ME 163 Using Mathematica to Solve First-Order Systems of Differential Equations

```
In[1]:= Off[General::spell1];
```

```
In[2]:= Off[General::spell];
```

In this notebook, we use *Mathematica* to solve systems of first-order equations, both analytically and numerically. We use DSolve to find analytical solutions and NDSolve to find numerical solutions. In each case, we will solve an initial value problem and plot the results. Because the phase plane has not yet been covered in class, the plots will be limited to plots of the dependent variables as functions of the independent variable.

■ ANALYTICAL SOLUTIONS

■ Constructing and Plotting a Single Solution

■ Example 1

For our first example, we take the system given by

$$\dot{x} = y\,,$$
$$\text{and}\quad \dot{y} = 6x - y\quad ,$$

with initial conditions x(0) = 1 and y(0) = -2. We define the equation, including these initial conditions:

```
In[3]:= equation1 := {x'[t] == y[t], y'[t] == - y[t] + 6 x[t], x[0] == 1, y[0] == -2};
```

Now we solve this equation analytically using DSolve.

```
In[4]:= DSolve[equation1, {x[t], y[t]}, t]
```

$$\text{Out[4]= } \left\{\left\{x[t] \to \frac{1}{5} e^{-3t} (4 + e^{5t}),\ y[t] \to \frac{2}{5} e^{-3t} (-6 + e^{5t})\right\}\right\}$$

We see that, as usual, DSolve has given us replacement rules for answers, along with an idiosyncratic way of expressing the sum of two exponentials. Let's repeat the calculation and assign the answer to the functions xans1[t] and yans1[t]. We also use ExpandAll to get *Mathematica* to use a better form for the answer.

```
In[5]:= {xans1[t_], yans1[t_]} =
         ExpandAll[{x[t], y[t]} /. Flatten[DSolve[equation1, {x[t], y[t]}, t]]]
```

Out[5]= $\left\{\frac{4\,e^{-3t}}{5} + \frac{e^{2t}}{5},\ -\frac{12}{5}\,e^{-3t} + \frac{2\,e^{2t}}{5}\right\}$

Let's check to see that xans1 and yans1 have been assigned the two components of the solution:

```
In[6]:= xans1[t]
```

Out[6]= $\frac{4\,e^{-3t}}{5} + \frac{e^{2t}}{5}$

```
In[7]:= yans1[t]
```

Out[7]= $-\frac{12}{5}\,e^{-3t} + \frac{2\,e^{2t}}{5}$

We can plot either component or both on the same graph. We construct the plots for *t* running from 0 to 2.

```
In[8]:= graphx1 = Plot[xans1[t], {t, 0, 2}, AxesLabel -> {"t", "x"}, PlotLabel -> "Equation 1"];
```

```
In[9]:= graphy1 = Plot[yans1[t], {t, 0, 2}, AxesLabel -> {"t", "y"}, PlotLabel -> "Equation 1"];
```

```
In[10]:= graphxy1 = Plot[{xans1[t], yans1[t]},
            {t, 0, 2}, AxesLabel -> {"t", "x, y"}, PlotLabel -> "Equation 1"];
```

■ Example 2

For our second example, we look at a damped free oscillator. The second order form of the equation is

$$m\ddot{x} + b\dot{x} + kx = 0\ .$$

We convert this to a first-order system by introducing $v = \dot{x}$. The result is

$$\dot{x} = v\ , \text{ and } \dot{v} = -(k/m)\,x - (b/m)\,v\ .$$

We define this equation for *Mathematica* in the special case when the initial displacement is 1 m and the initial velocity is - 2 m/s.

```
In[11]:= equation2[m_, b_, k_] :=
          {x'[t] == v[t], v'[t] == -(k/m) x[t] - (b/m) v[t] , x[0] == 1, v[0] == -2}
```

Now we ask DSolve to solve this, for the following parameter values: m = 1 kg, k = 101 n/m, and b = 2 N s/m. We assign the solution to xans2[t] and vans2[t].

```
In[12]:= {xans2[t_], vans2[t_]} =
          {x[t], v[t]} /. Flatten[DSolve[equation2[1, 2, 101], {x[t], v[t]}, t]]
```

