Using Vectors to Represent Lines and Find Distances

Introduction

OBJECTIVE: Visualize and interpret the use of vectors to represent lines in the plane.

Do you remember how you first learned about the equations of lines in a plane? In this module, you will gain insight into why it is to your advantage to interpret lines in the plane using vectors.

Technology Guidelines

NOTE: If you have just finished a module, restart *Mathematica* or close the *Kernel* before executing a new module.
TO OPEN CELLS, put your cursor on the right cell bracket and double click.
TO STOP AN EXECUTION
Select the *Kernel* pull-down menu and click on *Abort Evaluation.*
ORDER OF EXECUTION
Execute cells in the order given. Do not skip any Input cells within a given notebook.
SAVING NOTEBOOKS
You can save anytime to any directory you choose, and it is wise to save often.
However, before you do your final save, it is a good idea to delete all your output by selecting the
Delete All Output selection under the *Kernel* pull-down menu.
EXPERIENCING MAJOR PROBLEMS
Save if appropriate, then shut down *Mathematica* and start it up again.

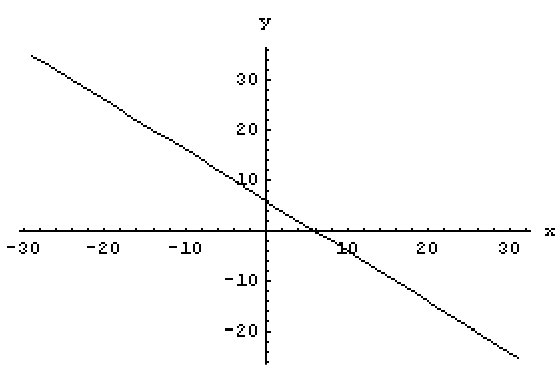
Part I: Interpreting Lines Using Their Vector Definition

You can construct straight lines in two dimensions using the vector definition. Given two points on the line, you will first determine the direction of the line and then write any other point on the line as the position vector to that point plus a multiple (t) of the vector in the direction of the line. We plot points along the line corresponding to different values of the parameter *t*. Our example uses points (1,5) and (4,2) and we plot points for values of *t* from -10 to 10.

In[1]:=

```
Off[General::spell]

Off[General::spell1]

Clear[x, y, t]

origin = {0, 0};

p1 = {1, 5};

p2 = {4, 2};

Print["The direction of the line is ", dir =

Print["{x,y} = ", eq = (p1 - origin) + t dir]

ParametricPlot[Evaluate[eq], {t, -10, 10}, Ax
```

```
The direction of the line is {-3, 3}

{x,y} = {1 - 3 t, 5 + 3 t}
```



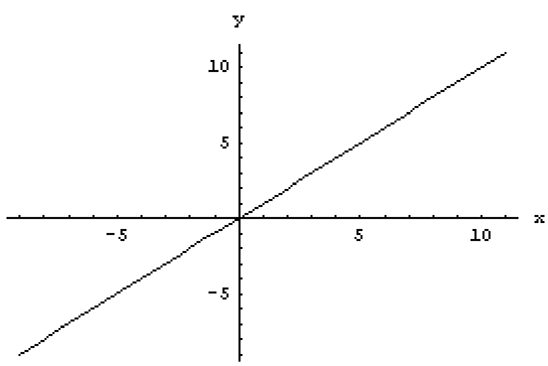
You Try It - Part I

Write and plot the line connecting the points (-23, -5) and (10, 12) in parametric form. You need to identify and enter the new points for *p1* and *p2* on the line and re-execute the input cell. Simply replace the numbers in red.

In[10]:=

```
origin = {0, 0};
p1 = {1, 1};
p2 = {2, 2};
Print["The direction of the line is ", dir =

Print["{x,y} = ", eq = (p1 - origin) + t dir]

plot1 = ParametricPlot[Evaluate[eq], {t, -10,
  AxesLabel -> {x, y}];
```

```
The direction of the line is {-1, -1}

{x,y} = {1 - t, 1 - t}
```

