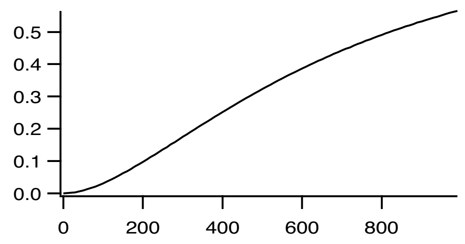
```
Display ChemKin[][%D]    // graph concentration of the product
                         // Note graph made with subrange of wave
IntegrateODE/M=1 ChemKinetic, KK, ChemKin
```



Note that the waves yw and dydt in the derivative function have four elements corresponding to the four equations in the system of ODEs. At a given value of X (or t) yw[0] and dydt[0] correspond to the first equation, yw[1] and dydt[1] to the second, etc.

Note also that we have used the /M=1 flag to request the Bulirsch-Stoer integration method. For well-behaved systems, it is likely to be the fastest method, taking the largest steps in the solution.

Optimizing the Derivative Function

The Igor compiler does no optimization of your code. Because IntegrateODE may call your function thousands (or millions!) of times, efficient code can significantly reduce the time it takes to calculate the solution. For instance, the `ChemKinetic` example function above was written to parallel the chemical equations to make the example clearer. There are three terms that appear multiple times. As written, these terms are calculated again from scratch each time they are encountered. You can save some computation time by pre-calculating these terms as in the following example:

```
Function ChemKinetic2(pw, tt, yw, dydt)
   Wave pw        // pw[0] = k1, pw[1] = k2, pw[2] = k3
   Variable tt    // time value at which to calculate derivatives
   Wave yw        // yw[0]-yw[3] containing concentrations of A,B,C,D
   Wave dydt      // wave to receive dA/dt, dB/dt etc. (output)

   // Calculate common subexpressions
   Variable t1mt2 = pw[0]*yw[0]*yw[1] - pw[1]*yw[2]
   Variable t3 = pw[2]*yw[2]

   dydt[0] =  -t1mt2
   dydt[1] = dydt[0]          // first two equations are the same
   dydt[2] = t1mt2 - t3
   dydt[3] = t3

   return 0
End
```

These changes reduced the time to compute the solution by about 13 per cent. Your mileage may vary. Larger functions with subexpression repeated many times are prime candidates for this kind of optimization.

Note also that IntegrateODE updates the display every time 10 result values are calculated. Screen updates can be very time-consuming, so IntegrateODE provides the /U flag to control how often the screen is updated. For timing this example we used /U=1000000 which effectively turned off screen updating.

Higher Order Equations

Not all differential equations (in fact, not many) are expressed as systems of coupled first-order equations, but IntegrateODE can only handle such systems. Fortunately, it is always possible to make substitutions to transform an Nth-order differential equation into N first-order coupled equations.

You need one equation for each order. Here is the equation for a forced, damped harmonic oscillator (using y for the displacement rather than x to avoid confusion with the independent variable, which is t in this case):

$$\frac{d^2y}{dt^2} + 2\lambda\frac{dy}{dt} + \omega^2 y = F(t)$$

If we define a new variable *v* (which happens to be velocity in this case):

$$v = \frac{dy}{dt}$$

Then

$$\frac{d^2y}{dt^2} = \frac{dv}{dt}$$

Substituting into the original equation gives us two coupled first-order equations:

$$\frac{dv}{dt} = -2\lambda v - \omega^2 y + F(t)$$

$$\frac{dy}{dt} = v$$

Of course, a real implementation of these equations will have to provide something for F(t). A derivative function to implement these equations might look like this:

```
Function Harmonic(pw, tt, yy, dydt)
   Wave pw     // pw[0]=damping, pw[1]=undamped frequency
               // pw[2]=Forcing amplitude, pw[3]=Forcing frequency
   Variable tt
   Wave yy     // yy[0] = velocity, yy[1] = displacement
   Wave dydt

   // simple sinusoidal forcing
   Variable Force = pw[2]*sin(pw[3]*tt)

   dydt[0] = -2*pw[0]*yy[0] - pw[1]*pw[1]*yy[1]+Force
   dydt[1] = yy[0]

   return 0
End
```

And the commands to integrate the equations:

```
Make/D/O/N=(300,2) HarmonicOsc
SetDimLabel 1,0,Velocity,HarmonicOsc
SetDimLabel 1,1,Displacement,HarmonicOsc
HarmonicOsc[0][%Velocity] = 5          // initial velocity
HarmonicOsc[0][%Displacement] = 0      // initial displacement
Make/D/O HarmPW={.01,.5,.1,.45}        // damping, freq, forcing amp and freq
Display HarmonicOsc[][%Displacement]
IntegrateODE Harmonic, HarmPW, HarmonicOsc
```