Out[12]= $\left\{\frac{1}{20}\,e^{(-1-10\,i)\,t}\left((10-i) + (10+i)\,e^{20\,i\,t}\right),\ \frac{1}{20}\,i\,e^{(-1-10\,i)\,t}\left((-99+20\,i) + (99+20\,i)\,e^{20\,i\,t}\right)\right\}$

If this were an exam, I would subtract 5 points from DSolve's score for giving an answer to a real problem containing i's. The answer is correct, but has not been properly simplifed. We can plot it anyway.

```
In[13]:= graphx2 = Plot[xans2[t], {t, 0, 2},
            AxesLabel -> {"t (s)", "x (m)"}, PlotLabel -> "Damped Oscillator"];
```

Let's also solve this the old way -- treating it as a second order equation.

```
In[14]:= xans2mod[t_] =
          x[t] /. Flatten[DSolve[{x''[t] + 2 x'[t] + 101 x[t] == 0, x[0] == 1, x'[0] == -2}, x[t], t]]
```

Out[14]= $e^{-t}\left(\text{Cos}[10\,t] - \frac{1}{10}\,\text{Sin}[10\,t]\right)$

Interesting that we got a much better and simpler form of the same solution. We plot this solution, using dashing, and then we will combine the two plots.

```
In[15]:= graphx2mod = Plot[xans2mod[t], {t, 0, 2}, AxesLabel -> {"t (s)", "x (m)"},
            PlotLabel -> "Damped Oscillator", PlotStyle -> Dashing[{0.02, 0.02}]];
```

```
In[16]:= Show[graphx2, graphx2mod];
```

The curves coincide, as they should.

■ Example 3

This next example is disappointing. We will see that DSolve founders on a rather simple problem. The problem is another damped oscillator. In second order form it is

$$\ddot{x} + \frac{1}{5}\,\dot{x} + x = 0, \text{ with } x(0) = 1, \ \dot{x}(0) = 0,$$

and in system form it is

$$\dot{x} = v, \ \ \dot{v} = -\frac{1}{5}\,v + \text{x}, \ \ \text{with } x(0) = 1, \ v(0) = 0.$$

We first solve this as a second-order equation.

```
In[17]:= DSolve[{x''[t] + (1/5) x'[t] + x[t] == 0, x[0] == 1, x'[0] == 0}, x[t], t]
```

Out[17]= $\left\{\left\{x[t] \to e^{-t/10}\left(\text{Cos}\left[\frac{3\sqrt{11}\,t}{10}\right] + \frac{\text{Sin}\left[\frac{3\sqrt{11}\,t}{10}\right]}{3\sqrt{11}}\right)\right\}\right\}$

For comparison with a later numerical solution, we graph x(t) between 0 and 10. We do this by repeating the calculation and assigning a function name to the output -- namely xans3.

```
In[18]:= xans3[t_] = x[t] /.
           Flatten[DSolve[{x''[t] + (1/5) x'[t] + x[t] == 0, x[0] == 1, x'[0] == 0}, x[t], t]]
```

Out[18]= $$e^{-t/10}\left(\mathrm{Cos}\left[\frac{3\sqrt{11}\,t}{10}\right]+\frac{\mathrm{Sin}\left[\frac{3\sqrt{11}\,t}{10}\right]}{3\sqrt{11}}\right)$$

```
In[19]:= graphx3 = Plot[xans3[t], {t, 0, 10}, AxesLabel -> {"t", "x"}];
```

Now we solve the problem as a first-order system.

```
In[20]:= ans = {x[t], v[t]} /. Flatten[DSolve[
            {x'[t] == v[t], v'[t] == -(1/5) v[t] - x[t], x[0] == 1, v[0] == 0}, {x[t], v[t]}, t]]
```