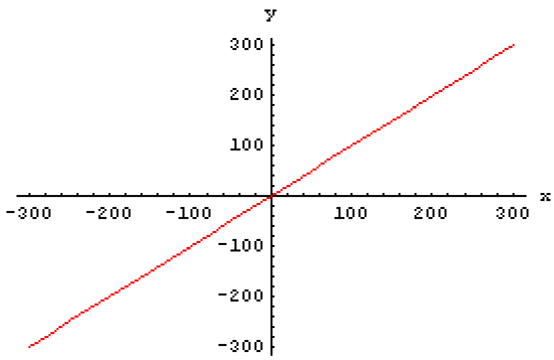


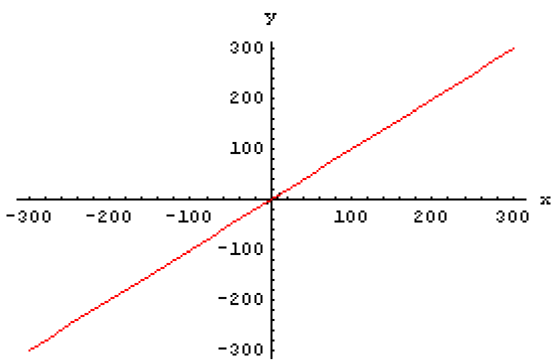
For the preceding function, match the parametric plot with the standard Cartesian form and plot. You need to solve for *y* as a function of *x* and put that into the expression for f[x_]. Replace the expression in red with the appropriate expression.

In[13]:=

```
f[x_] := x

pf = Plot[f[x], {x, -300, 300}, PlotStyle -> RG
  AxesLabel -> {"x", "y"}];
```

```
Show[plot1, pf];
```





Part II: Different Parametric Forms for the Same Line

Write the equation of the line y = 3x - 5 in parametric form.
You may be accustomed to seeing equations of lines in the standard slope intercept form. That form is a unique representation. In contrast, the parametric form can appear in different forms; however, the set of points represented is the same.

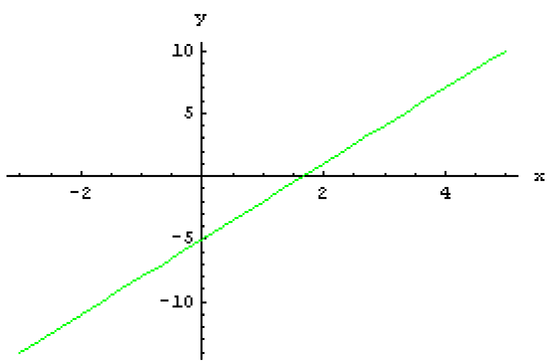
If we take the direction vector from the slope to be (1,3) and use the *y*-intercept (0,5) as our given point on the line, we get the following.

In[16]:=

```
Clear[x, f]

origin = {0, 0};

dir = {1, 3};
```

```
p1 = {0, -5};

Print["{x,y} = ", eq = (p1 - origin) + t dir]

plot1 = ParametricPlot[Evaluate[eq], {t, -3, !
  AxesLabel -> {x, y}, PlotStyle -> RGBColor[0
```

```
{x,y} = {t, -5 + 3 t}
```

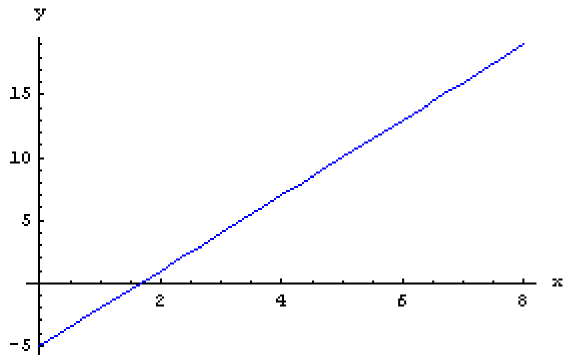


Had we used a different point on the line, for example, (2,1), and written the slope as (2, 6), the equation would look a bit different, although it would produce the same line.

In[22]:=

```
Clear[x, f]

dir = {2, 6};

p2 = {2, 1};

Print["{x,y} = ", eq = (p2 - origin) + t dir]

plot2 = ParametricPlot[Evaluate[eq], {t, -1, :
  AxesLabel -> {x, y}, PlotStyle -> RGBColor[0
```

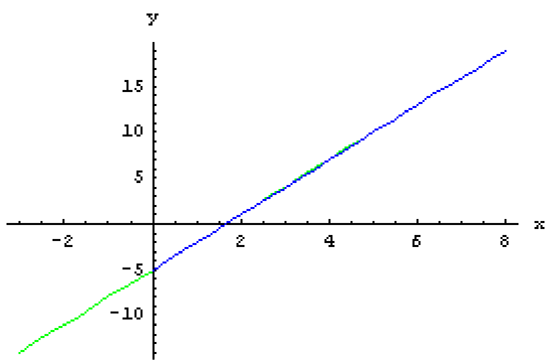
```
{x,y} = {2 + 2 t, 1 + 6 t}
```



We can look at these graphs together.

In[27]:=

```
Show[plot1, plot2];
```



Note that the parameter *t* is not the same in the two equations. However, all the points lie on the same line, even though we are plotting only segments of that line.

