2D and 3D Measurements

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

VC.04 2D Vector Fields and Their Trajectories
Basics

B.1) Vector fields and their trajectories

□B.1.a) Vector fields

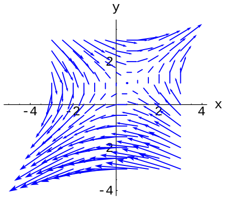
A vector field is a function that spits out vectors. You make a 2D vector field by taking two regular functions, m[x, y] and n[x, y], and throwing them into the two slots:

```
Clear[Field, m, n, x, y];
m[x_, y_] = 0.5 (y - 1);
n[x_, y_] = 0.3 (x - 0.5);
Field[x_, y_] = {m[x, y], n[x, y]}
```

```
{0.5 (-1 + y), 0.3 (-0.5 + x)}
```

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

```
vectorfieldplot = Table[
  Arrow[Field[x, y], Tail → {x, y}], {x, -3, 3, 0.5}, {y, -3, 3, 0.5}];
Show[vectorfieldplot, Axes → True, AxesLabel → {"x", "y"}];
```



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

□B.1.a.i)

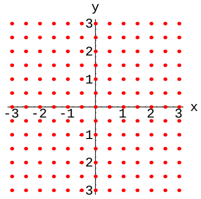
What's a good way of interpreting this plot?

□ **Answer:**

Look at what went into the plot.

The plot shows field vectors Field[x, y] plotted with their tails at {x, y} for a selection of points {x, y}. The selection of points was:

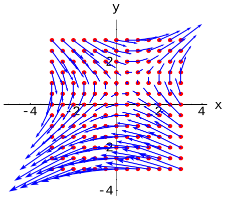
```
pointplot = Table[Graphics[{Red, PointSize[0.02], Point[{x, y}]}],
  {x, -3, 3, 0.5}, {y, -3, 3, 0.5}];
Show[pointplot, Axes → True, AxesLabel → {"x", "y"}];
```



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

Take a look:

```
Show[pointplot, vectorfieldplot, 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, some of the fluid is flowing off to the lower left and some of it is flowing off to the upper right.

The upshot:

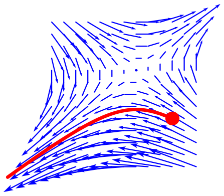
The field vector Field[x, y] with tail at {x, y} represents the speed and the direction with which a cork at {x, y} moves away from {x, y} as it is caught by the flow.

□B.1.a.ii) Trajectories

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

Continue to go with the same vector field, Field[x, y], as used in part i) above and look at this plot:

```
{a, b} = {2, -1};
starterpoint = {a, b};
Clear[Derivative, 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 = 6;

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

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

Show[vectorfieldplot, starterplot, trajectoryplot, PlotRange → All,
 DisplayFunction → $DisplayFunction];
```

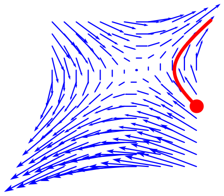


Some folks like to call the path you see a "trajectory" of the vector field. Other folks call this path a "streamline" of the vector field. What's going on here?

□ **Answer:**

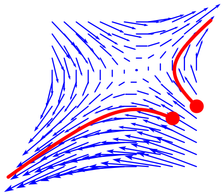
Run it again with a different starting point.

```
{a, b} = {3, -0.5};
starterpoint = {a, b};
Clear[Derivative, 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 = 6;

approxsolutions = NDSolve[{equationx, equationy, starterx, startery},
  {x[t], y[t]}, {t, 0, endtime}];
Clear[newtrajectory]
newtrajectory[t_] =
 {x[t] /. approxsolutions⟦1⟧, y[t] /. approxsolutions⟦1⟧};
newtrajectoryplot = ParametricPlot[newtrajectory[t], {t, 0, endtime},
  PlotStyle → {{Red, Thickness[0.015]}}, DisplayFunction → Identity];
newstarterplot =
 Graphics[{Red, PointSize[0.06], Point[starterpoint]}];
Show[vectorfieldplot, newstarterplot, newtrajectoryplot,
 PlotRange → All, DisplayFunction → $DisplayFunction];
```



See both plots together:

```
Show[vectorfieldplot, starterplot, newstarterplot,
 trajectoryplot, newtrajectoryplot, PlotRange → All,
 DisplayFunction → $DisplayFunction];
```



Now get down to brass tacks.

These plots depict the path of a cork dropped in the flow defined by

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

at the starter point.

Reason: The specifications in the differential equation:

starterpoint = {a, b},

equationx = (x'[t] = m[x[t], y[t]]),

equationy = (y'[t] = n[x[t], y[t]]),

starterx = (x[0] = a), and

startery = (y[0] = b)

tell you that the path will start at {a, b}, and that at each point {x[t], y[t]} on the path, the field vector

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

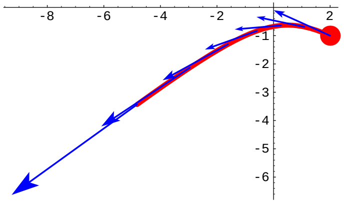
with tail at {x[t], y[t]} is tangent to the path. In fact, at each point {x[t], y[t]} on the path, the field vector

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

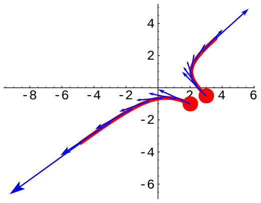
is the velocity vector of the cork as it floats on its merry way.

Take a look at this plot which makes the Field vectors twice as long as they really are:

```
scalefactor = 2;
velvectors = Table[Arrow[Field@@trajectory[t], Tail → trajectory[t],
   ScaleFactor → scalefactor], {t, 0, endtime, 1}];

Show[trajectoryplot, starterplot, velvectors,
 DisplayFunction → $DisplayFunction];
```



```
newvelvectors =
 Table[Arrow[Field@@newtrajectory[t], Tail → newtrajectory[t],
   ScaleFactor → scalefactor], {t, 0, endtime, 1}];

Show[trajectoryplot, starterplot, velvectors, newtrajectoryplot,
 newstarterplot, newvelvectors, DisplayFunction → $DisplayFunction];
```



To really get a lot out of this, go back to the beginning and type in a new vector field and redo everything above until you too go with the flow.

□B.1.a.iii)

How does the vector field

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

govern the actual path 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]
{0.5 (-1 + y), 0.3 (-0.5 + x)}
```

To go with the flow, the cork must leave {x, y} in the direction of the vector Field[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] with its tail at {x, y} is tangent to the actual path of the cork.

B.2) Flow of vector fields along curves; flow of vector fields across curves: Visual inspection

□B.2.a) Flow of a vector field ACROSS a curve

Describe the net flow across the ellipse parameterized by

P[t] = {6 Cos[t], 4 Sin[t]}

with $0 \leq t \leq 2\pi$ of a fluid whose velocity is given by the vector field

Field[x, y] = {x − 1, y}.

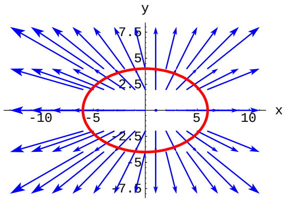
□Answer:

Enter the vector field:

```
Clear[Field, m, n, x, y]
m[x_, y_] = x - 1;
n[x_, y_] = y;
Field[x_, y_] = {m[x, y], n[x, y]}
{-1 + x, y}
```

Here's a look at the curve and the vector field:

```
vectorfieldplot =
 Table[Arrow[Field[x, y], Tail → {x, y}], {x, -6, 6, 1}, {y, -4, 4, 2}];
Clear[t]
x[t_] = 6 Cos[t];
y[t_] = 4 Sin[t];
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {{Red, Thickness[0.01]}}, DisplayFunction → Identity];

Show[vectorfieldplot, curveplot, Axes → True, AspectRatio → Automatic,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



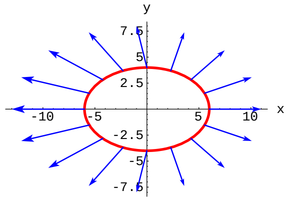
As you can see, this curve isn't a trajectory of the given field.

Your goal here is to analyze what the flow is doing at the points right on this curve.

To do this, look at the field vectors at points right on the curve and ignore what's happening elsewhere:

```
jump = π/8;
fieldvectors = Table[Arrow[Field[x[t], y[t]],
   Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];
```

```
Show[
 curveplot, fieldvectors, AxesOrigin → {0, 0}, AspectRatio → Automatic,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



These vectors indicate the direction and the speed of the flow at points of the curve at the tails of the field vectors.

No doubt about it.

The net flow of this vector field across this curve is from inside to outside.

□B.2.b)

Describe the net flow across the ellipse parameterized by
P[t] = {6 Cos[t], 4 Sin[t]}
with $0 \le t \le 2\pi$ of a fluid whose velocity is given by the vector field
Field[x, y] = {y − 1, y + x}.

□Answer:

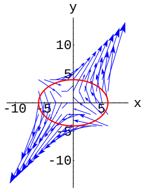
Enter the vector field:

```
Clear[Field, m, n, x, y]
m[x_, y_] = y - 1;
n[x_, y_] = y + x;
Field[x_, y_] = {m[x, y], n[x, y]}
```

```
{-1 + y, x + y}
```

Here's a look at the curve and the vector field:

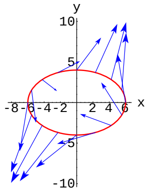
```
vectorfieldplot =
 Table[Arrow[Field[x, y], Tail → {x, y}], {x, -6, 6, 1}, {y, -4, 4, 2}];

Clear[t]
x[t_] = 6 Cos[t];
y[t_] = 4 Sin[t];
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
PlotStyle → {{Red, Thickness[0.01]}}, DisplayFunction → Identity];

Show[vectorfieldplot, curveplot,
 Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"},
 DisplayFunction → $DisplayFunction];
```



The curve and the vector field are two independent creatures. One has no influence on the other. Your goal here is to analyze what the flow is doing at the points right on this curve.

To do this, look at the field vectors at points right on the curve, and ignore what's happening elsewhere:

```
jump = π/8;
fieldvectors = Table[Arrow[Field[x[t], y[t]],
   Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

Show[
 curveplot, fieldvectors, AxesOrigin → {0, 0}, AspectRatio → Automatic,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



These vectors indicate the direction and the speed of the flow at points of the curve at the tails of the field vectors.

No doubt about it.

The net flow of this vector field across this curve is from inside to outside.

□B.2.c)

Describe the net flow across and along the ellipse parameterized by
P[t] = {5 Cos[t], 3 Sin[t]}
with $0 \le t \le 2\pi$ of a fluid whose velocity is given by the vector field
Field[x, y] = {x − y, $\frac{y}{2}$}.

□Answer:

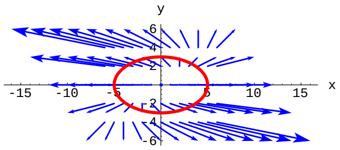
Enter the vector field:

```
Clear[Field, m, n, x, y]
m[x_, y_] = x - y;
n[x_, y_] = y/2;
Field[x_, y_] = {m[x, y], n[x, y]}
```

```
{x - y, y/2}
```

Here's a look the curve and the vector field:

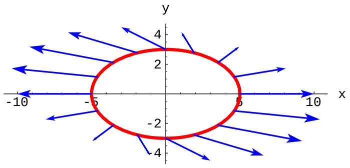
```
vectorfieldplot = Table[Arrow[Field[x, y], Tail → {x, y}],
                        {x, -6, 6, 1}, {y, -4, 4, 2}];

Clear[t]
x[t_] = 5 Cos[t];
y[t_] = 3 Sin[t];
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
 PlotStyle → {{Red, Thickness[0.01]}}, DisplayFunction → Identity];

Show[vectorfieldplot, curveplot,
 Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"},
 DisplayFunction → $DisplayFunction];
```



Your goal here is to analyze what the flow is doing at the points right on this curve.

To do this, look at the field vectors at points right on the curve and ignore what's happening elsewhere:

```
jump = π/8;
fieldvectors = Table[Arrow[Field[x[t], y[t]],
   Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

Show[
 curveplot, fieldvectors, AxesOrigin → {0, 0}, AspectRatio → Automatic,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



These vectors indicate the direction and the speed of the flow at points of the curve at the tails of the field vectors.

Looking at the plot, you can see: that:

→ The net flow of this vector field across this curve is from inside to outside.

→ The net flow of this vector field along this curve is counterclockwise.

B.3) Flow of vector fields along curves; flow of vector fields across curves

□B.3.a.i)

Try to describe the net flow across and along the ellipse parameterized by

P[t] = {5 Cos[t], 3 Sin[t]}

with $0 \le t \le 2\pi$ of a fluid whose velocity is given by the vector field

Field[x, y] = {$\frac{x}{4}$ + y − 1, 0.5 (x + y)}.

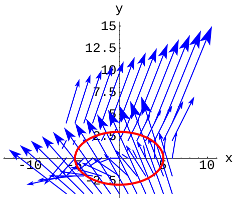
□Answer:

Enter the vector field:

```
Clear[Field, m, n, x, y]
m[x_, y_] = x/4 + y - 1;
n[x_, y_] = 0.5 (x + y^2);
Field[x_, y_] = {m[x, y], n[x, y]}
{-1 + x/4 + y, 0.5 (x + y^2)}
```

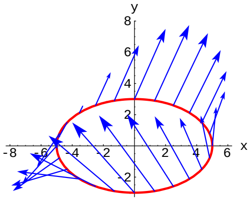
Here's a look at the curve and the vector field:

```
vectorfieldplot = Table[Arrow[Field[x, y], Tail → {x, y}],
                                  {x, -6, 6, 1}, {y, -4, 4, 2}];

Clear[t]
x[t_] = 5 Cos[t];
y[t_] = 3 Sin[t];
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
   PlotStyle → {{Red, Thickness[0.01]}},
   DisplayFunction → Identity];

Show[vectorfieldplot, curveplot,
  Axes → True, AspectRatio → Automatic,
  AxesLabel → {"x", "y"},
  DisplayFunction → $DisplayFunction];
```



Your goal here is to analyze what the flow is doing at the points right on this curve. To do this, look at the field vectors at points right on the curve and ignore what's happening elsewhere:

```
jump = π/12;
fieldvectors = Table[Arrow[Field[x[t], y[t]],
    Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

outcome = Show[curveplot, fieldvectors, AxesOrigin → {0, 0},
  AspectRatio → Automatic, AxesLabel → {"x", "y"},
  DisplayFunction → $DisplayFunction];
```



It's hard to tell from this plot whether the net flow along the curve is clockwise or counterclockwise.

