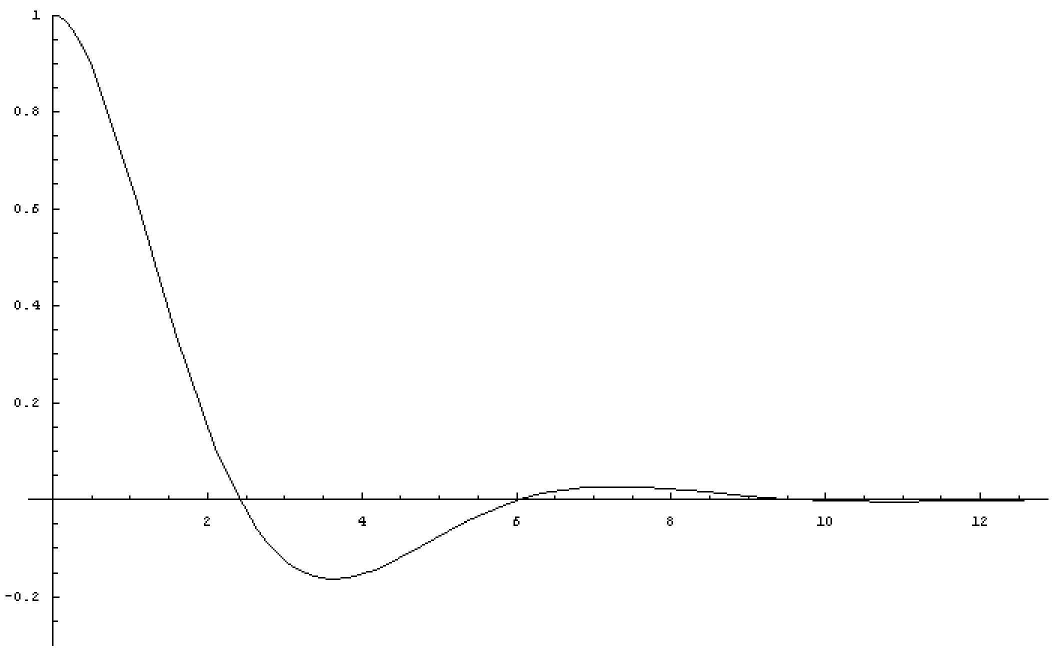
Mathematica (14)

One of Mathematica's great strengths is its ability to numerically solve differential equations. The first example illustrates a plot of a numerical solution of a damped, harmonic oscillator. The built in function NDSolve requires three arguments, the differential equation to be solve where the highest order derivative is solved in terms of lower order derivatives and functions, the dependent variable, in this case y, and the range of values of the independent variable x.

```
In[1]:= Clear[sol];
        sol = NDSolve[{y''[x] == -y[x] - y'[x], y'[0] == 0, y[0] == 1}, y, {x, 0, 4 Pi}]
        Plot[Evaluate[y[x] /. sol], {x, 0, 4 Pi}, PlotRange → {-0.3, 1}]

Out[2]= {{y → InterpolatingFunction[{{0., 12.5664}}, <>]}}
```



The first command simply clears the variable "sol". Then this variable is defined as the numerical solution of the differential equation y" = -y – y' with y'(0) = 0 and y(0) = 1. It can be plotted by evaluating y(x), where y(x) is replaced everywhere by its numerical solution defined by sol as x ranges over the interval 0 to 4π. (The symbol /. means replace y(x) by its numerical solution defined by sol.) Note that the y axis is defined by PlotRange to vary between –0.3 to 1.0.