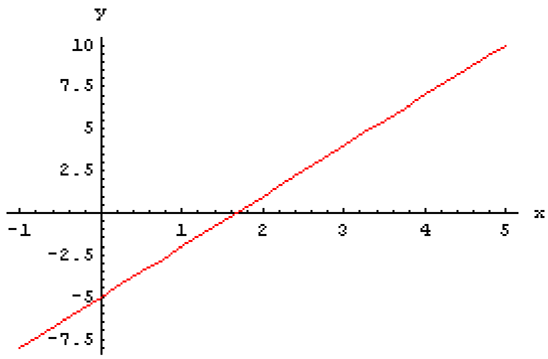
For the preceding function, match the parametric plot with the standard Cartesian form and plot.

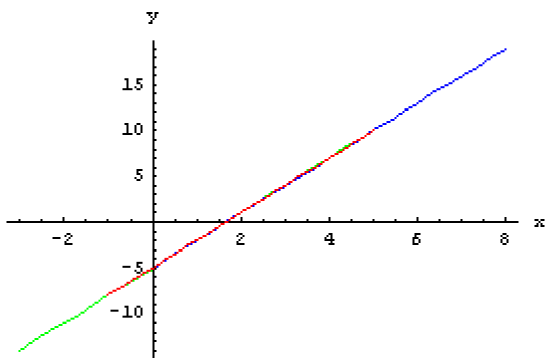
In[28]:=

```
f[x_] := 3 x - 5
```

```
pf = Plot[f[x], {x, -1, 5}, PlotStyle -> RGBCol
  AxesLabel -> {"x", "y"}];
```

```
Show[plot1, plot2, pf];
```





Do they match?

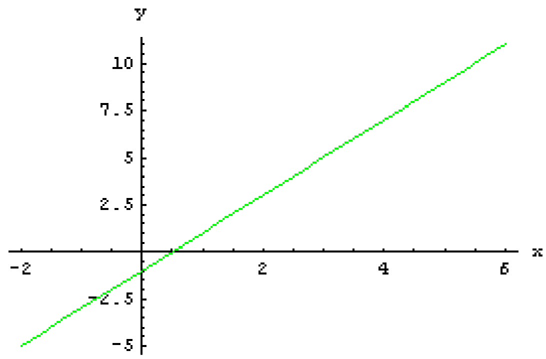
You Try It - Part II

Write the line 4x + 5y = 12 in parametric form.
You need to identify a point on the line and a vector in the direction of the line. Substitute these for **dir** and **p1** (items in red), and re-execute the input cell.

In[31]:=

```
dir = {1, 2};
p1 = {1, 1};
Print["{x,y} = ", eq = (p1 - origin) + t dir]

plot1 = ParametricPlot[Evaluate[eq], {t, -3, !
  AxesLabel -> {x, y}, PlotStyle -> RGBColor[0
```

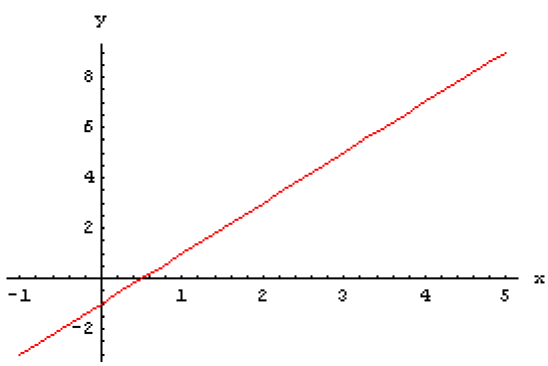
```
{x,y} = {1 + t, 1 + 2 t}
```

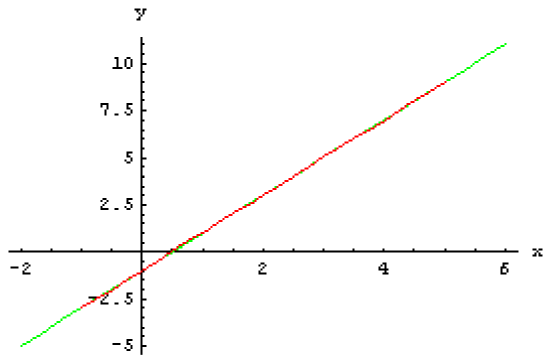


For the preceding function, match the parametric plot with the standard Cartesian form and plot.
You need to solve for *y* as a function of x and put that into the expression for f[x_]. Replace the red with the appropriate expression.

