ME 163
Using Mathematica to Construct Phase Plane Plots

■ Introduction

In this notebook, we learn how to use *Mathematica* to construct phase plane plots for autonomous systems of two first order equations. We will make use of the techniques we covered earlier on solving such equations with NDSolve. The phase plane plotting that we add here is accomplished by *Mathematica*'s command ParametricPlot, so we start with a discussion of that command. At the end of this notebook are five examples which give a kind of overview of the different types of behaviors of solutions of autonomous systems. These examples are a preview of some things we will do in more detail later.

There are many variables in this notebook with similar names. To avoid spelling check error messages from *Mathematica* (which are sometimes helpful, but in this case annoying), we turn off the spelling checks:

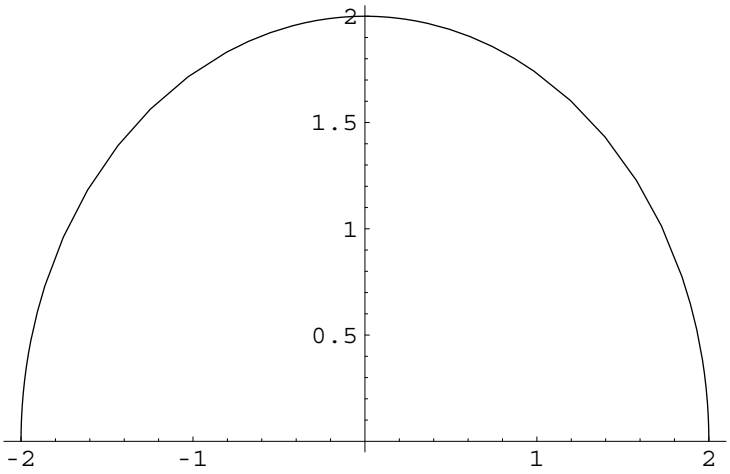
```
In[1]:= Off[General::spell1];
```

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

■ ParametricPlot

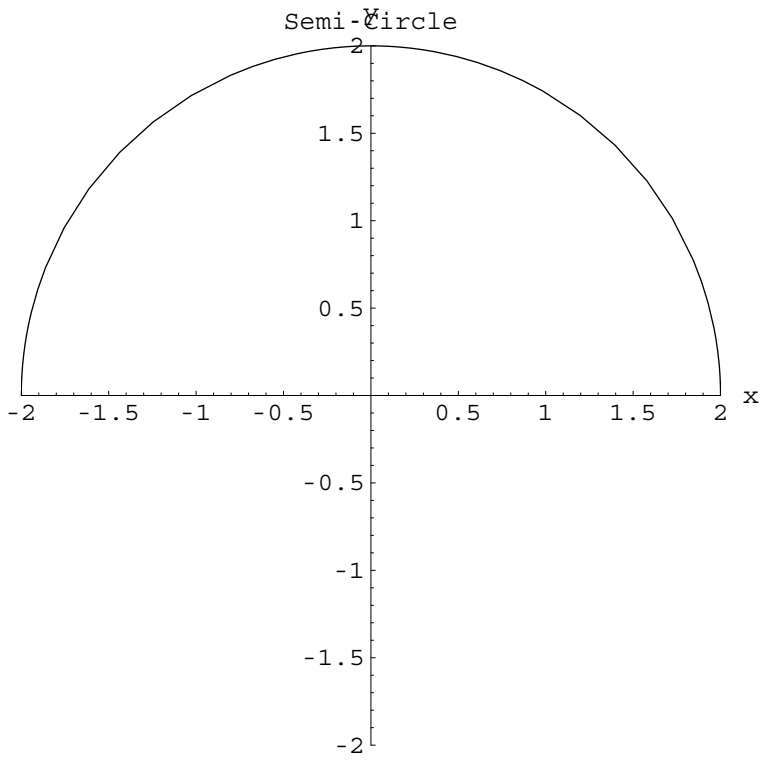
Given a two-dimensional motion defined by $x(t)$, $y(t)$, we could plot x or y or both as functions of t. For some purposes it is more useful to plot the orbit of the motion in the x-y plane. The functions $x(t)$ and $y(t)$ are then a parametric representation of that orbit, with t being the parameter. The *Mathematica* command ParametricPlot is designed to do just such a plot. Let's look at an example of ParametricPlot. We plot the equations for a semi-circle of radius 2, using the parametrization $x(t) = 2\cos(t)$, $y(t) = 2\sin(t)$, $0 \le t \le p$.

```
In[3]:= graph1 = ParametricPlot[{2 * Cos[t], 2 * Sin[t]}, {t, 0, Pi}];
```



We can add all the usual graphics options -- we can label axes, label the plot, specify the aspect ratio, and specify the plotting window.

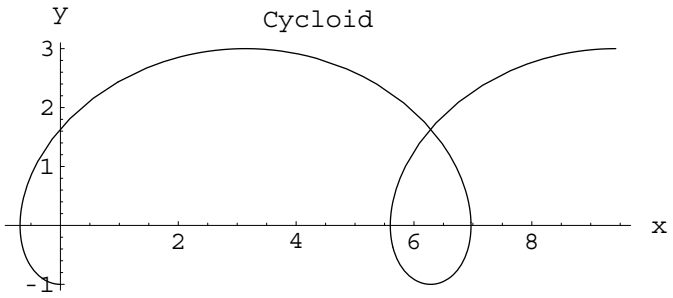
```
In[4]:= graph2 = ParametricPlot[{2 * Cos[t], 2 * Sin[t]}, {t, 0, Pi}, AxesLabel -> {"x", "y"},
  PlotRange -> {{-2, 2}, {-2, 2}}, PlotLabel -> "Semi-Circle", AspectRatio -> 1];
```



A familiar *Mathematica* bug has shown up -- the collision of the plot label and the *y*-axis label.

Here is another example with a more interesting geometry. It is called a cycloid.

```
In[5]:= graph3 = ParametricPlot[{t - 2 * Sin[t], 1 - 2 * Cos[t]}, {t, 0, 3 Pi},
          AxesLabel -> {"x", "y"}, PlotLabel -> "Cycloid", AspectRatio -> Automatic];
```



■ Phase Plot for a Single Solution

Now we are going to combine what we just learned about ParametricPlot with our earlier work using NDSolve to construct solutions of systems of differential equations. Although it is sometimes possible to obtain analytical solutions by using DSolve, such cases are rare in practice. For that reason, and also for the benefit of uniformity of approach, we will use NDSolve exclusively for all of our work in this notebook. We start by defining a canonical problem on which to test our software as we develop it, namely the linear damped oscillator. The second order equation is

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

The system form of this is obtained by introducing $v = \dot{x}$. The result is

$$\dot{x} = v\ ,\ \dot{v} = -(b/m)v - (k/m)x\ ,\ \text{with } x(0) = x_0\ ,\ v(0) = v_0\ .$$

We define the equation for *Mathematica*.

```
In[6]:= osc[m_, b_, k_, x0_, v0_] :=
          {x'[t] == v[t], v'[t] == -(b / m) * v[t] - (k / m) * x[t], x[0] == x0, v[0] == v0}
```

Now following our work in an earlier notebook, we define a routine which gives the result of integrating the differential equation with NDSolve, starting at $t = 0$, and integrating out to $t = tf$.

```
In[7]:= ans[m_, b_, k_, x0_, v0_, tf_] :=
          {x[t], v[t]} /. Flatten[NDSolve[osc[m, b, k, x0, v0], {x[t], v[t]}, {t, 0, tf}]]
```

We try this out:

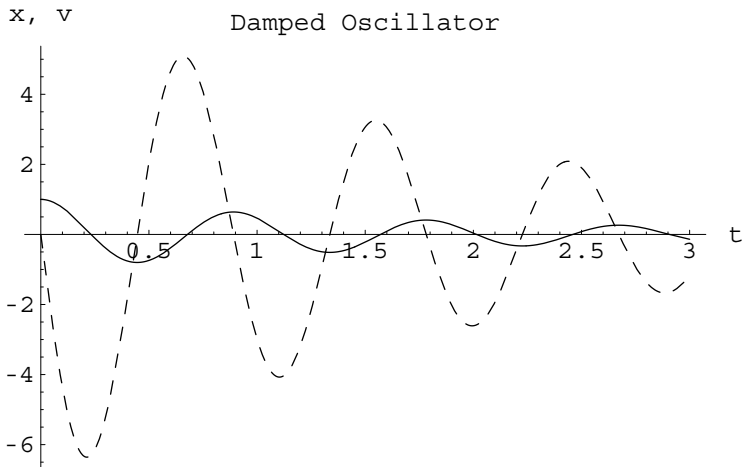
```
In[8]:= sol1 = ans[10, 10, 500, 1, 0, 3]

Out[8]= {InterpolatingFunction[{{0., 3.}}, <>][t], InterpolatingFunction[{{0., 3.}}, <>][t]}
```

The output sol1 contains the two interpolating functions which represent $x(t)$ and $v(t)$. We can evaluate them or plot them. For example, the values at $t = 1$ are

```
In[9]:= sol1 /. t -> 1

Out[9]= {0.465292, -2.99363}
```

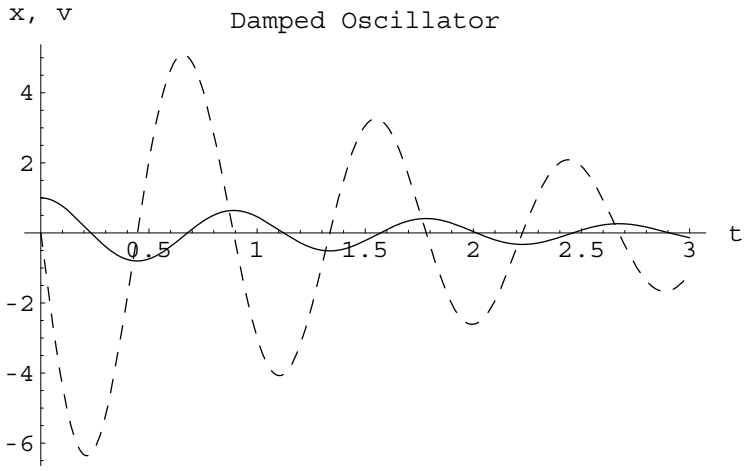
We now plot x and v as functions of t, using solid for x and dashed for v.

```
In[10]:= oscgraph1 = Plot[{First[sol1], Last[sol1]}, {t, 0, 3}, AxesLabel -> {"t", "x, v"},
           PlotLabel -> "Damped Oscillator", PlotStyle -> {Dashing[{}], Dashing[{0.02, 0.02}]}];
```



