Matrices, Geometry&*Mathematica*

Authors: Bruce Carpenter, Bill Davis and Jerry Uhl ©2001

Producer: Bruce Carpenter

Publisher: Math Everywhere, Inc.

MGM.01 Perpendicular Frames

GIVE THEM A TRY!

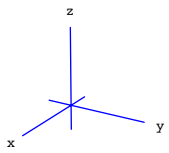
Experience with the starred problems will be very useful when you go into later lessons.

G.1) The x-y-z coordinate axes and x-y-z coordinate planes in three dimensions

□G.1.a) The x, y and z axes in 3D

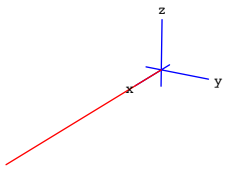
Here are the three coordinate axes in three dimensions:

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



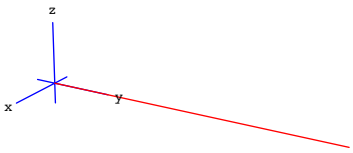
Here is the positive x-axis shooting out in the direction of the vector {1, 0, 0}:

```
xaxis = Graphics3D[{Red, Line[{{0, 0, 0}, {5, 0, 0}}]}];
Show[xaxis, Axes3D[1, 0.2], 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, Axes3D[1, 0.2], ViewPoint → CMView, Boxed → False];
```

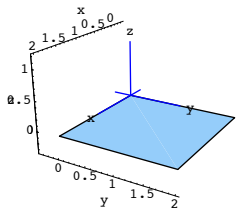


Copy,paste and edit to give a plot of the positive z-axis shooting out in the direction of the vector {0, 0, 1}.

□G.1.b) The xy plane, the xz plane and the yz plane

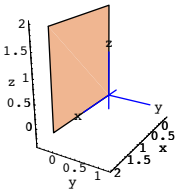
The xy-plane is the plane that you get by laying down a rigid flat sheet on the girders defined by the x-axis and the y-axis.
Here is a piece of it:

```
h = 2;
{slow, shigh} = {0, h};
{tlow, thigh} = {0, h};

Clear[s, t];
planeplot =
 ParametricPlot3D[ s {1, 0, 0} + t {0, 1, 0}, {s, slow, shigh},
  {t, tlow, thigh}, PlotPoints -> {2, 2}, DisplayFunction -> Identity];

Show[planeplot, Axes3D[1, 0.2], ViewPoint → CMView, Boxed → False,
  AxesLabel -> {"x", "y", "z"}, DisplayFunction -> $DisplayFunction];
```



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;
{slow, shigh} = {0, h};
{tlow, thigh} = {0, h};

Clear[s, t];
xyplaneplot =
 ParametricPlot3D[ s {1, 0, 0} + t {0, 0, 1}, {s, slow, shigh},
  {t, tlow, thigh}, PlotPoints -> {2, 2}, DisplayFunction -> Identity];

Show[xyplaneplot, Axes3D[1, 0.2], ViewPoint → CMView, Boxed → False,
  AxesLabel -> {"x", "y", "z"}, DisplayFunction -> $DisplayFunction];
```

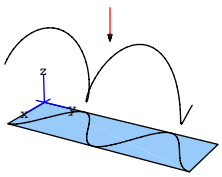


Copy, paste and edit to plot a piece of the yz-plane together with the coordinate axes.
Continue to use the option ViewPoint → CMView.

□G.1.c) Shadows

Here is a curve and its shadow in the xy-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, Axes3D[1], ViewPoint → CMView, Boxed → False,
  Axes → None, DisplayFunction → $DisplayFunction];
```



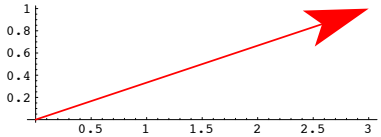
Plot the same curve and its shadow in the xz-plane cast by light rays shining from right to left, and all parallel to the y-axis.
Then plot the same curve and its shadow of the yz-plane cast by light rays shining from front to back, and all parallel to the x-axis.
Continue to use the option ViewPoint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

G.2) Just to be sure you get the idea*

□G.2.a.i) The tip

Here's 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];
```



What point is the tip of X sitting on?

□G.2.a.ii) Moving a vector

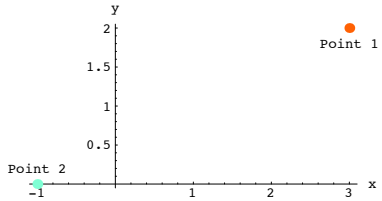
When you move the vector X = {3, 1} so that its tail is at {2, 2}, where do you find its tip?

□G.2.b) Joining two points with a vector

Here are two points:

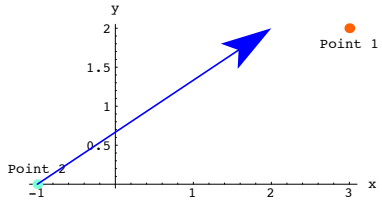
```
point1 = {3, 2};
point2 = {-1, 0};
labels = {Graphics[Text["Point 1", point1 - {0, 0.2}]],
          Graphics[Text["Point 2", point2 + {0, 0.2}]]};
pointplots =
  {Graphics[{CadmiumOrange, PointSize[0.03], Point[point1]}],
   Graphics[{Aquamarine, PointSize[0.03], Point[point2]}]};
```

```
setup = Show[pointplots, labels,
   PlotRange → All, Axes → True, AxesLabel → {"x", "y"}];
```



Here's a half-baked attempt at plotting a vector whose tail is at point2 and whose tip is at point1:

```
vector = {3, 2};
Show[setup, Arrow[vector, Tail -> point2]];
```

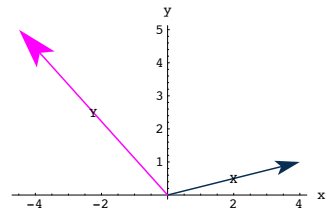


Change the plotted vector so that the tail of the plotted vector is at point2 and the tip of the vector is at point1.

□G.2.c) X, Y and X - Y

Here are vectors X and Y with their tails at {0, 0} in 2D:

```
X = {4, 1};
Y = {Random[Real, {-6, -1}], 5};
labels = Graphics[{{Text["X", X/2]}, {Text["Y", Y/2]}}];
setup = Show[Arrow[X, Tail → {0, 0}, VectorColor → Indigo], Arrow[Y,
    Tail → {0, 0}, VectorColor → Magenta], labels, PlotRange -> All,
   Axes → True, AxesLabel -> {"x", "y"}];
```

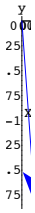


Throw in a plot of X - Y with its tail at the tip of Y.
When you do this, where does the tip of X − Y turn out to be?

□G.2.d.i) X and 2 X

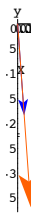
Here is a random vector X in 2D:

```
X = {Random[Real, {-2, 2}], Random[Real, {-2, 2}]};
Xplot = Show[Arrow[X, Tail → {0, 0}, VectorColor → Blue],
   Graphics[{Text["X", X/2]}], Axes → Automatic,
   AspectRatio → Automatic, AxesLabel -> {"x", "y"}];
```



Here is the same vector X shown with Y = 2.0 X:

```
Y = 2.0 X;
Show[Xplot, Arrow[Y, Tail → {0, 0}, VectorColor → CadmiumOrange],
  Graphics[{Text["Y = 2 X", 0.6 Y]}], Axes → Automatic,
  AspectRatio → Automatic, AxesLabel -> {"x", "y"}];
```

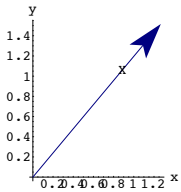


Describe what you see and explain why you see it.

□G.2.d.ii) X and 0.5 X

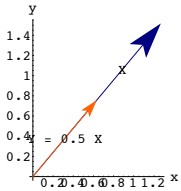
Here is a new random vector X in 2D:

```
X = {Random[Real, {-2, 2}], Random[Real, {-2, 2}]};
Xplot = Show[Arrow[X, Tail → {0, 0}, VectorColor → NavyBlue],
   Graphics[{Text["X", 0.7 X]}], Axes → Automatic,
   AspectRatio → Automatic, AxesLabel -> {"x", "y"}];
```



Here is the same vector X shown with
Y = 0.5 X:

```
Y = 0.5 X;
Show[Xplot, Arrow[Y, Tail → {0, 0}, VectorColor → CadmiumOrange],
  Graphics[Text["Y = 0.5 X", 0.5 Y]], Axes → Automatic,
  AspectRatio → Automatic, AxesLabel -> {"x", "y"}];
```

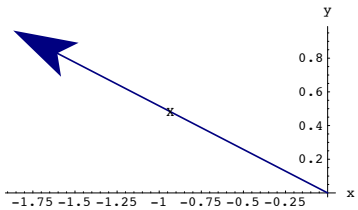


Describe what you see and explain why you see it.

□G.2.d.iii) X and - 2 X

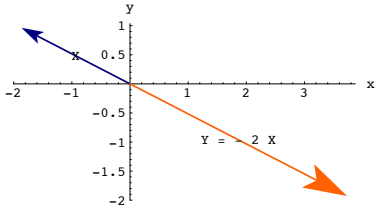
Here is yet another random vector X in 2D:

```
X = {Random[Real, {-2, 2}], Random[Real, {-2, 2}]};
Xplot = Show[Arrow[X, Tail → {0, 0}, VectorColor → NavyBlue],
   Graphics[{Text["X", 0.5 X]}], Axes → Automatic,
   AspectRatio → Automatic, AxesLabel -> {"x", "y"}];
```



Here is the same vector X shown with
Y = -2 X:

```
Y = -2 X;
Show[Xplot, Arrow[Y, Tail → {0, 0}, VectorColor → CadmiumOrange],
  Graphics[{Text["Y = - 2 X", 0.5 Y]}], Axes → Automatic,
  AspectRatio → Automatic, AxesLabel -> {"x", "y"}];
```

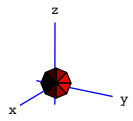


Describe what you see and explain why you see it.

□G.2.e) CMView

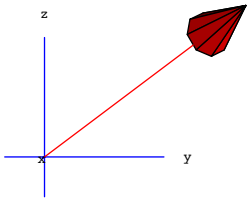
You can think of the CMView = {2.7, 1.6, 1.2} as a three-dimensional vector.
Here is this vector shown with ViewPoint→CMView:

```
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];
```

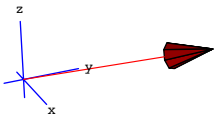


Gotcha!
Here is X = CMView from some other viewpoints:

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



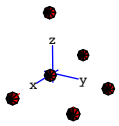

```
NewView = {2, -1, 1};
Show[Arrow[X, Tail → {0, 0, 0}, VectorColor → Red], threedims,
  PlotRange → All, ViewPoint → NewView, Boxed → False];
```



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];
```

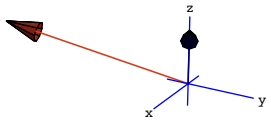


Duck before they pin you to your chair.
Play with the plots above until you get to the point at which you can explain what the viewpoint specification in a three-dimensional plot actually does. Then write up your own explanation of what you think the viewpoint specification in a three-dimensional plot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

□G.2.f.i) X.Y = 0

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 → NavyBlue],
  Arrow[Y, Tail → {0, 0, 0}, VectorColor → EnglishRed],
  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.2.f.ii.) {a,b} and {-b,a} are perpendicular

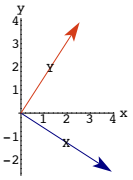
Here are two random numbers a and b:

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

{3.93218, -2.53703}

Here is a plot of the vectors X = {a,b} and Y = {-b,a}:

```
X = {a, b};
Y = {-b, a};
Show[Arrow[X, Tail → {0, 0}, VectorColor → NavyBlue],
  Arrow[Y, Tail → {0, 0}, VectorColor → EnglishRed],
  Graphics[{{Text["X", X/2]}, {Text["Y", Y/2]}}], Axes → Automatic,
  AxesLabel -> {"x", "y"}, AspectRatio → Automatic];
```



The two vectors look perpendicular.
Give them the acid test:

```
X.Y
```

0

They are perpendicular.
Part of the job of mathematics is to explain why things work out the way they do.