Out[20]= $$\left\{\frac{1}{66}\left(33\,e^{\frac{1}{10}(-1-3i\sqrt{11})t}+i\sqrt{11}\,e^{\frac{1}{10}(-1-3i\sqrt{11})t}+33\,e^{\frac{1}{10}(-1+3i\sqrt{11})t}-i\sqrt{11}\,e^{\frac{1}{10}(-1+3i\sqrt{11})t}\right),\ -\frac{5\,i\left(e^{\frac{1}{10}(-1-3i\sqrt{11})t}-e^{\frac{1}{10}(-1+3i\sqrt{11})t}\right)}{3\sqrt{11}}\right\}$$

Again DSolve has inappropriately left the answer to a real problem in complex form. We attempt to simplify.

```
In[21]:= Simplify[ans]
```

Out[21]= $$\left\{\frac{1}{66}\,e^{\frac{1}{10}(-1-3i\sqrt{11})t}\left(33+i\sqrt{11}+(33-i\sqrt{11})\,e^{\frac{3}{5}i\sqrt{11}\,t}\right),\ \frac{5\,i\,e^{\frac{1}{10}(-1-3i\sqrt{11})t}\left(-1+e^{\frac{3}{5}i\sqrt{11}\,t}\right)}{3\sqrt{11}}\right\}$$

Simplify didn't help. We can make it work by declaring t real, using ComplexExpand, and taking the real part.

```
In[22]:= Simplify[Re[ComplexExpand[ans]], t ∈ Reals]
```

Out[22]= $$\left\{\frac{1}{33}\,e^{-t/10}\left(33\,\mathrm{Cos}\left[\frac{3\sqrt{11}\,t}{10}\right]+\sqrt{11}\,\mathrm{Sin}\left[\frac{3\sqrt{11}\,t}{10}\right]\right),\ -\frac{10\,e^{-t/10}\,\mathrm{Sin}\left[\frac{3\sqrt{11}\,t}{10}\right]}{3\sqrt{11}}\right\}$$

What a lot of fussing to get DSolve to do what it should have done automatically! 10 points off this time.

■ A Parametric Study

■ Example 4

Here we look at an example in which we vary a system parameter and track the changes in the solution. This could be done by simply repeating the steps carried out above, but we wish to do it a little more systematically. The equation we study here is a damped spring-mass system with a varying damping. The second order form of the equation is

$$m\ddot{x}+b\dot{x}+kx=0.$$

To reduce the number of parameters, we look at a very special case which will still allow us to illustrate the point of a sequence of solutions. We assume that the mass is 1 kg and that the spring constant is 1 N/m. Then we express the damping constant in terms of the damping parameter ζ, using $\zeta = b/2\sqrt{km}$, hence $b = 2\zeta$. We convert the equation to a system by letting $v = \dot{x}$. Then the system is

$$\begin{aligned}\dot{x} &= v,\\ \dot{v} &= -2\,\zeta\,\dot{x} - x\,.\end{aligned}$$

We are going to solve this equation for a sequence of values of ζ, all for the initial conditions

$$x(0) = 1,\ \ \dot{x}(0) = 0.$$

We now define the equation for *Mathematica*. We define it as a function of ζ, which is the parameter we will be varying.

```
In[23]:= equation4[ζ_] := {x'[t] == v[t], v'[t] == -2 ζ v[t] - x[t], x[0] == 1, v[0] == 0};
```

We try this out for $\zeta = 1$.

```
In[24]:= DSolve[equation4[1], {x[t], v[t]}, t]

Out[24]= {{x[t] → e^-t (1 + t), v[t] → -e^-t t}}
```

We get the familiar critically damped solution. Now we begin to automate the process of solving and graphing. We define a solution which is a function of ζ:

```
In[25]:= Clear[ans]

In[26]:= ans[ζ_] := Flatten[DSolve[equation4[ζ], {x[t], v[t]}, t]]
```

We check this by repeating the calculation for $\zeta = 1$.

```
In[27]:= ans[1]

Out[27]= {x[t] → e^-t (1 + t), v[t] → -e^-t t}
```

