INSTRUCTOR: I want to illustrate the important notion of stiffness by running ode45, the primary MATLAB ODE solver, on our flame example. The differential equation is y prime is y squared minus y cubed, and I'm going to choose a fairly-- an extremely small initial condition, 10 to the minus sixth. The final value of t is 2 over y naught, and I'm going to impose a modest accuracy requirement, 10 to the minus fifth.

Now let's run ode45 with its default output. Now, see it's taking-- it's moving very slowly here. It's taking lots of steps. So I'm take- pressing the stop button here. It's working very hard. Let's zoom in and see why it's taking so many steps, very densely packed steps here.

This is stiffness. It's satisfying the accuracy requirements we imposed. All these steps are within 10 to the minus sixth of one, but it's taken very small steps to do it. These steps are so small that the graphics can't even discern the step size.

This is stiffness. It's an efficiency issue. It's doing what we asked for. It's meeting the accuracy requirements, but it's having to take very small steps to do it.

Let's try another ODE solver-- ode23. Just change this to 23 and see what it does. It's also taking very small steps for the same reason. If we zoom in on here, we'll see the same kind of behavior. But it's taking very small steps in order to achieve the desired accuracy.

Now let me introduce a new solver, ode23s. The s for stiffness. This was designed to solve stiff problems. And boom, it goes up, turns the corner, and it takes just a few steps to get to the final result. There it turns the corner very quickly.

We'll see how ode23s works in a minute, but first let's try to define stiffness. It's a qualitative notion that doesn't have a precise mathematical definition. It depends upon the problem, but also on the solver and the accuracy requirements.

But it's an important notion. We say that a problem is stiff if the solution being sought very slowly, but there are nearby solutions that very rapidly. So the numerical method must take small steps to obtain satisfactory results.

Stiff methods for ordinary differential equations must be implicit. They must involve formulas that involve looking backward from the forward timestep. The prototype of these methods is the backward Euler method, or the implicit Euler method.

This formula, it involves-- defines y n plus 1, but doesn't tell us how to compute it. We have to solve this equation for y n plus 1. And I'm not going to go into detail about how we actually do it. It involves something like a Newton method that would-- requires knowing the derivative, or an approximation to the derivative of f. But this gives you an idea of what you can expect in stiff methods.

I like to make an analogy with taking a hike in one of the slot canyons we have here in the Southwest. Explicit methods like ode23 and 45 take steps on the walls of the canyon and go back and forth across the sides of the canyon, make very slow progress down the canyon. Whereas implicit methods, like ode15s, look ahead down the canyon and look ahead to where you want to go and make rapid progress of the canyon.

The stiff solver, ode23s, uses an implicit second-order formula and an associated third-order error estimator. It evaluates the partial derivatives of f with respect to both t and f at each step, so that's expensive. It's efficient at crude error tolerances, like graphic accuracy. And it has relatively low overhead.

By way of comparison, the stiff solver ode15s, can be configured to use either the variable order numerical differentiation formula, NDF, or the related to backward differentiation formula BDF. Neither case it saves several values of the function over previous steps. The order varies automatically between one and five, it evaluates the partial derivatives less frequently, and did see efficient at higher tolerances then 23s.