And it's hard to tell from this plot whether the net flow across the curve is from outside to inside or from inside to outside. You can spot some features:

→ The plot indicates clockwise flow along the curve at the top and the bottom with counterclockwise flow along the curve on the left and on the right.

→ The plot also indicates that the flow across the curve is from inside to outside at the top and on the left, but, on the bottom, the flow across the curve is from ouside to inside.

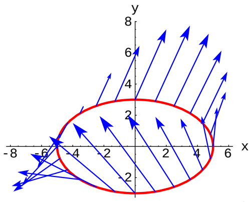
□B.3.a.ii) Using tangential components to help make a visual inspection of the flow ALONG a curve

Here's the plot of

Field[x, y] = {$\frac{x}{4}$ + y − 1, 0.5 (x + y)}

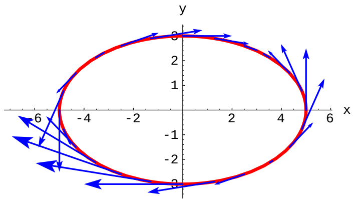
at points on the curve in part i) above:

```
Show[outcome];
```



Look at the components of these field vectors in the direction of the tangent vectors to the curve:

```
Clear[tangent, tancomponent]
tangent[t_] = {x′[t], y′[t]};

tancomponent[t_] = (Field[x[t], y[t]] . tangent[t]) / (tangent[t] . tangent[t]) tangent[t];

actualflowalong = Table[
  Arrow[tancomponent[t], Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

flowalongplot = Show[curveplot, actualflowalong,
  AxesOrigin → {0, 0}, AspectRatio → Automatic, AxesLabel → {"x", "y"},
  DisplayFunction → $DisplayFunction];
```



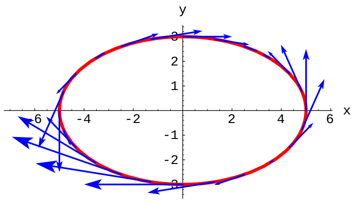
```
This is not a plot of the tangent vectors for the
            parameterization of the curve.
 This is a plot of the components of the field vectors in
       the direction of the tangent vectors for the curve.
```

What does this plot of the tangential components of the field vectors tell you?

□ **Answer:**

Look again:

```
Show[flowalongplot];
```



No doubt about it.

The net flow of this vector field along this curve is clockwise.

□B.3.a.iii) Using normal components to help make a visual inspection of the flow across a curve

What do you do to make a visual estimation of the flow of the vector field in part i) above across the ellipse?

□ **Answer:**

Plot the components of the field vectors in directions perpendicular to the curve. To do this, look at:

```
Clear[tangent, normal]
tangent[t_] = {x′[t], y′[t]};
normal[t_] = {y′[t], -x′[t]};
normal[t] . tangent[t]
```
```
0
```

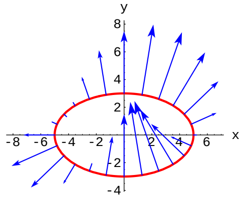
The upshot:

normal[t] = {y′[t], −x′[t]}

is perpendicular to the curve at the point {x[t], y[t]}.

Here comes the plot of the normal components of the field vectors on the curve:

```
Clear[normalcomponent]
                      Field[x[t], y[t]] . normal[t]
normalcomponent[t_] = ───────────────────────────── normal[t];
                          normal[t] . normal[t]
actualflowacross = Table[Arrow[normalcomponent[t],
    Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

flowacrossplot = Show[curveplot, actualflowacross,
  AxesOrigin → {0, 0}, AspectRatio → Automatic, AxesLabel → {"x", "y"},
  DisplayFunction → $DisplayFunction];
```



There is some flow from outside to inside, but there's a heck of a lot more flow from inside to outside.

No doubt about it; the net flow of this vector field across this curve is from INSIDE TO OUTSIDE.

VC.04 2D Vector Fields and Their Trajectories Tutorials

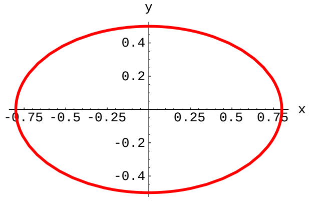
T.1) Flow across and flow along: Visual inspection

Here's a vector field:

```
Clear[Field, m, n, x, y]
m[x_, y_] = 1;
n[x_, y_] = y - Sin[x];
Field[x_, y_] = {m[x, y], n[x, y]}
```
```
{1, y - Sin[x]}
```

Here's a curve:

```
Clear[t]
{x[t_], y[t_]} = {0.8 Cos[t], 0.5 Sin[t]};

curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Red}}, AxesLabel → {"x", "y"}];
```



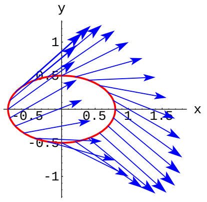
□T.1.a.i) Flow across

Give a visual analysis of the flow of this vector field across this curve.

□Answer:

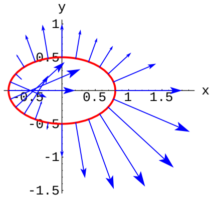
First, plot the vector field on the curve:

```
       π
jump = ──;
       12
fieldvectors = Table[Arrow[Field[x[t], y[t]],
    Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

outcome = Show[curveplot, fieldvectors];
```



It looks like the net flow of this vector field across this curve is from inside to outside, but there is some flow from outside to inside on the left. To get a more accurate picture of the flow of this vector field across this curve, look at the normal components of the field vectors plotted above.

```
Clear[normal]
normal[t_] = {y′[t], -x′[t]};

Clear[normalcomponent]
                      Field[x[t], y[t]] . normal[t]
normalcomponent[t_] = ───────────────────────────── normal[t];
                          normal[t] . normal[t]
actualflowacross = Table[Arrow[normalcomponent[t],
    Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

flowacrossplot = Show[curveplot, actualflowacross];
```



Now there's no doubt about it; the net flow of this vector field across this curve is from inside to outside.

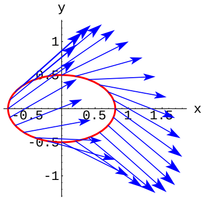
□T.1.a.ii) Flow along

Give a visual analysis of the flow of this vector field along this curve.

□Answer:

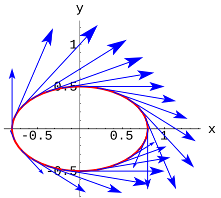
First, plot the vector field on the curve:

```
       π
jump = ──;
       12
fieldvectors = Table[Arrow[Field[x[t], y[t]],
    Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

outcome = Show[curveplot, fieldvectors];
```



It looks like the net flow of this vector field along this curve is clockwise, but there is some counterclockwise flow along this curve at the bottom. To get a more accurate picture of the flow of this vector field along this curve, look at the tangential components of the field vectors plotted above.

```
Clear[tangent]
tangent[t_] = {x′[t], y′[t]};
Clear[tancomponent]
                   Field[x[t], y[t]] . tangent[t]
tancomponent[t_] = ────────────────────────────── tangent[t];
                      tangent[t] . tangent[t]
actualflowalong = Table[
  Arrow[tancomponent[t], Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

flowalongplot = Show[curveplot, actualflowalong];
```



This is not a plot of the tangent vectors for the parameterization of the curve.
This is a plot of the components of the field vectors in the direction of the tangent vectors for the curve.

Now there's no doubt about it; clockwise flow along this curve overwhelms the counterclockwise flow. The net flow of this vector field along this curve is clockwise.

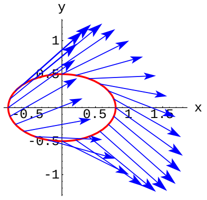
□T.1.a.iii)

Review what happened in the parts above.

□ **Answer:**

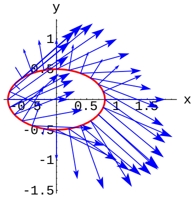
Here's the vector field plotted on the curve:

```
Show[outcome];
```



Here are the normal components of the vectors in the plot above:

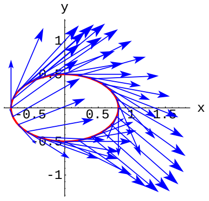
```
Show[outcome, actualflowacross];
```



You use these normal components to read off the net flow across the curve.

Here are the tangential components of the vectors in the plot above:

```
Show[outcome, actualflowalong];
```



You use these tangential components to read off the net flow across the curve.

T.2) Differential equations and and their associated vector fields

Calculus&Mathematica is pleased to say that this problem was greatly influenced by the book
Differential Equations, A Dynamical Systems Approach, Part 1
by J. H. Hubbard and B. H. West (Springer-Verlag, New York,1991).
If you like this problem, then you'll want to experiment with the vector fields in the first chapter of that book.

□T.2.a)

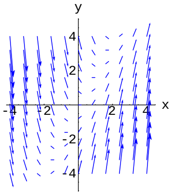
When you have a differential equation like

y′[x] = 2 x − y[x],

you can make the associated vector field

DEField[x, y] = {1, 2 x − y}:

```
Clear[DEField, x, y]
DEField[x_, y_] = {1, 2 x - y};
scalefactor = 0.2;
DEFieldplot =
 Table[Arrow[DEField[x, y], Tail → {x, y}, ScaleFactor → scalefactor],
  {x, -4, 4, 8/10}, {y, -4, 4, 8/10}];

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

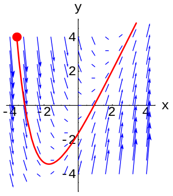


The flow of this field pulls all the trajectories down from the upper left, turns them around, and spits them out at the upper right.

Try it:

```
Clear[m, n]
{m[x_, y_], n[x_, y_]} = DEField[x, y];
{a, b} = {-3.6, 4};
starterpoint = {a, b};

Clear[Derivative, 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 = 7;
approxsolutions = NDSolve[{equationx, equationy, starterx, startery},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory]
```

```
trajectory[t_] =
 {x[t] /. approxsolutions⟦1⟧, y[t] /. approxsolutions⟦1⟧};
trajectoryplot = ParametricPlot[trajectory[t], {t, 0, endtime},
  PlotStyle → {{Red, Thickness[0.01]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];

trajectory = Show[DEFieldplot, starterplot, trajectoryplot,
  PlotRange → All, Axes → Automatic, AxesLabel → {"x", "y"},
  DisplayFunction → $DisplayFunction];
```



What's the significance of the trajectories in the DEField?

Answer:

The trajectory plotted above is nothing but a plot of a solution of the given differential equation

y'[x] = 2 x − y[x] with y[−3.6] = 4.

To see why, look at the ingredients of the trajectory plotter above:

```
{a, b}
```

```
{-3.6, 4}
```

```
equationx = x′[t] == m[x[t], y[t]]
```

```
x′[t] == 1
```

```
equationy = y′[t] == n[x[t], y[t]]
```

```
y′[t] == 2 x[t] - y[t]
```

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

```
x[0] == -3.6
```

```
startery = y[0] == b
```

```
y[0] == 4
```

These equations tell you that this trajectory is the same as the plot of solution of the differential equation

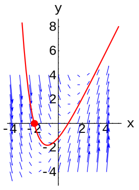
y'[x] = 2 x − y[x] with y[−3.6] = 4.

Here's the plot of the solution of the differential equation

y'[x] = 2 x − y[x] with y[−2] = 0.

as a trajectory in this vector field:

```
{a, b} = {-2, 0};
starterpoint = {a, b};

Clear[Derivative, 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;
begintime = -1;
endtime = 7;
approxsolutions = NDSolve[{equationx, equationy, starterx, startery},
  {x[t], y[t]}, {t, begintime, endtime}];

Clear[trajectory]
trajectory[t_] =
 {x[t] /. approxsolutions⟦1⟧, y[t] /. approxsolutions⟦1⟧};
trajectoryplot =
 ParametricPlot[trajectory[t], {t, begintime, endtime},
  PlotStyle → {{Red, Thickness[0.01]}}, DisplayFunction → Identity];
starterplot = Graphics[{Red, PointSize[0.06], Point[starterpoint]}];

trajectory = Show[DEFieldplot, starterplot, trajectoryplot,
  PlotRange → All, Axes → Automatic, AxesLabel → {"x", "y"},
  DisplayFunction → $DisplayFunction];
```



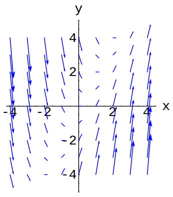
Beautiful, darn beautiful.

□T.2.b)

Here's another look at the vector field coming from the differential equation

y′[x] = 2 x − y[x]:

```
Clear[DEField, x, y]
DEField[x_, y_] = {1, 2 x - y};
scalefactor = 0.2;
DEFieldplot = Table[Arrow[DEField[x, y], Tail → {x, y},
   ScaleFactor → scalefactor], {x, -4, 4, 1}, {y, -4, 4, 1}];

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



What does this plot tell you about the solutions of this differential equation?

□ **Answer:**

Good question.

It tells you a lot. It tells you that no matter how you set a and b so that {a, b} is within the plot range above, the solution of

y′[x] = 2 x − y[x] with y[a] = b

will go down on the left and will turn itself around and go up on the right.

This plot tells you one heckuva lot more than does the formula coming from:

```
Clear[x, y, a, b]
DSolve[{y′[x] == 2 x - y[x], y[a] == b}, y[x], x]
```

$$\{\{y[x] \to E^{-x}\ (-(-2 + 2\ a - b)\ E^{a} - 2\ E^{x} + 2\ E^{x}\ x)\}\}$$

But this formula does tell you that any solution of

y′[x] = 2 x − y[x] with y[a] = b

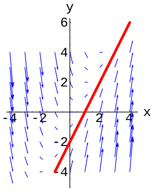
does get sucked onto the line

y = 2 x − 2

when x is large and positive.