Now we continue the automation by defining a routine which solves the equation for a given ζ, and then assigns the x-component of the answer to xans4.

```
In[28]:= xans4[ζ_] := x[t] /. First[ans[ζ]]
```

We check this:

```
In[29]:= xans4[1]

Out[29]= e^-t (1 + t)
```

The next step in the automation is to define a function which produces a graph of $x(t)$ for a given value of ζ. We will plot the answer from 0 to 10.

```
In[30]:= graph[ζ_] := Module[{xfunc}, xfunc = xans4[ζ];
           Plot[xfunc, {t, 0, 10}, AxesLabel -> {"t", "x"}, PlotRange -> {-1.01, 1.01},
            PlotLabel -> SequenceForm["ζ = ", PaddedForm[ζ, {4, 2}]]]]
```

This should produced a graph of $x(t)$, labeled with the value of ζ. We try it for $\zeta = 1$.

```
In[31]:= graph[1];
```

Now we can use a Do loop to produce a sequence of graphs. We do this for $\zeta = 0$ to 1 in increments of 0.2.

```
In[32]:=  Do[graph[i], {i, 0, 1, 0.2}]
```

This sequence shows how the solution changes with increasing damping.

■ NUMERICAL SOLUTIONS

■ Constructing and Plotting a Single Solution

■ Example 5

The numerical methods in *Mathematica* are so powerful, that it is really no harder to solve equations numerically than analytically. In fact, here is a professional secret: Many equations that can be solved analytically can be solved much more quickly numerically! The real purpose of solving equations analytically is not speed, but to produce formulas which show how the solutions depend on the parameters of the problem.

We begin with the same equation with which we started this notebook, only this time we solve it numerically. We first define the equation.

```
In[33]:=  equation5 := {x'[t] == y[t], y'[t] == - y[t] + 6 x[t], x[0] == 1, y[0] == -2};
```

Now we solve this equation numerically using NDSolve. The main change from DSolve is that we must now specify the time interval on which the integration takes place. In this case we specify 0 to 2. (Useful note: the left endpoint does not have to be the same as the initial point. NDSolve requires only that the initial point be somewhere in the time interval specified.)

```
In[34]:=  NDSolve[equation5, {x[t], y[t]}, {t, 0, 2}]

Out[34]=  {{x[t] → InterpolatingFunction[{{0., 2.}}, <>][t],
            y[t] → InterpolatingFunction[{{0., 2.}}, <>][t]}}
```

Mathematica has solved the problem, and the answer is returned in the form of a replacement rule pointing to the interpolating function output. Interpolating functions are rather complicated objects which provide a smooth representation of the solution, both on and between the grid points used in the numerical calculation. For convenience, we would prefer the output to be more like an ordinary function. We can do this -- at least in appearance -- by assigning the interpolating function to a function with a name which we choose. We now assign the answer for x to xans5 and the answer for y to yans5.

```
In[35]:= {xans5[t_], yans5[t_]} =
          {x[t], y[t]} /. Flatten[NDSolve[equation1, {x[t], y[t]}, {t, 0, 2}]]

Out[35]= {InterpolatingFunction[{{0., 2.}}, <>][t], InterpolatingFunction[{{0., 2.}}, <>][t]}
```

Now we will show that xans5 and yans5 can be used like ordinary functions. First we check the initial conditions, and a few sample values.

```
In[36]:= xans5[0]

Out[36]= 1.

In[37]:= yans5[0]

Out[37]= -2.

In[38]:= xans5[1]

Out[38]= 1.51764

In[39]:= yans5[1]

Out[39]= 2.83614
```

Interpolating functions even can be differentiated:

```
In[40]:= D[xans5[t], t] /. t -> 1.0

Out[40]= 2.83613
```

Note that these last two numerical results are consistent with the differential equation x'(t) = y(t).

Now we plot x and y on separate graphs.

```
In[41]:= graphx5 = Plot[xans5[t], {t, 0, 2},
            AxesLabel -> {"t", "x"}, PlotLabel -> "Numerical Solution of Equation 1"];


In[42]:= graphy5 = Plot[yans5[t], {t, 0, 2},
            AxesLabel -> {"t", "y"}, PlotLabel -> "Numerical Solution of Equation 1"];
```

These graphs are identical to the ones obtained earlier from the analytical solution.

