Differential Equations&*Mathematica*

©1999 Bill Davis and Jerry Uhl
Produced by Bruce Carpenter Published by Math Everywhere, Inc.
www.matheverywhere.com

DE.06 Systems and Flows Basics

B.1) Flows and their trajectories as pairs of solutions of a system of differential equations

□B.1.a) Vector fields

A vector field is a function that spits out vectors.
You make a 2D vector field by taking two regular functions,
m[x, y] and n[x, y],
and throwing them into the two slots:

```
Clear[Field, m, n, x, y];
m[x_, y_] = 1.4 (y - 0.3);
n[x_, y_] = 1.2 (x - 0.5);
Field[x_, y_] = {m[x, y], n[x, y]}
```

```
{1.4 (-0.3 + y), 1.2 (-0.5 + x)}
```

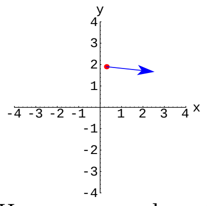
At the sample point
{xsample, ysample} = {0.3, 1.9},
the field vector is:

```
{xsample, ysample} = {0.3, 1.9};
Field[xsample, ysample]
```

```
{2.24, -0.24}
```

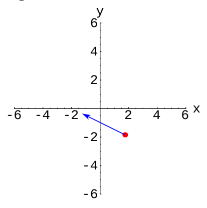
You plot this vector with its tail at {xsample, ysample}:

```
ranger = 4;
tailplot =
 Graphics[{Red, PointSize[0.03], Point[{xsample, ysample}]}];
fieldvectorplot = Arrow[Field[xsample, ysample],
  Tail → {xsample, ysample}, VectorColor → Blue, HeadSize → 0.8];

Show[tailplot, fieldvectorplot, Axes → True, AxesLabel → {"x", "y"},
 PlotRange → {{-ranger, ranger}, {-ranger, ranger}}];
```



Here are some plots of some random field vectors for the given field
Field[x, y] = {m[x, y], n[x, y]}:

```
ranger = 6;
h = 3.0;
{xsample, ysample} = {Random[Real, {-h, h}], Random[Real, {-h, h}]};
tailplot =
 Graphics[{Red, PointSize[0.03], Point[{xsample, ysample}]}];
fieldvectorplot = Arrow[Field[xsample, ysample],
  Tail → {xsample, ysample}, VectorColor → Blue, HeadSize → 0.6];

Show[tailplot, fieldvectorplot, Axes → True, AxesLabel → {"x", "y"},
 PlotRange → {{-ranger, ranger}, {-ranger, ranger}}];
```



Rerun several times.

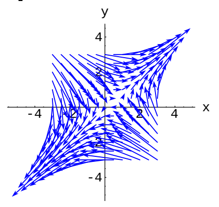
□B.1.a.i)

How do you see the whole thing?

□ **Answer:**

You can plot a big sample of the vector field by plotting the vector, Field[x, y], with its tail at {x, y} for a selection of points {x, y}:

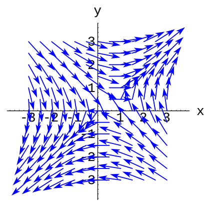
```
{xlow, xhigh} = {-3.0, 3.0};
{ylow, yhigh} = {-3.0, 3.0};
jump = 0.5;
flowplot = Table[Arrow[Field[x, y], Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.5, HeadSize → 0.4],
  {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



A mad rush to the lower left and the upper right.

If you want to see unit vectors only, you can go with the option
`ScaleFactor→Normalize`:

```
{xlow, xhigh} = {-3.0, 3.0};
{ylow, yhigh} = {-3.0, 3.0};
jump = 0.5;
unitflowplot = Table[Arrow[Field[x, y], Tail → {x, y},
    VectorColor → Blue, ScaleFactor → Normalize, HeadSize → 0.4],
  {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[unitflowplot, Axes → True, AxesLabel → {"x", "y"}];
```

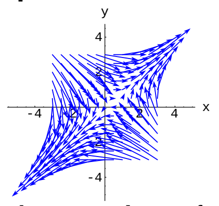


You are free to take your pick.

□B.1.a.ii) Floating corks

Stay with the flow plot:

```
Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



What's a good way of interpreting the flow plot?

□ **Answer:**

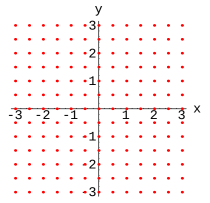
Look at what went into the plot.

The plot shows field vectors

Field[x, y] = {m[x, y], n[x, y]}

plotted with their tails at {x, y} for a selection of points {x, y}. The selection of points was:

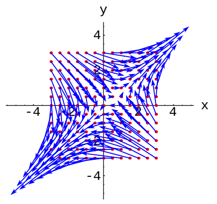
```
{xlow, xhigh} = {-3.0, 3.0};
{ylow, yhigh} = {-3.0, 3.0};
jump = 0.5;
pointplot = Table[Graphics[{Red, PointSize[0.015], Point[{x, y}]}],
  {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[pointplot, Axes → True, AxesLabel → {"x", "y"}];
```



The plot of the whole flow shows field vectors, Field[x, y], plotted with their tails at the points {x, y} shown above.

Take a look:

```
Show[pointplot, flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



Now you're ready to interpret the plot.

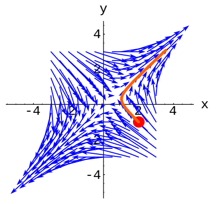
Think of the whole xy-plane as fluid flowing with currents and eddies.
The vectors represent the flow of the fluid at their tails.
The length of each vector indicates the speed of the fluid flow at its tail.
The direction of each vector indicates the direction of the flow.
In the plot above, lots of the fluid is flowing off to the lower left and lots of the fluid is flowing off to the upper right.
The upshot:
The field vector Field[x, y] with tail at {x, y} represents the speed and the direction with which a cork at {x, y} moves away from {x, y} as it is caught by the flow.

□B.1.a.iii) Trajectories in the flow

```
    This part will not run successfully unless all of the
instructions in parts i) and ii) above have been activated.
```

Stay with the same vector field as above and look at this plot:

```
{xstarter, ystarter} = {2.0, -1.0};
endtime = 3.2;
starterpoint = {xstarter, ystarter};
Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;

approxsolutions =
 NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[x, y, t];
{x[t_], y[t_]} =
 {x[t] /. approxsolutions⟦1⟧, y[t] /. approxsolutions⟦1⟧};

Clear[trajectory];
trajectory[t_] = {x[t], y[t]};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];

setup = Show[flowplot, starterplot, trajectoryplot,
  PlotRange → All, Axes → True, AxesLabel → {"x", "y"},
  DisplayFunction → $DisplayFunction];
```

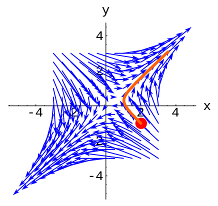


Some folks like to call the path you see a "trajectory" of the vector field. Other folks call this path a "streamline" of the vector field.
Just what is a trajectory in the field?

□**Answer:**

Take another look:

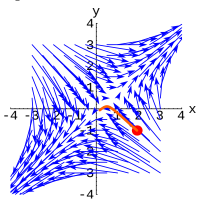
```
Show[setup];
```



This trajectory is part of the path of a cork thrown into the flow at the red dot.

See some more:

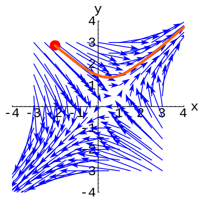
```
{xstarter, ystarter} =
 {Random[Real, {1.5, 3}], Random[Real, {-2, -1}]};
endtime = 3;
starterpoint = {xstarter, ystarter};

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
approxsolutions =
 NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[x, y, t];
{x[t_], y[t_]} =
 {x[t] /. approxsolutions⟦1⟧, y[t] /. approxsolutions⟦1⟧};

Clear[trajectory];
trajectory[t_] = {x[t], y[t]};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];

Show[flowplot, starterplot, trajectoryplot,
 PlotRange → {{xlow - 1, xhigh + 1}, {ylow - 1, yhigh + 1}}, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



```
Rerun several times.
```

And some more:

```
{xstarter, ystarter} = {Random[Real, {-2, -3}], Random[Real, {2, 3}]};
endtime = 7;
starterpoint = {xstarter, ystarter};

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
approxsolutions =
 NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[x, y, t];
{x[t_], y[t_]} =
 {x[t] /. approxsolutions⟦1⟧, y[t] /. approxsolutions⟦1⟧};

Clear[trajectory];
trajectory[t_] = {x[t], y[t]};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];

Show[flowplot, starterplot, trajectoryplot,
 PlotRange → {{xlow - 1, xhigh + 1}, {ylow - 1, yhigh + 1}}, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



```
Rerun several times.
```

Corks thrown into the flow at different points move on different paths.

□B.1.a.iv)

Here is the same vector field:

```
Clear[Field, m, n, x, y];
m[x_, y_] = 1.4 (y - 0.3);
n[x_, y_] = 1.2 (x - 0.5);

Field[x_, y_] = {m[x, y], n[x, y]}
{1.4 (-0.3 + y), 1.2 (-0.5 + x)}
```

How does this field govern the the shape of the trajectory of the cork dropped into the flow?

□Answer:

If the cork finds itself at a position {x, y}, then it has to go with the flow. The flow at {x, y} is in the direction of

Field[x, y] = {m[x, y], n[x, y]}:

```
Field[x, y]
{1.4 (-0.3 + y), 1.2 (-0.5 + x)}
{m[x, y], n[x, y]}
{1.4 (-0.3 + y), 1.2 (-0.5 + x)}
```

To go with the flow, the cork leaves {x, y} in the direction of the vector

Field[x, y] = {m[x, y], n[x, y]}.

As the cork progresses, it corrects its direction instantaneously at each point to keep on the course that the flow determines.

As you saw above, the result is that if {x, y} is any point on the actual path of the cork, then the vector

Field[x, y] = {m[x, y], n[x, y]}

with its tail at {x, y} is tangent to the actual path of the cork.

In this way, the vector field guides the cork as it moves along its path.

□B.1.a.v) Pairs of solutions of a system of diffeq's as parameterizations of the trajectories

Here is the same vector field:

```
Clear[Field, m, n, x, y];
m[x_, y_] = 1.4 (y - 0.3);
n[x_, y_] = 1.2 (x - 0.5);

Field[x_, y_] = {m[x, y], n[x, y]}
{1.4 (-0.3 + y), 1.2 (-0.5 + x)}
```

Use the given functions m[x, y] and n[x, y] to make this system of diffeq's:

```
diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
x′[t] == 1.4 (-0.3 + y[t])
y′[t] == 1.2 (-0.5 + x[t])
```

How are pairs of solutions {x[t], y[t]} of this system of diffeq's related to the trajectories in the flow?

□Answer:

Solutions {x[t], y[t]} of this system are parameterizations of the trajectories in the flow.

Reason: If at time t, the cork is at {x[t], y[t]}, then to go with the flow, the cork leaves {x[t], y[t]} in the direction of

Field[x[t], y[t]] = {m[x[t], y[t]], n [x[t], y[t]}

at the speed indicated by the length of

{m[x[t], y[t]], n [x[t], y[t]}.

On the other hand:

If at time t, the cork is at {x[t], y[t]}, then it leaves {x[t], y[t]} in the direction of

{x'[t], y'[t]}

at the speed indicated by the length of

{x'[t], y'[t]}.

This tells you that

{x'[t], y'[t]} = {m[x[t], y[t]], n [x[t], y[t]}.

This is the same as:

```
diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
x′[t] == 1.4 (-0.3 + y[t])
y′[t] == 1.2 (-0.5 + x[t])
```

Different starter data on {x[0], y[0]} lead to different trajectories.

B.2) Flow analysis of behavior of the solutions of the system

x'[t] = m[x[t], y[t]]

y'[t] = n[x[t], y[t]]

□B.2.a.i)

You enter a system of diffeq's by specifying two functions

m[x, y] and n[x, y]

like this:

```
Clear[m, n, x, y];
m[x_, y_] = -0.82 x - 0.17 y
n[x_, y_] = 0.50 x - 0.29 y
-0.82 x - 0.17 y
0.5 x - 0.29 y
```

And then you use these functions to make your diffeq system

x′[t] = m[x[t], y[t]]

y′[t] = n[x[t], y[t]]

with specified starting values for x[0] and y[0];

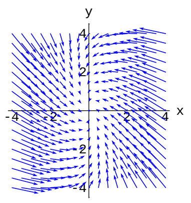
```
diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
x′[t] == -0.82 x[t] - 0.17 y[t]
y′[t] == 0.5 x[t] - 0.29 y[t]
```

Use flow plots to estimate what happens to the solutions {x[t], y[t]} as t grows and grows.

□Answer:

Go with a scaled plot of Field[x, y] = {m[x, y], n[x, y]}:

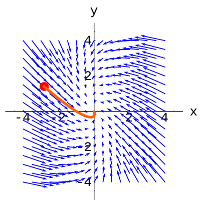
```
Clear[m, n, x, y];
m[x_, y_] = -0.82 x - 0.17 y;
n[x_, y_] = 0.50 x - 0.29 y;

Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
   VectorColor → Blue, ScaleFactor → 0.35, HeadSize → 0.25],
  {x, xlow, xhigh, 0.50}, {y, ylow, yhigh, 0.50}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



All the trajectories seem to be swirling in toward {0, 0}.

Check this out by dropping in a cork at a random point;

```
{xstarter, ystarter} =
 {Random[Real, {xlow, xhigh}], Random[Real, {ylow, yhigh}]};
endtime = 10;
starterpoint = {xstarter, ystarter};

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
approxsolutions =
 NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[x, y, t];
{x[t_], y[t_]} =
 {x[t] /. approxsolutions⟦1⟧, y[t] /. approxsolutions⟦1⟧};

Clear[trajectory];
trajectory[t_] = {x[t], y[t]};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.05], Point[starterpoint]}];

Show[flowplot, starterplot, trajectoryplot,
 PlotRange → {{xlow - 1, xhigh + 1}, {ylow - 1, yhigh + 1}}, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



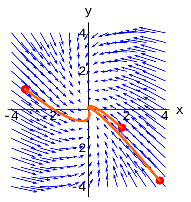
Rerun a couple of times.

Yep, the trajectories wind their way to {0, 0}.

See three more trajectories:

```
{xstarter1, ystarter1} =
 {Random[Real, {xlow, xhigh}], Random[Real, {ylow, 0}]};
{xstarter2, ystarter2} =
 {Random[Real, {xlow, 0}], Random[Real, {0, yhigh}]};
{xstarter3, ystarter3} =
 {Random[Real, {0, xhigh}], Random[Real, {ylow, 0}]};
endtime = 8;
starterpoint1 = {xstarter1, ystarter1};
starterpoint2 = {xstarter2, ystarter2};
starterpoint3 = {xstarter3, ystarter3};

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn1 = x[0] == xstarter1;
ystartereqn1 = y[0] == ystarter1;
xstartereqn2 = x[0] == xstarter2;
ystartereqn2 = y[0] == ystarter2;
xstartereqn3 = x[0] == xstarter3;
ystartereqn3 = y[0] == ystarter3;
ndssol1 = NDSolve[{xdiffeq, ydiffeq, xstartereqn1, ystartereqn1},
  {x[t], y[t]}, {t, 0, endtime}];
ndssol2 = NDSolve[{xdiffeq, ydiffeq, xstartereqn2, ystartereqn2},
  {x[t], y[t]}, {t, 0, endtime}];
ndssol3 = NDSolve[{xdiffeq, ydiffeq, xstartereqn3, ystartereqn3},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[x1, x2, x3, y1, y2, y3];
{x1[t_], y1[t_]} = {x[t] /. ndssol1⟦1⟧, y[t] /. ndssol1⟦1⟧};
{x2[t_], y2[t_]} = {x[t] /. ndssol2⟦1⟧, y[t] /. ndssol2⟦1⟧};
{x3[t_], y3[t_]} = {x[t] /. ndssol3⟦1⟧, y[t] /. ndssol3⟦1⟧};
```

```
Clear[trajectory1, trajectory2];
trajectory1[t_] = {x1[t], y1[t]};
trajectory2[t_] = {x2[t], y2[t]};
trajectory3[t_] = {x3[t], y3[t]};
trajectoryplots =
 ParametricPlot[{trajectory1[t], trajectory2[t], trajectory3[t]},
  {t, 0, endtime}, PlotStyle → {{CadmiumOrange, Thickness[0.015]}},
  DisplayFunction → Identity];
starterplots =
 {Graphics[{Red, PointSize[0.05], Point[starterpoint1]}],
  Graphics[{Red, PointSize[0.05], Point[starterpoint2]}],
  Graphics[{Red, PointSize[0.05], Point[starterpoint3]}]};

Show[
 flowplot, starterplots, trajectoryplots, PlotRange → All, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```

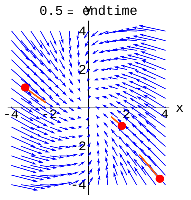


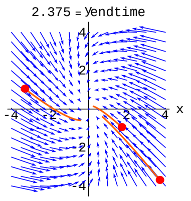
Rerun several times.

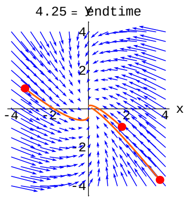
The trajectories are sucked toward {0, 0}.

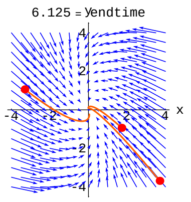
See corks flow on the last trajectories:

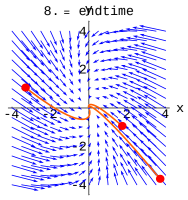
```
Clear[trajplots, s];
trajplots[s_] :=
 ParametricPlot[{{x1[t], y1[t]}, {x2[t], y2[t]}, {x3[t], y3[t]}},
  {t, 0, s}, PlotStyle → {{Thickness[0.01], CadmiumOrange}},
  DisplayFunction → Identity];

Table[Show[flowplot, trajplots[s],
  starterplots, Axes → True, PlotLabel → s "= endtime",
  AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction],
 {s, 0.5, endtime, (endtime - 0.5)/4}];
```











```
Grab the last group of plots and and animate slowly.
```

The corks are sucked into {0, 0}.

The visual evidence is that when you go with:

```
diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == -0.82 x[t] - 0.17 y[t]
y′[t] == 0.5 x[t] - 0.29 y[t]
```

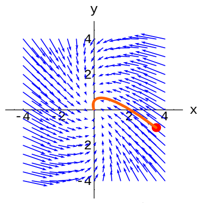
then when t gets big, both x[t] and y[t] get close to 0.

□B.2.a.ii)

Stay with the same setup as in part i) immediately above and look at this trajectory shown with the flow plot:

```
{xstarter, ystarter} = {3.5, -1};
endtime = 10;
starterpoint = {xstarter, ystarter};

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
ndssol = NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[x, y, t];
{x[t_], y[t_]} = {x[t] /. ndssol⟦1⟧, y[t] /. ndssol⟦1⟧};

Clear[trajectory];
trajectory[t_] = {x[t], y[t]};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
```

```
starterplot = Graphics[{Red, PointSize[0.05], Point[starterpoint]}];

Show[flowplot, starterplot, trajectoryplot,
 PlotRange → {{xlow - 1, xhigh + 1}, {ylow - 1, yhigh + 1}}, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



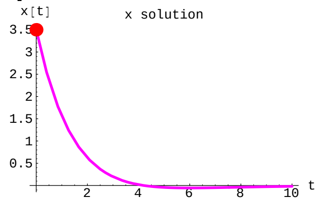
This is a plot of the trajectory starting at the indicated point.
Plot the individual solutions x[t] and y[t] corresponding to the given starter data.

□**Answer:**

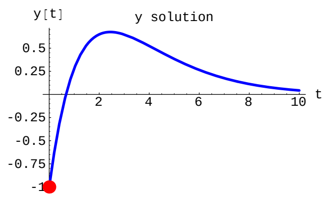
The solutions come in pairs {x[t], y[t]} for this trajectory.

Here are the individual solution plots:

```
xplot =
 Plot[x[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Magenta}},
  AxesLabel → {"t", "x[t]"}, PlotRange → All,
  AspectRatio → 1/GoldenRatio, PlotLabel → "x solution",
  Epilog → {Red, PointSize[0.05], Point[{0, xstarter}]}];
```

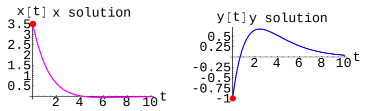


```
yplot =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Blue}},
  AxesLabel → {"t", "y[t]"}, PlotRange → All,
  AspectRatio → 1/GoldenRatio, PlotLabel → "y solution",
  Epilog → {Red, PointSize[0.05], Point[{0, ystarter}]}];
```



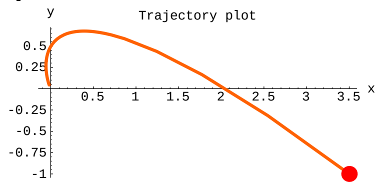
See them side-by-side:

```
Show[GraphicsArray[{xplot, yplot}]];
```



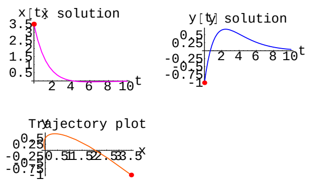
The corresponding trajectory {x[t], y[t]} plots out this way:

```
trajectoryplot =
 ParametricPlot[{x[t], y[t]}, {t, 0, endtime}, PlotStyle →
   {{Thickness[0.01], CadmiumOrange}}, AxesLabel → {"x", "y"},
  PlotRange → All, PlotLabel → "Trajectory plot",
  Epilog → {Red, PointSize[0.05], Point[{x[0], y[0]}]}];
```



Here are plots of x[t], and y[t], and of the trajectory, {x[t], y[t]} all at once:

```
Show[GraphicsArray[{{xplot, yplot}, {trajectoryplot}}]];
```



The pair {x[t], y[t]} IS the solution.

The trajectory IS NOT the solution, but it does do a good job of showing off what the solutions do.

You can see from the trajectory plot that the corresponding x[t] solution plot goes down as time t advances.

You can also see from the trajectory plot that the corresponding y[t] solution plot goes up and then down as time t advances.

□B.2.a.iii)

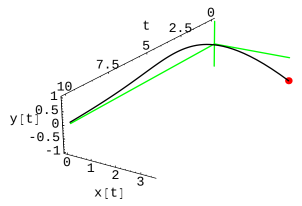
Keep everything the same as immediately above.
Can you see the solution plots and the trajectory all in one plot?

□**Answer:**

If you are willing to look at a 3D plot, then you can.

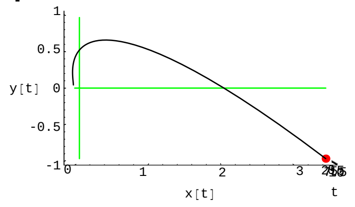
Here you go:

```
threeDstarterpoint = {0, 3.5, -1};
threeDstarterplot =
 Graphics3D[{Red, PointSize[0.03], Point[threeDstarterpoint]}];
allthree = ParametricPlot3D[{t, x[t], y[t]}, {t, 0, endtime},
  AxesLabel → {"t", "x[t]", "y[t]"}, DisplayFunction → Identity];
taxis = Graphics3D[{Green, Line[{{0, 0, 0}, {endtime, 0, 0}}]}];
xaxis = Graphics3D[{Green, Line[{{0, 0, 0}, {0, 3.5, 0}}]}];
yaxis = Graphics3D[{Green, Line[{{0, 0, -1}, {0, 0, 1}}]}];

everything = Show[allthree, threeDstarterplot, taxis, xaxis,
  yaxis, Boxed → False, PlotRange → All, ViewPoint → CMView,
  DisplayFunction → $DisplayFunction];
```



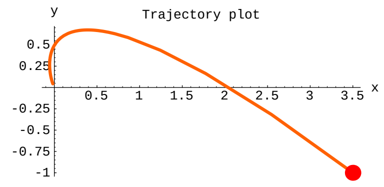
To see the trajectory plot, look at this 3D curve from a view point down the t - axis:

```
Show[everything, ViewPoint → {25, 0, 0}];
```



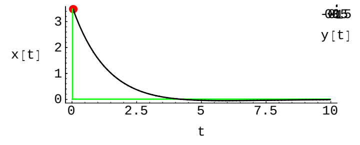
Compare:

```
Show[trajectoryplot];
```



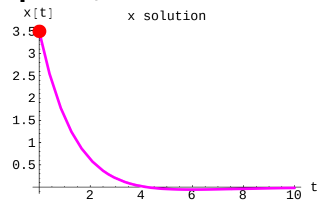
To see the x[t] solution plot, look at the 3D curve from a view point down the y - axis:

```
Show[everything, ViewPoint → {0, 0, 25}];
```



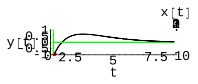
Compare:

```
Show[xplot];
```



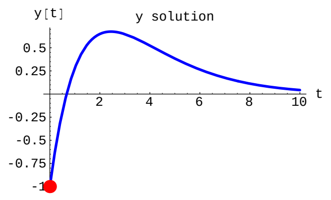
To see the y[t] solution plot, look at the 3 D curve from a view point down the negative x- axis:

```
Show[everything, ViewPoint → {0, -25, 0}];
```



Compare:

```
Show[yplot];
```



□B.2.b.i) Oscillations

Here's a new system of differential equations

x′[t] = 1.8 y[t]

y′[t] = −3.2 x[t]

with given starter data on x[0] and y[0].

Do a flow analysis to predict what happens to the solutions {x[t], y[t]} as t grows and grows.

□**Answer:**

Look at the system:

x′[t] = 1.8 y[t]

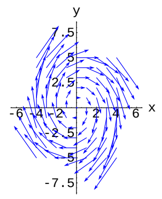
y′[t] = −3.2 x[t].

Enter the corresponding functions m[x, y] and n[x, y] this way:

```
Clear[m, n, x, y, t, starterx, startery];
m[x_, y_] = 1.8 y;
n[x_, y_] = -3.2 x;

diffeqsystem = {x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]};
 diffeqsystem
x'[t] == 1.8 y[t]
y'[t] == -3.2 x[t]
```

Now that you have entered m[x, y] and n[x, y], you can look at a scaled plot of Field[x, y] = {m[x, y], n[x, y]}:

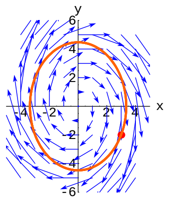
```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-5, 5};
jump = 1;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.25, HeadSize → 0.5],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
{1.8 y, -3.2 x}
```



All the trajectories seem to be swirling around {0, 0}.

Check this out by dropping in a cork at {3, −2};

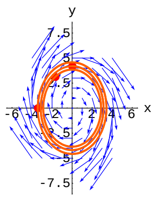
```
endtime = 4;
{xstarter, ystarter} = {3, -2};
endtime = 10;
starterpoint = {xstarter, ystarter};

Clear[x, y, t];
xdiffeq = x'[t] == m[x[t], y[t]];
ydiffeq = y'[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
ndssol = NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
   {x[t], y[t]}, {t, 0, endtime}];

Clear[x, y, t];
{x[t_], y[t_]} = {x[t] /. ndssol[[1]], y[t] /. ndssol[[1]]};

Clear[trajectory];
trajectory[t_] = {x[t], y[t]};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.05], Point[starterpoint]}];

Show[flowplot, starterplot, trajectoryplot,
 PlotRange → {{xlow - 1, xhigh + 1}, {ylow - 1, yhigh + 1}}, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



Yep, the trajectory swirls around {0, 0}

See three more trajectories starting at some partially random points:

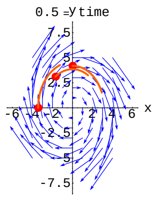
```
{xstarter1, ystarter1} = {Random[Real, {-6, -2}], 0.0};
{xstarter2, ystarter2} = {-1.7, Random[Real, {0, 6}]};
{xstarter3, ystarter3} = {0.0, Random[Real, {1, 6}]};
endtime = 5;
starterpoint1 = {xstarter1, ystarter1};
starterpoint2 = {xstarter2, ystarter2};
starterpoint3 = {xstarter3, ystarter3};

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn1 = x[0] == xstarter1;
ystartereqn1 = y[0] == ystarter1;
xstartereqn2 = x[0] == xstarter2;
ystartereqn2 = y[0] == ystarter2;
xstartereqn3 = x[0] == xstarter3;
ystartereqn3 = y[0] == ystarter3;
approxsolutions1 =
 NDSolve[{xdiffeq, ydiffeq, xstartereqn1, ystartereqn1},
  {x[t], y[t]}, {t, 0, endtime}];
approxsolutions2 =
 NDSolve[{xdiffeq, ydiffeq, xstartereqn2, ystartereqn2},
  {x[t], y[t]}, {t, 0, endtime}];
approxsolutions3 =
 NDSolve[{xdiffeq, ydiffeq, xstartereqn3, ystartereqn3},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[x1, x2, x3, y1, y2, y3];
{x1[t_], y1[t_]} =
 {x[t] /. approxsolutions1⟦1⟧, y[t] /. approxsolutions1⟦1⟧};
{x2[t_], y2[t_]} =
 {x[t] /. approxsolutions2⟦1⟧, y[t] /. approxsolutions2⟦1⟧};
{x3[t_], y3[t_]} =
 {x[t] /. approxsolutions3⟦1⟧, y[t] /. approxsolutions3⟦1⟧};

Clear[trajectory1, trajectory2];
trajectory1[t_] = {x1[t], y1[t]};
trajectory2[t_] = {x2[t], y2[t]};
trajectory3[t_] = {x3[t], y3[t]};
trajectoryplots =
 ParametricPlot[{trajectory1[t], trajectory2[t], trajectory3[t]},
  {t, 0, endtime}, PlotStyle → {{CadmiumOrange, Thickness[0.015]}},
  DisplayFunction → Identity];
starterplots =
 {Graphics[{Red, PointSize[0.06], Point[starterpoint1]}],
  Graphics[{Red, PointSize[0.06], Point[starterpoint2]}],
  Graphics[{Red, PointSize[0.06], Point[starterpoint3]}]};

Show[
 flowplot, starterplots, trajectoryplots, PlotRange → All, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```

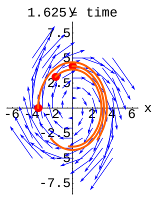


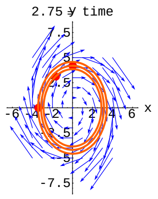
The trajectories swirl around {0, 0}.

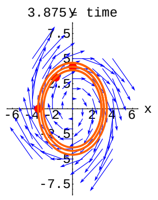
See them move as time advances:

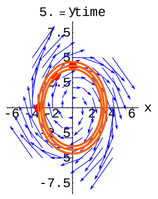
```
Clear[trajplots, s];
trajplots[s_] :=
 ParametricPlot[{{x1[t], y1[t]}, {x2[t], y2[t]}, {x3[t], y3[t]}},
  {t, 0, s}, PlotStyle → {{Thickness[0.015], CadmiumOrange}},
  DisplayFunction → Identity];

Table[Show[flowplot, starterplots,
  trajplots[s], Axes → True, PlotLabel → s "= time",
  AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction],
 {s, 0.5, endtime, (endtime - 0.5)/4}];
```











```
Grab the last group of plots and and animate slowly.
```

The corks swirl around {0, 0}.

Once it finishes one cycle, it repeats its trip over and over.

Play with some other starting points.

The visual evidence is that when you go with:

```
Clear[m, n, x, y, t, starterx, startery];
m[x_, y_] = 1.8 y;
n[x_, y_] = -3.2 x;

diffeqsystem = {x′[t], y′[t]} == {m[x[t], y[t]], n[x[t], y[t]]};
diffeqsystem
```

```
x′[t] == 1.8 y[t]
y′[t] == -3.2 x[t]
```

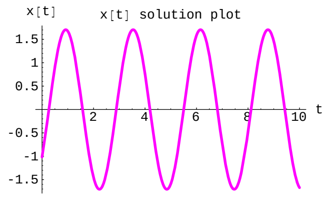
Then when t gets bigger and bigger, all trajectories swirl around {0, 0} over and over.

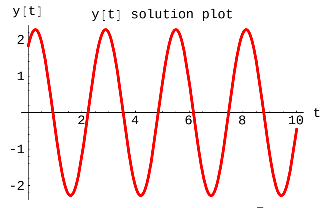
This means that plots of the individual solutions x[t] and y[t] oscillate like sine waves.

Here's the plot of the solution x[t] corresponding to some random starting data on {x[0], y[0]} followed by the plot of the companion solution y[t]:

```
endtime = 8;
{xstarter, ystarter} =
 {Random[Real, {xlow, xhigh}], Random[Real, {ylow, yhigh}]};
endtime = 10;
```

```
starterpoint = {xstarter, ystarter};

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
approxsolutions =
 NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[x, y, t];
{x[t_], y[t_]} =
 {x[t] /. approxsolutions[[1]], y[t] /. approxsolutions[[1]]};

xplot = Plot[x[t], {t, 0, endtime},
  PlotStyle → {{Magenta, Thickness[0.01]}}, AxesLabel → {"t", "x[t]"},
  AspectRatio → 1/GoldenRatio, PlotLabel → "x[t] solution plot"];

yplot = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Red, Thickness[0.01]}}, AxesLabel → {"t", "y[t]"},
  AspectRatio → 1/GoldenRatio, PlotLabel → "y[t] solution plot"];
```

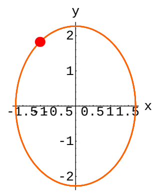




Rerun many times.

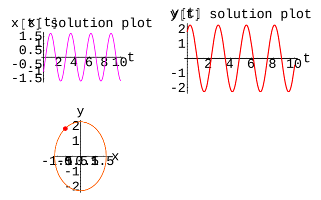
Here's the trajectory {x[t], y[t]} with its starter point:

```
trajplot = ParametricPlot[{x[t], y[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.01], CadmiumOrange}},
  AxesLabel → {"x", "y"}, DisplayFunction → Identity];

trajectory = Show[trajplot,
  Graphics[{PointSize[0.08], Red, Point[starterpoint]}],
  DisplayFunction → $DisplayFunction];
```



See them all together:

```
Show[GraphicsArray[{{xplot, yplot}, {trajectory}}]];
```



☐B.2.b.ii)

Stay with the same diffeq system as in part i):

```
Clear[m, n, x, y, t, starterx, startery];
m[x_, y_] = 1.8 y;
n[x_, y_] = -3.2 x;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
x′[t] == 1.8 y[t]
y′[t] == -3.2 x[t]
```

If you want a plot of only y[t], then can you get by with given starter data on y[0] without any given starter data on x[0]?

☐**Answer:**

No way.

You cannot get by with given starter data on y[0] without any given starter data on x[0].

Reason:

The solution plots come from the trajectories and in order to get a trajectory you have to know where the cork was thrown into the flow.

To have any chance of doing this, you need starter data on both x[0] and y[0].

B.3) Flow analysis of the unforced linearoscillator differential equation by converting it to a system of two first order differential equations

☐B.3.a.i) The undamped linear oscillator differential equation

Here's a prototype of the undamped linear oscillator differential equation:

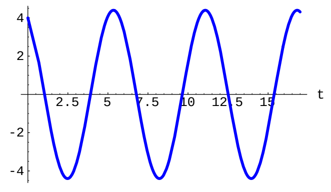
```
Clear[t, y];
oscillatordiffeq = (y''[t] + 1.2 y[t] == 0)
1.2 y[t] + y″[t] == 0
```

```
  Little second order differential equations like this one pop up
           almost everywhere in science and engineeering.
                   You met them first in DE.02.
```

See a sample solution:

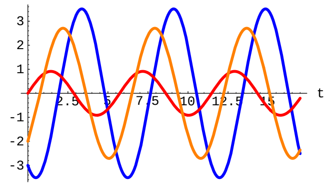
```
Clear[y1];
endtime = 17;
y1[t_] = y[t] /. NDSolve[{oscillatordiffeq,
      y[0] == 4, y′[0] == -2}, y[t], {t, 0, endtime}][[1]];
```

```
Plot[y1[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Blue}},
 AspectRatio → 1/GoldenRatio, PlotRange → All, AxesLabel → {"t", ""}];
```



It oscillates just as a sine wave oscillates.
See three more sample solutions:

```
Clear[y1, y2, y3, y, t];
y1[t_] = y[t] /. NDSolve[{oscillatordiffeq,
      y[0] == -3, y′[0] == -2}, y[t], {t, 0, endtime}][[1]];
y2[t_] = y[t] /. NDSolve[{oscillatordiffeq,
      y[0] == 0, y′[0] == 1}, y[t], {t, 0, endtime}][[1]];
y3[t_] = y[t] /. NDSolve[{oscillatordiffeq,
      y[0] == -2, y′[0] == 2}, y[t], {t, 0, endtime}][[1]];

solutionplots = Plot[{y1[t], y2[t], y3[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Blue},
   {Thickness[0.01], Red}, {Thickness[0.01], Orange}},
  AspectRatio → 1/GoldenRatio, PlotRange → All,
  AxesLabel → {"t", ""}];
```



They all oscillate just the way a sine wave oscillates.

□B.3.a.ii) Starter data on y[0] and y′[0]

Go back and look at the code that produced these solutions and notice that starter data on y[0] and y′[0] are given for each solution. Why is this natural?

□**Answer:**

To see why it is as natural as apple pie, you just make this oscillator differential equation into a system of two first order differential equations.

To do this you make up a new function. Call it x[t], and set

x[t] = y′[t].

And then:

→ Every time you see a y′[t] in the differential equation, replace it with an x[t].

→ Every time you see a y″[t] in the equation, replace it with an x′[t].

Take a look.

```
Clear[x];
ColumnForm[Thread[{new = (y'[t] == x[t]),
oscillatordiffeq /. {y'[t] -> x[t], y''[t] -> x'[t]}}]]
```

```
y′[t] == x[t]
1.2 y[t] + x′[t] == 0
```

Clean this up by putting the x′[t] on the left and the y[t] on the right:

```
Clear[m, n, x, y, t, starterx, startery];
m[x_, y_] = - 1.2 y;
n[x_, y_] = x;

(diffeqsystem = {x'[t] == m[x[t], y[t]], y'[t] == n[x[t], y[t]]}) //
 ColumnForm
```

```
x′[t] == -1.2 y[t]
y′[t] == x[t]
```

This system is exactly the same as:

```
oscillatordiffeq
```

```
1.2 y[t] + y″[t] == 0
```

Any y[t] that solves the diffeq system also solves the oscillator diffeq.

To nail down a y[t] for the system, you need starter data on x[0] and y[0].

But because

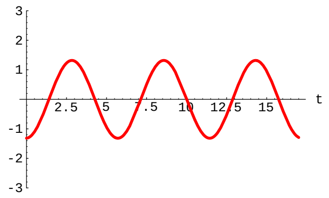
x[t] = y′[t],

the starter data on x[0] is actually starter data on y′[0].

That's why you need starter data on both y[0] and y′[0] to nail down a solution of the oscillator diffeq.

Play by going with random starter data on y[0] and y′[0] and re-running several times:

```
{startery, starteryprime} =
 {Random[Real, {-2, 2}], Random[Real, {-2, 2}]};
Clear[y1];
endtime = 17;

y1[t_] = y[t] /. NDSolve[
{oscillatordiffeq, y[0] == startery, y'[0] == starteryprime},
    y[t], {t, 0, endtime}][[1]];

Plot[y1[t], {t, 0, endtime}, PlotStyle ->
 {{Thickness[0.01], Red}}, PlotRange -> {-3, 3},
 AspectRatio -> 1/GoldenRatio, PlotRange -> All, AxesLabel -> {"t", ""}];
```



□B.3.a.iii) Converting the unforced undamped oscillator diffeq to a system to see why solutions must oscillate like sine waves

Here's another undamped, unforced oscillator diffeq:

```
Clear[t, y];
oscillatordiffeq = (y''[t] + 2.2 y[t] == 0)
```

```
2.2 y[t] + y″[t] == 0
```

Explain how you can use a flow plot to tell that all solutions of this diffeq oscillate like sine waves.

□**Answer:**

Convert it to a system and look at a flow plot:

```
Clear[x];
ColumnForm[Thread[{new = (y'[t] == x[t]),
oscillatordiffeq /. {y'[t] -> x[t], y''[t] -> x'[t]}}]]
```

```
y′[t] == x[t]
2.2 y[t] + x′[t] == 0
```

Clean this up:

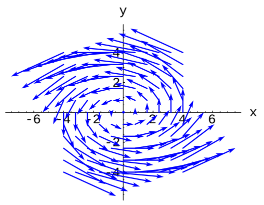
```
Clear[m, n, x, y, t, starterx, startery];
m[x_, y_] = - 2.2 y;
n[x_, y_] = x;

(diffeqsystem = {x'[t] == m[x[t], y[t]], y'[t] == n[x[t], y[t]]}) //
 ColumnForm
```

```
x′[t] == -2.2 y[t]
y′[t] == x[t]
```

See the flow:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.8;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.4, HeadSize → 0.5],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

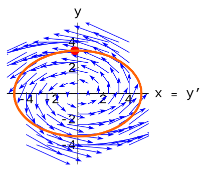
```
{-2.2 y, x}
```



Your eyes tell you that the trajectories swirl around {0, 0}. On a specific trajectory, y[t] goes up and down over and over. That's why the oscillator oscillates.

See a random trajectory:

```
endtime = 4;
{xstarter, ystarter} =
 {Random[Real, {xlow, xhigh}], Random[Real, {ylow, yhigh}]};
endtime = 10;
starterpoint = {xstarter, ystarter};

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
ndssol = NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
   {x[t], y[t]}, {t, 0, endtime}];

Clear[x, y, t];
{x[t_], y[t_]} = {x[t] /. ndssol⟦1⟧, y[t] /. ndssol⟦1⟧};

Clear[trajectory];
trajectory[t_] = {x[t], y[t]};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];

Show[flowplot, starterplot, trajectoryplot, PlotRange →
   {{xlow - 1.5, xhigh + 1.5}, {ylow - 1.5, yhigh + 1.5}}, Axes → True,
 AxesLabel → {"x = y'", "y"}, DisplayFunction → $DisplayFunction];
```

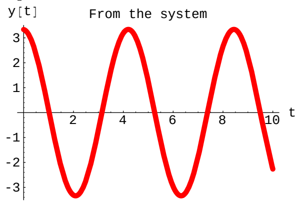


```
Rerun a couple of times.
```

See a solution y[t] of the linear oscillator diffeq oscillate up and down.

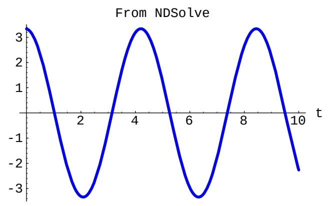
Check it out:

```
yfromsystem =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Red, Thickness[0.02]}},
  AxesLabel → {"t", "y[t]"}, PlotLabel → "From the system"];
```



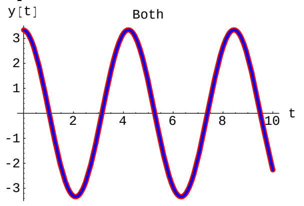
Compare with a direct **NDSolve**:

```
Clear[y1, y];
y1[t_] = y[t] /. NDSolve[{oscillatordiffeq,
     y[0] == ystarter, y′[0] == xstarter}, y[t], {t, 0, endtime}][[1]];

directplot = Plot[y1[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Blue}}, AspectRatio → 1/GoldenRatio,
  PlotRange → All, AxesLabel → {"t", ""}, PlotLabel → "From NDSolve"];
```



See them both:

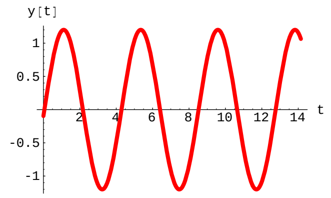
```
Show[yfromsystem, directplot, PlotLabel → "Both"];
```



Dead ringers. And this is no accident. It will happen everytime.

Check out other solutions corresponding to random starting data on y[0] and y′[0]:

```
{startery, starteryprime} =
 {Random[Real, {xlow, xhigh}], Random[Real, {ylow, yhigh}]};

Clear[y, t, approxy];
endtime = Random[Real, {10, 25}];

Clear[y1];
y1[t_] = y[t] /. NDSolve[{oscillatordiffeq, y[0] == startery,
     y′[0] == starteryprime}, y[t], {t, 0, endtime}][[1]];

Plot[y1[t], {t, 0, endtime},
 PlotStyle → {{Thickness[0.015], Red}}, AspectRatio → 1/GoldenRatio,
 PlotRange → All, AxesLabel → {"t", "y[t]"}];
```



Rerun until you feel good.

□B.3.b) The damped unforced oscillator

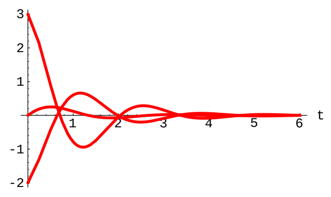
You get a damped unforced oscillator by inserting a b y′[t] term (with b > 0) into the oscillator diffeq:

```
Clear[t, y];
b = 1.8;
oscillatordiffeq = (y''[t] + b y'[t] + 6.4 y[t] == 0)
```

```
6.4 y[t] + 1.8 y′[t] + y″[t] == 0
```

Take a look at some sample solutions coming from different starter data on y[0] and y′[0]:

```
Clear[y1, y2, y3, y, t];
endtime = 6;
y1[t_] = y[t] /. NDSolve[{oscillatordiffeq,
     y[0] == 3, y′[0] == -2}, y[t], {t, 0, endtime}][[1]];
y2[t_] = y[t] /. NDSolve[{oscillatordiffeq,
     y[0] == 0, y′[0] == 1}, y[t], {t, 0, endtime}][[1]];
y3[t_] = y[t] /. NDSolve[{oscillatordiffeq,
     y[0] == -2, y′[0] == 2}, y[t], {t, 0, endtime}][[1]];

solutionplots = Plot[{y1[t], y2[t], y3[t]}, {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, AspectRatio → 1/GoldenRatio,
  PlotRange → All, AxesLabel → {"t", ""}];
```



These sample solutions look like damped (squished) sine waves. They try to oscillate but are squashed out as time goes on.
Convert this damped oscillator diffeq into a diffeq system and use the resulting flow plot of the system to explain the behavior you see above.

□ Answer:

Convert it to a system by replacing all y′[t]'s with x[t] and all y″[t]'s by x′[t]:

```
Clear[x];
ColumnForm[Thread[{new = (y'[t] == x[t]),
oscillatordiffeq /. {y'[t] -> x[t], y''[t] -> x'[t]}}]]
```

```
y′[t] == x[t]
1.8 x[t] + 6.4 y[t] + x′[t] == 0
```

Enter the correct m[x, y] and n[x, y] and clean this up:

```
Clear[m, n, x, y, t, starterx, startery];
m[x_, y_] = - 6.4 y - 1.9 x;
n[x_, y_] = x;

(diffeqsystem = {x'[t] == m[x[t], y[t]], y'[t] == n[x[t], y[t]]}) //
 ColumnForm
```

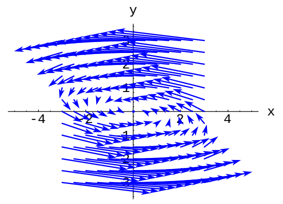
```
x′[t] == -1.9 x[t] - 6.4 y[t]
y′[t] == x[t]
```

Look at the flow plot:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
{xlow, xhigh} = {-3, 3};
{ylow, yhigh} = {-3, 3};
jump = (xhigh - xlow)/12;
```

```
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.15, HeadSize → 0.4],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

```
{-1.9 x - 6.4 y, x}
```



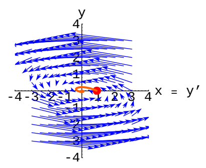
Your eyes tell you that the trajectories swirl TOWARD {0, 0}.

On a specific trajectory, y[t] goes up and down over and over but it gets closer and closer to 0 as time goes on.

That's why the damped oscillator oscillates while it is being squashed. See a random trajectory:

```
endtime = 9;
{xstarter, ystarter} =
 {Random[Real, {xlow, xhigh}], Random[Real, {ylow, yhigh}]};
endtime = 10;
starterpoint = {xstarter, ystarter};

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
ndssol = NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
   {x[t], y[t]}, {t, 0, endtime}];

Clear[x, y, t];
{x[t_], y[t_]} = {x[t] /. ndssol⟦1⟧, y[t] /. ndssol⟦1⟧};
```

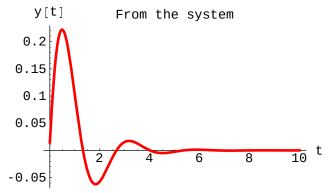
```
Clear[trajectory];
trajectory[t_] = {x[t], y[t]};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
    {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];

Show[flowplot, starterplot, trajectoryplot,
 PlotRange → {{xlow - 1, xhigh + 1}, {ylow - 1, yhigh + 1}}, Axes → True,
 AxesLabel → {"x = y'", "y"}, DisplayFunction → $DisplayFunction];
```



```
Rerun a couple of times.
```

See the y[t] coming from the last plot oscillate up and down on its way to 0:

```
yplot =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Red, Thickness[0.01]}},
  AxesLabel → {"t", "y[t]"}, PlotRange → All,
  AspectRatio → 1/GoldenRatio, PlotLabel → "From the system"];
```



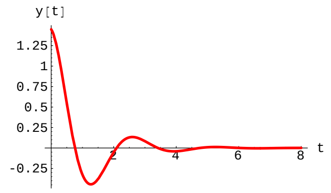
Check out other solutions of

```
Clear[y];
 oscillatordiffeq
```

```
6.4 y[t] + 1.8 y′[t] + y″[t] == 0
```

corresponding to random starting data on y[0] and y′[0]:

```
{starterY, starteryprime} =
 {Random[Real, {-3, 3}], Random[Real, {-3, 3}]};

Clear[y, t, approxy];
endtime = 8;
Clear[y1, y]
y1[t_] = y[t] /. NDSolve[{oscillatordiffeq, y[0] == startery,
     y′[0] == starteryprime}, y[t], {t, 0, endtime}]⟦1⟧;

Plot[y1[t], {t, 0, endtime},
 PlotStyle → {{Thickness[0.01], Red}}, AspectRatio → 1/GoldenRatio,
 PlotRange → All, AxesLabel → {"t", "y[t]"}];
```



Rerun until you are in the swing.

B.4) The danger zone: Where your cork flow intuition can be blown to bits

☐B.4.a) Weirdness in the Danger Zone

Here's a little system:

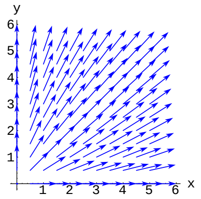
```
Clear[m, n, x, y, t];
m[x_, y_] = x^(2/3);
n[x_, y_] = y^(2/3) ;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

$x'[t] == x[t]^{2/3}$

$y'[t] == y[t]^{2/3}$

And a flow plot:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
scalefactor = 0.2;
{xlow, xhigh} = {0, 5};
{ylow, yhigh} = {0, 5};
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → Normalize, HeadSize → 0.3],
   {x, xlow, xhigh, 0.5}, {y, ylow, yhigh, 0.5}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



Here are three formulas for pairs of solutions popping out of {0, 0}:

```
Clear[x1, y1, x2, y2, x3, y3, t];
{x1[t_], y1[t_]} = {t^3/27, t^3/27}

{x2[t_], y2[t_]} = {t^3/27, 0}

{x3[t_], y3[t_]} = {0, t^3/27}
```

$\left\{\frac{t^3}{27}, \frac{t^3}{27}\right\}$

$\left\{\frac{t^3}{27}, 0\right\}$

$\left\{0, \frac{t^3}{27}\right\}$

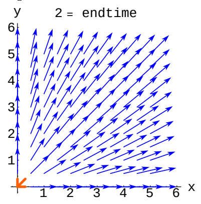
Short hand calculations show each pair solves:

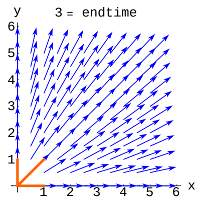
```
diffeqsystem
```

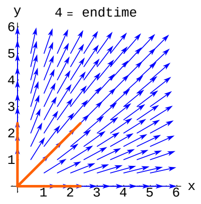
$\{x'[t], y'[t]\} == \{x[t]^{2/3}, y[t]^{2/3}\}$

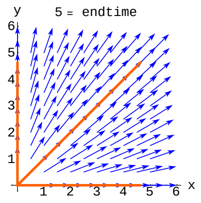
Watch their trajectories go with the flow:

```
Clear[trajplots, endtime];
trajplots[endtime_] :=
 ParametricPlot[{{x1[t], y1[t]}, {x2[t], y2[t]}, {x3[t], y3[t]}},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.015], CadmiumOrange}},
  DisplayFunction → Identity];

Table[Show[flowplot, trajplots[endtime],
  Axes → True, PlotLabel → endtime "= endtime",
  AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction],
 {endtime, 2, 5, 1}];
```









```
Grab all three plots and animate.
```

Wait a minute!
You drop the cork into the flow at {0, 0} and it floats off in three different directions simultaneously!
Doesn't this shoot to hell all the intuition you get by thinking of trajectories as corks going with the flow?

□ **Answer:**

Not if you are aware of the strange things that can and sometimes do happen in the danger zone.

To locate the danger zone, look again at the setup:

```
Clear[m, n, x, y, t];
m[x_, y_] = x^(2/3);
n[x_, y_] = y^(2/3);

DEsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[DEsystem]]
```

$x'[t] == x[t]^{2/3}$
$y'[t] == y[t]^{2/3}$

Take a look at the gradients of m[x, y] and n[x, y]:

```
Clear[gradm];
gradm[x_, y_] = {∂x m[x, y], ∂y m[x, y]}
```

$\left\{\frac{2}{3\,x^{1/3}},\ 0\right\}$

```
Clear[gradn];
gradn[x_, y_] = {∂x n[x, y], ∂y n[x, y]}
```

$\left\{0,\ \frac{2}{3\,y^{1/3}}\right\}$

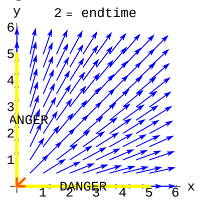
The danger zone consists of the singularities (blow-ups) of gradm[x, y] or the singularities (blow-ups) of gradn[x, y].

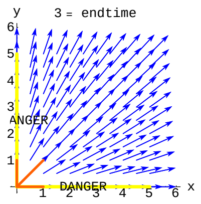
By inspection, you can see that the danger zone consists of the lines

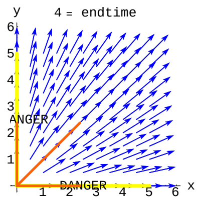
x = 0 and y = 0.

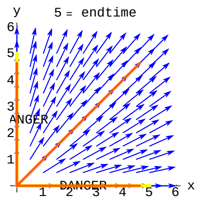
See the three trajectories plotted with the danger zone:

```
dangerzone =
 {Graphics[{Yellow, Thickness[0.02], Line[{{0, 0}, {5, 0}}]}],
  Graphics[{Yellow, Thickness[0.02], Line[{{0, 0}, {0, 5}}]}]};
dangerlabel = {Graphics[Text["DANGER", {5/2, 0}]],

  Graphics[Text["  DANGER", {0, 5/2}]]};

Table[Show[flowplot, flowplot, dangerzone, dangerlabel,
  trajplots[endtime], Axes → True, PlotLabel → endtime "= endtime",
  AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction],
 {endtime, 2, 5, 1}];
```









All three solutions start in the danger zone and two of them never leave the danger zone.

Morals:

Whenever you start a trajectory in a danger zone you should be on the lookout for something that violates your intuition.

Anytime your trajectory enters a danger zone, expect the unexpected.

