. Can we actually see it? I mean, you know what I'm thinking, right? I'm thinking that the local error is a border delta t cubed. That's a second order method for me. delta t cubed at each step. 1 over delta t steps, so delta t squared overall in that second order. But do we see why this is? I mean, instinctively, we say, it's centered. It should be, but let me expand this to see, are we -- is e to the a delta t -- how close is it to 1 -- so it's approximately 1 plus a over 2 delta t divided by 1 minus a half a delta t. I just want to see that sure enough, this is second order.

So of course I have a denominator here that I want to multiply. So that's 1 plus a delta t over 2, the explicit part and then everybody knows the 1 -- there are only two series to know, the truth is. We study infinite series a lot, but the two that everybody needs to know are the exponential series, which is for the left side and the geometric series for the right side, 1 over 1 minus x is 1 plus x plus x squared plus so on. So that's the -- that this came from the denominator, getting it up in the numerator where I can multiply the other term and get 1 plus a delta t, which is good and what's the coefficient of a squared delta t squared? Can you see what it is? If I do that multiplication, what does -- you see it? Where do I get a delta t squared from this? I get one here, times 1/4 and I get 1 here times 1/4, so that's two 1/4s is 1/2.

So it's great, right? It got the next term correct in the exponential series. so that's why it's one order higher; because it got the right number there, but it won't get the right number at the next step. What's the -- I'd have to find out what the cube -- this would be a delta t over 2 cubed, so what will it -- the I'll get a wrong coefficient for a delta t cubed. I'll get -- that would give me 1/8 and this would give me another 1/8, I think, so I think I'm getting two 1/8s out of that. I'm getting 1/4, which is wrong. What's right there? For the exponential series, it should be one over three factorial. It should be 1/6. So I got 1/4, not 1/6 and the error is the difference between them, which is 1/12. So 1/12 would be this number. The local error, so the local error, de, would be like the missing 1/12, the delta t to the third power now and the third derivative. That's what would come out if I patiently went through the estimate, just to see, because we'll -- in these weeks, we'll be creating difference methods for partial differential equations and we'll want to know, what's controlling the error? This is the kind of calculation, the beginning of a Taylor series that answers that question.

So the main point is, local error delta t cubed, stable for every a delta t. So our method would move from Euler to trapezoidal and it would give us a final error e -- sorry, small e. I can say it without writing it. The error, small e, would be delta t to -- what power? So squared, delta t squared. Good. The same kind of reasoning -- now I want to ask about -- so that takes care of trapezoidal, which is a pretty good method. I think it's -- in METLAB, it would be ode something small t for trapezoid. Forgot -- maybe 1, 2. I should know. It's used quite a bit.

Of course, we're going to get higher than second order, but second order is often in computational mathematics a good balance. Newton's method for solving nonlinear equation is second order and it doesn't pay to go to higher. You could invent a -- you could beat Newton by going, by inventing some third order method for solving a nonlinear equation, but in the end, Newton would be the favorite. So second order is pretty important. So let me point to -- so that was implicit, because it involved fn plus 1, but Adams had another idea, which seems like a great idea. Instead of an implicit using fn plus 1, he used the previous value, fn minus 1, which we already know. We've already at that previous step substituted un minus 1 n to f. That might have been time consuming, but we sure had to do it once and we only have to do it once with Adams. We're using something we already know. This is the one new time that we need it, that we have to compute f and we get explicitly the new u. These numbers, 3/2 and minus 1/2 were chosen to make it second order and the notes check that, that with a series that it comes out to give us the correct second term. Good.