You can also see this fact in the DEField plot:

```
sucker = Plot[2 x - 2, {x, -1, 4},
  PlotStyle → {{Thickness[0.02], Red}}, DisplayFunction → Identity];

Show[DEFieldplot, sucker, Axes → Automatic, AxesLabel → {"x", "y"},
 DisplayFunction → $DisplayFunction];
```



That's mathematics at work.

T.3) Flow across a curve and the sign of the dot product

Field[x[t], y[t]] . {y′[t], −x′[t]}

Flow along a curve and the sign of the dot product

Field[x[t], y[t]] . {x′[t], y′[t]}

Given a two dimensional curve parameterized by

{x[t], y[t]} with a ≤ t ≤ b,

you can get a tangent vector at {x[t], y[t]} that points in the direction of the parameterization by setting

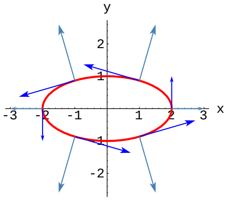
tangent[t] = {x′[t], y′[t]}.

You can also get a normal vector at {x[t], y[t]} by setting

normal[t] = {y′[t], −x′[t]}.

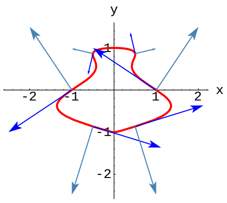
Try them out:

```
Clear[x, y, t, tangent, normal]
{x[t_], y[t_]} = {2 Cos[t], Sin[t]};
tangent[t_] = {x′[t], y′[t]};
normal[t_] = {y′[t], -x′[t]};
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Red}}, DisplayFunction → Identity];
jump = π/3;
tangentvectors =
 Table[Arrow[tangent[t], Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];
normalvectors = Table[Arrow[normal[t], Tail → {x[t], y[t]},
   VectorColor → SteelBlue], {t, 0, 2 π - jump, jump}];

Show[curveplot, tangentvectors, normalvectors, AxesLabel → {"x", "y"},
 AspectRatio → Automatic, DisplayFunction → $DisplayFunction];
```



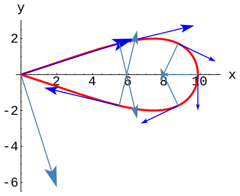
The tangent vectors, {x′[t], y′[t]}, tell you that the curve is parameterized in the counterclockwise way.
The normal vectors, {y′[t], −x′[t]}, point to the outside of the curve.
Try another curve:

```
Clear[x, y, t, tangent, normal]
{x[t_], y[t_]} = {Cos[t] (1 - 0.5 Sin[3 t]), Sin[t]};
tangent[t_] = {x′[t], y′[t]};
normal[t_] = {y′[t], -x′[t]};
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Red}}, DisplayFunction → Identity];
jump = π/3;
tangentvectors =
 Table[Arrow[tangent[t], Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];
normalvectors = Table[Arrow[normal[t], Tail → {x[t], y[t]},
   VectorColor → SteelBlue], {t, 0, 2 π - jump, jump}];

Show[curveplot, tangentvectors, normalvectors, AxesLabel → {"x", "y"},
 AspectRatio → Automatic, DisplayFunction → $DisplayFunction];
```



Again, the tangent vectors {x′[t], y′[t]} tell you that the curve is parameterized in the counterclockwise way.
The normal vectors {y′[t], −x′[t]} point to the outside of the curve.
Try another curve:

```
Clear[x, y, t, tangent, normal]
{x[t_], y[t_]} = {t (2 π - t), 2 Sin[t]};
tangent[t_] = {x′[t], y′[t]};
normal[t_] = {y′[t], -x′[t]};
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Red}}, DisplayFunction → Identity];
jump = π/3;
tangentvectors =
 Table[Arrow[tangent[t], Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];
normalvectors = Table[Arrow[normal[t], Tail → {x[t], y[t]},
   VectorColor → SteelBlue], {t, 0, 2 π - jump, jump}];

Show[curveplot, tangentvectors, normalvectors, AxesLabel → {"x", "y"},
 DisplayFunction → $DisplayFunction];
```



This time, the tangent vectors {x′[t], y′[t]} tell you that the curve is parameterized in the clockwise way.
This time, the normal vectors {y′[t], −x′[t]} point to the inside of the curve.

□T.3.a.i) Counterclockwise versus clockwise

Is it true that:
→ When a closed curve is parameterized by {x[t], y[t]} in the counterclockwise way, then the normal vectors {y′[t], −x′[t]} always point to the outside of the curve?
→ When a closed curve is parameterized by {x[t], y[t]} in the clockwise way, then the normal vectors {y′[t], −x′[t]} always point to the inside of the curve?

□ **Answer:**

Yes.

In fact, as you walk around the curve in the direction of the parameterization, the normal vector {y′[t], −x′[t]} always points to your right.

□T.3.a.ii) Your choice of normals

Given a curve parameterized by
{x[t], y[t]} with a ≤ t ≤ b,
you have your choice of normal vectors at {x[t], y[t]}:
You can opt for
{y′[t], −x′[t]},
or you can go with
D[unittan[t], t]
where

$$\text{unittan}[t] = \frac{\{x'[t], y'[t]\}}{\sqrt{x'[t]^2 + y'[t]^2}}.$$

Discuss the merits of each.

□ **Answer:**

For hand work, {y′[t], −x′[t]} is more easily calculated than is D[unittan[t], t]. The choice of which to use depends on what you want to do.

→ If you want a normal that always points to the right as you advance on the curve in the direction of the parameterization, go with {y′[t], −x′[t]}.

→ If you want a normal that always points with the curvature, go with D[unittan[t], t].

You probably know how handy D[unittan[t], t] can be.

If you want to see how handy {y′[t], −x′[t]} can be, go on to the next problem.

□T.3.b) Flow across a curve and the sign of the dot product

Field[x[t], y[t]] . {y′[t], −x′[t]}

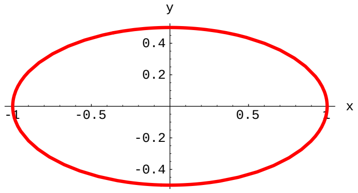
Here's a vector field:

```
Clear[Field, m, n, x, y]
m[x_, y_] = 1.4 x^2;
n[x_, y_] = y - Sin[x];
Field[x_, y_] = {m[x, y], n[x, y]}
```

```
{1.4 x^2, y - Sin[x]}
```

Here's a curve:

```
Clear[t]
{x[t_], y[t_]} = {Cos[t], 0.5 Sin[t]};

curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Red}}, AxesLabel → {"x", "y"}];
```

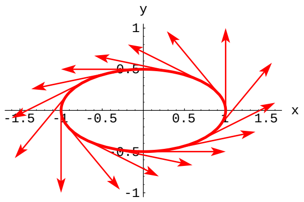


Look at this plot of some of the unit tangent vectors

$$\text{unittangent}[t] = \frac{\{x'[t], y'[t]\}}{\sqrt{x'[t]^2 + y'[t]^2}}$$

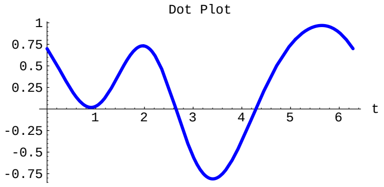
pointing in the direction of the parameterization:

```
Clear[t, unittangent]
unittangent[t_] = {x′[t], y′[t]}/Sqrt[x′[t]^2 + y′[t]^2];

jump = π/8;
unittangents = Table[Arrow[unittangent[t],
   Tail → {x[t], y[t]}, VectorColor → Red], {t, 0, 2 π - jump, jump}];

Show[curveplot, unittangents];
```



This plot tells you that the curve is parameterized in the counterclockwise way.
And this tells you that the normal vector {y′[t], −x′[t]} at {x[t], y[t]} points to the outside of the curve.
Now look at this plot of
Field[x[t], y[t]] . {y′[t], −x′[t]}:

```
dotplot = Plot[Field[x[t], y[t]] . {y′[t], -x′[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Blue}}, AxesLabel → {"t", ""},
  AspectRatio → 1/2, PlotLabel → "Dot Plot"];
```



Interpret this plot.

□ **Answer:**

This is a plot of

Field[x[t], y[t]] . {y′[t], −x′[t]}.

When

Field[x[t], y[t]] . {y′[t], −x′[t]} > 0,

the flow of the vector field across the curve is in the direction of the outward normal vector {y′[t], −x′[t]}.

When

Field[x[t], y[t]] . {y′[t], −x′[t]} < 0,

the flow of the vector field across the curve is opposite to the direction of the outward normal vector {y′[t], −x′[t]}.

Since the normals {y′[t], −x′[t]} point out from the curve, the upshot is this:

At a point {x[t], y[t]} on the curve, the flow across the curve is:

→ from inside to outside when

Field[x[t], y[t]] . {y′[t], −x′[t]} > 0

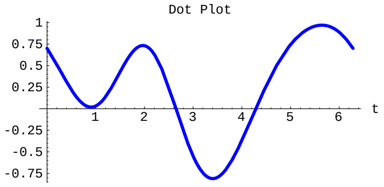
→ from outside to inside when

Field[x[t], y[t]] . {y′[t], −x′[t]} < 0.

Armed with this insight, take another look at the plot of

Field[x[t], y[t]] . {y′[t], −x′[t]}:

```
Show[dotplot];
```



Eye-ball estimates tell you that the flow of this vector field across this curve is

→ from inside to outside at points {x[t], y[t]} with

0 ≤ t < 2.5 and 4.3 < t ≤ 2 π

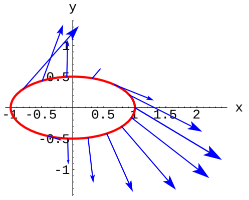
→ from outside to inside at points {x[t], y[t]} with

2.7 < t < 4.1.

Using FindRoot, you can get more accurate estimates if you like.

Check it out:

```
out1 = Table[
  Arrow[Field[x[t], y[t]], Tail → {x[t], y[t]}], {t, 0, 2.5, (2.5 + 0)/6}];
out2 = Table[Arrow[Field[x[t], y[t]], Tail → {x[t], y[t]}],
  {t, 4.3, 2 π, 1/6 (2 π - 4.3)}];
in1 = Table[Arrow[Field[x[t], y[t]], Tail → {x[t], y[t]}],
  {t, 2.7, 4.1, (4.1 - 2.7)/6}];
Show[curveplot, out1, out2];
```

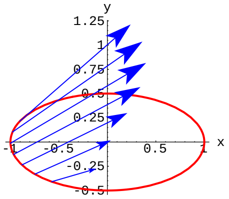


Outward flow across the curve at the plotted points on the curve.

This is a good representative plot of the part of this vector field that flows from inside to outside across this curve.

Now look at:

```
Show[curveplot, in1];
```



Inward flow across the curve at the plotted points on the curve.

This is a good representative plot of the part of this vector field that flows from outside to inside across this curve.

And you set up both plots by doing nothing more than eyeballing a plot of

Field[x[t], y[t]] . {y′[t], −x′[t]}.

Math works.

□T.3.c) Flow along a curve and the sign of the dot product Field[x[t], y[t]] . {x′[t], y′[t]}

Alert C&M participants will note that this problem is nothing but a copy and paste job of part b) above.

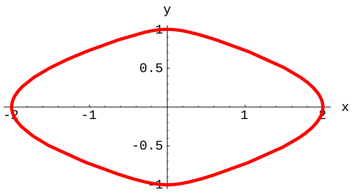
Here's a vector field:

```
Clear[Field, m, n, x, y]
m[x_, y_] = 3 x - y^2;
n[x_, y_] = 2 y;
Field[x_, y_] = {m[x, y], n[x, y]}
```

$\{3x - y^2, 2y\}$

Here's a curve:

```
Clear[t]
{x[t_], y[t_]} = {Cos[t] (Cos[t]^2 + 1), Sin[t]};

curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Red}}, AxesLabel → {"x", "y"}];
```

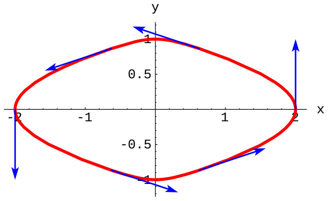


Look at this plot of some of the unit tangent vectors

$$\text{unittangent}[t] = \frac{\{x'[t], y'[t]\}}{\sqrt{x'[t]^2 + y'[t]^2}}$$

pointing in the direction of the parameterization:

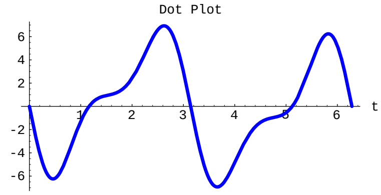
```
Clear[t, unittangent]
unittangent[t_] = {x′[t], y′[t]} / Sqrt[x′[t]^2 + y′[t]^2];

jump = π/3;
unittangents = Table[
  Arrow[unittangent[t], Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

Show[curveplot, unittangents];
```



This plot tells you that the curve is parameterized in the counterclockwise way.
Now look at this plot of

Field[x[t], y[t]] . {x′[t], y′[t]}:

```
dotplot = Plot[Field[x[t], y[t]] . {x′[t], y′[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Blue}}, AxesLabel → {"t", ""},
  AspectRatio → 1/2, PlotLabel → "Dot Plot"];
```



Interpret this plot.

□**Answer:**

This is a plot of

Field[x[t], y[t]] . {x′[t], y′[t]}.

When

Field[x[t], y[t]] . {x′[t], y′[t]} > 0,

the flow of the vector field along the curve is in the direction of the tangent vector {x′[t], y′[t]}.

When

Field[x[t], y[t]] . {x′[t], y′[t]} < 0,

the flow of the vector field along the curve is opposite the direction of the tangent vector {x′[t], y′[t]}.

Because the tangents {x′[t], y′[t]} point in the counterclockwise direction, the upshot is:

At a point {x[t], y[t]} on the curve, the flow along the curve is:

→ counterclockwise when Field[x[t], y[t]] . {x′[t], y′[t]} > 0

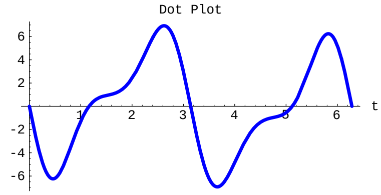
→ clockwise when Field[x[t], y[t]] . {x′[t], y′[t]} < 0.