■ Example 6

Let's return to Example 3, which pushed DSolve beyond the limits of reason. We repeat the calculation, but with NDSolve this time. We name the components of the output xans6 and vans6.

```
In[43]:= {xans6[t_], vans6[t_]} =
          {x[t], v[t]} /. Flatten[NDSolve[{x'[t] == v[t], v'[t] == -(1/5) v[t] - x[t],
              x[0] == 1, v[0] == 0}, {x[t], v[t]}, {t, 0, 10}]]

Out[43]= {InterpolatingFunction[{{0., 10.}}, <>][t], InterpolatingFunction[{{0., 10.}}, <>][t]}
```

Let's graph the x-component of the solution, using dashing, so that we can compare it with the earlier graph of the solution obtained analytically.

```
In[44]:= graphx6 =
           Plot[xans6[t], {t, 0, 10}, AxesLabel -> {"t", "x"}, PlotStyle -> Dashing[{0.02, 0.02}]];
```

Now we show this graph with graphx3, the graph of the analytical solution.

```
In[45]:= Show[graphx3, graphx6];
```

The curves coincide complete at this level of resolution. Let's compare numerical values of the analytical and numerical solution at x = 6.

```
In[46]:= xans3[6]
```

Out[46]= $\dfrac{\text{Cos}\left[\frac{9\sqrt{11}}{5}\right] + \frac{\text{Sin}\left[\frac{9\sqrt{11}}{5}\right]}{3\sqrt{11}}}{e^{3/5}}$

```
In[47]:= N[%]

Out[47]= 0.505106

In[48]:= xans6[6]

Out[48]= 0.505101
```

We see that they agree to five decimal places. It is possible to ask for increased accuracy from NDSolve in those situations where higher accuracy is needed, but there is a cost in computer time, especially if you are running many cases.

■ Example 7 - A Predator Prey Model

In class, we formulated a predator-prey model to describe the interaction of two species. In that model, *x* was the population of the prey, and *y* was the population of the predator. The differential equations of the model are

$$\dot{x} = rx\left(1 - \frac{\mathrm{x}}{\mathrm{x_m}}\right) - axy,$$
$$\text{and } \dot{y} = bxy - cy.$$

Here *r* is the unconstrained growth rate of the prey, x_m is the maximum sustainable population of prey, *a* and *b* are interaction coefficients, and *c* is the natural death rate of the predator. We define the equation for *Mathematica*, including initial values x0 and y0.

```
In[49]:= equation7[r_, xm_, a_, b_, c_, x0_, y0_] := {x'[t] == r x[t] (1 - (x[t] / xm)) - a x[t] y[t],
           y'[t] == b x[t] y[t] - c y[t], x[0] == x0, y[0] == y0}
```

We use years for the time unit and millions for the population unit. We look at a solution with $r = 1$, $a = 3$, $b = 2$, $c = 1$, $x_m = 4$, $x_0 = 1.5$, and $y_0 = 0.5$.

```
In[50]:= ans7 = NDSolve[equation7[1, 4, 3, 2, 1, 1.5, 0.5], {x[t], y[t]}, {t, 0, 5}]

Out[50]= {{x[t] → InterpolatingFunction[{{0., 5.}}, <>][t],
           y[t] → InterpolatingFunction[{{0., 5.}}, <>][t]}}
```