Explain this neat factoid:

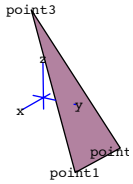
When you go any two numbers a and b (not both 0) and put
X = {a,b}
and
Y = {-b,a},
then you are guaranteed that X and Y are perpendicular.

□G.2.f.iii) Is it really a right triangle?

Here is a triangle sitting happily in three dimensions:

```
point1 = {0, 2, -4};
point2 = {-6/5, 41/10, -5/2};
```

```
point3 = {1, 0, 5};
labels = {Graphics3D[Text["point1", point1]],
   Graphics3D[Text["point2", point2 + 0.6 {1, 0, 0}]],
   Graphics3D[Text["point3", point3]]};
triangle = Graphics3D[Polygon[{point1, point2, point3}]];
threedims = Axes3D[2, 0.2];

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

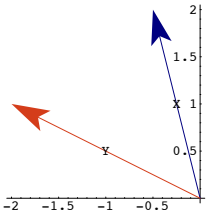


That triangle sure looks like a right triangle.
Use a dot product to confirm or dispel this observation.

□G.2.f.iv.) X.Y > 0

Here are two 2D vectors X and Y:

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

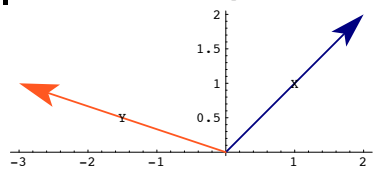


How does the picture reveal with no calculation that X.Y > 0?
Confirm what you say by calculating X.Y.

□G.2.f.v) X.Y < 0

Here are two new vectors X and Y:

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

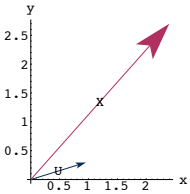


How does the picture reveal with no calculation that X.Y < 0?
Confirm what you say by calculating X.Y.

□G.2.g.i) A vector and a unit vector

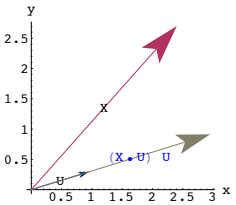
Here 's a vector X shown with a unit vector U:

```
X = {2.4, 2.7};
s = 0.3;
U = {Cos[s], Sin[s]};
setup = Show[Arrow[X, Tail → {0, 0}, VectorColor -> Maroon],
   Graphics[Text["X", X/2]],
  Arrow[U, Tail → {0, 0}, VectorColor -> Indigo],
   Graphics[Text["U", U/2]], AspectRatio → Automatic,
  Axes → True, AxesLabel → {"x", "y"}];
```



Here are the same two vectors shown with
(X .U) U:

```
newsetup = Show[setup,
    Arrow[(X.U) U, Tail → {0, 0}, VectorColor → WarmGray],
  Graphics[Text["(X.U) U", 0.6 (X.U) U]],
  AspectRatio → Automatic, Axes → True, AxesLabel → {"x", "y"}];
```

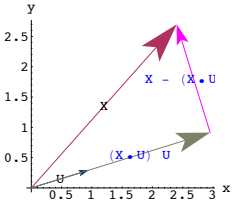


What is the significance of the plotted vector (X.U) U?

□G.2.g.ii) Perpendicular components

Here are the same two vectors shown with
X - (U. X) U

```
lastestsetup = Show[newsetup,
    Arrow[X - (X.U) U, Tail → (X.U) U, VectorColor → Magenta],
  Graphics[Text["X - (X.U) U", 0.5 (X + (X.U) U)]],
  AspectRatio → Automatic, Axes → True, AxesLabel → {"x", "y"}];
```



Notice that (X.U) U and X - (X.U) U are perpendicular:

```
(X.U) U.(X - (X.U) U)
```

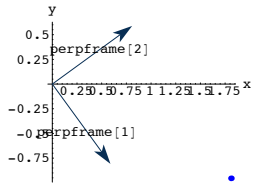
0

They are perpendicular.
Was this a fluke or was in guaranteed in advance?

□G.2.g.iii) A perpendicular frame and a random point

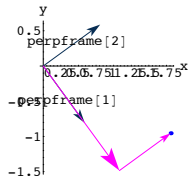
Here is a random perpendicular frame and a random point:

```
s = Random[Real, {-π/3, π/3}];
Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}};

t = s + Random[Real, {π/8, 0.4 π}];
point = Random[Real, {1.5, 2.5}] {Cos[t], Sin[t]};

setup = Show[Graphics[{Blue, PointSize[0.03], Point[point]}],
   Table[Arrow[perpframe[k], Tail → {0, 0},
     VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
   Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
   Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
   AspectRatio → Automatic, Axes → True,
   AxesLabel → {"x", "y"}, PlotRange → All];
```



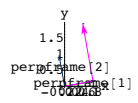
Here is the same thing with plots of
X1 = (point.perpframe[1]) perpframe[1];
X2 = (point.perpframe[2]) perpframe[2]
thrown in:

```
X1 = (point.perpframe[1]) perpframe[1];
X2 = (point.perpframe[2]) perpframe[2];

newsetup = Show[setup, Arrow[X1, Tail → {0, 0}, VectorColor → Magenta],
   Arrow[X2, Tail → X1, VectorColor → Magenta]];
```



Try it again for a new for a new point and a new perpendicular frame:

```
s = Random[Real, {-π/3, π/3}];
Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}};

t = s + Random[Real, {π/8, 0.4 π}];
point = Random[Real, {1.5, 2.5}] {Cos[t], Sin[t]};

X1 = point.perpframe[1] perpframe[1];
X2 = point.perpframe[2] perpframe[2];

Show[Graphics[{Blue, PointSize[0.03], Point[point]}],
  Table[Arrow[perpframe[k], Tail → {0, 0},
    VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
  Arrow[X1, Tail → {0, 0}, VectorColor → Magenta],
  Arrow[X2, Tail → X1, VectorColor → Magenta],
  Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
  Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
  AspectRatio → Automatic, Axes → True,
  AxesLabel → {"x", "y"}, PlotRange → All];
```

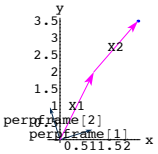


What do these plots depict?

□G.2.g.iv) Resolving into perpendicular components

Look at this:

```
s = 0.3;
Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}};

point = {2.3, 3.5};
X1 = {1, 2};
X2 = point - X1;

Show[Graphics[{Blue, PointSize[0.03], Point[point]}],
  Table[Arrow[perpframe[k], Tail → {0, 0},
    VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
  Arrow[X1, Tail → {0, 0}, VectorColor → Magenta],
  Arrow[X2, Tail → X1, VectorColor → Magenta],
  Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
  Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
  Graphics[Text["X1", 0.5 X1]], Graphics[Text["X2", X1 + 0.5 X2]],
  AspectRatio → Automatic, Axes → True,
  AxesLabel → {"x", "y"}, PlotRange → All];
```



Copy paste, edit and rerun so that X1 runs in the same direction as perpframe[1] and X2 runs in the same direction as perpframe[2].

□G.2.g.v) Getting there along the perpendicular frame vectors in 2D

Here is a random 2D perpendicular frame:

```
s = Random[Real, {-π/4, π/2}];
Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}}
```

```
{{0.456331, 0.88981}, {-0.88981, 0.456331}}
```

Here is a random 2D point:

```
point = {Random[Real, {-3, 3}], Random[Real, {-3, 3}]}
```

```
{2.36472, -0.74019}
```

Here is a calculation of
(point.perpframe[1]) perpframe[1] + (point.perpframe[2]) perpframe[2]:

```
(point.perpframe[1]) perpframe[1] + (point.perpframe[2]) perpframe[2]
```

```
{2.36472, -0.74019}
```

Rerun all three cells a couple of times.
Notice anything worth remarking on? If so, make your remark.

□G.2.g.vi) Getting there along the perpendicular frame vectors in 3D

Here is a random 3D perpendicular frame:

```
r = Random[Real, {-π/4, π/2}];
s = Random[Real, {-π/4, π/2}];
t = Random[Real, {-π/4, π/2}];

Clear[perpframe];
{perpframe[1], perpframe[2], perpframe[3]} =
  {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
    Cos[t] Sin[r] + Cos[r] Cos[s] Sin[t], Sin[s] Sin[t]},
   {-Cos[s] Cos[t] Sin[r] - Cos[r] Sin[t],
    Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[t] Sin[s]},
   {Sin[r] Sin[s], -Cos[r] Sin[s], Cos[s]}};

ColumnForm[{perpframe[1], perpframe[2], perpframe[3]}]
```

```
{-0.554488, 0.831223, -0.0401354}
{-0.831494, -0.555353, -0.0141612}
{-0.0340604, 0.0255201, 0.999094}
```

Here is a random 3D point:

```
point =
  {Random[Real, {-3, 3}], Random[Real, {-3, 3}], Random[Real, {-3, 3}]}
```

```
{2.36524, -0.749361, 1.53802}
```

Here is a calculation of
(point.perpframe[1]) perpframe[1] +
(point.perpframe[2]) perpframe[2] +
(point.perpframe[3]) perpframe[3] :

```
(point.perpframe[1]) perpframe[1] +
 (point.perpframe[2]) perpframe[2] +
 (point.perpframe[3]) perpframe[3]
```

```
{2.36524, -0.749361, 1.53802}
```

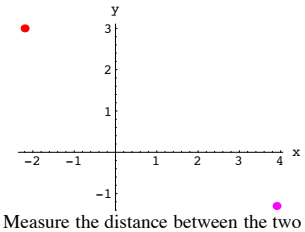
Rerun all three cells a couple of times.
What is going on here?

G.3) Measurements you need to know about*

□G.3.a.i) Measuring the distance between two points in 2D

Here are the points {−2.2, 3.0} and {3.9, −1.3} in 2D:

```
point1 = {-2.2, 3.0};
point2 = {3.9, -1.3};
setup = Show[Graphics[{Red, PointSize[0.03], Point[point1]}],
    Graphics[{Magenta, PointSize[0.03], Point[point2]}],
   Axes -> True, AxesLabel -> {"x", "y"}, AspectRatio -> Automatic];
```



Measure the distance between the two plotted points.

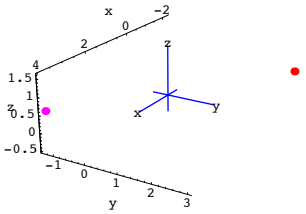
Click on the right for a tip.

The distance between these points is the same as the length of the vector running from one to the other.

□G.3.a.ii) Measuring the distance between two points in 3D

Here are the points {−2.2, 3.0, 0.6} and {3.9, −1.3, 0.6} in 3D:

```
point1 = {-2.2, 3.0, 0.6};
point2 = {3.9, -1.3, 0.6};
setup = Show[Graphics3D[{Red, PointSize[0.03], Point[point1]}],
    Graphics3D[{Magenta, PointSize[0.03], Point[point2]}],
   Axes3D[1.5, 0.1], Axes -> True, AxesLabel -> {"x", "y", "z"},
   Boxed -> False, BoxRatios -> Automatic, ViewPoint -> CMView];
```



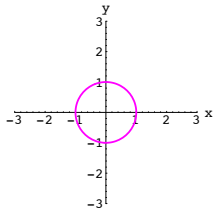
Measure the distance between the two plotted points.

Click on the right for a tip.

□G.3.b.i) Measuring the area enclosed by a 2D ellipse

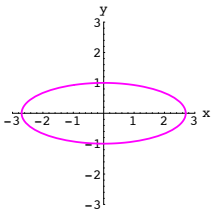
You plot the circle of radius 1 centered at {0,0} by plotting
{Cos[t], Sin[t]}
running t from 0 to 2 π:

```
Clear[t];
ranger = 3;
{tlow, thigh} = {0, 2 Pi};
    circleplot = ParametricPlot[{Cos[t], Sin[t]}, {t, tlow, thigh},
  PlotStyle -> {{Thickness[0.01], Magenta}}, AxesLabel -> {"x", "y"},
       PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
       AspectRatio -> Automatic];
```



Here's the ellipse you get by plotting
{xstretch Cos[t], ystretch Sin[t]}
running t from 0 to 2 π with xstretch = 2.7 and ystretch = 1.0:

```
Clear[t];
xstretch = 2.7;
ystretch = 1.0;
ellipseplot =
 ParametricPlot[{xstretch Cos[t], ystretch Sin[t]}, {t, tlow, thigh},
  PlotStyle -> {{Thickness[0.01], Magenta}}, AxesLabel -> {"x", "y"},
           PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
       AspectRatio -> Automatic];
```



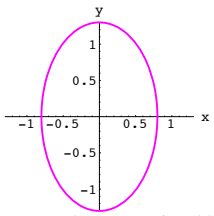
Grab both plots and animate.

Given that the area enclosed by the circle measures out to π square units, measure the area enclosed by the plotted ellipse.

☐G.3.b.ii) Measuring the area enclosed by another 2D ellipse

Here's the ellipse you get by plotting
{xstretch Cos[t], ystretch Sin[t]}
running t from 0 to 2 π with xstretch = 0.8 and ystretch = 1.3:

```
Clear[t];
xstretch = 0.8;
ystretch = 1.3;
ranger = Max[{xstretch, ystretch}];
{tlow, thigh} = {0, 2 Pi};
ellipseplot =
 ParametricPlot[{xstretch Cos[t], ystretch Sin[t]}, {t, tlow, thigh},
  PlotStyle -> {{Thickness[0.01], Magenta}}, AxesLabel -> {"x", "y"},
            PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
        AspectRatio -> Automatic];
```



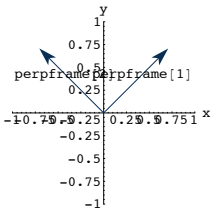
Measure the area enclosed by the plotted ellipse.

☐G.3.b.iii) Measuring the area enclosed by a 2D ellipse hung on a perpendicular frame

Here's a random 2D perpendicular frame:

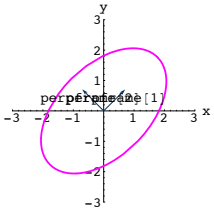
```
Clear[perpframe, s];
s = Random[Real, {0, 0.45 π}];
```

```
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, (-1)^Random[Integer,{0,1}] {Cos[s + π/2], Sin[s + π/2]}};

frameplot = Show[Table[Arrow[perpframe[k], Tail → {0, 0},
     VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
   Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
   Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
   PlotRange → {{-1, 1}, {-1, 1}}, Axes → True, AxesLabel → {"x", "y"}];
```



You can hang a nice ellipse on this perpendicular frame by plotting
xstretch Cos[t] perpframe[1] + ystretch Sin[t] perpframe[2]
running t from 0 to 2 π with xstretch = 2.5 and ystretch = 1.5:

```
{xstretch, ystretch} = {2.5, 1.5};
ranger = 3;

hungellipseplot = ParametricPlot[
  xstretch Cos[t] perpframe[1] + ystretch Sin[t] perpframe[2],
  {t, tlow, thigh}, PlotStyle -> {{Thickness[0.01], Magenta}},
  DisplayFunction -> Identity];
Show[hungellipseplot, frameplot, AxesLabel -> {"x", "y"},
            PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
        AspectRatio -> Automatic,
  DisplayFunction -> $DisplayFunction];
```

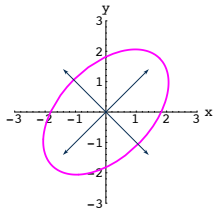


How long is the long axis of the ellipse?
How long is the short axis of the ellipse?
Measure the area enclosed by the plotted ellipse.

☐G.3.b.iv) Setting the factors

Stay with the ellipse in part iii) immediately above and look at this:

```
factor[1] = 2;
factor[2] = 2;

adjustedframeplot = {
      Arrow[factor[1] perpframe[1],
   Tail → {0, 0}, VectorColor → Indigo, HeadSize → 0.2],
      Arrow[-factor[1] perpframe[1], Tail → {0, 0},
   VectorColor → Indigo, HeadSize → 0.2],
      Arrow[factor[2] perpframe[2], Tail → {0, 0},
   VectorColor → Indigo, HeadSize → 0.2],
      Arrow[-factor[2] perpframe[2], Tail → {0, 0},
   VectorColor → Indigo, HeadSize → 0.2]};

Show[hungellipseplot, adjustedframeplot, AxesLabel → {"x", "y"},
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  AspectRatio → Automatic, DisplayFunction → $DisplayFunction];
```



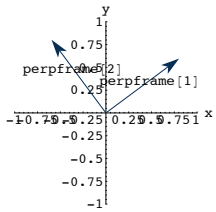
This plot shows the hung ellipse together with
factor[1] perpframe[1],
-factor[1] perpframe[1],
factor[2] perpframe[2],
and
-factor[2] perpframe[2].
Go back into the code and reset factor[1] and factor[2] so that the tips of the plotted arrows are right on the ellipse.

☐G.3.b.v) A collapsed 2D ellipse

Here's another random 2D perpendicular frame:

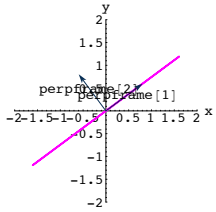
```
Clear[perpframe, s];
s = Random[Real, {0.2, 0.45 π}];
```

```
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, (-1)^Random[Integer,{0,1}] {Cos[s + π/2], Sin[s + π/2]}};

frameplot = Show[Table[Arrow[perpframe[k], Tail → {0, 0},
     VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
   Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
   Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
   PlotRange → {{-1, 1}, {-1, 1}}, Axes → True, AxesLabel → {"x", "y"}];
```



Look at what you get when you plot
xstretch Cos[t] perpframe[1] + ystretch Sin[t] perpframe[2]
running t from 0 to 2 π with xstretch = 2.0 and ystretch = 0.0:

```
{xstretch, ystretch} = {2.0, 0.0};
ranger = 2;
{tlow, thigh} = {0, 2 π};

hungellipseplot = ParametricPlot[
  xstretch Cos[t] perpframe[1] + ystretch Sin[t] perpframe[2],
  {t, tlow, thigh}, PlotStyle -> {{Thickness[0.01], Magenta}},
  DisplayFunction -> Identity];

Show[hungellipseplot, frameplot, AxesLabel -> {"x", "y"},
            PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
        AspectRatio -> Automatic,
  DisplayFunction -> $DisplayFunction];
```

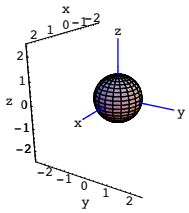


Describe what you see and explain why you see it.
How long is the plotted line segment?
Give a unit vector parallel to the plotted line segment.

□G.3.c.i) Measuring the volume enclosed by a 3D ellipsoid

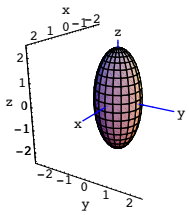
You plot the sphere of radius 1 centered at {0,0,0} by plotting
{Sin[s] Cos[t], Sin[s] Sin[t], Cos[s]}
running t from 0 to 2 π and running s from 0 to π:

```
Clear[s, t];
{slow, shigh} = {0, π};
{tlow, thigh} = {0, 2 π};

ranger = 2.5;
sphereplot = ParametricPlot3D[{Sin[s] Cos[t], Sin[s] Sin[t], Cos[s]},
  {s, slow, shigh}, {t, tlow, thigh}, PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False,
  ViewPoint → CMView, DisplayFunction → Identity];

Show[sphereplot, Axes3D[ranger], DisplayFunction → $DisplayFunction];
```



Here's the ellipsoid (football) you get by plotting
{xstretch Sin[s] Cos[t], ystretch Sin[s] Sin[t], zstretch Cos[s]}
running t from 0 to 2 π and running s from 0 to π with
xstretch = 1, ystretch = 1, and zstretch = 2.2:

```
xstretch = 1.0;
ystretch = 1.0;
zstretch = 2.2;

Clear[s, t];
{slow, shigh} = {0, π};
{tlow, thigh} = {0, 2 π};

ranger = 2.5;
ellipsoidplot = ParametricPlot3D[
  {xstretch Sin[s] Cos[t], ystretch Sin[s] Sin[t], zstretch Cos[s]},
  {s, slow, shigh}, {t, tlow, thigh}, PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False,
  ViewPoint → CMView, DisplayFunction → Identity];

Show[ellipsoidplot, Axes3D[ranger],
  DisplayFunction → $DisplayFunction];
```



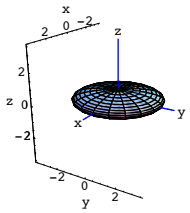
Grab both plots and animate.

Given that the volume enclosed by the sphere measures out to $\frac{4}{3}\pi$ cubic units, measure the volume enclosed by the plotted ellipsoid..

□G.3.c.ii) Measuring the volume enclosed by another 3D ellipsoid

Here's the ellipsoid (football) you get by plotting
{xstretch Sin[s] Cos[t], ystretch Sin[s] Sin[t], zstretch Cos[s]}
running t from 0 to 2 π and running s from 0 to π with
xstretch = 2.7, ystretch = 2.8 and zstretch = 0.6:

```
xstretch = 2.7;
ystretch = 2.8;
zstretch = 0.6;