Armed with this insight, take another look at the plot of

Field[x[t], y[t]] . {x′[t], y′[t]}

again:

```
Show[dotplot];
```



Eye-ball estimates tell you that the flow of this vector field along this curve is

→ clockwise at points {x[t], y[t]} with

$0 \le t \le 1$ and $3.1 \le t \le 5.1$

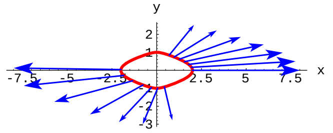
→ counterclockwise at points {x[t], y[t]} with

$1.2 \le t \le 3$ and $5.2 \le t \le 2\pi$.

Using FindRoot, you can get more precise estimates if you like.

Check it out:

```
clockwise1 = Table[
  Arrow[Field[x[t], y[t]], Tail → {x[t], y[t]}], {t, 0, 1, (1 + 0)/6}];
clockwise2 = Table[Arrow[Field[x[t], y[t]],
   Tail → {x[t], y[t]}], {t, 3.1, 5.1, (5.1 - 3.1)/6}];
counterclockwise1 = Table[
  Arrow[Field[x[t], y[t]], Tail → {x[t], y[t]}], {t, 1.2, 3, (3 - 1.2)/6}];
counterclockwise2 = Table[Arrow[Field[x[t], y[t]],
   Tail → {x[t], y[t]}], {t, 5.2, 2 π, 1/6 (2 π - 5.2)}];

Show[curveplot, clockwise1, clockwise2];
```

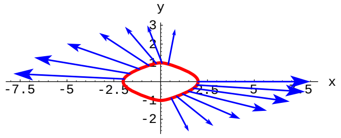


Clockwise flow along the curve at the plotted points on the curve.

This is a good representative plot of the part of this vector field that flows in the clockwise direction along this curve.

Now look at:

```
Show[curveplot, counterclockwise1, counterclockwise2];
```



Counterclockwise flow along the curve at the plotted points on the curve.

This is a good representative plot of the part of this vector field that flows in the counterclockwise direction along this curve.

And you set up both plots by doing nothing more than eyeballing a plot of

Field[x[t], y[t]] . {x′[t], y′[t]}.

Math works again.

T.4) The 2D electric field

In the electric interpretation, a vector field

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

is a vector such that

ElectricField[x, y] . U

measures the instantaneous voltage drop when you leave {x, y} in the direction of a unit vector U.
This is the same as saying ElectricField[x, y] represents the force the field places on a unit charge at the point {x, y}.

□T.4.a)

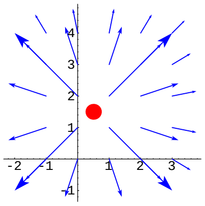
Here's a plot of the electric field resulting from a point charge of size 2 at {0.50, 1.50}:

```
Clear[ElectricField, x, y]
q = 2;
{a, b} = {0.50, 1.50};
ElectricField[x_, y_] = q {x - a, y - b}/((x - a)^2 + (y - b)^2)
```

$$\left\{\frac{2\,(-0.5+x)}{(-0.5+x)^2+(-1.5+y)^2},\ \frac{2\,(-1.5+y)}{(-0.5+x)^2+(-1.5+y)^2}\right\}$$

Here's how this electric field looks:

```
Show[Graphics[{Red, PointSize[0.08], Point[{a, b}]}],
     Table[Arrow[ElectricField[x, y], Tail → {x, y}],
       {x, -1, 3}, {y, 0, 4}], Axes → Automatic];
```



Ta-Da.
How do you interpret this plot?

□ **Answer:**

The electric field vector, ElectricField[x, y], plotted as above, with tail at {x, y} points in the direction of biggest voltage drop, and the length of the vector represents the size of the voltage drop in that direction.

Reason: If you leave {x, y} in the direction of a unit vector U, then the instantaneous voltage drop is

ElectricField[x, y] . U

= ||ElectricField[x, y]|| ||U|| Cos[angle between]

= ||ElectricField[x, y]|| Cos[angle between]

because ||U|| = 1.

This tells you that the instantaneous voltage drop is as big as possible when

ElectricField[x, y] . U

= ||ElectricField[x, y]|| Cos[angle between]

is as big as possible. This happens when

Cos[angle between] = 1,

which is the same as saying that angle between = 0.

The upshot:

To get the biggest instantaneous voltage drop, you take

$$U = \frac{\text{ElectricField[x,y]}}{\text{||ElectricField[x,y]||}}.$$

The direction of the biggest instantaneous voltage drop is the direction of

ElectricField[x, y].

The size of the biggest instantaneous voltage drop is

$$\text{ElectricField[x, y]} . \frac{\text{ElectricField[x,y]}}{\text{||ElectricField[x,y]||}}$$

$$= \frac{\text{||ElectricField[x,y]||}^2}{\text{||ElectricField[x,y]||}}$$

$$= \text{||ElectricField[x, y]||}.$$

This means that ElectricField[x, y] points in the direction of the biggest instantaneous voltage drop, and that the instantaneous voltage drop in this direction is ||ElectricField[x, y]||.

□T.4.b)

What do you get when you put a charge of size
1.5 at {−0.875, 0.375},
and a charge of size
−1.5 at {0.875, 0.375}?

□Answer:

You get something pretty interesting.

Start by adding the electric fields resulting from the two point charges:

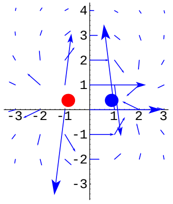
```
Clear[ElectricField1, ElectricFieldF2, CombinedElectricField, x, y]
q1 = 1.5;
{a1, b1} = {-0.875, 0.375};
```

$$\text{ElectricField1[x\_, y\_]} = \frac{\text{q1 \{x - a1, y - b1\}}}{\text{(x - a1)}^2 + \text{(y - b1)}^2};$$

```
q2 = -1.5;
{a2, b2} = {0.875, 0.375};
```

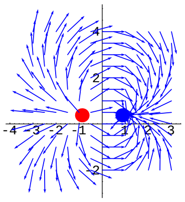
$$\text{ElectricField2[x\_, y\_]} = \frac{\text{q2 \{x - a2, y - b2\}}}{\text{(x - a2)}^2 + \text{(y - b2)}^2};$$

```
CombinedElectricField[x_, y_] =
 ElectricField1[x, y] + ElectricField2[x, y];

Show[Graphics[{Red, PointSize[0.08], Point[{a1, b1}]}],
 Graphics[{Blue, PointSize[0.08], Point[{a2, b2}]}], Table[Arrow[
   CombinedElectricField[x, y], Tail → {x, y}], {x, -3, 3}, {y, -2, 4}],
 Axes → Automatic];
```



That's a little hard to interpret. To get a better idea, look at the unit electric field, which shows the direction of the greatest instantaneous voltage drop but does not show the size of the drop.

```
Clear[unitfieldvector]
unitfieldvector[x_, y_] = CombinedElectricField[x, y] /
   (√(CombinedElectricField[x, y] . CombinedElectricField[x, y]));
Show[Graphics[{Red, PointSize[0.08], Point[{a1, b1}]}],
 Graphics[{Blue, PointSize[0.08], Point[{a2, b2}]}],
 Table[Arrow[unitfieldvector[x, y], Tail → {x, y}],
  {x, -3, 3, 0.5}, {y, -2, 4, 0.5}],
 Axes → Automatic];
```



Hot plot.

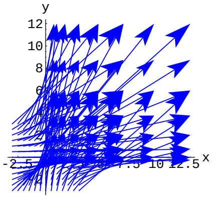
Well worth waiting for.

The juice is pouring away from the positive charge toward the negative charge just like it does in the movies.

T.5) Troubleshooting plots of vector fields

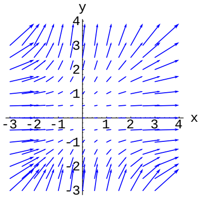
To get a good visualization of a vector field, you can plot the field as it comes:

```
Clear[Field, x, y]
Field[x_, y_] = {x^2 + 1, y^2};
fieldplot = Table[
   Arrow[Field[x, y], Tail → {x, y}], {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 scale factor:

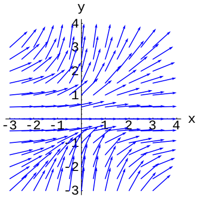
```
scalefactor = 0.1;
scaledfieldplot =
 Table[Arrow[Field[x, y], Tail → {x, y}, ScaleFactor → scalefactor],
  {x, -3, 3, 0.5}, {y, -3, 3, 0.5}];

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.

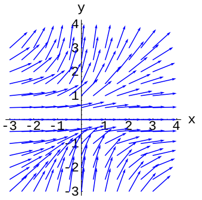
Sometimes you can't get any satisfactory scaled plot. In this case you might want to try the unit field plot.

```
Clear[unitField]
                         Field[x, y]
unitField[x_, y_] = ─────────────────────────── ;
                    √Field[x, y] . Field[x, y]
unitfieldplot = Table[Arrow[unitField[x, y], Tail → {x, y}],
   {x, -3, 3, 0.5}, {y, -3, 3, 0.5}];

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



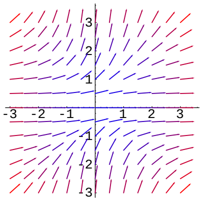
This plot of unitField[x, y] shows only direction and not speed of the flow. You can also get the unitfieldplot with the option ScaleFactor → Normalize:

```
unitfieldplot = Table[Arrow[Field[x, y], Tail → {x, y},
    ScaleFactor → Normalize], {x, -3, 3, 0.5}, {y, -3, 3, 0.5}];

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



There's one more neat trick that folks like to do: 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]]] , 0, 1. - N[Norm[Field[x, y]]] ];
           ───────────────────          ───────────────────
                   max                          max
colorfieldplot = Table[Arrow[unitField[x, y], Tail → {x, y},
    VectorColor → Evaluate[color[x, y]], ScaleFactor → 0.5],
   {x, -3, 3, .5}, {y, -3, 3, .5}];

Show[colorfieldplot, Axes → Automatic];
```



The hottest vectors are red and the coolest are blue.
Not bad, huh?

□T.5.a)

Plots of unit fields like the plot above are usually a joy to view but producing them has some inherent drawbacks.
What are these drawbacks?

□ **Answer:**

→ Because all the vectors are the same length, you are not able to use this plot to visualize the speed of the flow. All you see is the direction.

→ If Field[x, y] has a complicated formula, then making the table of unit field vectors can burn a heck of a lot of computer time.

Still, when the conditions are right, unit field plots can be irresistible.

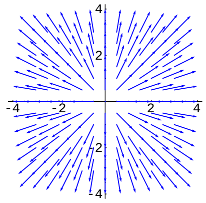
As always in Calculus&Mathematica, the choice is yours.

□T.5.b)

Here's an unsuccessful attempt to plot unitField[x, y] for
Field[x, y] = {x, y}:

```
Clear[Field, x, y]
Field[x_, y_] = {x, y};
Clear[unitField]
                         Field[x, y]
unitField[x_, y_] = ─────────────────────────── ;
                    √Field[x, y] . Field[x, y]
unitfieldplot = Table[Arrow[unitField[x, y], Tail → {x, y}],
   {x, -3, 3, 0.5}, {y, -3, 3, 0.5}];
Show[unitfieldplot, Axes → Automatic];
```

```
∞::indet : Indeterminate expression 0. ComplexInfinity encountered.
```



Your job is to say why the plot didn't run successfully, and to replace it with a workable plot.

□ **Answer:**

This plot went with Field[x, y] = {x, y}.

It didn't run because it attempted to plot

unitField[x, y] = $\frac{\{x, y\}}{\sqrt{x^2 + y^2}}$ when {x, y} = {0, 0}.

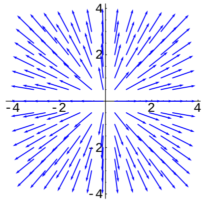
This turned out to be garbage because

unitField[0, 0] computes out to be $\frac{\{0, 0\}}{\sqrt{0^2 + 0^2}}$,

and neither Mathematica nor anything else can make sense of this.

You can fix this by jiggling the plotting limits so the point {x, y} = {0, 0} doesn't come up in the plot:

```
unitfieldplot = Table[Arrow[unitField[x, y], Tail → {x, y}],
   {x, -3.1, 2.9, 0.5}, {y, -3, 3, 0.5}];
Show[unitfieldplot, Axes → Automatic];
```



All right.

VC.04 2D Vector Fields and Their Trajectories
Give Them a Try!

Experience with the starred problems will be useful for understanding developments later in the course.

G.1) Looking for sinks (drains)*

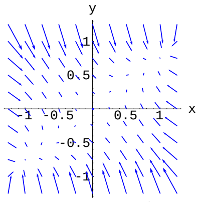
□G.1.a)

Calculus&Mathematica thanks former C&M student
Jennifer Lee Cassidy for suggesting this problem.

Here is a vector field and a plot:

```
Clear[Field, x, y]
Field[x_, y_] = {-0.036 x E^(x^2) + 0.123 y, 0.123 x - 0.087 y E^(y^2)};
```

```
fieldplot = Table[Arrow[Field[x, y]/(1 + Norm[Field[x, y]]), Tail → {x, y}],
   {x, -1.25, 1.25, 2.5/10}, {y, -1.25, 1.25, 2.5/10}];

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



As you can see from the plot, if you drop a cork into this flow at any starting point within the plot other than {0, 0}, then it will float along to one of two special points and stop dead.
Plot some trajectories in an effort to find the approximate locations of these two special points.

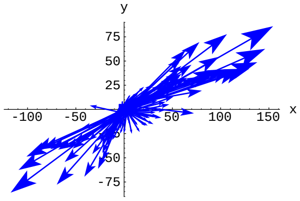
□ **Tip:**

Copy, paste and edit.

