PROFESSOR: Software that implements modern numerical methods has two features that aren't present in codes like ODE4 and classical Runge-Kutta. The methods in the software can estimate error and provide automatic step size control. You don't specify the step size h. You specify an accuracy you want. And the methods estimate the errors as they go along and adjust the step size accordingly. And they provide a fully accurate continuous interpolant. They don't just provide the solution at the discrete set of points. They provide a function that defines the solution everywhere in the interval. And so you can plot it, find zeroes of the function, provide a facility called event handling, and so on.

Larry Shampine is an authority on the numerical solution of ordinary differential equations. He is the principal author of this textbook about solving ODEs with MATLAB. He's a, now, emeritus professor at the Southern Methodist University in Dallas. And he's been a long time consultant to the MathWorks about the development of our ODE Suite. Shampine and his student, Przemyslaw Bogacki, published this method in 1989. And it's the basis for ODE23, the first of the methods we will use out of the MATLAB ODE Suite.

The basic method is order three. And the error estimate is based on the difference between the order three method and then the underlying order two method. There are four slopes involved.

The first one is the value of the function at the start of the interval. But that's based on something called FSAL, first same as last, where that slope is most likely left over from the previous step. If the previous step was successful, this function value is the same as the last function value from the previous step.

That slope is used to step into the middle of the interval, function is evaluated there. That slope is used to step 3/4 of the way across the interval and a third slope obtained there. Then these three values are used to take the step. yn plus 1 is a linear combination of these three function values. Then the function is evaluated to get a fourth slope at the end of the interval. And then, these four slopes are used to estimate the error.

The error estimate here is the difference between yn plus 1 and another estimate of the solution that's obtained from a second order method that we don't actually evaluate. We just need the difference between that method and yn plus 1 to estimate the error.

This estimated error is compared with a user-supplied tolerance. If the estimated error is less than a tolerance, then the step is successful. And this fourth slope, s4, becomes the s1 of the next step.

If the answer is bigger than the tolerance, then the error could be the basis for adjusting the step size. In either case, the error estimate is the basis for adjusting the step size for the next step. This is the Bogacki-Shampine Order 3(2) Method that's the basis for ODE 23.

Let's look at some very simple uses of ODE23 just to get started. I'm going to take the differential equation y prime is equal to y. So I'm going to compute e to the t. And just call ODE23 on the interval from 0 to 1, with initial value 1. No output arguments. If I call it ODE23, it just plots the solution. Here it is. It just produces a plot. It picks a step size, goes from 0 to 1, and here it gets the final value of e-- 2.7 something.

If I do supply output arguments. I say t comma y equals ODE23, it comes back with values of t and y. ODE23 picks the values of t it wants. This is a trivial problem. It ends up picking a step size of 0.1. After it gets started, it chooses an initial step size of .08 for whatever error tolerances. And the final value of y is 2.718, which is the value of e.

So these are the two simple uses of ODE23. If you don't supply any output arguments, it draws a graph. If you do supply output arguments, t and y, it comes back with the values of t and y choosing the values of t to meet the error. The default error tolerances is 10 to the minus 3. So this value is going to be accurate to three digits. And sure enough that's what we got.

Now let's try something a little more challenging to see the automatic error-controlled step size choice in action. Set a equal to a quarter. And then set y0 equal to 15.9. If I would set it to 16, which is 1 over a squared, I'd run into a singularity. Now the differential equation is y prime is equal to 2 (a minus t) times y squared. I'm going to integrate this with the ODE23 on the interval from 0 to 1 starting at y0, and saving the results in t and y, and then plotting them. So here's my plot command, and there is the solution.

So there is a near singularity at a. It nearly blows up. And then it settles back down. So the points are bunched together as you go up to the singularity and come back down, but then get farther apart as the solution settles down. And the ODE solver is able to take bigger steps.

To see what steps were actually taken, let's compute the difference of t, and then plot that. So

here are the step sizes that were taken. And we see that a small step size was taken near the almost singularity at that 0.25. And then as we get towards the end of the interval, a larger step size is taken. And then, finally, the step size just to reach the end of the interval is taken as the last step. So that's the automatic step size choice of ODE23.

BS23 has a nice natural interpolant that goes along with it that's actually been known for over 100 years. It's called Hermite Cubic Interpolation. We know that two points determine a straight line. Well, two points and two slopes determine a cubic.

On each interval we have the values of y and yn plus 1. We also have two slopes, namely this. We have the derivatives at the end points, yn prime and yn plus 1 prime, that's the values of the differential equation at those points. So those four values determine a cubic that goes through those two points and has those two slopes.

This cubic allows the software to evaluate the solution at any point in the interval without additional cost as defined by addition evaluations of the function f. This can be used to draw graphs of the solution, nice smooth graphs of the solution, find zeroes of the solution, do event handling, and so on. Another feature provided by ODE23.