2D and 3D Measurements

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

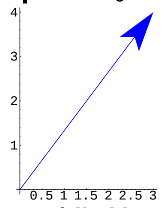
VC.01 Vectors Point the Way
Basics

B.1) Vectors:

How you move them, how you add them, how you subtract them, and how you multiply them by numbers

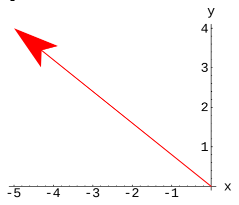
A vector X in the plane is a stick with:
→ one end (its tail) at {0, 0} and
→ the other end (its tip) at a specified location {x[1], x[2]}.
Most folks like to draw a vector as an arrow with the arrowhead at its tip.
Here's a look at the vector running from {0, 0} to {3, 4}:

```
X = {3, 4};
Show[Arrow[X, Tail → {0, 0}, VectorColor → Blue], Axes → True,
 PlotRange → All, AxesOrigin → {0, 0}];
```



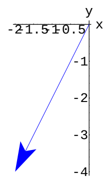
Most folks like to see the arrowhead at the tip so they can tell which end of the stick is the tip. Here is the vector running from
{0, 0} (tail) to {−5, 4} (tip):

```
X = {-5, 4};
Show[Arrow[X, VectorColor → Red], Axes → True, PlotRange → All,
 AxesLabel → {"x", "y"}, AxesOrigin → {0, 0}];
```

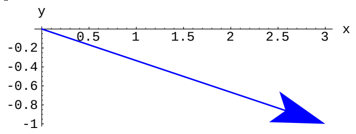


Here are some others:

```
X = {-2, -4};
Show[Arrow[X], Axes → True, PlotRange → All, AxesLabel → {"x", "y"},
 AxesOrigin → {0, 0}];
```

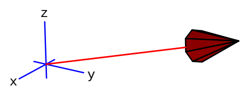


```
X = {3, -1};
Show[Arrow[X], Axes → True, PlotRange → All, AxesLabel → {"x", "y"},
 AxesOrigin → {0, 0}];
```



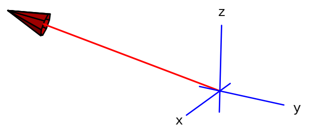
Play with these by rerunning with vectors of your own choice.
A vector
X = {x[1], x[2]}
is written the same as the coordinates of its tip because everyone knows its tail is at {0, 0}.
You can also work with three-dimensional vectors by writing
X = {x[1], x[2], x[3]}
for the vector whose tail is at {0, 0, 0} and whose tip is at
{x[1], x[2], x[3]}.
Here's the vector {1.5, 5.3, 2.0} shown along with the three-dimensional coordinate axes:

```
X = {1.5, 5.3, 2.0};
spacer = 0.2;
h = 1;

threedims = Axes3D[h, spacer];

Show[Arrow[X, VectorColor → Red], threedims, PlotRange → All,
 ViewPoint → CMView, Boxed → False];
```



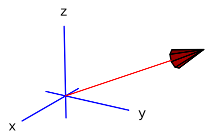
Here's another:

```
X = {1, -3, 1};
Show[Arrow[X, VectorColor → Red], threedims, PlotRange → All,
 ViewPoint → CMView, Boxed → False];
```



Here's a vector parallel to the yz-plane:

```
X = {0, 2, 1};
Show[Arrow[X, VectorColor → Red], threedims, PlotRange → All,
 ViewPoint → CMView, Boxed → False];
```



Play by rerunning with three-dimensional vectors of your own choice.

□B.1.a.i) Moving vectors

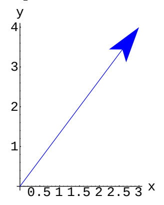
How do you move vectors to new positions?

□Answer:

Very easily.

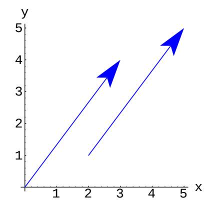
Here is a vector in two dimensions with its tail at {0, 0}:

```
X = {3, 4};
Show[Arrow[X, Tail → {0, 0}], AxesLabel → {"x", "y"}, Axes → True,
 AxesOrigin → {0, 0}];
```



Here's the same vector X shown twice, once with its tail at {0, 0} and once with its tail at {2, 1}:

```
X = {3, 4};
Show[Arrow[X, Tail → {0, 0}], Arrow[X, Tail → {2, 1}],
 Axes → True, AxesLabel → {"x", "y"}, PlotRange → All,
 AxesOrigin → {0, 0}];
```

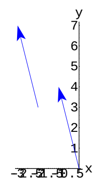


The two arrows have equal lengths and they both point in the same direction.

They're parallel.

Try it again for a different vector and a different tail:

```
X = {-1, 4};
Show[Arrow[X, Tail → {0, 0}], Arrow[X, Tail → {-2, 3}],
 Axes → True, AxesLabel → {"x", "y"}, PlotRange → All,
 AxesOrigin → {0, 0}];
```

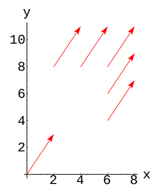


Same length; same direction.

Here's a vector X shown with its tail at {0, 0}, and shown with a whole squadron of its transplants:

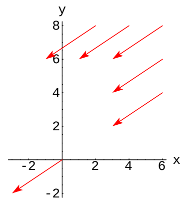
```
X = {2, 3};
Clear[tail, k];
tail[1] = {0, 0};
tail[2] = {6, 4};
tail[3] = {6, 6};
tail[4] = {6, 8};
tail[5] = {4, 8};
tail[6] = {2, 8};
```

```
squadron =
 Table[Arrow[X, Tail → tail[k], VectorColor → Red], {k, 1, 6}];

Show[squadron, Axes → True, AxesLabel → {"x", "y"}, PlotRange → All,
 AxesOrigin → {0, 0}];
```



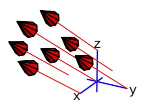
Check it out for a different X:

```
X = {-3, -2};
Clear[tail, k];
tail[1] = {0, 0};
tail[2] = {6, 4};
tail[3] = {6, 6};
tail[4] = {6, 8};
tail[5] = {4, 8};
tail[6] = {2, 8};

squadron =
 Table[Arrow[X, Tail → tail[k], VectorColor → Red], {k, 1, 6}];

Show[squadron, Axes → True, AxesLabel → {"x", "y"}, PlotRange → All,
 AxesOrigin → {0, 0}];
```



Check it out in three dimensions:

```
X = {3, 0, 2};
Clear[tail, k];
tail[1] = {0, 0, 0};
tail[2] = {1, 0, 0};
tail[3] = {-1, 0, 0};
tail[4] = {0, 1, 0};
tail[5] = {0, -1, 0};
tail[6] = {0, 0, 1};
tail[7] = {1, 0, 1};

squadron =
 Table[Arrow[X, Tail → tail[k], VectorColor → Red], {k, 1, 7}];

Show[squadron, threedims, PlotRange → All, ViewPoint → CMView,
 Boxed → False];
```



Same length; same direction.

□B.1.a.ii)

Are there any rules about moving vectors to new positions?

□Answer:

You can put the tail anywhere you like, but you must be careful not to change the direction or the length of the vector.

□B.1.b) Adding vectors

How do you add vectors?

□Answer:

Very easily.

For instance, if X = {3, 8} and Y = {5, 4} are vectors, then you add them to get

X + Y = {3, 8} + {5, 4} = {8, 12}

just by adding the corresponding components.

Mathematica can do this too:

```
X = {3, 8};
Y = {5, 4};
X + Y
{8, 12}
```

You can add three-dimensional vectors:

```
X = {3, -5, 2};
Y = {3, 4, -7};
X + Y
{6, -1, -5}
```

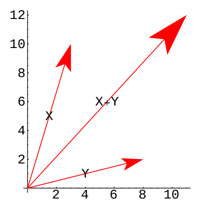
You cannot add vectors from different dimensions:

```
X = {3, -5};
Y = {3, 4, -7};
X + Y
Thread::tdlen : Objects of unequal length in {3, -5} + {3, 4, -7} cannot be combined.
{3, -5} + {3, 4, -7}
```

Here is a way of seeing what's happening in two dimensions:

Look at a picture of X = {3, 10} and Y = {8, 2} and X + Y:

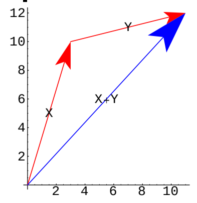
```
X = {3, 10};
Y = {8, 2};
Show[
 Arrow[X, Tail → {0, 0}, VectorColor → Red], Graphics[Text["X", X/2]],
 Arrow[Y, Tail → {0, 0}, VectorColor → Red], Graphics[Text["Y", Y/2]],
 Arrow[X + Y, Tail → {0, 0}, VectorColor → Red],
 Graphics[Text["X+Y", (X + Y)/2]], Axes → Automatic];
```



X + Y represents the combined push of X and Y.

See what happens when you move Y without changing its direction, so that its tail is at the tip of X:

```
Show[
 Arrow[X, Tail → {0, 0}, VectorColor → Red], Graphics[Text["X", X/2]],
 Arrow[Y, Tail → X, VectorColor → Red], Graphics[Text["Y", X + Y/2]],
 Arrow[X + Y, Tail → {0, 0}, VectorColor → Blue],
 Graphics[Text["X+Y", (X + Y)/2]], Axes → Automatic];
```



A triangle!

This triangle shows that you have your choice of ways of getting from {0, 0} to the tip of X + Y.

→ Route 1: You can walk directly on the vector X + Y from its tail to its tip.

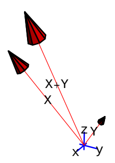
→ Route 2: You can walk on X to the tip of X, and then hook the tail of Y on the tip of X, and finish the trip by walking along Y to its tip.

Lazy folks usually take Route 1.

Rerun this for two-dimensional vectors of your own choice.

Now check out what happens in three dimensions:

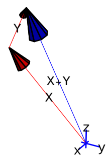
```
X = {3, -4, 7};
Y = {1, 2, 3};

Show[Arrow[X, Tail → {0, 0, 0}, VectorColor → Red],
 Graphics3D[Text["X", X/2]], Arrow[Y,
  Tail → {0, 0, 0}, VectorColor → Red], Graphics3D[Text["Y", Y/2]],
 Arrow[X + Y, Tail → {0, 0, 0}, VectorColor → Red],
 Graphics3D[Text["X+Y", (X + Y)/2]], threedims, PlotRange → All,
 ViewPoint → CMView, Boxed → False];
```



Again, in three dimensions, X + Y represents the combined push of X and Y.

See what happens when you move Y without changing its direction so that its tail is at the tip of X:

```
Show[Arrow[X, Tail → {0, 0, 0}, VectorColor → Red],
 Graphics3D[Text["X", X/2]],
 Arrow[Y, Tail → X, VectorColor → Red], Graphics3D[Text["Y", X + Y/2]],
 Arrow[X + Y, Tail → {0, 0, 0}, VectorColor → Blue],
 Graphics3D[Text["X+Y", (X + Y)/2]], threedims, PlotRange → All,
 ViewPoint → CMView, Boxed → False];
```



Another triangle.

Just as in two dimensions, this triangle shows that you have your choice of ways of getting from {0, 0} to the tip of X + Y.

→ Route 1: You can walk directly on the vector X + Y from its tail to its tip.

→ Route 2: You can walk on X to the tip of X, and then hook the tail of Y on the tip of X, and finish the trip by walking along Y to its tip.

Rerun the pictures above for different vectors X and Y until you get the hang of vector addition.

□B.1.c)

How do you subtract vectors?

□Answer:

With no trouble.

For instance, if X = {8, 2} and Y = {3, 4} are vectors, then you subtract Y from X to get

$$X - Y = \{8, 2\} - \{3, 4\} = \{5, -2\}.$$

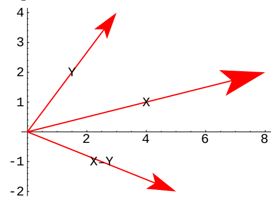
Mathematica can do this too:

```
X = {8, 2};
Y = {3, 4};

X - Y
{5, -2}
```

Here is a way of seeing what's happening.

Look at a picture of X = {8, 2}, Y = {3, 4}, and X − Y:

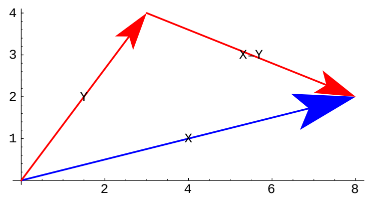
```
X = {8, 2};
Y = {3, 4};

Show[
 Arrow[X, Tail → {0, 0}, VectorColor → Red], Graphics[Text["X", X/2]],
 Arrow[Y, Tail → {0, 0}, VectorColor → Red], Graphics[Text["Y", Y/2]],
 Arrow[X - Y, Tail → {0, 0}, VectorColor → Red],
 Graphics[Text["X-Y", (X - Y)/2]], Axes → Automatic];
```



This time X is the combined push of Y and X − Y.

See what happens when you move X − Y without changing its direction so that its tail is at the tip of Y:

```
Show[Arrow[X, Tail → {0, 0}, VectorColor → Blue],
 Graphics[Text["X", X/2]], Arrow[Y, Tail → {0, 0}, VectorColor → Red],
 Graphics[Text["Y", Y/2]], Arrow[X - Y, Tail → Y, VectorColor → Red],
 Graphics[Text["X-Y", Y + (X - Y)/2]], Axes → Automatic];
```



Another triangle.

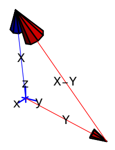
You can get to the tip of X by going directly along X, or you can go from the tail of Y to the tip of Y, and then ride on X − Y to the tip of X.

This is not a surprise because X = (X − Y) + Y.

Rerun for some other vectors of your own choice.

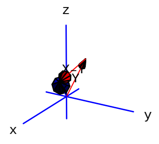
Here it is in three dimensions:

```
X = {4, 2, 8};
Y = {-2, 6, -3};

spacer = 0.2;
h = 1;
threedims = Axes3D[h, spacer];
CMView = {2.7, 1.6, 1.2};

Show[Arrow[X, Tail → {0, 0, 0}, VectorColor → Blue],
 Graphics3D[Text["X", X/2]],
 Arrow[Y, Tail → {0, 0, 0}, VectorColor → Red],
 Graphics3D[Text["Y", Y/2]],
 Arrow[X - Y, Tail → Y, VectorColor → Red],
 Graphics3D[Text["X-Y", Y + (X - Y)/2]],
 threedims, PlotRange → All, ViewPoint → CMView, Boxed → False];
```



Here it is for two random vectors in three dimensions:

```
X = 2 {Random[], Random[], Random[]};
Y = 2 {Random[], Random[], Random[]};

spacer = 0.2;
h = 1;
threedims = Axes3D[h, spacer];
CMView = {2.7, 1.6, 1.2};

Show[Arrow[X, Tail → {0, 0, 0}, VectorColor → Blue],
 Graphics3D[Text["X", X/2]],
 Arrow[Y, Tail → {0, 0, 0}, VectorColor → Red],
 Graphics3D[Text["Y", Y/2]],
 Arrow[X - Y, Tail → Y, VectorColor → Red],
 Graphics3D[Text["X-Y", Y + (X - Y)/2]],
 threedims, PlotRange → All, ViewPoint → CMView, Boxed → False];
```



Rerun for other X and Y of your own choice until you get the picture down pat.

□**B.1.d)**

How do you multiply vectors by numbers?

□**Answer:**

You just do it.

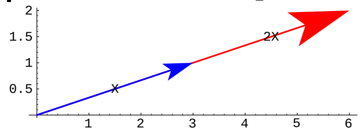
For instance, if X = {3, 1} , then 2 X = {6, 2}.

Mathematica can do this too:

```
X = {3, 1};
2 X
{6, 2}
```

Here are X and 2 X both shown with their tails at {0, 0}:

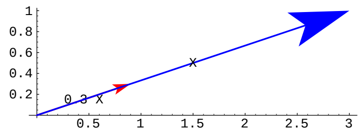
```
X = {3, 1};
Show[Arrow[2 X, Tail → {0, 0}, VectorColor → Red],
     Graphics[Text["2X", 3 X/2]],
     Arrow[X, Tail → {0, 0}, VectorColor → Blue],
     Graphics[Text["X", X/2]], Axes → Automatic];
```



2 X points the same direction as X, but 2 X is twice as long as X. That makes some sense because 2 X is the same thing as X + X.