Clear[s, t];
{slow, shigh} = {0, π};
{tlow, thigh} = {0, 2 π};

ranger = 1.3 Max[{xstretch, ystretch, zstretch}];
ellipsoidplot = ParametricPlot3D[
  {xstretch Sin[s] Cos[t], ystretch Sin[s] Sin[t], zstretch Cos[s]},
  {s, slow, shigh}, {t, tlow, thigh}, PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False,
  ViewPoint → CMView, DisplayFunction → Identity];

Show[ellipsoidplot, Axes3D[ranger],
  DisplayFunction → $DisplayFunction];
```

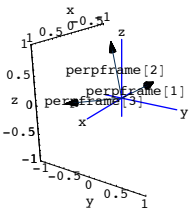


Measure the volume enclosed by the plotted ellipsoid.

□G.3.c.iii) Measuring the volume enclosed by a 3D ellipsoid hung on a perpendicular frame

Here's a random 3D perpendicular frame:

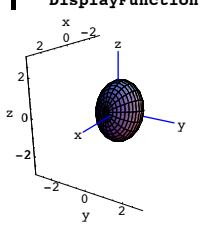
```
Clear[perpframe, r];
r = Random[Real, {π/4, π/2}];
s = Random[Real, {0, π/2}];
t = Random[Real, {-π/2, π/2}];

Clear[perpframe];
{perpframe[1], perpframe[2], perpframe[3]} =
  {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
    Cos[t] Sin[r] + Cos[r] Cos[s] Sin[t], Sin[s] Sin[t]},
   {-Cos[s] Cos[t] Sin[r] - Cos[r] Sin[t],
    Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[t] Sin[s]},
   {Sin[r] Sin[s], -Cos[r] Sin[s], Cos[s]}};

ranger = 1.0;
frameplot = Show[
   Table[Arrow[perpframe[k], Tail → {0, 0, 0}, VectorColor → Indigo],
    {k, 1, 3}], Graphics3D[Text["perpframe[1]", 0.5 perpframe[1]]],
   Graphics3D[Text["perpframe[2]", 0.5 perpframe[2]]],
   Graphics3D[Text["perpframe[3]", 0.5 perpframe[3]]],
   Axes3D[1], PlotRange → {{-ranger, ranger},
     {-ranger, ranger}, {-ranger, ranger}}, Boxed → False,
   Axes → True, ViewPoint → CMView, AxesLabel → {"x", "y", "z"}];
```



You can hang a nice ellipsoid on this perpendicular frame by plotting
xstretch Sin[s] Cos[t] perpframe[1] +
ystretch Sin[s] Sin[t] perpframe[2] +
zstretch Cos[s] perpframe[3]

running t from 0 to 2 π and running s from 0 to π with
xstretch = 2.1 and ystretch = 1.5 and zstretch = 0.7:

```
{xstretch, ystretch, zstretch} = {2.1, 1.5, 0.7};
ranger = 3;
{slow, shigh} = {0, Pi};
{tlow, thigh} = {0, 2 Pi};

hungellipsoidplot =
 ParametricPlot3D[ xstretch  Sin[s] Cos[t] perpframe[1] +
                   ystretch Sin[s] Sin[t] perpframe[2] +
                   zstretch Cos[s] perpframe[3],
  {s, slow, shigh}, {t, tlow, thigh},
    PlotRange -> {{-ranger, ranger},
    {-ranger, ranger}, {-ranger, ranger}},
    Axes -> True, AxesLabel -> {"x", "y", "z"},
    Boxed -> False,
  ViewPoint -> CMView, DisplayFunction -> Identity];

Show[hungellipsoidplot, Axes3D[ranger],
  DisplayFunction -> $DisplayFunction];
```

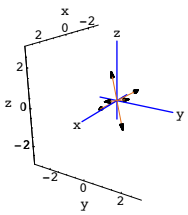


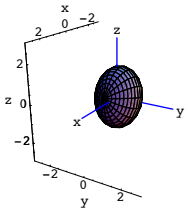
Measure the volume enclosed by the plotted ellipsoid.

□G.3.c.iv) Framing up the ellipsoid

Stay with the ellipsoid in part iii) immediately above and look at these two plots:

```
Clear[factor];
factor[1] = xstretch;
factor[2] = ystretch;
factor[3] = zstretch;

adjustedframeplot = {
      Arrow[factor[1] perpframe[1],
    Tail → {0, 0, 0}, VectorColor → MarsYellow, HeadSize → 0.3],
      Arrow[-factor[1] perpframe[1], Tail → {0, 0, 0},
    VectorColor → MarsYellow, HeadSize → 0.3],
      Arrow[factor[2] perpframe[2], Tail → {0, 0, 0},
    VectorColor → MarsYellow, HeadSize → 0.3],
      Arrow[-factor[2] perpframe[2], Tail → {0, 0, 0},
    VectorColor → MarsYellow, HeadSize → 0.3],
      Arrow[factor[3] perpframe[3], Tail → {0, 0, 0},
    VectorColor → MarsYellow, HeadSize → 0.3],
      Arrow[-factor[3] perpframe[3], Tail → {0, 0, 0},
    VectorColor → MarsYellow, HeadSize → 0.3]};

Show[adjustedframeplot, Axes3D[ranger], PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  AspectRatio → Automatic, ViewPoint → CMView, Boxed → False,
  Axes → True, AxesLabel → {"x", "y", "z"}];

Show[hungellipsoidplot,
  Axes3D[ranger], DisplayFunction → $DisplayFunction];
```



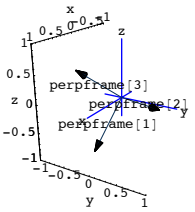


Grab both plots and animate.
Describe what you see and explain why you see it.

□G.3.c.v) A collapsed ellipsoid

Here's another random 3D perpendicular frame:

```
Clear[perpframe, r];
r = Random[Real, {π/4, π/2}];
s = Random[Real, {0, π/2}];
t = Random[Real, {-π/2, π/2}];

Clear[perpframe];
{perpframe[1], perpframe[2], perpframe[3]} =
  {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
     Cos[t] Sin[r] + Cos[r] Cos[s] Sin[t], Sin[s] Sin[t]},
    {-Cos[s] Cos[t] Sin[r] - Cos[r] Sin[t],
     Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[t] Sin[s]},
    {Sin[r] Sin[s], -Cos[r] Sin[s], Cos[s]}};

ranger = 1.0;
frameplot = Show[
   Table[Arrow[perpframe[k], Tail → {0, 0, 0}, VectorColor → Indigo],
     {k, 1, 3}], Graphics3D[Text["perpframe[1]", 0.5 perpframe[1]]],
   Graphics3D[Text["perpframe[2]", 0.5 perpframe[2]]],
   Graphics3D[Text["perpframe[3]", 0.5 perpframe[3]]],
   Axes3D[ranger], PlotRange → {{-ranger, ranger},
     {-ranger, ranger}, {-ranger, ranger}}, Boxed → False,
   Axes → True, ViewPoint → CMView, AxesLabel → {"x", "y", "z"}];
```

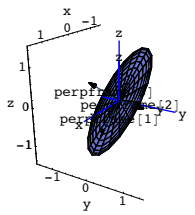


Look at what you get when you plot

xstretch Sin[s] Cos[t] perpframe[1] +
ystretch Sin[s] Sin[t] perpframe[2] +
zstretch Cos[s] perpframe[3]

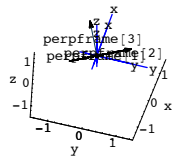
running t from 0 to 2 π and running s from 0 to π with
xstretch = 1.5 and ystretch = 0.8 and zstretch = 0:

```
{xstretch, ystretch, zstretch} = {1.5, 0.8, 0};
ranger = 1.5;
{slow, shigh} = {0, Pi};
{tlow, thigh} = {0, 2 Pi};

hungellipsoidplot =
 ParametricPlot3D[ xstretch  Sin[s] Cos[t] perpframe[1] +
                   ystretch Sin[s] Sin[t] perpframe[2] +
                   zstretch Cos[s] perpframe[3],
  {s, slow, shigh}, {t, tlow, thigh},
    PlotRange -> {{-ranger, ranger},
    {-ranger, ranger}, {-ranger, ranger}},
    Axes -> True, AxesLabel -> {"x", "y", "z"},
    Boxed -> False,
  ViewPoint -> CMView, DisplayFunction -> Identity];

setup = Show[hungellipsoidplot, frameplot,
   Axes3D[ranger], DisplayFunction -> $DisplayFunction];
```



Because zstretch = 0, this ellipsoid is totally flat:

```
Show[setup, ViewPoint -> 12 perpframe[1]];
```



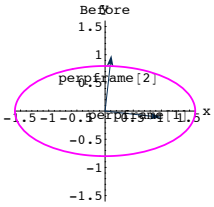
What plane does this ellipsoid plot out on?
Measure the planar area of this flat thing.

G.4) Hanging and aligning*

□G.4.a.i) Hanging a 2D ellipse on a perpendicular frame

Here is an ellipse in the usual position shown together with a perpendicular frame:

```
Clear[x, y, t];
{xstretch, ystretch} = {1.6, 0.8};
{x[t_], y[t_]} = {xstretch Cos[t], ystretch Sin[t]};

ranger = Max[{xstretch, ystretch}];
{tlow, thigh} = {0, 2 π};

curveplot = ParametricPlot[{x[t], y[t]},
  {t, tlow, thigh}, PlotStyle → {{Magenta, Thickness[0.01]}},
  DisplayFunction → Identity];
s = Random[Real, {-π/4, π/4}];

Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, {-Sin[s], Cos[s]}};

frameplot = {Table[Arrow[perpframe[k], Tail → {0, 0},
     VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
   Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
   Graphics[Text["perpframe[2]", 0.6 perpframe[2]]]};

before = Show[frameplot, curveplot,
   PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
   Axes → True, AxesLabel → {"x", "y"}, PlotLabel → "Before"];
```

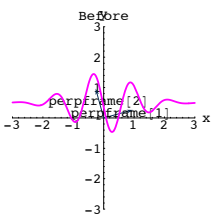


Pick this curve up and hang it on the plotted frame with
perpframe[1] playing the former role of {1,0} pointing along of the positive x-axis
and with
perpframe[2] playing the former role of {0,1} pointing along the positive y-axis.

□G.4.a.ii) Hanging another curve on a perpendicular frame

Here is a damped Sine wave in the usual position shown together with a perpendicular frame:

```
Clear[x, y, t];
{x[t_], y[t_]} = {t, E^(-0.5 t^2) Sin[5 t - 3] + 0.5};

ranger = 3;
{tlow, thigh} = {-3, 3};

curveplot = ParametricPlot[{x[t], y[t]},
  {t, tlow, thigh}, PlotStyle → {{Magenta, Thickness[0.01]}},
  DisplayFunction → Identity];
s = Random[Real, {0.1, 0.7}];

Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, {Cos[s + Pi / 2], Sin[s + Pi / 2]}};

frameplot = {Table[Arrow[perpframe[k], Tail → {0, 0},
     VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
   Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
   Graphics[Text["perpframe[2]", 0.6 perpframe[2]]]};

before = Show[frameplot, curveplot,
   PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
   Axes → True, AxesLabel → {"x", "y"}, PlotLabel → "Before"];
```

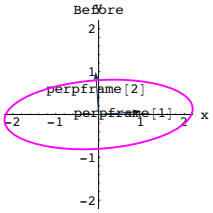


Pick this curve up and hang it on the plotted frame with
perpframe[1] playing the former role of {1,0} pointing along of the positive x-axis
and with
perpframe[2] playing the former role of {0,1} pointing along the positive y-axis.

□G.4.b.i) Aligning an ellipse on the x and y axes

Here is an ellipse skewered on a perpendicular frame.