□G.1.b) Scaling a vector field

Here's an unscaled plot of a two dimensional vector field:

```
Clear[Field, x, y]
Field[x_, y_] = {18.4 - 0.8 x^3 - 8.2 y, -8.2 x - 0.4 y^3};
fieldplot =
 Table[Arrow[Field[x, y], Tail → {x, y}], {x, -5, 5}, {y, -5, 5}];

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



This plot sucks. You can't see what's happening.
Give a scaled plot of this vector field at the same points to show it off to good advantage.
Pretend that this field represents fluid flow and use your good plot to determine whether there are any drains within the plot. If you can't spot them visually, run some trajectories and see where they lead you.

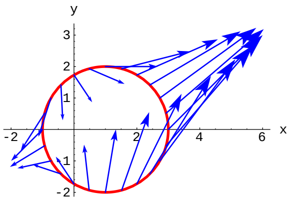
G.2) Flow along and flow across*

Here's the vector field

Field[x, y] = {x + $\frac{y}{3}$, x − $\frac{y}{2}$}

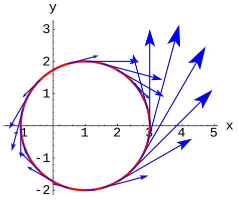
plotted at selected points on the circle of radius 2 centered at {1, 0}:

```
Clear[Field, m, n, x, y]
m[x_, y_] = x + y/3;
n[x_, y_] = x - y/2;
Field[x_, y_] = {m[x, y], n[x, y]};
Clear[t]
{x[t_], y[t_]} = {1, 0} + 2 {Cos[t], Sin[t]};
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {Thickness[0.01], Red}}, DisplayFunction → Identity];
jump = π/12;
fieldoncurve = Table[Arrow[Field[x[t], y[t]],
   Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

Show[
 curveplot, fieldoncurve, AxesOrigin → {0, 0}, AspectRatio → Automatic,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



Here are the tangential components of the field vectors shown above:

```
Clear[tangent, tancomponent]
tangent[t_] = {x′[t], y′[t]};
tancomponent[t_] = (Field[x[t], y[t]] . tangent[t])/(tangent[t] . tangent[t]) tangent[t];
actualflowalong = Table[
  Arrow[tancomponent[t], Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

flowalongplot = Show[curveplot, actualflowalong,
  AxesOrigin → {0, 0}, AspectRatio → Automatic, AxesLabel → {"x", "y"},
  DisplayFunction → $DisplayFunction];
```



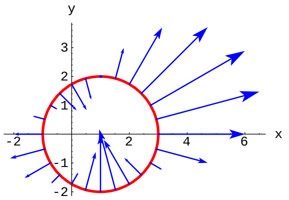
□G.2.a.i)

You make the call:
Is the net flow of this vector field along this curve clockwise or counterclockwise?

□G.2.a.ii)

Go with the same vector field and the same curve as above, but this time look at normal components of the vector field at points on the curve:

```
Clear[normal, normalcomponent]
normal[t_] = {y′[t], -x′[t]};
normalcomponent[t_] = (Field[x[t], y[t]] . normal[t])/(normal[t] . normal[t]) normal[t];
actualflowacross = Table[Arrow[normalcomponent[t],
   Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

flowacrossplot = Show[curveplot, actualflowacross,
  AspectRatio → Automatic, AxesLabel → {"x", "y"},
  DisplayFunction → $DisplayFunction];
```

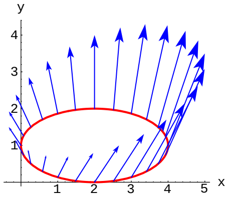


You make the call:
Is the net flow of this vector field across this curve from inside to outside or from outside to inside?

G.2.b)

Here's a new vector field plotted at a selection of points on an ellipse:

```
Clear[Field, m, n, x, y, t]
m[x_, y_] = (x - y)/3;
n[x_, y_] = (x + y)/2;
Field[x_, y_] = {m[x, y], n[x, y]};
{x[t_], y[t_]} = {2, 1} + {2 Cos[t], Sin[t]};
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {Thickness[0.01], Red}}, DisplayFunction → Identity];
jump = π/12;
fieldoncurve = Table[Arrow[Field[x[t], y[t]],
   Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

Show[
 curveplot, fieldoncurve, AxesOrigin → {0, 0}, AspectRatio → Automatic,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```

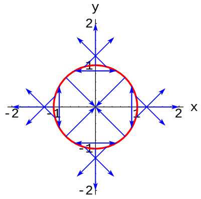


Plot, as in parts a.i) and a.ii), the tangential and normal components of the field vectors on this curve and then make the calls:
→ Is the net flow of this vector field along this curve clockwise or counterclockwise?
→ Is the net flow of this vector field across this curve from inside to outside or from outside to inside?

□G.2.c)

Here's a new vector field plotted at a selection of points on the circle of radius 1 centered at {0, 0}:

```
Clear[Field, m, n, x, y, t]
m[x_, y_] = x^3 - 3 x y^2;
n[x_, y_] = -3 x^2 y + y^3;
Field[x_, y_] = {m[x, y], n[x, y]};
{x[t_], y[t_]} = {Cos[t], Sin[t]};
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Red}}, DisplayFunction → Identity];
jump = π/12;
fieldoncurve = Table[Arrow[Field[x[t], y[t]],
   Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

Show[curveplot, fieldoncurve,
 Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"},
 DisplayFunction → $DisplayFunction];
```



Plot, as in parts a.i) and a.ii), the tangential and normal components of these field vectors on this curve and then make the calls:
→ Is the net flow of this vector field along this curve clockwise, counterclockwise, or is the flow of this vector field along this curve 0 in the sense that the counterclockwise flow balances the clockwise flow?
→ Is the net flow of this vector field across this curve from inside to outside or from outside to inside, or is the flow of this vector field across this curve 0 in the sense that the inward flow balances the outward flow?

G.3) Normals, tangents and dot plots*

□G.3.a.i) Direction of the tangent vector {x′[t], y′[t]} at {x[t], y[t]}

When a closed curve is parameterized by {x[t], y[t]}, then as you advance along the curve in the direction of the parameterization, which way do the tangent vectors {x′[t], y′[t]} at {x[t], y[t]} point; in the direction you are going, or in the direction opposite to the direction you are going?

□G.3.a.ii) Direction of the normal vector {y′[t], −x′[t]} at {x[t], y[t]}

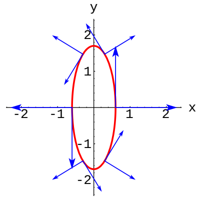
for a counterclockwise parameterization

When a closed curve is parameterized by {x[t], y[t]} in the counterclockwise way, then as you advance along the curve in the direction of the parameterization, which way do the normal vectors {y′[t], −x′[t]} at {x[t], y[t]} point, to the right to the outside of the curve, or to the left to the inside of the curve?

□ **Tip:**

Take a look at this:

```
Clear[x, y, t, tangent, normal]
{x[t_], y[t_]} = {0.6 Cos[t], 1.7 Sin[t]};
tangent[t_] = {x′[t], y′[t]};
normal[t_] = {y′[t], -x′[t]};
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Red}}, DisplayFunction → Identity];
jump = π/3;
tangentvectors =
 Table[Arrow[tangent[t], Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];
normalvectors =
 Table[Arrow[normal[t], Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

Show[curveplot, tangentvectors, normalvectors, AxesLabel → {"x", "y"},
 AspectRatio → Automatic, DisplayFunction → $DisplayFunction];
```



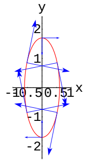
□G.3.a.iii) Direction of the normal vector {y′[t], −x′[t]} at {x[t], y[t]}

When a closed curve is parameterized by {x[t], y[t]} in the clockwise way, then as you advance along the curve in the direction of the parameterization, which way do the normal vectors {y′[t], −x′[t]} at {x[t], y[t]} point, to the right to the inside of the curve, or to the left to the outside of the curve?

□ **Tip:**

Take a look at this:

```
Clear[x, y, t, tangent, normal]
{x[t_], y[t_]} = {0.6 Sin[t], 1.7 Cos[t]};
tangent[t_] = {x′[t], y′[t]};
normal[t_] = {y′[t], -x′[t]};
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Red}}, DisplayFunction → Identity];
jump = π/3;
tangentvectors =
 Table[Arrow[tangent[t], Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];
normalvectors =
 Table[Arrow[normal[t], Tail → {x[t], y[t]}], {t, 0, 2 π - jump, jump}];

Show[curveplot, tangentvectors, normalvectors, AxesLabel → {"x", "y"},
 AspectRatio → Automatic, DisplayFunction → $DisplayFunction];
```



□G.3.b.i)

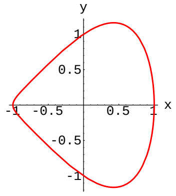
Here's a vector field:

```
Clear[Field, m, n, x, y]
m[x_, y_] = 1;
n[x_, y_] = Sin[x y];
Field[x_, y_] = {m[x, y], n[x, y]}
```

```
{1, Sin[x y]}
```

Here's a curve:

```
Clear[t]
{x[t_], y[t_]} = {Cos[t], Sin[t] + 1/3 Sin[2 t]};
curveplot = ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Red}}, AxesLabel → {"x", "y"}];
```

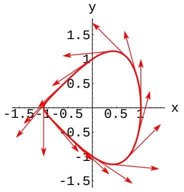


Look at this plot of some of the unit tangent vectors

$$\text{unittangent}[t] = \frac{\{x'[t], y'[t]\}}{\sqrt{x'[t]^2 + y'[t]^2}}$$

pointing in the direction of the parameterization:

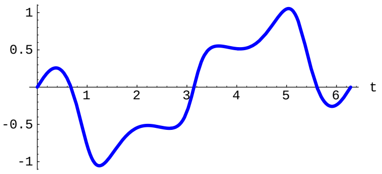
```
Clear[t, unittangent]
unittangent[t_] = {x′[t], y′[t]} / √(x′[t]^2 + y′[t]^2);
jump = π/8;
unittangents = Table[Arrow[unittangent[t],
    Tail → {x[t], y[t]}, VectorColor → Red], {t, 0, 2 π - jump, jump}];
Show[curveplot, unittangents];
```



Now look at this plot of

Field[x[t], y[t]] . unittangent[t]:

```
dotplot = Plot[Field[x[t], y[t]] . unittangent[t], {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Blue}}, AxesLabel → {"t", ""},
  AspectRatio → 1/2];
```



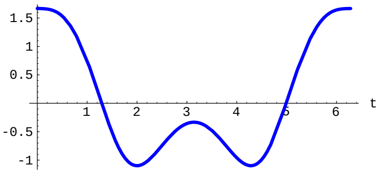
Interpret this plot.

□G.3.b.ii)

Go with the same vector field and curve as in part i) and look at this plot of

Field[x[t], y[t]] . {y′[t], −x′[t]}:

```
dotplot = Plot[Field[x[t], y[t]] . {y′[t], -x′[t]}, {t, 0, 2 π},
  PlotStyle → {{Thickness[0.01], Blue}}, AxesLabel → {"t", ""},
  AspectRatio → 1/2];
```



Interpret this plot. Does your answer change if you replace

{y′[t], −x′[t]}

by the normalized vector

$\frac{\{y'[t], -x'[t]\}}{\sqrt{x'[t]^2 + y'[t]^2}}$?

G.4) The most important vector field of them all: The gradient field*

One way to come up with a vector field is to take a function f[x, y] and put

Field[x, y] = gradf[x, y] = ∇f[x, y].

Try it out:

```
Clear[f, gradf, Field, x, y]
f[x_, y_] = 1 - 1/2 (x^2 + 1.5 y^2);
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
Field[x_, y_] = gradf[x, y]
{-x, -1.5 y}
```

This is the gradient field of

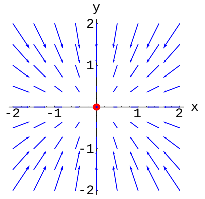
$f[x, y] = 1 - \frac{1}{2}(x + 1.5\, y).$

Note that {x, y} = {0, 0} maximizes f[x, y].

Here's a plot of the scaled gradient field of f[x, y] shown with the maximizer at {0, 0}:

```
maximizerplot = Graphics[{Red, PointSize[0.04], Point[{0, 0}]}];
scalefactor = 0.2;
```

```
gradfieldplot =
  Table[Arrow[gradf[x, y], Tail → {x, y}, ScaleFactor → scalefactor],
   {x, -2, 2, 0.5}, {y, -2, 2, 0.5}];

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



□G.4.a.i)

Why did this happen?

Where are the trajectories in this gradient field headed?

What do you learn about a function f[x, y] by looking at a plot of its gradient field?

□G.4.a.ii)

This time go with the function

$f[x, y] = \frac{2}{1 + (y-2)^2 + 1.5\,(x-1)^2}.$

Say why you are sure that {x, y} = {1, 2} maximizes f[x, y]. Then plot the (scaled) gradient field of f[x, y] near {1, 2}.

Describe what you see and explain why you see it.

□G.4.a.iii)

You are given a certain function f[x, y] and the information that

{a, b, f[a, b]}

sits on the top of a crest of the surface given by

z = f[x, y].

You center a small circle at {a, b} and plot the gradient field of f[x, y] on this circle.

Is the net flow of the gradient field of f[x, y] across this circle from inside to outside or from outside to inside?

□G.4.b.i) Minimizers

This time go with

$f[x, y] = e^{\frac{1}{9}(x^2 + 2\,y^2)}.\ .$

```
Clear[f, gradf, Field, x, y]
f[x_, y_] = E^(1/9 (x^2+2 y^2));
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
Field[x_, y_] = gradf[x, y]
{2/9 E^(1/9 (x^2+2 y^2)) x, 4/9 E^(1/9 (x^2+2 y^2)) y}
```

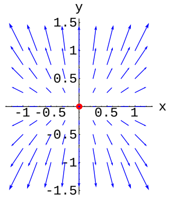
This is the gradient field of

$f[x, y] = e^{\frac{1}{9}(x^2 + 2\,y^2)}.$

Note that {x, y} = {0, 0} minimizes f[x, y].