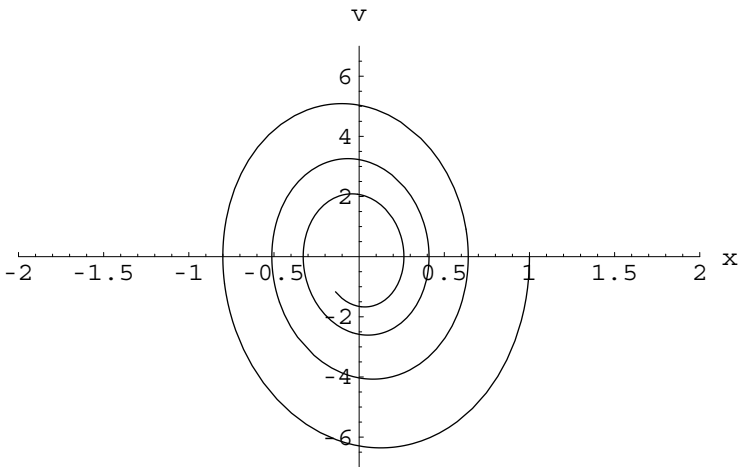
All of this is very similar to what we did before, except that now we have not assigned separate names to the x and y components of the solution. We can easily do that.

```
In[11]:= xsol1[t_] := First[sol1]

In[12]:= ysol1[t_] := Last[sol1]

In[13]:= oscgraph1 = Plot[{xsol1[t], ysol1[t]}, {t, 0, 3}, AxesLabel -> {"t", "x, v"},
           PlotLabel -> "Damped Oscillator", PlotStyle -> {Dashing[{}], Dashing[{0.02, 0.02}]}];
```



Now for something new. We construct a phase plane plot of the solution by applying ParametricPlot to xsol1 and ysol1. The above graph of x and v versus t gives us good information for choosing the plotting window in the phase plane: {-2,2} will include all of the x-values, and {-7,7} will include all of the y-values.

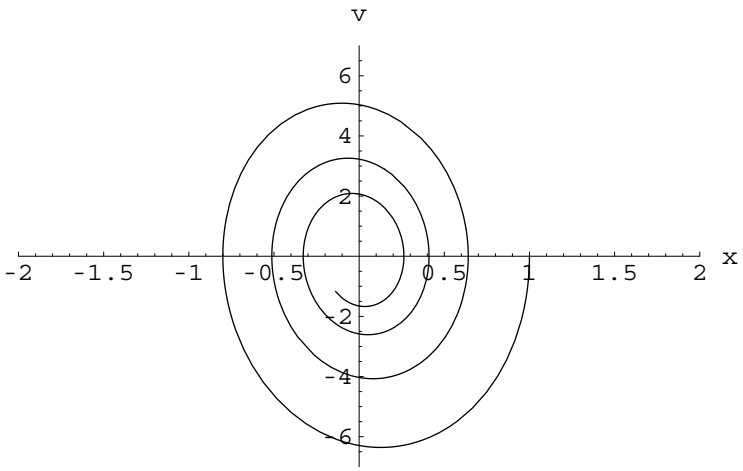
```
In[14]:= oscgraph2 = ParametricPlot[{xsol1[t], ysol1[t]},
           {t, 0, 3}, PlotRange -> {{-2, 2}, {-7, 7}}, AxesLabel -> {"x", "v"}];
```



For this underdamped oscillator, we see that the phase plot is a spiral. As time increases, the system point moves inward towards the equilibrium at $x = 0$, $v = 0$. In physical terms, the oscillations decrease in amplitude as the damping takes energy out of the system.

We could also construct the plot without referring separately to xsol1 and ysol1.

```
In[15]:= oscgraph3 =
          ParametricPlot[sol1, {t, 0, 3}, PlotRange -> {{-2, 2}, {-7, 7}}, AxesLabel -> {"x", "v"}];

ParametricPlot::ppcom :
 Function sol1 cannot be compiled; plotting will proceed with the uncompiled function.
```



We can ignore *Mathematica*'s complaint about not being able to compile the interpolating function, or we can avoid the nagging by splitting sol into x and y parts as we have done already.

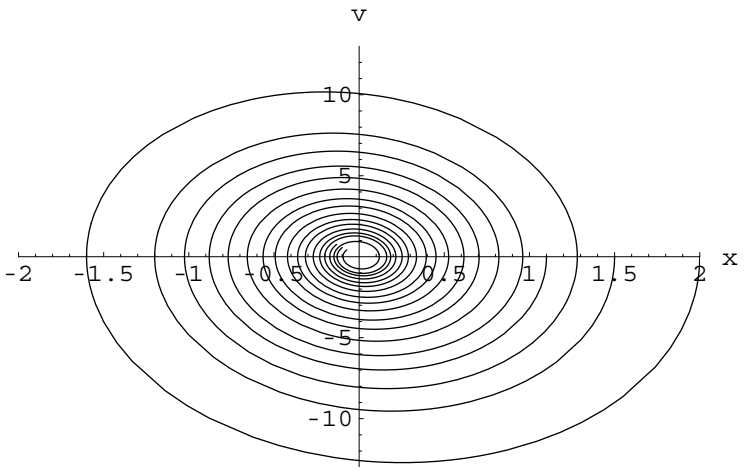
■ Phase Plots for Multiple Solutions

The real value of the phase plane plot is the view it gives us of families of solutions, and the insight which that view provides into overall system behavior. We now learn how to plot several solutions on one phase plane plot. We start by constructing three solutions for our damped oscillator, with different initial conditions. We choose initial displacements of 1.1, 1.5 and 2.0 m, and zero initial velocity.

```
In[16]:= sol1 = ans[10, 10, 500, 1.1, 0, 5];

In[17]:= sol2 = ans[10, 10, 500, 1.5, 0, 5];

In[18]:= sol3 = ans[10, 10, 500, 2.0, 0, 5];
```

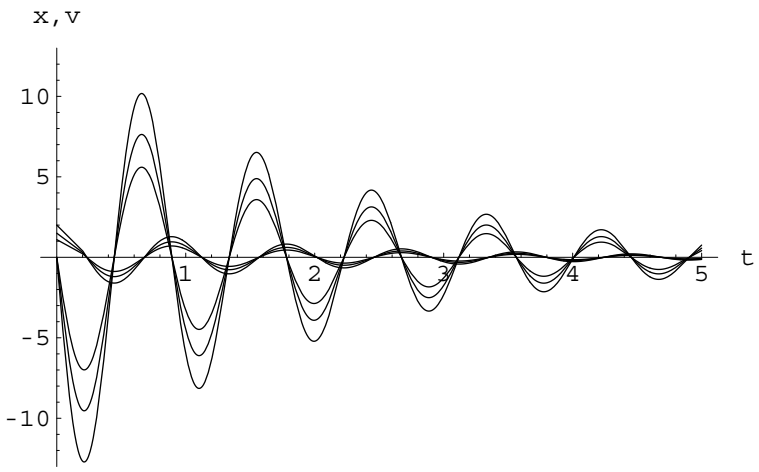
Now we plot them.

```
In[19]:= oscgraph4 = ParametricPlot[
           {{First[sol1], Last[sol1]}, {First[sol2], Last[sol2]}, {First[sol3], Last[sol3]}},
           {t, 0, 5}, PlotRange -> {{-2, 2}, {-13, 13}}, AxesLabel -> {"x", "v"}];
```



Every solution of this equation spirals in toward the equilibrium point at {0,0}. None of these spirals intersect other spirals, and none intersects itself. The phase plane provides an excellent way to visualize such a family of solutions. If we attempt to view these solutions simultaneously as functions of *t*, the result is quite cluttered:

```
In[20]:= oscgraph5 =
          Plot[{First[sol1], Last[sol1], First[sol2], Last[sol2], First[sol3], Last[sol3]},
           {t, 0, 5}, AxesLabel -> {"t", "x,v"}, PlotRange -> {-13, 13}];
```



■ Direction Fields

Early in the course, we discussed the direction field for first order equations. The construction of a direction field is equally useful in the study of autonomous systems of two first-order equations. It provides an overall view of where the solution curves go, and the arrows show which way the system moves as time increases. We define here Mathematica code for plotting such direction fields. As in all programming, there is a trade-off between flexibility and simplicity. Here we opt for simplicity. The resulting code, although limited, will serve our purposes.

We consider systems of first order differential equations of the form

$$\frac{dx}{dt} = F(x, y), \quad \frac{dy}{dt} = G(x, y) .$$

We wish to plot the direction field defined by the slope vector {F(x,y), G(x,y)}. Our first step is to load into Mathematica a special graphics package called PlotField which is needed to construct the direction fields. This is accomplished by the command below.

```
In[21]:= Needs["Graphics`PlotField`"]
```

We are now going to define a function called dirfield which will construct a direction field plot. The function will have four arguments: statevec, slopevec, horange, and verange. The argument statevec is the list of the two independent variables -- {x,y} in the above example. The argument slopevec is the vector of the right-hand sides -- {F(x,y), G(x,y)} in the above example. The argument horange is the plotting range for the horizontal coordinate, and verange is the plotting range for the vertical coordinate. The definition of dirfield is given below. It makes use of a Mathematica function PlotVectorField which is part of the package PlotField that we just loaded.

In[22]:=

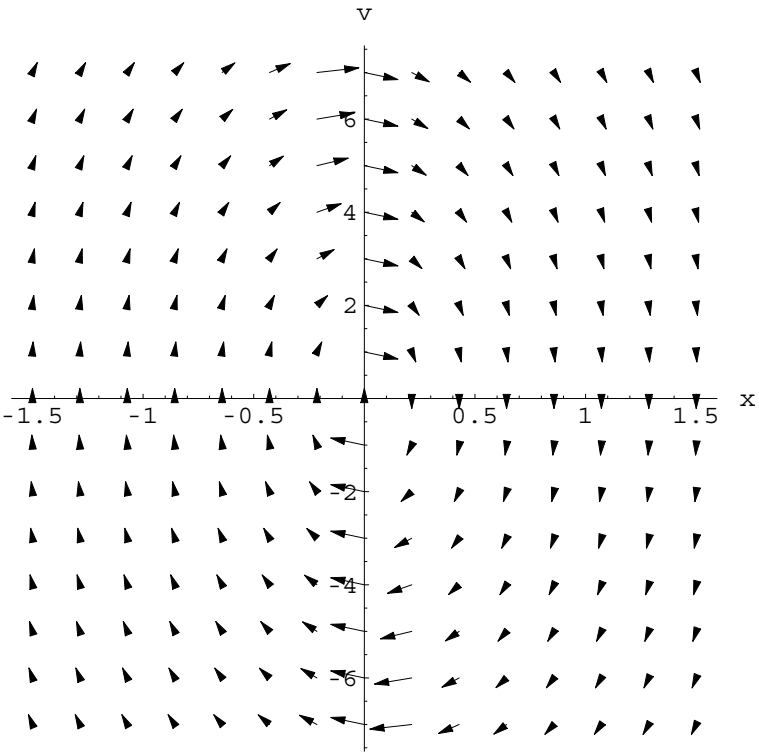
```
dirfield[statevec_, slopevec_, horange_, verange_] :=
 PlotVectorField[
  slopevec, {statevec[[1]], horange[[1]], horange[[2]]}, {statevec[[2]],
   verange[[1]], verange[[2]]}, ScaleFunction -> (1 &),
  Prolog -> {Thickness[0.001]}, Axes -> True, AspectRatio -> 1, PlotPoints -> 15,
  AxesLabel -> {statevec[[1]], statevec[[2]]}]
```