In[33]:=

```
f[x_] := 2 x - 1

pf = Plot[f[x], {x, -1, 5}, PlotStyle -> RGBCol
  AxesLabel -> {"x", "y"}];

Show[plot1, pf];
```





Part III: Finding the Distance from a Point to a Line

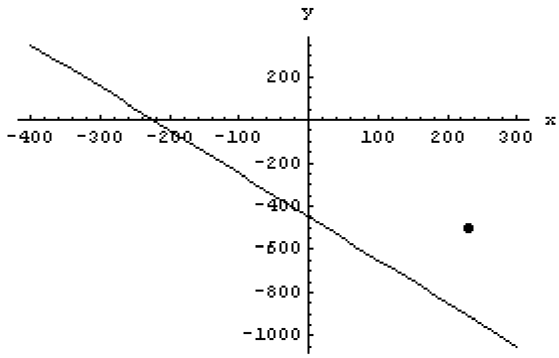
Suppose you are driving on a long, straight road on a flat planar region of desert land. You have a grid-map of the region and have been told that there is a radiation site in the at the point (230, -500). You started at the point (-400, 350) and are traveling in a path that is always in the direction of the vector (1, -2). You are worried about how close to the radiation site you will pass.

Let's begin by visualizing the problem.

In[36]:=

```
origin = {0, 0};

site = {230, -500};

start = {-400, 350};

dir = {1, -2};

Print["{x,y} = ", eq = (start - origin) + t dir]

pp = ParametricPlot[Evaluate[eq], {t, 0, 700},
  AxesLabel -> {x, y}, DisplayFunction -> Ident

rad = ListPlot[{site}, PlotStyle -> PointSize[
  DisplayFunction -> Identity];

Show[pp, rad, DisplayFunction -> $DisplayFunc
```

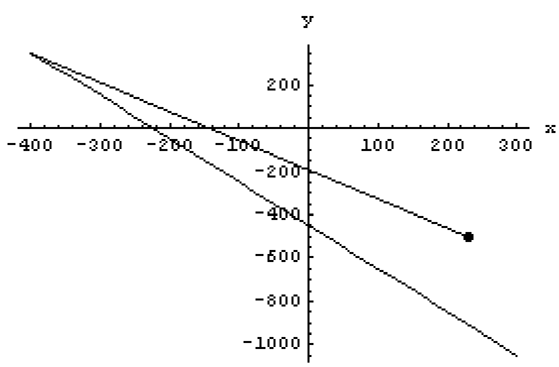
```
{x,y} = {-400 + t, 350 - 2 t}
```



Consider the vector from the starting point to the radiation site.

In[44]:=

```
vector = site - start;

vp = ListPlot[{start, site}, PlotJoined -> Tru
  DisplayFunction -> Identity];

Show[pp, rad, vp, DisplayFunction -> $DisplayI
```



Now, use the dot product to find the projection of this vector onto a direction perpendicular to the line. That will represent the shortest distance between the radiation site and the road.

Finding the perpendicular to a vector in two-space is very easy. Think of how perpendicular lines have slopes that are negative reciprocals of each other. You will find the perpendicular to the line by simply reversing the *x* and *y* components of the direction vector and negating one of them.

In[47]:=

```
perpdir = {-dir[[2]], dir[[1]]}
```

Out[47]=

```
{2, 1}
```

Then to find the projection of the vector connecting the starting point with the radiation site onto this perpendicular direction, use the dot product of those two vectors and divide by the magnitude of the perpendicular vector.

In[48]:=

```
distance = Abs[vector.perpdir] / √perpdir.pe
```

Out[48]=

```
183.358
```

You Try It - Part III

Find the shortest distance from the line 4x + 5y = 12 to the point (-2,3).
In the following set of commands, put in the new site, select a point on the line, and identify the direction of the line. To do this, just replace the red with the appropriate expressions.

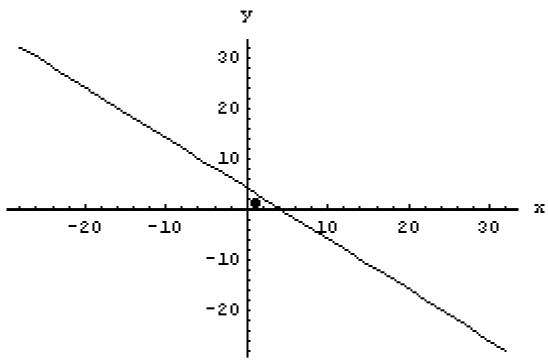
In[49]:=

```
origin = {0, 0};
site = {1, 1};
start = {2, 2};
dir = {-3, 3};
Print["{x,y}  = ", eq = (start - origin) + t dir]

pp = ParametricPlot[Evaluate[eq], {t, -10, 10
  AxesLabel -> {x, y}, DisplayFunction -> Ident

rad = ListPlot[{site}, PlotStyle -> PointSize[
  DisplayFunction -> Identity];

Show[pp, rad, DisplayFunction -> $DisplayFunc

vector = site - start;

perpdir = {-dir[[2]], dir[[1]]};
```

```
distance = Abs[vector.perpdir] / √perpdir.pe
```

```
Print["The shortest distance from the point
 distance]
```

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




```
The shortest distance from
   the point to the line is 1.41421
```

Created by *Mathematica* (April 2, 2005)