We can also numerically solve coupled differential equations for which analytical solutions cannot be found. Consider the motion of a projectile in a gravitational field subject to quadratic air friction. The vector equation which govern the motion of this projectile is

$$\mathbf{a} = -g\,\mathbf{k} - c\,|\,v\,|\,\mathbf{v}$$

where a is the vector acceleration of the object, g is the acceleration of gravity (9.8) and c is the drag coefficient (0.0085) where the units are all mks. Thus we can write

$$z'' = -g - c\,(z'^2 + x'^2)^{1/2}\, z'$$

$$x'' = - c\,(z'^2 + x'^2)^{1/2}\, x'$$

If the projectile starts from the point (0,0) at t = 0 and has a velocity of 180 mi/hr at an angle of 30^{o} above the horizontal, we have z(0) = 0, x(0) = 0, z'(0) = 40.23 m/sec, and x'(0) = 69.68 m/sec for the initial conditions. We can then use Mathematica to find the trajectory as follows:

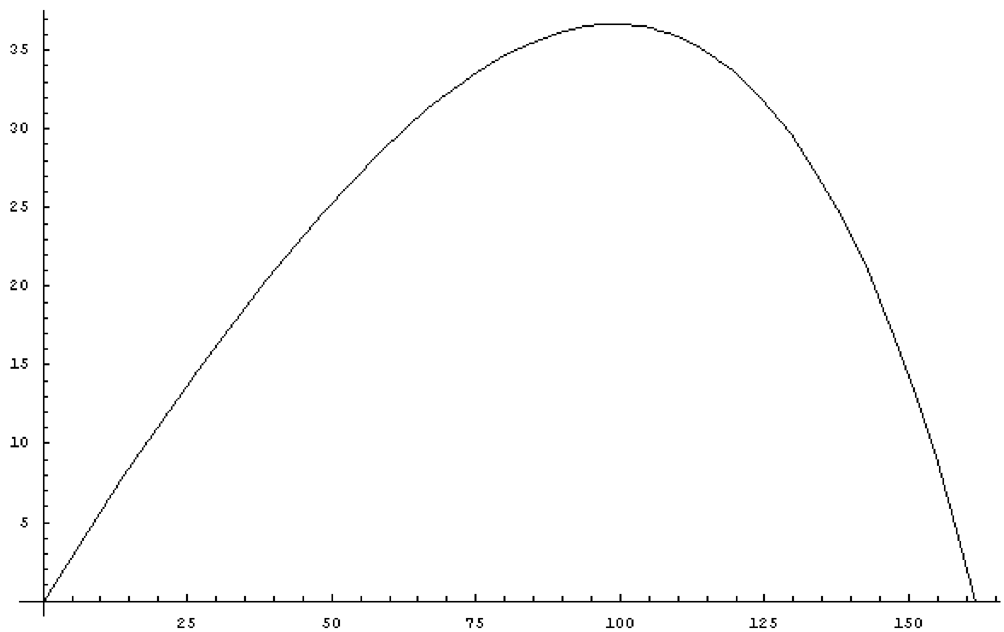
```
In[1]:= sol = NDSolve[{z''[t] == -9.8 - 0.0085 ((z'[t])^2 + (x'[t])^2)^(1/2) z'[t],
          x''[t] == -0.0085 ((z'[t])^2 + (x'[t])^2)^(1/2) x'[t], z[0] == 0, z'[0] == 40.23,
          x[0] == 0, x'[0] == 69.68}, {z, x}, {t, 0, 9.0}]

Out[1]= {{z → InterpolatingFunction[{{0., 9.}}, <>], x → InterpolatingFunction[{{0., 9.}}, <>]}}

In[2]:= t2 = FindRoot[Evaluate[z[t] == 0 /. sol], {t, 4.0, 7.0}]

Out[2]= {t → 5.36036}

In[3]:= ParametricPlot[Evaluate[{x[t], z[t]} /. sol], {t, 0, 5.36036}]
```



Note that the equations and their initial conditions are given to NDSolve as a list, the two dependent variables z and x are given as a list as the second argument, and the range of t is given as a list as the third argument. The time the projectile hits the earth is found by using FindRoot on the numerical solution of z given by sol and evaluated as t ranges from 4.0 to 7.0. The trajectory is then plotted using ParametricPlot on the values of x and z given by sol and evaluated as t ranges from 0 to the time the projectile returns to z=0 at t=5.36036 seconds.