Here are X and 0.3 X both shown with their tails at {0, 0}:

```
Show[Arrow[0.3 X, Tail → {0, 0}, VectorColor → Red],
     Graphics[Text["0.3 X", 0.3 X/2]],
    Arrow[X, Tail → {0, 0}, VectorColor → Blue],
    Graphics[Text["X", X/2]], Axes → Automatic];
```



0.3 X points in the same direction as X points.

The only difference is that the length of 0.3 X is 0.3 times the length of X.

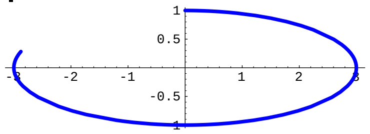
The same idea carries over to three dimensions as well.

B.2) Tangent vectors, velocity vectors, and tangent lines

□B.2.a.i) Tangent and velocity vectors

Here's a curve in two dimensions:

```
Clear[x, y, t, P];
x[t_] = 3 Sin[t];
y[t_] = Cos[t];
P[t_] = {x[t], y[t]};

curveplot = ParametricPlot[{P[t]}, {t, 0, 5}, AspectRatio → Automatic,
  PlotStyle → {{Thickness[0.01], Blue}}];
```

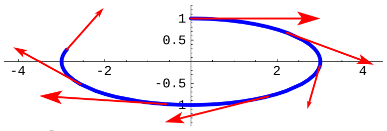


Here's what you get when you add plots of the vectors

P′[t] = {x′[t], y′[t]}

with their tails at P [t] for some choices of t:

```
hotplot = Show[curveplot,
  Table[Arrow[P′[t], Tail → P[t], VectorColor → Red],
   {t, 0, 5, 5/6}]];
```



Hot plot.
Describe what you see in terms of tangent vectors and velocity vectors.

□**Answer:**

The vectors you see are

P′[t] = {x′[t], y′[t]}

plotted with their tails at

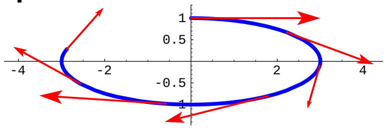
P [t] = {x[t], y[t]}

for selected t's. The vectors are tangent to the curve.
If you imagine t to be time, and you agree that you are at

P [t] = {x[t], y[t]}

at time t, then the vector P′[t] measures your velocity at time t.
Take another look at the plot:

```
Show[hotplot];
```



The direction of P′[t] measures the instantaneous direction you are moving at time t.
The length of P′[t] measures your instantaneous speed at time t. The plot above shows that you start at the top at time t = 0, moving rather quickly to the right. As you enter the turn on the right, you slow down until the curve flattens out. Then you speed up, slowing down as you go into the turn on the left.

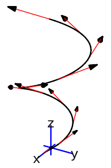
□B.2.a.ii)

Does this good stuff work in three dimensions as well?

□**Answer:**

Try it and see:

```
Clear[x, y, z, t, P];
x[t_] = √(t + 1) Cos[t];
y[t_] = √(t + 1) Sin[t];
z[t_] = t;
P[t_] = {x[t], y[t], z[t]};

curveplot =
 ParametricPlot3D[P[t], {t, 0, 10}, DisplayFunction → Identity];

tangentvectors =
 Table[Arrow[P′[t], Tail → P[t], VectorColor → Red], {t, 0, 10}];

threedims = Axes3D[2, 0.2];

Show[threedims, curveplot, tangentvectors, ViewPoint → CMView,
 Boxed → False, DisplayFunction → $DisplayFunction];
```

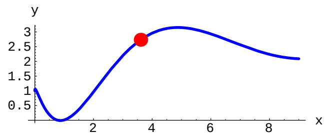


Sure it does.

□B.2.b.i) Tangent lines

Here's a curve shown with a certain point on the curve:

```
Clear[x, y, t, P];
x[t_] = t^2;
y[t_] = t + Cos[3 t];
P[t_] = {x[t], y[t]};

curveplot = ParametricPlot[{P[t]}, {t, 0, 3},
  PlotStyle → {{Thickness[0.01], Blue}}, AxesLabel → {"x", "y"},
  Epilog → {PointSize[0.05], Red, Point[P[1.9]]}];
```



The special point is:

```
P[1.9]
```

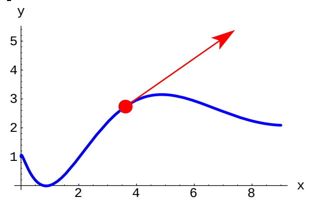
```
{3.61, 2.73471}
```

Plot the line tangent to this curve at this special point.

□**Answer:**

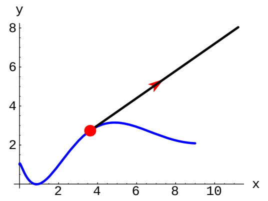
As a first step, show the plot above along with the tangent vector at P [1.9]:

```
Show[curveplot, Arrow[P′[1.9], Tail → P[1.9], VectorColor → Red]];
```



The line tangent to the curve at P [1.9] runs right through the shaft of the arrow. Here's how you can use the tangent vector P′[1.9] to come up with a plot of the tangent line.

```
Clear[tanline, s]
tanline[s_] = P[1.9] + s P′[1.9];

tanlineplot = ParametricPlot[tanline[s], {s, 0, 2},
  PlotStyle → Thickness[0.01], DisplayFunction → Identity];

Show[curveplot, Arrow[P′[1.9], Tail → P[1.9], VectorColor → Red],
 tanlineplot];
```

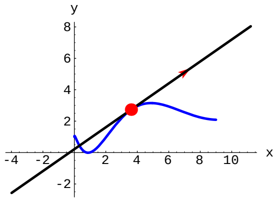


That's only part of the tangent line; you got it by plotting

P [1.9] + s P′[1.9]

with s running from 0 to 2. To get the stuff on the left, run s through some negative numbers as well as positive numbers:

```
Clear[tanline, s]
tanline[s_] = P[1.9] + s P′[1.9];

tanlineplot = ParametricPlot[tanline[s], {s, -2, 2},
  PlotStyle → Thickness[0.01], DisplayFunction → Identity];

Show[curveplot, Arrow[P′[1.9], Tail → P[1.9], VectorColor → Red],
 tanlineplot];
```



There's that tangent line in all its glory.

The upshot of all this is:

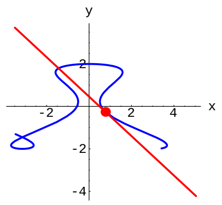
If you have a curve described by the plot of P [t] for a ≤ t ≤ b and you want to plot the tangent line through a point P [c] on the curve, you plot

P [c] + s P′[c]

and run s from negative to positive.

Here's how it goes for a different curve:

```
Clear[x, y, s, t, tanline, P];
a = -4;
b = 3;
c = 1.7;

x[t_] = t + Sin[3 t];
y[t_] = 2 Cos[t];
P[t_] = {x[t], y[t]};

tanline[s_] = P[c] + s P′[c];

curveplot = ParametricPlot[P[t], {t, a, b},
  PlotStyle → {{Thickness[0.01], Blue}}, DisplayFunction → Identity];

tanlineplot = ParametricPlot[tanline[s], {s, -2, 2},
  PlotStyle → {{Thickness[0.01], Red}}, DisplayFunction → Identity];

pointplot = Graphics[{PointSize[0.05], Red, Point[P[c]]}];

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



Play with this.

□B.2.b.ii)

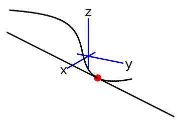
Does this work in three dimensions too?

□Answer:

Get a life!

Of course it works in three dimensions. Take a look:

```
Clear[x, y, s, t, tanline, P];
a = 0;
b = 2;
c = 0.5;

x[t_] = t + Sin[3 t] + 1;
y[t_] = 2 Cos[t];
z[t_] = t;
P[t_] = {x[t], y[t], z[t]};

tanline[s_] = P[c] + s P′[c];

curveplot =
 ParametricPlot3D[P[t], {t, a, b}, DisplayFunction → Identity];

tanlineplot = ParametricPlot3D[
  tanline[s], {s, -1.5, 1.5}, DisplayFunction → Identity];

pointplot = Graphics3D[{PointSize[0.03], Red, Point[P[c]]}];

threedims = Axes3D[1.2, 0.2];

Show[threedims, curveplot, tanlineplot, pointplot, ViewPoint → CMView,
 Boxed → False, PlotRange → All, DisplayFunction → $DisplayFunction];
```



Hard to resist.

B.3) Length of a vector, dot product, and distance between two points

□B.3.a) Dot products

Here are two cleared vectors

X = {x[1], x[2]} and Y = {y[1], y[2]}

in two dimensions:

```
Clear[X, Y, x, y]
X = {x[1], x[2]}
{x[1], x[2]}
Y = {y[1], y[2]}
{y[1], y[2]}
```

Here is the dot product, X . Y, of these vectors:

```
X.Y
x[1] y[1] + x[2] y[2]
```

Here are two vectors

X = {x[1], x[2], x[3]} and Y = {y[1], y[2], y[3]}

in three dimensions:

```
Clear[X, Y, x, y]
X = {x[1], x[2], x[3]}
{x[1], x[2], x[3]}
Y = {y[1], y[2], y[3]}
{y[1], y[2], y[3]}
```

Here is the dot product, X . Y, of the three-dimensional vectors X and Y:

```
X.Y
x[1] y[1] + x[2] y[2] + x[3] y[3]
```

Describe how dot products are calculated.

□Answer:

In two or three dimensions, the dot product X . Y just multiplies each slot in X by the corresponding slot in Y, and then adds them up.

Check it out:

```
{1, 2}.{0, 1}
2
{1, 2}.{0, 1} == 1 0 + 2 1
True
{-1, 1}.{8, 2}
-6
{-1, 1}.{8, 2} == -8 + 2
True
{1, 0, 1}.{1, 2, 3}
4
{1, 0, 1}.{1, 2, 3} == 1 + 0 + 3
True
```

You cannot take the dot product of vectors from different dimensions:

```
X = {1, 0};
Y = {2, 1, 4};
X.Y
Dot::dotsh : Tensors {1, 0} and {2, 1, 4} have incompatible shapes.
{1, 0}.{2, 1, 4}
```

That hacked off Mathematica.

□B.3.b) Dot product and length

Given a two-dimensional vector,

X = {x[1], x[2]},

the length of X is measured by

$$\|X\| = \sqrt{x[1]^2 + x[2]^2}.$$

Given a three-dimensional vector,

X = {x[1], x[2], x[3]},

the length of X is measured by

$$\|X\| = \sqrt{x[1]^2 + x[2]^2 + x[3]^2}.$$

Explain why the formula

$$\|X\| = \sqrt{X.X}$$

works in either dimension.

□Answer:

Try it out in two dimensions:

```
Clear[x]
X = {x[1], x[2]};
length = √(x[1]² + x[2]²)
```

$\sqrt{x[1]^2 + x[2]^2}$

$\sqrt{X.X}$ is given by:

```
√(X.X)
```

$\sqrt{x[1]^2 + x[2]^2}$

This tells you that the formula

$||X|| = \sqrt{X.X}$ works in two dimensions.

Try it out in three dimensions:

```
Clear[x]
X = {x[1], x[2], x[3]};
length = √(x[1]² + x[2]² + x[3]²)
```

$\sqrt{x[1]^2 + x[2]^2 + x[3]^2}$

$\sqrt{X.X}$ is given by:

```
√(X.X)
```

$\sqrt{x[1]^2 + x[2]^2 + x[3]^2}$

This tells you that the formula

$||X|| = \sqrt{X.X}$ works in three dimensions too.

Handy little formula

And it's nothing more or less than your old friend, the Pythagorean theorem, in action.

□B.3.c.i) Dot products and distance between points

Why does

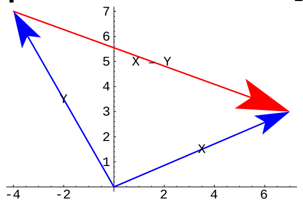
$||X - Y|| = \sqrt{(X - Y).(X - Y)}$

calculate the distance between the tip of X and the tip of Y when X and Y are positioned so that their tails are at the origin?

□**Answer:**

Look at a picture:

```
X = {7, 3};
Y = {-4, 7};

Show[Arrow[X, Tail → {0, 0}, VectorColor → Blue],
  Graphics[Text["X", X/2]],
  Arrow[Y, Tail → {0, 0}, VectorColor → Blue],
  Graphics[Text["Y", Y/2]],
  Arrow[X - Y, Tail → Y, VectorColor → Red],
  Graphics[Text["X - Y", Y + (X - Y)/2]], Axes → True];
```



Rerun for different vectors X and Y, and you will see that the distance between the tip of X and the tip of Y is the same as the length of X − Y.

The length of X − Y is

$\sqrt{(X - Y).(X - Y)}$,

so the distance between the tip of X and the tip of Y is given by

$||X - Y|| = \sqrt{(X - Y).(X - Y)}$.

That's all there is to it.

□B.1.c.ii)

Measure the distance between {2, 7} and {3, −5} in two dimensions.
Measure the distance between {2, 7, −6} and {3, −5, 6} in three dimensions.

□**Answer:**

You measure thedistance between the points {2, 7} and {3, −5} in two dimensions by running:

```
X = {2, 7};
Y = {3, -5};
N[√((X - Y) . (X - Y))]
```

```
12.0416
```

You measure distance between the points {2, 7, −6} and {3, −5, 6} in three dimensions by running:

```
X = {2, 7, -6};
Y = {3, -5, 4};
N[√((X - Y) . (X - Y))]
```

```
15.6525
```

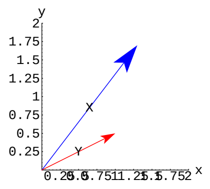
B.4) The push of one vector in the direction of another, and the formula

X.Y = ||X|| ||Y|| Cos[b],

where b is the angle between X and Y

Here are two vectors in two dimensions shown with their tails at {0, 0} in true scale:

```
X = {1.3, 1.7};
Y = {1, 0.5};

Show[Arrow[X, Tail → {0, 0}],
  Graphics[Text["X", X/2]], Arrow[Y, Tail → {0, 0}, VectorColor → Red],
  Graphics[Text["Y", Y/2]], PlotRange → {{0, 2}, {0, 2}},
  AspectRatio → Automatic, Axes → True, AxesLabel → {"x", "y"}];
```

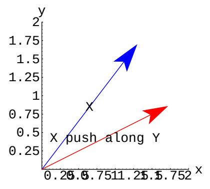


The question addressed here is how to measure the push of X in the direction of Y.
The answer is that the push of X in the direction of Y is calculated by the clean formula:

$$\text{XpushalongY} = \frac{X\cdot Y}{Y\cdot Y}\, Y.$$

Take a look:

```
XpushalongY = (X.Y)/(Y.Y) Y;

Show[Arrow[X, Tail → {0, 0}],
  Graphics[Text["X", X/2]],
  Arrow[XpushalongY, Tail → {0, 0}, VectorColor → Red],
  Graphics[
   Text["X push along Y", XpushalongY/2]], PlotRange → {{0, 2}, {0, 2}},
  AspectRatio → Automatic, Axes → True, AxesLabel → {"x", "y"}];
```

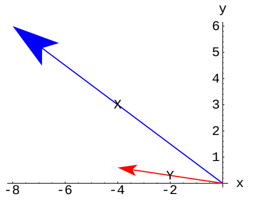


```
Grab the last two plots, align them and then animate running a slow
                              speed.
     You can find the align and animate commands in the Cell menu
```

Here are two new vectors X and Y:

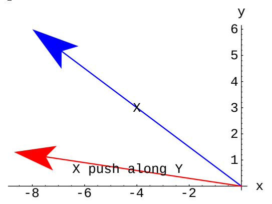
```
X = {-8, 6};
Y = {-4, 0.6};
```

```
Show[Arrow[X, Tail → {0, 0}],
  Graphics[Text["X", X/2]],
  Arrow[Y, Tail → {0, 0}, VectorColor → Red],
  Graphics[Text["Y", Y/2]], AspectRatio → Automatic, Axes → True,
  AxesLabel → {"x", "y"}];
```



Here are X and XpushalongY = $\frac{X.Y}{Y.Y}$ Y:

```
XpushalongY = (X . Y)/(Y . Y) Y;

Show[Arrow[X, Tail → {0, 0}],
  Graphics[Text["X", X/2]],
  Arrow[XpushalongY, Tail → {0, 0}, VectorColor → Red],
  Graphics[Text["X push along Y", XpushalongY/2]],
  AspectRatio → Automatic, Axes → True, AxesLabel → {"x", "y"}];
```



Try it for other X's and Y's until the plots make sense to you.

□B.4.a)

According to what was done above, you calculate the push of a vector X in the direction of another vector Y by calculating

XpushalongY = $\frac{X.Y}{Y.Y}$ Y.

Explain where this formula comes from, and explain what the push of X in the direction of Y means.

□ **Answer:**

Go with cleared vectors X and Y.

Put

$$f[t] = ||X - t\,Y|| = \sqrt{(X - t\,Y).(X - t\,Y)}.$$

The function f[t] measures the distance between the tip of X and the tip of the vector t Y when both vectors have their tails at {0, 0}.