We try this out on our damped oscillator, for $m = 10$, $b = 10$, $k = 500$. The system of differential equations describing this is

$$\dot{x} = v, \quad \dot{v} = -50\,x - v\,.$$

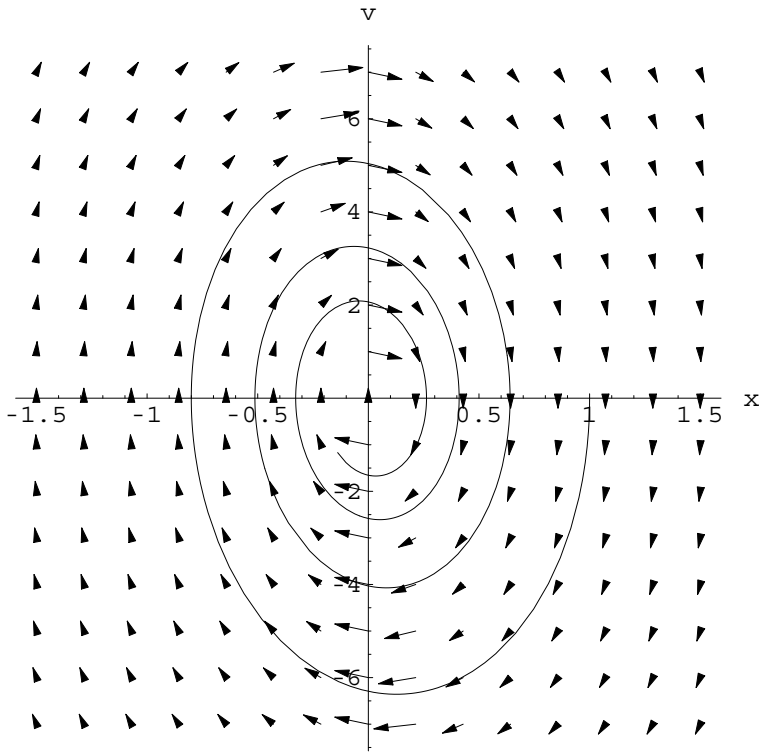
In[23]:=

```
dirgraph = dirfield[{x, v}, {v, -50 x - v}, {-1.5, 1.5}, {-7, 7}];
```



We now superimpose the direction field and one of the solutions to this equation:

```
In[24]:= Show[dirgraph, oscgraph2];
```



■ A Small Zoo of Phase Plots

We conclude this notebook by looking at five examples of systems with interesting phase plots. Some of these examples will be a preview of topics yet to come. Taken together, the five examples illustrate the qualitatively different ways in which solutions of autonomous systems can behave. The main point to look for in these examples is the way in which a phase plane picture of several solutions gives insight into the general behavior of the system.

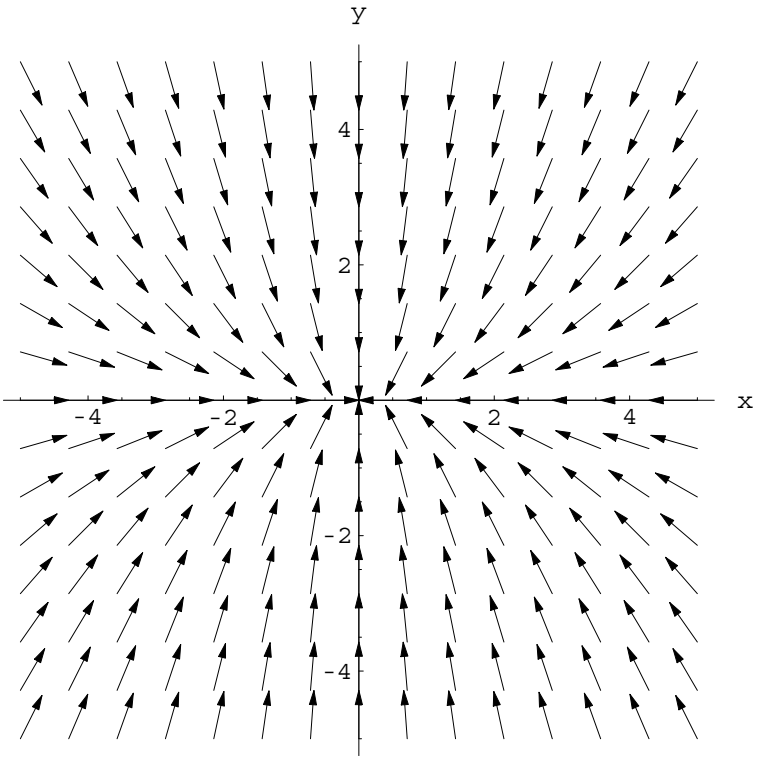
ü Example 1 -- Approach to Equilibrium

We consider the system defined by

$$\frac{dx}{dt} = -x\,, \quad \frac{dy}{dt} = -2\,y\,.$$

We start with a direction field plot to give us a general idea of what's going on. We choose a plotting window of {-5,5} in both x and y.

```
In[25]:= dirgraphz1 = dirfield[{x, y}, {-x, -2 * y}, {-5, 5}, {-5, 5}];
```



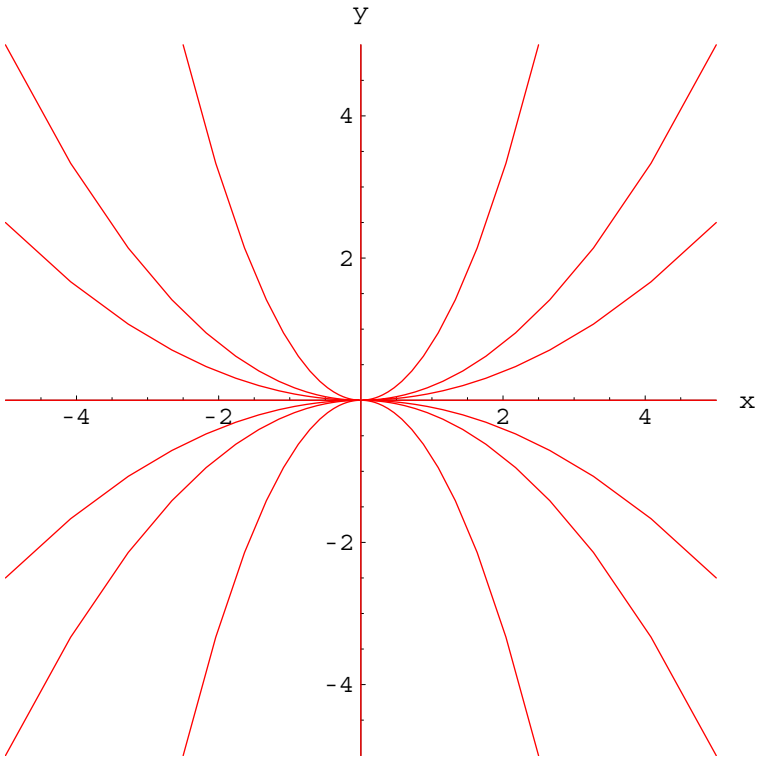
All the arrows point generally toward the origin, suggesting that no matter where we start, we will end up there. Let's construct some solutions and superimpose them on our direction field plot. We look at 16 solutions, all starting on the edges of plotting window, and spaced more or less evenly around the window. We expect that all 16 solutions will follow the arrows and head for the equilibrium point at {0,0}. First we define the equation as a function of the initial values.

```
In[26]:= equatz1[x0_, y0_] := {x'[t] == -x[t], y'[t] == -2 * y[t], x[0] == x0, y[0] == y0}

In[27]:= {xz1, yz1} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[5, 5], {x[t], y[t]}, {t, 0, 5}]];

In[28]:= {xz2, yz2} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[-5, 5], {x[t], y[t]}, {t, 0, 5}]];

In[29]:= {xz3, yz3} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[-5, -5], {x[t], y[t]}, {t, 0, 5}]];

In[30]:= {xz4, yz4} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[5, -5], {x[t], y[t]}, {t, 0, 5}]];

In[31]:= {xz5, yz5} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[5, 2.5], {x[t], y[t]}, {t, 0, 5}]];

In[32]:= {xz6, yz6} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[2.5, 5], {x[t], y[t]}, {t, 0, 5}]];

In[33]:= {xz7, yz7} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[-2.5, 5], {x[t], y[t]}, {t, 0, 5}]];

In[34]:= {xz8, yz8} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[-5, 2.5], {x[t], y[t]}, {t, 0, 5}]];

In[35]:= {xz9, yz9} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[-5, -2.5], {x[t], y[t]}, {t, 0, 5}]];

In[36]:= {xz10, yz10} =
          {x[t], y[t]} /. Flatten[NDSolve[equatz1[-2.5, -5], {x[t], y[t]}, {t, 0, 5}]];
```

```
In[37]:= {xz11, yz11} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[2.5, -5], {x[t], y[t]}, {t, 0, 5}]];

In[38]:= {xz12, yz12} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[5, -2.5], {x[t], y[t]}, {t, 0, 5}]];

In[39]:= {xz13, yz13} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[5, 0], {x[t], y[t]}, {t, 0, 5}]];

In[40]:= {xz14, yz14} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[0, 5], {x[t], y[t]}, {t, 0, 5}]];

In[41]:= {xz15, yz15} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[-5, 0], {x[t], y[t]}, {t, 0, 5}]];

In[42]:= {xz16, yz16} = {x[t], y[t]} /. Flatten[NDSolve[equatz1[0, -5], {x[t], y[t]}, {t, 0, 5}]];
```