□B.4.b)

Try to explain why the gradient test for the danger zone works

□ **Answer:**

Go with a cleared diffeq system:

```
Clear[diffeqsystem, x, y, m, n, t];
DEsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[DEsystem]]
```

$x'[t] == m[x[t], y[t]]$
$y'[t] == n[x[t], y[t]]$

Because

$$x'[t] = m[x[t], y[t]]$$

and

$$y'[t] = n[x[t], y[t]]$$

the chain rule from vector calc guarantees that x″[t] = gradm[x[t], y[t]] . {x′[t], y′[t]}.

y″[t] = gradn[x[t], y[t]] . {x′[t], y′[t]}.

```
The periods, above and below, represent dot products.
```

Now here's the key point:

If a trajectory {x[t], y[t]} hits the danger zone (i.e. a singularity of gradm[x, y] or of gradn[x, y]), at a time t = tdanger, then the facts that

x″[tdanger] = gradm[x[tdanger], y[tdanger]] . {x′[tdanger], y′[tdanger]}.

y″[tdanger] = gradn[x[tdanger], y[tdanger]] . {x′[tdanger], y′[tdanger]}.

opens up the possibility of singularities of

x″[t] or y″[t] at t = tdanger.

When this happens things can get out of hand!

And they did in part i) above.

DE.06 Systems and Flows Tutorials

T.1) Flow analysis of systems of two first order diffeq's

T.1.a) This system sucks

Here's a system of differential equations:

```
Clear[m, n, x, y, t];
m[x_, y_] = -1.4 x - 0.43 y;
n[x_, y_] = 0.70 x - 0.78 y;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == -1.4 x[t] - 0.43 y[t]
y′[t] == 0.7 x[t] - 0.78 y[t]
```

Do a flow analysis of this system.

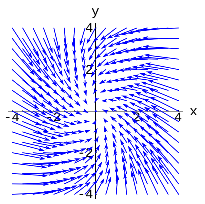
Answer:

To get an idea about what happens to the solutions x[t] and y[t] as t grows and grows, look at this scaled plot of

Field[x, y] = {m[x, y], n[x, y]}:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.25, HeadSize → 0.3],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

```
{-1.4 x - 0.43 y, 0.7 x - 0.78 y}
```



The plot makes it pretty clear that all the trajectories are sucked in toward {0, 0}.

Interpretation:

If {x[t], y[t]} solves:

```
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == -1.4 x[t] - 0.43 y[t]
y′[t] == 0.7 x[t] - 0.78 y[t]
```

Then you can be very confident that

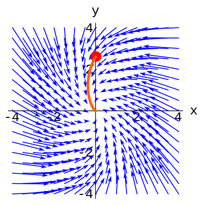
x[t] → 0 and y[t] → 0 as t → ∞.

You can confirm this by dropping in some corks.

Try a cork at a random starting point;

```
{xstarter, ystarter} =
  {Random[Real, {xlow, xhigh}], Random[Real, {ylow, yhigh}]};
endtime = 8;
starterpoint = {xstarter, ystarter}

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
ndssol = NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
   {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory];
trajectory[t_] = {x[t] /. ndssol⟦1⟧, y[t] /. ndssol⟦1⟧};
trajectoryplot =
  ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
    {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];

Show[
  flowplot, starterplot, trajectoryplot, PlotRange → All, Axes → True,
  AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```

```
{0.00216082, 2.61158}
```



```
Rerun a couple of times.
```

Sucked into {0, 0} everytime.

T.1.b) This system propels

Here's a new system of differential equations:

```
Clear[m, n, x, y, t];
m[x_, y_] = 1.2 x - 0.2 y;
n[x_, y_] = -0.2 x + 0.7 y;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == 1.2 x[t] - 0.2 y[t]
y′[t] == -0.2 x[t] + 0.7 y[t]
```

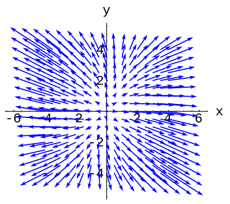
Do a flow analysis of this system.

Answer:

To get an idea about what happens to the solutions x[t] and y[t] as t grows and grows, look at this scaled plot of

Field[x, y] = {m[x, y], n[x, y]}:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
scalefactor = 0.35;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.4, HeadSize → 0.4],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



A vector orgasm.

The plot makes it pretty clear that all the trajectories are propelled away from {0, 0}.

Interpretation:

If {x[t], y[t]} solves:

```
ColumnForm[Thread[diffeqsystem]]
x′[t] == 1.2 x[t] - 0.2 y[t]
y′[t] == -0.2 x[t] + 0.7 y[t]
```

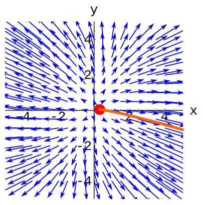
Then you can be very confident that

either Abs[x[t]] or Abs[y[t]]

(and probably both) will be very large when t is big. You can confirm this by dropping in some corks.

Try a cork starting at a random point near {0, 0}.

```
h = 0.3;
s = Random[Real, {0, 2 π}];
{xstarter, ystarter} = {h Cos[s], h Sin[s]};
endtime = 5;
starterpoint = {xstarter, ystarter};

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
ndssol = NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory];
trajectory[t_] = {x[t] /. ndssol⟦1⟧, y[t] /. ndssol⟦1⟧};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];

Show[flowplot, starterplot, trajectoryplot,
 PlotRange → {{xlow - 1, xhigh + 1}, {ylow - 1, yhigh + 1}}, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



Rerun a couple of times.

Away it goes off the screen.

□T.1.c) This system sucks and propels

Here's another system of differential equations:

```
Clear[m, n, x, y, t];
m[x_, y_] = 1.2 y;
n[x_, y_] = 0.7 x;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
x′[t] == 1.2 y[t]
y′[t] == 0.7 x[t]
```

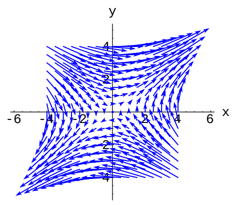
Do a flow analysis of this system.

□Answer:

To get an idea about what happens to the solutions x[t] and y[t] as t grows and grows, you look at this scaled plot of

Field[x, y] = {m[x, y], n[x, y]}:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
scalefactor = 0.4;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.4, HeadSize → 0.4],
  {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



In and out.

The plot makes it pretty clear that all the trajectories are eventually propelled away from {0, 0}, but many of them have been sucked toward {0, 0} first.

Interpretation:

If {x[t], y[t]} solves:

```
ColumnForm[Thread[diffeqsystem]]
x′[t] == 1.2 y[t]
y′[t] == 0.7 x[t]
```

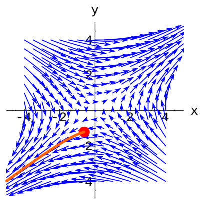
Then you can be very confident that

Abs[x[t]] and Abs[y[t]] will be very large when t is big.

You can confirm this by dropping in some corks.

Try a cork in the flow starting at a random point:;

```
h = Random[Real, {1, 3}];
s = Random[Real, {0, 2 π}];
{xstarter, ystarter} = {h Cos[s], h Sin[s]};
endtime = 5;
starterpoint = {xstarter, ystarter};

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
ndssol = NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
  {x[t], y[t]}, {t, 0, endtime}]; Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol⟦1⟧, y[t] /. ndssol⟦1⟧};

trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];

Show[flowplot, starterplot, trajectoryplot,
 PlotRange → {{xlow - 1, xhigh + 1}, {ylow - 1, yhigh + 1}}, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```

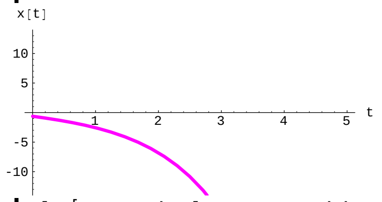


Rerun a couple of times.

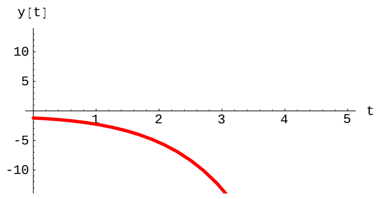
Sucked in toward {0, 0} and then propelled away.

See the individual solution plots for the last one:

```
Plot[x[t] /. ndssol⟦1⟧, {t, 0, endtime},
 PlotStyle → {{Thickness[0.01], Magenta}}, AxesLabel → {"t", "x[t]"},
 PlotRange → {xlow - 10, xhigh + 10}, AspectRatio → 1/2];
```



```
Plot[y[t] /. ndssol⟦1⟧, {t, 0, endtime},
 PlotStyle → {{Thickness[0.01], Red}}, AxesLabel → {"t", "y[t]"},
 PlotRange → {ylow - 10, yhigh + 10}, AspectRatio → 1/2];
```



Play until you're comfortable.

☐T.1.d) This system oscillates

Here's another system of differential equations:

```
Clear[m, n, x, y, t];
m[x_, y_] = 1.3 y;
n[x_, y_] = -0.7 x;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

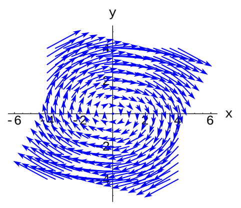
```
x′[t] == 1.3 y[t]
y′[t] == -0.7 x[t]
```

Do a flow analysis of this system.

☐Answer:

To get an idea about what happens to the solutions as t grows and grows, look at this scaled plot of Field[x, y] = {m[x, y], n[x, y]}:

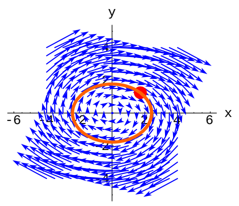
```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
scalefactor = 0.4;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.4, HeadSize → 0.5],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



Whirl.

The plot gives the strong message that all the trajectories oscillate around {0,0}. Throw in an actual trajectory to confirm:

```
h = Random[Real, {1, 3}];
s = Random[Real, {0, 2 π}];
{xstarter, ystarter} = {h Cos[s], h Sin[s]};
endtime = 8;
starterpoint = {xstarter, ystarter};
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
ndssol = NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
   {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory];
trajectory[t_] = {x[t] /. ndssol⟦1⟧, y[t] /. ndssol⟦1⟧};
trajectoryplot =
  ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
    {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];

Show[flowplot, starterplot, trajectoryplot,
  PlotRange → All, Axes → True, AxesLabel → {"x", "y"},
  DisplayFunction → $DisplayFunction];
```



Rerun a couple of times.

Flow interpretation:

If {x[t], y[t]} solves:

```
ColumnForm[Thread[diffeqsystem]]
```

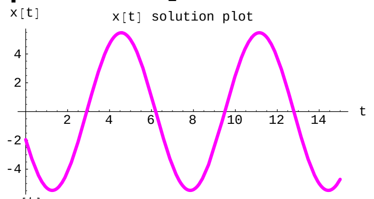
```
x′[t] == 1.3 y[t]
y′[t] == -0.7 x[t]
```

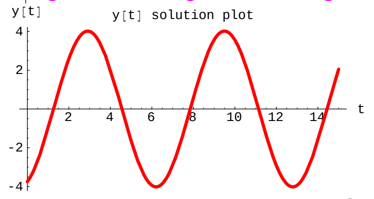
Then you can be very confident that both

x[t] and y[t] will cycle over and over like shifted sine or cosine curves.

Take a look for a random starting point:

```
{xstarter, ystarter} =
  {Random[Real, {xlow, xhigh}], Random[Real, {ylow, yhigh}]};
endtime = 15;

Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
ndssol = NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
   {x[t], y[t]}, {t, 0, endtime}];
x[t_] = x[t] /. ndssol⟦1⟧;
y[t_] = y[t] /. ndssol⟦1⟧;

Plot[x[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Magenta}}, AxesLabel → {"t", "x[t]"},
  AspectRatio → 1/2, PlotLabel → "x[t] solution plot"];

Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, AxesLabel → {"t", "y[t]"},
  AspectRatio → 1/2, PlotLabel → "y[t] solution plot"];
```





Rerun several times.

Yessiree Beavis.

□T.1.e) This system sucks and whirls

Here's another system of differential equations:

```
Clear[m, n, x, y, t];
m[x_, y_] = -1.2 x + 2.6 y;
n[x_, y_] = -0.7 x + 0.2 y;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == -1.2 x[t] + 2.6 y[t]
y′[t] == -0.7 x[t] + 0.2 y[t]
```

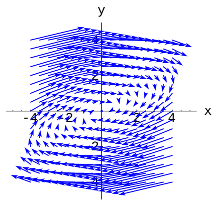
Do a flow analysis of this system.

□Answer:

To get an idea about what happens to the solutions as t grows and grows, you look at this scaled plot of Field[x, y] = {m[x, y], n[x, y]}:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.2, HeadSize → 0.4],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

```
{-1.2 x + 2.6 y, -0.7 x + 0.2 y}
```

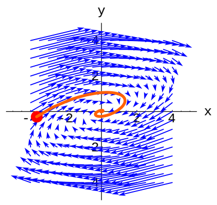


Whirl and suck.

The plot gives the strong message that all the trajectories oscillate and that all the trajectories are sucked toward {0, 0}.

Throw in an actual trajectory to confirm:

```
{xstarter, ystarter} =
  {Random[Real, {xlow, xhigh}], Random[Real, {ylow, yhigh}]};
endtime = 12;
starterpoint = {xstarter, ystarter};
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn = x[0] == xstarter;
ystartereqn = y[0] == ystarter;
ndssol = NDSolve[{xdiffeq, ydiffeq, xstartereqn, ystartereqn},
   {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory];
trajectory[t_] = {x[t] /. ndssol[[1]], y[t] /. ndssol[[1]]};
trajectoryplot =
  ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
    {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];

Show[
  flowplot, starterplot, trajectoryplot, PlotRange → All, Axes → True,
  AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



```
Rerun several times.
```

Swirling down the drain at {0, 0}.

Flow interpretation:

If {x[t], y[t]} solves:

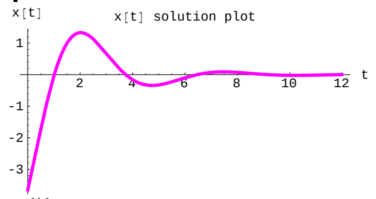
```
ColumnForm[Thread[diffeqsystem]]
```

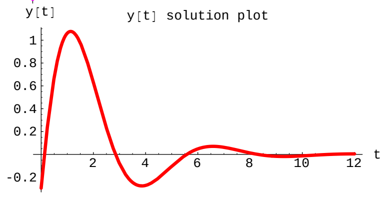
```
x′[t] == -1.2 x[t] + 2.6 y[t]
y′[t] == -0.7 x[t] + 0.2 y[t]
```

Then you can be very confident that both x[t] and y[t] oscillate but fall in size towards 0 as time t gets larger and larger.

Take a look the correspong solution plots for the last one:

```
x[t_] = x[t] /. ndssol[[1]];
y[t_] = y[t] /. ndssol[[1]];

 Plot[x[t], {t, 0, endtime},
  PlotStyle ->
   {{Thickness[0.01], Magenta}}, AxesLabel -> {"t", "x[t]"},
   AspectRatio -> 1 / 2,
  PlotRange -> All, PlotLabel -> "x[t] solution plot"];

Plot[y[t], {t, 0, endtime},
   PlotStyle -> {{Thickness[0.01], Red}}, AxesLabel -> {"t", "y[t]"},
   AspectRatio -> 1 / 2, PlotRange -> All,
  PlotLabel -> "y[t] solution plot"];
```





Squashed oscillations.

□T.1.f) This system propels and oscillates

Here's yet another system of differential equations:

```
Clear[m, n, x, y, t];
m[x_, y_] = 0.2 (1.3 x - 1.6 y) ;
n[x_, y_] = 0.2 (0.9 x - 0.2 y);

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

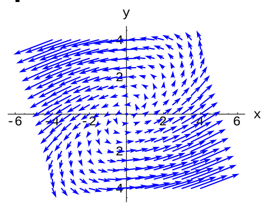
```
x′[t] == 0.2 (1.3 x[t] - 1.6 y[t])
y′[t] == 0.2 (0.9 x[t] - 0.2 y[t])
```

Do a flow analysis of this system.

□Answer:

To get an idea about what happens to the solutions as t grows and grows, you look at this scaled plot of Field[x, y] = {m[x, y], n[x, y]}:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
   VectorColor → Blue, ScaleFactor → 0.9, HeadSize → 0.4],
  {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



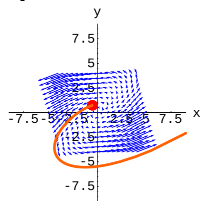
Whirl and propel.

The plot gives the strong message that all the trajectories oscillate and that all the trajectories are propelled away from {0, 0}.

Throw in a random trajectory to confirm:

```
h = Random[Real, {0.2, 1}];
s = Random[Real, {0, 2 Pi}];
{xstarter, ystarter} = {h Cos[s], h Sin[s]};;
endtime = 30;
starterpoint = {xstarter, ystarter};
xdiffeq = (x'[t] == m[x[t], y[t]]);
ydiffeq = (y'[t] == n[x[t], y[t]]);
```

```
xstartereqn = (x[0] == xstarter);
ystartereqn = (y[0] == ystarter);

ndssol = NDSolve[{xdiffeq, ydiffeq,
  xstartereqn, ystartereqn}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory];
trajectory[t_] = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};

trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
   PlotStyle -> {{CadmiumOrange, Thickness[0.015]}},
        DisplayFunction -> Identity];

starterplot = Graphics[{Red, PointSize[0.06],
          Point[starterpoint]}];

Show[flowplot, starterplot, trajectoryplot,
  PlotRange -> {{xlow - 5, xhigh + 5}, {ylow - 5, yhigh + 5}},
 Axes -> True, AxesLabel -> {"x", "y"},
  DisplayFunction -> $DisplayFunction];
```



And away it goes.

Flow interpretation:

If {x[t], y[t]} solves:

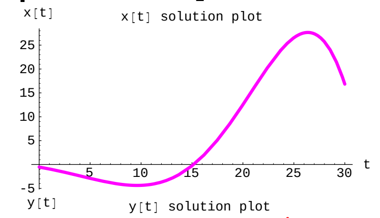
```
ColumnForm[Thread[diffeqsystem]]
```

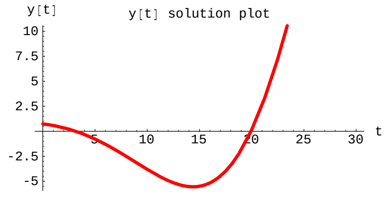
```
x′[t] == 0.2 (1.3 x[t] - 1.6 y[t])
y′[t] == 0.2 (0.9 x[t] - 0.2 y[t])
```

Then you can be very confident that both x[t] and y[t] oscillate above and below 0; the oscillations increase in size as time t goes on.

Take a look at the corresponding individual solution plots for the last one:

```
x[t_] = x[t] /. ndssol[[1]];
y[t_] = y[t] /. ndssol[[1]];

Plot[x[t], {t, 0, endtime},
  PlotStyle ->
  {{Thickness[0.01], Magenta}}, AxesLabel -> {"t", "x[t]"},
  AspectRatio -> 1/2, PlotLabel -> "x[t] solution plot"];
Plot[y[t], {t, 0, endtime},
  PlotStyle -> {{Thickness[0.01], Red}}, AxesLabel -> {"t", "y[t]"},
  AspectRatio -> 1/2, PlotLabel -> "y[t] solution plot"];
```





Yes ma'am.

T.2) Damped oscillators and undamped oscillators

□T.2.a) An undamped oscillator

Go with c = 2 and look at:

```
c = 2;
Clear[t, y];
oscillatordiffeq = (y''[t] + c y[t] == 0)
```

```
2 y[t] + y″[t] == 0
```

To make this second order differential equation into a system of two first order differential equations, you put

x[t] = y′[t]

and then replace y′[t] by x[t] and replace y″[t] by x′[t]:

```
Clear[x];
ColumnForm[Thread[{new = (y'[t] == x[t]),
oscillatordiffeq /. {y'[t] -> x[t], y''[t] -> x'[t]}}]]
```

```
y′[t] == x[t]
2 y[t] + x′[t] == 0
```

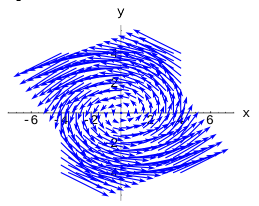
Clean this up:

```
Clear[m, n, x, y, t];
m[x_, y_] = -c y;
n[x_, y_] = x;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == -2 y[t]
y′[t] == x[t]
```

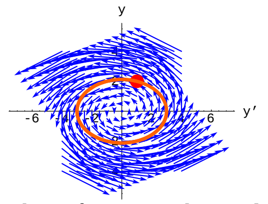
Now look at:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.4, HeadSize → 0.5],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



Throw in a trajectory:

```
{a, b} = {1, 2};
starterpoint = {a, b};
Clear[x, y, t];
equationx = (x'[t] == m[x[t], y[t]]);
equationy = (y'[t] == n[x[t], y[t]]);
starterx = (x[0] == a);
startery = (y[0] == b);
endtime = 8.0;

ndssol = NDSolve[{equationx, equationy,
  starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory];
trajectory[t_] = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};

trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
   PlotStyle -> {{CadmiumOrange, Thickness[0.015]}},
        DisplayFunction -> Identity];

starterplot = Graphics[{Red, PointSize[0.06],
          Point[starterpoint]}];

phaseportrait = Show[flowplot, starterplot,
  trajectoryplot,
  PlotRange -> All, Axes -> True, AxesLabel -> {"y'", "y"},
   DisplayFunction -> $DisplayFunction];
```



What information about solutions of

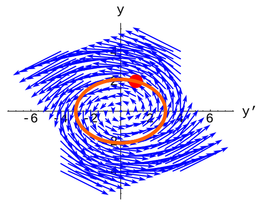
$$y''[t] + 2\,y[t] = 0$$

do you pick up from this plot?

□**Answer:**

Take another look:

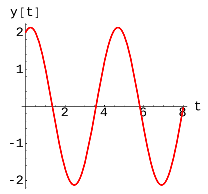
```
Show[phaseportrait];
```



As t increases, y[t] (along the vertical axis) oscillates between positive and negative values and because the trajectory goes right through the starter point, y[t] repeats its action over and over.

It cycles:

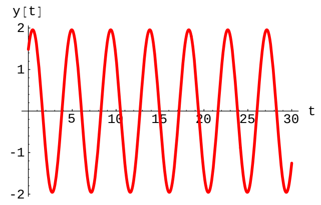
```
Plot[y[t] /. ndssol[[1]], {t, 0, endtime},
 PlotStyle → {{Thickness[0.01], Red}}, AxesLabel → {"t", "y[t]"},
 AspectRatio → 1];
```



Try it out for random starting data for y[0] and y′[0]:

```
Clear[t, y];
oscillatordiffeq = y″[t] + 2 y[t] == 0;
{ystart, yprimestart} = {Random[Real, {-2, 2}], Random[Real, {-2, 2}]}

Clear[y, t, approxy];
endtime = 30;
solution = NDSolve[{oscillatordiffeq,
   y[0] == ystart, y′[0] == yprimestart}, y[t], {t, 0, endtime}];
approxy[t_] = y[t] /. solution[[1]];

Plot[
 approxy[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Red}},
 AxesLabel → {"t", "y[t]"}, AspectRatio → 1/GoldenRatio];
```

```
{1.48424, 1.80339}
```



```
Rerun a couple of times.
```

Yep.

y[t] repeats itself over and over. That's why folks call y[t] an oscillator.

And that's why folks like to say that

$$y''[t] + 2\,y[t] = 0$$

is a differential equation of a harmonic oscillator.

□T.2.b) A damped oscillator

Go with c=2 and look at:

```
c = 2;
Clear[t, y];
oscillatordiffeq = y″[t] + c y[t] == 0
```

```
2 y[t] + y″[t] == 0
```

This is a differential equation of an undamped oscillator.
To get the differential equation of a damped oscillator, you go with a small, positive
number b, and you put:

```
c = 2;
b = 0.5;
Clear[t, y];
dampedoscillatordiffeq = (y''[t] + b y'[t] + c y[t] == 0)
```

```
2 y[t] + 0.5 y′[t] + y″[t] == 0
```

To make this second order differential equation into a system of two first order differential equations, you put

$$x[t] = y'[t]$$

and then replace y′[t] by x[t] and replace y″[t] by x′[t]:

```
Clear[x];
ColumnForm[Thread[{new = (y'[t] == x[t]),
dampedoscillatordiffeq /. {y'[t] -> x[t], y''[t] -> x'[t]}}]]
```

```
y′[t] == x[t]
0.5 x[t] + 2 y[t] + x′[t] == 0
```

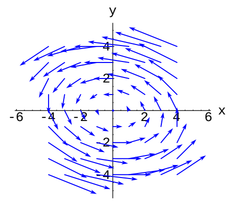
Clean this up:

```
Clear[m, n, x, y, t];
m[x_, y_] = -b x - c y;
n[x_, y_] = x;

diffeqsystem = {x′[t], y′[t]} == {m[x[t], y[t]], n[x[t], y[t]]}
```

```
{x′[t], y′[t]} == {-0.5 x[t] - 2 y[t], x[t]}
```

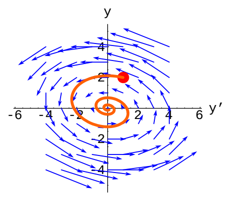
Now look at:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 1;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.3, HeadSize → 0.4],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



Throw in a trajectory:

```
starterpoint = {1, 2};
Clear[x, y, t];
equationx = (x'[t] == m[x[t], y[t]]);
equationy = (y'[t] == n[x[t], y[t]]);
starterx = (x[0] == starterpoint[[1]]);
startery = (y[0] == starterpoint[[2]]);
endtime = 12.0;

ndssol = NDSolve[{equationx, equationy,
  starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory];
trajectory[t_] = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};

trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
   PlotStyle -> {{CadmiumOrange, Thickness[0.015]}},
       DisplayFunction -> Identity];

starterplot = Graphics[{Red, PointSize[0.06],
          Point[starterpoint]}];

phaseportrait = Show[flowplot, starterplot,
  trajectoryplot,
  PlotRange -> All, Axes -> True, AxesLabel -> {"y'", "y"},
   DisplayFunction -> $DisplayFunction];
```



What information about solutions of

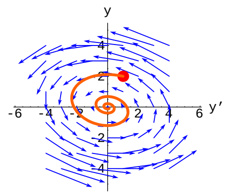
$$y''[t] + 0.5\,y'[t] + 2\,y[t] = 0$$

do you pick up from this plot?

□ Answer:

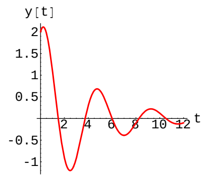
Take another look:

```
Show[phaseportrait];
```



As t increases, y[t] (along the vertical axis) oscillates between positive and negative values. But as time t advances, y[t] finds itself drawn toward 0:

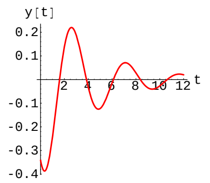
```
y[t_] = y[t] /. ndssol[[1]];

Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Red}},
 AxesLabel → {"t", "y[t]"}, PlotRange → All, AspectRatio → 1];
```



Just as the trajectory revealed.

Try it out for random starting data on y[0] and y′[0]:

```
Clear[t, y];
dampedoscillatordiffeq = y″[t] + 0.5 y′[t] + 2 y[t] == 0;
{ystart, yprimestart} = {Random[Real, {-2, 2}], Random[Real, {-2, 2}]}

Clear[y, t, approx];
endtime = 12;
ndssol = NDSolve[{dampedoscillatordiffeq,
    y[0] == ystart, y′[0] == yprimestart}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];

Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Red}},
 AxesLabel → {"t", "y[t]"}, PlotRange → All, AspectRatio → 1];
{-0.341301, -0.27329}
```

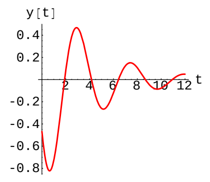


Yes.

Damped oscillations.

Another:

```
Clear[t, y];
dampedoscillatordiffeq = y″[t] + 0.5 y′[t] + 2 y[t] == 0;
{ystart, yprimestart} = {Random[Real, {-2, 2}], Random[Real, {-2, 2}]}

Clear[y, t, approx];
endtime = 12;
ndssol = NDSolve[{dampedoscillatordiffeq,
    y[0] == ystart, y′[0] == yprimestart}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];

Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Red}},
 AxesLabel → {"t", "y[t]"}, PlotRange → All, AspectRatio → 1];
{-0.467591, -1.08418}
```



Damped oscillations.

It will happen everytime.

T.3) A pursuit model

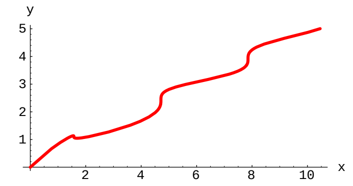
□T.3.a.i)

Bubba is really enjoying one of those big-time keg parties out in the woods. Having learned to monitor his blood alcohol level in earlier Calculus&*Mathematica* lessons, Bubba took his bicycle to the party instead of driving his car. At time (in minutes) t = 0, Bubba leaves the party and gets on his bicycle to ride home.
At time t, after leaving the party, Bubba is at the point

$$\{2\,t + \tfrac{1}{2}\,\mathrm{Sin}[4\,t],\ t + E^{-t}\,\mathrm{Sin}[3\,t]\}.$$

Here is his route:

```
Clear[bubbax, bubbay, t];
{bubbax[t_], bubbay[t_]} = {2 t + 1/2 Sin[4 t], t + E^-t Sin[3 t]};

bubbaroute = ParametricPlot[{bubbax[t], bubbay[t]},
  {t, 0, 5}, PlotStyle → {{Thickness[0.01], Red}},
  AxesLabel → {"x", "y"}, DisplayFunction → Identity];
pointplot = {Graphics[{PointSize[0.06], Point[bubba[0]]}],
  Graphics[{PointSize[0.06], Point[bubba[5]]}]};
labels = {Graphics[Text["Party", bubba[0], {-2, 0}]],
  Graphics[Text["Home", bubba[5], {0, 5}]]};

Show[bubbaroute, pointplot, labels, AspectRatio → Automatic,
 PlotRange → All, DisplayFunction → $DisplayFunction];
```



Slightly unsteady is your friend Bubba.
While Bubba was at the party, his Blue Tick Hound Dog was sleeping at the point {0, 2}. As Bubba left the party, the dog woke up and chased after Bubba.

Here is the dog's scheme:
If the dog is at
{dogx[t], dogy[t]}
at time t, then the dog leaves {dogx[t], dogy[t]} with instantaneous velocity
{bubbax[t], bubbay[t]} − {dogx[t], dogy[t]}.
This means that at any time t, the dog is always running toward the point {bubbax[t], bubbay[t]}, which is Bubba's current position.

Does the dog catch up with Bubba before Bubba gets home and locks the dog out?

□ Answer:

You guessed it.

This is a job for a system of differential equations.

Put:

```
Clear[dogvelx, dogvely, x, y, t];
{dogvelx[t_], dogvely[t_]} = {bubbax[t], bubbay[t]} - {x[t], y[t]}
```

$\{2\,t + \frac{1}{2}\,\text{Sin}[4\,t] - x[t],\ t + E^{-t}\,\text{Sin}[3\,t] - y[t]\}$

The system of differential equations is:

```
equationx = x'[t] == dogvelx[t]
```

$x'[t] == 2\,t + \frac{1}{2}\,\text{Sin}[4\,t] - x[t]$

```
equationy = y'[t] == dogvely[t]
```

$y'[t] == t + E^{-t}\,\text{Sin}[3\,t] - y[t]$

```
starterx = x[0] == 0
```

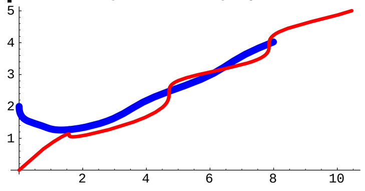
```
x[0] == 0
```

```
startery = y[0] == 2
```

```
y[0] == 2
```

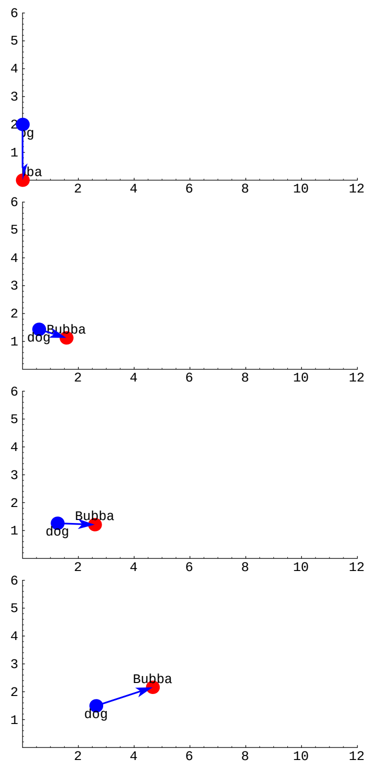
Here comes an approximate plot of the dog's route for the first five minutes:

```
endtime = 5;
Clear[x, y, dogx, dogy, t];
ndssol = NDSolve[{equationx, equationy, starterx, startery},
   {x[t], y[t]}, {t, 0, endtime}];
dogx[t_] = x[t] /. ndssol[[1]];
dogy[t_] = y[t] /. ndssol[[1]];
dogplot = ParametricPlot[{dogx[t], dogy[t]}, {t, 0, endtime},
   PlotStyle → {{Blue, Thickness[0.02]}}, DisplayFunction → Identity];

outcome =
  Show[dogplot, bubbaroute, pointplot, labels, AspectRatio → Automatic,
   PlotRange → All, DisplayFunction → $DisplayFunction];
```



The dog didn't make it.

Hope it doesn't rain.



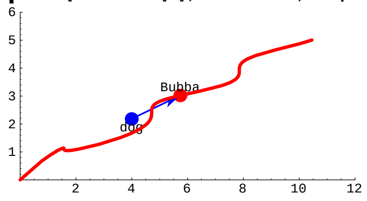
□T.3.a.ii)

Illustrate the dog's strategy by making a movie reviewing the chase.

□ **Answer:**

Set up the movie code and look at one sample frame:

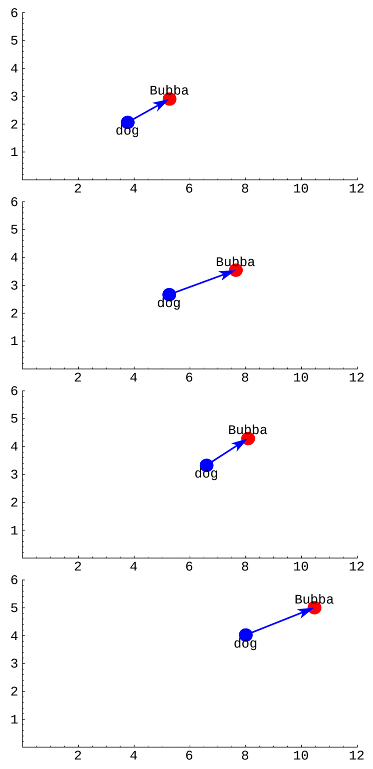
```
Clear[bubbapoint, bubbavelocityvector,
 dogpoint, dogvelocityvector, situation, t];
bubbapoint[t_] :=
 {Graphics[{Red, PointSize[0.04], Point[{bubbax[t], bubbay[t]}]}],
  Graphics[Text["Bubba", {bubbax[t], bubbay[t]} + {0, 0.3}]]};
dogpoint[t_] :=
 Graphics[{{Blue, PointSize[0.04], Point[{dogx[t], dogy[t]}]},
   Text["dog", {dogx[t], dogy[t]} - {0, 0.3}]}];
dogvelocityvector[t_] := Arrow[{dogx'[t], dogy'[t]},
  Tail → {dogx[t], dogy[t]}, VectorColor → Blue, HeadSize → 0.6];
situation[t_] := Show[pointplot,
  labels, bubbapoint[t], dogpoint[t], dogvelocityvector[t],
  AspectRatio → Automatic, PlotRange → {{0, 12}, {0, 6}},
  Axes → True, DisplayFunction → Identity];

Show[situation[3], bubbaroute, DisplayFunction → $DisplayFunction];
```



The arrow is the dog's velocity vector at the plotted location of the dog.

See the whole chase from beginning to end:

```
Table[Show[situation[t], DisplayFunction → $DisplayFunction];,
 {t, 0, endtime, endtime/7}];
```



Animate these by grabbing all the plots and selecting the animation instruction in the Cell menu.

The dog almost catches up with Bubba early in the chase.

Bubba is going so fast when he gets to his home that you've got to be nervous about whether or not he crashes right through the front door.

□T.3.a.iii)

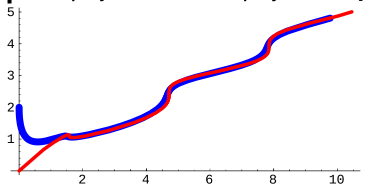
What could the dog have done to catch Bubba?

Answer:

Common sense says that the dog should have run faster.

Here's what could have happened if the dog had run 5 times faster than he did above:

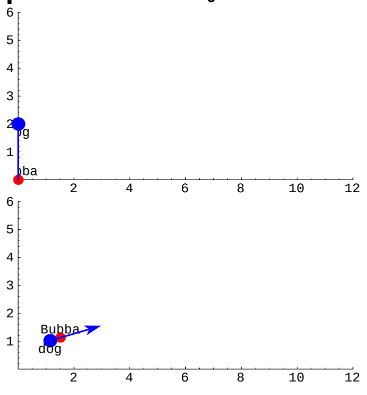
```
Clear[x, y, t, newdogvelx, newdogvely];
r = 5;
{newdogvelx[t_], newdogvely[t_]} = r {dogvelx[t], dogvely[t]};
equationx = x′[t] == newdogvelx[t];
equationy = y′[t] == newdogvely[t];
starterx = x[0] == 0;
startery = y[0] == 2;
endtime = 5;

Clear[x, y, t];
ndssol = NDSolve[{equationx, equationy, starterx, startery},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[newdogx, newdogy];
newdogx[t_] = x[t] /. ndssol⟦1⟧;
newdogy[t_] = y[t] /. ndssol⟦1⟧;
newdogplot = ParametricPlot[{newdogx[t], newdogy[t]}, {t, 0, endtime},
  PlotStyle → {{Blue, Thickness[0.02]}}, DisplayFunction → Identity];

outcome = Show[newdogplot, bubbaroute,
  pointplot, labels, AspectRatio → Automatic, PlotRange → All,
  DisplayFunction → $DisplayFunction];
```

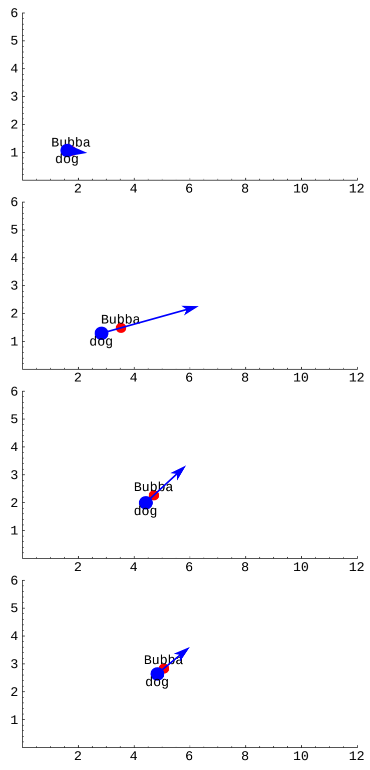


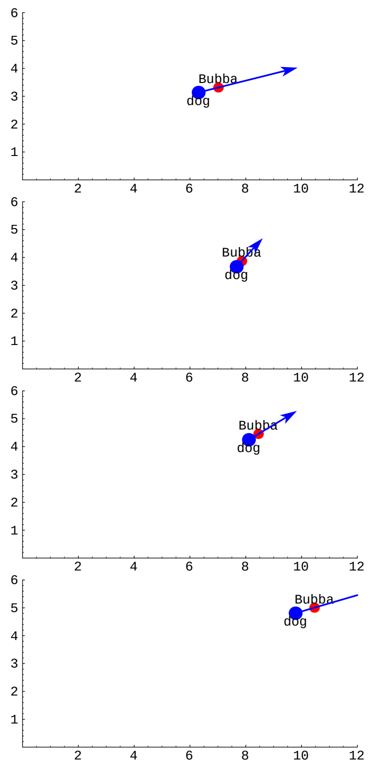
The smart money bets that the dog trotted through the door only seconds after Bubba opened the door. In fact, the dog was running only slightly behind Bubba for most of the chase.

Take a look at the whole chase:

```
Clear[bubbapoint, bubbavelocityvector,
 newdogpoint, newdogvelocityvector, newsituation, t];
bubbapoint[t_] :=
 {Graphics[{Red, PointSize[0.03], Point[{bubbax[t], bubbay[t]}]}],
  Graphics[Text["Bubba", {bubbax[t], bubbay[t]} + {0, 0.3}]]};
newdogpoint[t_] :=
 Graphics[{{Blue, PointSize[0.04], Point[{newdogx[t], newdogy[t]}]},
   Text["dog", {newdogx[t], newdogy[t]} - {0, 0.3}]}];
newdogvelocityvector[t_] := Arrow[{newdogx′[t], newdogy′[t]},
  Tail → {newdogx[t], newdogy[t]}, VectorColor → Blue, HeadSize → 0.6];
newsituation[t_] := Show[pointplot, labels,
  bubbapoint[t], newdogpoint[t], newdogvelocityvector[t],
  AspectRatio → Automatic, PlotRange → {{0, 12}, {0, 6}},
  Axes → True, DisplayFunction → Identity];

Table[Show[newsituation[t], DisplayFunction → $DisplayFunction];,
 {t, 0, endtime, endtime/9}];
```







The dog is with him almost all the way.

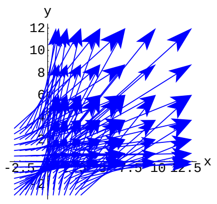
Surely the wise old dog got through the door before Bubba locked the door and passed out.

And the dog is no fool; the dog knows that it's safer to run behind Bubba than to run in front of him.

T.4) Troubleshooting plots of vector fields

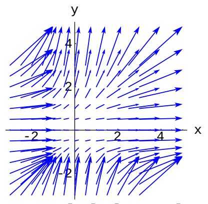
To get a good visualization of a vector field, you can plot the field as it comes:

```
Clear[Field, x, y];
Field[x_, y_] = {x^2 + 1, y^2};
fieldplot = Table[Arrow[Field[x, y], Tail → {x, y}, VectorColor → Blue],
   {x, -3, 3, 0.5}, {y, -3, 3, 0.5}];

Show[fieldplot, Axes → Automatic, AxesLabel → {"x", "y"}];
```



In some cases, such as this one, the vectors plot out so long that you have a hard time visualizing the flow of the field.
Other times, the field vectors might plot out so short that you can't see them clearly.
In such cases, you might want to apply a ScaleFactor:

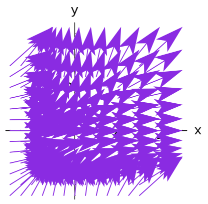
```
scalefactor = 0.2;
scaledfieldplot =
 Table[Arrow[Field[x, y], Tail → {x, y}, VectorColor → Blue,
   ScaleFactor → scalefactor], {x, -3, 3, 0.5}, {y, -3, 3, 0.5}];

Show[scaledfieldplot, Axes → Automatic, AxesLabel → {"x", "y"}];
```



In a scaled plot such as this one, the lengths of the vectors are proportional to the speed of the flow, just as they were in the unscaled plot.

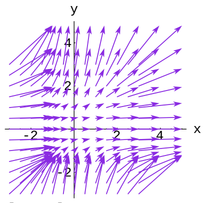
To clean things up, you can also adjust the size of the tips of the arrows.
Look at the messy plot below:

```
scaledfieldplot = Table[Arrow[Field[x, y], Tail → {x, y},
    VectorColor → BlueViolet, ScaleFactor → 0.2, HeadSize → 1.3],
   {x, -3, 3, 0.5}, {y, -3, 3, 0.5}];

Show[scaledfieldplot, Axes → Automatic, AxesLabel → {"x", "y"}];
```



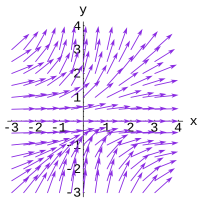
Those tips are too big!
Reduce their size using HeadSize:

```
scaledfieldplot = Table[Arrow[Field[x, y], Tail → {x, y},
    VectorColor → BlueViolet, ScaleFactor → 0.2, HeadSize → 0.4],
   {x, -3, 3, 0.5}, {y, -3, 3, 0.5}];

Show[scaledfieldplot, Axes → Automatic, AxesLabel → {"x", "y"}];
```



That's better.
Sometimes you can't get any satisfactory scaled plot. In this case you might want to to scale each vector so that they all turn out to be unit vectors. You can do this by setting ScaleFactor → Normalize.

```
unitfieldplot = Table[Arrow[Field[x, y], Tail -> {x, y},
   VectorColor -> BlueViolet,
    ScaleFactor -> Normalize, HeadSize -> 0.3],
     {x, -3, 3, 0.5}, {y, -3, 3, 0.5}];

Show[unitfieldplot, Axes -> Automatic,
 AxesLabel -> {"x", "y"}];
```

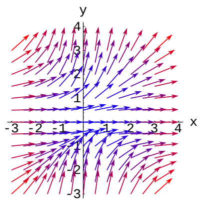


This plot of unitField[x, y] shows only direction but not the speed of the flow.

If you want a field plot using unit vectors, and don't want to lose magnitude information, there is a way to get around it. You can plot a scaled unit field plot, and color the vectors according to their magnitudes. It's not too hard to do, but does require a little work first.

```
Clear[colorfieldplot, max, color];
max = Max[Flatten[
    Table[Norm[N[Field[x, y]]], {x, -3, 3, .5}, {y, -3, 3, .5}]]];
```

```
color[x_, y_] :=
 RGBColor[N[Norm[Field[x, y]]]/max, 0, 1. - N[Norm[Field[x, y]]]/max];
colorfieldplot = Table[Arrow[Field[x, y],
    Tail → {x, y}, VectorColor → Evaluate[color[x, y]],
    ScaleFactor → Normalize, HeadSize → 0.3],
   {x, -3, 3, .5}, {y, -3, 3, .5}];

Show[colorfieldplot, Axes → Automatic, AxesLabel → {"x", "y"}];
```



The larger (magnitude) vectors are red and the smaller are blue.
Not bad, huh?
Some folks feel that these color-coded plots are an information overload.
What do you think?

DE.06 Systems and Flows
Give It a Try!

Experience with starred problems will be especially beneficial for understanding later lessons.

G.1) Flow analysis of behavior of the solutions of systems of diffeqs*

□G.1.a.i)

Here's a diffeq system ripe for study:

```
Clear[m, n, x, y, t];
m[x_, y_] = -1.2 x + 0.2 y;
n[x_, y_] = 0.3 x - 0.4 y;
```

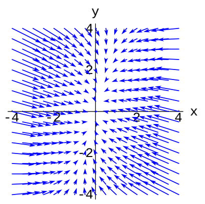
```
diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == -1.2 x[t] + 0.2 y[t]
y′[t] == 0.3 x[t] - 0.4 y[t]
```

The first thing you do is a flow analysis:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.25, HeadSize → 0.3],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

```
{-1.2 x + 0.2 y, 0.3 x - 0.4 y}
```



Read the flow plot and use what you see to write up your description of the general behavior of the trajectories.

```
Fill in a couple of starting points
in the code below and run to help illustrate your answer.
```

```
{xstarter1, ystarter1} = {here, here}
{xstarter2, ystarter2} = {here, here}
endtime = 8;
starterpoint1 = {xstarter1, ystarter1};
starterpoint2 = {xstarter2, ystarter2};
Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn1 = x[0] == xstarter1;
ystartereqn1 = y[0] == ystarter1;
xstartereqn2 = x[0] == xstarter2;
ystartereqn2 = y[0] == ystarter2;
ndssol1 = NDSolve[{xdiffeq, ydiffeq, xstartereqn1, ystartereqn1},
  {x[t], y[t]}, {t, 0, endtime}];
ndssol2 = NDSolve[{xdiffeq, ydiffeq, xstartereqn2, ystartereqn2},
  {x[t], y[t]}, {t, 0, endtime}];
Clear[x1, x2, y1, y2];
{x1[t_], y1[t_]} = {x[t] /. ndssol1[[1]], y[t] /. ndssol1[[1]]};
{x2[t_], y2[t_]} = {x[t] /. ndssol2[[1]], y[t] /. ndssol2[[1]]};
Clear[trajectory1, trajectory2];
trajectory1[t_] = {x1[t], y1[t]};
trajectory2[t_] = {x2[t], y2[t]};
trajectoryplots = ParametricPlot[{trajectory1[t], trajectory2[t]},
  {t, 0, endtime}, PlotStyle → {{CadmiumOrange, Thickness[0.015]}},
  DisplayFunction → Identity];
starterplots =
 {Graphics[{Red, PointSize[0.06], Point[starterpoint1]}],
  Graphics[{Red, PointSize[0.06], Point[starterpoint2]}]};

Show[
 flowplot, starterplots, trajectoryplots, PlotRange → All, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```

□G.1.a.ii)

Agree that {x[t], y[t]} is the solution of the diffeq system in part i) corresponding to starter data

{x[0], y[0]} = {3.0, 4.0}.

Plot x[t] for t running from 0 to 6.
Plot y[t] for t running from 0 to 6.

□G.1.b.i)

Here's a new diffeq system ripe for study:

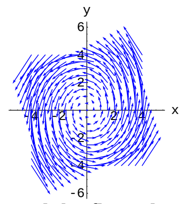
```
Clear[m, n, x, y, t];
m[x_, y_] = -1.7 y;
n[x_, y_] = 2.6 x;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == -1.7 y[t]
y′[t] == 2.6 x[t]
```

The first thing you do is a flow analysis:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
scalefactor = 0.25;
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.2, HeadSize → 0.3],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

```
{-1.7 y, 2.6 x}
```



Read the flow plot and use what you see to write up your description of the general behavior of the trajectories and the solutions.

```
Fill in a couple of starting points
in the code below and run to help illustrate your answer.
```

```
{xstarter1, ystarter1} = {here, here}
{xstarter2, ystarter2} = {here, here}
endtime = 8;
starterpoint1 = {xstarter1, ystarter1};
starterpoint2 = {xstarter2, ystarter2};
Clear[x, y, t];
xdiffeq = x′[t] == m[x[t], y[t]];
ydiffeq = y′[t] == n[x[t], y[t]];
xstartereqn1 = x[0] == xstarter1;
ystartereqn1 = y[0] == ystarter1;
xstartereqn2 = x[0] == xstarter2;
ystartereqn2 = y[0] == ystarter2;
ndssol1 = NDSolve[{xdiffeq, ydiffeq, xstartereqn1, ystartereqn1},
  {x[t], y[t]}, {t, 0, endtime}];
ndssol2 = NDSolve[{xdiffeq, ydiffeq, xstartereqn2, ystartereqn2},
  {x[t], y[t]}, {t, 0, endtime}];
Clear[x1, x2, y1, y2];
{x1[t_], y1[t_]} = {x[t] /. ndssol1[[1]], y[t] /. ndssol1[[1]]};
{x2[t_], y2[t_]} = {x[t] /. ndssol2[[1]], y[t] /. ndssol2[[1]]};
Clear[trajectory1, trajectory2];
trajectory1[t_] = {x1[t], y1[t]};
trajectory2[t_] = {x2[t], y2[t]};
trajectoryplots = ParametricPlot[{trajectory1[t], trajectory2[t]},
  {t, 0, endtime}, PlotStyle → {{CadmiumOrange, Thickness[0.015]}},
  DisplayFunction → Identity];
starterplots =
 {Graphics[{Red, PointSize[0.06], Point[starterpoint1]}],
  Graphics[{Red, PointSize[0.06], Point[starterpoint2]}]};

Show[
 flowplot, starterplots, trajectoryplots, PlotRange → All, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```

□G.1.b.ii)

Agree that {x[t], y[t]} is the solution of the diffeq system in part i) corresponding to starter data

{x[0], y[0]} = {3.0, 4.0}.

Plot x[t] for t running from 0 to 7.
Plot y[t] for t running from 0 to 7.
Do other solutions have the same rough characteristics?

□G.1.c.i)

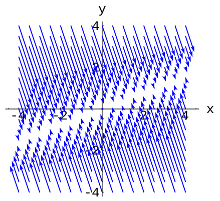
Here's yet another diffeq system ripe for study:

```
Clear[m, n, x, y, t];
m[x_, y_] =  -0.26 x + 0.68 y;
n[x_, y_] =  0.78 x - 2.04 y;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == -0.26 x[t] + 0.68 y[t]
y′[t] == 0.78 x[t] - 2.04 y[t]
```

The first thing you do is a flow analysis:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
scalefactor = 0.2;
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.25, HeadSize → 0.3],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



Read the flow plot and use what you see to write up your description of the general behavior of the trajectories.
Try increasing the endtime. Do the trajectories seem to stall?

Fill in a couple of starting points
in the code below and run to help illustrate your answer.

```
{xstarter1, ystarter1} = {here, here}
{xstarter2, ystarter2} = {here, here}
endtime = 8;
starterpoint1 = {xstarter1, ystarter1};
starterpoint2 = {xstarter2, ystarter2};
Clear[x, y, t];
xdiffeq = x'[t] == m[x[t], y[t]];
ydiffeq = y'[t] == n[x[t], y[t]];
xstartereqn1 = x[0] == xstarter1;
ystartereqn1 = y[0] == ystarter1;
xstartereqn2 = x[0] == xstarter2;
ystartereqn2 = y[0] == ystarter2;
ndssol1 = NDSolve[{xdiffeq, ydiffeq, xstartereqn1, ystartereqn1},
  {x[t], y[t]}, {t, 0, endtime}];
ndssol2 = NDSolve[{xdiffeq, ydiffeq, xstartereqn2, ystartereqn2},
  {x[t], y[t]}, {t, 0, endtime}];
Clear[x1, x2, y1, y2];
{x1[t_], y1[t_]} = {x[t] /. ndssol1[[1]], y[t] /. ndssol1[[1]]};
{x2[t_], y2[t_]} = {x[t] /. ndssol2[[1]], y[t] /. ndssol2[[1]]};
Clear[trajectory1, trajectory2];
trajectory1[t_] = {x1[t], y1[t]};
trajectory2[t_] = {x2[t], y2[t]};
trajectoryplots = ParametricPlot[{trajectory1[t], trajectory2[t]},
  {t, 0, endtime}, PlotStyle → {{CadmiumOrange, Thickness[0.015]}},
  DisplayFunction → Identity];
starterplots =
 {Graphics[{Red, PointSize[0.06], Point[starterpoint1]}],
  Graphics[{Red, PointSize[0.06], Point[starterpoint2]}]};
```

```
Show[
 flowplot, starterplots, trajectoryplots, PlotRange → All, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```

□G.1.c.ii)

Agree that {x[t], y[t]} is the solution of the diffeq system in part i) corresponding to starter data
{x[0], y[0]} = {2.0, −3.0}.
Plot x[t] for t running from 0 to 6.
Plot y[t] for t running from 0 to 6.
Use the plotting options:
AspectRatio → $\frac{1}{2}$ and PlotRange → All.

□G.1.d)

Here's still another diffeq system very ripe for study:

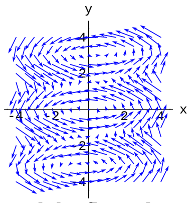
```
Clear[m, n, x, y, t];
m[x_, y_] =  -0.36 x  - 3.2 Sin[2 y];
n[x_, y_] =  0.78 x  - 0.11 y ;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == -3.2 Sin[2 y[t]] - 0.36 x[t]
y′[t] == 0.78 x[t] - 0.11 y[t]
```

The first thing you do is a flow analysis:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
scalefactor = 0.3;
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.25, HeadSize → 0.3],
  {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

```
{-0.36 x - 3.2 Sin[2 y], 0.78 x - 0.11 y}
```



Read the flow plot and use what you see to write up your description of the general behavior of the trajectories as they pass through the plotted part of the xy-plane.
Are all the trajectories that start on the plotted part of the xy-plane above sucked to {0, 0}?