```
Clear[f, t, x, y, X, Y]
X = {x[1], x[2]};
Y = {y[1], y[2]};

f[t_] = √((X - t Y) . (X - t Y))
```

$\sqrt{(x[1] - t\,y[1])^2 + (x[2] - t\,y[2])^2}$

f[t] is as small as it can be when f′[t] = 0:

```
Solve[f′[t] == 0, t]
```

$\{\{t \to -\frac{-x[1]\,y[1] - x[2]\,y[2]}{y[1]^2 + y[2]^2}\}\}$

This is the same as bestt = $\frac{X.Y}{Y.Y}$:

```
bestt = (X . Y)/(Y . Y)
```

$\frac{x[1]\,y[1] + x[2]\,y[2]}{y[1]^2 + y[2]^2}$

This is the t that makes

f[t] = ||X − t Y||

as small as it can be.

But you already know that

XpushalongY = $\frac{X.Y}{Y.Y}$ Y.

The upshot:

The push of X in the direction of Y is that multiple of Y whose tip is closest to the tip of X when you put the tails of both vectors at {0, 0}.

This works in three dimensions as well.

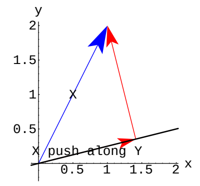
□B.4.b)

Take a look at

→ X and XpushalongY with their tails at {0, 0}, and

→ (X − XpushalongY) with its tail at the tip of XpushalongY for two sample vectors X and Y in true scale:

```
X = {1, 2};
Y = {4, 1};

XpushalongY = (X . Y Y)/(Y . Y);

Show[Arrow[X, Tail → {0, 0}],
  Graphics[Text["X", X/2]],
  Arrow[XpushalongY, Tail → {0, 0}, VectorColor → Red],
  Graphics[Text["X push along Y", XpushalongY/2]],
  Arrow[X - XpushalongY, Tail → XpushalongY, VectorColor → Red],
  Graphics[Line[{-0.2 Y, 0.8 Y}]], AspectRatio → Automatic,
  Axes → True, AxesLabel → {"x", "y"}];
```



The line is the plot of the tips of relevant multiples t Y of Y.
Describe what you see and explain why you see it.

□ **Answer:**

You see a right triangle.

And this is what you'll see for any two vectors X and Y unless X and Y are parallel.

Reason:

→ The push of X in the direction of Y is the multiple of Y whose tip is closest to the tip of X when you put the tails of both vectors at {0, 0}.

The upshot:

The tip of XpushalongY is the closest point on the line to the tip of X.

And the shortest distance between the line and the tip of X is the perpendicular distance.

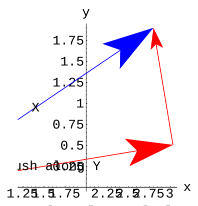
□B.4.c) The formula (X . Y) = ||X|| ||Y|| Cos[b] where b is the angle between X and Y

Take another look at

→ X and XpushalongY with their tails at {0, 0} and

→ (X − XpushalongY) with its tail at the tip of XpushalongY for two sample vectors X and Y in true scale:

```
X = {2.8, 1.9};
Y = {1.2, 0.2};

XpushalongY = (X . Y)/(Y . Y) Y;

plot = Show[Arrow[X, Tail → {0, 0}],
   Graphics[Text["X", X/2]],
   Arrow[XpushalongY, Tail → {0, 0}, VectorColor → Red],
   Graphics[Text["X push along Y", XpushalongY/2]],
   Arrow[X - XpushalongY, Tail → XpushalongY, VectorColor → Red],
   Axes → True, AxesLabel → {"x", "y"}];
```



Use the plot to help explain the formula
$(X.Y) = ||X||\ ||Y||\ \mathrm{Cos}[b]$
where
b is the angle between X and Y,
$||X|| = \sqrt{X.X}$, and
$||Y|| = \sqrt{Y.Y}$.

□ **Answer:**

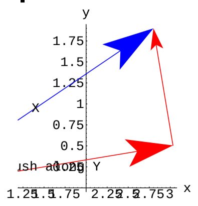
The goal is to explain the formula

$X.Y = ||X||\ ||Y||\ \mathrm{Cos}[b]$,

where b is the angle between X and Y when their tails are at the origin.

Take another look:

```
Show[plot];
```



Remembering that

$\mathrm{Cos}[b] = \frac{\text{adjacent}}{\text{hypotenuse}}$,

read off

$\mathrm{Cos}[b] = t\ \frac{||Y||}{||X||}$

where

$t = \frac{X.Y}{Y.Y}$

because

$\mathrm{XpushalongY} = \frac{X.Y}{Y.Y}\ Y$.

Now you know why

$||X||\ \mathrm{Cos}[b] = t\ ||Y||$.

Next, multiply both sides by ||Y|| to get

$||X||\ ||Y||\ \mathrm{Cos}[b] = t\ ||Y||^2$.

This tells you that to explain why

$X.Y = ||X||\ ||Y||\ \mathrm{Cos}[b]$,

you gotta explain why

$t\ ||Y||^2 = X.Y$.

This is easy because

$t = \frac{X.Y}{Y.Y}$;

so

$t\ ||Y||^2 = t\ Y.Y$

$= \frac{X.Y}{Y.Y}\ (Y.Y) = X.Y$.

That's it.

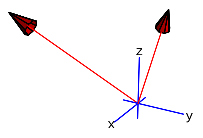
This formula also works in three dimensions.

B.5) X . Y = 0 means X is perpendicular to Y

□B.5.a.i)

Look at
$X = \{1, \frac{4}{3}, \frac{7}{2}\}$ and $Y = \{-\frac{1}{3}, -5, 2\}$
with their tails at {0, 0, 0}:

```
X = {1, 4/3, 7/2};

Y = {-1/3, -5, 2};

threedims = Axes3D[1.5, 0.2];

Show[Arrow[X, VectorColor → Red],
      Arrow[Y, VectorColor → Red],
      threedims, PlotRange → All, ViewPoint → CMView, Boxed → False];
```



These vectors look like they might be perpendicular.
How can you tell for sure?

□ **Answer:**

Just look at X . Y:

```
X . Y
0
```

Now you know that X . Y = 0; so you know for sure that X is indeed perpendicular to Y.

□B.5.a.ii) The perpendicularity test

Explain the statement:
If X . Y = 0, then X is perpendicular to Y.

□ **Answer:**

Well, if X . Y = 0, then because

$(X.Y) = ||X||\ ||Y||\ (\mathrm{Cos}[b])$,

you know that

$\mathrm{Cos}[b] = 0 = \mathrm{Cos}[\frac{\pi}{2}] = \mathrm{Cos}[-\frac{\pi}{2}]$.

This tells you that the angle between X and Y is a right angle.

□B.5.b)

How do you know that
X = {6, 2}
is perpendicular to
Y = {3, −9}?

□ **Answer:**

Check to see that X . Y = 0:

```
X = {6, 2};
Y = {3, -9};
X . Y
0
```

Yep; X = {6, 2} is perpendicular to Y = {3, −9}.

VC.01 Vectors Point the Way Tutorials

T.1) Velocity and acceleration

At time t, an object is at the location
P[t] = {x[t], y[t]}.

```
Clear[t, x, y, P]
P[t_] = {x[t], y[t]}
{x[t], y[t]}
```

The velocity of the object at time t is given by:

```
Clear[vel]
vel[t_] = D[P[t], t]
{x′[t], y′[t]}
```

The velocity is a vector quantity. When you put its tail at {x[t], y[t]}, the velocity vector
{x′[t], y′[t]} = P′[t] = D[P[t], t]
is tangent to the curve traced out by the motion of P[t]. This velocity

vector points in the instantaneous direction that the object is going.
The speed of the object is the length of the velocity vector:

```
Clear[speed]
speed[t_] = √(vel[t] . vel[t])
```

$\sqrt{x'[t]^2 + y'[t]^2}$

The acceleration of the object is given by:

```
Clear[accel]
accel[t_] = D[vel[t], t]
```

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

□T.1.a)

What does the acceleration vector measure?

□Answer:

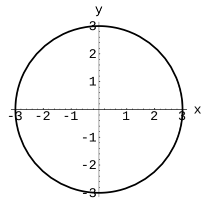
Because the velocity vector is the derivative of the position vector, the velocity vector measures the rate of change of the position.
Because the acceleration vector is the derivative of the velocity vector, the acceleration vector measures the rate of change of the velocity.

□T.1.b)

At time t, an object is at the point
P[t] = {3 Cos[t], 3 Sin[t]}.
Plot the motion of the object in true scale for $0 \le t \le 2\pi$, and include several velocity and acceleration vectors.
Discuss what the plot reveals.

□Answer:

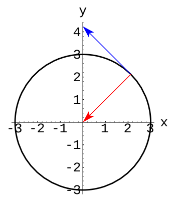
```
Clear[t, P];
P[t_] = {3 Cos[t], 3 Sin[t]};

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



Circular motion - no big surprise.

Now set up the velocity and acceleration vectors:

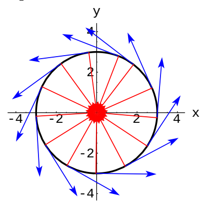
```
Clear[vel, accel];
vel[t_] = D[P[t], t];

accel[t_] = D[vel[t], t];

Clear[velvector]
velvector[t_] := Arrow[vel[t], Tail → P[t], VectorColor → Blue];

Clear[accelvector]
accelvector[t_] := Arrow[accel[t], Tail → P[t], VectorColor → Red];

Show[
 curveplot, velvector[π/4], accelvector[π/4], AspectRatio → Automatic];
```



The velocity vector indicates the direction of the motion, and the acceleration vector indicates that the force on the object (force=mass times acceleration) is directed toward the origin, perpendicular to the velocity vector.

Check out some more velocity and acceleration vectors:

```
Show[curveplot, Table[velvector[t], {t, 1.2, 7.2, .5}],
 Table[accelvector[t], {t, 1.2, 7.2, .5}], AspectRatio → Automatic];
```



Neato.

In this set up, the acceleration vectors at P[t] are all perpendicular to the velocity vectors at P[t]. This means the force on the object neither speeds up nor slows down the object. To confirm this, check out the speed:

```
Clear[speed]
speed[t_] = √(TrigExpand[vel[t] . vel[t]])
```

3

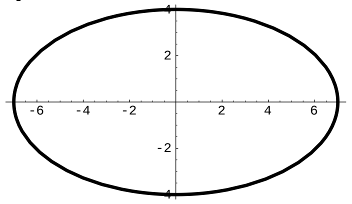
The object moves with a constant speed of 3 units of length per unit of time. Another way to see this is to look at the plot and note that all the velocity vectors have the same length.

□T.1.c.i)

At time t, an object is at the point
P[t] = {7 Cos[t], 4 Sin[t]}.
Plot the motion of the object in true scale for $0 \le t \le 2\pi$, and include several velocity and acceleration vectors.
Discuss what the plot reveals.

□Answer:

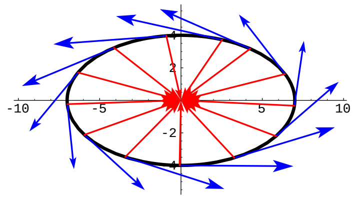
```
Clear[t, P]
P[t_] = {7 Cos[t], 4 Sin[t]};

curveplot = ParametricPlot[P[t], {t, 0, 2 π},
  PlotStyle → Thickness[0.01], AspectRatio → Automatic];
```



The object is moving on an ellipse.

To see which way the object is moving, look at the velocity and acceleration vectors:

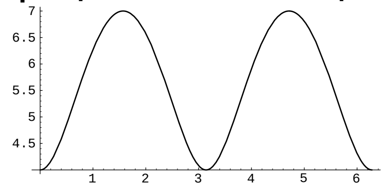
```
Clear[vel, accel];
vel[t_] = D[P[t], t];

accel[t_] = D[vel[t], t];

Clear[velvector]
velvector[t_] := Arrow[vel[t], Tail → P[t], VectorColor → Blue];

Clear[accelvector]
accelvector[t_] := Arrow[accel[t], Tail → P[t], VectorColor → Red];

Show[curveplot, Table[velvector[t], {t, 1.2, 7.2, .5}],
 Table[accelvector[t], {t, 1.2, 7.2, .5}], AspectRatio → Automatic];
```



The velocity vectors have different lengths, and the acceleration vectors are not perpendicular to the velocity vectors. This signals that the speed is not constant. Looking again, you can see that the acceleration vectors are working to speed up the object in the first and third quadrants, and the acceleration vectors are slowing the object down in the second and fourth quadrants.

Confirm with a plot of the speed:

```
Plot[√(vel[t] . vel[t]), {t, 0, 2 π}];
```



Going faster, then slower, then faster, and then slower.

□T.1.c.ii) The tangential component of the acceleration

Folks like to call the push of the acceleration vector in the direction of the velocity vector "the tangential component of the acceleration."
Go back to the plot in part i) above and show some velocity vectors and the tangential components of the acceleration vectors.
Discuss what your plot reveals.

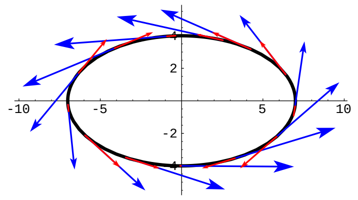
□**Answer:**

Remember that the push of a vector X in the direction of another vector Y is calculated by

$$\frac{X.Y}{Y.Y}\,Y.$$

Here come the tangential components of the acceleration vectors plotted with the velocity vectors on the curve:

```
Clear[tancompaccel, tanaccelvector]
tancompaccel[t_] = (accel[t] . vel[t] vel[t])/(vel[t] . vel[t]);

tanaccelvector[t_] :=
 Arrow[tancompaccel[t], Tail → P[t], VectorColor → Red];

Show[curveplot, Table[velvector[t], {t, 1.2, 7.2, .5}],
 Table[tanaccelvector[t], {t, 1.2, 7.2, .5}],
 AspectRatio → Automatic];
```



When the tangential components of the acceleration vectors are pushing in the same direction as the velocity vectors, the object is gaining speed.

When the tangential components of the acceleration vectors are pushing in the direction opposite to the velocity vectors, the object is losing speed.

Look at the brakes go on as the object goes into the sharp turns, and look at how the object speeds up as it comes out of the turns.

□T.1.c.iii) The normal component of the acceleration

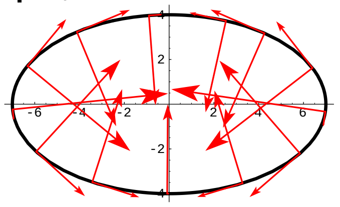
Folks like to call the push of the acceleration vector in the direction perpendicular to the velocity vector by the name "the normal component of the acceleration."
Go back to the plot in part i) above and show some acceleration vectors split into normal and tangential components.
Discuss what your plot reveals.

□**Answer:**

You already have the tangential components of the acceleration vectors live in part ii) above. The perpendicular component is just

accel[t] – tancompaccel[t]:

```
Clear[perpcompaccel, perpaccelvector]
perpcompaccel[t_] = accel[t] - tancompaccel[t];

perpaccelvector[t_] :=
 Arrow[perpcompaccel[t], Tail → P[t], VectorColor → Red];

Show[curveplot, Table[tanaccelvector[t], {t, 1.2, 7.2, 0.5}],
 Table[perpaccelvector[t], {t, 1.2, 7.2, 0.5}],
 AspectRatio → Automatic];
```



The tangential components govern the speed of the object, and the perpendicular components measure the tug on the object as it goes on its elliptical path. Note that the tug is greater in the sharp turns than it is on the flatter parts of the curve. Just as you expect.

T.2) Using the normal vector to bounce light beams off two-dimensional curves

When you have a parametric formula P[t] = {x[t], y[t]} for a curve in two dimensions, then you can calculate a tangent vector at P[t] by calculating

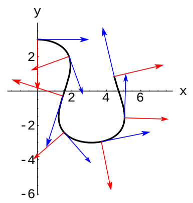
tan[t] = {x′[t], y′[t]}.

You can also calculate a vector perpendicular to the curve at P[t] by calculating

normal[t] = {y′[t], −x′[t]}.