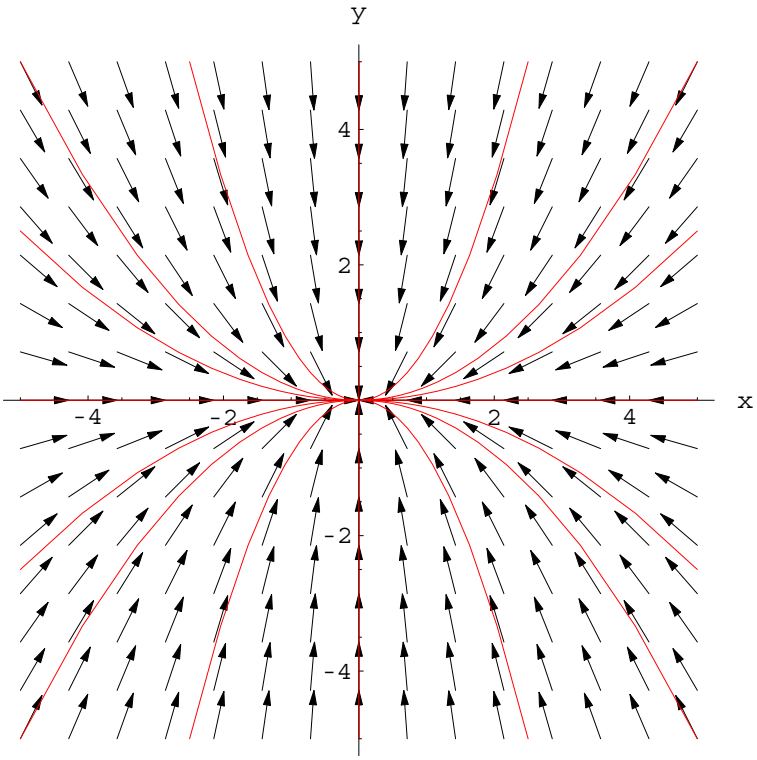
There are much more efficient and systematic ways of constructing families of solutions, but this works. We will do better with the remaining examples. Now we graph these solutions. We show them in red, so that when we combine them with the direction field, they will be visible.

```
In[43]:= solgraphz1 =
          ParametricPlot[{{xz1, yz1}, {xz2, yz2}, {xz3, yz3}, {xz4, yz4}, {xz5, yz5}, {xz6, yz6},
            {xz7, yz7}, {xz8, yz8}, {xz9, yz9}, {xz10, yz10}, {xz11, yz11}, {xz12, yz12},
            {xz13, yz13}, {xz14, yz14}, {xz15, yz15}, {xz16, yz16}},
           {t, 0, 5}, PlotRange -> {{-5, 5}, {-5, 5}}, AxesLabel -> {"x", "y"},
           AspectRatio -> 1, PlotStyle -> {RGBColor[1, 0, 0]}];
```



We combine this with the direction field.

```
In[44]:= Show[dirgraphz1, solgraphz1];
```



The combination of the solutions and the direction field illustrate the basic behavior of this system: no matter where we start, we end up at the equilibrium at the origin. We will see later how to establish such results with analysis. This kind of equilibrium point is called a stable node.

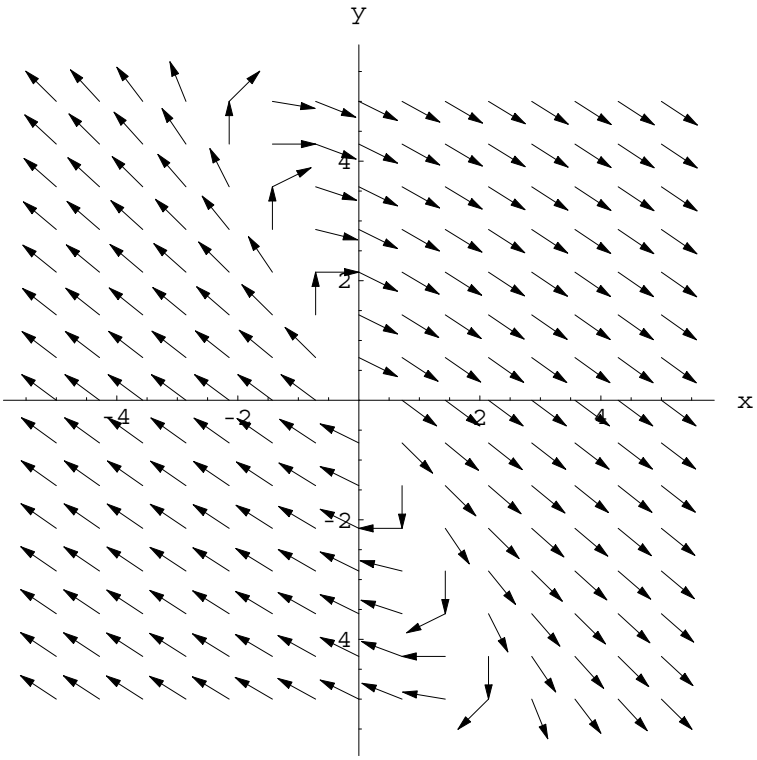
ü Example 2 -- Divergence to Infinity

For our second example, we consider the system

$$\dot{x} = 4x + 2y, \quad \dot{y} = -3x - y.$$

We start with a direction field plot.

```
In[45]:= dirgraphz2 = dirfield[{x, y}, {4 x + 2 y, -3 x - y}, {-5, 5}, {-5, 5}];
```



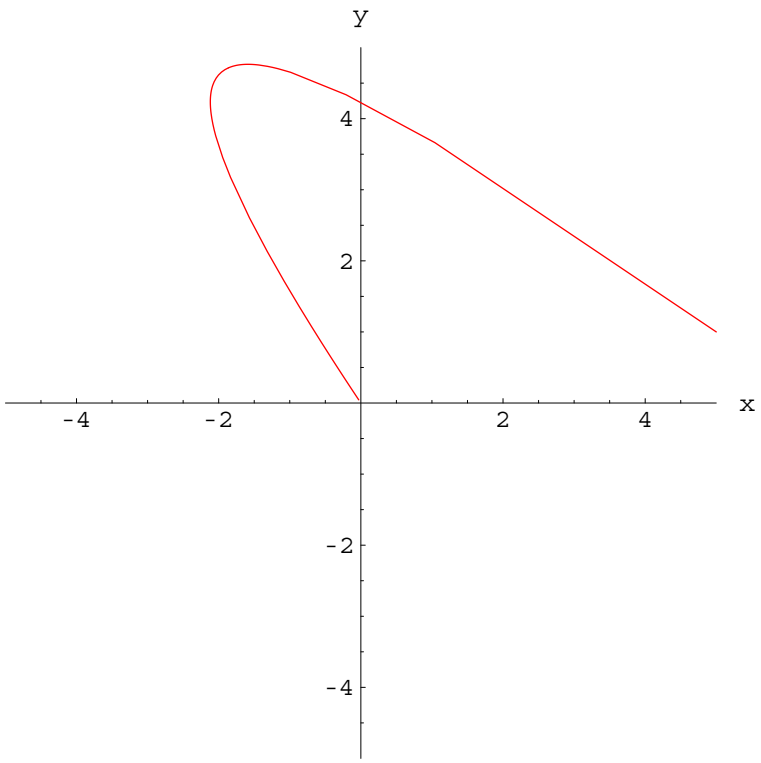
The direction field suggests that the solutions all run off to inifinty, no matter where they start, although some solutions turn around before taking their final run to infinity. We will construct 8 solutions to add to this plot. The solutions will be those which cross the frame of the plotting window at the following points:

```
In[46]:= initlistz2 = {{5, 1}, {5, 2}, {-5, 1}, {-5, 2}, {-5, -1}, {-5, -2}, {5, -1}, {5, -2}};
```

At first there would seem to be a problem constructing these solutions, because in every case the solution will be *leaving* the window at those points, so the question is where do such solutions start? Presumably somewhere near the origin. We can deal with this in a simple way. We take the coordinates in the above list to be initial positions at $t = 0$, and then we integrate over a time interval {-6, 0}. NDSolve will begin the integration at $t = 0$, and then go back in time to $t = -6$, thus tracing the solution back in time. Let's try this for one solution before generating all 8. First we define the equation as a function of the initial values.

```
In[47]:= equatz2[x0_, y0_] :=
          {x'[t] == 4 x[t] + 2 y[t], y'[t] == -3 x[t] - y[t], x[0] == x0, y[0] == y0};

In[48]:= {xz2test, yz2test} =
          {x[t], y[t]} /. Flatten[NDSolve[equatz2[5, 1], {x[t], y[t]}, {t, -6, 0}]];
```

```
In[49]:= testgraphz2 =
          ParametricPlot[{xz2test, yz2test}, {t, -6, 0}, PlotRange -> {{-5, 5}, {-5, 5}},
           AxesLabel -> {"x", "y"}, AspectRatio -> 1, PlotStyle -> RGBColor[1, 0, 0]];
```

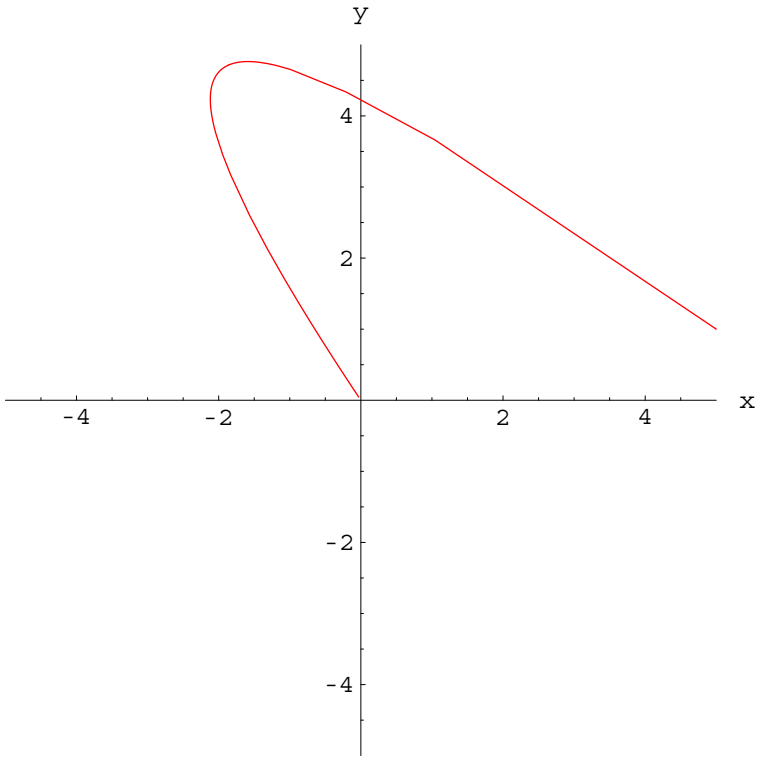


We see that this solution starts very near the origin at t = -6, and has progressed to the point {5,1} when t has increased to 0. Now we construct graphs of all of the other solutions with the given initial conditions in initlistz2 -- 8 solutions all together. We first define a routine that calculates and then graphs a solution for a given set of initial conditions.

```
In[50]:= graphz2[x0_, y0_] := Module[{xans, yans}, {xans, yans} =
            {x[t], y[t]} /. Flatten[NDSolve[equatz2[x0, y0], {x[t], y[t]}, {t, -6, 0}]];
           ParametricPlot[{xans, yans}, {t, -6, 0}, PlotRange -> {{-5, 5}, {-5, 5}},
            AxesLabel -> {"x", "y"}, AspectRatio -> 1, PlotStyle -> RGBColor[1, 0, 0]]];
```

We try this on the case we just did.

```
In[51]:= graphz2[5, 1];
```



Our routine works, but when we apply it to get our 8 solutions, we are going to get individual graphs of each. This is not what we want. We want a composite graph of all 8 solutions. We can do this by telling our basic graph routine NOT to display the graph it creates. This is accomplished by the option DisplayFunction -> Identity.