Here's a plot of the scaled gradient field of f[x, y] shown with the minimizer at {0, 0}:

```
minimizerplot = Graphics[{Red, PointSize[0.04], Point[{0, 0}]}];
scalefactor = 0.8;
gradfieldplot =
  Table[Arrow[gradf[x, y], Tail → {x, y}, ScaleFactor → scalefactor],
   {x, -1, 1, 0.25}, {y, -1, 1, 0.25}];

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



Why did this happen?

What are the trajectories in this gradient field trying to get away from?

□G.4.b.ii)

You are given a certain function f[x, y] and the information that

{a, b, f[a, b]}

sits on the bottom of a dip of the surface given by

z = f[x, y].

You center a small circle at {a, b} and plot the gradient field of f[x, y] on this circle. Is the net flow of the gradient field of f[x, y] across this circle from inside to outside or from outside to inside?

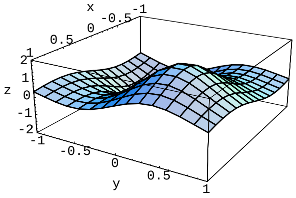
□G.4.c) Crests and dips

Here's a plot of the surface

$$z = f[x, y] = \frac{\mathrm{Sin}[2\,x] + \mathrm{Sin}[3\,y]}{0.5 + x^2 + y^2}$$

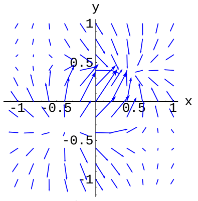
for $-1 \le x \le 1$ and $-1 \le y \le 1$:

```
Clear[f, x, y]
f[x_, y_] = (Sin[2 x] + Sin[3 y])/(0.5 + x^2 + y^2);
Plot3D[f[x, y], {x, -1, 1}, {y, -1, 1}, AxesLabel → {"x", "y", "z"},
 ViewPoint → CMView];
```



Here's a plot of the (scaled) gradient field of the same function for the same x's and the same y's:

```
Clear[gradf]
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
scalefactor = 0.07;
gradfieldplot =
 Table[Arrow[gradf[x, y], Tail → {x, y}, ScaleFactor → scalefactor],
  {x, -1, 1, 0.2}, {y, -1, 1, 0.2}];

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



What information about the crests and dips on the surface does this plot of the gradient field of f[x, y] convey?

Use the information in this plot and the FindMinimum instruction to locate the position {a, b, f[a, b]} of the top of the highest crest, and the position {c, d, f[c, d]} of the bottom of the deepest dip on the part of the surface

z = f[x, y]

plotted above.

□G.4.d) The negative gradient field

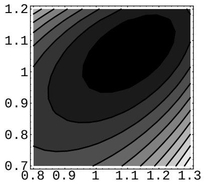
Here's a contour plot of

$$f[x, y] = 1.2\,e^{x^2} + 2.9\,e^{y^2} - 8.2\,x\,y^2.$$

for

$0.8 \le x \le 1.3$ and $0.7 \le y \le 1.2$:

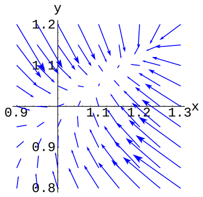
```
Clear[f, x, y]
f[x_, y_] = 1.2 E^(x^2) + 2.9 E^(y^2) - 8.2 x y^2;
ContourPlot[f[x, y], {x, 0.8, 1.3}, {y, 0.7, 1.2}];
```



There's a dip bottoming out at {a, b, f[a, b]} for some {a, b} near {1, 1}.

Look at a plot of the (scaled) negative gradient in the vicinity of {1, 1}.

```
Clear[negativegradient]
negativegradf[x_, y_] = -{D[f[x, y], x], D[f[x, y], y]};
scalefactor = 0.01;
neggradfieldplot = Table[
  Arrow[negativegradf[x, y], Tail → {x, y}, ScaleFactor → scalefactor],
  {x, 0.9, 1.3, 0.4/8}, {y, 0.8, 1.2, 0.4/8}];

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



Look at that flow.

Use what you see in the plot above, and then use the FindMinimum instruction to get a good estimate of the point {a, b, f[a, b]} at the bottom of the dip.

Add a plot of {a, b} to the plot of the negative gradient field above.

Describe what you see and say why you see it.

How do trajectories in this negative gradient field look when they pass through the part of the negative gradient field plotted above?

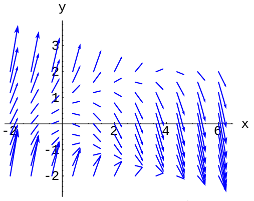
G.5) Differential equations and their associated vector fields*

□G.5.a)

Here's the vector field coming from the differential equation

$$y'[x] = y[x]^2 - x.$$

```
Clear[DEField, x, y]
DEField[x_, y_] = {1, y^2 - x};
scalefactor = 0.3;
DEFieldplot =
 Table[Arrow[DEField[x, y], Tail → {x, y}, ScaleFactor → scalefactor],
  {x, -2, 6, 8/10}, {y, -2, 2, 4/10}];

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



Look at the slopes of those field vectors.

Add plots of the solutions of

$y'[x] = y[x]^2 - x$ with $y[-1.5] = -2$

and

$y'[x] = y[x]^2 - x$ with $y[0] = -1$

as trajectories of the vector field to the DEField plot above, and discuss the relation between the plots of the solutions and the DEField.

Finally, examine the flow of this vector field, and use it to describe the behavior of solutions of the differential equation

$y'[x] = y[x]^2 - x$

that pass through the part of the xy-plane plotted above.

□Tip:

DSolve cannot find a formula for the solution of either of these differential equations:

```
Clear[x, y, Derivative]
DSolve[{y'[x] == y[x]^2 - x, y[-1.5] == -2}, y[x], x]
```

```
{{y[x] → -(1.00000000000000
      (AiryAiPrime[x] + 0.65782344695619 AiryBiPrime[x])) /
    (AiryAi[x] + 0.65782344695619 AiryBi[x])^1.00000000000000}}
```

```
Clear[x, y, Derivative]
DSolve[{y'[x] == y[x]^2 - x, y[0] == -1}, y[x], x]
```

$$\left\{\left\{y[x] \to -\frac{\mathrm{AiryAiPrime}[x] + \frac{\mathrm{AiryBiPrime}[x]\,\left(-3^{1/6}\,\mathrm{Gamma}[\frac{1}{3}] - \sqrt{3}\,\mathrm{Gamma}[\frac{2}{3}]\right)}{3^{2/3}\,\mathrm{Gamma}[\frac{1}{3}] - 3\,\mathrm{Gamma}[\frac{2}{3}]}}{\mathrm{AiryAi}[x] + \frac{\mathrm{AiryBi}[x]\,\left(-3^{1/6}\,\mathrm{Gamma}[\frac{1}{3}] - \sqrt{3}\,\mathrm{Gamma}[\frac{2}{3}]\right)}{3^{2/3}\,\mathrm{Gamma}[\frac{1}{3}] - 3\,\mathrm{Gamma}[\frac{2}{3}]}}\right\}\right\}$$

This is not a deficiency of Mathematica. No man, woman, or machine has ever been able to come up with a formula for the solution. It's an unfortunate fact of science that most differential equations have solutions for which no clean formula is available.

The consequence:

With no formula available, the best way to get an idea of how the solutions of

$$y'[x] = y[x]^2 - x$$

look is to look at the flow of the DEField coming from this differential equation.

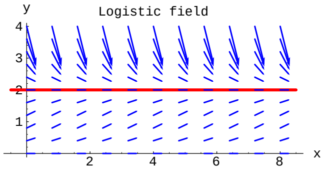
□G.5.b.i)

Here's the vector field coming from the differential equation

$$y'[x] = y[x]\,(1 - \tfrac{y[x]}{2}):$$

```
  Those who have been on the C&M bandwagon from the beginning
   will recognize this as the logistic differential equation.
```

```
Clear[DEField, x, y]
DEField[x_, y_] = {1, y (1 - y/2)};
scalefactor = 0.3;
DEFieldplot =
 Table[Arrow[DEField[x, y], Tail → {x, y}, ScaleFactor → scalefactor],
  {x, 0, 8, 8/10}, {y, 0, 4, 4/10}];
cutoff = Graphics[{Red, Thickness[0.01], Line[{{-0.5, 2}, {8.5, 2}}]}];

Show[cutoff, DEFieldplot, Axes → True, AxesLabel → {"x", "y"},
 PlotLabel → "Logistic field"];
```



Use what you see to describe the fundamental difference between the behavior of a solution of

$$y'[x] = y[x]\,(1 - \tfrac{y[x]}{2})$$

whose plot goes through a point {a, b} with b > 2
and the behavior of a solution of

$$y'[x] = y[x]\,(1 - \tfrac{y[x]}{2})$$

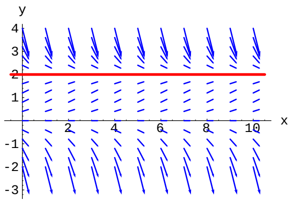
whose plot goes through a point {a, b} with 0 < b < 2.

□G.5.b.ii)

Here's more of the vector field coming from the logistic differential equation

$$y'[x] = y[x]\,(1 - \tfrac{y[x]}{2})$$

```
Clear[DEField, x, y]
DEField[x_, y_] = {1, y (1 - y/2)};
scalefactor = 0.3;
DEFieldplot =
 Table[Arrow[DEField[x, y], Tail → {x, y}, ScaleFactor → scalefactor],
  {x, 0, 10, 1}, {y, -2, 4, 4/10}];
cutoff =
 Graphics[{Red, Thickness[0.01], Line[{{-0.5, 2}, {10.5, 2}}]}];

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



Now you see three distinct types of solutions of

$$y'[x] = y[x]\,(1 - \tfrac{y[x]}{2}).$$

Describe the three distinct types of solutions in words.

□G.5.c)

Plot the vector field coming from the differential equation

$$y'[x] = x\,y[x]$$

for a healthy selection of {x, y}'s with

$$-4 \le x \le 4 \text{ and } -3 \le y \le 3.$$

Apply an appropriate scale factor so you can see what's going on.
Look at the resulting plot of the vector field, and describe how solutions with $y[-4] > 0$ are fundamentally different from solutions with $y[-4] < 0$.

G.6) Trajectories: Can they cross?

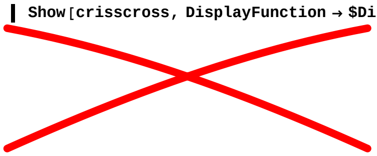
□G.6.a) Crossing of trajectories

```
   Calculus&Mathematica thanks Professor Stephanie Alexander of the
University of Illinois for helpful conversations about these problems.
```

Can two trajectories of a vector field 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];
```



Why or why not?

□Tip:

Think of it this way:

At the cross-over point, which way must the floating cork move?

□G.6.b) Crossing of plots of solutions of differential equations

You have a differential equation

$$y'[x] = f[x, y[x]].$$

If you have a formula for f[x, y] and you start with

$$y[0] = 1,$$

then you can use NDSolve to plot out a function y[x] with

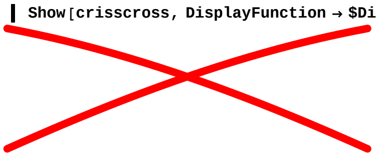
$$y[0] = 1 \text{ and } y'[x] = f[x, y[x]].$$

On the other hand if you start with y[0] = 2, then you can use NDSolve to plot out a function y[x] with

$$y[0] = 2 \text{ and } y'[x] = f[x, y[x]].$$

The question here is: Can the plots of these two solutions ever cross over each other like this:

```
Show[crisscross, DisplayFunction → $DisplayFunction];
```



Why or why not?

G.7) Drifting along with a tumbleweed

□G.7.a)

Wind is swirling through the desert region

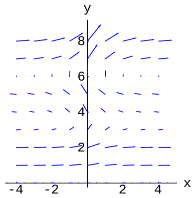
$$-4 \le x \le 4,\ 0 \le y \le 8.$$

At a point {x, y} in this region, the velocity of the wind is

$$\text{wind}[x, y] = \{0.18 + 0.6\,\text{Sin}[y],\ \tfrac{0.12\,y}{1 + x^2}\}.$$

Here is a picture of the wind as you look down from a helicopter at this desert region:

```
Clear[wind, x, y]
wind[x_, y_] = {0.18 + 0.6 Sin[y], 0.12 y/(1 + x^2)};
windplot =
 Show[Table[Arrow[wind[x, y], Tail → {x, y}], {x, -4, 4}, {y, 0, 8}],
  Axes → Automatic, AxesLabel → {"x", "y"}];
```



A tumbleweed drifting along was spotted at the point {−2, 2}.
Add a plot of the approximate path of this drifting tumbleweed to the plot above.