```
Clear[perpframe];
s = Random[Real, {-Pi/2, Pi/2}];

{perpframe[1], perpframe[2]} = {{Cos[s], Sin[s]}, {-Sin[s], Cos[s]}};

Clear[x, y, t];
{x[t_], y[t_]} = 2.2 Cos[t] perpframe[1] + 0.8 Sin[t] perpframe[2];

ranger = 2.2;
{tlow, thigh} = {0, 2 π};
curveplot = ParametricPlot[{x[t], y[t]},
  {t, tlow, thigh}, PlotStyle → {{Magenta, Thickness[0.01]}},
  DisplayFunction → Identity];

frameplot = {Table[Arrow[perpframe[k], Tail → {0, 0},
     VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
   Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
   Graphics[Text["perpframe[2]", 0.6 perpframe[2]]]};

before = Show[frameplot, curveplot,
   PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
   Axes → True, AxesLabel → {"x", "y"}, PlotLabel → "Before",
   DisplayFunction → $DisplayFunction];
```

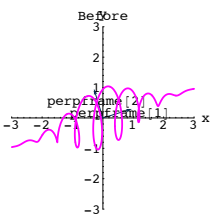


Pick this curve up and use perpendicular frame coordinates to align it on the usual x and y axes with
{1,0} pointing out the positive x-axis playing the former role of perpframe[1]
and with
{0,1} pointing out the positive y-axis playing the former role of perpframe[2].

□G.4.b.ii) Aligning another curve in 2D

Here's a whimsical curve shown with a perpendicular frame in 2D:

```
s = 0.3;
Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, {Cos[s + Pi/2], Sin[s + Pi/2]}};

Clear[x, y, t];
{x[t_], y[t_]} =
  {t - 0.3 E^(-0.5 t^2) Cos[3 π t], 0.32 t + 1.0 E^(-0.5 t^2) Sin[3 π t]};

ranger = 3;
{tlow, thigh} = {-3, 3};
curveplot = ParametricPlot[{x[t], y[t]},
  {t, tlow, thigh}, PlotStyle → {{Magenta, Thickness[0.01]}},
  DisplayFunction → Identity];

frameplot = {Table[Arrow[perpframe[k], Tail → {0, 0},
     VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
   Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
   Graphics[Text["perpframe[2]", 0.6 perpframe[2]]]};

before = Show[frameplot, curveplot,
   PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
   Axes → True, AxesLabel → {"x", "y"}, PlotLabel → "Before"];
```

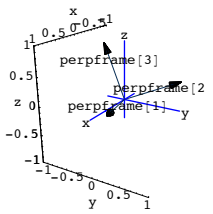


Pick this curve up use perpendicular frame coordinates to align it on the usual x and y axes with
{1,0} pointing out the positive x-axis playing the former role of perpframe[1]
and with
{0,1} pointing out the positive y-axis playing the former role of perpframe[2].

□G.4.c.i) Hanging an ellipsoid on a 3D perpendicular frame

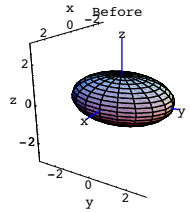
Here's a 3D perpendicular frame:

```
Clear[perpframe];
r = 0;
s = 0.4;
t = 0.3;
Clear[perpframe];
{perpframe[1], perpframe[2], perpframe[3]} =
  {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
    Cos[t] Sin[r] + Cos[r] Cos[s] Sin[t], Sin[s] Sin[t]},
   {-Cos[s] Cos[t] Sin[r] - Cos[r] Sin[t],
    Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[t] Sin[s]},
   {Sin[r] Sin[s], -Cos[r] Sin[s], Cos[s]}};

ranger = 1.0;
frameplot = Show[
   Table[Arrow[perpframe[k], Tail → {0, 0, 0}, VectorColor → Indigo],
    {k, 1, 3}], Graphics3D[Text["perpframe[1]", 0.4 perpframe[1]]],
   Graphics3D[Text["perpframe[2]", 0.7 perpframe[2]]],
   Graphics3D[Text["perpframe[3]", 0.7 perpframe[3]]],
   Axes3D[1, 0.1], PlotRange → {{-ranger, ranger},
     {-ranger, ranger}, {-ranger, ranger}}, Boxed → False,
   Axes → True, ViewPoint → CMView, AxesLabel → {"x", "y", "z"}];
```



Here's a fat ellipsoid (football) skewered on the x,y,and z axes:

```
Clear[x, y, z, s, t, pointcolor];
{x[s_, t_], y[s_, t_], z[s_, t_]} =
  {2.0 Sin[s] Cos[t], 2.7 Sin[s] Sin[t], 1.2 Cos[s]};

{slow, shigh} = {0, π};
{tlow, thigh} = {0, 2 π};
ranger = 3.0;
football = ParametricPlot3D[{x[s, t], y[s, t], z[s, t]},
  {s, slow, shigh}, {t, tlow, thigh}, PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False,
  ViewPoint → CMView, DisplayFunction → Identity];

Show[football, Axes3D[3, 0.2],
  PlotLabel → "Before", DisplayFunction → $DisplayFunction];
```

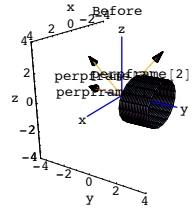


Hang this ellipsoid on the plotted perpendicular frame with
perpframe[1] playing the former role of {1,0,0} pointing along of the positive x-axis,
perpframe[2] playing the former role of {0,1,0} pointing along the positive y-axis
and with
perpframe[3] playing the former role of {0,0,1} pointing along the positive z-axis

□G.4.d.ii) Hanging a surface on a 3D perpendicular frame

Here's a surface shown together with a scaled 3D perpendicular frame:

```
Clear[x, y, z, s, t];
{x[s_, t_], y[s_, t_], z[s_, t_]} =
  s {2 Cos[t], 1, 1.5 Sin[t]} + (1 - s) {2 Cos[t], 3, 1.5 Sin[t]};

{tlow, thigh} = {0, 2 π};
{slow, shigh} = {0, 1};
ranger = 4;
surfaceplot = ParametricPlot3D[{x[s, t], y[s, t], z[s, t]},
  {s, slow, shigh}, {t, tlow, thigh}, DisplayFunction → Identity];

r = -0.3;
s = π/3;
t = π/6;
Clear[perpframe];
{perpframe[1], perpframe[2], perpframe[3]} =
  {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
    Cos[t] Sin[r] + Cos[r] Cos[s] Sin[t], Sin[s] Sin[t]},
   {-Cos[s] Cos[t] Sin[r] - Cos[r] Sin[t],
    Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[t] Sin[s]},
   {Sin[r] Sin[s], -Cos[r] Sin[s], Cos[s]}};

scaledframeplot = {Table[Arrow[ranger perpframe[k],
     Tail → {0, 0, 0}, VectorColor → LightCadmiumYellow], {k, 1, 3}],
   Graphics3D[Text["perpframe[1]", 0.5 ranger perpframe[1]]],
   Graphics3D[Text["perpframe[2]", 0.5 ranger perpframe[2]]],
   Graphics3D[Text["perpframe[3]", 0.5 ranger perpframe[3]]]};

before =
  Show[scaledframeplot, surfaceplot, Axes3D[ranger], PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   AspectRatio → Automatic, Axes → True, AxesLabel → {"x", "y", "z"},
   ViewPoint → CMView, PlotLabel → "Before",
   Boxed → False, DisplayFunction → $DisplayFunction];
```

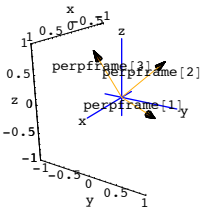


Hang this surface on the plotted perpendicular frame with
perpframe[1] playing the former role of {1,0,0} pointing along of the positive x-axis,
perpframe[2] playing the former role of {0,1,0} pointing along the positive y-axis
and with
perpframe[3] playing the former role of {0,0,1} pointing along the positive z-axis

□G.4.e.i) Aligning an ellipsoid on the x,y and z axes in 3D

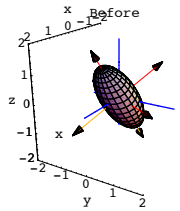
Here's a 3D perpendicular frame:

```
r = π/3;
s = π/6;
t = 0;
Clear[perpframe];
{perpframe[1], perpframe[2], perpframe[3]} =
  {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
    Cos[t] Sin[r] + Cos[r] Cos[s] Sin[t], Sin[s] Sin[t]},
   {-Cos[s] Cos[t] Sin[r] - Cos[r] Sin[t],
    Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[t] Sin[s]},
   {Sin[r] Sin[s], -Cos[r] Sin[s], Cos[s]}};

ranger = 1.0;
frameplot = Show[Table[Arrow[perpframe[k], Tail → {0, 0, 0},
     VectorColor → LightCadmiumYellow], {k, 1, 3}], Axes3D[ranger],
   Graphics3D[Text["perpframe[1]", 0.4 perpframe[1]]],
   Graphics3D[Text["perpframe[2]", 0.7 perpframe[2]]],
   Graphics3D[Text["perpframe[3]", 0.7 perpframe[3]]], PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   Boxed → False, Axes → True, ViewPoint → CMView,
   AxesLabel → {"x", "y", "z"}];
```



Here's a fat ellipsoid (football) skewered on the same perpendicular frame:

```
Clear[x, y, z, s, t, pointcolor];
{x[s_, t_], y[s_, t_], z[s_, t_]} = 0.9 Sin[s] Cos[t] perpframe[1] +
   0.6 Sin[s] Sin[t] perpframe[2] + 1.3 Cos[s] perpframe[3];

{slow, shigh} = {0, π};
{tlow, thigh} = {0, 2 π};
ranger = 2.0;
football = ParametricPlot3D[{x[s, t], y[s, t], z[s, t]},
  {s, slow, shigh}, {t, tlow, thigh}, PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False,
  ViewPoint → CMView, DisplayFunction → Identity];

scaledframeplot = {Table[Arrow[ranger perpframe[k],
     Tail → {0, 0, 0}, VectorColor → Red], {k, 1, 3}],
   Table[Arrow[-ranger perpframe[k], Tail → {0, 0, 0},
     VectorColor → LightCadmiumYellow], {k, 1, 3}]};

Show[football, Axes3D[3, 0.2], scaledframeplot,
  PlotLabel → "Before", DisplayFunction → $DisplayFunction];
```



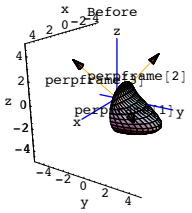
Pick up this surface and use perpendicular frame coordinates to align it on the usual x , y and z axes with
{1,0,0} pointing out the positive x-axis playing the former role of perpframe[1]
{0,1,0} pointing out the positive y-axis playing the former role of perpframe[2].
and with
{0,0,1} pointing out the positive y-axis playing the former role of perpframe[3].

□G.4.e.ii) Aligning a surface on the x,y and z axes in 3D

Here is a surface shown together with a 3D perpendicular frame:

```
r = π/4;
s = π/4;
t = 0;
Clear[perpframe];
{perpframe[1], perpframe[2], perpframe[3]} =
  {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
    Cos[t] Sin[r] + Cos[r] Cos[s] Sin[t], Sin[s] Sin[t]},
   {-Cos[s] Cos[t] Sin[r] - Cos[r] Sin[t],
    Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[t] Sin[s]},
   {Sin[r] Sin[s], -Cos[r] Sin[s], Cos[s]}};

Clear[x, y, z, s, t];
{x[s_, t_], y[s_, t_], z[s_, t_]} =
```

```
  {√2 + 2 Cos[t] Sin[s], √2 + 2 Sin[s] (1 - Sin[2 s]) Sin[t], 2 Cos[s]};
{tlow, thigh} = {0, π};
{slow, shigh} = {0, π};

ranger = 5;
surfaceplot = ParametricPlot3D[{x[s, t], y[s, t], z[s, t]},
  {s, slow, shigh}, {t, tlow, thigh}, DisplayFunction → Identity];

scaledframeplot = {Table[Arrow[ranger perpframe[k],
     Tail → {0, 0, 0}, VectorColor → LightCadmiumYellow], {k, 1, 3}],
   Graphics3D[Text["perpframe[1]", 0.5 ranger perpframe[1]]],
   Graphics3D[Text["perpframe[2]", 0.5 ranger perpframe[2]]],
   Graphics3D[Text["perpframe[3]", 0.5 ranger perpframe[3]]]};

before =
  Show[scaledframeplot, surfaceplot, Axes3D[ranger], PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   AspectRatio → Automatic, Axes → True, AxesLabel → {"x", "y", "z"},
   ViewPoint → CMView, PlotLabel → "Before",
   Boxed → False, DisplayFunction → $DisplayFunction];
```



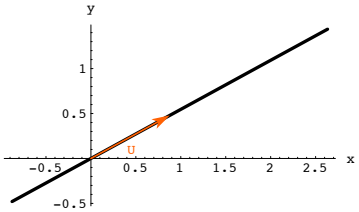
Pick up this surface and align it on the usual x , y and z axes with
{1,0,0} pointing out the positive x-axis playing the former role of perpframe[1]
{0,1,0} pointing out the positive y-axis playing the former role of perpframe[2].
and with
{0,0,1} pointing out the positive y-axis playing the former role of perpframe[3].

G.5) Closest points*

□G.5.a.i) Closest points on 2D lines through {0,0}

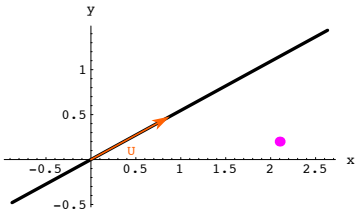
Here's a unit vector U in 2D shown with a plot of the line through {0,0} defined by this unit vector:

```
s = 0.5;
U = {Cos[s], Sin[s]};
unitvectorplot = Arrow[U, Tail → {0, 0},
   VectorColor → CadmiumOrange, HeadSize -> 0.2];

lineplot = ParametricPlot[t U, {t, -1, 3}, PlotStyle → Thickness[0.01],
  AxesLabel → {"x", "y"}, DisplayFunction → Identity];
Ulabel = Graphics[{CadmiumOrange, Text["U", 0.6 U, {1, 2}]}];

all = Show[lineplot, unitvectorplot,
   Ulabel, DisplayFunction → $DisplayFunction];
```



Throw in this point not on the line:

```
point = {2.1, 0.2};
pointplot = Graphics[{Magenta, PointSize[0.03], Point[point]}];

setup = Show[all, pointplot];
```

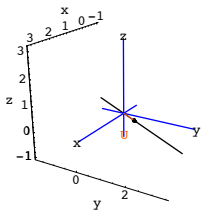


Come up with the point on the line closest to the plotted point and show it in a new plot.

□G.5.a.ii) On the 3D line or not on the line?

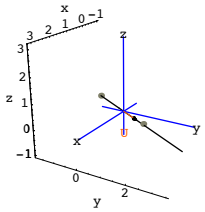
Look at this line through {0,0,0} in 3D:

```
s = Pi / 2;
t = Pi / 3;
U = {Sin[s] Cos[t], Sin[s] Sin[t], Cos[s]};
unitvectorplot = Arrow[U, Tail → {0, 0, 0},
   VectorColor → CadmiumOrange, HeadSize -> 0.2];
lineplot = ParametricPlot3D[t U, {t, -2, 4}, Axes -> True,
  AxesLabel → {"x", "y", "z"}, DisplayFunction → Identity];
Ulabel = Graphics3D[{CadmiumOrange, Text["U", 0.6 U, {1, 2}]}];

all = Show[lineplot, unitvectorplot, Ulabel,
  Axes3D[3, 0.1], Boxed → False, ViewPoint -> CMView,
  PlotRange -> All, DisplayFunction → $DisplayFunction];
```



Throw in these two new points:

```
point1 = {3/4, 3√3/4, 1/32};
point2 = {-1, -√3, 0};

point1plot = Graphics3D[{WarmGray, PointSize[0.03], Point[point1]}];
point2plot = Graphics3D[{WarmGray, PointSize[0.03], Point[point2]}];
setup = Show[all, point1plot, point2plot, PlotRange -> All];
```



Both points seem to be on the line.
The truth is that one of the points is on the line and the other is not.
Your job is to figure out which is which.

Take each point and find the point on the line closest to it.

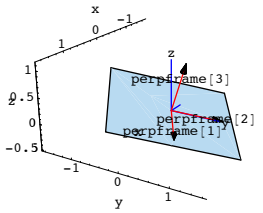
If the point itself is the point on the line closest to it, then that point is on the line.

If not, then that point is not really on the line.

□G.5.b.i) Closest points on 3D planes through {0,0,0}

Here's a perpendicular frame in 3D
{perpframe[1],perpframe[2],perpframe[3]}
and a look at part of the plane through {0,0,0} determined by
perpframe[1] and perpframe[2]:

```
r = 0.1;
s = -0.3;
t = 0.5;
Clear[perpframe];
{perpframe[1], perpframe[2], perpframe[3]} =
  {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
    Cos[t] Sin[r] + Cos[r] Cos[s] Sin[t], Sin[s] Sin[t]},
   {-Cos[s] Cos[t] Sin[r] - Cos[r] Sin[t],
    Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[t] Sin[s]},
   {Sin[r] Sin[s], -Cos[r] Sin[s], Cos[s]}};

frameplot = {Table[
  Arrow[perpframe[k],
     Tail -> {0, 0, 0}, VectorColor -> Red], {k, 1, 3}],
  Graphics3D[Text["perpframe[1]", 0.7 perpframe[1]]],
  Graphics3D[Text["perpframe[2]", 0.7 perpframe[2]]],
  Graphics3D[Text["perpframe[3]", 0.7 perpframe[3]]]};

Clear[u, v];
{ulow, uhigh} = {-1.2, 1.2};
{vlow, vhigh} = {-1.2, 1.2};

plane =
 ParametricPlot3D[u perpframe[1] + v  perpframe[2], {u, ulow, uhigh},
  {v, vlow, vhigh}, PlotPoints -> {2, 2}, DisplayFunction → Identity];

planeplot = Show[frameplot, plane, Axes3D[1],

  Axes -> True, ViewPoint -> CMView, AxesLabel -> {"x", "y", "z"},
   PlotRange → All, ViewPoint → CMView, Boxed -> False,
   BoxRatios → Automatic, DisplayFunction → $DisplayFunction];
```

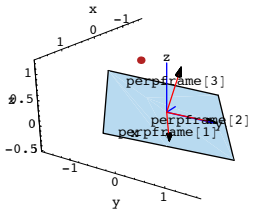


Here's a look at the same plane a point not on the plane:

```
0.5 perpframe[1] - 0.5 perpframe[2]
```

```
{0.694101, -0.121499, 0.0588317}
```

```
point = {0.694, -0.125, 1.225};


pointplot = Graphics3D[{Firebrick, PointSize[0.03], Point[point]}];


Show[planeplot, pointplot, PlotRange -> All];
```

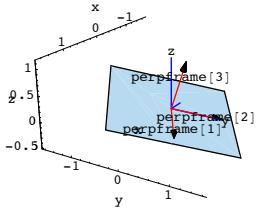


Use perpendicular components to come up with the point on the plane closest to the plotted point.

□G.5.b.ii) Are they or aren't they?

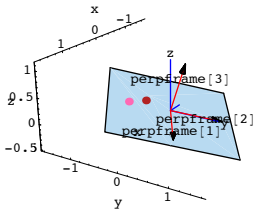
Here's a perpendicular frame in 3D
{perpframe[1],perpframe[2],perpframe[3]}
and a look at part of the plane through {0,0,0} determined by
perpframe[1] and perpframe[2]:

```
r = 0.1;
s = -0.3;
t = 0.5;
Clear[perpframe];
{perpframe[1], perpframe[2], perpframe[3]} =
  {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
    Cos[t] Sin[r] + Cos[r] Cos[s] Sin[t], Sin[s] Sin[t]},
   {-Cos[s] Cos[t] Sin[r] - Cos[r] Sin[t],
    Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[t] Sin[s]},
   {Sin[r] Sin[s], -Cos[r] Sin[s], Cos[s]}};

frameplot = {Table[
  Arrow[perpframe[k],
    Tail -> {0, 0, 0}, VectorColor -> Red], {k, 1, 3}],
  Graphics3D[Text["perpframe[1]", 0.7 perpframe[1]]],
  Graphics3D[Text["perpframe[2]", 0.7 perpframe[2]]],
  Graphics3D[Text["perpframe[3]", 0.7 perpframe[3]]]};

Clear[u, v];
{ulow, uhigh} = {-1.2, 1.2};
{vlow, vhigh} = {-1.2, 1.2};

plane =
 ParametricPlot3D[u perpframe[1] + v  perpframe[2], {u, ulow, uhigh},
  {v, vlow, vhigh}, PlotPoints -> {2, 2}, DisplayFunction → Identity];

planeplot = Show[frameplot, plane, Axes3D[1],

  Axes -> True, ViewPoint -> CMView, AxesLabel -> {"x", "y", "z"},
   PlotRange → All, ViewPoint → CMView, Boxed -> False,
   BoxRatios → Automatic, DisplayFunction → $DisplayFunction];
```



Here's a look at the same plane and two points:

```
point1 = {0.2774144, -0.363763, 0.225205};
point2 = {0.448584, -0.629068, 0.207475};

pointplots =
  {Graphics3D[{Firebrick, PointSize[0.03], Point[point1]}],
    Graphics3D[{HotPink, PointSize[0.03], Point[point2]}]};