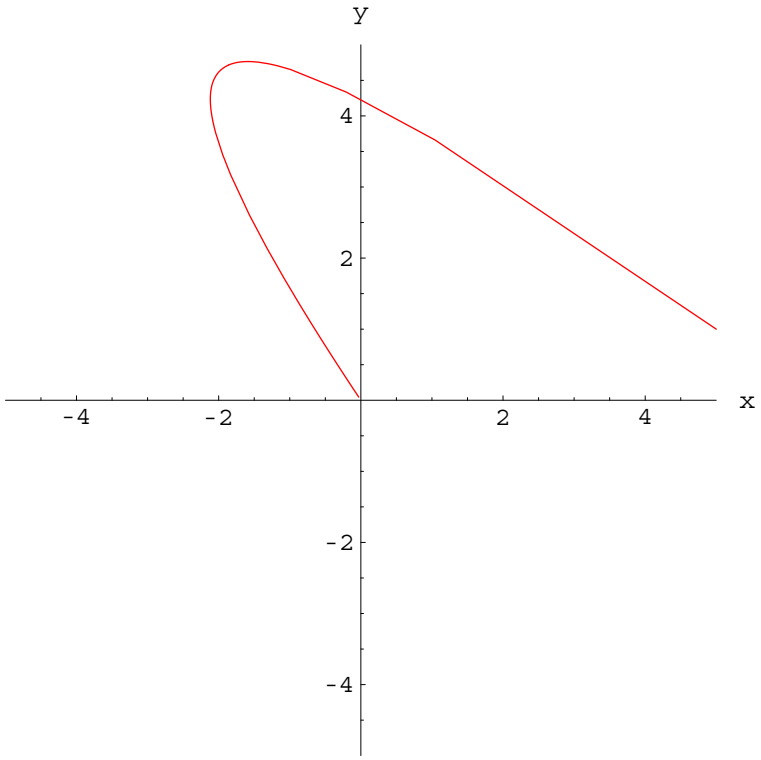
```
In[52]:= graphz2[x0_, y0_] := Module[{xans, yans}, {xans, yans} =
           {x[t], y[t]} /. Flatten[NDSolve[equatz2[x0, y0], {x[t], y[t]}, {t, -6, 0}]];
         ParametricPlot[{xans, yans}, {t, -6, 0}, PlotRange -> {{-5, 5}, {-5, 5}},
          AxesLabel -> {"x", "y"}, AspectRatio -> 1,
          DisplayFunction -> Identity, PlotStyle -> RGBColor[1, 0, 0]]];
```

We try it again.

```
In[53]:= testgraphz2a = graphz2[5, 1];
```

Our modification has been successful, but now how do we see the graph when we want to see it? We use a Show command with the option DisplayFunction->$DisplayFunction, which tells *Mathematica* to show the graph.

```
In[54]:= Show[testgraphz2a, DisplayFunction -> $DisplayFunction];
```

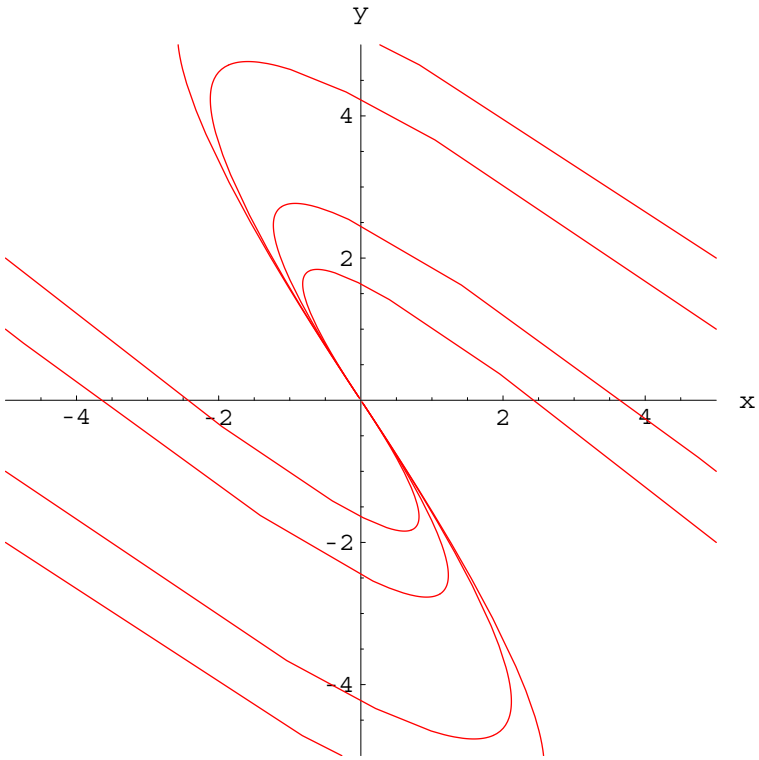


Now we are going to use a Do loop to create all 8 graphs, without showing them individually. At the end we will show them all together, and after that, we will combine them with the direction field. We collect the graphs in a list named graphlistz2.

```
In[55]:= Module[{i, x0temp, y0temp, pair, newgraph}, graphlistz2 = {}; Do[pair = initlistz2[[i]];
           x0temp = First[pair]; y0temp = Last[pair]; newgraph = graphz2[x0temp, y0temp];
           graphlistz2 = Append[graphlistz2, newgraph], {i, 1, 8}]]
```

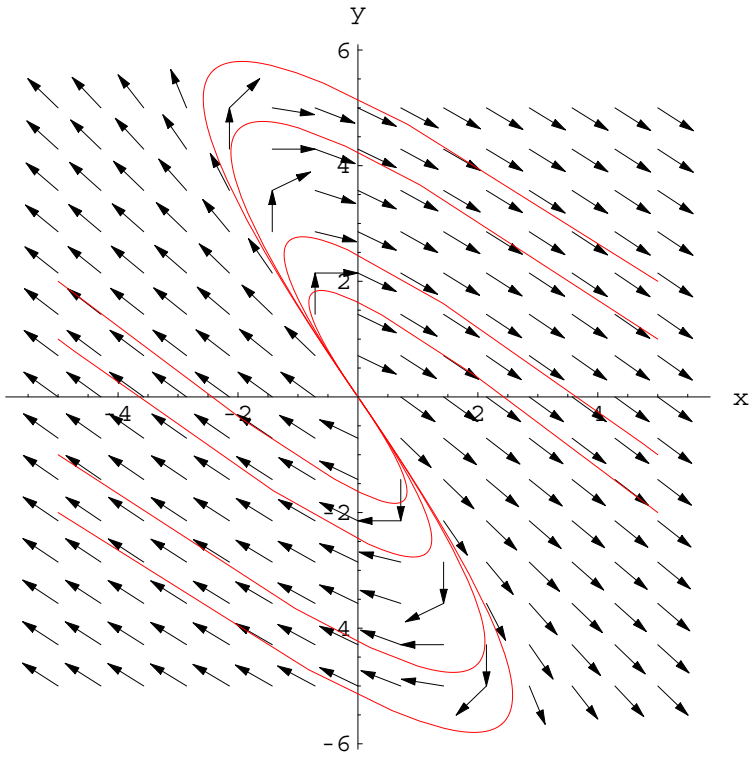
Now we plot them.

```
In[56]:= solgraphz2 = Show[graphlistz2, DisplayFunction -> $DisplayFunction];
```



Now we combine this with the direction field.

```
In[57]:= Show[dirgraphz2, solgraphz2];
```



Although the solutions exhibit considerable change in direction, they all eventually do run off to infinity. We will see later how to establish this by analysis. This kind of equilibrium point is called an unstable node.

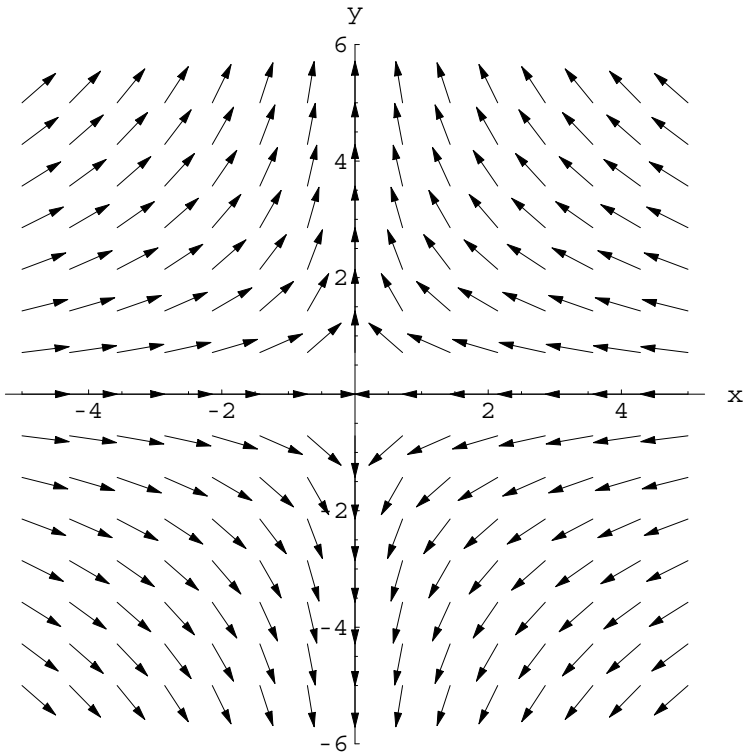
ü Example 3 -- Divergence to Infinity With Some Exceptions

In our next example, we see again a system in which most initial points run off to inifinity, but with the new feature that there are some exceptional points which do not. The equations are

$$\dot{x} = -x, \quad \dot{y} = y .$$

We begin by constructing the direction field.

```
In[58]:= dirgraphz3 = dirfield[{x, y}, {-x, y}, {-5, 5}, {-5, 5}];
```



Although some of the solutions approach the equilibrium at {0,0} initially, they eventually curve and go off to infinity -- except for the solutions on the x-axis. Those solutions which start on the *x*-axis remain on the *x*-axis and move toward the origin. We now construct a family of solutions so that we can display them with the direction field. We define the equation as a function of the initial values.