G.8) Logistic harvesting revisited

```
Those who have been with Calculus&Mathematica
from the beginning may recognize this problem.
```

Take the logistic equation

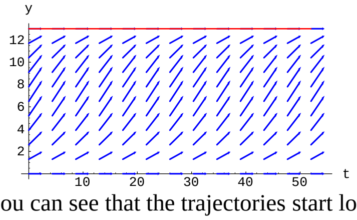
$$y'[t] = a\,y[t]\,(1 - \tfrac{y[t]}{b})$$

with 0 < a and 0 < b.
And start with 0 < y[0] < b.
As t advances from 0, you are guaranteed that y[t] grows with some pep until y[t] gets near b.
Once y[t] gets near b, then y[t] settles into global scale with

$$\lim_{t\to\infty} y[t] = b.$$

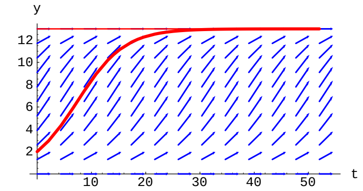
Check it out with with a = 0.23 and b = 13:

```
a = 0.23;
b = 13;
Clear[DEField, t, y]
DEField[t_, y_] = {1, a y (1 - y/b)};
scalefactor = 2.5;
DEFieldplot =
 Table[Arrow[DEField[t, y], Tail → {t, y}, ScaleFactor → scalefactor],
  {t, 0, 52, 52/12}, {y, 0, b, b/10}];
bline = Graphics[{Red, Line[{{0, b}, {52, b}}]}];

Show[DEFieldplot, bline, Axes → Automatic, AxesLabel → {"t", "y"},
 AspectRatio → 1/2];
```



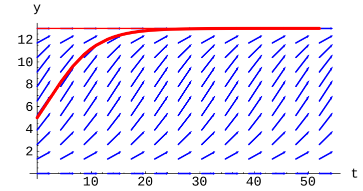
You can see that the trajectories start low on the left and are sucked onto the line y = b plotted at the very top:

```
endtime = 52;
starter = 2;
Clear[t, y, Derivative, fakey]
solution = NDSolve[
  {y′[t] == a y[t] (1 - y[t]/b), y[0] == starter}, y[t], {t, 0, endtime}];
fakey[t_] = y[t] /. solution[[1]];
solutionplot = Plot[{fakey[t]}, {t, 0, endtime},
  PlotStyle → {{Red, Thickness[0.01]}}, DisplayFunction → Identity];

Show[DEFieldplot, solutionplot, bline,
 Axes → Automatic, AxesLabel → {"t", "y"}, AspectRatio → 1/2,
 DisplayFunction → $DisplayFunction];
```



Try it again with y[0] = 5:

```
starter = 5;
Clear[t, y, Derivative, fakey]
solution = NDSolve[
  {y′[t] == a y[t] (1 - y[t]/b), y[0] == starter}, y[t], {t, 0, endtime}];
fakey[t_] = y[t] /. solution[[1]];
solutionplot = Plot[{fakey[t]}, {t, 0, endtime},
  PlotStyle → {{Red, Thickness[0.01]}}, DisplayFunction → Identity];

Show[DEFieldplot, solutionplot, bline,
 Axes → Automatic, AxesLabel → {"t", "y"}, AspectRatio → 1/2,
 DisplayFunction → $DisplayFunction];
```



You get a reasonable interpretation of the logistic differential equation

$$y'[t] = a\,y[t]\,(1 - \tfrac{y[t]}{b}) \text{ with } 0 < y[0] < b$$

by imagining that y[t] is number of catfish (in thousands) in a given lake on a catfish farm t weeks after the lake was stocked with y[0] catfish. As time goes on, the catfish population increases until it reaches its steady-state population of b catfish.
But the catfish farmer doesn't grow catfish as pets; the farmer is in business to harvest catfish and to sell them so that hungry folks can fry them up and then wash them down with a couple of cold beers or iced teas.

□G.8.a.i)

Measure time in weeks, assume that the farmer wants to harvest r fish per week, and explain why

$$y'[t] = a\,y[t]\,(1 - \tfrac{y[t]}{b}) - r$$

lays the base for a reasonable model.

□G.8.a.ii)

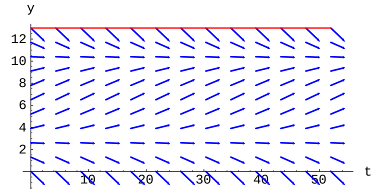
Go with the model

$$y'[t] = a\,y[t]\,(1 - \tfrac{y[t]}{b}) - r,$$

where t measures the number of weeks that harvesting goes on,
a = 0.23, b = 13 (thousand fish), and the weekly harvest is r = 0.5 (thousand fish).
Here is the associated DEField:

The reason that b = 13 is that 13000 fish is the greatest number of fish the lake can support.

```
a = 0.23;
b = 13;
r = 0.5;
Clear[DEField, t, y]
DEField[t_, y_] = {1, a y (1 - y/b) - r};
scalefactor = 2.5;
DEFieldplot =
 Table[Arrow[DEField[t, y], Tail → {t, y}, ScaleFactor → scalefactor],
  {t, 0, 52, 52/12}, {y, 0, b, b/10}];
bline = Graphics[{Red, Line[{{0, b}, {52, b}}]}];

Show[DEFieldplot, bline, Axes → Automatic, AxesLabel → {"t", "y"},
 AspectRatio → 1/2];
```

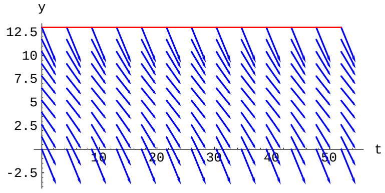


Examine the vector field and use what you see to estimate the smallest number of fish the lake must contain at the beginning (t = 0) to sustain the weekly harvest of r = 0.5 (thousand fish) for a full 52 consecutive weeks.

□G.8.a.iii)

Here's what happens with a harvest of r = 1.5 (thousand fish) per week from the same lake:

```
r = 1.5;
Clear[DEField, t, y]
DEField[t_, y_] = {1, a y (1 - y/b) - r};
scalefactor = 2.5;
DEFieldplot =
 Table[Arrow[DEField[t, y], Tail → {t, y}, ScaleFactor → scalefactor],
  {t, 0, 52, 52/12}, {y, 0, b, b/10}];
bline = Graphics[{Red, Line[{{0, b}, {52, b}}]}];

Show[DEFieldplot, bline, Axes → Automatic, AxesLabel → {"t", "y"},
 AspectRatio → 1/2];
```



Harvesting at the level r = 1.5 (thousand fish) per week is a very bad idea because all the trajectories that start on the far left are going negative before the end of the 52-week period under study. This means that before the end of the 52-week period, all the fish will be gone and harvesting will have to stop no matter how many fish you start out with.

Your job is to use DEField plots to estimate the largest possible weekly harvest rate r (in thousands) and a number

y[0] = starter

so that if the lake starts out containing starter fish,

then the harvest can continue for the full 52-week period.

Once you have your estimates for r and y[0] = starter, show the resulting DEField together with the plot of the solution of

$$y'[t] = a\, y[t]\,\left(1 - \frac{y[t]}{b}\right) - r$$

with y[0] = starter.

□Tip:

If you can't come up with a grand theory, prospect a little bit. What happens with r = 1.25? How about r = 0.95?

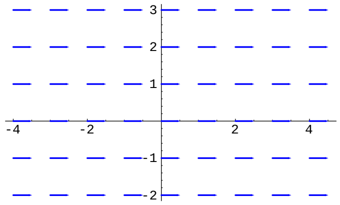
Remember you are after an estimate, not an exact number.

G.9) Water flow with spigots and drains

□G.9.a)

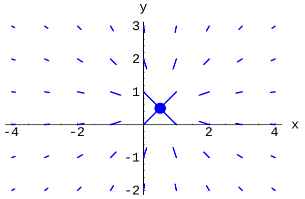
Here is water flowing from left to right:

```
Clear[currentField, x, y]
currentstrength = 0.51;
currentField[x_, y_] = currentstrength {1, 0};
currentfieldplot = Table[
  Arrow[currentField[x, y], Tail → {x, y}], {x, -4, 4}, {y, -2, 3}];

Show[currentfieldplot, Axes → Automatic];
```



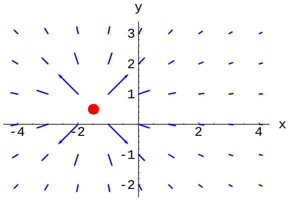
Here is water flowing into a drain at {0.5, 0.5}:

```
Clear[drainField, x, y]
{a, b} = {0.5, 0.5};
drainstrength = 0.57;
drainField[x_, y_] = - drainstrength {x - a, y - b}/((x - a)^2 + (y - b)^2);
drainfieldplot =
 Table[Arrow[drainField[x, y], Tail → {x, y}], {x, -4, 4}, {y, -2, 3}];
drain = Graphics[{Blue, PointSize[0.04], Point[{a, b}]}];

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



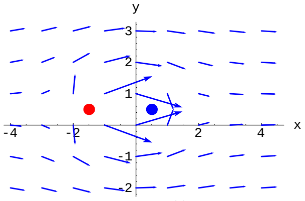
Here is water flowing out of a spigot at {−1.5, 0.5}:

```
Clear[spigotField, x, y]
{aa, bb} = {-1.5, 0.5};
spigotstrength = 0.67;
spigotField[x_, y_] = spigotstrength {x - aa, y - bb}/((x - aa)^2 + (y - bb)^2);
spigotfieldplot =
 Table[Arrow[spigotField[x, y], Tail → {x, y}], {x, -4, 4}, {y, -2, 3}];
spigot = Graphics[{Red, PointSize[0.04], Point[{aa, bb}]}];

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



Here is water flowing under the influence of all three together:

```
Clear[combinationField, x, y]
combinationField[x_, y_] =
 currentField[x, y] + drainField[x, y] + spigotField[x, y];
combinationfieldplot = Table[
  Arrow[combinationField[x, y], Tail → {x, y}], {x, -4, 4}, {y, -2, 3}];

Show[combinationfieldplot, drain, spigot,
 AxesLabel → {"x", "y"}, Axes → True];
```



Experiment with different strengths and positions of the sources and sinks and report on your observations.

What happens you make the spigot strength equal to the drain strength, and you put the drain and the spigot close together?

How big does the current strength have to be in relation to the spigot strength and the drain strength to make the influence of the drain and the spigot negligible?

G.10) 2D Electrical fields

□G.10.a)

Plot and describe the two dimensional electric field resulting from equal charges of the same sign at {1.5, 0} and {−1.5, 0}.
Include a plot of the unit vector field.

□**Tip:**

Make sure that your plot does not ask for the vectors at the points {1.5, 0} and {−1.5, 0}.

□G.10.b) Dipoles

A dipole can be approximated by two charges of the same magnitude but opposite sign separated by a small distance.
Dipoles are especially important in atomic theory.
Richard Feynman explains it this way:

"Although an atom or molecule remains neutral in an external electronic field, there is a tiny separation of its positive and negative charges, and it becomes a microscopic dipole."

From The Feynman Lectures on Physics
by Richrad P. Feynman, Robert B. Leighton, and Matthew Sands,
Addison-Wesley, 1964.

Plot and describe the electric field resulting from a positive charge at {0.01, 0} and the opposite charge of the same magnitude at {−0.01, 0}.
Include a plot of the unit vector field.

□**Tip:**

All hell breaks loose near {0, 0}, so make sure your plot does not ask for the vectors too close to {0, 0}.

G.11) Gradient fields for max-min. Hamiltonian fields for level curves, and implicitly defined functions

□G.11.a.i) Gradient systems

Given a function f[x, y], you get the gradient of f[x, y] by going with
gradf[x, y] = {∂_x f[x, y], ∂_y f[x, y]}.
This is the same as saying

$$\nabla f[x, y] = \left\{\frac{\partial f[x,y]}{\partial x}, \frac{\partial f[x,y]}{\partial y}\right\}.$$

To get the gradient field you just go with
Field[x, y] = gradf[x[t], y[t]],
which is the same as putting

$$\text{Field}[x, y] = \left\{\frac{\partial f[x,y]}{\partial x}, \frac{\partial f[x,y]}{\partial y}\right\}.$$

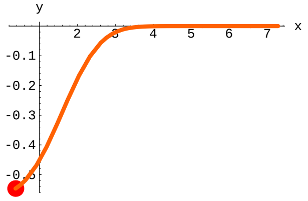
Here's the gradient field coming from
f[x, y] = x Cos[y] + y Sin[x]:

```
Clear[m, n, x, y, t, f, gradf, Field];
f[x_, y_] = x Cos[y] + 0.3 y^2 Sin[x];
gradf[x_, y_] = {∂x f[x, y], ∂y f[x, y]};
{m[x_, y_], n[x_, y_]} = gradf[x, y];

Field[x_, y_] = {m[x, y], n[x, y]}
```

```
{0.3 y^2 Cos[x] + Cos[y], 0.6 y Sin[x] - x Sin[y]}
```

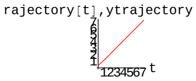
Here's a trajectory in this gradient field of f[x, y] starting at a random point:

```
{a, b} = {Random[Real, {-1, 1}], Random[Real, {-1, 1}]};
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 = 7;
approxsolutions = NDSolve[{equationx, equationy, starterx, startery},
  {x[t], y[t]}, {t, 0, endtime}];

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

Show[starterplot, trajectoryplot, PlotRange → All, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction,
 AspectRatio → 1/GoldenRatio];
```



Agree that {xtrajectory[t], ytrajectory[t]} specifies the point on this trajectory at time t.
Here's what happens to f[xtrajectory[t], ytrajectory[t]] as t advances from 0:

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

Plot[f[xtrajectory[t], ytrajectory[t]],
 {t, 0, endtime}, PlotStyle → {{Red, Thickness[0.015]}},
 AxesLabel → {"t", "f[xtrajectory[t],ytrajectory[t]]"}];
```



Going up. Use your knowledge of the fact that when you plot gradf[x, y] with tail at {x, y}, the direction of gradf[x, y] is the direction of greatest initial increase in f[x, y] to explain why this curve had no choice but to go up.
Explain this bold statement:
No matter what function f[x, y] you go with, if
{xtrajectory[t], ytrajectory[t]}
parameterizes a trajectory in the gradient field of f[x, y], then you are guaranteed that f[xtrajectory[t], ytrajectory[t]] goes up as t goes up.

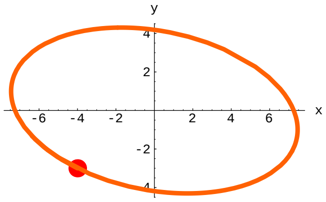
□G.11.a.ii)

Here's a new vector field:

```
Clear[m, n, x, y, t, Field];
m[x_, y_] = 0.2 x + 1.5 y;
n[x_, y_] = -0.5 x - 0.2 y;

Field[x_, y_] = {m[x, y], n[x, y]};
```

And one of its trajectories:

```
{a, b} = {-4, -3};
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;
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[starterplot, trajectoryplot, PlotRange → All, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction];
```



An elliptical closed curve.
How does the shape of this trajectory signal (with no extra calculation) that the given system is not a gradient system?

□ **Tip:**

Remember from part i) above:

No matter what function f[x, y] you go with, if

{xtrajectory[t], ytrajectory[t]}

parameterizes a trajectory in the gradient field of f[x, y], then you are guaranteed that

f[xtrajectory[t], ytrajectory[t]]

goes up as t goes up.

□G.11.b.i)

Here's the gradient of

$$f[x, y] = x^3 + E^{3y} - 3 x E^y:$$

```
Clear[f, gradf, x, y];
f[x_, y_] = x^3 + E^(3 y) - 3 x E^y;

gradf[x_, y_] = {∂x f[x, y], ∂y f[x, y]}
```

$$\{-3 E^y + 3 x^2,\ 3 E^{3y} - 3 E^y x\}$$

Here's the corresponding gradient field:

```
Clear[m, n, x, y, t, Field];
{m[x_, y_], n[x_, y_]} = gradf[x, y];

Field[x_, y_] = gradf[x, y]
```

$$\{-3 E^y + 3 x^2,\ 3 E^{3y} - 3 E^y x\}$$

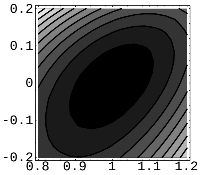
You can see that {1, 0} is a point at which gradf[x, y] = {0, 0}:

```
{xcritical, ycritical} = {1, 0};
gradf[xcritical, ycritical]
```

```
{0, 0}
```

Look at the level curves of f[x, y] in the vicinity of {xcritical, ycritical}:

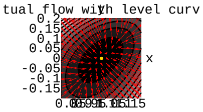
```
h = 0.2;
levelcurves = ContourPlot[Evaluate[f[x, y]],
  {x, xcritical - h, xcritical + h}, {y, ycritical - h, ycritical + h},
  ContourSmoothing → Automatic, AxesLabel → {"x", "y"}];
```



The lighter shading indicates larger values of f[x, y].

Throw in the flow of this gradient field:

```
flowplot = Table[Arrow[Field[x, y], Tail → {x, y},
    VectorColor → Red, ScaleFactor → 0.1, HeadSize → 0.02],
  {x, xcritical - h, xcritical + h, h/8},
  {y, ycritical - h, ycritical + h, h/8}];
equilibplot =
 Graphics[{PointSize[0.04], Gold, Point[{xcritical, ycritical}]}];

Show[levelcurves, flowplot, equilibplot, Axes → True, AxesOrigin →
  {xcritical, ycritical}, AxesLabel → {"x", "y"}, PlotRange →
  {{xcritical - h, xcritical + h}, {ycritical - h, ycritical + h}},
 PlotLabel → "Actual flow with level curves"];
```



What does all this graphical information give you about what happens to f[x, y] at the plotted critical point {xcritical, ycritical}?

□G.11.b.ii)

Go with this function:

```
Clear[f, gradf, x, y];
f[x_, y_] = (x + 2 y)/(1 + x^2 + y^2)
```

$$\frac{x + 2 y}{1 + x^2 + y^2}$$

Say how you know in advance that this function has a global maximizer and a global minimizer.

□G.11.b.iii)

Stay with the same function and calculate its gradient:

```
Clear[f, gradf, x, y];
f[x_, y_] = (x + 2 y)/(1 + x^2 + y^2);

gradf[x_, y_] = {∂x f[x, y], ∂y f[x, y]}
```

$$\left\{-\frac{2 x (x + 2 y)}{(1 + x^2 + y^2)^2} + \frac{1}{1 + x^2 + y^2},\ -\frac{2 y (x + 2 y)}{(1 + x^2 + y^2)^2} + \frac{2}{1 + x^2 + y^2}\right\}$$

Look at this:

```
Chop[N[Solve[gradf[x,y] == {0,0}]]]
```

```
{{x → -0.447214, y → -0.894427}, {x → 0.447214, y → 0.894427}}
```

This gives you two candidates for global maximizers and minimizers. Study in the manner of part i) each of these points in effort to determine which is the global maximizer and which is the global minimizer.
Can you come up with an alternate way of making this determination?

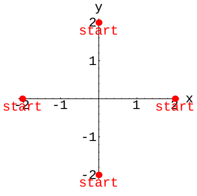
□G.11.c) Fishing for possible maximizers

Just as you are working on gradients and max-min, an old retired prof (sort of a geezer) sits down at a computer next to you and says, "I don't use Solve commands or anything like that when I am looking for maximizers in a certain region. Instead I just go fishing in the gradient flow. For instance if I am trying to find local max's near {0, 0} of

$$f[x, y] = \frac{E^{-0.3 (x-0.4)^2} \operatorname{Sin}[x y + 0.4]}{1 + y^2},$$

I throw four corks in like this:

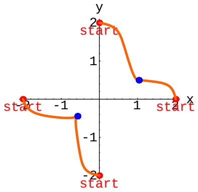
```
Clear[xstarter, ystarter, k];
h = 2;
{xstarter[1], ystarter[1]} = {h, 0};
{xstarter[2], ystarter[2]} = {0, h};
{xstarter[3], ystarter[3]} = {-h, 0};
{xstarter[4], ystarter[4]} = {0, -h};
starterplots = Table[Graphics[{Red, PointSize[0.04],
    Point[{xstarter[k], ystarter[k]}]}], {k, 1, 4}];
starterlabels = Table[Graphics[{Red,
    Text["start", {xstarter[k], ystarter[k]}, {0, 1}]}], {k, 1, 4}];

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



"Then I enter the function and plot the trajectories in the gradient field that start at the plotted points and then I see where they go:"

```
Clear[f, gradf, m, n, x, y, t];
f[x_, y_] = E^(-0.3 (x-0.4)^2) Sin[x y + 0.4]/(1 + y^2);
gradf[x_, y_] = {∂x f[x, y], ∂y f[x, y]};
{m[x_, y_], n[x_, y_]} = gradf[x, y];
```

```
equationx = x′[t] == m[x[t], y[t]];
equationy = y′[t] == n[x[t], y[t]];
endtime = 25;

ndssol1 = NDSolve[{equationx, equationy, x[0] == h, y[0] == 0},
  {x[t], y[t]}, {t, 0, endtime}];
ndssol2 = NDSolve[{equationx, equationy, x[0] == 0, y[0] == h},
  {x[t], y[t]}, {t, 0, endtime}];
ndssol3 = NDSolve[{equationx, equationy, x[0] == -h, y[0] == 0},
  {x[t], y[t]}, {t, 0, endtime}];
ndssol4 = NDSolve[{equationx, equationy, x[0] == 0, y[0] == -h},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[trajectory1, trajectory2, trajectory3, trajectory4];
trajectory1[t_] = {x[t], y[t]} /. ndssol1[[1]];
trajectory2[t_] = {x[t], y[t]} /. ndssol2[[1]];
trajectory3[t_] = {x[t], y[t]} /. ndssol3[[1]];
trajectory4[t_] = {x[t], y[t]} /. ndssol4[[1]];
trajectoryplots = ParametricPlot[
  {trajectory1[t], trajectory2[t], trajectory3[t], trajectory4[t]},
  {t, 0, endtime}, PlotStyle → {{CadmiumOrange, Thickness[0.015]}},
  DisplayFunction → Identity];
endplots =
 {Graphics[{Blue, PointSize[0.04], Point[trajectory1[endtime]]}],
  Graphics[{Blue, PointSize[0.04], Point[trajectory2[endtime]]}],
  Graphics[{Blue, PointSize[0.04], Point[trajectory3[endtime]]}],
  Graphics[{Blue, PointSize[0.04], Point[trajectory4[endtime]]}]};

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



The old guy goes on to say, "Those blue dots at trajectory1[endtime], trajectory2[endtime], trajectory3[endtime] and trajectory4[endtime] are the points at which the trajectories stalled. So, goldurn it, I'm pretty well convinced that these are worth investigating as maximizers.

Is this guy still with it?
Does his idea have any merit?.
If so, explain why.

□G.11.d.i) Hamiltonian systems

Given a function f[x, y], you get the gradient system coming from f[x, y] by going with

$m[x, y] = \partial_x f[x, y]$ (which is the same as $\frac{\partial f[x,y]}{\partial x}$)

$n[x, y] = \partial_y f[x, y]$ (which is the same as $\frac{\partial f[x,y]}{\partial y}$)

```
If you're running this lesson from Windows, the funny characters above
                are partial derivative symbols.
```

and putting

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

Given a function f[x, y], you get the Hamiltonian system coming from f[x, y] by going with

$m[x, y] = -\partial_y f[x, y]$ (which is the same as $-\frac{\partial f[x,y]}{\partial y}$)

$n[x, y] = \partial_x f[x, y]$ (which is the same as $\frac{\partial f[x,y]}{\partial x}$)

and putting

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

Here's the Hamiltonian system coming from

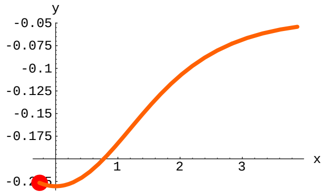
$f[x, y] = y^2 - y\,\text{Sin}[\frac{x}{3}]^2$:

```
Clear[m, n, x, y, t, f];
f[x_, y_] = y^2 - y Sin[x/3]^2;

{m[x_, y_], n[x_, y_]} = {-∂yf [x, y], ∂xf [x, y]};
hamiltoniansystem =
 ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});

ColumnForm[Thread[hamiltoniansystem]]
```

$$x'[t] == \text{Sin}[\frac{x[t]}{3}]^2 - 2\,y[t]$$
$$y'[t] == -\frac{2}{3}\,\text{Cos}[\frac{x[t]}{3}]\,\text{Sin}[\frac{x[t]}{3}]\,y[t]$$

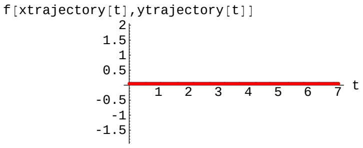
Here's a random trajectory in this Hamiltonian flow:

```
{a, b} = {Random[Real, {-2, 2}], Random[Real, {-2, 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 = 7;

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[starterplot, trajectoryplot, PlotRange → All, Axes → True,
 AxesLabel → {"x", "y"}, DisplayFunction → $DisplayFunction,
 AspectRatio → 1/GoldenRatio];
```



Agree that {xtrajectory[t], ytrajectory[t]} specifies the point on this trajectory at time t.
Here's what happens to f[xtrajectory[t], ytrajectory[t]]:

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

Plot[f[xtrajectory[t], ytrajectory[t]],
 {t, 0, endtime}, PlotStyle → {{Red, Thickness[0.015]}},
 AxesLabel → {"t", "f[xtrajectory[t],ytrajectory[t]]"},
 PlotRange → {f[a, b] - 2, f[a, b] + 2}];
```



Got any idea why that happened?

□G.11.d.ii)

Here's the Hamiltonian system coming from

$f[x, y] = x^2 - 5\,\text{Sin}[x]\,y + 4\,y^2$:

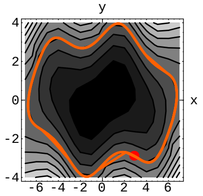
```
Clear[f, gradf, m, n, x, y];
f[x_, y_] = x^2 - 5 Sin[x] y + 4 y^2;
{m[x_, y_], n[x_, y_]} = {-∂yf [x, y], ∂xf [x, y]};
hamiltoniansystem =
 ({x'[t], y'[t]} == {m[x[t], y[t]], n[x[t], y[t]]});

ColumnForm[Thread[hamiltoniansystem]]
```

```
x′[t] == 5 Sin[x[t]] - 8 y[t]
y′[t] == 2 x[t] - 5 Cos[x[t]] y[t]
```

Here's a trajectory for this system shown with a contour plot of some of the level curves of f[x, y]:

```
{a, b} = {Random[Real, {-5, 5}], Random[Real, {-3, 3}]};
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 = 5;

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]}];
levelcurves = ContourPlot[Evaluate[f[x, y]], {x, -7, 7}, {y, -4, 4},
  ContourSmoothing → Automatic, DisplayFunction → Identity];

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



Rerun several times

Give your opinion about why each plot turns out the way it does.

□G.11.d.iii) Defining y in terms of x implicitly via f[x,y] = f[a,b]

Go with:

```
Clear[f, x, y];
f[x_, y_] = -2.3 Sin[y] + 0.9 Cos[3.0 x]
0.9 Cos[3. x] - 2.3 Sin[y]
```

When you set

f[x, y] = f[0, 0]

and vary x, then y also has to vary to maintain the equality

f[x, y] = f[0, 0].

So, when you set

f[x, y] = f[0, 0],

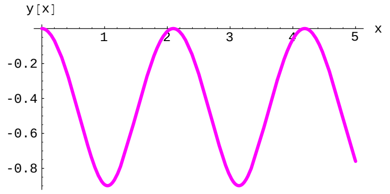
you are implicitly defining y in terms of of x.

Mathematica can give you the formula for y in terms of x:

```
sols = Solve[f[x, y] == f[0, 0], y];
y[x_] = N[y /. sols[[1]], 6]
1. ArcSin[0.0434783 (-9. + 9. Cos[3. x])]
```

And you can plot at least part of y[x] as a function of x:

```
ParametricPlot[{x, y[x]}, {x, 0, 5},
 PlotStyle → {{Magenta, Thickness[0.01]}}, AxesLabel → {"x", "y[x]"},
 AspectRatio → 1/2];
```

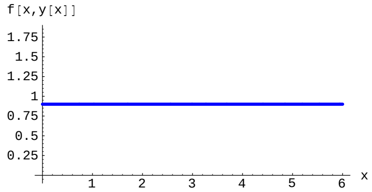


At every point {x, y} on this curve, you are guaranteed that

f[x, y] = f[0, 0]:

```
x = Random[Real, {0, 5}];
f[x, y[x]] == f[0, 0]
True
```

```
Plot[f[x, y[x]], {x, 0, 6}, PlotStyle → {{Blue, Thickness[0.01]}},
 AxesLabel → {"x", "f[x,y[x]]"}, AspectRatio → 1/2,
 PlotRange → {f[0, 0] - 1, f[0, 0] + 1}];
```



Now go with:

```
Clear[f, x, y];
f[x_, y_] = -2.3 Sin[y] + E^Cos[y] Cos[3.0 x]
E^Cos[y] Cos[3. x] - 2.3 Sin[y]
```

When you set

f[x, y] = f[0, 0],

you are implicitly defining y in terms of x. But when you go to *Mathematica* for a formula for y in terms of x, you get:

```
sols = Solve[f[x,y] == f[0,0],y]
Solve[E^Cos[y] Cos[3. x] - 2.3 Sin[y] == E, y]
```

No dice. No formula is available.

In spite of this apparent setback, you can use a trajectory plot in an appropriate Hamiltonian system to come up with a nice plot of y in terms of x through the implicit definition

f[x, y] = f[0, 0].

Do it.