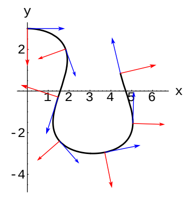
Fancy folks like to call the perpendicular vector by the name "normal vector."
Here is a sample curve shown with a selection of tangents and normals:

```
Clear[t, x, y, P]
x[t_] = t + Sin[2 t];
y[t_] = 3 Cos[t];
P[t_] = {x[t], y[t]};

a = 0;
b = 5;
curveplot = ParametricPlot[P[t], {t, a, b},
  PlotStyle → Thickness[0.01], DisplayFunction → Identity];

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

Clear[tanvector]
tanvector[t_] := Arrow[tan[t], Tail → P[t], VectorColor → Blue];

Clear[normalvector]
normalvector[t_] := Arrow[normal[t], Tail → P[t], VectorColor → Red];

Show[
 curveplot, Table[{tanvector[t], normalvector[t]}, {t, a, b, (b - a)/6}],
 AspectRatio → Automatic, AxesLabel → {"x", "y"},
 DisplayFunction → $DisplayFunction];
```



The vectors overwhelm the plot.
Tone them down by applying a scale factor to the tangent and normal vectors:

```
scalefactor = 0.6;

Clear[tanvector]
tanvector[t_] := Arrow[tan[t],
  Tail → P[t], VectorColor → Blue, ScaleFactor → scalefactor ];

Clear[normalvector]
normalvector[t_] := Arrow[normal[t],
  Tail → P[t], VectorColor → Red, ScaleFactor → scalefactor];

Show[
 curveplot, Table[{tanvector[t], normalvector[t]}, {t, a, b, (b - a)/6}],
 AspectRatio → Automatic, AxesLabel → {"x", "y"},
 DisplayFunction → $DisplayFunction];
```

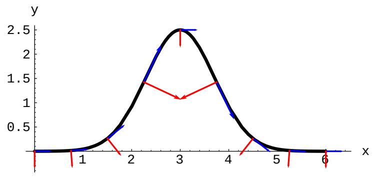


That's a bit better.

changing the independent variable, P[t] = {t, f[t]}. That makes this job easy:

Replot in parametric form, scale the tangent and normal vectors, and run:

```
Clear[t, x, y, P]
x[t_] = t;
y[t_] = f[t];
P[t_] = {x[t], y[t]};

curveplot = ParametricPlot[P[t], {t, a, b},
  PlotStyle → Thickness[0.01], DisplayFunction → Identity];

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

normal[t_] = {y′[t], -x′[t]};

scalefactor = 0.6;
Clear[tanvector]
tanvector[t_] := Arrow[scalefactor tan[t],
  Tail → P[t], VectorColor → Blue, ScaleFactor → scalefactor];

Clear[normalvector]
normalvector[t_] := Arrow[scalefactor normal[t],
  Tail → P[t], VectorColor → Red, ScaleFactor → scalefactor];

Show[
 curveplot, Table[{tanvector[t], normalvector[t]}, {t, a, b, (b - a)/8}],
 AspectRatio → Automatic, AxesLabel → {"x", "y"},
 DisplayFunction → $DisplayFunction];
```



There ya go.

□T.2.a)

How did you know in advance that the normal vectors

$\{y'[t], -x'[t]\}$

are guaranteed to be perpendicular to the tangent vectors

$\{x'[t], y'[t]\}$?

□**Answer:**

The perpendicularity test says:

Two vectors X and Y are perpendicular if X . Y = 0.

Taking

$X = \{x'[t], y'[t]\}$ and $Y = \{y'[t], -x'[t]\}$,

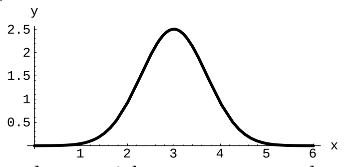
you don't need Mathematica to see that

$X . Y = x'[t]\, y'[t] + y'[t]\,(-x'[t]) = 0.$

□T.2.b) How to handle curves specified in nonparametric form

Here is a curve plotted in nonparametric form:

```
Clear[f, x]
f[x_] = 2.5 E^-(x-3)^2;

a = 0;
b = 6;
curveplot = Plot[f[x], {x, a, b}, PlotStyle → Thickness[0.01],
  AxesLabel → {"x", "y"}];
```



How do you stick some tangent and normal vectors onto this curve?

□**Answer:**

Every function specified as y = f[x] can be thought of as a shorthand version of the parametric version, P[x] = {x, y} = {x, f[x]}, or, after

□T.2.c)

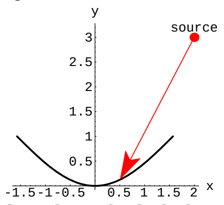
Here is the curve

$f[x] = 1 - \text{Cos}[x]$ for $-\frac{\pi}{2} \le x \le \frac{\pi}{2}$

shown with a light beam emanating from {2, 3} and hitting the curve at

{0.5, f[0.5]}:

```
Clear[t, f, x, y, P]
f[x_] = 1 - Cos[x];
x[t_] = t;
y[t_] = f[t];
P[t_] = {x[t], y[t]};

a = -π/2;
b = π/2;
curveplot =
 ParametricPlot[P[t], {t, a, b}, PlotStyle → Thickness[0.01],
  AxesLabel → {"x", "y"}, DisplayFunction → Identity];
source = {2, 3};
hit = {0.5, f[0.5]};
label = Graphics[Text["source", source + {0, 0.2}]];
sourceplot = Graphics[{Red, PointSize[0.05], Point[source]}];
incominglight = Arrow[hit - source, Tail → source, VectorColor → Red];

Show[curveplot, incominglight, label, sourceplot, PlotRange → All,
 DisplayFunction → $DisplayFunction];
```

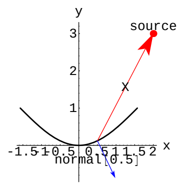


Plot where the light beam goes after it bounces off the curve.

□**Answer:**

The physical principle behind bouncing light is that the angle of incidence is the same as the angle of reflection. To get an idea of how to make use of this physical principle, include the normal vector at the hit and run the vector from the hit to the source.

```
X = source - hit;

reverseincominglight = Arrow[X, Tail → hit, VectorColor → Red];

extralabels = {Graphics[Text["X", hit + (source - hit)/2]],
                   Graphics[Text["normal[0.5]", hit + normal[0.5]/2]]};

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

normalvector[t_] := Arrow[normal[t], Tail → hit, VectorColor → Blue];

Show[curveplot, reverseincominglight, normalvector[0.5],
 extralabels, sourceplot, label, PlotRange → All,
 DisplayFunction → $DisplayFunction];
```



Agree that R is a vector specifying the direction of the reflected light and note that the angle R makes with normal[0.5] is the same as the angle X makes with normal. So you want

$$X\,.\,\text{normal}[0.5] = \|X\|\ \|\text{normal}[0.5]\|\ \text{Cos}[b]$$

$$R\,.\,\text{normal}[0.5] = \|R\|\ \|\text{normal}[0.5]\|\ \text{Cos}[b]$$

where b is the angle between X and normal[0.5].

When you make

$$R\,.\,R = \|R\|^2 = \|X\|^2 = X\,.\,X,$$

you get two equations to solve for the two slots of R:

```
Clear[R, r1, r2]
R = {r1, r2};

equation1 = R . normal[0.5] == X . normal[0.5];
equation2 = R . R == X . X;

solutions = Solve[{equation1, equation2}]
```

```
{{r1 → -3.18283, r2 → 0.632514}, {r1 → 1.5, r2 → 2.87758}}
```

One solution for R is X itself:

```
X
```

```
{1.5, 2.87758}
```

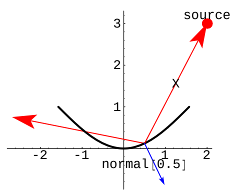
Discard it and go with:

```
r1 = -3.18283;
r2 = 0.632514;
R = {r1, r2}
```

```
{-3.18283, 0.632514}
```

Try it out:

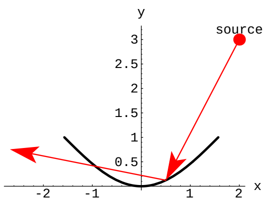
```
outgoinglight = Arrow[R, Tail → hit, VectorColor → Red];

Show[sourceplot, outgoinglight, curveplot, reverseincominglight,
 normalvector[0.5], extralabels, sourceplot, label, Axes → True,
 PlotRange → All, DisplayFunction → $DisplayFunction];
```



Beautiful.

Here's the path of the light:

```
Show[curveplot, incominglight, outgoinglight, label, sourceplot,
 Axes → True, PlotRange → All, DisplayFunction → $DisplayFunction];
```



Lookin' just fine, thank you.

T.3) Lines

To specify a line, you need a point to run the line through and you need a vector to specify the direction of the line.

□T.3.a.i) 2D lines

Come up with a parametric formula for the line that runs through {−2, −1} and runs parallel to the vector {3, 1}.
Show the line, the vector, and the point in a single plot.
Give the equation of the line in the nonparametric form
y = f[x].

□**Answer:**

Here comes the parametric formula:

```
point = {-2, -1};
parallelvector = {3, 1};

Clear[line, t]
line[t_] = point + t parallelvector
```

```
{-2 + 3 t, -1 + t}
```

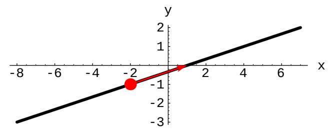
A parametric formula for the line that runs through {−2, −1} and runs parallel to the vector {3, 1} is

$$\{x[t], y[t]\} = \{-2, -1\} + t\,\{3, 1\}.$$

Note how the parametric formula displays the given point and the parallel vector.

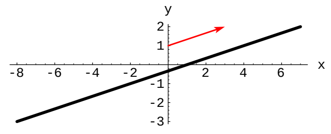
Here comes the plot:

```
directionvector =
 Arrow[parallelvector, Tail → point, VectorColor → Red];

pointplot = Graphics[{PointSize[0.04], Red, Point[point]}];

lineplot =
 ParametricPlot[line[t], {t, -2, 3}, PlotStyle → Thickness[0.01],
  AxesLabel → {"x", "y"}, DisplayFunction → Identity];

Show[lineplot, pointplot, directionvector,
 DisplayFunction → $DisplayFunction];
```



Here's how the plot looks when you stick the tail of the parallel vector at {0, 1}:

```
directionvector =
 Arrow[parallelvector, Tail → {0, 1}, VectorColor → Red];

Show[lineplot, directionvector, DisplayFunction → $DisplayFunction];
```



Parallel.

To come up with the formula of the line in the nonparametric form y = f[x], look at:

```
Clear[x, y]
equation = {x, y} == line[t]
```

```
{x, y} == {-2 + 3 t, -1 + t}
```

Eliminate t by writing

$t = \frac{x+2}{3}$ and $t = y + 1$

and setting

$\frac{x+2}{3} == y + 1$:

```
squasht = Eliminate[equation, t]
x == 1 + 3 y
```

Now solve for y:

```
Solve[squasht, y]
```

$\{\{y \to \frac{1}{3}(-1+x)\}\}$

The nonparametric formula of this line is

$y = \frac{x-1}{3}$.

□T.3.a.ii) 3D lines

Come up with a parametric formula for the line that runs through {2, −3, −5} and runs parallel to the vector {−1.5, 2.5, 3.3}.
Show the line, the vector, and the point in a single plot.

□Answer:

You do it the same way you handle the 2D case.

Here comes the parametric formula:

```
point = {2, -3, -5};
parallelvector = {-1.5, 2.5, 3.3};
Clear[line, t]
line[t_] = point + t parallelvector
{2 - 1.5 t, -3 + 2.5 t, -5 + 3.3 t}
```

A parametric formula for the line that runs through

{2, −3, −5}

and runs parallel to the vector

{−1.5, 2.5, 3.3}

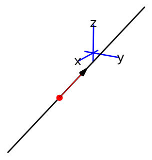
is

({x[t], y[t], z[t]} = {2, −3, −5} + t {−1.5, 2.5, 3.3})

Note how the parametric formula conveniently displays the given point and the parallel vector.

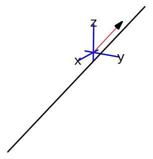
Here comes the plot:

```
directionvector =
  Arrow[parallelvector, Tail → point, VectorColor → Red];

pointplot = Graphics3D[{PointSize[0.04], Red, Point[point]}];
lineplot =
  ParametricPlot3D[line[t], {t, -2, 3}, DisplayFunction → Identity];
threedims = Axes3D[3, .2];

Show[
  threedims, lineplot, pointplot, directionvector, ViewPoint → CMView,
  Boxed → False, DisplayFunction → $DisplayFunction];
```



Here's how the plot looks when you stick the tail of the parallel vector at {0, 0, 0}:

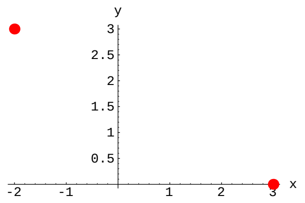
```
directionvector =
  Arrow[parallelvector, Tail → {0, 0, 0}, VectorColor → Red];

Show[threedims, lineplot, directionvector, ViewPoint → CMView,
  Boxed → False, DisplayFunction → $DisplayFunction];
```



□T.3.b.i) 2D lines

Here's a look at the two points {−2, 3} and {3, 0} in two dimensions:

```
point1 = {-2, 3};
point2 = {3, 0};
pointplot = Graphics[{{PointSize[0.04], Red, Point[point1]},
    {PointSize[0.04], Red, Point[point2]}}];

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



Come up with a parametric formula for the line that runs through these two points
Show the line and the two points in a single plot.

□Answer:

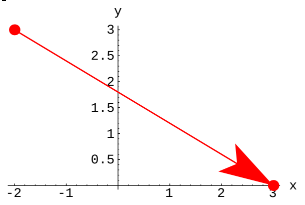
This line is parallel to the vector that runs from {−2, 3} to {3, 0}, which is given by

{3, 0 − {−2, 3} = {5, −3}.

```
parallelvector = point2 - point1
{5, -3}
```

Take a look:

```
Show[pointplot,
  Arrow[parallelvector, Tail → point1, VectorColor → Red],
  Axes → True, AxesLabel → {"x", "y"}];
```

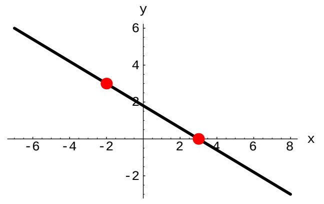


Here's a parametric formula:

```
Clear[line, t]
line[t_] = point1 + t parallelvector
{-2 + 5 t, 3 - 3 t}
```

You can see one of the points – namely, {−2, 3} – prominently displayed in this parametric formula. Here comes the plot:

```
lineplot =
  ParametricPlot[line[t], {t, -1, 2}, PlotStyle → Thickness[0.01],
   AxesLabel → {"x", "y"}, DisplayFunction → Identity];

Show[lineplot, pointplot, DisplayFunction → $DisplayFunction];
```



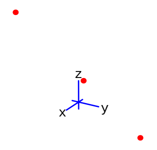
There you go.

□T.3.b.ii) 3D lines

Here's a look at three points in three dimensions:

```
point1 = {-1.51, 2.32, -1.81};
point2 = {2.69, -1.14, 4.43};
point3 = {0.59, 0.59, 1.31};
```

```
pointplot = {Graphics3D[{PointSize[0.04], Red, Point[point1]}],
   Graphics3D[{PointSize[0.04], Red, Point[point2]}],
   Graphics3D[{PointSize[0.04], Red, Point[point3]}]};
threedims = Axes3D[1, .3];

Show[threedims, pointplot, ViewPoint → CMView, PlotRange → All,
 Boxed → False];
```



These three points certainly look like they're lined up in a straight line. How can you tell for sure?

□Answer:

Put a line between two of them and see whether the third point is on this line:

```
Clear[line, t]
line[t_] = point1 + t (point2 - point1);
distance[t_] = √((line[t] - point3) . (line[t] - point3));

Solve[distance[t] == 0, t]
{{t → 0.5}}
```

Compare:

```
point3
{0.59, 0.59, 1.31}
line[0.5]
{0.59, 0.59, 1.31}
```

Yep, the line goes through all three points.

Another way of seeing this is to make a vector running from point1 to point2 and another vector running from point1 to point3:

```
vector12 = point2 - point1
{4.2, -3.46, 6.24}
```

```
vector13 = point3 - point1
{2.1, -1.73, 3.12}
vector12PushAlongvector13 = (vector12 . vector13)/(vector13 . vector13) vector13
{4.2, -3.46, 6.24}
vector12 == vector12PushAlongvector13
True
```

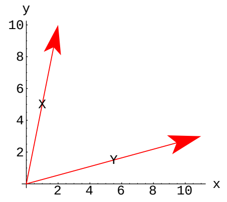
This tells you that the push of vector12 in the direction of vector13 is the same as vector12.

This is enough to confirm that the three points all reside on a single line.

□T.3.c) Midpoints

Look at:

```
X = {2, 10};
Y = {11, 3};
setup = Show[Arrow[X, Tail → {0, 0}, VectorColor → Red],
   Graphics[Text["X", X/2]], Arrow[Y, Tail → {0, 0}, VectorColor → Red],
   Graphics[Text["Y", Y/2]], Axes → True, AxesLabel → {"x", "y"}];
```



Use vectors to find the midpoint of the line segment that runs from the tip of X to the tip of Y.

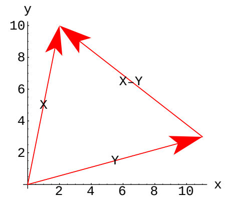
□Answer:

Remember that if you hook the tail of X − Y to the tip of Y, then X − Y runs from the tip of Y to the tip of X. Why?

Because

$$X = Y + (X - Y).$$

```
Show[setup, Arrow[X - Y, Tail → Y, VectorColor → Red],
 Graphics[Text["X-Y", Y + (X - Y)/2]]];
```

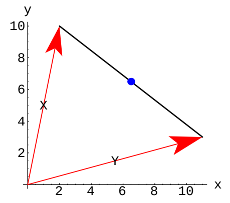


To get to the midpoint of the line segment connecting the tip of Y and the tip of X, you can travel on Y to the tip of Y, and then travel along X − Y, but you should only go halfway. This means:

```
midpoint = Y + (X - Y)/2
{13/2, 13/2}
```

Check with a plot:

```
midpointplot = Graphics[{Blue, PointSize[0.04], Point[midpoint]}];
lineplot = Graphics[Line[{X, Y}]];

Show[setup, lineplot, midpointplot];
```



Copacetic.

T.4) Pursuits

□T.4.a.i)

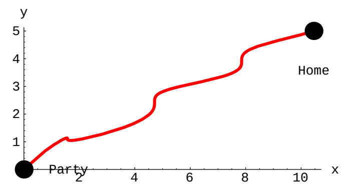
Bubba is really enjoying one of those big time keg parties out in the woods. Having learned to monitor his blood alcohol level in earlier Calculus&Mathematica lessons, Bubba took his bicycle to the party instead of driving his car. At time t = 0, Bubba leaves the party and gets on his bicycle to ride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

At time t, after leaving the party, Bubba is at the point

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

Here is his route:

```
Clear[bubba, t]
bubba[t_] = {2 t + 1/2 Sin[4 t], t + E^-t Sin[3 t]};
bubbaroute = ParametricPlot[
   bubba[t], {t, 0, 5}, PlotStyle → {{Thickness[0.01], Red}},
   AxesLabel → {"x", "y"}, DisplayFunction → Identity];

pointplot = {Graphics[{PointSize[0.06], Point[bubba[0]]}],
   Graphics[{PointSize[0.06], Point[bubba[5]]}]};
labels = {Graphics[Text["Party", bubba[0], {-2, 0}]],
   Graphics[Text["Home", bubba[5], {0, 5}]]};

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



Slightly unsteady is your friend Bubba.
While Bubba was at the party, his Blue Tick hound dog was sleeping at the point {0, 2}. As Bubba left the party, the dog wakes up and chases Bubba. Here is the dog's scheme:
If the dog is at
{x[t], y[t]}
at time t, then he leaves {x[t], y[t]} with instantaneous velocity
bubba[t] – {x[t], y[t]}.
This means that at any time t, the dog is always running toward the point bubba[t], Bubba's current position.
Does the dog catch up with Bubba before Bubba gets home and locks the dog out?

□ **Answer:**

You guessed it.

This is a job for a system of differential equations.

Put:

```
Clear[dogvelocity, x, y, t]
dogvelocity[t_] = bubba[t] - {x[t], y[t]}
```

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

The system of differential equations is:

```
Clear[Derivative]
equationx = x'[t] == dogvelocity[t][[1]]
```

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

```
equationy = y'[t] == dogvelocity[t][[2]]
```

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

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

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

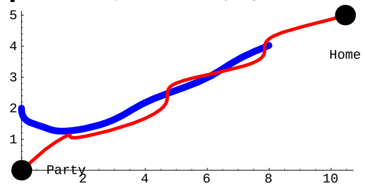
```
startery = y[0] == 2
```

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

Here comes the approximate plot of the dog's route for

$0 \le t \le 5$:

```
endtime = 5;
Clear[x, y, t, Derivative]
approxsolutions = NDSolve[{equationx, equationy, starterx, startery},
  {x[t], y[t]}, {t, 0, endtime}];

Clear[fakex, fakey]
fakex[t_] = x[t] /. approxsolutions[[1]];
fakey[t_] = y[t] /. approxsolutions[[1]];

dogplot = ParametricPlot[{fakex[t], fakey[t]}, {t, 0, endtime},
  PlotStyle → {{Blue, Thickness[0.02]}}, DisplayFunction → Identity];

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



The dog didn't make it.

Hope it doesn't rain.

□T.4.a.ii)

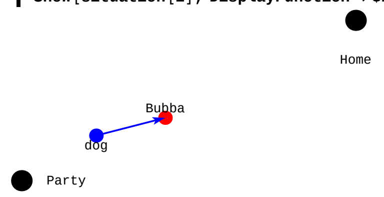
Illustrate the dog's strategy by making a movie reviewing the chase.

□ **Answer:**

Set up the movie code and look at one frame:

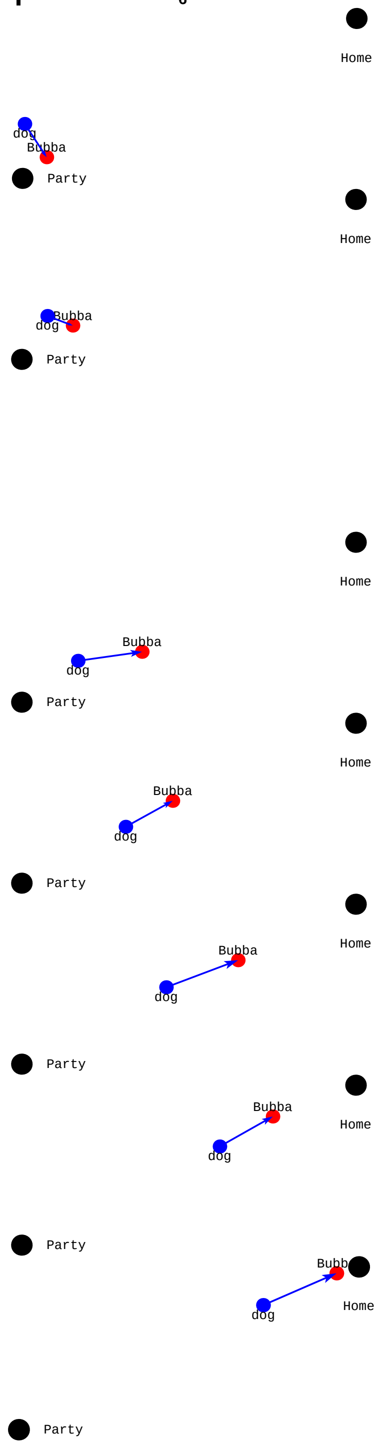
```
Clear[bubbapoint, bubbavelocityvector,
 dogpoint, dogvelocityvector, situation, t]
bubbapoint[t_] := Graphics[{{Red, PointSize[0.04], Point[bubba[t]]},
   Text["Bubba", bubba[t] + {0, 0.3}]}];
```

```
dogpoint[t_] :=
 Graphics[{{Blue, PointSize[0.04], Point[{fakex[t], fakey[t]}]},
   Text["dog", {fakex[t], fakey[t]} - {0, 0.3}]}];

dogvelocityvector[t_] := Arrow[{fakex'[t], fakey'[t]},
  Tail → {fakex[t], fakey[t]}, VectorColor → Blue];

situation[t_] := Show[pointplot, labels, bubbapoint[t],
  dogpoint[t], dogvelocityvector[t], AspectRatio → Automatic,
  PlotRange → All, DisplayFunction → Identity];

Show[situation[2], DisplayFunction → $DisplayFunction];
```



See the chase:

```
Table[Show[situation[t], DisplayFunction → $DisplayFunction];,
 {t, 0.2, 4.8, 4.6/6}];
```



```
Animate these by grabbing all the plots and selecting the animation
instruction in the Cell menu.
After the animation begins, press 3.
```

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

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

□T.4.a.iii)

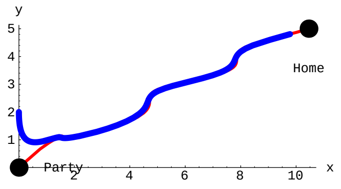
What should the dog have done to catch Bubba?

□ **Answer:**

Common sense says that the dog should have run faster.

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

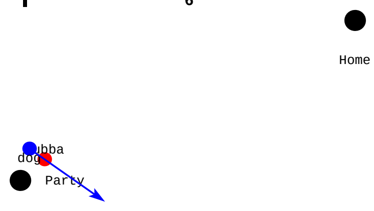
```
Clear[newdogvelocity, x, y, t]
newdogvelocity[t_] = 5 dogvelocity[t];
Clear[Derivative]
equationx = x′[t] == newdogvelocity[t]⟦1⟧;
equationy = y′[t] == newdogvelocity[t]⟦2⟧;
starterx = x[0] == 0;
startery = y[0] == 2;
endtime = 5;
Clear[x, y, t, Derivative]
approxsolutions = NDSolve[{equationx, equationy, starterx, startery},
  {x[t], y[t]}, {t, 0, endtime}];
Clear[fakex, fakey]
fakex[t_] = x[t] /. approxsolutions⟦1⟧;
fakey[t_] = y[t] /. approxsolutions⟦1⟧;
dogplot = ParametricPlot[{fakex[t], fakey[t]}, {t, 0, endtime},
  PlotStyle → {{Blue, Thickness[0.02]}}, DisplayFunction → Identity];
outcome =
 Show[bubbaroute, dogplot, pointplot, labels, AspectRatio → Automatic,
  PlotRange → All, DisplayFunction → $DisplayFunction];
```

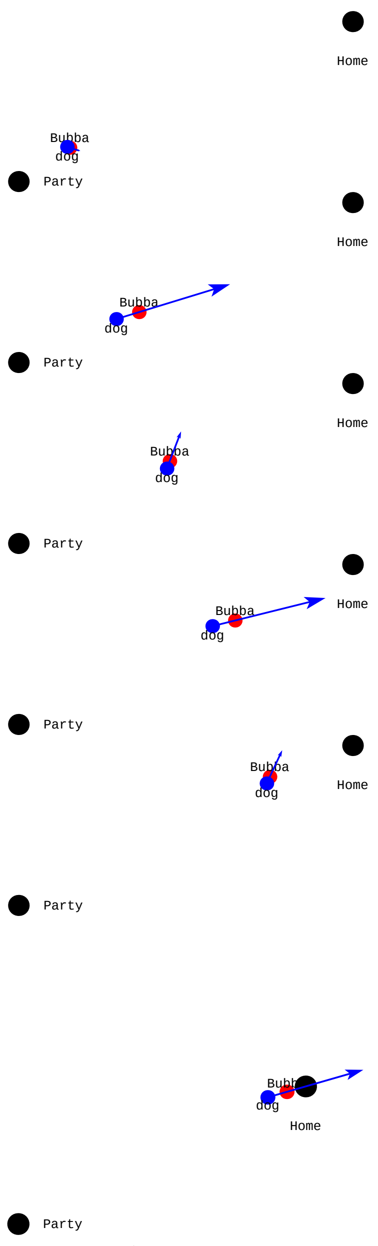


The smart money bets that the dog trotted through the door only seconds after Bubba opened the door. In fact, the dog was running only slightly behind Bubba for most of the chase.

Take a look:

```
Clear[dogpoint, dogvelocityvector, situation, t]
dogpoint[t_] :=
 Graphics[{{Blue, PointSize[0.04], Point[{fakex[t], fakey[t]}]},
   Text["dog", {fakex[t], fakey[t]} - {0, 0.3}]}];

dogvelocityvector[t_] := Arrow[{fakex′[t], fakey′[t]},
  Tail → {fakex[t], fakey[t]}, VectorColor → Blue];

situation[t_] := Show[pointplot, labels, bubbapoint[t],
  dogpoint[t], dogvelocityvector[t], AspectRatio → Automatic,
  PlotRange → All, DisplayFunction → Identity];

Table[Show[situation[t], DisplayFunction → $DisplayFunction];,
 {t, 0.2, 4.8, 4.6/6}];
```





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

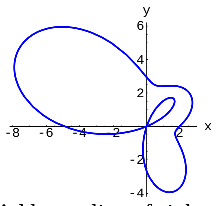
T.5) Spying along the tangent

At time t, you are at the point P[t] specified through the polar parameterization:

```
Clear[P, x, y, r, t]
r[t_] = 3.12 - 0.65 Cos[t] + 0.32 Cos[2 t] -
  0.83 Cos[3 t] + 1.92 Sin[t] - 2.68 Sin[2 t] + 1.79 Sin[3 t];

P[t_] = r[t] {Cos[t], Sin[t]};
```

Take a look:

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

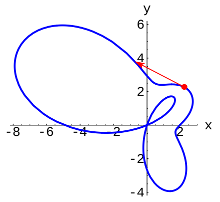


Add your line of sight to the plot when you look forward in the instantaneous direction you are going when t = 0.8.
Then describe, in terms of a clean formula, all the points you can see at the instant t = 0.8.

□ **Answer:**

You are looking in the direction of the tangent vector P'[0.8] with its tail at P[0.8]:

```
point = Graphics[{Red, PointSize[0.03], Point[P[0.8]]}];
Show[path, point, Arrow[P′[0.8], Tail → P[0.8], VectorColor → Red]];
```

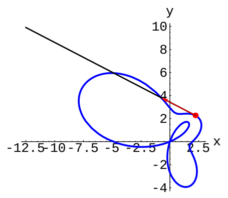


The points you can see are descibed by the clean formula

```
Clear[lineofsight, s]
lineofsight[s_] = N[P[0.8] + s P′[0.8]]
{2.2137 - 2.94617 s, 2.27931 + 1.52708 s}
```

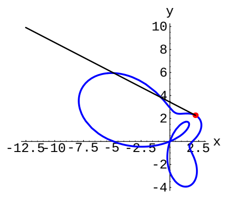
where you take s ≥ 0 because you are looking forward along the tangent vector. Here's part of your line of sight:

```
sightplot = ParametricPlot[
  lineofsight[s], {s, 0, 5}, DisplayFunction → Identity];

Show[path, point, sightplot,
 Arrow[P′[0.8], Tail → P[0.8], VectorColor → Red],
 DisplayFunction → $DisplayFunction];
```



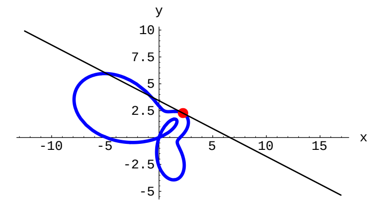
Without the tangent vector:

```
Show[path, point, sightplot, DisplayFunction → $DisplayFunction];
```



If you had eyes in the very back of your head, you would also be able to see points on this line:

```
bothways = ParametricPlot[
  lineofsight[s], {s, -5, 5}, DisplayFunction → Identity];

Show[path, point, bothways, DisplayFunction → $DisplayFunction];
```



The line you see is what lots of the good old folks like to call a tangent line at P[0.8].

VC.01 Vectors Point the Way
Give it a try!

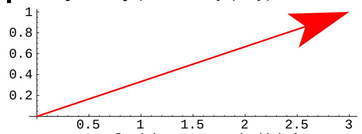
Experience with the starred problems will be especially beneficial for understanding later lessons

G.1) Vector and line fundamentals*

□G.1.a.i)

Here is the vector X = {3, 1} shown with its tail at {0, 0}:

```
X = {3, 1};
Show[Arrow[X, Tail → {0, 0}, VectorColor → Red], Axes → True];
```



```
If this command didn't execute satisfactorily,
   go back to the top of the lesson
 and run the initialization cells there.
```

What point is the tip of X sitting on?

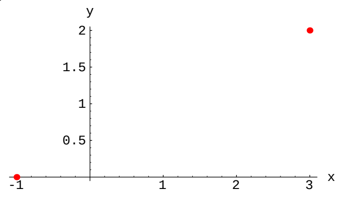
□G.1.a.ii)

When you move the vector X = {3, 1} so that its tail is at {2, 2}, where will you find its tip?

□G.1.a.iii)

Here are the points {3, 2} and {−1, 0}:

```
point1 = {3, 2};
point2 = {-1, 0};

Show[Graphics[{Red, PointSize[0.02], Point[point1]}],
 Graphics[{Red, PointSize[0.02], Point[point2]}], PlotRange → All,
 Axes → True, AxesLabel → {"x", "y"}];
```

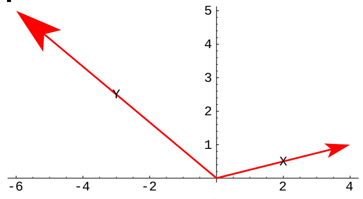


Add to the plot the vector whose tail is at {3, 2} and whose tip is at {−1, 0}.
Come up with parametric equations for the line that goes through both of these points.

□G.1.b.i)

Here are vectors X and Y with their tails at {0, 0}:

```
X = {4, 1};
Y = {-6, 5};
labels = Graphics[{{Text["X", X/2]}, {Text["Y", Y/2]}}];
Show[Arrow[X, Tail → {0, 0}, VectorColor → Red],
 Arrow[Y, Tail → {0, 0}, VectorColor → Red], labels, Axes → True];
```

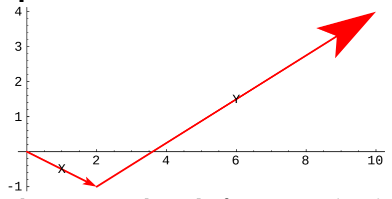


When you put the tail of X − Y on the tip of Y, where does the tip of X − Y turn out to be?

□G.1.b.ii)

Here are vectors X and Y with the tail of X at {0, 0} and the tail of Y at the tip of X:

```
X = {2, -1};
Y = {8, 5};
labels = Graphics[{{Text["X", X/2]}, {Text["Y", X + Y/2]}}];
Show[Arrow[X, Tail → {0, 0}, VectorColor → Red],
 Arrow[Y, Tail → X, VectorColor → Red], labels, Axes → True];
```

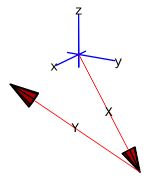


When you put the tail of X + Y at {0, 0}, where does the tip of X + Y turn out to be?
In the plot above, why is it that the label Y, which is planted at the point given by X + $\frac{Y}{2}$, lands midway between the tail of Y and the tip of Y?

□G.1.b.iii)

Here are vectors X and Y with the tail of X at {0, 0, 0} and the tail of Y at the tip of X:

```
X = {1, 4, -5};
Y = {2, -6, 4};

labels = Graphics3D[{{Text["X", X/2]}, {Text["Y", X + Y/2]}}];
threedims = Axes3D[2, 0.2];
Show[Arrow[X, Tail → {0, 0, 0}, VectorColor → Red],
 Arrow[Y, Tail → X, VectorColor → Red], threedims, labels,
 ViewPoint → CMView, Boxed → False];
```



When you put the tail of X + Y at {0, 0, 0}, where does the tip of X + Y turn out to be?
In the plot above, why is it that the label Y, which is planted at the point given by $X + \frac{Y}{2}$, lands midway between the tail of Y and the tip of Y?

□Tip:

We have used the shorthand command for building the 3-dimensional axes:

```
?Axes3D
```

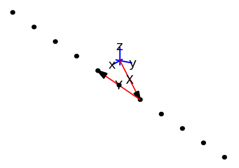
```
Axes3D[a, b] creates a Graphics3D object of
  cartesian axes with x, y, and z running from -a/3
  to a, and with axes labels b units beyond the
  tips of the axes.  Axes3D[a] is Axes3D[a, a/8].
```

□G.1.b.iv)

Here are the same vectors as shown in the last part together with some points of the form

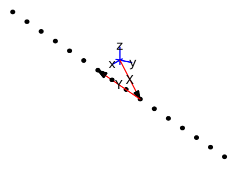
{x, y} = X + t Y for various choices of t:

```
X = {1, 4, -5};
Y = {2, -6, 4};
Clear[t]
points =
 Table[Graphics3D[{PointSize[0.02], Point[X + t Y]}], {t, -2, 3, 1/2}];
Show[Arrow[X, Tail → {0, 0, 0}, VectorColor → Red],
 Arrow[Y, Tail → X, VectorColor → Red], points, threedims,
 labels, ViewPoint → CMView, Boxed → False];
```



More points:

```
morepoints =
 Table[Graphics3D[{PointSize[0.02], Point[X + t Y]}], {t, -2, 3, 5/15}];
Show[
 Arrow[X, VectorColor → Red], Arrow[Y, Tail → X, VectorColor → Red],
 morepoints, threedims, labels, ViewPoint → CMView, Boxed → False];
```

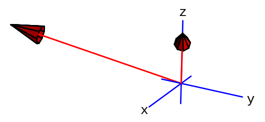


What kind of curve do these points come from?
Give a parametric formula for this curve.

□G.1.c)

Here are two vectors in three dimensions:

```
X = {2, 1, 2};
Y = {1, -4, 1};
threedims = Axes3D[1.5, 0.2];

Show[Arrow[X, Tail → {0, 0, 0}, VectorColor → Red],
 Arrow[Y, Tail → {0, 0, 0}, VectorColor → Red], threedims,
 ViewPoint → CMView, Boxed → False];
```



And here's a calculation of X . Y:

```
X . Y
0
```

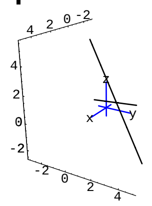
What piece of definite information about the relationship between X and Y did you get from the calculation of X . Y?

□G.1.d.i)

Here are plots of two lines in three dimensions:

```
point1 = {1, 1, 1};
point2 = {2, 2, 1.5};
X1 = {2, 2, 1};
X2 = {0, -2, 4};

Clear[line1, line2, s, t]
line1[t_] = point1 + t X1;
line2[s_] = point2 + s X2;

line1plot =
 ParametricPlot3D[line1[t], {t, -2, 2}, DisplayFunction → Identity];
line2plot =
 ParametricPlot3D[line2[s], {s, -1, 1}, DisplayFunction → Identity];
threedims = Axes3D[2, 0.2];
```

```
setup = Show[line1plot, line2plot, threedims, ViewPoint → CMView,
  Boxed → False, DisplayFunction → $DisplayFunction];
```



The plot gives the strong hint that the lines hit each other.
Explain how the following calculation confirms that the lines do hit each other:

```
Solve[Thread[line1[t] == line2[s]][[{1, 2}]], {s, t}]
```

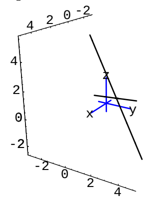
$\{\{s \to 0,\ t \to \frac{1}{2}\}\}$

Use what you see to pin down the coordinates of the point at which the lines hit each other.

□G.1.d.ii)

Take another look at the lines in part i) above.

```
Show[setup];
```

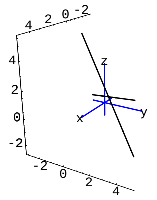


The two lines also look like they cross each other at right angles (perpendicularly). Use the dot product of two well-chosen vectors to confirm or contradict this observation.

□G.1.d.iii)

Here are plots of two new lines in three dimensions:

```
point1 = {1, 1, 1};
point2 = {2, 1.5, 1.5};
X1 = {2, 2, 1};
X2 = {0, -2, 4};

Clear[line1, lin2, s, t]
line1[t_] = point1 + t X1;
line2[s_] = point2 + s X2;

line1plot =
 ParametricPlot3D[line1[t], {t, -2, 2}, DisplayFunction → Identity];
line2plot =
 ParametricPlot3D[line2[s], {s, -1, 1}, DisplayFunction → Identity];
threedims = Axes3D[3, 0.2];

setup = Show[line1plot, line2plot, threedims, ViewPoint → CMView,
  Boxed → False, DisplayFunction → $DisplayFunction];
```

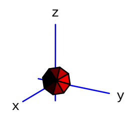


The plot hints that these two lines hit each other.
Do they?

□G.1.e)

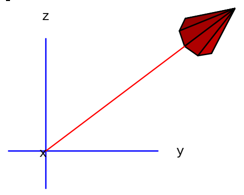
You can think of the CMView = {2.7, 1.6, 1.2} as a three-dimensional vector.
Here is this vector shown with ViewPoint → CMView:

```
CMView = {2.7, 1.6, 1.2};
X = CMView;
spacer = 0.2;
h = 1;
threedims = Axes3D[h, spacer];

Show[Arrow[X, Tail → {0, 0, 0}, VectorColor → Red], threedims,
 PlotRange → All, ViewPoint → CMView, Boxed → False];
```

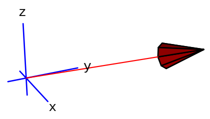


Gotcha!
Here is X = CMView from some other viewpoints:

```
NewView = {10, 0, 0};
Show[Arrow[X, Tail → {0, 0, 0}, VectorColor → Red], threedims,
 PlotRange → All, ViewPoint → NewView, Boxed → False];
```

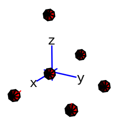


```
NewView = {2, -1, 1};
Show[Arrow[X, Tail → {0, 0, 0}, VectorColor → Red], threedims,
 PlotRange → All, ViewPoint → NewView, Boxed → False];
```



Here is the vector X = CMView = {2.7, 1.6, 1.2} shown with a squadron of its transplants with ViewPoint → CMView:

```
Clear[tail, k];
tail[1] = {0, 0, 0};
tail[2] = {2, 0, 0};
tail[3] = {0, 2, 0};
tail[4] = {0, 0, 2};
tail[5] = {-2, 0, 0};
tail[6] = {2, 2, 0};
squadron =
 Table[Arrow[X, Tail → tail[k], VectorColor → Red], {k, 1, 6}];
Show[squadron, threedims, PlotRange → All, ViewPoint → CMView,
 Boxed → False];
```



Duck before they pin you to your chair.
Play with the plots above until you get to the point at which you can explain what the viewpoint specification in a three-dimensional plot actually does. Then write up your own explanation of what you think the viewpoint specification in a three-dimensional plot actually does for you.

G.2) Measurements*

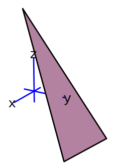
□G.2.a)

Measure the distance between {2, 7} and {3, −5} in two dimensions.
Measure the distance between {2, 7, −8} and {3, −5, 9} in three dimensions.

□G.2.b.i)

Here is a triangle sitting happily in three dimensions:

```
point1 = {0, 2, -4};
point2 = {-6/5, 41/10, -5/2};
point3 = {1, 0, 5};
triangle = Graphics3D[Polygon[{point1, point2, point3}]];
threedims = Axes3D[2, 0.2];

Show[threedims, triangle, ViewPoint → CMView, PlotRange → All,
 Boxed → False];
```

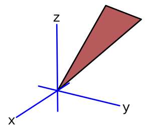


That triangle sure looks like a right triangle.
Use a dot product to confirm or dispel this observation.

□G.2.b.ii)

Mrs. Stephens is one of those sly math teachers who uses Mathematica in her office but doesn't allow the students to use computers or calculators for class work. In fact, she doesn't even let her students know that she even has a computer.
In preparation for yet another captivating lecture, she needs to generate a right triangle in three dimensions with one vertex at {0, 0, 0}.
She locks her office and gets out her computer and types:

```
X = {1, 2, 3};
Y = {0, 1, 1};
point1 = {0, 0, 0};
point2 = X;
point3 = (X.Y)/(Y.Y) Y;
triangle = Graphics3D[Polygon[{point1, point2, point3}]];
threedims = Axes3D[2, 0.2];

Show[threedims, triangle, ViewPoint → CMView, PlotRange → All,
 Boxed → False];
```



She knows in advance that, unless X and Y are parallel vectors, the triangle with vertices (corners) at {0, 0, 0}, the tip of

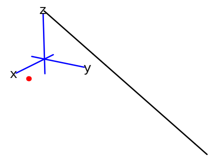
$\frac{X.Y}{Y.Y}$ Y (with tail at {0, 0, 0})

and the tip of X (with tail at {0, 0, 0}) will always give her a right triangle.
Why does this always work?

□G.2.c)

Here is a plot of a line and a point:

```
Clear[L, t]
L[t_] = {-1.3, 5.6, -1.8} + t {1.2, 7.3, -3.6};
point = {2.4, 0.7, -0.2};
lineplot =
 ParametricPlot3D[L[t], {t, -1, 1/2}, DisplayFunction → Identity];
threedims = Axes3D[2.5, 0.2];
Show[threedims, lineplot,
 Graphics3D[{Red, PointSize[0.02], Point[point]}], ViewPoint → CMView,
 PlotRange → All, Boxed → False, DisplayFunction → $DisplayFunction];
```



Measure the shortest distance between the line and the point.

□Tip:

The distance between L[t] and the point is measured by:

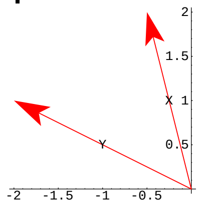
```
dist[t_] = √((L[t] - point) . (L[t] - point))
√((-1.6 - 3.6 t)^2 + (-3.7 + 1.2 t)^2 + (4.9 + 7.3 t)^2)
```

G.3) With or against?*

□G.3.a.i)

Here are two vectors X and Y:

```
X = {-0.5, 2};
Y = {-2, 1};
Show[Arrow[X, Tail → {0, 0}, VectorColor → Red],
 Arrow[Y, Tail → {0, 0}, VectorColor → Red],
 Graphics[{{Text["X", X/2]}, {Text["Y", Y/2]}}], Axes → Automatic,
 AspectRatio → Automatic];
```

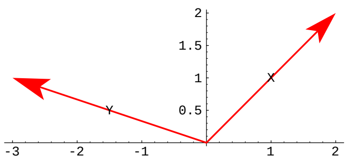


How does the picture reveal with no calculation that X . Y > 0?
Confirm what you say by calculating X . Y.
Add to the plot the vector that measures the push of X in the direction of Y.
Would you say that the push of X in the direction of Y is with Y or against Y?

□G.3.a.ii)

Here are two new vectors X and Y:

```
X = {2, 2};
Y = {-3, 1};
Show[Arrow[X, Tail → {0, 0}, VectorColor → Red],
 Arrow[Y, Tail → {0, 0}, VectorColor → Red],
 Graphics[{{Text["X", X/2]}, {Text["Y", Y/2]}}], Axes → Automatic,
 AspectRatio → Automatic];
```



How does the picture reveal with no calculation that X . Y < 0?
Confirm what you say by calculating X . Y.
Add to the plot the vector that measures the push of X in the direction of Y.
Would you say that the push of X in the direction of Y is with Y or against Y?

□G.3.a.iii)

Given any two vectors X and Y, how does the sign of the dot product X . Y tell you whether the push of X in the direction of Y is with Y or against Y?

□G.3.b)

At time t with $0 \le t \le 2\pi$, Luke Skywalker is at the point

{x[t], y[t]} = {Cos[t], 3 Sin[t]}.

Darth Vader has activated a force field that puts a force (= push or pull) equal to {x, −y} on any object at {x, y}.
At time t:
→ Is the force pushing with or against Skywalker's movement if
{x[t], −y[t]} . {x′[t], y′[t]} < 0?
→ Is the force pushing with or against Skywalker's movement if
{x[t], −y[t]} . {x′[t], y′[t]} > 0?
For approximately which t's is Vader's force with young Skywalker?

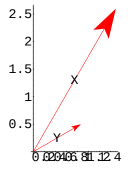
□Tip:

Plot {x[t], −y[t]} . {x′[t], y′[t]}.

□G.3.c)

Here is a pair of vectors.