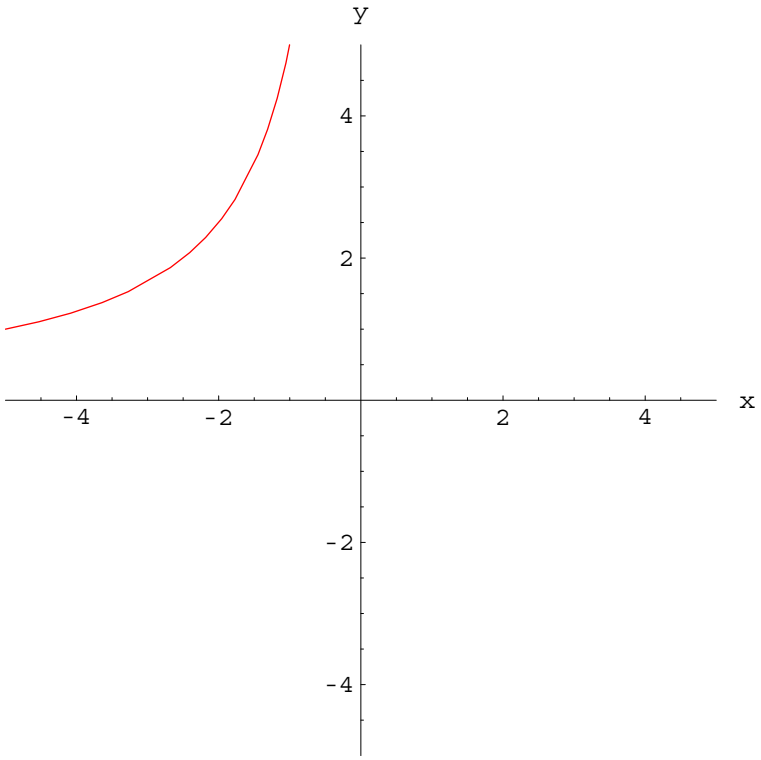
```
In[59]:= equatz3[x0_, y0_] := {x'[t] == - x[t] , y'[t] == y[t], x[0] == x0, y[0] == y0};
```

As before we define a function which will produce, but not display, the phase plane plot for given x0, y0.

```
In[60]:= graphz3[x0_, y0_] := Module[{xans, yans},
           {xans, yans} = {x[t], y[t]} /. Flatten[NDSolve[equatz3[x0, y0], {x[t], y[t]}, {t, 0, 5}]]
           ParametricPlot[{xans, yans}, {t, 0, 5},
            PlotRange -> {{-5, 5}, {-5, 5}}, AxesLabel -> {"x", "y"}, AspectRatio -> 1,
            DisplayFunction -> Identity, PlotStyle -> RGBColor[1, 0, 0]]];
```

We try this for one initial condition.

```
In[61]:= Show[graphz3[-5, 1], DisplayFunction -> $DisplayFunction];
```



This appears to be OK. Now we define a set of initial conditions from which we will construct solutions.

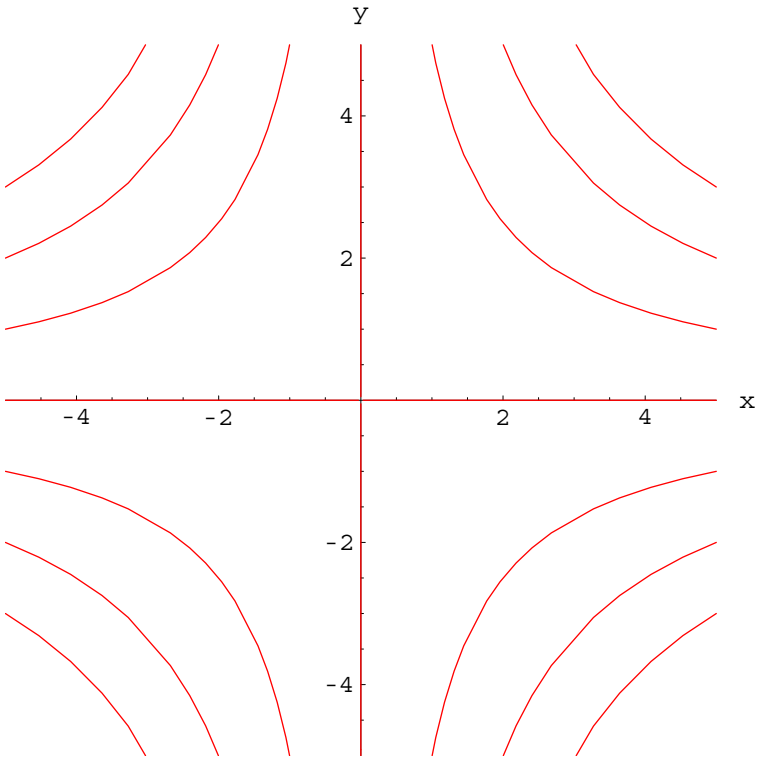
```
In[62]:= initlistz3 = {{5, -3}, {5, -2}, {5, -1}, {5, 0}, {5, 1}, {5, 2}, {5, 3}, {-5, -3},
           {-5, -2}, {-5, -1}, {-5, 0}, {-5, 1}, {-5, 2}, {-5, 3}, {0, 0.05}, {0, -0.05}};
```

We produce all of the graphs without showing them.

```
In[63]:= Module[{i, x0temp, y0temp, pair, newgraph}, graphlistz3 = {}; Do[pair = initlistz3[[i]];
           x0temp = First[pair]; y0temp = Last[pair]; newgraph = graphz3[x0temp, y0temp];
           graphlistz3 = Append[graphlistz3, newgraph], {i, 1, 16}]]
```

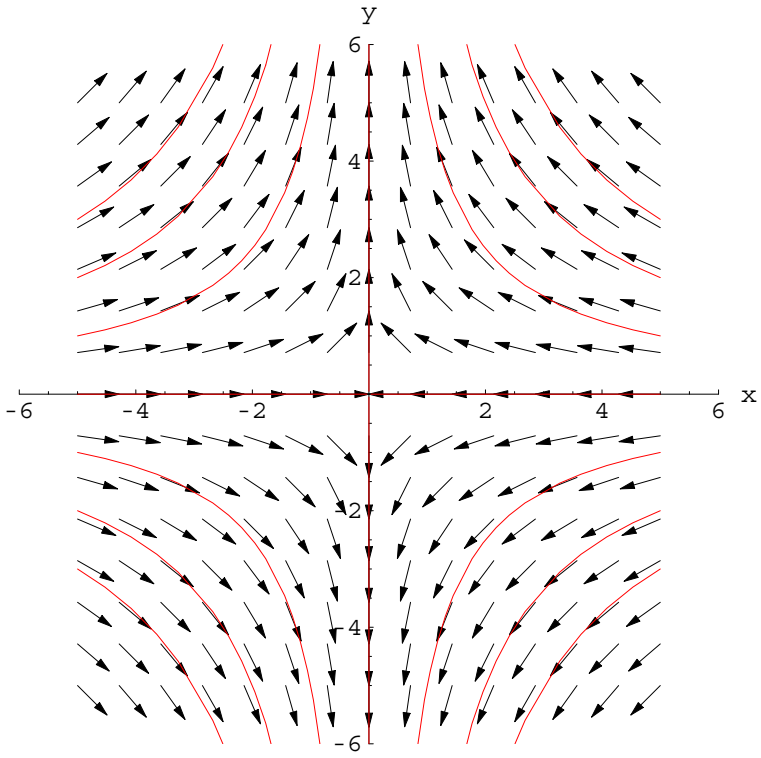
We plot them:

```
In[64]:= solgraphz3 = Show[graphlistz3, DisplayFunction -> $DisplayFunction];
```



Finally we combine the solutions and the direction field.

```
In[65]:= Show[dirgraphz3, solgraphz3, PlotRange -> {{-6, 6}, {-6, 6}}];
```



We see that all of the solutions that do not start on the x-axis go off to infinity. Those that start on the x-axis remain on the axis and head towards the origin. However, it is misleading to leave the discussion at that. If a solution is exactly on the x-axis and it suffers a small disturbance, it will in general be displaced slightly off the x-axis. After that, it will be on one of the curves that go off to infinity. Thus solutions exactly on the x-axis are not stable to small disturbances, and therefore are not realizable in any practical sense. We will explore these ideas in much more detail and with more precision later. The equilibrium point in this example is known as a saddle point.

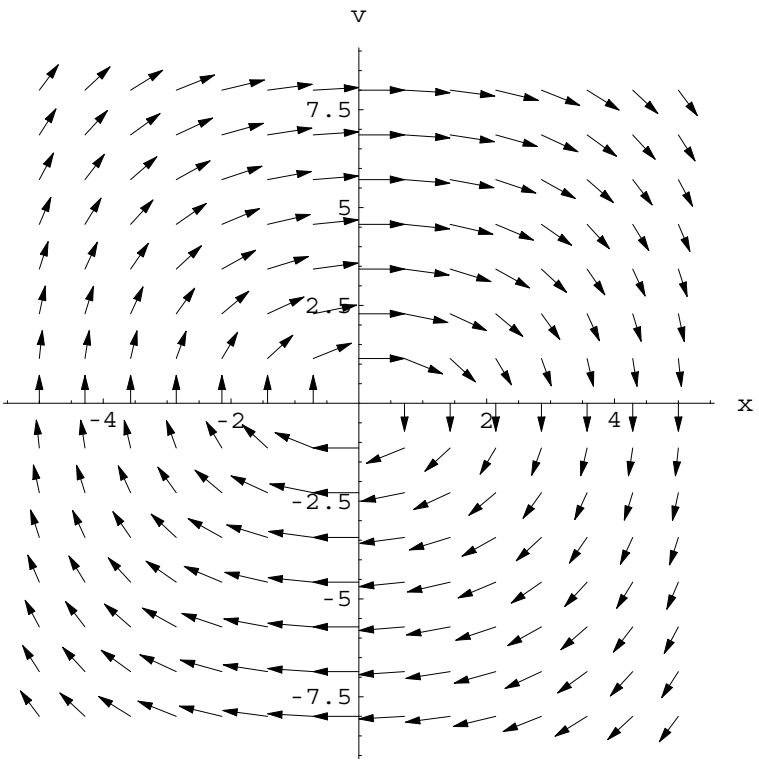
ü Example 4 -- Family of Periodic Solutions

So far, we have seen systems in which all initial points approach an equilibrium, systems in which all initial points go off to infinity, and systems in which almost all initial points go off to infinity. In this example, we get a very different kind of behavior. We are going to see that all initial points lead to periodic motions, the orbits of which show up in the phase plane as closed curves. The system we study is the undamped Duffing equation.

$$\dot{x} = v, \quad \dot{v} = -x - 0.1\,x^3 .$$

We start with the direction field.

```
In[66]:= dirgraphz4 = dirfield[{x, v}, {v, -x - 0.1 x^3}, {-5, 5}, {-8, 8}];
```



There is a strong suggestion of periodic motion, in the way that the arrows circle around. Now we will construct some solutions and then display them with the direction field. We define the equation first.