Fill in a couple of starting points
in the code below and run to help illustrate your answer.

```
{xstarter1, ystarter1} = {here, here}
{xstarter2, ystarter2} = {here, here}
endtime = 8;
starterpoint1 = {xstarter1, ystarter1};
starterpoint2 = {xstarter2, ystarter2};
Clear[x, y, t];
xdiffeq = x'[t] == m[x[t], y[t]];
ydiffeq = y'[t] == n[x[t], y[t]];
xstartereqn1 = x[0] == xstarter1;
ystartereqn1 = y[0] == ystarter1;
xstartereqn2 = x[0] == xstarter2;
ystartereqn2 = y[0] == ystarter2;
ndssol1 = NDSolve[{xdiffeq, ydiffeq, xstartereqn1, ystartereqn1},
  {x[t], y[t]}, {t, 0, endtime}];
ndssol2 = NDSolve[{xdiffeq, ydiffeq, xstartereqn2, ystartereqn2},
  {x[t], y[t]}, {t, 0, endtime}];
Clear[x1, x2, y1, y2];
{x1[t_], y1[t_]} = {x[t] /. ndssol1[[1]], y[t] /. ndssol1[[1]]};
{x2[t_], y2[t_]} = {x[t] /. ndssol2[[1]], y[t] /. ndssol2[[1]]};
Clear[trajectory1, trajectory2];
trajectory1[t_] = {x1[t], y1[t]};
trajectory2[t_] = {x2[t], y2[t]};
trajectoryplots = ParametricPlot[{trajectory1[t], trajectory2[t]},
  {t, 0, endtime}, PlotStyle → {{CadmiumOrange, Thickness[0.015]}},
  DisplayFunction → Identity];
starterplots =
 {Graphics[{Red, PointSize[0.06], Point[starterpoint1]}],
  Graphics[{Red, PointSize[0.06], Point[starterpoint2]}]};
```

```
Show[
 flowplot, starterplots, trajectoryplots, PlotRange → All, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```

G.2) Sensitive dependence on starter data

Here's a diffeq system:

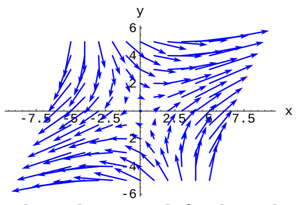
```
Clear[m, n, x, y, t];
m[x_, y_] = 2.4 x + 1.9 y;
n[x_, y_] = 1.7 x - 0.9 y;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == 2.4 x[t] + 1.9 y[t]
y′[t] == 1.7 x[t] - 0.9 y[t]
```

To get an idea about what happens to the solutions as t grows and grows, you look at this scaled plot of Field[x, y] = {m[x, y], n[x, y]}:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
{xlow, xhigh} = {-5, 5};
{ylow, yhigh} = {-5, 5};
jump = 1;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.2, HeadSize → 0.6],
  {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

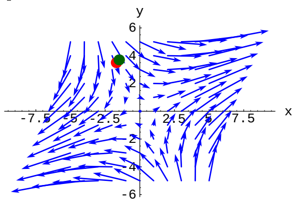
```
{2.4 x + 1.9 y, 1.7 x - 0.9 y}
```



Look on the upper left where the flows diverge and throw in two corks at the plotted points:

```
starterpoint1 = {-1.7, 3.5};
starterpoint2 = {-1.5, 3.7};
```

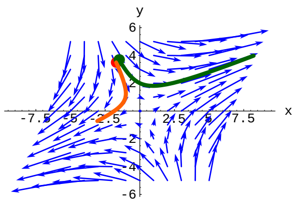
```
starterplots =
 {Graphics[{Red, PointSize[0.04], Point[starterpoint1]}],
  Graphics[{GreenDark, PointSize[0.04], Point[starterpoint2]}]};

Show[flowplot, starterplots, PlotRange → All, Axes → True,
 AxesLabel → {"x", "y"}];
```



Now see trajectories starting at these plotted points:

```
{xstarter1, ystarter1} = starterpoint1;
{xstarter2, ystarter2} = starterpoint2;
endtime = 1.2;

Clear[x, y, t];
xdiffeq = (x'[t] == m[x[t], y[t]]);
ydiffeq = (y'[t] == n[x[t], y[t]]);
xstartereqn1 = (x[0] == xstarter1);
ystartereqn1 = (y[0] == ystarter1);
xstartereqn2 = (x[0] == xstarter2);
ystartereqn2 = (y[0] == ystarter2);

ndssol1 = NDSolve[{xdiffeq, ydiffeq,
  xstartereqn1, ystartereqn1}, {x[t], y[t]}, {t, 0, endtime}];
ndssol2 = NDSolve[{xdiffeq, ydiffeq,
  xstartereqn2, ystartereqn2}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory1, trajectory2];
trajectory1[t_] = {x[t] /. ndssol1[[1]],
       y[t] /. ndssol1[[1]]};
trajectory2[t_] = {x[t] /. ndssol2[[1]],
       y[t] /. ndssol2[[1]]};

trajectoryplots =
 ParametricPlot[{trajectory1[t], trajectory2[t]}, {t, 0, endtime},
   PlotStyle -> {{CadmiumOrange, Thickness[0.015]},
    {GreenDark, Thickness[0.015]}},
       DisplayFunction -> Identity];
```

```
Show[flowplot, starterplots, trajectoryplots,
  PlotRange -> All, Axes -> True, AxesLabel -> {"x", "y"},
  DisplayFunction -> $DisplayFunction];
```



Even though the starter data are almost the same, the solutions of:

```
ColumnForm[Thread[diffeqsystem]]
x′[t] == 2.4 x[t] + 1.9 y[t]
y′[t] == 1.7 x[t] - 0.9 y[t]
```

with:

```
{xstarter1, ystarter1}
{-1.7, 3.5}
```

are nothing like the solutions of:

```
ColumnForm[Thread[diffeqsystem]]
x′[t] == 2.4 x[t] + 1.9 y[t]
y′[t] == 1.7 x[t] - 0.9 y[t]
```

with:

```
{xstarter2, ystarter2}
{-1.5, 3.7}
```

The pros say that specific solutions of this system,

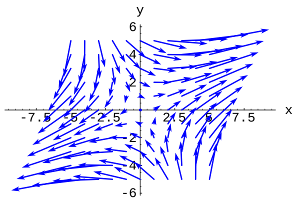
```
ColumnForm[Thread[diffeqsystem]]
x′[t] == 2.4 x[t] + 1.9 y[t]
y′[t] == 1.7 x[t] - 0.9 y[t]
```

can be sensitive to errors in the given starting data.

□G.2.a.i)

Take another look at the flow plot above.

```
Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

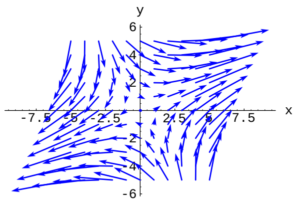


Inspect the lower right carefully. Find and plot two wildly different trajectories starting at nearly the same spot in the lower right.

□G.2.a.ii)

Take another look at the flow plot above.

```
Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



This is the flow for the system:

```
ColumnForm[Thread[diffeqsystem]]
x′[t] == 2.4 x[t] + 1.9 y[t]
y′[t] == 1.7 x[t] - 0.9 y[t]
```

If you go with starter data
x[0] > 1 and y[0] > 1,
do you expect to run into the problem of sensitive dependence on errors in the starting data?
Why or why not?

□G.2.b)

Take a look at the flow of the trajectories of this system

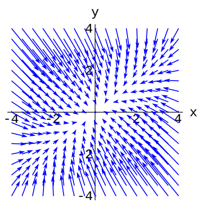
$x'[t] = -1.1\, x[t] + 0.6\, y[t]$
$y'[t] = 0.6\, x[t] - 1.5\, y[t]$

of two differential equations:

```
Clear[m, n, x, y, t];
m[x_, y_] = -1.1 x + 0.6 y;
n[x_, y_] = 0.6 x - 1.5 y;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
x′[t] == -1.1 x[t] + 0.6 y[t]
y′[t] == 0.6 x[t] - 1.5 y[t]
```

And the flow:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
scalefactor = 0.2;
flowplot =
 Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y}, VectorColor → Blue,
   ScaleFactor → 0.2, HeadSize → 0.3], {x, -4, 4, 0.5}, {y, -4, 4, 0.5}];

Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
{-1.1 x + 0.6 y, 0.6 x - 1.5 y}
```



Do you detect any serious sensitivity of this system to small errors in starter data on x[0] and y[0]?

□G.2.c)

Here is a protype of the differential equation of the damped linear oscillator:

```
Clear[t, y];
b = 0.6;
```

```
c = 9.5;

oscillatordiffeq = y″[t] + b y′[t] + c y[t] == 0
9.5 y[t] + 0.6 y′[t] + y″[t] == 0
```

You can rewrite this as a system of two first order differential equations as follows by putting

x[t] = y′[t]

and then replacing y′[t] by x[t] and replacing y″[t] by x′[t]:

```
Clear[x];
ColumnForm[Thread[{new = (y'[t] == x[t]),
oscillatordiffeq /. {y'[t] -> x[t], y''[t] -> x'[t]}}]]
y′[t] == x[t]
0.6 x[t] + 9.5 y[t] + x′[t] == 0
```

Clean this up:

```
Clear[m, n, x, y, t];
m[x_, y_] = -0.5 x - 3.1 y;
n[x_, y_] = x;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
x′[t] == -0.5 x[t] - 3.1 y[t]
y′[t] == x[t]
```

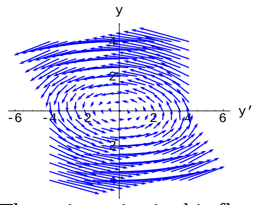
This system is exactly the same as:

```
oscillatordiffeq
9.5 y[t] + 0.6 y′[t] + y″[t] == 0
```

Look at this scaled plot of

Field[x, y] = {m[x, y], n[x, y]}:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
flowplot =
  Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y}, VectorColor → Blue,
    ScaleFactor → 0.2, HeadSize → 0.3], {x, -4, 4, 0.5}, {y, -4, 4, 0.5}];

Show[flowplot, PlotRange → All, Axes → True, AxesLabel → {"y'", "y"}];
{-0.5 x - 3.1 y, x}
```



The trajectories in this flow are plots of {y′[t], y[t]} where y[t] is a solution of:

```
oscillatordiffeq
9.5 y[t] + 0.6 y′[t] + y″[t] == 0
```

determined by starting data on y[0] and y′[0] (= x[0]).
Do you detect any indication of serious sensitive dependence of the behavior of solutions y[t] of

```
oscillatordiffeq
9.5 y[t] + 0.6 y′[t] + y″[t] == 0
```

on small errors in starting data on y[0] and y′[0]?

G.3) The roles of m[x, y] and n[x, y] in a diffeq system

x'[t] = m[x[t], y[t]]

y'[t] = n[x[t], y[t]]

Usually you enter a diffeq system by specifying functions m[x, y] and n[x, y] like this:

```
Clear[m, n, x, y, t];
m[x_, y_] = x - 0.9 Sin[y];
n[x_, y_] = 0.70 x - 0.78 y;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
x′[t] == -0.9 Sin[y[t]] + x[t]
y′[t] == 0.7 x[t] - 0.78 y[t]
```

But sometimes you run across a diffeq system that doesn't explicitly mention m[x, y] and n[x, y].
Here's one:

$$x'[t] = y[t] - x[t]^2$$
$$y'[t] = y[t]\,\mathrm{Cos}[x[t]]$$

Take the code below and fill in the x and y formulas that make the code spit out

$$x'[t] = y[t] - x[t]^2$$
$$y'[t] = y[t]\,\mathrm{Cos}[x[t]]$$

```
Clear[m, n, x, y, t];
m[x_, y_] =
n[x_, y_] =

diffeqsystem = {x′[t], y′[t]} == {m[x[t], y[t]], n[x[t], y[t]]};
 diffeqsystem
```

□G.3.a.i)

Here's a new diffeq system:

```
Clear[m, n, x, y, t];
m[x_, y_] = x^2 + y^2 + 1;
n[x_, y_] = x;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
x′[t] == 1 + x[t]^2 + y[t]^2
y′[t] == x[t]
```

Look at the formula for m[x, y]:

```
m[x, y]
1 + x^2 + y^2
```

And note that

m[x, y] > 0

no matter what x and y are.
Explain why this tells you that no matter what solution pair

{x[t], y[t]}

you go with, you are guaranteed that x[t] is going up as t advances.

□G.3.a.ii)

Stay with the same diffeq system:

```
Clear[m, n, x, y, t];
m[x_, y_] = x^2 + y^2 + 1;
n[x_, y_] = x;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
x′[t] == 1 + x[t]^2 + y[t]^2
y′[t] == x[t]
```

Look at the formula for n[x, y]:

```
n[x, y]
x
```

Explain why this tells you that no matter what solution pair

{x[t], y[t]}

you go with, you are guaranteed that:
→ If x[t] > 0, then y[t] is going up as t advances.
→ If x[t] < 0, then y[t] is going down as t advances.

□G.3.a.iii)

Here's a new diffeq system:

```
Clear[m, n, x, y, t];
m[x_, y_] = x - y;
n[x_, y_] = x - Sin[y];

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
x′[t] == x[t] - y[t]
y′[t] == -Sin[y[t]] + x[t]
```

Look at the formula for m[x, y]:

```
m[x, y]
x - y
```

Explain why this tells you that no matter what solution pair

{x[t], y[t]}

you go with, you are guaranteed that:
→ If x[t] > y[t], then x[t] is going up as t advances.
→ If x[t] < y[t], then x[t] is going down as t advances.

□G.3.a.iv)

Stay with the same diffeq sytem as in part iii) above:

```
Clear[m, n, x, y, t];
m[x_, y_] = x - y;
n[x_, y_] = x - Sin[y];
```

```
diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```
```
x′[t] == x[t] - y[t]
y′[t] == -Sin[y[t]] + x[t]
```
Look at the formula for n[x, y]:
```
n[x, y]
```
```
x - Sin[y]
```
Explain why this tells you that no matter what solution pair

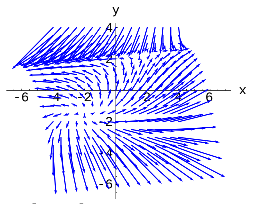
{x[t], y[t]}

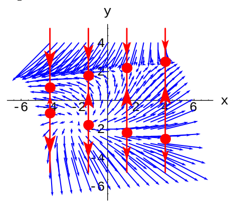
you go with, you are guaranteed that:

→ If x[t] > Sin[y[t]], then y[t] is going up as t advances.

→ If x[t] < Sin[y[t]], then y[t] is going down as t advances.

□G.3.b.i) 2D phase lines

Here's a new diffeq system:
```
Clear[m, n, x, y, t];
m[x_, y_] = x - y + 0.5;
n[x_, y_] = 0.4 x - 0.5 y^2 + 2;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```
```
x′[t] == 0.5 + x[t] - y[t]
y′[t] == 2 + 0.4 x[t] - 0.5 y[t]^2
```
And a flow plot:
```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.35, HeadSize → 0.3],
  {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

flows = Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```
```
{0.5 + x - y, 2 + 0.4 x - 0.5 y^2}
```


Look at this:
```
Clear[yphaseline];
yphaseline[x_] := PhaseLine[n[x, y], {y, ylow - 1, yhigh + 1}, Red, x];
yphaselines = Table[yphaseline[x], {x, xlow, xhigh, (xhigh - xlow)/3}];

Show[flows, yphaselines];
```


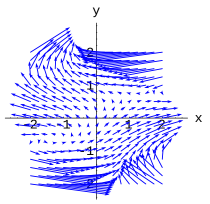
On each plotted phase line:

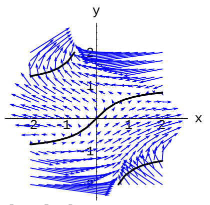
→ The up arrow signals an interval of y's for which n[x, y] > 0

→ The down arrow signals an interval of y's for which n[x, y] < 0.

Your job is to explain why this information meshes beautifully with the information in the flow plot.

□G.3.c.i)

Here is another diffeq system:
```
Clear[m, n, x, y, t];
m[x_, y_] = - (y^2 - 1) x - y;
n[x_, y_] = 0.4 E^(-0.3 x y) - 0.3 x y;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```
```
x′[t] == -y[t] + x[t] (1 - y[t]^2)
y′[t] == 0.4 E^(-0.3 x[t] y[t]) - 0.3 x[t] y[t]
```
And a flow plot:
```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
{xlow, xhigh} = {-2, 2};
{ylow, yhigh} = {-2, 2};
jump = 0.25;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.3, HeadSize → 0.15],
  {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

flows = Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```


Now look at this plot:
```
mcontourplot = ContourPlot[m[x, y],
  {x, xlow, xhigh}, {y, ylow, yhigh}, Contours → {0},
  ContourStyle → Thickness[0.01], ContourSmoothing → Automatic,
  ContourShading → False, DisplayFunction → Identity];

Show[flows, mcontourplot, DisplayFunction → $DisplayFunction];
```


The dark curves you see are the curves made of the points {x, y} for which

m[x, y] = 0.

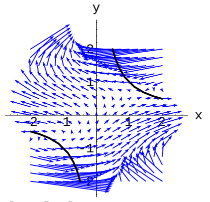
For {x, y}'s between the curves you are guaranteed that m[x, y] is either positive or negative. Use the flow plot to determine the strips on which

m[x, y] > 0

and the strips on which

m[x, y] < 0.

□G.3.c.ii)

Stay with the same diffeq system and look at this plot:
```
ncontourplot = ContourPlot[n[x, y],
  {x, xlow, xhigh}, {y, ylow, yhigh}, Contours → {0},
  ContourStyle → Thickness[0.01], ContourSmoothing → Automatic,
  ContourShading → False, DisplayFunction → Identity];

Show[flows, ncontourplot, DisplayFunction → $DisplayFunction];
```


The dark curves you see are the curves made of the points {x, y} for which

n[x, y] = 0.

For {x, y}'s between the curves you are guaranteed that n[x, y] is either positive or negative. Use the flow plot to determine the strips on which

n[x, y] > 0

and the strips on which

n[x, y] < 0.

G.4) Equilibrium points*

□G.4.a.i)

Here is a system of diffeq's:

```
Clear[m, n, x, y, t];
m[x_, y_] = -0.5 (x - 0.6) (1 - y/2);
n[x_, y_] = 0.4 x - 0.5 y + 1.7;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == -0.5 (-0.6 + x[t]) (1 - y[t]/2)
y′[t] == 1.7 + 0.4 x[t] - 0.5 y[t]
```

Folks search for the equilibrium points for this diffeq system by looking for points {x, y} for which both

m[x, y] = 0

and

n[x, y] = 0:

```
eqsols = Solve[{m[x, y] == 0, n[x, y] == 0}, {x, y}]
```

```
{{x → -1.75, y → 2.}, {x → 0.6, y → 3.88}}
```

This gives two equilbrium points:

```
{xeq1, yeq1} = {x, y} /. eqsols⟦1⟧
```

```
{-1.75, 2.}
```

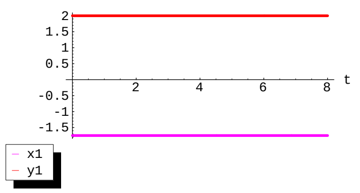
```
{xeq2, yeq2} = {x, y} /. eqsols⟦2⟧
```

```
{0.6, 3.88}
```

Here are plots of the solutions {x[t], y[t]} that start with

{x[0], y[0]} = {xeq1, yeq1}:

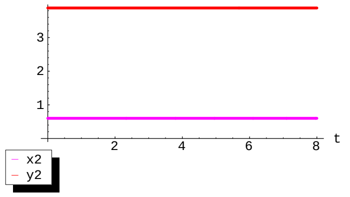
```
starterpoint = {xeq1, yeq1};
Clear[x, y, x1, y1, t];
equationx = x′[t] == m[x[t], y[t]];
equationy = y′[t] == n[x[t], y[t]];
starterx = x[0] == starterpoint⟦1⟧;
startery = y[0] == starterpoint⟦2⟧;
endtime = 8;
ndssol = NDSolve[{equationx, equationy, starterx, startery},
   {x[t], y[t]}, {t, 0, endtime}];
{x1[t_], y1[t_]} = {x[t] /. ndssol⟦1⟧, y[t] /. ndssol⟦1⟧};

Plot[{x1[t], y1[t]}, {t, 0, endtime},
 PlotStyle → {{Thickness[0.01], Magenta}, {Thickness[0.01], Red}},
 AxesLabel → {"t", ""}, PlotLegend → {"x1", "y1"}, LegendSize → 0.3];
```



Here are plots of the solutions {x[t], y[t]} that start with

{x[0], y[0]} = xeq2, yeq2}:

```
starterpoint = {xeq2, yeq2};
Clear[x, y, x2, y2, t];
equationx = (x'[t] == m[x[t], y[t]]);
equationy = (y'[t] == n[x[t], y[t]]);
starterx = (x[0] == starterpoint[[1]]);
startery = (y[0] == starterpoint[[2]]);
endtime = 8;

ndssol = NDSolve[{equationx, equationy,
   starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

{x2[t_], y2[t_]} = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};

Plot[{x2[t], y2[t]}, {t, 0, endtime},
       PlotStyle →
  {{Thickness[0.01], Magenta}, {Thickness[0.01], Red}},
 AxesLabel -> {"t", ""}, PlotLegend -> {"x2", "y2"}, LegendSize -> 0.3];
```



Why do these plots turn out the way they do?
What do the facts that

m[xeq1, yeq1] = 0,
n[xeq1, yeq1] = 0,
m[xeq2, yeq2] = 0,
n[xeq2, yeq2] = 0

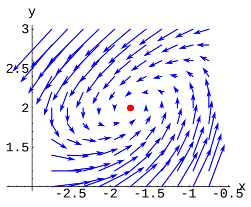
have to do with your answer?

□G.4.a.ii)

Stay with the same diffeq system as in part i) and look at the flow plot in the vicinity of equilibrium point {xeq1, yeq1}:

```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]}
{xlow, xhigh} = {xeq1 - 1, xeq1 + 1};
{ylow, yhigh} = {yeq1 - 1, yeq1 + 1};
jump = 0.2;
scalefactor = 0.6;
flowplot1 = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.6, HeadSize → 0.1],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];
equilib1 = Graphics[{Red, PointSize[0.03], Point[{xeq1, yeq1}]}];

Show[flowplot1, equilib1, Axes → True, AxesLabel → {"x", "y"}];
```

```
{-0.5 (-0.6 + x) (1 - y/2), 1.7 + 0.4 x - 0.5 y}
```

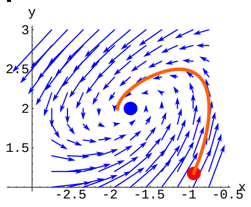


That dot in the middle is the equilibrium point {xeq1,yeq1}.

Read the flow plot to predict what happens to solutions {x[t], y[t]} that start near but not on the plotted equilbrium point.
Back up your predictions with some trajectory plots.
Got any idea why some folks call the plotted equilbrium point by the name "attractor?"

Click on the right for some code for you to edit to test out your predictions.

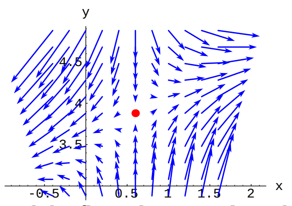
```
equilibriumpoint = {xeq1, yeq1};
starterpoint =
 {Random[Real, {xeq1 - 1, xeq1 + 1}], Random[Real, {yeq1 - 1, yeq1 + 1}]};

endtime = 10;

Clear[x, y, x1, y1, t];
equationx = x′[t] == m[x[t], y[t]];
equationy = y′[t] == n[x[t], y[t]];
starterx = x[0] == starterpoint⟦1⟧;
startery = y[0] == starterpoint⟦2⟧;
ndssol = NDSolve[{equationx, equationy, starterx, startery},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory];
trajectory[t_] = {x[t] /. ndssol⟦1⟧, y[t] /. ndssol⟦1⟧};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];
equilbriumplot =
 Graphics[{Blue, PointSize[0.06], Point[equilibriumpoint]}];

Show[flowplot1, starterplot, trajectoryplot, equilbriumplot,
 PlotRange → All, Axes → True, AxesLabel → {"x", "y"},
 DisplayFunction → $DisplayFunction];
```



□G.4.a.iii)

Stay with the same diffeq system as in part i) and look at the flow plot in the vicinity of equilibrium point {xeq2, yeq2}:

```
{xlow, xhigh} = {xeq2 - 1, xeq2 + 1};
{ylow, yhigh} = {yeq2 - 1, yeq2 + 1};
jump = 0.2;
flowplot2 = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.7, HeadSize → 0.1],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];

equilib2 = Graphics[{Red, PointSize[0.03], Point[{xeq2, yeq2}]}];
Show[flowplot2, equilib2, Axes → True, AxesLabel → {"x", "y"}];
```



Read the flow plot to predict what happens to solutions {x[t], y[t]} that start near but not on the plotted equilbrium point.
Back up your predictions with some trajectory plots.
Would it be fair to call this equilibrium point by the name "attractor?"

□G.4.b.i) The predator-prey model

This is another visit with the predator-prey model:

```
Clear[m, n, a, b, c, d, x, y, t];
m[x_, y_] = a x - b x y;
n[x_, y_] = -c y + d y x;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == a x[t] - b x[t] y[t]
y′[t] == -c y[t] + d x[t] y[t]
```

Here x[t] is the prey population at time t
and y[t] is the predator population at time t.

The equilibrium points come from:

```
Solve[{m[x, y] == 0, n[x, y] == 0}, {x, y}]
```

$\{\{x \to 0,\ y \to 0\},\ \{x \to \frac{c}{d},\ y \to \frac{a}{b}\}\}$

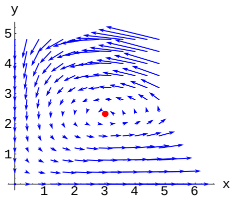
The only equilibrium point of any interest is:

$\{xeq, yeq\} = \{\frac{c}{d}, \frac{a}{b}\}$

$\{\frac{c}{d}, \frac{a}{b}\}$

Go with these sample choices of a, b, c and d and look at the flow plot:

```
a = 0.7;
b = 0.3;
c = 0.3;
d = 0.1;
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
{xlow, xhigh} = {0, 5};
{ylow, yhigh} = {0, 5};
jump = 0.4;
scalefactor = 0.5;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.5, HeadSize → 0.2],
  {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];
equilib = Graphics[{Red, PointSize[0.03], Point[{xeq, yeq}]}];

Show[flowplot, equilib, Axes → True, AxesLabel → {"x", "y"}];
```



What happens to solutions that start near the plotted equilbrium point?
Would it be fair to call this equilibrium point by the name "attractor?"

□G.4.b.ii) Feeding prey into the predator-prey model at a constant rate

This time you are asked to work with a modification of the predator-prey model in which you throw in extra prey at a constant rate of r prey units per time unit.
The resulting diffeq model is:

```
Clear[m, n, a, b, c, d, x, y, t, r];
m[x_, y_] = a x - b x y + r;
n[x_, y_] = -c y + d y x;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == r + a x[t] - b x[t] y[t]
y′[t] == -c y[t] + d x[t] y[t]
```

For this modification, the equilibrium points come from:

```
sols = Solve[{m[x, y] == 0, n[x, y] == 0}, {x, y}]
```

$\{\{y \to 0,\ x \to -\frac{r}{a}\},\ \{y \to \frac{a\,c + d\,r}{b\,c},\ x \to \frac{c}{d}\}\}$

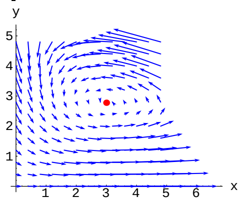
The only equilibrium point of any interest is:

```
{xeq, yeq} = {x, y} /. sols[[2]]
```

$\{\frac{c}{d}, \frac{a\,c + d\,r}{b\,c}\}$

Here is the flow plot shown with the equilibrium point {xeq, yeq} for the following sample values of r, a, b, c,and d:

```
r = 0.4;
a = 0.7;
b = 0.3;
c = 0.3;
d = 0.1;
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
{xlow, xhigh} = {0, 5};
{ylow, yhigh} = {0, 5};
jump = 0.4;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.5, HeadSize → 0.18],
  {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];
equillib = Graphics[{Red, PointSize[0.03], Point[{xeq, yeq}]}];

Show[flowplot, equillib, Axes → True, AxesLabel → {"x", "y"}];
```



What happens to solutions that start near the plotted equilbrium point?
Would it be fair to call this equilibrium point by the name "attractor?"