Let me mention backward differences. Now we're just using 1 f and actually the implicit 1, so I'm making it implicit. So why am I interested in number 3 at all? Because it's going to have better stability. By being implicit and by choosing these numbers, 3/2, minus 1/2 and 1/2, I've upped the accuracy to 2, equal to 2 and I've maintained high stability by making it implicit. So this one competes with this one for importance, for use. So you're really seeing three pretty good methods that can be written down. So these numbers were chosen to get that extra accuracy, which we didn't get from backward Euler. When those numbers were 1 and minus 1, backward Euler was only first order. Now this goes up to second order and you can guess that we can that every term we allow ourselves, we can choose our coefficients well, so that the order goes up by one more. Now if you look at that, you might say, why not keep -- you could come back to this.

Why not combine the idea of backward differences, more left hand sides, with the Adams-Bashforth idea of more right hand sides? Suppose I come back to Adams-Bashforth and create an 1806 method? That'll be third order, but still will only go back one step. So it will somehow use some combination like this that goes back -- I'm not writing this method down, for a good reason, of course, but it will use something like this on the left side and something like this on the right side and that gives us enough freedom, enough coefficients to choose -- two coefficients there, three there and enough that we can get third order accuracy. In other words, the natural idea is use both un minus 1 and fn minus 1 in the formula to get that extra accuracy.

So why isn't that the most famous method of all? Why isn't that even on the board? You can guess. It's violently unstable, so sadly, the most accurate methods, which use both an old value of u and an old value of f of u, use both of this and this -- the numbers that show up here are bad news. It's a fact of life and so that's why we go backwards. We have to make a choice: We go backwards with u or Adams goes backwards with f or we think of something way to hype up the accuracy even further within one step method.

So the notes give a table, the beginning of a table. People could figure out the formulas for any order p, so the notes will give the beginning of a table for p equal 1, which is Euler, p equal 2, which is this. p equal 3 and p equal 4, which is Adams-Bashforth further back, backward differences further back and those tables show the constants. So they show the 1/12 or the 1/2 or whatever that is and they show the stability limit and of course, that's critical. The stability limit is much better for backward differences than it is for Adams-Bashforth.

So actually, I only learned recently that Adams-Bashforth, which we all teach, which all books teach, I should say, isn't that much used way back -- maybe the astronomers might use it or they might use backward differences, which are more stable. So backward differences has an important role and then one step methods will have an important role. Backward differences are implicit, so those are great for stiff -- you turn that way for stiff equations and for nonstiff equations, let me show you what the workhorse method is in a moment. So I have a number 4 here, whose name I better put up here.

Let me put up the name of method 4, which is two names: Runge-Kutta. So that's a sort of one compound step. So what do I mean by -- what's the point of -- this looked pretty efficient, but there are two aspects that we didn't really, I didn't really mention and on those two aspects, Runge-Kutta wins by being just a single self-contained step.

So what are the drawbacks of a multistep method like this or like this? You have to store the old value, no problem, but at the beginning, at t equals 0, what's the problem? You don't know the old value, so you have to get started somehow with some separate -- you can't use this at t equals 0, because it's calling for a value at minus delta t and you haven't got it. So you need -- it takes a separate method. You can deal with that, create -- use Runge-Kutta, for example, to get that first step. But then there's another problem with these backwards step methods, which again, can be resolved. So this is a live method. The other problem is, if you want to change delta t -- suppose you want to cut delta t in half, then this is looking for a value that's a half delta t back in time and you haven't got that either, but you've got values 1 delta t back and 2 delta t back and some interpolation process will produce that half value. So what am I saying? I'm just saying that getting started takes a special attention, changing delta t takes a little special attention, but still, it's all doable.