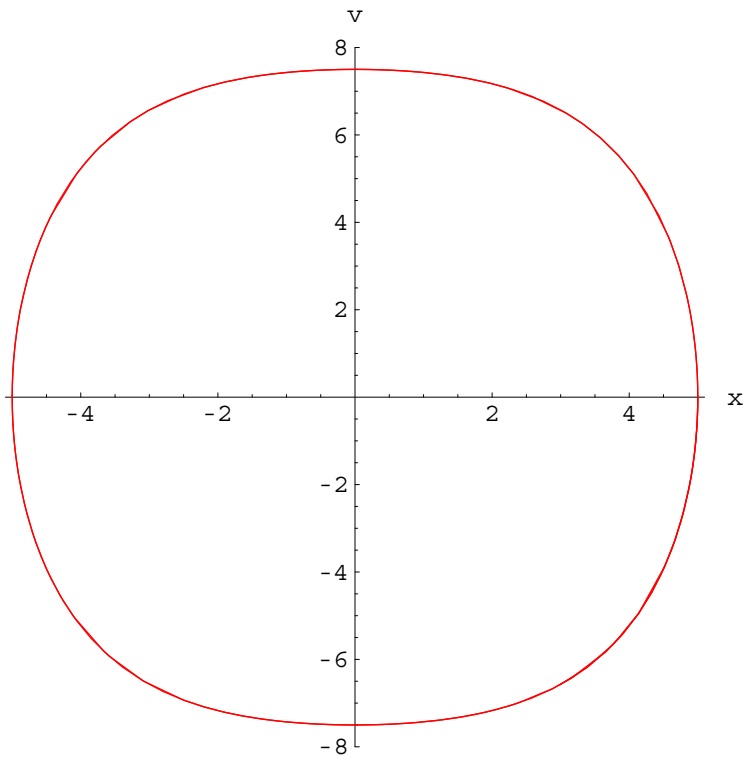
```
In[67]:= equatz4[x0_, v0_] := {x'[t] == v[t] , v'[t] == -x[t] - 0.1 * (x[t])^3, x[0] == x0, v[0] == v0};
```

As before we define a function which will produce, but not display, the phase plane plot for given x0, y0.

```
In[68]:= graphz4[x0_, v0_] := Module[{xans, vans}, {xans, vans} =
           {x[t], v[t]} /. Flatten[NDSolve[equatz4[x0, v0], {x[t], v[t]}, {t, 0, 8}]];
         ParametricPlot[{xans, vans}, {t, 0, 8}, PlotRange -> {{-5.1, 5.1}, {-8, 8}},
          AxesLabel -> {"x", "v"}, AspectRatio -> 1,
          DisplayFunction -> Identity, PlotStyle -> RGBColor[1, 0, 0]]];
```

We try this for one initial condition.

```
In[69]:= Show[graphz4[5, 0], DisplayFunction -> $DisplayFunction];
```



We get a closed curve, which is the signature of a periodic solution. Now we construct a family of solutions, using the list of initial conditions given below.

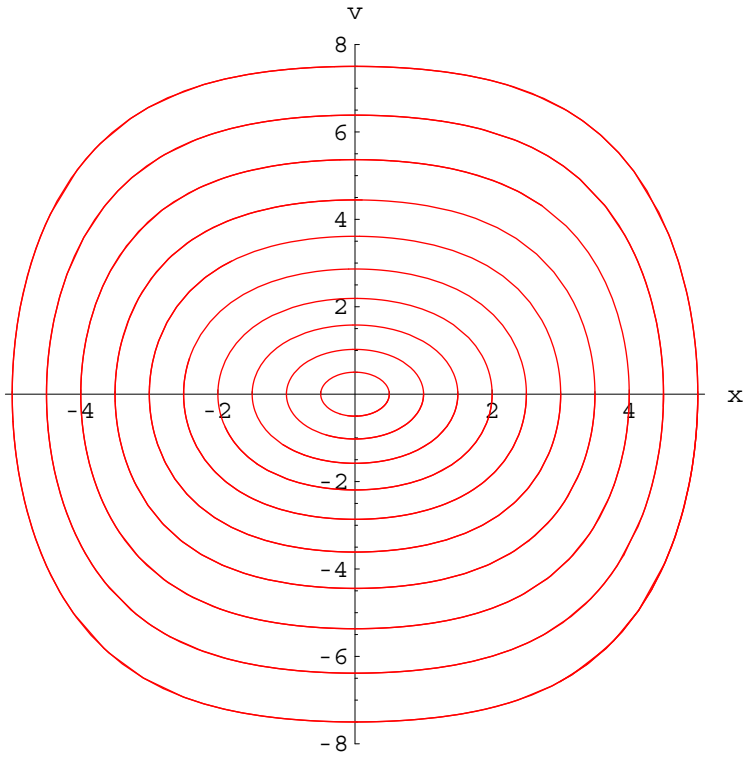
```
In[70]:= initlistz4 =
          {{0.5, 0}, {1, 0}, {1.5, 0}, {2, 0}, {2.5, 0}, {3, 0}, {3.5, 0}, {4, 0}, {4.5, 0}, {5, 0}};
```

Now we produce all of the graphs without showing them individually.

```
In[71]:= Module[{i, x0temp, v0temp, pair, newgraph}, graphlistz4 = {}; Do[pair = initlistz4[[i]];
           x0temp = First[pair]; v0temp = Last[pair]; newgraph = graphz4[x0temp, v0temp];
           graphlistz4 = Append[graphlistz4, newgraph], {i, 1, 10}]]
```

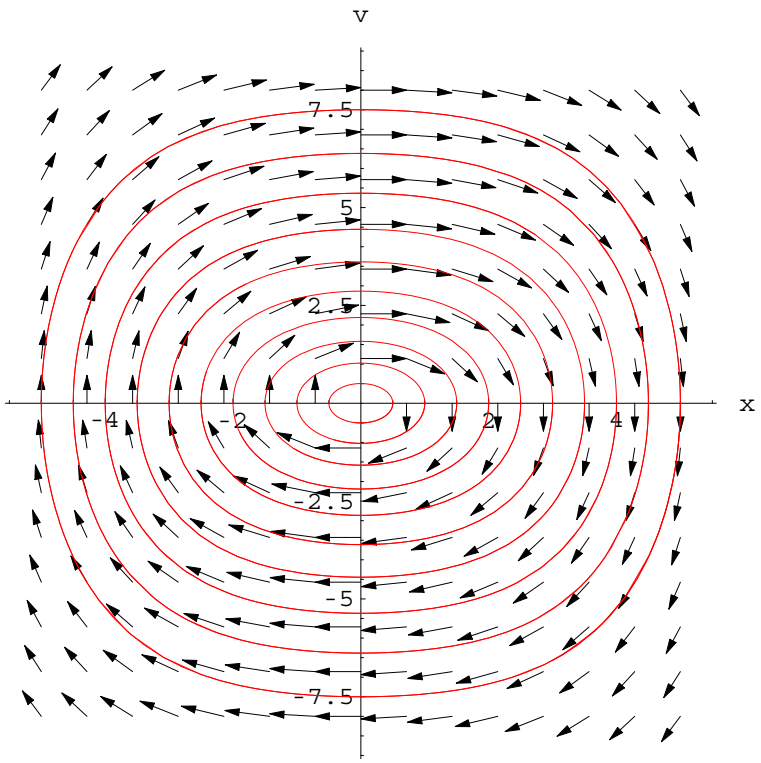
Now we plot them.

```
In[72]:= solgraphz4 = Show[graphlistz4, DisplayFunction -> $DisplayFunction];
```



Finally we combine the solutions and the direction field.

```
In[73]:= Show[dirgraphz4, solgraphz4];
```



Our graph shows that this interesting system has the property that every initial condition gives rise to a periodic solution. Of course if we remember that the equations describe an undamped spring mass system, such behavior is not so surprising. The equilibrium point at the origin is enclosed by the periodic solutions, but the system does not go to the equilibrium point. If we introduced some damping, all of the closed curves would change to spirals which would asymptotically approach the equilibrium point.

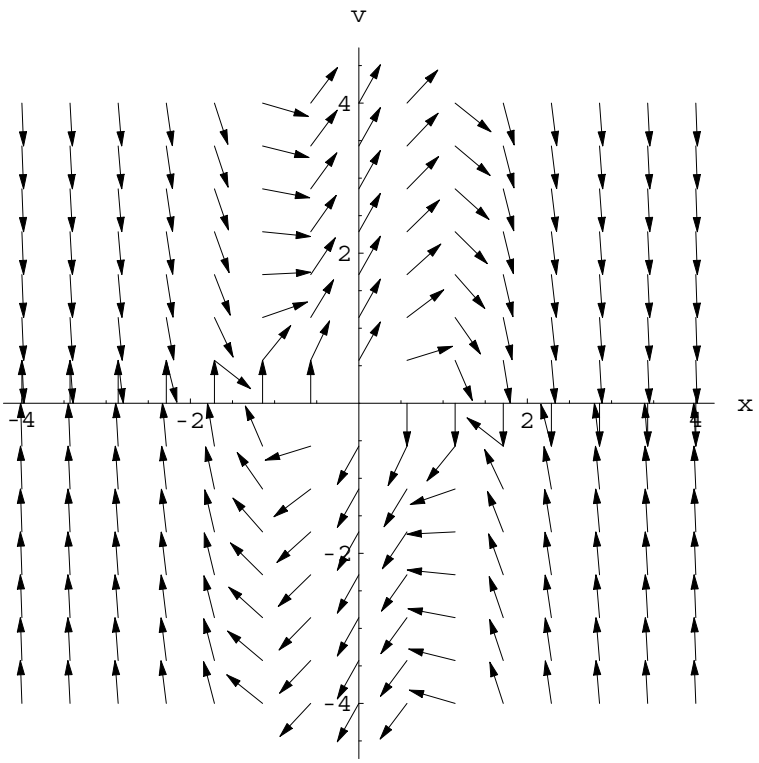
ü Example 5 -- Isolated Periodic Solution

Our last example exhibits a kind of behavior that we have not seen before. The equation, called the van der Pol equation, is given below.

$$\dot{x} = v, \quad \dot{v} = -\mu(x^2 - 1)v - x.$$

The solution depends in an interesting way on the parameter μ, but that is something which we leave for later. For now we take $\mu = 2$. We construct the direction field first.

```
In[74]:= dirgraphz5 = dirfield[{x, v}, {v, -x - 2 ( x^2 - 1) v }, {-4, 4}, {-4, 4}];
```



It is not so easy to tell what is going on from the direction field, although there is a hint of periodic motion. Let's construct some solutions. We define the equation for *Mathematica*, as a function of the initial values x0 and v0.