□G.4.b.iii)

Stay with the modification of the predator-prey model in which you throw in extra prey at a constant rate of r prey units per time unit.
The resulting diffeq system is:

```
Clear[m, n, a, b, c, d, x, y, t, r];
m[x_, y_] = a x - b x y + r;
n[x_, y_] = -c y + d y x;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == r + a x[t] - b x[t] y[t]
y′[t] == -c y[t] + d x[t] y[t]
```

Again, the equilibrium points come from:

```
sols = Solve[{m[x, y] == 0, n[x, y] == 0}, {x, y}]
```

$\{\{y \to 0,\ x \to -\frac{r}{a}\},\ \{y \to \frac{a\,c + d\,r}{b\,c},\ x \to \frac{c}{d}\}\}$

Again, the only equilibrium point of any interest is:

```
{xeq, yeq} = {x, y} /. sols[[2]]
```

$\{\frac{c}{d}, \frac{a\,c + d\,r}{b\,c}\}$

Use the same sample a, b, c,and d as used in the last part and look at the resulting simple formulas for {xeq, yeq}:

```
a = 0.7;
b = 0.3;
c = 0.3;
d = 0.1;
Expand[{xeq, yeq}]
```

```
{3., 2.33333 + 1.11111 r}
```

This tells you that, no matter what r is, the equilibrium population of the prey is 3 units.
This also tells you that the equilibrium population of the predators is about 2.3 + 1.1 r units.

Try to set r to conrol the long-term predator population to settle in at about 3.7 units.
Illustrate your choice of r with several decisive solution or trajectory plots.

□G.4.b.iv)

So far you have investigated only a couple of possible variations on harvesting and feeding the predator-prey system. Investigate some others. You might want to choose from:
What happens when you harvest prey at a constant rate?
What happens when you harvest predators at a constant rate?
What happens when you throw more predators in at a constant rate?

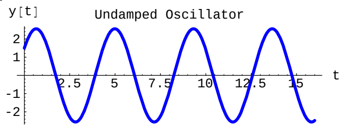
G.5) Oscillators and shock absorbers*

Calculus&*Mathematica* thanks lead mechanics Aaron Finley
and Don Happ of Don's Automotive in Homer, Illinois
for some expert advice on shock absorbers.

□G.5.a.i) The undamped oscillator

The differential equation of the undamped oscillator is
$y''[t] + c\,y[t] = 0$ with $c > 0$.
Here is a sample with
$c = 2.1$, $y[0] = 1.5$, $y'[0] = 3.0$:

```
c = 2.1;
initialdisplacement = 1.5;
initialvel = 3.0;
endtime = 16;

Clear[s, y, y, t];
ndssol = NDSolve[
 {y''[t] + c y[t]  == 0,
  y[0] == initialdisplacement,
   y'[0] == initialvel}, y[t], {t, 0, endtime}];

y[t_] = y[t] /. ndssol[[1]];
Plot[y[t], {t, 0, endtime},
   PlotStyle -> {{Thickness[0.01], Blue}},
   PlotRange -> All,
   AxesLabel -> {"t", "y[t]"}, PlotLabel -> "Undamped Oscillator"];
```



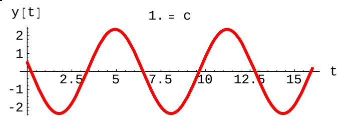
This gives you an idea of how your grandfather's tired old
Cadillac with worn-out shock aborbers will
oscillate immediately after you drive it over a big pot hole.

Say why you should have been able to predict in advance, with no plotting, that as t advances from from 0, the plot of y[t] had to go up before it could go down.
What happens when you keep y[0] the same but you go with
$y'[0] < 0$?
Back up you answer with a nice plot.

□G.5.a.ii) Frequency of the undamped oscillator

The differential equation of the undamped oscillator is
$y''[t] + c\,y[t] = 0$ with $c > 0$.
Here's a new sample with
$c = 1.0$, $y[0] = 0.5$, $y'[0] = -2.3$:

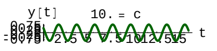
```
c = 1.0;
initialdisplacement = 0.5;
initialvel = -2.3;
endtime = 16;

Clear[s, y, y, t];
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;

littlec = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → c "= c"];
```



You may want to enlarge this graph by clicking on it and then dragging
one of the corners.

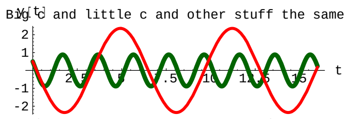
Here is what you get when you give a hefty increase to c and keep everything else the same:

```
c = 10.0;
Clear[y, t];
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;

bigc = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.015], DarkGreen}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → c "= c"];
```



Here they are together:

```
Show[bigc, littlec,
 PlotLabel → "Big c and little c and other stuff the same"];
```

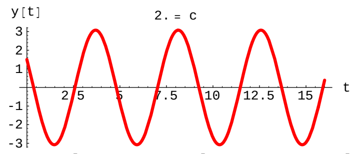


Most folks like to say that the frequency of an undamped oscillator is the measurement of the number of oscillations the oscillator makes every unit of time t.
Run additional experiments if you like and then you make the call:
Does increasing c increase or decrease the frequency of the oscillator?

□G.5.a.iii) Amplitude of undamped oscillators

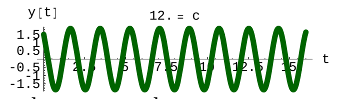
The differential equation of the undamped oscillator is
$y''[t] + c\,y[t] = 0$ with $c > 0$.
Here's yet another new sample with
$c = 2.0$, $y[0] = 1.5$, $y'[0] = -3.8$:

```
c = 2.0;
initialdisplacement = 1.5;
initialvel = -3.8;
endtime = 16;

Clear[y, t];
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;

littlec = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → c "= c"];
```



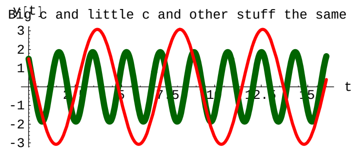
Here is what you get when you give a hefty increase to c and keep everything else the same:

```
c = 12.0;
Clear[y, t];
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;

bigc = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.02], DarkGreen}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → c "= c"];
```



Here they are together:

```
Show[bigc, littlec,
 PlotLabel → "Big c and little c and other stuff the same"];
```



Most folks like to say that the amplitude of an undamped oscillator is the measurement of the perpendicular distance between the t-axis and the highest point on the plot of the oscillator.
Run additional experiments if you like and then you make the call:
Does increasing c result in an increase or a decrease of the amplitude of the oscillator?

□G.5.a.iv) Cadillac versus Porsche

When you remove the shock absorbers from a car and you drive the car over a pot hole, then the the oscillation of the car immediately after hitting the pot hole is approximately modeled by the differential equation

$$y''[t] + c\,y[t] = 0.$$

When the car is your grandfather's Cadillac Sedan deVille you get a certain c in the model. When the car is your sister's Porsche 911, you get another c in the model.

Which do you think gives you thae larger c : The Cadillac Sedan deVille or the Porsche 911?
Explain how you came to your opinion.

□G.5.b.i) The damped oscillator

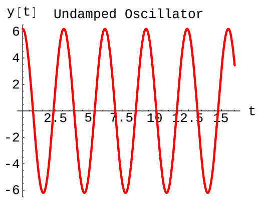
The differential equation of the undamped oscillator is

$$y''[t] + c\,y[t] = 0 \text{ with } c > 0.$$

Here's yet another sample with

$$c = 4.1, y[0] = 6.2, y'[0] = 0:$$

```
c = 4.1;
initialdisplacement = 6.2;
initialvel = 0;
endtime = 16;

Clear[y, t];
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];

undamped = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Undamped Oscillator"];
```



This gives you an idea of how your grandfather's tired old Cadillac with worn-out shock aborbers will oscillate immediately after you drive it over a big pot hole.

Here is what you get when you go with a small, positive b and throw a $b\,y'[t]$ term into

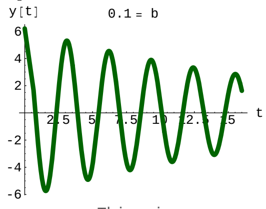
$$y''[t] + c\,y[t] = 0$$

to get

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

and you keep everything else the same:

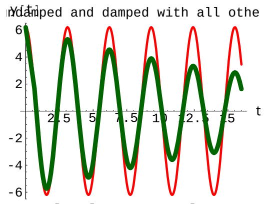
```
b = 0.1;
Clear[y, t];
ndssol =
 NDSolve[{y″[t] + b y′[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];

smallb = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.02], DarkGreen}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → b "= b"];
```



This gives you an idea of how your grandfather's Cadillac equipped with brand new light duty Monroe-matic shock aborbers will oscillate immediately after you drive it over a big pot hole.

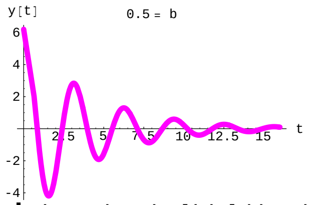
Here they are together:

```
Show[undamped, smallb,
 PlotLabel → "              undamped and damped with all
    other stuff the same"];
```

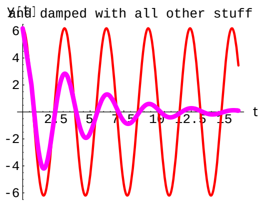


See what happens when you increase b just a little bit:

```
b = 0.5;

Clear[y, t];
ndssol = NDSolve[
  {y''[t] + b y'[t] + c y[t] == 0,
   y[0] == initialdisplacement,
    y′[0] == initialvel}, y[t], {t, 0, endtime}];

y[t_] = y[t] /. ndssol[[1]];
slightlybiggerb = Plot[y[t], {t, 0, endtime},
   PlotStyle -> {{Thickness[0.02], Magenta}},
   PlotRange -> All,
   AxesLabel -> {"t", "y[t]"}, PlotLabel -> b "= b"];
```



```
Show[undamped, slightlybiggerb,
 PlotLabel → "undamped and damped with all other stuff the same"];
```



This gives you an idea of how your grandfather's Cadillac equipped with brand new medium duty Monroe-matic shock aborbers will oscillate immediately after you drive it over a big pot hole.

Why do you think folks like to say that when you go with a small, positive b, and a positive c, then

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

is the differential equation of a damped oscillator?
Does increasing b increase or decrease the effect of the damping?

□G.5.b.ii) Too much of a good thing: The overdamped oscillator

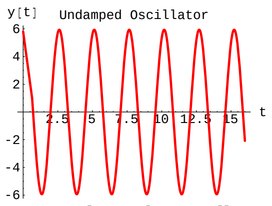
The differential equation of the undamped oscillator is

$$y''[t] + c\,y[t] = 0 \text{ with } c > 0.$$

Here's a new sample with

$$c = 6.2, y[0] = 5.8, y'[0] = 3.1 :$$

```
c = 6.2;
initialdisplacement = 5.8;
initialvel = 3.1;
endtime = 16;

Clear[y, t];
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];

undamped = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Undamped Oscillator"];
```



You can damp this oscillator by going with a small positive b and throwing in a

b y′[t] term

into

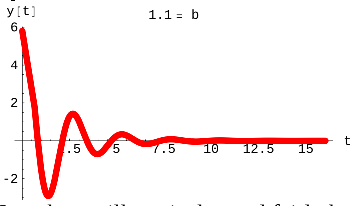
$$y''[t] + c\,y[t] = 0$$

to get

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

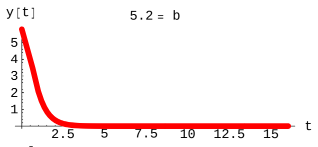
and keeping everything else the same:

```
b = 1.1;
Clear[y, t];
ndssol =
 NDSolve[{y″[t] + b y′[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;

smallb = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.02], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → b "= b"];
```



Now the oscillator is damped fairly heavily.
See what happens when you damp it even more:

```
b = 5.2;
Clear[y, t];
ndssol =
 NDSolve[{y″[t] + b y′[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;

smallb = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.02], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → b "= b"];
```



Whoosh.

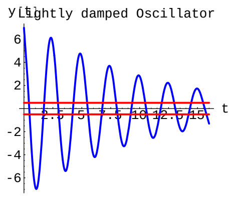
```
This is what your grandfather's old Cadillac equipped with
brand new extra heavy duty Monroe-matic Gas Magnum shock absorbers will
do immediately after you drive it over a big pot hole.
```

Why do most folks say that when you go with b's that are too large, you run the risk of overdamping the oscillator?
Why is it unwise to put dump truck shock absorbers on a Porsche 911?

□G.5.b.iii) Custom damping

Look at this:

```
b = 0.2;
c = 6.2;
initialdisplacement = 7.0;
initialvel = -9.0;
endtime = 16;

Clear[y, t];
ndssol =
 NDSolve[{y″[t] + b y′[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;

lightlydamped = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Blue}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Lightly damped Oscillator",
  Epilog → {{Thickness[0.01], Red, Line[{{0, 0.5}, {endtime, 0.5}}]},
    {Thickness[0.01], Red, Line[{{0, -0.5}, {endtime, -0.5}}]}}];
```



This is the plot of the damped oscillator coming from

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

with $y[0] = 7.0$, $y'[0] = -9.0$, $b = 0.2$ and $c = 6.2$.

Your job is to come up with a new damping coefficient b so that when you keep everything else the same, then the plot of the new damped oscillator:
→ oscillates between the red lines for when t is bigger than 6 but
→ oscillates above, between and below the red lines for as t runs from 0 to 6.
Illustrate your answer with a plot.

G.5.c) Formulas

Here is *Mathematica* cranking out an exact formula for the solution of the undamped oscillator diffeq

$$y''[t] + 4.0\,y[t] = 0$$

with $y[0] = 3.3$ and $y'[0] = 0$:

```
You will learn how to get this formula for yourself later in the course.
```

```
thisc = 4.0;
thisyzero = 3.3;
thisyprime0 = 0;

Clear[c, y1, y, t, initialdisplacement, initialvel];
undampedoscdiffeq = y″[t] + c y[t] == 0;
dsol = DSolve[{undampedoscdiffeq, y[0] == initialdisplacement,
    y′[0] == initialvel}, y[t], t] /. {c → thisc,
   initialdisplacement → thisyzero, initialvel → thisyprime0};

y1[t_] = Chop[ComplexExpand[y[t] /. dsol⟦1⟧]]
3.3 Cos[2. t]
```

Here is *Mathematica* cranking out an exact formula for the solution of the same oscillator after a damping term has been inserted:

```
You will learn how to get this formula for yourself later in the course.
```

```
thisb = 0.8;
thisc = 4.0;
thisyzero = 3.3;
thisyprime0 = 0;

Clear[b, c, y2, y, t, initialdisplacement, initialvel];
dampedoscdiffeq = y″[t] + b y′[t] + c y[t] == 0;
dsol = DSolve[{dampedoscdiffeq, y[0] == initialdisplacement,
    y′[0] == initialvel}, y[t], t] /. {b → thisb, c → thisc,
   initialdisplacement → thisyzero, initialvel → thisyprime0};

y2[t_] = Chop[ComplexExpand[y[t] /. dsol⟦1⟧]]
```

$3.3\,E^{-0.4\,t}\,\mathrm{Cos}[1.95959\,t] + 0.67361\,E^{-0.4\,t}\,\mathrm{Sin}[1.95959\,t]$

Look at both formulas:

```
y1[t]
3.3 Cos[2. t]
y2[t]
```

$3.3\,E^{-0.4\,t}\,\mathrm{Cos}[1.95959\,t] + 0.67361\,E^{-0.4\,t}\,\mathrm{Sin}[1.95959\,t]$

When you put the damping term on the oscillator, do you change the frequency of the oscillator?

□G.5.d.i) The negatively damped oscillator: b < 0

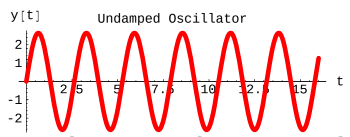
The differential equation of the undamped linear oscillator is

$$y''[t] + c\,y[t] = 0 \text{ with } c > 0.$$

Here's yet another new sample with

$$c = 5.7,\ y[0] = 0,\ y'[0] = 6.3:$$

```
c = 5.7;
initialdisplacement = 0;
initialvel = 6.3;
endtime = 16;
Clear[y, t]
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;
undamped = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.015], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Undamped Oscillator"];
```



Here is what you get when you go with a small, negative b and throw a

b y'[t] term

into

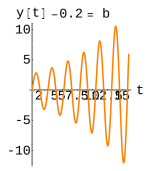
y''[t] + c y[t] = 0

to get

y''[t] + b y'[t] + c y[t] = 0

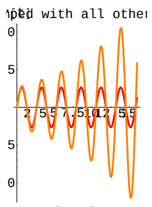
and keep everything else the same:

```
b = -0.2;
Clear[y, t]
ndssol =
 NDSolve[{y''[t] + b y'[t] + c y[t] == 0, y[0] == initialdisplacement,
   y'[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];
smallnegativeb =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.015], Orange}},
  PlotRange → All, AxesLabel → {"t", "y[t]"}, PlotLabel → b "= b"];
```



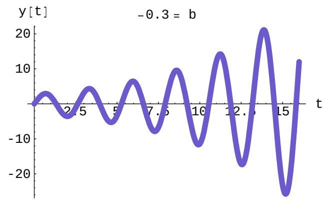
Here they are together:

```
Show[undamped, smallnegativeb,
 PlotLabel → "undamped and damped with all other stuff the same"];
```



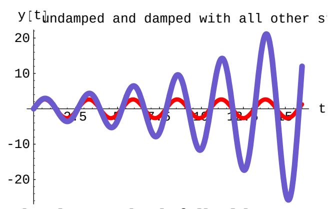
See what happens when you make b slightly more negative:

```
b = -0.3;
Clear[y, t]
ndssol =
 NDSolve[{y''[t] + b y'[t] + c y[t] == 0, y[0] == initialdisplacement,
   y'[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];
slightlymorenegb = Plot[y[t],
  {t, 0, endtime}, PlotStyle → {{Thickness[0.02], SlateBlue}},
  PlotRange → All, AxesLabel → {"t", "y[t]"}, PlotLabel → b "= b",
  AspectRatio → 1/GoldenRatio];
```



Compare:

```
Show[undamped,
 slightlymorenegb, PlotLabel → "                    undamped
   and damped with all other stuff the same",
 AspectRatio → 1/GoldenRatio];
```



Why do you think folks like to say that when you go with a negative b, and a positive c, then

y''[t] + b y'[t] + c y[t] = 0

is the differential equation of a self-excited oscillator?

□G.5.d.ii) What about c < 0?

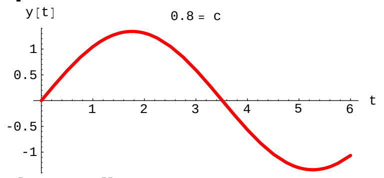
The differential equation of the undamped oscillator is

y''[t] + c y[t] = 0 with c > 0.

Here's a new sample with

c = 0.8, y[0] = 0, y'[0] = 1.2 :

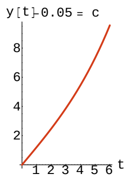
```
c = 0.8;
initialdisplacement = 0;
initialvel = 1.2;
endtime = 6;
Clear[y, t]
ndssol = NDSolve[{y''[t] + c y[t] == 0, y[0] == initialdisplacement,
    y'[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];
posc =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Red}},
  PlotRange → All, AxesLabel → {"t", "y[t]"}, PlotLabel → c "= c"];
```



A lazy oscillator.

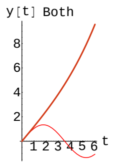
Here's what you get when you make c just slightly negative and keep everything else the same:

```
c = -0.05;
Clear[y, t]
ndssol = NDSolve[{y''[t] + c y[t] == 0, y[0] == initialdisplacement,
    y'[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];
negc = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.02], EnglishRed}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → c "= c"];
```



Here they are together:

```
Show[posc, negc, PlotLabel → "Both"];
```



Give your opinion on why most folks say that when c < 0, then

y''[t] + c y[t] = 0

is not the differential equation of any oscillator.

G.6) Damped oscillators, undamped oscillators and van der Pol's nonlinear oscillator*

□G.6.a) Undamped oscillators

The differential equation of the undamped oscillator is

y''[t] + c y[t] = 0 with c > 0.

Here is one such:

```
Clear[t, y]
c = 1.2;
undampedoscillator = y″[t] + c y[t] == 0
```

```
1.2 y[t] + y″[t] == 0
```

To make this second order diffeq into a system of two first order diffeq's, you put

x[t] = y'[t]

and then replace y'[t] with x[t] and replace y''[t] with x'[t]:

```
Clear[x]
ColumnForm[Thread[{new = (y'[t] == x[t]),
undampedoscillator /. {y'[t] -> x[t], y''[t] -> x'[t]}}]]
```

```
y′[t] == x[t]
1.2 y[t] + x′[t] == 0
```

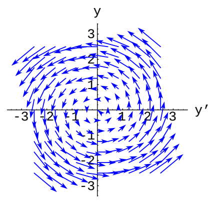
Clean this up:

```
Clear[m, n, x, y, t]
m[x_, y_] = - 1.2 y;
n[x_, y_] = x;
diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == -1.2 y[t]
y′[t] == x[t]
```

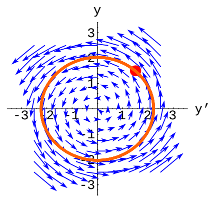
Now look at:

```
Clear[Field]
Field[x_, y_] = {m[x, y], n[x, y]};
scalefactor = 0.3;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.3, HeadSize → 0.3],
  {x, -2.5, 2.5, 5/12}, {y, -2.5, 2.5, 5/12}];
Show[flowplot, PlotRange → All, Axes → True, AxesLabel → {"y'", "y"},
 DisplayFunction → $DisplayFunction];
```



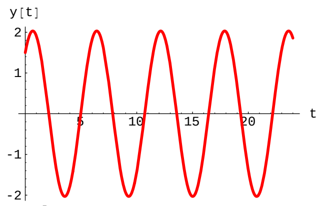
Throw in a trajectory:

```
starterpoint = {1.5, 1.5};
{a, b} = starterpoint;
Clear[x, y, t]
equationx = (x′[t] == m[x[t], y[t]]);
equationy = (y′[t] == n[x[t], y[t]]);
starterx = (x[0] == a);
startery = (y[0] == b);
endtime = 24.0;

ndssol = NDSolve[{equationx, equationy,
  starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};

trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
   PlotStyle -> {{CadmiumOrange, Thickness[0.015]}},
       DisplayFunction -> Identity];

starterplot = Graphics[{Red, PointSize[0.06],
          Point[starterpoint]}];

firstphaseportrait = Show[flowplot, starterplot,
  trajectoryplot,
  PlotRange -> All, Axes -> True, AxesLabel -> {"y'", "y"},
    DisplayFunction -> $DisplayFunction];
```



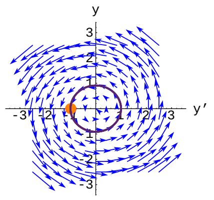
Look at the corresponding solution y[t]:

```
firstyplot = Plot[y[t] /. ndssol[[1]], {t, 0, endtime},
  PlotStyle → {{Red, Thickness[0.01]}}, AspectRatio → 1/GoldenRatio,
  AxesLabel → {"t", "y[t]"}];
```



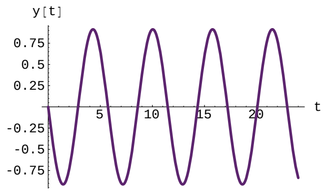
Another:

```
starterpoint = {-1.0, 0};
{a, b} = starterpoint;
Clear[x, y, t]
equationx = (x′[t] == m[x[t], y[t]]);
equationy = (y′[t] == n[x[t], y[t]]);
starterx = (x[0] == a);
startery = (y[0] == b);
endtime = 24.0;

ndssol = NDSolve[{equationx, equationy,
  starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};

trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
   PlotStyle -> {{Thickness[0.01], UltramarineViolet}},
       DisplayFunction -> Identity];

starterplot = Graphics[{PointSize[0.06], Orange,
          Point[starterpoint]}];

secondphaseportrait = Show[flowplot, starterplot,
  trajectoryplot,
  PlotRange -> All, Axes -> True, AxesLabel -> {"y'", "y"},
    DisplayFunction -> $DisplayFunction];
```



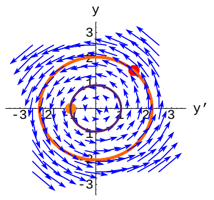
And the corresponding solution y[t]:

```
secondyplot = Plot[y[t] /. ndssol[[1]], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], UltramarineViolet}},
  AspectRatio → 1/GoldenRatio, AxesLabel → {"t", "y[t]"}];
```



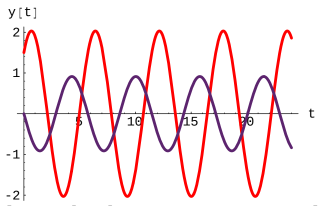
Compare the trajectories on the same axes:

```
Show[firstphaseportrait, secondphaseportrait];
```



Compare the solution curves y[t]:

```
Show[firstyplot, secondyplot];
```



Play with other starting points and then describe how solutions of the undamped oscillator

y"[t] + c y[t] = 0

look.

Why do you think folks call these things undamped oscillators?

□G.6.b) Damped oscillators

Here is the diffeq of an undamped oscillator as studied above:

```
Clear[t, y]
c = 1.2;
undampedoscillator = y″[t] + c y[t] == 0
```
```
1.2 y[t] + y″[t] == 0
```

To get a damped version of this oscillator, you throw in a shock absorber term

b y'[t]

with b positive but not too large:

```
b = 0.2;
dampedoscillator = y″[t] + b y′[t] + c y[t] == 0
```
```
1.2 y[t] + 0.2 y′[t] + y″[t] == 0
```

To make this second order differential equation into a system of two first order differential equations, you put

x[t] = y'[t]

and then replace y'[t] with x[t] and replace y"[t] with x'[t]:

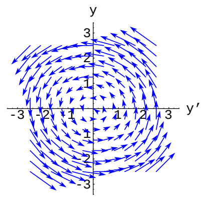
```
Clear[x];
ColumnForm[Thread[{new = (y'[t] == x[t]),
dampedoscillator /. {y'[t] -> x[t], y''[t] -> x'[t]}}]]
```
```
y′[t] == x[t]
0.2 x[t] + 1.2 y[t] + x′[t] == 0
```

Clean this up:

```
Clear[m, n, x, y, t];
m[x_, y_] = - c y - b x;
n[x_, y_] = x;

diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```
```
x′[t] == -0.2 x[t] - 1.2 y[t]
y′[t] == x[t]
```

Now look at:

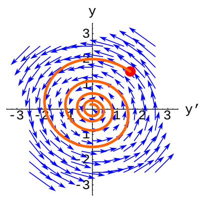
```
Clear[Field];
Field[x_, y_] = {m[x, y], n[x, y]};
scalefactor = 0.3;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.3, HeadSize → 0.3],
   {x, -2.5, 2.5, 5/12}, {y, -2.5, 2.5, 5/12}];
Show[flowplot, PlotRange → All, Axes → True, AxesLabel → {"y'", "y"},
 DisplayFunction → $DisplayFunction];
```



Throw in a trajectory:

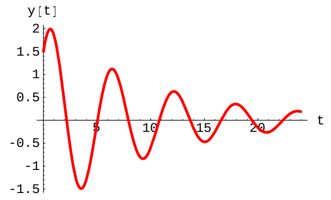
```
starterpoint = {1.5, 1.5};
{a, b} = starterpoint;
Clear[x, y, t]
equationx = (x′[t] == m[x[t], y[t]]);
equationy = (y′[t] == n[x[t], y[t]]);
starterx = (x[0] == a);
startery = (y[0] == b);
endtime = 24.0;

ndssol = NDSolve[{equationx, equationy,
   starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};
```

```
trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
    PlotStyle -> {{CadmiumOrange, Thickness[0.015]}},
        DisplayFunction -> Identity];

starterplot = Graphics[{Red, PointSize[0.06],
          Point[starterpoint]}];

firstphaseportrait = Show[flowplot, starterplot,
   trajectoryplot,
  PlotRange -> All, Axes -> True, AxesLabel -> {"y'", "y"},
   DisplayFunction -> $DisplayFunction];
```



Look at the corresponding solution y[t]:

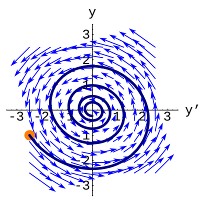
```
firstsolutionplot = Plot[y[t] /. ndssol[[1]], {t, 0, endtime},
  PlotStyle → {{Red, Thickness[0.01]}}, AspectRatio → 1/GoldenRatio,
  AxesLabel → {"t", "y[t]"}];
```



Another:

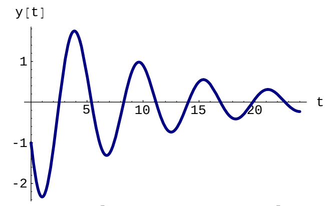
```
starterpoint = {-2.5, -1.0};
{a, b} = starterpoint;
Clear[x, y, t]
equationx = (x′[t] == m[x[t], y[t]]);
equationy = (y′[t] == n[x[t], y[t]]);
```

```
starterx = (x[0] == a);
startery = (y[0] == b);
endtime = 24.0;

ndssol = NDSolve[{equationx, equationy,
  starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};

trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
    PlotStyle -> {{Thickness[0.015], NavyBlue}},
        DisplayFunction -> Identity];

starterplot = Graphics[{Orange, PointSize[0.06],
          Point[starterpoint]}];

secondphaseportrait = Show[flowplot, starterplot,
   trajectoryplot,
  PlotRange -> All, Axes -> True, AxesLabel -> {"y'", "y"},
   DisplayFunction -> $DisplayFunction];
```



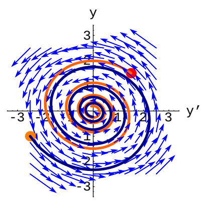
Look at the corresponding solution y[t]:

```
secondsolutionplot = Plot[y[t] /. ndssol[[1]],
  {t, 0, endtime}, PlotStyle → {{Thickness[0.01], NavyBlue}},
  AspectRatio → 1/GoldenRatio, AxesLabel → {"t", "y[t]"}];
```



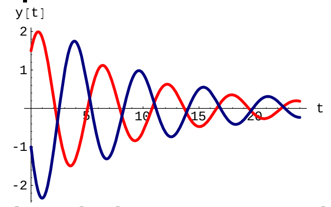
Compare the trajectories on the same axes:

```
Show[firstphaseportrait, secondphaseportrait];
```



Look at the two solution curves:

```
Show[firstsolutionplot, secondsolutionplot];
```



Play with other starting points and then describe how solutions of the undamped oscillator

y"[t] + b y'[t] + c y[t] = 0

with b small and positive, and c positive look.
Why do you think folks call these things damped oscillators?

□G.6.c) Negatively damped oscillators

Here is the differential equation of an undamped oscillator as studied in part a) above.