Show[planeplot, pointplots, PlotRange -> All];
```



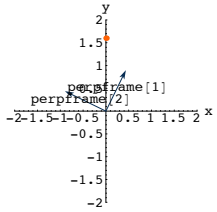
Both points seem to be on the plane. Are they really?

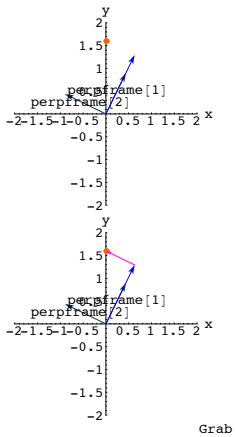
G.6) Perpendicular components and perpendicular frame coordinates*

□G.6.a.i) Resolution into perpendicular components

Run this a couple of times:

```
point = {Random[Real, {-2, 2}], 1.6};

s = Random[Real, {π/4, 3π/8}];

Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}};
setup = Show[Table[Arrow[perpframe[k], Tail → {0, 0},
     VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
   Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
   Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
   Graphics[{CadmiumOrange, PointSize[0.03], Point[point]}],
   Axes → True, AxesLabel → {"x", "y"}, PlotRange → {{-2, 2}, {-2, 2}}];
one = Show[setup,
  Arrow[(point.perpframe[1]) perpframe[1], Tail -> {0, 0},
                VectorColor -> Blue, HeadSize -> 0.2]];
both = Show[one, Arrow[(point.perpframe[2]) perpframe[2],
   Tail -> (point.perpframe[1]) perpframe[1], VectorColor -> Magenta]];
```





Grab and animate each time you run it.

Say what you think these plots depict.

□G.6.b.i) What's going on in 2D?

Enter a cleared 2D perpendicular frame:

```
Clear[perpframe, s, x, y];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}}
```

```
{{Cos[s], Sin[s]}, {-Sin[s], Cos[s]}}
```

Now enter a cleared 2D point {x,y} and calculate
({x,y} . perpframe[1]) perpframe[1] + ({x,y}. perpframe[2]) perpframe[2]

```
Clear[x, y];
calculation = ({x, y}.perpframe[1]) perpframe[1] +
   ({x, y}.perpframe[2]) perpframe[2]
```

```
{-Sin[s] (y Cos[s] - x Sin[s]) + Cos[s] (x Cos[s] + y Sin[s]),
 Cos[s] (y Cos[s] - x Sin[s]) + Sin[s] (x Cos[s] + y Sin[s])}
```

Apply trig identities:

```
TrigExpand[({x, y}.perpframe[1]) perpframe[1] +
   ({x, y}.perpframe[2]) perpframe[2]]
```

```
{x, y}
```

- What does this calculation illustrate?
- Try to describe in words the relationship between the location of {x,y} and the location of ({x,y}.perpframe[1]) perpframe[1]:

□ G.6.b.ii) What's going on in 3D?

Enter a cleared 3D perpendicular frame:

```
Clear[perpframe, r, s, t];
{perpframe[1], perpframe[2], perpframe[3]} =
 {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
   Cos[s] Cos[t] Sin[r] + Cos[r] Sin[t], Sin[r] Sin[s]},
  {-Cos[t] Sin[r] - Cos[r] Cos[s] Sin[t],
   Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[r] Sin[s]},
  {Sin[s] Sin[t], -Cos[t] Sin[s], Cos[s]}}
```

```
{{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
  Cos[s] Cos[t] Sin[r] + Cos[r] Sin[t], Sin[r] Sin[s]},
 {-Cos[t] Sin[r] - Cos[r] Cos[s] Sin[t],
  Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[r] Sin[s]},
 {Sin[s] Sin[t], -Cos[t] Sin[s], Cos[s]}}
```

Now enter a cleared 3D point {x,y,z} and calculate

({x,y,z} . perpframe[1]) perpframe[1] +

({x,y,z} . perpframe[2]) perpframe[2]

+ ({x,y,z}.perpframe[3]) perpframe[3]

```
Clear[x, y, z];
calculation = ({x, y, z}.perpframe[1]) perpframe[1] +
  ({x, y, z}.perpframe[2]) perpframe[2] +
  + ({x, y, z}.perpframe[3]) perpframe[3]
```

```
{Sin[s] Sin[t] (z Cos[s] - y Cos[t] Sin[s] + x Sin[s] Sin[t]) +
  (-Cos[t] Sin[r] - Cos[r] Cos[s] Sin[t])
   (z Cos[r] Sin[s] + x (-Cos[t] Sin[r] - Cos[r] Cos[s] Sin[t]) +
    y (Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t])) +
  (Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t])
   (z Sin[r] Sin[s] + y (Cos[s] Cos[t] Sin[r] + Cos[r] Sin[t]) +
    x (Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t])),
 -Cos[t] Sin[s] (z Cos[s] - y Cos[t] Sin[s] + x Sin[s] Sin[t]) +
  (Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t])
   (z Cos[r] Sin[s] + x (-Cos[t] Sin[r] - Cos[r] Cos[s] Sin[t]) +
    y (Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t])) +
  (Cos[s] Cos[t] Sin[r] + Cos[r] Sin[t])
   (z Sin[r] Sin[s] + y (Cos[s] Cos[t] Sin[r] + Cos[r] Sin[t]) +
    x (Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t])),
 Cos[s] (z Cos[s] - y Cos[t] Sin[s] + x Sin[s] Sin[t]) + Cos[r] Sin[s]
   (z Cos[r] Sin[s] + x (-Cos[t] Sin[r] - Cos[r] Cos[s] Sin[t]) +
    y (Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t])) +
  Sin[r] Sin[s] (z Sin[r] Sin[s] + y (Cos[s] Cos[t] Sin[r] + Cos[r] Sin[t]) +
    x (Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t]))}
```

Apply trig identities:

```
TrigExpand[({x, y, z}.perpframe[1]) perpframe[1] +
  ({x, y, z}.perpframe[2]) perpframe[2] +
  + ({x, y, z}.perpframe[3]) perpframe[3]]
```

```
{x, y, z}
```

▪ What does this calculation illustrate?

Put your answer here.

▪ Try to describe in words the relationship between the location of {x,y,z} and the location of

({x,y,z}.perpframe[1]) perpframe[1].

Put your answer here.

▪ And while you're at it, try to describe in words the relationship between the location of {x,y,z} and the location of

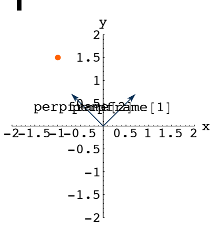
({x,y,z}.perpframe[1]) perpframe[1] + ({x,y,z}.perpframe[2]) perpframe[2] .

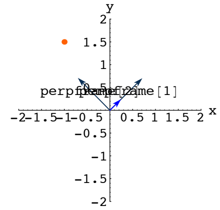
Put your answer here.

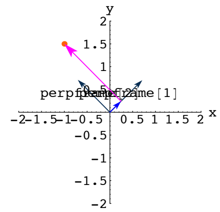
□ G.6.c.i) Perpendicular frame coordinates

See another one:

```
point = {-1.0, 1.5};
s = 0.78;
Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}};
setup = Show[Table[Arrow[perpframe[k], Tail → {0, 0},
      VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
    Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
    Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
    Graphics[{CadmiumOrange, PointSize[0.03], Point[point]}],
    Axes → True, AxesLabel → {"x", "y"}, PlotRange → {{-2, 2}, {-2, 2}}];
one = Show[setup,
   Arrow[(point.perpframe[1]) perpframe[1], Tail -> {0, 0},
         VectorColor -> Blue, HeadSize -> 0.2]];
both = Show[one, Arrow[(point.perpframe[2]) perpframe[2], Tail ->
      (point.perpframe[1]) perpframe[1], VectorColor -> Magenta]];
```







Grab the plots and animate.

The usual {x,y} coordinates of this point are:

```
point
```

```
{-1., 1.5}
```

Give the coordinates of the plotted point relative to the plotted perpendicular frame.

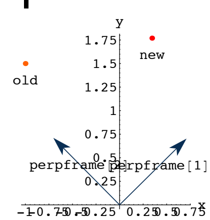
Tip: You can copy and paste them from the code above.

□ G.6.c.ii) Using the perpendicular frame coordinates to align a point

Here's that same plot shown with a the new point whose usual xy coordinates are

{(point.perpframe[1]), (point.perpframe[2])}:

```
point = {-1.0, 1.5};
s = 0.78;
Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}};
newpoint = {(point.perpframe[1]), (point.perpframe[2])};
setup = Show[Table[Arrow[perpframe[k], Tail → {0, 0},
      VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
    Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
    Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
    Graphics[{CadmiumOrange, PointSize[0.03], Point[point]}],
    Graphics[{Red, PointSize[0.03], Point[newpoint]}],
    Graphics[Text["old", point, {0, 2}]],
    Graphics[Text["new", newpoint, {0, 2}]],
    AspectRatio -> Automatic,
    Axes → True, AxesLabel → {"x", "y"}, PlotRange → All];
```



How is the new point whose usual xy coordinates are

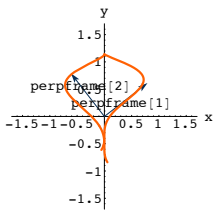
{(point.perpframe[1]), (point.perpframe[2])}:

related to the old point?

□ G.6.d.i) Repackaging a curve with perpendicular components

Here are a curve and a perpendicular frame:

```
s = 0.21 Pi;
Clear[perpframe];
{perpframe[1], perpframe[2]} =
      {{Cos[s], Sin[s]}, {Cos[s + Pi/2], Sin[s + Pi/2]}};

frameplot = {Table[
   Arrow[perpframe[k],
     Tail -> {0, 0}, VectorColor -> Indigo], {k, 1, 2}],
   Graphics[Text["perpframe[1]", 0.4 perpframe[1]]],
   Graphics[Text["perpframe[2]", 0.7 perpframe[2]]]};

Clear[x, y, t];
    {x[t_], y[t_]} = 0.6 {1.2 Sin[1.9 t]^3, 1.9 Cos[1.2 t] + 0.1 t};
{tlow, thigh} = {-0.6 π, 0.6 π};
ranger = 1.7;
curveplot = ParametricPlot[{x[t], y[t]}, {t, tlow, thigh},
   PlotStyle -> {{Thickness[0.012], CadmiumOrange}},
   DisplayFunction -> Identity];
original = Show[frameplot, curveplot, Axes -> True, AxesLabel ->
    {"x", "y"}, PlotRange -> {{-ranger, ranger}, {-ranger, ranger}}];
```

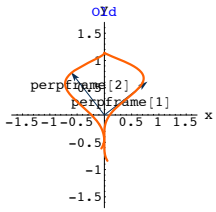


When you resolve {x[t],y[t]} into perpendicular components relative to this perpendicular frame

({x[t], y[t]}.perpframe[1]) perpframe[1] + ({x[t], y[t]}.perpframe[2]) perpframe[2]

you get the same curve.