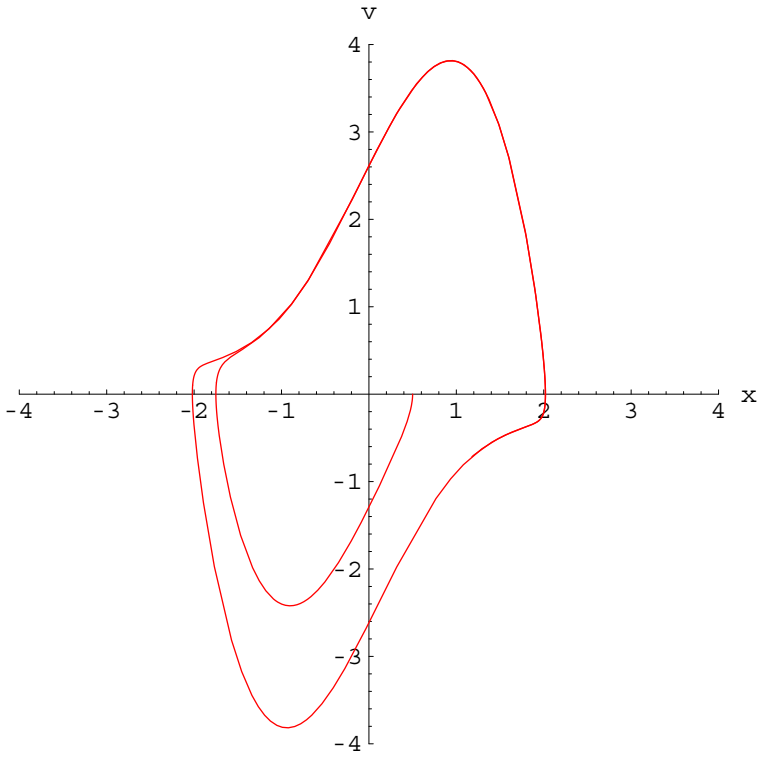
```
In[75]:= equatz5[x0_, v0_] :=
          {x'[t] == v[t] , v'[t] == -x[t] - 2 * ((x[t]) ^2 - 1) * v[t], x[0] == x0, v[0] == v0};
```

As before we define a function which will produce, but not display, the phase plane plot for given x0, y0.

```
In[76]:= graphz5[x0_, v0_] := Module[{xans, vans}, {xans, vans} =
            {x[t], v[t]} /. Flatten[NDSolve[equatz5[x0, v0], {x[t], v[t]}, {t, 0, 15}]];
           ParametricPlot[{xans, vans}, {t, 0, 15}, PlotRange -> {{-4, 4}, {-4, 4}},
            AxesLabel -> {"x", "v"}, AspectRatio -> 1,
            DisplayFunction -> Identity, PlotStyle -> RGBColor[1, 0, 0]]];
```

We try this for one initial condition.

```
In[77]:= Show[graphz5[0.5, 0], DisplayFunction -> $DisplayFunction];
```



The solution looks like it is winding out towards a closed curve. We need more solutions to have a better picture of what is happening. We use the initial conditions in the list below.

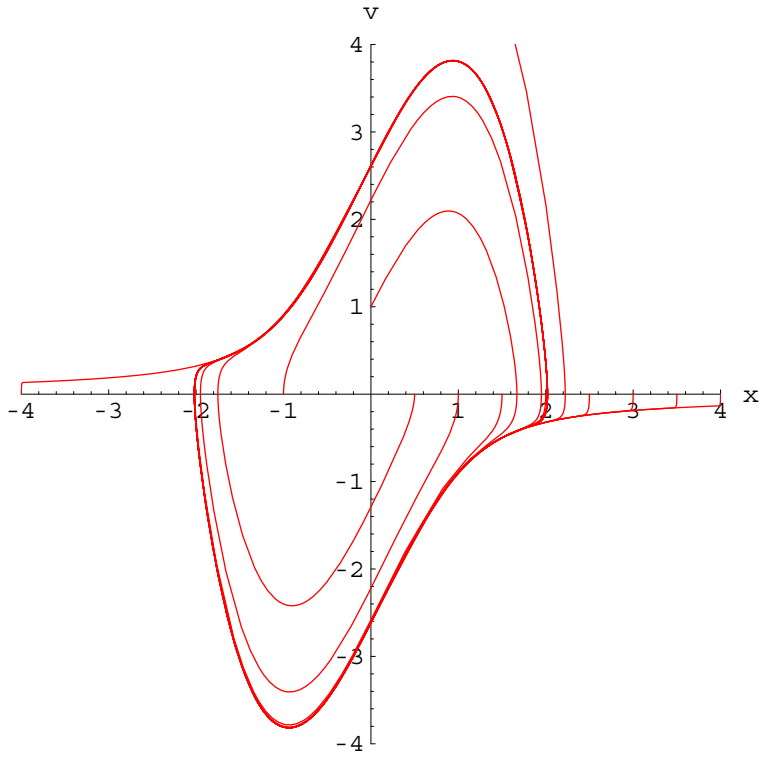
```
In[78]:= initlistz5 = {{0.5, 0}, {1, 0}, {1.5, 0}, {2, 0},
           {2.5, 0}, {3, 0}, {3.5, 0}, {4, 0}, {-4, 0}, {0, 4}, {0, 1}, {-1, 0}};
```

Now we produce all of the graphs without showing them individually.

```
In[79]:= Module[{i, x0temp, v0temp, pair, newgraph}, graphlistz5 = {}; Do[pair = initlistz5[[i]];
           x0temp = First[pair]; v0temp = Last[pair]; newgraph = graphz5[x0temp, v0temp];
           graphlistz5 = Append[graphlistz5, newgraph], {i, 1, 12}]]
```

We plot them.

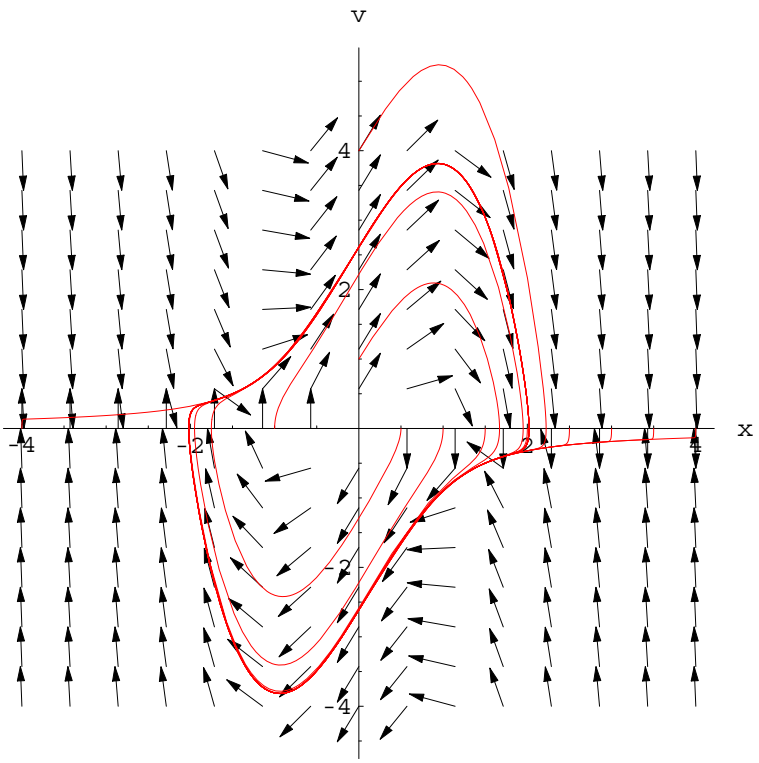
```
In[80]:= solgraphz5 = Show[graphlistz5, DisplayFunction -> $DisplayFunction];
```



It looks like there is a single periodic solution with a rather strange shape. Initial points starting inside the solution wind out towards it, and initial points starting outside that solution wind in towards it. Such an isolated periodic solution is called a *limit cycle*. For this system, every initial condition is eventually attracted onto the limit cycle. No matter where we start this system, it ends up in the same periodic solution. If a disturbance pushes the system temporarily off the limit cycle, the suqsequent motion will take it back onto the limit cycle. Thus this periodic solution is stable. This equation was first studied in the context of electronics, with the problem being the construction of a stable oscillator.

We combine the solutions and the direction field.

```
In[81]:= Show[dirgraphz5, solgraphz5];
```



Can we produce a graph of the pure periodic solution without the clutter of the transient approaches? Yes, if we work a little harder. We start by integrating the equation for any initial condition -- say {1,0}.

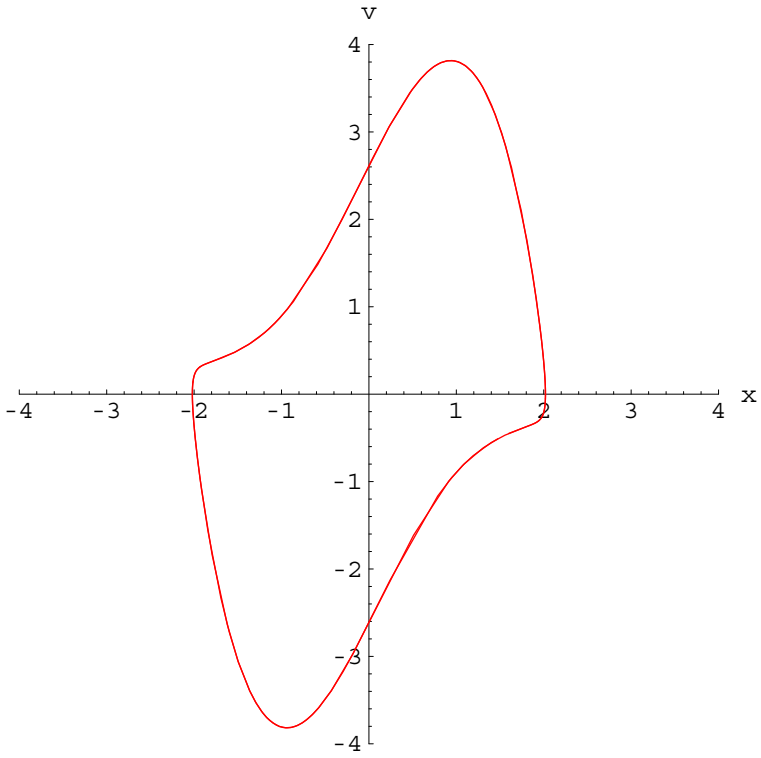
```
In[82]:= {xans, vans} = {x[t], v[t]} /. Flatten[NDSolve[equatz5[1, 0], {x[t], v[t]}, {t, 0, 15}]]

Out[82]= {InterpolatingFunction[{{0., 15.}}, <>][t], InterpolatingFunction[{{0., 15.}}, <>][t]}
```

Now we evaluate the solution at $t = 15$. The pictures we have already constructed, suggest that the solution should be very close to the periodic part by then.

```
In[83]:= pt15 = {xans, vans} /. t -> 15

Out[83]= {1.47374, -0.511029}
```

Now we use this point as an initial condition and do a second integration and graph.

```
In[84]:= Show[graphz5[First[pt15], Last[pt15]], DisplayFunction -> $DisplayFunction];
```



Now we have a nice picture of the isolated periodic solution. We will study this example in more detail later.