So that this method, which I gave a name to last time -- I've forgotten what it was. I think it was something like ode 15 s for stiff is much used for stiff equations. So now I'm left -- let me not only write down Runge-Kutta's name, but the other -- so multistep I've spoken about. I just -- this is -- I'm just going to write this to remind myself to say one word about it -- dae. Can I say one word even now? I'm sorry. Then I'll say one word later. So that's a differential algebraic equation. That's like a situation which comes up in chemical engineering and many other places, in which out of our n given equations, some of them are differential equations, but others are ordinary algebra equations, so there's no d by dt in them. Nevertheless, we have to solve and there is a d by dt in the others. So this is a whole little world of daes, which you may never meet. I have never personally met, but it has to be dealt with and there are codes -- I'll just mention the code dasil by [? pencil ?] I'll finish saying the one word. A model problem here would be some matrix times u prime equal f of u and t, so a sort of implicit differential equation, because I'm not giving u prime, I'm only giving mu prime. If m is singular -- otherwise I could multiply by m inverse and make it totally normal, but if m and sometimes t or m may depend on u and t, it could go singular. If it goes singular, then this system is degenerating and we have to think what to do. this dasil code would be one of the good ones that does. So that's sort of like my memory to myself to say something about the daes before completing this topic of ordinary differential equations.

Now for Runge-Kutta. Let me take p equal to 2 first. The famous Runge-Kutta is the one at p with forth order accuracy. That will take more of the little internal compounded steps than p equal to 2 takes, but p equal to 2 makes the point. So this'll be simplified Runge-Kutta -- p equal to 2. It'll be un plus 1 minus un. You see, it's just one step, but what goes here? Do I remember? The notes will tell me. I better -- since it's going to be kept forever, I better get it right. It involves fn and f of f, so Runge-Kutta, here we go.

It has a half of fn, the usual Euler figure out this slope. But then if the other half, is the same f at -- I take Euler, so doing this part allowed me to taken a forward Euler and now I put that forward Euler step in as like a un plus 1 delta tfn at time -- what's the right time? Probably -- that's sort of -- this is like un plus 1, but we didn't have to do -- we're not implicit and so I presume that that's a time n plus 1.

So I've written in one line what the code would have to write in two lines. The first line of code would be compute f at the old un, then take an Euler step, multiply it by delta t and add to un, so this give something that's like un plus 1, but it didn't cost us anything. So can you compare that with trapezoidal? You see the comparison with trapezoidal is trapezoidal has the 1/2 fn the same, but it has the 1/2 f n plus 1 at the new step involving the new un plus 1 where this 1 -- here is something like it, but something we already knew from Euler. So that's the natural idea. All these ideas actually would occur to all of us if we started thinking about these for a few weeks. B I guess I'm not planning to spend a few weeks on that, so I'm just jumping ahead to what people have thought of.

So that's, you could say, two step Euler. There's two evaluations of f within a step. So what's p equal 4? The famous Runge-Kutta is four of those. So you take -- there's an Euler step. That'll be step one. Then you'll stick that into something, that's step n plus 1/2. That'll be a second evaluation of f. That'll give you some output. You plug that in and that will actually, if it's correctly chosen, be another, better approximation of 1/2, at n plus 1/2. You put that into the fourth one, which will give you something close to un plus 1 and then you take the right weights. Here they were 1/2 and 1/2. Anyway, you can do it.

The algebra begins to get messy beyond p equal 4, but you can go beyond p equal 4. I mean, there's books on Runge-Kutta methods because the algebra gets -- of these f of f of f. If there's anything that -- any construction in math that looks so easy, just take f of f of f of f of a function, of a number. That seems so simple, but actually it's extremely hard to understand what happens. Maybe you know that

Mandelbrot's fractal sets and all these problems of chaos come exactly that way and they're extremely hard to see what's happening. When you do repeated composition, you would call that f of f of f.

So Runge-Kutta stops at 4 and then there is actually an implicit Runge-Kutta that people work on, but I don't dare to begin to write that down and I won't even write the details of that one, which are in our notes and in every set of notes. But what's the point? First, we get the accuracy, so what's the other thing you have to ask about? Stability. So what does our stability calculation give for Runge-Kutta? The thing that you have to work with, actually, turns out to be quite nice. It's just the series for e to the at chopped off at the power, however far you're going. So this is Runge-Kutta stability, p equal to 2. The growth factor will be from -- if I just do this and I take the model problem, f equal au, you could -- if I took that out of there, the growth factor will be 1 plus a delta t plus 1/2 a delta t squared. Stop. That's what this would give for the model problem as the growth factor that multiplies each u to get the next u. What p equal 4 would give would be the same thing plus -- it's just the right -- it's just the exponential series chopped off at the fourth term.