```
X = 3 {Cos[π/3], Sin[π/3]};
Y = {Cos[π/6], Sin[π/6]};
Show[Arrow[X, Tail → {0, 0}, VectorColor → Red],
 Arrow[Y, Tail → {0, 0}, VectorColor → Red],
 Graphics[{{Text["X", X/2]}, {Text["Y", Y/2]}}], Axes → Automatic,
 AspectRatio → Automatic];
```



Come up with vectors U and V with:
→ U parallel to Y
→ V perpendicular to Y
→ X = U + V.
Once you have your vectors U and V, add them to the plot above by plotting U with its tail at {0, 0}, and V with its tail at the tip of U.
Describe what you see.

G.4) Velocity and acceleration*

□G.4.a)

At time t with $0 \le t \le 6$, an object is at the position

P[t] = {Cos[t], Cos[2 t] Sin[t]}.

Plot the curve and some of its velocity vectors and acceleration vectors at a selection of points.
Use your plot to analyze the direction and the speed of the object.
Then plot its speed as a function of t.

□G.4.b)

Ballistic projectiles (like a cannonball from a cannon) fired from the origin with muzzle velocity $v_0 \frac{\text{ft}}{\text{sec}}$ and angle b with the horizontal are at the position

$P[t] = \{v_0 \text{Cos}[b]\, t,\ v_0 \text{Sin}[b]\, t - 16 t^2\}$

t seconds after firing.
Take $v_0 = 180 \frac{\text{ft}}{\text{sec}}$ and $b = \frac{\pi}{3}$ and plot the trajectory, some of its velocity vectors, and some of its acceleration vectors at a selection of points.
Explain anything of interest you note.

□G.4.c)

For

$P[t] = \left\{2 \text{Sin}[t],\ 6 \text{Sin}[\frac{t}{2}]^2,\ 3 \text{Cos}[t]\right\}$,

plot in true scale the motion of the object for $1 \le t \le 6$, including several velocity vectors.
Then, on a separate plot, show the motion of the object for $1 \le t \le 6$, including several tangential components of acceleration.
On a third plot, show the motion of the object for $1 \le t \le 6$, including several tangential and normal components of acceleration.
Discuss what each plot reveals.

□G.4.d.i)

An object is at

$P[t] = \{9 \text{Sin}[t],\ 3 \text{Cos}[t]\}$

at time t.
Plot in true scale the motion of the object for $0 \le t \le \frac{2\pi}{3}$ including several velocity vectors.
Then, on a separate plot, show in true scale the motion of the object for $0 \le t \le \frac{2\pi}{3}$ including several tangential and normal components of acceleration.
Discuss what each plot reveals.
Finally, show the plots together and discuss relations between them.

□G.4.d.ii)

For $0 \le t \le \frac{2\pi}{3}$ the object in part i) is constrained to move on an ellipse. Suppose, at the instant $t = \frac{2\pi}{3}$ the object is released from all constraints and allowed to move of its own free will independent of any accelerations due to forces. Plot the path the object takes.

□Tip:

Does it continue along on the ellipse or does it do something else?

□G.4.d.iii)

The speed of the object in part i) at time t is defined to be the length of the velocity vector at time t; in other words

$\text{speed}[t] = \sqrt{\text{velocity}[t]\,.\,\text{velocity}[t]}$.

Put

$\text{unittan}[t] = \frac{\text{tan}[t]}{\sqrt{\text{tan}[t].\text{tan}[t]}}$

where

$\text{tan}[t] = \{x'[t],\ y'[t]\}$

Calculate D[speed[t], t] and compare it with
acceleration[t] . unittan[t].
Try to explain why you think that the result is natural or weird.

□G.4.d.iv)

If you know speed[0], then how do you give a formula for speed[t] in terms of acceleration[t] . unittan[t]?

□Tip:

Two-four-six-eight,

What do you integrate?

□G.4.d.v)

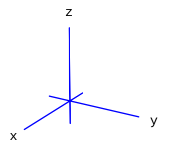
Can you get any information about speed[t] if you know the tangential component of acceleration but lose all your information about the normal component of acceleration?

G.5) The coordinate axes and coordinate planes in three dimensions*

```
If you have been around Calculus&Mathematica for a while, the beginning
          parts of this problem will be old hat to you.
```

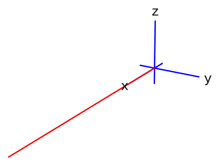
Here are the three coordinate axes in three dimesnsions:

```
threedims = Axes3D[1, 0.2];
Show[threedims, ViewPoint → CMView, Boxed → False];
```



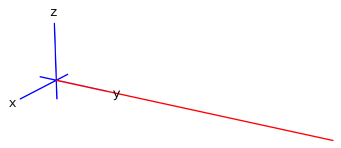
Here is the positive x-axis shooting out in the direction of the vector {1, 0, 0}:

```
xaxis = Graphics3D[{Red, Line[{{0, 0, 0}, {5, 0, 0}}]}];
Show[xaxis, threedims, ViewPoint → CMView, Boxed → False];
```



Here is the positive y-axis shooting out in the direction of the vector {0, 1, 0}:

```
yaxis = Graphics3D[{Red, Line[{{0, 0, 0}, {0, 5, 0}}]}];
Show[yaxis, threedims, ViewPoint → CMView, Boxed → False];
```



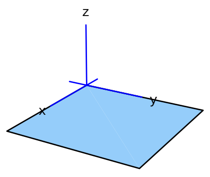
□G.5.a)

Give a plot of the positive z-axis shooting out in the direction of the vector {0, 0, 1}.
Then show all three axes in one plot.
Continue to use the option ViewPoint → CMView.

□G.5.b)

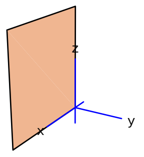
The xy-plane is the plane that you get by laying down a rigid flat sheet on the girders defined by the x-axis and the y-axis.
Here is a piece of it:

```
h = 2;
xyplane =
 Graphics3D[Polygon[{{0, 0, 0}, {h, 0, 0}, {h, h, 0}, {0, h, 0}}]];
Show[xyplane, threedims, ViewPoint → CMView, Boxed → False];
```



The xz-plane is the plane that you get by nailing a rigid flat sheet on the girders defined by the x-axis and the z-axis.
Here is a piece of it:

```
h = 2;
xzplane =
 Graphics3D[Polygon[{{0, 0, 0}, {0, 0, h}, {h, 0, h}, {h, 0, 0}}]];
Show[xzplane, threedims, ViewPoint → CMView, Boxed → False];
```

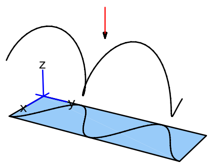


Show a piece of the yz-plane together with the coordinate vectors.
Continue to use the option ViewPoint → CMView.

□G.5.c)

Here is a curve and its shadow in the xz-plane cast by light rays shining from top to bottom, all parallel to the z-axis.

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

curve = ParametricPlot3D[
  {x[t], y[t], z[t]}, {t, 0, 2 π}, DisplayFunction → Identity];
samplelightray = Arrow[{0, 0, -1}, Tail → {1, π, 4}, VectorColor → Red];
h = 2;
xyplane =
 Graphics3D[Polygon[{{0, 0, 0}, {h, 0, 0}, {h, π h, 0}, {0, π h, 0}}]];
shadowonxy = ParametricPlot3D[
  {x[t], y[t], 0}, {t, 0, 2 π}, DisplayFunction → Identity];

Show[curve, samplelightray, xyplane, shadowonxy,
 threedims, ViewPoint → CMView, Boxed → False, Axes → None,
 DisplayFunction → $DisplayFunction];
```



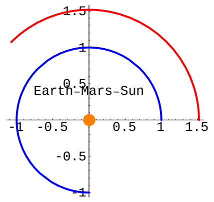
Plot the same curve and its shadow in the xz-plane cast by light rays shining from right to left, and all parallel to the y-axis.
Then plot the same curve and its shadow of the yz-plane cast by light rays shining from front to back, and all parallel to the x-axis.
Continue to use the option ViewPoint → CMView.

G.6) Serious plotting: Parametric planets

```
        This problem was adapted from an article by
     Professor Donald Saari of Northwestern University
   in the American Mathematical Monthly, February, 1990.
```

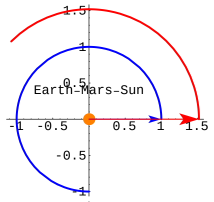
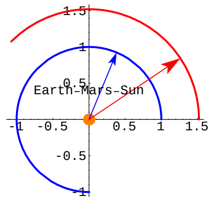
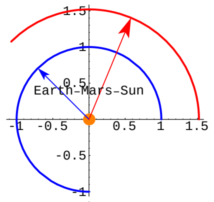
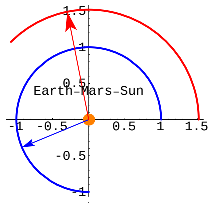
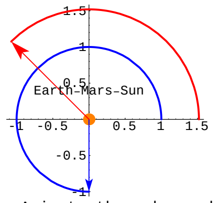
Many persons wonder why astronomers of antiquity had so much trouble predicting the paths of the planets. The truth is that their view that the Earth was the center of the solar system made the job of charting the motion of the other planets very, very difficult.
To see why, take a look at a simplified version of the Sun-Earth-Mars system.
Here are some simplified data:
→ Both the Earth and Mars move on orbits that are nearly circular, and both orbits are in the same plane.
→ One astronomical unit is the distance from the Earth to the Sun.
→ Mars is about 1.52 times as far from the Sun as is the Earth.
→ Mars takes about 2 (Earth) years to orbit the sun; the Earth takes one year.
Setting the Sun at the origin, measuring distance in astronomical units, and measuring time t in Earth years with the Earth and Mars in alignment on the x-axis when t = 0, you can give a pleasing plot of the motion during the first nine months of the first year.

```
Clear[earth, mars, t]
earth[t_] = {Cos[2 π t], Sin[2 π t]};
mars[t_] = 1.52 {Cos[2 π t/2], Sin[2 π t/2]};
orbits = ParametricPlot[{earth[t], mars[t]}, {t, 0, 9/12},
  PlotStyle → {{Thickness[0.01], Blue}, {Thickness[0.01], Red}},
  AspectRatio → Automatic, Epilog → Text["Earth-Mars-Sun", {0, 0.4}],
  DisplayFunction → Identity];
sun = Graphics[{RGBColor[1, 0.5, 0], PointSize[0.06], Point[{0, 0}]}];
ninemonths = Show[orbits, sun, DisplayFunction → $DisplayFunction];
```



That's the sun in the center, Mars on the outside track, and Earth on the inside track.
Here's a little movie that illustrates the action:

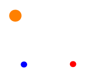
```
Clear[pointers, t]
pointers[t_] := {Arrow[earth[t], Tail → {0, 0}, VectorColor → Blue],
  Arrow[mars[t], Tail → {0, 0}, VectorColor → Red]};
Table[Show[ninemonths, pointers[t]], {t, 0, 9/12, 9/(12 4)}];
```

```
Animate these by grabbing all the plots and selecting the animation
                instruction in the Cell menu.
              After the animation begins, press 3.
```

Here's a little movie that shows the movement of Earth and Mars:

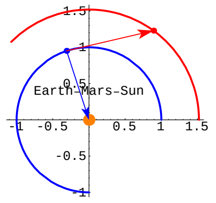
```
Clear[earthplotter, marsplotter, t]
earthplotter[t_] :=
 Graphics[{Blue, PointSize[0.03], Point[earth[t]]}];
marsplotter[t_] := Graphics[{Red, PointSize[0.03], Point[mars[t]]}];
jump = 2/9;
Table[Show[sun, earthplotter[t],
  marsplotter[t], PlotRange → {{-2, 2}, {-2, 2}}],
 {t, 0, 2 - jump, jump}];
```



```
Animate these by grabbing all the plots and selecting the animation
                  instruction in the Cell menu.
                 After the anination begins, press 3.
```

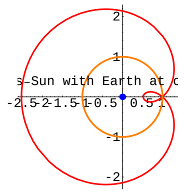
To see why the oldtimers who insisted that the Earth is at the center of the solar system had some tough explaining to do, you look at a plot of the movement of Mars and the Sun with the Earth plotted at {0, 0}.
To see how to do this, look at:

```
Clear[t, earthtosun, earthtomars]
earthtosun[t_] = {0, 0} - earth[t];
earthtomars[t_] = mars[t] - earth[t];
t = 0.3;
Show[ninemonths, marsplotter[t], earthplotter[t],
 Arrow[earthtosun[t], Tail → earth[t], VectorColor → Blue],
 Arrow[earthtomars[t], Tail → earth[t], VectorColor → Red]];
```



Change t and rerun.
To plot the motion of Mars and the Sun with the earth at {0, 0}, you just put the tails of the vectors earthtomars[t] and earthtosun[t] at {0, 0} and plot the curves traced out by their tips as t advances from 0.
Here's what you get for the first two years:

```
Clear[t]
ParametricPlot[{earthtomars[t], earthtosun[t]}, {t, 0, 2}, PlotStyle →
  {{Thickness[0.01], Red}, {Thickness[0.01], RGBColor[1, 0.5, 0]}},
 AspectRatio → Automatic,
 Epilog → {Text["Earth-Mars-Sun with Earth at center", {-1, 0.4}],
   {Blue, PointSize[0.04], Point[{0, 0}]}}];
```



The sun goes around the Earth in a nice circle, but look at the catchy dance Mars is doing. No wonder the ancient astronomers had a lot of trouble predicting and explaining where Mars was going.

□G.6.a)

Explain the presence of the number 2 in the denominators inside the original parametric formula

$$\text{mars}[t] = 1.52\,\{\text{Cos}[2\pi\,\tfrac{t}{2}],\ \text{Sin}[2\pi\,\tfrac{t}{2}]\}.$$

□G.6.b)

Set Mars at {0, 0} and plot the motion of Earth for the first two years.

□G.6.c)

→ Jupiter sits 5.20 times as far from the Sun as does the Earth.
→ It also moves in a (nearly) circular orbit in the plane of the Earth's orbit.
→ Jupiter takes about 12 Earth years to complete one trip around the Sun.
Setting the Earth at {0, 0}, measuring distance in astronomical units, and measuring time t in Earth years with the Earth and Jupiter in alignment on the x-axis when t = 0, give a plot of the motion of Jupiter during the first five Earth years.

□G.6.d)

Comment on the following sentiment:
The old timers were not wrong in saying that the Sun goes around the Earth, but their insistence that this is the only way to look at the solar system was wrong and held back science for many years.

□Historical marker:

```
 The Polish astromoner, Nicolaus Copernicus (1473-1543), successfully
advanced the theory that the solar system is greatly simplified if you
        put the Sun at the center. Coperinicus's idea so shook
      the thinking of its time that it even changed the language.
       Before Copernicus the word revolution meant going around.
        After Copernicus, the word revolution also came to mean
                 a sudden, radical, or complete change.
   C&M is all for revolutions - especially in the way mathematics is
                               learned.
```

□G.6.e) Orbiting around a moving object

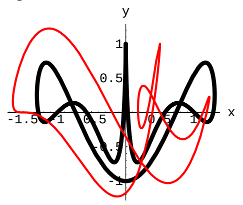
Just to see whether you got the idea, look at the following plots of two objects. At time t, object1 is at the point

$\{\text{Sin}[t]\,(1 - \text{Cos}[t]), \text{Cos}[t]\,\text{Cos}[4\,t]\}$

and object2 is at the point

$\{\text{Sin}[t]\,(1 - \text{Cos}[t]), \text{Cos}[t]\,\text{Cos}[4\,t]\} + 0.5\,\{\text{Cos}[t], \text{Sin}[t]\}$

```
Clear[object1, object2, t]
object1[t_] = {Sin[t] (1 - Cos[t]), Cos[t] Cos[4 t]};
object2[t_] = object1[t] + 0.5 {Cos[2 t], Sin[2 t]};

ParametricPlot[{object1[t], object2[t]}, {t, 0, 2 π},
 PlotStyle → {{Thickness[0.02]}, {Red, Thickness[0.01]}},
 AxesLabel → {"x", "y"}];
```



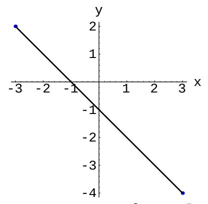
What a mess!
Plot the motion of object2 when you place object1 at {0, 0} and use what you see to explain what's happening.

G.7) Lines*

□G.7.a)

Here is the line segment running from {−3, 2} to {3, −4}.

```
point1 = {-3, 2};
point2 = {3, -4};
segment = Show[
  Graphics[{PointSize[0.02], Blue, Point[point1], Point[point2]}],
  Graphics[Line[{point1, point2}]], Axes → Automatic,
  AxesOrigin → {0, 0}, AxesLabel → {"x", "y"}];
```

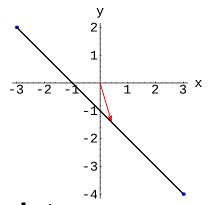


A parametric formula of the line through these two points is:

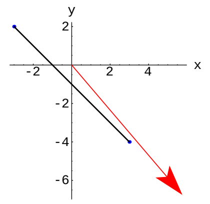
```
Clear[L, t]
L[t_] = point1 + t (point2 - point1)
{-3 + 6 t, 2 - 6 t}
```

What value of t makes L[t] land on point1?
What value of t makes L[t] land on point2?
What value of t makes L[t] land on the point halfway between point1 and point2 on the indicated segment?
What values of t make L[t] land on the indicated segment?
Illustrate with plots similar to these:

```
Show[segment, Arrow[L[0.57], Tail → {0, 0}, VectorColor → Red]];
```



```
Show[segment, Arrow[L[1.46], Tail → {0, 0}, VectorColor → Red]];
```



□G.7.b.i)

Write down a parametric formula for the line that passes through the points {2, 0} and {3, 4}.
Give a vector parallel to this line.
Give a vector perpendicular to this line.

