ODE45

The most frequently used ODE solver in MATLAB and Simulink is ODE45. It's based on method published by British mathematicians JR Dormand and PJ Prince in 1980.

The basic method is order 5. The error correction uses a companion order four method.

$$s_1 = FSAL$$
$$s_2, \cdots, s_6 = f(t,y) \; at \; t_n + \frac{1}{5}h, \frac{3}{10}h, \frac{4}{5}h, \frac{8}{9}h, h$$
$$y_{n+1} = y_n + \cdots$$
$$s_7 = f(t_{n+1}, y_{n+1})$$
$$e_{n+1} = \cdots$$

The first slope of t_n is, first same as last left over from the previous successful step. Then there are five more slopes from function values at $\frac{1}{5}h$, $\frac{3}{10}h$, $\frac{4}{5}h$, $\frac{8}{9}h$, h. These six slopes, linear combinations of them are used to produce y_{n+1}. Then function is evaluated at t_{n+1} and y_{n+1} to get a seventh slope. And then linear combinations of these are used to produce the error estimate. Again, if the error estimate is less than the specified accuracy requirements the step is successful. And then that error estimate is used to get the next step size. If the error is too big, the step is unsuccessful and that error estimate is used to get the step size to do the step over again.

If we want to see the actual coefficients that are used, you can go into the code for ODE45.

```
>> dbtype 201:213 ode45
```

There is a table with the coefficients. Or you can go to the Wikipedia page for the information.

As an aside, here is an interesting fact about higher order Runge-Kutta methods. Classical Runge-Kutta required four function evaluations per step to get order four. Dormand-Prince requires six function evaluations per step to get order five. You cannot get order five with just five function evaluations. And then, if we were to try and achieve higher order, it would take even more function evaluations per step.

Let's use ODE45 to compute e^t

```
>> F = @(t,y) y
```

We can ask for output by supplying an argument called **tspan**.

```
tspan = (0:0.1:1)'
```

Zero and steps of 0.1 to 1. If we supply that as the input argument to solve this differential equation and get output at those points.

```
>> [t,y] = ode45(F,tspan,1)
```

We get that back as the output. And now here's the approximations to the solution to that differential equation at those points.

If we plot it, here's the solution at those points.

```
>> plot(t,y,'o-')
```

And to see how accurate it is, we see that we are actually getting this result to nine digits.

```
>> format short e
   exp(t)-y
```

ODE45 is very accurate.

Let's look at step size choice on our problem with near singularity is a quarter.

```
>> a = 0.25
   y0 = 15.9
   F = @(t,y) 2*(a-t)*y^2
   [t,y] = ode45(F,[0,1],y0)
   plot(t,y,'-',t,y,'.','markersize',24)
```

You can see a lot of points here, but this is misleading because ODE45, by default, is using the refine option. It is only actually evaluating the function at every fourth one of these points and then using the interpolant to fill in in between. So we actually need a little different plot here.