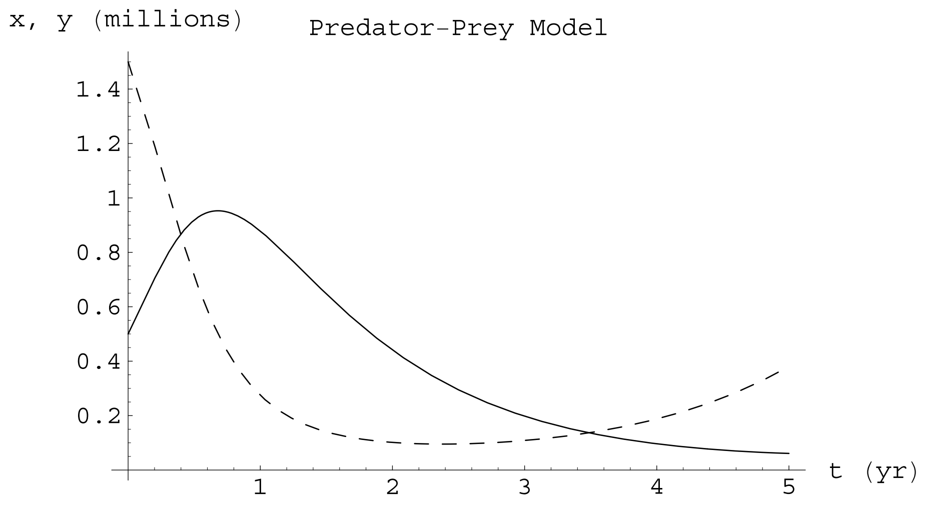
We define the components xans6 and yans6 of this solution.

```
In[51]:= {xans7, yans7} = {x[t], y[t]} /. Flatten[ans7]

Out[51]= {InterpolatingFunction[{{0., 5.}}, <>][t], InterpolatingFunction[{{0., 5.}}, <>][t]}
```

Now we plot them. To help distinguish them, we use dashing for the prey line.

```
In[52]:= graphxy7 = Plot[{xans7, yans7}, {t, 0, 5},
             AxesLabel -> {"t (yr)", "x, y (millions)"}, PlotLabel -> "Predator-Prey Model",
             PlotStyle -> {Dashing[{0.02, 0.02}], Dashing[{}]}, ImageSize -> 350];
```



We can get considerable insight into the system from this graph. We see that the prey population drops initially, and the predator population grows. Eventually the prey become sufficiently scarce that the predator population starts to drop after reaching a peak. When the predator population drops far enough, the prey are able to recover, and the prey population begins to rise. This suggests cyclic behavior, but on a longer time scale than we have shown. Let's look at 20 years and see what happens.

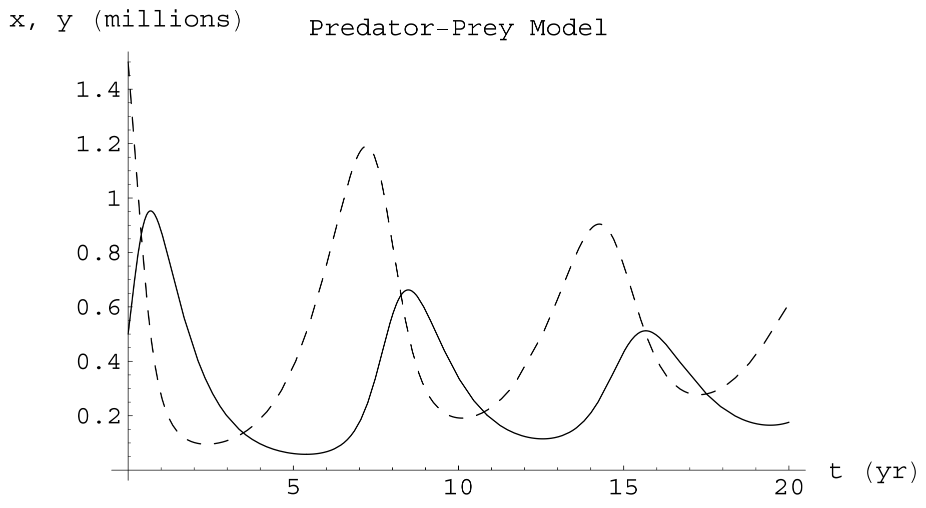
```
In[53]:= ans7mod = NDSolve[equation7[1, 4, 3, 2, 1, 1.5, 0.5], {x[t], y[t]}, {t, 0, 20}]

Out[53]= {{x[t] → InterpolatingFunction[{{0., 20.}}, <>][t],
           y[t] → InterpolatingFunction[{{0., 20.}}, <>][t]}}

In[54]:= {xans7mod, yans7mod} = {x[t], y[t]} /. Flatten[ans7mod]

Out[54]= {InterpolatingFunction[{{0., 20.}}, <>][t], InterpolatingFunction[{{0., 20.}}, <>][t]}
```

```
In[55]:= graphxy7mod = Plot[{xans7mod, yans7mod}, {t, 0, 20},
           AxesLabel -> {"t (yr)", "x, y (millions)"}, PlotLabel -> "Predator-Prey Model",
           PlotStyle -> {Dashing[{0.02, 0.02}], Dashing[{}]}, ImageSize -> 350];
```