□G.7.b.ii)

Write a parametric formula for the line that has xy-equation

$(y + 2) = 1.52\,(x - 1).$

Give a vector parallel to this line.
Give a vector perpendicular to this line.
Give a point on this line.

□G.7.b.iii)

Are the lines with parametric formulas

$L_1[t] = \{2, 3\} + t\,\{-3, 5\}$

and

$L_2[t] = \{2, 3\} + t\,\{6, -10\}$

the same line or different lines?
Are the lines with vector equations

$L_1[t] = \{2, 3\} + t\,\{-3, 5\}$

and

$L_2[t] = \{-4, 13\} + t\,\{-3, 5\}$

the same line or different lines?

□G.7.c.i)

Give a parametric formula for the line passing through the point {1, 4, 5} and moving away in the direction of the vector {2, 1, 1}.

□G.7.c.ii)

Give a parametric formula for the line in three dimensions through the tips of the vectors X = {2, 6, 2} and Y = {4, 2, 1} when their tails are at the origin.

□G.7.d)

Parametric equations for the line through {1, 2, 3} parallel to the vector {5, 6, −4} are

$x = 1 + 5\,t,$
$y = 2 + 6\,t,$ and
$z = 3 - 4\,t.$

A friend taking the old-fashioned course tells you that the xyz-equations for this line are

$\frac{x-1}{5} = \frac{y-2}{6} = \frac{z-3}{-4}.$

Your friend is right.
Why?

G.8) Lasers

Calculus&Mathematica thanks Todd Will
of Davidson College for suggesting this problem.

□G.8.a)

A Spartan missile and a Trojan missle are both flying at the same constant altitude.
At time t, the Spartan missile is at the point:

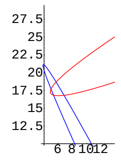
```
Clear[spartan, t]
spartan[t_] = {16.1 - 7 t + t^2, 13 t - 2 t^2}
```
```
{16.1 - 7 t + t^2, 13 t - 2 t^2}
```

At the same time t, the Trojan missile is at the point:

```
Clear[trojan]
trojan[t_] = {26 - 13 t + 2 t^2, 23 - 5 t + t^2}
```
```
{26 - 13 t + 2 t^2, 23 - 5 t + t^2}
```

Here is a plot of the paths of the two missiles:

```
spartanpath = ParametricPlot[spartan[t], {t, 0, 6},
  PlotStyle → {{Blue, Thickness[0.01]}}, DisplayFunction → Identity];
trojanpath = ParametricPlot[trojan[t], {t, 0, 6},
  PlotStyle → {{Red, Thickness[0.01]}}, DisplayFunction → Identity];

Show[spartanpath, trojanpath, DisplayFunction → $DisplayFunction];
```



Their paths cross, but do they crash?

□G.8.a.ii)

Continue with the set-up in part i) immediately above, but with additional information:
Each of the missiles has a laser in its nose that can shoot straight ahead and zap instantaneously.
Can either missile ever zap the other?
If so, who can zap whom and when?

□Tip:

Look at:

```
Clear[s]
Solve[trojan[t] + s D[trojan[t], t] == spartan[t]]
```
```
{{s → -0.627094, t → 4.46842}, {s → 0.408828, t → 1.76117},
 {s → 2.91827, t → 3.17041}}
```

If you use this, explain why it helps.

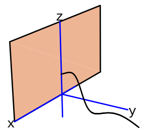
□G.8.b)

At time t ≥ 0, your new Luke Skywalker laser rocket scooter is at the position:

```
Clear[P, t]
P[t_] = {6 - t + Sin[t], 10 - 2 t, 2 + 0.5 Sin[2 t]}
```
```
{6 - t + Sin[t], 10 - 2 t, 2 + 0.5 Sin[2 t]}
```

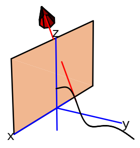
Here is its path for 0 ≤ t ≤ 5:

```
path = ParametricPlot3D[
  Evaluate[P[t]], {t, 0, 5}, DisplayFunction → Identity];
h = 6;
xzplane =
 Graphics3D[Polygon[{{-h, 0, 0}, {-h, 0, h}, {h, 0, h}, {h, 0, 0}}]];
threedims = Axes3D[6, .4];
setup =
 Show[threedims, path, xzplane, ViewPoint → CMView, Boxed → False,
  Axes → None, PlotRange → All, DisplayFunction → $DisplayFunction];
```



A laser beam emanates from the nose cone of your scooter and shoots out in in a straight line tangent to the path lkiie this:

```
time = 3.3;
samplebeam =
 Arrow[P′[time], Tail → P[time], VectorColor → Red, ScaleFactor → 4.5];
Show[setup, samplebeam];
```



Note that the beam pierces the xz-plane.
Imagine that the xz-plane is made of cardboard, and plot the curve burned into the xz-plane by your scooter's laser during the time interval 0 ≤ t ≤ 5.

□Tip:

Look at:

```
P[t] + s P′[t]
```
```
{6 - t + s (-1 + Cos[t]) + Sin[t], 10 - 2 s - 2 t, 2 + 1. s Cos[2 t] + 0.5 Sin[2 t]}
```
```
(P[t] + s P′[t])[[2]]
```
```
10 - 2 s - 2 t
```
```
Solve[(P[t] + s P′[t])[[2]] == 0, s]
```
```
{{s → 5 - t}}
```

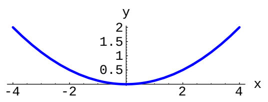
G.9) Parabolic reflectors, spherical reflectors, and elliptical reflectors

□G.9.a.i) Parabolic reflectors

Here is part of the parabola

$$f[x] = \frac{x^2}{8}:$$

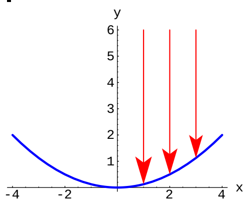
```
Clear[f, x]
f[x_] = x^2/8;
{x[t_], y[t_]} = {t, f[t]};

parabola = ParametricPlot[{x[t], y[t]}, {t, -4, 4},
  PlotStyle → {{Blue, Thickness[0.01]}}, AxesLabel → {"x", "y"}];
```



```
3
```
```
3
```

Three vertical light rays, emanating from
{x[1], 6}, {x[2], 6}, and {x[3], 6},
hit the parabola at the points
{x[1], y[1]}, {x[2], y[2]}, and {x[3], y[3]}
respectively:

```
Clear[beam]
beam[t_] :=
 Arrow[{x[t], y[t]} - {x[t], 6}, Tail → {x[t], 6}, VectorColor → Red];
Show[parabola, beam[1], beam[2], beam[3], PlotRange → All];
```



On one plot, show the paths the three beams take after they have bounced off the parabola. Make sure you show enough of the reflected beams to see where each beam crosses the y-axis.
Describe what you see, and mention anything interesting.

□G.9.a.ii)

You are given a positive number p, but you are not told what the specific value of p is.
A vertical light beam comes from high above the plot of

$$f[x] = \frac{x^2}{4p}$$

and bounces off the curve at a given point point {a, f[a]}. You are not told what the specific value of a is.
Calculate in terms of p the point at which the reflected beam crosses the y-axis.

□G.9.a.iii)

Most traditional reference books say that the focus of the parabola

$$f[x] = \frac{x^2}{4p}$$

is located at the point {0, p} on the vertical-axis. Why do they say this?
Why do folks use parabolas to build television satellite dish antennas to sell to rednecks in the boonies who want to watch wrestling and roller derby?
The great Greek scientist, Archimedes (287-212 B.C.), was the first scientist to understand what parabolic mirrors can do. In fact, Archimedes once used parabolic mirrors to concentrate sunlight on the sails of attacking Roman ships, thereby burning them up before they could attack. How do you think Archimedes went about this?

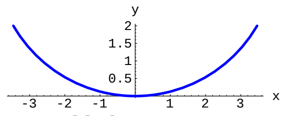
□Tip:

You'll have trouble with this if your answer to part ii) above isn't correct.

□G.9.b) Spherical reflectors

Here is part of the circle of radius 4 centered at {0, 4}:

```
Clear[x, y, t]
{x[t_], y[t_]} = {0, 4} + 4 {Cos[t], Sin[t]};
starter = 3π/2 - π/3;
stopper = 3π/2 + π/3;
circulardish = ParametricPlot[{x[t], y[t]}, {t, starter, stopper},
  PlotStyle → {{Blue, Thickness[0.01]}}, AspectRatio → Automatic,
  AxesLabel → {"x", "y"}];
```



Three vertical light rays, emanating from

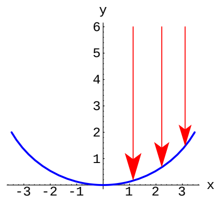
{x[5], 6}, {x[5.3], 6}, and {x[5.6], 6},

hit the circular dish at the points

{x[5], y[5]}, {x[5.3], y[5.3]}, and {x[5.6], y[5.6]},

respectively:

```
Clear[beam]
beam[t_] :=
 Arrow[{x[t], y[t]} - {x[t], 6}, Tail → {x[t], 6}, VectorColor → Red];

Show[circulardish, beam[5], beam[5.3], beam[5.6],
 AspectRatio → Automatic, PlotRange → All];
```



On one plot, show the path each of the three beams takes after it has bounced off the parabola. Make sure you show enough of the reflected beams to see where each beam crosses the y-axis.
Why don't spherical dish antennas work very well?

□G.5.c.i) Elliptical reflectors

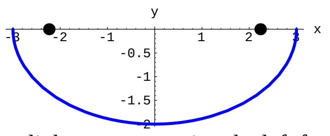
Here is the lower half of the ellipse

$$\left(\frac{x}{3}\right)^2 + \left(\frac{y}{2}\right)^2 = 1$$

shown with its two focuses at

$$\left\{\sqrt{3^2 - 2^2}, 0\right\} \text{ and } \left\{-\sqrt{3^2 - 2^2}, 0\right\}:$$

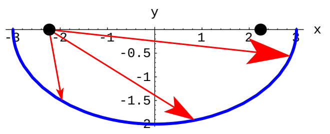
```
Clear[x, y, t]
a = 3;
b = 2;
{x[t_], y[t_]} = {a Cos[t], b Sin[t]};
starter = π;
stopper = 2 π;
leftfocus = {-√(a² - b²), 0};
rightfocus = {√(a² - b²), 0};

ellipticaldish = ParametricPlot[{x[t], y[t]},
  {t, starter, stopper}, PlotStyle → {{Blue, Thickness[0.01]}},
  AspectRatio → Automatic, AxesLabel → {"x", "y"},
  Epilog → {{PointSize[0.04], Point[leftfocus]},
    {PointSize[0.04], Point[rightfocus]}}];
```



Three light rays emanating the left focus hit the elliptical dish at the points

{x[4], y[4]}, {x[5], y[5]}, and {x[6], y[6]}:

```
Clear[beam]
beam[t_] :=
 Arrow[{x[t], y[t]} - leftfocus, Tail → leftfocus, VectorColor → Red];

Show[ellipticaldish, beam[4], beam[5], beam[6], PlotRange → All];
```



On one plot, show the path each of the three beams takes after it has bounced off the parabola. Make sure you show enough of the reflected beams to see where each beam crosses the x-axis.
Describe what you see.

□G.9.c.ii)

The ceiling of the rotunda in the United States Capitol building in Washington D.C. is in the form of an elliptical dish. Tourists are often surprised that they can sometimes hear very clearly what strangers well across the room are saying.
Use what you did above to explain this spooky phenom.

□G.9.c.iii)

Lewis Carroll (actual name Charles Dodgson, 1832 - 1898) was one far-out fellow. In addition to writing the famous tale of "Alice in Wonderland," he was an accomplished mathematician and logician.
As a prank, he invented an elliptical pool table with a mark at one focus and a hole at the other focus and challenged expert pool players to sink more balls in the holes than he.
What advantage did he have over those who didn't know about reflecting properties of ellipses?

□True Tale:

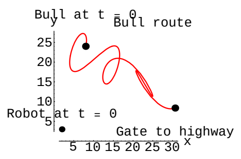
One of the authors of C&M once found an elliptical pool table in the back corner pool hall and challenged others to play. The games on the elliptical pool table were the only games he won that night.

G.10) Pursuits by a robotic cowhand

Calculus&*Mathematica* thanks cattleman Thomas O. Smith of Homer, Illinois for some help with this problem.

The scene is the C&M Electronic Ranch, where all the cattle are prize winners and where electronic robots do all the work.
One night, the prize bull breaks out of the pen, meanders around the ranch grounds, and then heads for the gate to the highway on the path plotted below:

```
Clear[bull, t]
bull[t_] = 4 {1 + 0.1 t + Cos[0.3 t], 6 - 0.05 t + Sin[0.4 t]};

bullroute = ParametricPlot[bull[t], {t, 0, 60},
  PlotStyle → {{Thickness[0.01], Red}}, AxesLabel → {"x", "y"},
  DisplayFunction → Identity, Epilog → Text["Bull route", {25, 30}]];
pointplot = {Graphics[{PointSize[0.06], Point[bull[0]]}],
  Graphics[{PointSize[0.06], Point[bull[60]]}],
  Graphics[{PointSize[0.05], Point[{2, 3}]}]};
labels = {Graphics[Text["Bull at t = 0", bull[0], {0, -4}]],
  Graphics[Text["Gate to highway", bull[60], {0, 3}]],
  Graphics[Text["Robot at t = 0", {2, 3}, {0, -2}]]};

setup = Show[bullroute, pointplot, labels,
  PlotRange → All, AxesOrigin → {0, 0}, AspectRatio → Automatic,
  DisplayFunction → $DisplayFunction];
```



Here x and y are measured in yards and t is measured in seconds

At time t = 0, the robot is at position {2, 3} on the plot above.
The robot is programmed to move so that if the robot's position at time t is {x[t], y[t]}, then the robot's velocity vector at that time is

{x'[t], y'[t]} = r (bull[t] – {x[t], y[t]})

where r is a positive number yet to be determined.
This is good because:
→ The robot is always moving toward the bull.
→ The robot slows down when it gets near the bull so that the robot neither smashes into the bull nor stampedes the bull.

□G.10.a.i)

Given that the robot can lasso the bull anytime the robot gets within 4.5 yards of the bull, your job is to come up with a specific positive number r, as small as practical, so that the robot will be successful lassoing the bull before the bull gets to the gate to the highway.

□Tip:

One way to go about this problem is by trial and error with different r's.

The distance between the bull and the robot at time t is

$$\text{dist}[t] = \sqrt{(\text{bull}[t] - \{x[t], y[t]\}) . (\text{bull}[t] - \{x[t], y[t]\})}.$$

To check out a guess for a good r, you might want to plot dist[t].

□G.10.a.ii)

After you have settled on a good number r in part i), make a nice plot of the actual lassoing of the bull by the robot.

□Tip:

You can use

lassorad = Graphics[Circle[{a, b}, 4.5]]

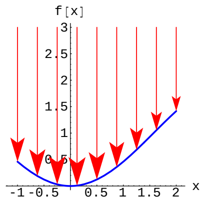
to plot a circle of radius 4.5 centered on {a, b}.

G.11) Stealth technology

□G.11.a)

Here is the curve y = 1 – Cos[x] shown with several vertical light beams:

```
Clear[f, x, t]
f[x_] = 1 - Cos[x];
x[t_] = t;
y[t_] = f[t];

curve = ParametricPlot[{x[t], y[t]},
  {t, -1, 2}, PlotStyle → {{Thickness[0.01], Blue}},
  AxesLabel → {"x", "f[x]"}, DisplayFunction → Identity];
beams = Table[Arrow[{x[t], y[t]} - {x[t], 3},
   Tail → {x[t], 3}, VectorColor → Red], {t, -1, 2, 3/8}];

setup = Show[curve, beams, DisplayFunction → $DisplayFunction];
```



Throw the plots of the reflected light into the plot, and use your plot to study the question:
Which parts of the curve plotted above are good at concentrating the reflected light? Which parts are not so good?

□G.11.b)

Stealth bombers and fighters were designed to try to resist detection by radar. When they were first unveiled, lots of folks asked why the skin of the planes is made with flat panels and absolutely no curved indentations. Look at your answer to part a), turn on your brain, and speculate about why stealth bombers and fighters are designed this way.