```
Clear[t, y]
c = 1.2;
undampedoscillator = y″[t] + c y[t] == 0
```

```
1.2 y[t] + y″[t] == 0
```

To get a negatively damped version of this oscillator, you throw in a shock absorber term

b y'[t]

with b negative but not too large:

```
b = -0.15;
dampedoscillator = y″[t] + b y′[t] + c y[t] == 0
```

```
1.2 y[t] - 0.15 y′[t] + y″[t] == 0
```

To make this second order differential equation into a system of two first order differential equations, you put

x[t] = y'[t]

and then replace y'[t] with x[t] and replace y"[t] withj x'[t]:

```
Clear[x]
ColumnForm[Thread[{new = (y'[t] == x[t]),
dampedoscillator /. {y'[t] -> x[t], y''[t] -> x'[t]}}]]
```

```
y′[t] == x[t]
-0.15 x[t] + 1.2 y[t] + x′[t] == 0
```

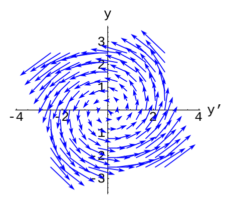
Clean this up:

```
Clear[m, n, x, y, t]
m[x_, y_] = - c y - b x;
n[x_, y_] = x;
diffeqsystem = ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});
ColumnForm[Thread[diffeqsystem]]
```

```
x′[t] == 0.15 x[t] - 1.2 y[t]
y′[t] == x[t]
```

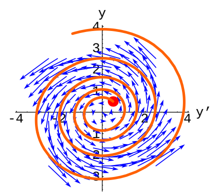
Now look at:

```
Clear[Field]
Field[x_, y_] = {m[x, y], n[x, y]};
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.4, HeadSize → 0.3],
   {x, -2.5, 2.5, 5/12}, {y, -2.5, 2.5, 5/12}];
Show[flowplot, PlotRange → All, Axes → True, AxesLabel → {"y'", "y"},
 DisplayFunction → $DisplayFunction];
```



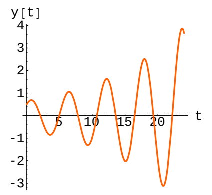
Throw in a trajectory:

```
starterpoint = {0.5, 0.5};
{a, b} = starterpoint;
Clear[x, y, t];
equationx = (x′[t] == m[x[t], y[t]]);
equationy = (y′[t] == n[x[t], y[t]]); starterx = (x[0] == a);
startery = (y[0] == b);
endtime = 24.0;

ndssol = NDSolve[{equationx, equationy,
  starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};

trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
   PlotStyle -> {{CadmiumOrange, Thickness[0.015]}},
       DisplayFunction -> Identity];

starterplot = Graphics[{Red, PointSize[0.06],
          Point[starterpoint]}];

firstphaseportrait = Show[flowplot, starterplot,
  trajectoryplot,
  PlotRange -> All, Axes -> True, AxesLabel -> {"y'", "y"},
   DisplayFunction -> $DisplayFunction];
```



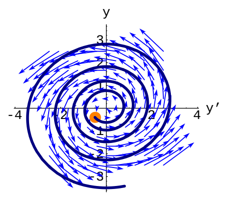
Look at the corresponding solution y[t]:

```
firstsolutionplot =
 Plot[y[t] /. ndssol⟦1⟧, {t, 0, endtime}, AspectRatio → 1,
  PlotRange → All, PlotStyle → {{CadmiumOrange, Thickness[0.01]}},
  AxesLabel → {"t", "y[t]"}];
```



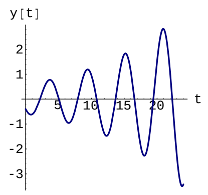
Another:

```
starterpoint = {-0.5, -0.4};
{a, b} = starterpoint;
Clear[x, y, t]
equationx = x′[t] == m[x[t], y[t]];
equationy = y′[t] == n[x[t], y[t]];
starterx = x[0] == a;
startery = y[0] == b;
endtime = 24.0;
ndssol = NDSolve[{equationx, equationy, starterx, startery},
  {x[t], y[t]}, {t, 0, endtime}];
Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol⟦1⟧, y[t] /. ndssol⟦1⟧};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{Thickness[0.015], NavyBlue}}, DisplayFunction → Identity];
starterplot =
 Graphics[{PointSize[0.06], Orange, Point[starterpoint]}];
secondphaseportrait = Show[flowplot, starterplot, trajectoryplot,
  PlotRange → All, Axes → True, AxesLabel → {"y'", "y"},
  DisplayFunction → $DisplayFunction];
```



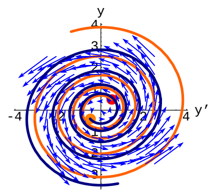
And the corresponding solution y[t]:

```
secondsolutionplot = Plot[y[t] /. ndssol⟦1⟧, {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], NavyBlue}}, AspectRatio → 1,
  PlotRange → All, AxesLabel → {"t", "y[t]"}];
```



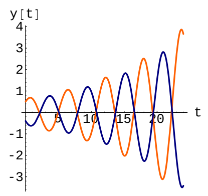
Compare the two trajectories on the same axes:

```
Show[firstphaseportrait, secondphaseportrait];
```



Compare the solution curves y[t]:

```
Show[firstsolutionplot, secondsolutionplot];
```



Play with other starting points and then describe how solutions of the negatively damped oscillator

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

with b small and negative, and c positive look.
Why do you think folks call these things self-excited oscillators?

□ G.6.d.i) The nonlinear van der Pol heartbeat oscillator

You can get an undamped oscillator this way:

```
Clear[y, t, b]
c = 1.2;
y″[t] + b y′[t] + c y[t] == 0 /. b → 0
```

```
1.2 y[t] + y″[t] == 0
```

You can get a damped oscillator by making b positive and small:

```
Clear[y, t, b]
y″[t] + b y′[t] + c y[t] == 0 /. b → 0.3
```

```
1.2 y[t] + 0.3 y′[t] + y″[t] == 0
```

You can get a negatively damped (self-excited) oscillator by making b negative and small:

```
Clear[y, t, b]
y″[t] + b y′[t] + c y[t] == 0 /. b → -0.23
```

```
1.2 y[t] - 0.23 y′[t] + y″[t] == 0
```

Here's how you can get a van der Pol oscillator:

```
Clear[t, y]
vanderpoloscillator = y″[t] + (y[t]^2 - 1) y′[t] + c y[t] == 0
```

```
1.2 y[t] + (-1 + y[t]^2) y′[t] + y″[t] == 0
```

When $-1 < y[t] < 1$, do you expect the van der Pol oscillator to behave like an undamped, a damped or a negatively damped oscillator?

When $1 < y[t]$ or $y[t] < -1$, do you expect the van der Pol oscillator to behave like an undamped, a damped or a negatively damped oscillator?

□G.6.d.ii)

Another look at a van der Pol oscillator:

```
Clear[t, y]
vanderpoloscillator = (y''[t] + (y[t]^2 - 1) y'[t] + c y[t] == 0)
```

```
1.2 y[t] + (-1 + y[t]^2) y′[t] + y″[t] == 0
```

To make this second order differential equation into a system of two first order differential equations, you put

$$x[t] = y'[t]$$

and then replace y'[t] by x[t] and replace y"[t] by x'[t]:

```
Clear[x]
ColumnForm[Thread[{new = (y'[t] == x[t]),
vanderpoloscillator /. {y'[t] -> x[t], y''[t] -> x'[t]}}]]
```

```
y′[t] == x[t]
1.2 y[t] + x[t] (-1 + y[t]^2) + x′[t] == 0
```

Clean this up:

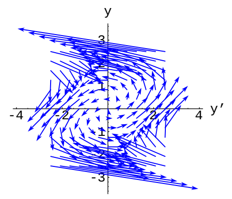
```
Clear[m, n, x, y, t]
m[x_, y_] = -(y^2 - 1) x - c y;
n[x_, y_] = x;
diffeqsystem = {x′[t], y′[t]} == {m[x[t], y[t]], n[x[t], y[t]]};
 diffeqsystem
```

```
x′[t] == -1.2 y[t] + x[t] (1 - y[t]^2)
y′[t] == x[t]
```

Now look at:

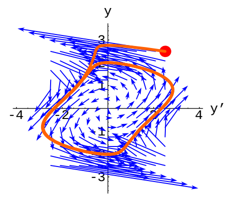
```
Clear[Field]
Field[x_, y_] = {m[x, y], n[x, y]}
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.4, HeadSize → 0.3],
   {x, -2.5, 2.5, 5/12}, {y, -2.5, 2.5, 5/12}];
Show[flowplot, PlotRange → All, Axes → True, AxesLabel → {"y'", "y"},
 DisplayFunction → $DisplayFunction];
```

```
{-1.2 y + x (1 - y^2), x}
```



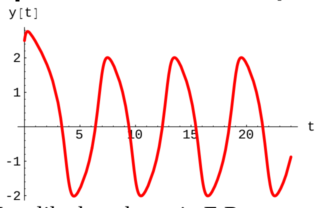
Cowabunga.
Throw in a trajectory:

```
{a, b} = {2.5, 2.5};
starterpoint = {a, b};
Clear[x, y, t]
equationx = x′[t] == m[x[t], y[t]];
equationy = y′[t] == n[x[t], y[t]];
starterx = x[0] == a;
startery = y[0] == b;
endtime = 24.0;
ndssol = NDSolve[{equationx, equationy, starterx, startery},
   {x[t], y[t]}, {t, 0, endtime}];
Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol⟦1⟧, y[t] /. ndssol⟦1⟧};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];
firstphaseportrait = Show[flowplot, starterplot, trajectoryplot,
  PlotRange → All, Axes → True, AxesLabel → {"y'", "y"},
  DisplayFunction → $DisplayFunction];
```



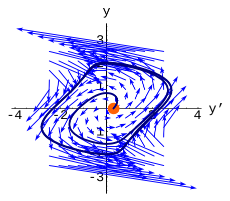
Look at the corresponding solution y[t]:

```
firstsolutionplot = Plot[y[t] /. ndssol[[1]], {t, 0, endtime},
  PlotStyle → {{Red, Thickness[0.01]}}, AspectRatio → 1/GoldenRatio,
  AxesLabel → {"t", "y[t]"}];
```



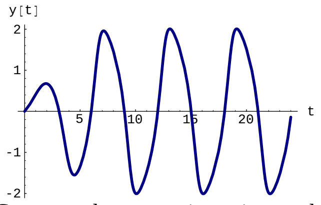
Just like heartbeats in E.R.
Another:

```
starterpoint = {0.3, 0.0};
{a, b} = starterpoint;
Clear[x, y, t]
equationx = (x'[t] == m[x[t], y[t]]);
equationy = (y'[t] == n[x[t], y[t]]);
starterx = (x[0] == a);
startery = (y[0] == b);
endtime = 24.0;

ndssol = NDSolve[{equationx, equationy,
  starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};

trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
   PlotStyle -> {{Thickness[0.01], NavyBlue}},
        DisplayFunction -> Identity];

starterplot = Graphics[{CadmiumOrange, PointSize[0.06],
          Point[starterpoint]}];

secondphaseportrait = Show[flowplot, starterplot,
  trajectoryplot,
  PlotRange -> All, Axes -> True, AxesLabel -> {"y'", "y"},
   DisplayFunction -> $DisplayFunction];
```



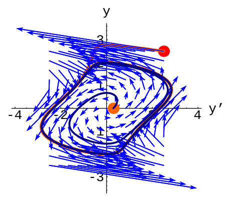
And the corresponding solution y[t]:

```
secondsolutionplot = Plot[y[t] /. ndssol[[1]],
  {t, 0, endtime}, PlotStyle → {{Thickness[0.01], NavyBlue}},
  AspectRatio → 1/GoldenRatio, AxesLabel → {"t", "y[t]"}];
```



Compare the two trajectories on the same axes:

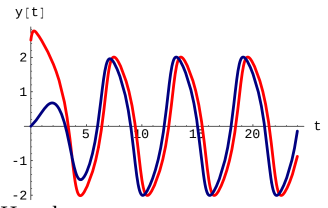
```
Show[firstphaseportrait, secondphaseportrait];
```



Folks like to call this a "limit cycle."

See the two solution curves:

```
Show[firstsolutionplot, secondsolutionplot];
```

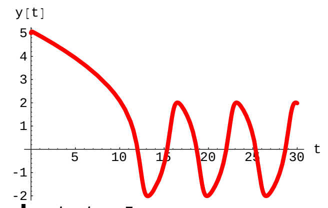


Heartbeats.
Look carefully at the flow plot and try to see how the flow forces the limit cycle to happen.

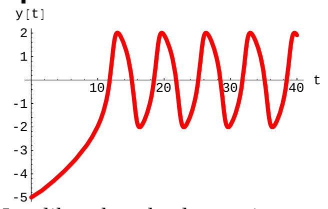
☐G.6.d.iii) For your information

Van der Pol oscillators come up from time to-time in advanced work. You are among the very first students in an introductory course to get hands-on experience with them. Control theorists love them because they can take weird starting data and make the transition to the steady limit cycle so smoothly.
Some medical folks have used van der Pol nonlinear oscillators to model the pumping heart (See Edward Beltrami, Mathematics for dynamic modeling, Academic Press, 1987). Take a look for yourself:

```
ystart = 5;
yprimestart = 2;
Clear[t, y]
vanderpoloscillator = y''[t] + (y[t]^2 - 1) y'[t] + y[t] == 0;
endtime = 30;
Clear[s, y, t]
ndssol =
 NDSolve[{vanderpoloscillator, y[0] == ystart, y'[0] == yprimestart},
  y[t], {t, 0, endtime}, MaxSteps → 1000];
y[t_] = y[t] /. ndssol[[1]];
Plot[y[t], {t, 0, endtime},
 PlotStyle → {{Thickness[0.015], Red}}, AspectRatio → 1/GoldenRatio,
 PlotRange → All, AxesLabel → {"t", "y[t]"}];
```



```
ystart = -5;
yprimestart = -1;
Clear[t, y]
vanderpoloscillator = y''[t] + (y[t]^2 - 1) y'[t] + y[t] == 0;
endtime = 40;
Clear[s, y, t]
ndssol =
 NDSolve[{vanderpoloscillator, y[0] == ystart, y'[0] == yprimestart},
  y[t], {t, 0, endtime}, MaxSteps → 1000];
y[t_] = y[t] /. ndssol[[1]];
Plot[y[t], {t, 0, endtime},
 PlotStyle → {{Thickness[0.015], Red}}, AspectRatio → 1/GoldenRatio,
 PlotRange → All, AxesLabel → {"t", "y[t]"}];
```



Just like when the doctors jump start a heart in the movies!

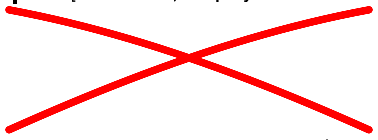
G.7) Trajectories and starting data

Throughout this problem, assume that the systems have no danger zones.

□G.7.a.i)

When you go with a given system
x'[t] = m[x[t], y[t]]
y'[t] = n[x[t], y[t]],
then different starting data on x[0] and y[0] will result in different trajectories {x[t], y[t]}.
The question here is:
Do you think that two trajectories coming from a given system
x'[t] = m[x[t], y[t]]
y'[t] = n[x[t], y[t]]
can ever cross over each other like this:

```
This instruction will not run successfully unless you have run the
                    intializations of this notebook.
Show[crisscross, DisplayFunction → $DisplayFunction];
```



```
Assume no danger zone.
```

Why or why not?

□Tip:

Think of it this way:

At the cross-over point, which way must the floating cork move?

□G.7.a.ii)

You are going with specific functions m[x, y], n[x, y], and the resulting system
x'[t] = m[x[t], y[t]]
y'[t] = n[x[t], y[t]].

First you go with given starter data
{x[0], y[0]} = {a1, b1}
and get solutions
{x1[t], y1[t]}.

Later, you go with starter data
{x[0], y[0]} = {a2, b2}
and get solutions
{x2[t], y2[t]}.

And by some fluke, you learn for a certain time $t^* > 0$ that
{x1[t^*], y1[t^*]} = {x2t^*], y2[t^*]}.

What relationship between the two sets of starter data {a1, b1} and {a2, b2} do you expect?
What relationship between the two solutions {x1[t], y1[t]} and {x2[t], y2[t]} do you expect?
In both questions, assume no danger zone.

□G.7.b.i) Crossing of plots of solutions of first order differential equations

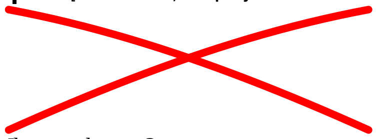
You have a differential equation
y'[t] = f[t, y[t]]
and a given formula for f[t, y].

If you start with
y[0] = 2.3,
then you can use `NDSolve` to plot out a function y1[t] with
y1[0] = 2.3 and y1'[t] = f[t, y1[t]].

On the other hand if you start with
y[0] = 0.8,
then you can use NDSolve to plot out another function y2[t] with
y2[0] = 0.8 and y2'[t] = f[t, y2[t]].

The question here is:
Do you think that that the plots of y1[t] and y2[t] can ever cross over each other like this:

```
Show[crisscross, DisplayFunction → $DisplayFunction];
```



Why or why not?
Assume no danger zone.

□G.7.b.ii)

You are going with a specific function f[t, y] and the resulting first order differential equation
y'[t] = f[t, y[t]].

You go with given starter data
y[0] = a1
and get a solution
y1[t].

Later, you go with starter data
y[0] = a2
and get a solution
y2[t].

And by some fluke, you learn for a certain time $t^* > 0$ that
y1[t^*] = y2[t^*].
What relationship between a1 and a2 do you expect?
What relationship between the two solutions y1[t] and y2[t] do you expect?

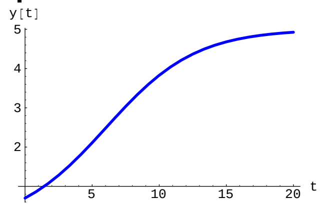
□G.7.c.i)

Here's `NDSolve` cranking out the plot of the solution of
$y'[t] = 0.3\, y[t] \left(1 - \frac{y[t]}{5}\right)$
with y[0] = 0.7:

```
endtime = 20;
Clear[y, t]
ndssol = NDSolve[{y'[t] == 0.3 y[t] (1 - y[t]/5), y[0] == 0.7},
  y[t], {t, 0, endtime}]
{{y[t] → InterpolatingFunction[{{0., 20.}}, <>][t]}}
```

And a plot:

```
y[t_] = y[t] /. ndssol[[1]];
y1plot = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Blue}}, AspectRatio → 1/GoldenRatio,
  PlotRange → All, AxesLabel → {"t", "y[t]"}];
```



Here's an attempt to use **`NDSolve`** to crank out the plot of the solution of

y''[t] + 0.3 y'[t] + 1.5 y[t] = 0

with y[0] = 0.7:

```
endtime = 20;
Clear[s, y, y, t]
ndssol = NDSolve[{y''[t] + 0.3 y[t] + 1.5 y[t] == 0, y[0] == 0.7},
  y[t], {t, 0, endtime}]
```

```
NDSolve::ndnco : The number of constraints (1) (initial conditions) is not equal
  to the total differential order of the system (2).
```

```
NDSolve[{1.8 y[t] + y''[t] == 0, y[0] == 0.7}, y[t], {t, 0, 20}]
```

Explain what the error message is trying to tell you and say what additional data **`NDSolve`** needs before it can run successfully.

□G.7.c.ii)

You have an oscillator differential equation

y''[t] + b y'[t] + c y[t] = 0

If you have specific values of b and c, and you start with

y[0] = 1.6 and y'[0] = 2.4,

then you can use NDSolve to plot out a function y1[t] with

y1''[t] + b y1'[t] + c1 y[t] = 0

and with

y1[0] = 1.6 and y1′[0] = 2.4

On the other hand if you start with

y[0] = 1.5 and y'[0] = −0.5,

then you can use **`NDSolve`** to plot out a different function y2[t] with

y2''[t] + b y2'[t] + c y2[t] = 0

and with

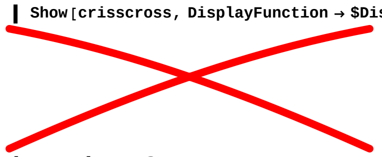
y2[0] = 1.5 and y2'[0] = −0.5.

The question here is:
Can the plots of the two trajectories

{y1'[t], y1[t]} and {y2'[t], y2[t]}

ever cross over each other like this:

```
Show[crisscross, DisplayFunction → $DisplayFunction];
```



Why or why not?

□G.7.c.iii)

You are going with a specific values of b and c and the resulting oscillator differential equation

y'[t] + b y'[t] + c y[t] = 0.

You go with given starter data

y[0] = a1 and y'[0] = b1

and get a solution

y1[t].

Later, you go with starter data

y[0] = a2 and y'[0] = b2

and get a solution

y2[t].

And by some fluke, you learn for a certain time $t^* > 0$ that

$y1[t^*] = y2[t^*]$

and

$y1'[t^*] = y2'[t^*]$.

What relationship between a1 and a2 do you expect?
What relationship between b1 and b2 do you expect?
What relationship between the two solutions y1[t] and y2[t] do you expect?

□G.7.c.iv) Whoops

You have an oscillator differential equation

y''[t] + b y'[t] + c y[t] = 0

If you have specific values of b and c, and you start with

y[0] = 1.6 and y'[0] = 2.4,

then you can use NDSolve to plot out a function y1[t] with

y1''[t] + b y1'[t] + c1 y[t] = 0

and with

y1[0] = 1.6 and y1'[0] = 2.4.

On the other hand if you start with

y[0] = 1.5 and y'[0] = −0.5,

then you can use NDSolve to plot out a different function y2[t] with

y2''[t] + b y2'[t] + c y2[t] = 0

and with

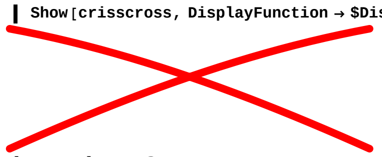
y2[0] = 1.5 and y2'[0] = −0.5.

The question here is:
Can the plots of the solutions

y1[t] and y2[t]

ever cross over each other like this:

```
Show[crisscross, DisplayFunction → $DisplayFunction];
```



Why or why not?
Why does you answer not conflict with what you said in part i)?

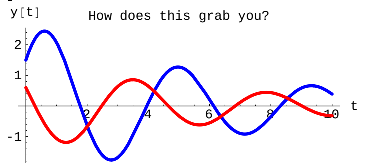
□ Tip:

Careful, you are being set-up.

Before you jump to any conclusion, look at this:

```
b = 0.3;
c = 2.1;
```

```
a1 = 1.5;
b1 = 3.0;
a2 = 0.6;
b2 = -2.0;

endtime = 10;
Clear[s, y, y1, y2, t]
ndssol1 = NDSolve[
 {y''[t] + b y'[t] + c y[t] == 0,
  y[0] == a1, y'[0] == b1}, y[t], {t, 0, endtime}];

y1[t_] = y[t] /. ndssol1[[1]];
y1plot = Plot[y1[t], {t, 0, endtime},
   PlotStyle -> {{Thickness[0.01], Blue}},
   PlotRange -> All,
   DisplayFunction -> Identity];
ndssol2 = NDSolve[
 {y''[t] + b y'[t] + c y[t] == 0,
  y[0] == a2, y'[0] == b2}, y[t], {t, 0, endtime}];

y2[t_] = y[t] /. ndssol2[[1]];
y2plot = Plot[y2[t], {t, 0, endtime},
   PlotStyle -> {{Thickness[0.01], Red}},
   PlotRange -> All,
   DisplayFunction -> Identity];

Show[y1plot, y2plot, AxesLabel -> {"t", "y[t]"},
   DisplayFunction -> $DisplayFunction,
   PlotLabel -> "How does this grab you?"];
```



G.8) Coefficient matrices and linear systems*

Don't worry if you haven't taken a course in matrices.

Linear systems are those that look this way:

x'[t] = a x[t] + b y[t]
y'[t] = c x[t] + d y[t].

Each of a, b, c and d is a constant.

```
Clear[x, y, t, a, b, c, d]
linearsystem =
 ({x'[t], y'[t]} == {a x[t] + b y[t], c x[t]  + d y[t]})
{x′[t], y′[t]} == {a x[t] + b y[t], c x[t] + d y[t]}
```

Here it is in slightly better form:

```
ColumnForm[Thread[linearsystem]]
x′[t] == a x[t] + b y[t]
y′[t] == c x[t] + d y[t]
```

You can read off a matrix A:

Lots of folks call A by the name "coefficient matrix."