Now we see clearly the cyclic behavior. It looks like a damped oscillation. The graph raises the following question: does the system eventually reach a time-independent equilibrium? We will consider such questions very systematically a little later in the course.

■ A Parametric Study

■ Example 8 - A Spring Mass System with a Nonlinear Spring

For our final example, we will study an undamped spring mass system with a nonlinear spring. For this system, the spring force will be $F = -kx - \alpha x^3$. For small x, the spring behave like a linear spring. But for larger x, the x^3 term dominates and the spring force becomes much larger. Because of this, there is a stronger restoring force for large-amplitude motions, and the periodic oscillations will have a period which depends on the amplitude -- a notion that is quite alien in linear theory. Let's test our philosophizing by solving the equation. The second order form of the equation is

$$m\ddot{x} + kx + \alpha x^3 = 0.$$

The equation with this cubic form of spring nonlinearity is called Duffing's equation.

We convert this to a system by introducing the velocity $v = \dot{x}$.

$$\dot{x} = v, \ \dot{v} = -(k/m)x - (\alpha/m)x^3.$$

We are going to solve the equation with a varying initial displacement x_0 and a zero initial velocity. We define the equation for *Mathematica,* as a function of m, k, α, and the initial displacement x_0.

```
In[56]:= equation8[m_, k_, α_, x0_] :=
           {x'[t] == v[t], v'[t] == -(k/m) x[t] - (α/m) (x[t])^3, x[0] == x0, v[0] == 0}
```

We are going to study how the solution depends on the amplitude of the motion. We will fix the values of $m = 1$ kg, $k = 1$ N/m, and $\alpha = 0.1$ N/m^3. We specialize the equation to these values.

```
In[57]:= equation8mod[x0_] := equation8[1, 1, 0.1, x0]
```

Now we define a routine to carry out the integration for a given x_0.

```
In[58]:= ans8[x0_] := NDSolve[equation8mod[x0], {x[t], v[t]}, {t, 0, 10}]
```

Finally, we assign the x(t) of the solution to xans8.

```
In[59]:= xans8[x0_] := x[t] /. First[Flatten[ans8[x0]]]
```

The last step before constructing a sequence of graphs is to define a function which produces a graph of the solution for a given value of x_0.

```
In[60]:= graph8[x0_] := Module[{sol}, sol = xans8[x0];
            Plot[sol, {t, 0, 10}, PlotRange -> {-x0, x0}, AxesLabel -> {"t", "x"},
             PlotLabel -> SequenceForm["Duffing's Equation, x0 = ", PaddedForm[x0, {4, 1}]]]];
```

Finally we try all of this out by finding and graphing the solution when x0 = 0.1 -- small enough so that the spring should look linear.

```
In[61]:= graph8[0.1];
```

Now let's construct a sequence of graphs, showing how the solution changes with increasing amplitude. We let x_0 vary from 0.5 to 3 in increments of 0.5.

```
In[62]:= Do[graph8[i], {i, 0.5, 3, 0.5}];
```

It is obvious from these graphs that as the amplitude increases, the period of the motion decreases, and the wave forms become sharper. A change of period with amplitude is a very common nonlinear phenomenon. You can hear it in some musical instruments. In a clarinet, for example, blowing hard will cause a change in pitch.