So of course, you see immediately, what's the discreditization error? It's still to t of the fifth. It's the first missing term that Runge-Kutta didn't capture, multiplied by 1 over 5 factorials, so you know right away that the constant is 1 over 120, which is pretty good. But the question is the stability. So what do we want to know here? We want to know -- so that means that a delta tb sum -- because that same number is coming up. Let me call it z, right? So the stability test for p equal to 2 is to look at the magnitude of 1 plus z plus 1/2 z squared and we want it to be less than or equal to 1. Now, the point is, this is something -- I hope you try this this weekend. Somehow get MATLAB to plot in the complex plane, so you really -- up there now we've just -- at some point z, some negative value of z -- it's going to hit 1 and then after that, it's going to be unstable. Of course, there's got to be a point where it's unstable. This is an explicit message. Runge-Kutta is explicit. So it can't -- if the problem is too stiff, we're going to be killed, but if it's an ordinary problem, this is a winner. So p equal 4, this is the code ode 45, which uses -- combines Runge-Kutta with 4 on a higher order formula to get an estimate for the error in that step. So I could make some comment on predictor, corrector methods, but I'll leave that in the notes.

So what does that region look like? Then the other one is, what does the region 1 plus the z plus 1/2 c squared plus 1/6 c cubed plus 1/24 v to the fourth, less or equal 1 -- what's that region look like? I mean, if those regions are small, the method loses. If those regions are bigger than we get from Adams-Bashforth or backward differences, then the methods in there are good, successful. I've forgotten exactly what the picture is like there.

Why am I in the complex plane? Because the number a -- z is a delta t. That number a can be complex. How can it be complex? I'm not really thinking that our model equation u prime equal au will be complex, but our model system would be u prime equals a matrix times u and then it's the Eigen values of that matrix that take the place of little a. So we have n different little as going for a system and the Eigen values of a totally real matrix can be nonreal, can be complex numbers. Let's just think when that happens and then I'll -- what does the figure looks like for p equals 4? I sort of remember that the figure even captures a little bit of that plane and then it goes -- why should I mess up the board with -- it's a pretty good sized region. It's probably not shaped like that at all, but the point is, maybe this reaches out to about e, minus e. I think it does, somewhere around -- I don't know if that's an accurate -- right, reaches out around minus -- and reaches up to around 2.

So in this last second, I was just going to say -- and we'll see it plenty of times -- where do we get complex -- where do we get nice, real, big systems, stiff systems with complex Eigen values in real calculations? By real by real calculations, I really mean pdes. So in a partial differential equation -- so this is just a throw away comment that we'll deal with again -- in partial differential equations, a uxx term or a uyy term that get multiply typically by diffusion, those are diffusion terms, those produce symmetric things, symmetric negative definitely, but convection, advection terms, velocity terms that get multiplied by some -- that are first derivatives -- those produce -- those are antisymmetric. Those produce these complex Eigen values that we have to deal with.

So everybody wants to solve convection diffusion. Convection has these terms, diffusion has these. The coefficient may be very small. The hard -- that's the hard case, when the convection is serious. It's pushing us up the imaginary axis, where this is bringing us to the left and we're going to spend time thinking of good methods for that.

So my homework -- would you like figure out this region and make a better pictures? Get MATLAB to do it and try any one of these methods on the model problem. Try a method or two. See whether the stability, the theoretical stability limit is serious. I mean, does the limit of a delta t -- can we -- as I'm driving to MIT, I sometimes exceed the speed limit, the stability limit. Is that OK or not with finite differences?

I'll see you Monday. Alright, thanks. Have a good weekend.