```
A = {{a, b}, {c, d}};
MatrixForm[A]
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

You can use multiplication by this coefficient matrix to write the given linear system in this compact way:

```
Clear[x, y, t]
matrixlinearsystem = ({x'[t], y'[t]} == A.{x[t], y[t]});
ColumnForm[Thread[matrixlinearsystem]]
x′[t] == a x[t] + b y[t]
y′[t] == c x[t] + d y[t]
```

And bringing in the coefficient matrix A makes it really easy to write down the flow field for this linear system:

```
Clear[Field]
Field[x_, y_] = A.{x, y}
{a x + b y, c x + d y}
```

□G.8.a.i) Coming up with the matrices

Here is a linear system:

```
Clear[x, y, t]
linearsystem = {x′[t], y′[t]} == {2.1 x[t] - 3.9 y[t], 0.3 x[t] + 2.4 y[t]}
{x′[t], y′[t]} == {2.1 x[t] - 3.9 y[t], 0.3 x[t] + 2.4 y[t]}
```

Here it is in slightly better form:

```
ColumnForm[Thread[linearsystem]]
x′[t] == 2.1 x[t] - 3.9 y[t]
y′[t] == 0.3 x[t] + 2.4 y[t]
```

Read off, as above, the coefficient matrix A and fill in the correct values of a, b ,c ,d in the code below:

```
a =  ;
b =  ;
c =  ;
d = ;
A = {{a, b}, {c, d}};
MatrixForm[A]
```

Check youself:

```
Clear[x, y, t]
matrixlinearsystem = ({x'[t], y'[t]} == A.{x[t], y[t]});
ColumnForm[Thread[matrixlinearsystem]]
```

□G.8.a.ii) Coming up with the matrices

The given linear system is:

```
Clear[x, y, t]
linearsystem = {y′[t], x′[t]} == {0.3 x[t] + 2.4 y[t], 2.1 x[t] - 3.9 y[t]}
{y′[t], x′[t]} == {0.3 x[t] + 2.4 y[t], 2.1 x[t] - 3.9 y[t]}
```

Here it is in slightly better form:

```
ColumnForm[Thread[linearsystem]]
y′[t] == 0.3 x[t] + 2.4 y[t]
x′[t] == 2.1 x[t] - 3.9 y[t]
```

Read off, as above, the coefficient matrix A and fill the correct values of a, b ,c and d into the code below:

```
a =  ;
b =  ;
c =  ;
d = ;
A = {{a, b}, {c, d}};
MatrixForm[A]
```

□G.8.b.i) Linear flows and trajectories

Here's a linear system:

```
Clear[x, y, t]
linearsystem =
 ({x'[t], y'[t]} == {0.6 x[t] + 0.5 y[t], 1.9 x[t] - 1.2 y[t]});
ColumnForm[Thread[linearsystem]]
x′[t] == 0.6 x[t] + 0.5 y[t]
y′[t] == 1.9 x[t] - 1.2 y[t]
```

The coefficient matrix is:

```
A = {{0.6, 0.5}, {1.9, -1.2}};
MatrixForm[A]
```

$$\begin{pmatrix} 0.6 & 0.5 \\ 1.9 & -1.2 \end{pmatrix}$$

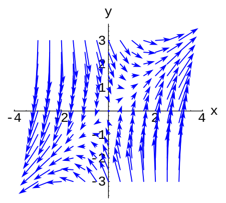
Check:

```
Clear[x, y, t]
matrixlinearsystem = ({x'[t], y'[t]} == A.{x[t], y[t]});
ColumnForm[Thread[matrixlinearsystem]]
x′[t] == 0.6 x[t] + 0.5 y[t]
y′[t] == 1.9 x[t] - 1.2 y[t]
```

This checks.

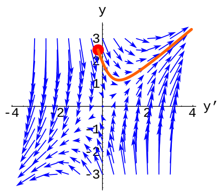
Now enter the field and plot the flow:

```
Clear[Field]
Field[x_, y_] = A.{x, y}
flowplot = Table[Arrow[Field[x, y], Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.25, HeadSize → 0.35],
   {x, -3, 3, 6/12}, {y, -3, 3, 6/12}];
Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
{0.6 x + 0.5 y, 1.9 x - 1.2 y}
```



Throw in a trajectory:

```
starterpoint = {-0.2, 2.5};
endtime = 2.5;
Clear[m, n]
{m[x_, y_], n[x_, y_]} = Field[x, y];
{a, b} = starterpoint;
Clear[x, y, t]
equationx = (x'[t] == m[x[t], y[t]]);
equationy = (y'[t] == n[x[t], y[t]]);
starterx = (x[0] == a);
startery = (y[0] == b);

ndssol = NDSolve[{equationx, equationy,
  starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};

trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
   PlotStyle -> {{CadmiumOrange, Thickness[0.015]}},
        DisplayFunction -> Identity];

starterplot = Graphics[{Red, PointSize[0.06],
          Point[starterpoint]}];

Show[flowplot, starterplot,
  trajectoryplot,
 PlotRange -> All, Axes -> True, AxesLabel -> {"y'", "y"},
    DisplayFunction -> $DisplayFunction];
```



Play with other starting points and endtimes and then write your own description of the trajectories in this flow.
Would it be fair to say that {0, 0} is an "attractor?"

□G.8.b.ii)

Here's a new linear system:

```
Clear[x, y, t]
linearsystem =
 ({x'[t], y'[t]} == {-1.3 x[t] + 0.5 y[t], 0.3 x[t] - 1.2 y[t]});
ColumnForm[Thread[linearsystem]]
```

```
x′[t] == -1.3 x[t] + 0.5 y[t]
y′[t] == 0.3 x[t] - 1.2 y[t]
```

The coefficient matrix is:

```
A = {{-1.3, 0.5}, {0.3, -1.2}};
MatrixForm[A]
```

$$\begin{pmatrix} -1.3 & 0.5 \\ 0.3 & -1.2 \end{pmatrix}$$

Check:

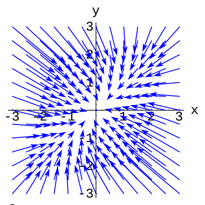
```
Clear[x, y, t]
matrixlinearsystem = ({x'[t], y'[t]} == A . {x[t], y[t]});
ColumnForm[Thread[matrixlinearsystem]]
```

```
x′[t] == -1.3 x[t] + 0.5 y[t]
y′[t] == 0.3 x[t] - 1.2 y[t]
```

This checks.
Now enter the field and plot the flow:

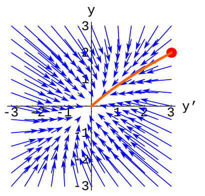
```
Clear[Field]
Field[x_, y_] = A . {x, y}
flowplot = Table[Arrow[Field[x, y], Tail → {x, y}, VectorColor → Blue,
    ScaleFactor → 0.3, HeadSize → 0.3], {x, -3, 3, 6/12}, {y, -3, 3, 6/12}];
Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

```
{-1.3 x + 0.5 y, 0.3 x - 1.2 y}
```



Throw in a trajectory:

```
starterpoint = {3, 2.0};
endtime = 10;
Clear[m, n]
{m[x_, y_], n[x_, y_]} = Field[x, y];
{a, b} = starterpoint;
Clear[x, y, t]
equationx = (x'[t] == m[x[t], y[t]]);
equationy = (y'[t] == n[x[t], y[t]]);
starterx = (x[0] == a);
startery = (y[0] == b);

ndssol = NDSolve[{equationx, equationy,
  starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};

trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
   PlotStyle -> {{CadmiumOrange, Thickness[0.015]}},
        DisplayFunction -> Identity];

starterplot = Graphics[{Red, PointSize[0.06],
          Point[starterpoint]}];

Show[flowplot, starterplot,
  trajectoryplot,
 PlotRange -> All, Axes -> True, AxesLabel -> {"y'", "y"},
    DisplayFunction -> $DisplayFunction];
```



If you increase the endtime, does the trajectory move much more?
Play with other starting points and endtimes and then write your own description of the trajectories in this flow.
Would it be fair to say that {0, 0} is an "attractor?"

□G.8.b.iii)

Here's a new linear system:

```
Clear[x, y, t]
linearsystem =
 ({x'[t], y'[t]} == {-1.2 x[t] + 1.5 y[t], -2.3 x[t] + 1.2 y[t]});
ColumnForm[Thread[linearsystem]]
```

```
x′[t] == -1.2 x[t] + 1.5 y[t]
y′[t] == -2.3 x[t] + 1.2 y[t]
```

The coefficient matrix is:

```
A = {{-1.2, 1.5}, {-2.3, 1.2}};
MatrixForm[A]
```

$$\begin{pmatrix} -1.2 & 1.5 \\ -2.3 & 1.2 \end{pmatrix}$$

```
Clear[x, y, t]
matrixlinearsystem = ({x'[t], y'[t]} == A . {x[t], y[t]});
ColumnForm[Thread[matrixlinearsystem]]
```

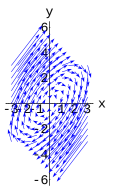
```
x′[t] == -1.2 x[t] + 1.5 y[t]
y′[t] == -2.3 x[t] + 1.2 y[t]
```

```
Clear[x, y, t]
matrixlinearsystem = ({x'[t], y'[t]} == A . {x[t], y[t]});
ColumnForm[Thread[matrixlinearsystem]]
```

```
x′[t] == -1.2 x[t] + 1.5 y[t]
y′[t] == -2.3 x[t] + 1.2 y[t]
```

This checks.
Enter the field and plot the flow:

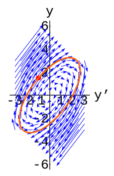
```
Clear[Field]
Field[x_, y_] = A . {x, y};
flowplot = Table[Arrow[Field[x, y], Tail → {x, y}, VectorColor → Blue,
    ScaleFactor → 0.3, HeadSize → 0.4], {x, -3, 3, 6/12}, {y, -3, 3, 6/12}];
Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



Throw in a trajectory:

```
starterpoint = {-1, 1.5};
endtime = 10;
Clear[m, n]
{m[x_, y_], n[x_, y_]} = Field[x, y];
{a, b} = starterpoint;
Clear[x, y, t]
equationx = (x'[t] == m[x[t], y[t]]);
equationy = (y'[t] == n[x[t], y[t]]);
starterx = (x[0] == a);
startery = (y[0] == b);

ndssol = NDSolve[{equationx, equationy,
  starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory]
```

```
trajectory[t_] = {x[t] /. ndssol[[1]],
        y[t] /. ndssol[[1]]};

trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
   PlotStyle -> {{CadmiumOrange, Thickness[0.015]}},
        DisplayFunction -> Identity];

starterplot = Graphics[{Red, PointSize[0.06],
          Point[starterpoint]}];

Show[flowplot, starterplot,
  trajectoryplot,
 PlotRange -> All, Axes -> True, AxesLabel -> {"y'", "y"},
   DisplayFunction -> $DisplayFunction];
```



If you increase the endtime, what happens to the trajectory?
Play with other starting points and endtimes and then write your own description of the trajectories in this flow.
Would it be fair to say that {0, 0} is an "attractor?"

□G.8.b.iv)

Here's a new linear system:

```
Clear[x, y, t]
linearsystem =
 ({x'[t], y'[t]} == {-1.0 x[t] + 1.5 y[t], -2.3 x[t] + 1.2 y[t]});
ColumnForm[Thread[linearsystem]]
```

```
x′[t] == -1. x[t] + 1.5 y[t]
y′[t] == -2.3 x[t] + 1.2 y[t]
```

The coefficient matrix is:

```
A = {{-1.0, 1.5}, {-2.3, 1.2}};
MatrixForm[A]
```

$$\begin{pmatrix} -1. & 1.5 \\ -2.3 & 1.2 \end{pmatrix}$$

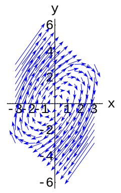
```
Clear[x, y, t]
matrixlinearsystem = ({x'[t], y'[t]} == A.{x[t], y[t]});
ColumnForm[Thread[matrixlinearsystem]]
```

```
x′[t] == -1. x[t] + 1.5 y[t]
y′[t] == -2.3 x[t] + 1.2 y[t]
```

This checks.
Enter the field and plot the flow:

```
Clear[Field]
Field[x_, y_] = A.{x, y}
flowplot = Table[Arrow[Field[x, y], Tail → {x, y}, VectorColor → Blue,
   ScaleFactor → 0.3, HeadSize → 0.4], {x, -3, 3, 6/12}, {y, -3, 3, 6/12}];
Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

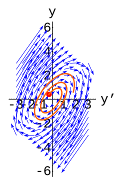
```
{-1. x + 1.5 y, -2.3 x + 1.2 y}
```



Throw in a trajectory:

```
starterpoint = {-0.3, 0.4};
endtime = 10;
Clear[m, n]
{m[x_, y_], n[x_, y_]} = Field[x, y];
{a, b} = starterpoint;
Clear[x, y, t]
equationx = x′[t] == m[x[t], y[t]];
equationy = y′[t] == n[x[t], y[t]];
starterx = x[0] == a;
startery = y[0] == b;
ndssol = NDSolve[{equationx, equationy, starterx, startery},
  {x[t], y[t]}, {t, 0, endtime}];
Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol⟦1⟧, y[t] /. ndssol⟦1⟧};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, 0, endtime}, PlotStyle →
   {{CadmiumOrange, Thickness[0.015]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];
```

```
Show[
 flowplot, starterplot, trajectoryplot, PlotRange → All, Axes → True,
 AxesLabel → {"y'", "y"}, DisplayFunction → $DisplayFunction];
```



If you increase the endtime, what happens to the trajectory?
Play with other starting points and endtimes and then write your own description of the trajectories in this flow.
Would it be fair to say that {0, 0} is an "attractor?"

□G.8.b.v)

Here's a new linear system:

```
Clear[x, y, t]
linearsystem =
 ({x'[t], y'[t]} == {0.2 x[t] + 0.8 y[t], -0.6 x[t] - 0.5 y[t]});
ColumnForm[Thread[linearsystem]]
```

```
x′[t] == 0.2 x[t] + 0.8 y[t]
y′[t] == -0.6 x[t] - 0.5 y[t]
```

The matrix is:

```
A = {{0.2, 0.8}, {-0.6, -0.5}};
MatrixForm[A]
```

$$\begin{pmatrix} 0.2 & 0.8 \\ -0.6 & -0.5 \end{pmatrix}$$

```
Clear[x, y, t]
matrixlinearsystem = ({x'[t], y'[t]} == A.{x[t], y[t]});
ColumnForm[Thread[matrixlinearsystem]]
```

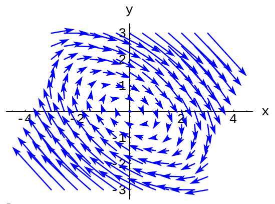
```
x′[t] == 0.2 x[t] + 0.8 y[t]
y′[t] == -0.6 x[t] - 0.5 y[t]
```

This checks.
Enter the field and plot the flow:

```
Clear[Field]
Field[x_, y_] = A.{x, y}
```

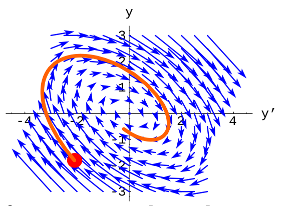
```
flowplot = Table[Arrow[Field[x, y], Tail → {x, y}, ScaleFactor → 0.5,
   HeadSize → 0.4], {x, -3, 3, 6/12}, {y, -3, 3, 6/12}];
Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

```
{0.2 x + 0.8 y, -0.6 x - 0.5 y}
```



Throw in a trajectory:

```
starterpoint = {-2.1, -1.8};
endtime = 10;
Clear[m, n]
{m[x_, y_], n[x_, y_]} = Field[x, y];
{a, b} = starterpoint;
Clear[x, y, t]
equationx = (x'[t] == m[x[t], y[t]]);
equationy = (y'[t] == n[x[t], y[t]]);
starterx = (x[0] == a);
startery = (y[0] == b);

ndssol = NDSolve[{equationx, equationy,
  starterx, startery}, {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory]
trajectory[t_] = {x[t] /. ndssol[[1]],
       y[t] /. ndssol[[1]]};

trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
   PlotStyle -> {{CadmiumOrange, Thickness[0.015]}},
       DisplayFunction -> Identity];

starterplot = Graphics[{Red, PointSize[0.06],
         Point[starterpoint]}];

Show[flowplot, starterplot,
  trajectoryplot,
 PlotRange -> All, Axes -> True, AxesLabel -> {"y'", "y"},
   DisplayFunction -> $DisplayFunction];
```



If you increase the endtime, what happens to the trajectory?
Play with other starting points and endtimes and then write your own description of the trajectories in this flow.
Would it be fair to say that {0, 0} is an attractor?

G.9) Eigenvectors of coefficient matrices of linear systems can tell you a lot*

□G.9.a.i)

Here's a linear system of differential equations:

```
Clear[x, y, t]
linearsystem = {x′[t], y′[t]} == {1.7 x[t] + 0.8 y[t], 1.5 x[t] - 0.7 y[t]}
```

```
{x′[t], y′[t]} == {1.7 x[t] + 0.8 y[t], 1.5 x[t] - 0.7 y[t]}
```

Here it is in slightly better form:

```
ColumnForm[Thread[linearsystem]]
```

```
x′[t] == 1.7 x[t] + 0.8 y[t]
y′[t] == 1.5 x[t] - 0.7 y[t]
```

You can read off the coefficient matrix A:

```
A = {{1.7, 0.8}, {1.5, -0.7}};
MatrixForm[A]
```

$$\begin{pmatrix} 1.7 & 0.8 \\ 1.5 & -0.7 \end{pmatrix}$$

Now you can use this matrix to write the given linear system in this compact way:

```
Clear[x, y, t]
matrixlinearsystem = ({x'[t], y'[t]} == A . {x[t], y[t]});
ColumnForm[Thread[matrixlinearsystem]]
```

```
x′[t] == 1.7 x[t] + 0.8 y[t]
y′[t] == 1.5 x[t] - 0.7 y[t]
```

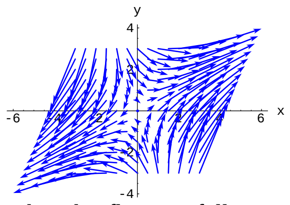
Introducing the matrix A makes it really easy to write down the flow field for this linear system:

```
Clear[Field]
Field[x_, y_] = A . {x, y}
```

```
{1.7 x + 0.8 y, 1.5 x - 0.7 y}
```

Look at the scaled flow:

```
scalefactor = 0.35;
flowplot = Table[Arrow[Field[x, y], Tail → {x, y}, VectorColor → Blue,
    ScaleFactor → 0.4, HeadSize → 0.4], {x, -3, 3, 0.5}, {y, -3, 3, 0.5}];
Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```



Look at this flow carefully.
After you've got a good feel for this flow, let *Mathematica* calculate two special vectors related to the coefficient matrix A:

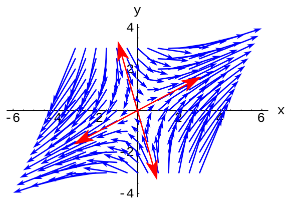
```
Clear[eigenvector];
{eigenvector[1], eigenvector[2]} = Eigenvectors[A]
```

```
{{0.883204, 0.46899}, {-0.272488, 0.962159}}
```

Just as you don't have to know how a fine meal was cooked in order to enjoy it,
you don't have to know how these special vectors were calculated in order to enjoy them.

Here comes a plot of scaled versions of these special vectors and their negatives shown with the flow plot:

```
scalefactor = 3.5;
headsize = 1;
Show[flowplot, Arrow[eigenvector[1], Tail → {0, 0},
   VectorColor → Red, ScaleFactor → scalefactor, HeadSize → headsize],
  Arrow[-eigenvector[1], Tail → {0, 0}, VectorColor → Red,
   ScaleFactor → scalefactor, HeadSize → headsize],
  Arrow[eigenvector[2], Tail → {0, 0}, VectorColor → Red,
   ScaleFactor → scalefactor, HeadSize → headsize],
  Arrow[-eigenvector[2], Tail → {0, 0}, VectorColor → Red,
   ScaleFactor → scalefactor, HeadSize → headsize], Axes → True,
  AxesLabel → {"x", "y"}];
```



The pros call these special vectors by the name: eigenvectors.

Cool plot.
Let the visual impression soak in and then explain how these eigenvectors are related to the flow.

□G.9.a.ii)

Try it again with this new linear system:

$$x'[t] = -0.7\,x[t] - 0.9\,y[t]$$
$$y'[t] = -0.6\,x[t] - 0.7\,y[t].$$

The coefficient matrix is:

```
A = {{-0.7, -0.9}, {-0.6, -0.7}};
MatrixForm[A]
```

$$\begin{pmatrix} -0.7 & -0.9 \\ -0.6 & -0.7 \end{pmatrix}$$

Compare with the given linear system

$$x'[t] = -0.7\,x[t] - 1.3\,y[t]$$
$$y'[t] = 0.5\,y[t].$$

```
Clear[x, y, t]
linearsystem = ({x'[t], y'[t]} == A . {x[t], y[t]});
ColumnForm[Thread[linearsystem]]
```

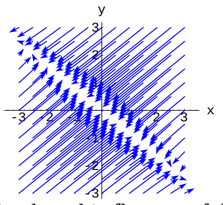
```
x′[t] == -0.7 x[t] - 0.9 y[t]
y′[t] == -0.6 x[t] - 0.7 y[t]
```

That's the right matrix.
Look at the flow:

```
Clear[Field]
Field[x_, y_] = A . {x, y}
```

```
flowplot = Table[Arrow[Field[x, y], Tail → {x, y}, VectorColor → Blue,
    ScaleFactor → 0.6, HeadSize → 0.3], {x, -3, 3, 6/12}, {y, -3, 3, 6/12}];
Show[flowplot, Axes → True, AxesLabel → {"x", "y"}];
```

```
{-0.7 x - 0.9 y, -0.6 x - 0.7 y}
```



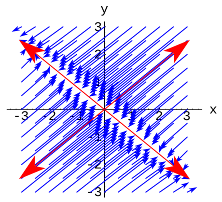
Look at this flow carefully.
After you've got a good feel for this flow, let *Mathematica* calculate the eigenvectors of the matrix A:

```
Clear[eigenvector];
{eigenvector[1], eigenvector[2]} = Eigenvectors[A]
```

```
{{0.774597, 0.632456}, {0.774597, -0.632456}}
```

Plot scaled versions of these special vectors and their negatives together with the field:

```
scalefactor = 4;
headsize = 1;

Show[flowplot, Arrow[eigenvector[1], Tail → {0, 0},
   VectorColor → Red, ScaleFactor → scalefactor, HeadSize → headsize],
  Arrow[-eigenvector[1], Tail → {0, 0}, VectorColor → Red,
   ScaleFactor → scalefactor, HeadSize → headsize],
  Arrow[eigenvector[2], Tail → {0, 0}, VectorColor → Red,
   ScaleFactor → scalefactor, HeadSize → headsize],
  Arrow[-eigenvector[2], Tail → {0, 0}, VectorColor → Red,
   ScaleFactor → scalefactor, HeadSize → headsize], Axes → True,
  AxesLabel → {"x", "y"}];
```



Another cool plot.
Explain how these eigenvectors help you to describe the flow.

G.10) Boundary value problems: Shooting for a specified outcome

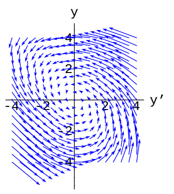
□G.10.a) Shooting a system trajectory to be in a specified place at a specified time

This problem starts with a little system:

```
Clear[m, n, x, y, t, starterx, startery]
m[x_, y_] = -0.8 x - y;
n[x_, y_] = x E^-0.1 y;
(diffeqsystem = {x'[t] == m[x[t], y[t]], y'[t] == n[x[t], y[t]],
x[0] == starterx, y[0] == startery}) // ColumnForm
x′[t] == -0.8 x[t] - y[t]
y′[t] == E^-0.1 y[t] x[t]
x[0] == starterx
y[0] == startery
```

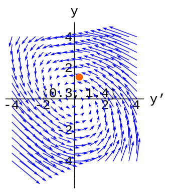
Check out the flow plot for this linear system:

```
Clear[Field]
Field[x_, y_] = {m[x, y], n[x, y]};
scalefactor = 0.3;
{xlow, xhigh} = {-4, 4};
{ylow, yhigh} = {-4, 4};
jump = 0.5;
flowplot = Table[Arrow[{m[x, y], n[x, y]}, Tail → {x, y},
    VectorColor → Blue, ScaleFactor → 0.25, HeadSize → 0.3],
   {x, xlow, xhigh, jump}, {y, ylow, yhigh, jump}];
Show[flowplot, PlotRange → All, Axes → True, AxesLabel → {"y'", "y"},
  DisplayFunction → $DisplayFunction];
```



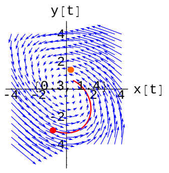
Here is a plotted point:

```
target = {0.3, 1.4};
bullseye = Graphics[{PointSize[0.05], CadmiumOrange, Point[target]}];
targetlabel = Graphics[Text[target, target, {0, 2}]];
Show[flowplot, bullseye, targetlabel, Axes → True,
  AxesLabel → {"y'", "y"}, DisplayFunction → $DisplayFunction];
```



Your job is to come up with starter values on x[0] and y[0] so that the corresponding trajectory comes as close as you can make it to the plotted point at time t = 3.
Here's a first try:

```
{starterx, startery} = {-1, -3};
endtime = 3;
Clear[thistry];
thistry[t_] = {x[t], y[t]} /.
   NDSolve[diffeqsystem, {x[t], y[t]}, {t, 0, endtime}][[1]];
trajplots = ParametricPlot[thistry[t],
   {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Red}},
   AxesLabel → {"x[t]", "y[t]"}, DisplayFunction → Identity];
starterplot =
  Graphics[{PointSize[0.05], Red, Point[{starterx, startery}]}];
Show[trajplots, flowplot, starterplot, bullseye, targetlabel,
  Axes → True, PlotRange → All, DisplayFunction → $DisplayFunction];
```



Close, but no cigar. See how well you can do.

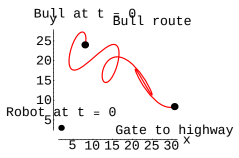
The trial and error shooter method you used above may seem dinky and simple-minded.
It is.
But many professional experts prefer this method over any other.

G.11) Pursuits by a robotic cowhand

DiffEq&*Mathematica* thanks cattleman Thomas O.Smith of Homer, Illinois for some help with this problem.

The scene is the DE&M Electronic Ranch, where all the cattle are prize winners and where electronic robots do all the work.
One night, the prize bull breaks out of the pen, meanders around the ranch grounds, and then heads for the gate to the highway on the path plotted below:

```
Clear[bullx, bullyt]
{bullx[t_], bully[t_]} = 4 {1 + 0.1 t + Cos[0.3 t], 6 - 0.05 t + Sin[0.4 t]};
bullroute = ParametricPlot[{bullx[t], bully[t]}, {t, 0, 60},
   PlotStyle → {{Thickness[0.01], Red}}, AxesLabel → {"x", "y"},
   DisplayFunction → Identity, Epilog → Text["Bull route", {25, 30}]];
pointplot =
  {Graphics[{PointSize[0.06], Point[{bullx[0], bully[0]}]}],
   Graphics[{PointSize[0.06], Point[{bullx[60], bully[60]}]}],
   Graphics[{PointSize[0.05], Point[{2, 3}]}]};
labels =
  {Graphics[Text["Bull at t = 0", {bullx[0], bully[0]}, {0, -4}]],
   Graphics[Text["Gate to highway", {bullx[60], bully[60]}, {0, 3}]],
   Graphics[Text["Robot at t = 0", {2, 3}, {0, -2}]]};
setup = Show[bullroute, pointplot, labels,
   PlotRange → All, AxesOrigin → {0, 0}, AspectRatio → Automatic,
   DisplayFunction → $DisplayFunction];
```



Here x and y are measured in yards and t is measured in seconds

At time t = 0, the robot is at position {2, 3} on the plot above.
The robot is programmed to move so that if the robot's position at time t is {robotx[t], roboty[t]}, then the robot's velocity vector at that time is

{robotx'[t], roboty'[t]} = r ({bullx[t], bully[t]} − {robotx[t], roboty[t]})
where r is a positive number yet to be determined.
This is good because:
→ The robot is always moving toward the bull.
→ The robot slows down when it gets near the bull so that the robot neither smashes into the bull nor stampedes the bull.

□G.11.a.i)

Given that the robot can lasso the bull anytime the robot gets within 4.5 yards of the bull, your job is to come up with a specific positive number r, as small as practical, so that the robot will be successful in lassoing the bull before the bull gets to the gate to the highway.

□Tip:

One way to go about this problem is by trial and error with different r's.

The distance between the bull and the robot at time t is

dist[t] =
Sqrt[{bullx[t], bully[t]} − {x[t], y[t]}.({bullx[t], bully[t]} − {x[t], y[t]

The period represents a dot product.

To check out a guess for a good r, you might want to plot dist[t].

□G.11.a.ii)

After you have settled on a good number r in part i), make a movie
and then make a plot of the actual lassoing of the bull by the robot.

□Tip:

You can use

```
lassorad = Graphics[Circle[{a, b}, 4.5]]
```

to plot a circle of radius 4.5 centered on {a, b}.