```
alternatecurveplot = ParametricPlot[
  ({x[t], y[t]}.perpframe[1]) perpframe[1] +
   ({x[t], y[t]}.perpframe[2]) perpframe[2], {t, tlow, thigh},
  PlotStyle -> {{Thickness[0.012], CadmiumOrange}},
  DisplayFunction -> Identity];
old = Show[frameplot, alternatecurveplot,
  Axes -> True, AxesLabel -> {"x", "y"}, PlotRange ->
   {{-ranger, ranger}, {-ranger, ranger}}, PlotLabel -> "Old"];
```



Why did that happen?

□G.6.d.ii) Flipping a curve with perpendicular components

In part i), you saw that

{x[t], y[t]} and
({x[t], y[t]}.perpframe[1]) perpframe[1] + ({x[t], y[t]}.perpframe[2]) perpframe[2]

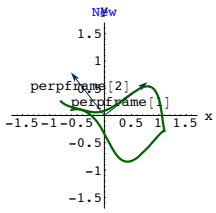
plot out the same way.

See what happens when you insert a minus sign like this

({x[t], y[t]}.perpframe[1]) perpframe[1] − ({x[t], y[t]}.perpframe[2]) perpframe[2]

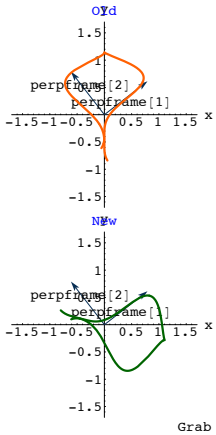
and plot:

```
newcurveplot =
 ParametricPlot[({x[t], y[t]}.perpframe[1]) perpframe[1] -
   ({x[t], y[t]}.perpframe[2]) perpframe[2],
  {t, tlow, thigh}, PlotStyle -> {{Thickness[0.012], GosiaGreen}},
  DisplayFunction -> Identity];
new = Show[frameplot, newcurveplot, Axes -> True,
  AxesLabel -> {"x", "y"}, PlotRange ->
   {{-ranger, ranger}, {-ranger, ranger}}, PlotLabel -> "New"];
```



See both curves on after the other

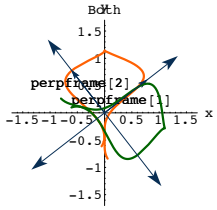
```
Show[old];
Show[new];
```



Grab and animate the last two plots slowly.

See them together:

```
scaledframeplot = {Table[
  Arrow[ranger perpframe[k],
    Tail -> {0, 0}, VectorColor -> Indigo], {k, 1, 2}],
     Table[
 Arrow[-ranger perpframe[k],
    Tail -> {0, 0}, VectorColor -> Indigo], {k, 1, 2}]};

Show[old, new, scaledframeplot, PlotLabel -> "Both"];
```



Your job is to try to describe the relationship between the two curves.

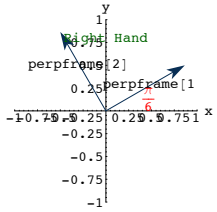
G.7) Perpendicular frames in 2D and 3D:

Left hand versus right hand. Rotations versus flips*

□G.7.a.i) Right hand versus left hand perpendicular frames in 2D

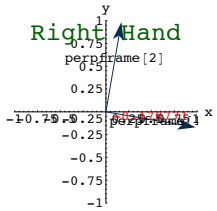
Perpendicular frames come in two flavors -right hand and left hand.
See a right hand 2D perpendicular frame:

```
s = π/6;
Clear[perpframe];
{perpframe[1], perpframe[2]} =
 {{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}};

Show[Table[Arrow[perpframe[k], Tail → {0, 0},
    VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
  Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
  Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
  Graphics[{Red, Text[s, 0.25 ({1, 0} + perpframe[1])]}],
  Graphics[{GosiaGreen, Text["Right Hand", {0, 0.8}]}], Axes → True,
  AxesLabel → {"x", "y"}, PlotRange → {{-1, 1}, {-1, 1}}];
```



See some random right hand 2D perpendicular frames.

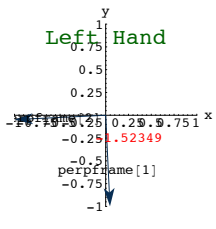
```
s = Random[Real, {-π/2, π/2}];
Clear[perpframe];
{perpframe[1], perpframe[2]} =
 {{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}};

Show[Table[Arrow[perpframe[k], Tail → {0, 0},
    VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
  Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
  Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
  Graphics[{Red, Text[s, 0.25 ({1, 0} + perpframe[1])]}],
  Graphics[{GosiaGreen, Text["Right Hand", {0, 0.85}]}],
  Axes → True, AxesLabel → {"x", "y"}, PlotRange → {{-1, 1}, {-1, 1}}];
```



Rerun many times.

Here's a random left hand perpendicular frame:

```
s = Random[Real, {-π/2, π/2}];
Clear[perpframe];
{perpframe[1], perpframe[2]} =
 {{Cos[s], Sin[s]}, -{Cos[s + π/2], Sin[s + π/2]}};

Show[Table[Arrow[perpframe[k], Tail → {0, 0},
    VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
  Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
  Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
  Graphics[{Red, Text[s, 0.25 ({1, 0} + perpframe[1])]}],
  Graphics[{GosiaGreen, Text["Left Hand", {0, 0.85}]}],
  Axes → True, AxesLabel → {"x", "y"}, PlotRange → {{-1, 1}, {-1, 1}}];
```

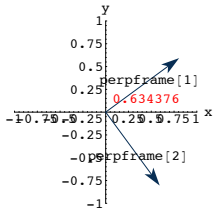


Rerun many times.

Here's how you can tell at a glance whether a given perpendicular frame {perpframe[1],perpframe[2]} is a left hand perpendicular frame or is a right hand perpendicular frame:

- If the smaller angle from perpframe[1] to perpframe[2] is counterclockwise, you have a right hand perpendicular frame
- If the smaller angle from perpframe[1] to perpframe[2] is clockwise, you have a left hand perpendicular frame

```
s = Random[Real, {-π/2, π/2}];
Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, (-1)^Random[Integer,{0,1}] {Cos[s + π/2], Sin[s + π/2]}};

Show[Table[Arrow[perpframe[k], Tail → {0, 0},
    VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
  Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
  Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
  Graphics[{Red, Text[s, 0.25 ({1, 0} + perpframe[1])]}], Axes → True,
  AxesLabel → {"x", "y"}, PlotRange → {{-1, 1}, {-1, 1}}];
```

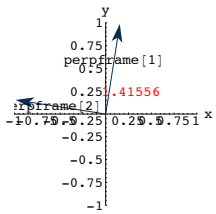


You make the call:
Which is it: Left hand or right hand?:

□G.7.a.ii) Getting the opposite

Here's another random perpendicular frame:

```
s = Random[Real, {-π/2, π/2}];
Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, (-1)^Random[Integer,{0,1}] {Cos[s + π/2], Sin[s + π/2]}};

Show[Table[Arrow[perpframe[k], Tail → {0, 0},
    VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
  Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
  Graphics[Text["perpframe[2]", 0.6 perpframe[2]]],
  Graphics[{Red, Text[s, 0.25 ({1, 0} + perpframe[1])]}], Axes → True,
  AxesLabel → {"x", "y"}, PlotRange → {{-1, 1}, {-1, 1}}];
```

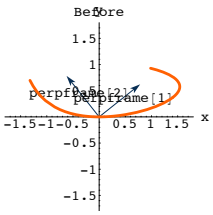


If you got a right hand frame in part i), then rerun until you get a left hand frame.
If you got a left hand frame in part i), then rerun until you get a right hand frame.

□G.7.b.i) Right hand frames: Rotation or flip or both?

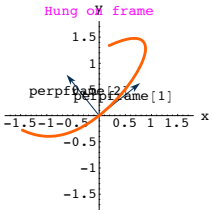
Here is a curve shown together with a 2D perpendicular frame:

```
Clear[x, y, t];
{x[t_], y[t_]} = {2 Cos[t] Sin[2 t], 3 Sin[t] t E^-t};
{tlow, thigh} = {-0.4, 1};
ranger = 1.8;
curveplot = ParametricPlot[{x[t], y[t]}, {t, tlow, thigh},
  PlotStyle → {{CadmiumOrange, Thickness[0.015]}},
  DisplayFunction → Identity];

s = Random[Real, {0.2 π, 0.4 π}];
Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}};

frameplot = {Table[Arrow[perpframe[k], Tail → {0, 0},
    VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
  Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
  Graphics[Text["perpframe[2]", 0.6 perpframe[2]]]};

before = Show[frameplot, curveplot,
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  AspectRatio → Automatic, Axes → True,
  AxesLabel → {"x", "y"}, PlotLabel → "Before"];
```



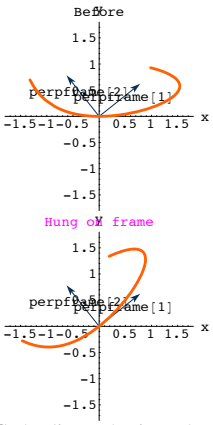
When you pick up this curve up and hang it on the plotted frame with
perpframe[1] playing the former role of the positive x-axis
and with
perpframe[2] playing the former role of the positive y-axis,
you get:

```
hungcurveplot = ParametricPlot[
  x[t] perpframe[1] + y[t] perpframe[2], {t, tlow, thigh},
PlotStyle -> {{CadmiumOrange, Thickness[0.015]}},
  DisplayFunction -> Identity];

after = Show[frameplot,
      hungcurveplot,
      PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
      AspectRatio -> Automatic,
      Axes -> True,
  AxesLabel -> {"x", "y"}, PlotLabel -> "Hung on frame"];
```



Here they are one after another:

```
Show[before];
Show[after];
```

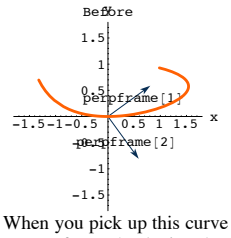


Grab, align and animate both plots slowly and then you make the calls:
- Is the plotted frame a right hand frame?
- Do you say that the act of hanging the curve on the plotted perpendicular frame is a rotation of the curve or is it a rotation and a flip?

□G.7.b.ii) Left hand frames: Rotation or flip or both?

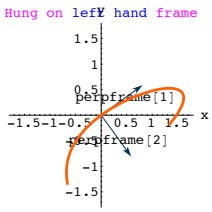
Here is the same curve shown together with a a left hand perpendicular frame:

```
s = 0.2 π;
Clear[perpframe];
{perpframe[1], perpframe[2]} =
  {{Cos[s], Sin[s]}, -{Cos[s + π/2], Sin[s + π/2]}};

frameplot = {Table[Arrow[perpframe[k], Tail → {0, 0},
    VectorColor → Indigo, HeadSize → 0.2], {k, 1, 2}],
  Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
  Graphics[Text["perpframe[2]", 0.6 perpframe[2]]]};

before = Show[frameplot, curveplot,
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  AspectRatio → Automatic, Axes → True,
  AxesLabel → {"x", "y"}, PlotLabel → "Before"];
```



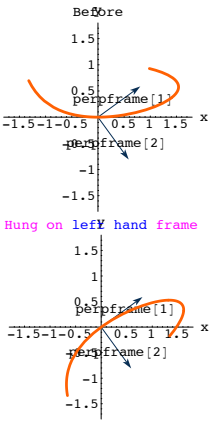
When you pick up this curve up and hang it on the plotted frame with
perpframe[1] playing the former role of the positive x-axis
and with
perpframe[2] playing the former role of the positive y-axis,
you get:

```
hungcurveplot = ParametricPlot[
  x[t] perpframe[1] + y[t] perpframe [2], {t, tlow, thigh},
PlotStyle -> {{CadmiumOrange, Thickness[0.015]}},
  DisplayFunction -> Identity];

after = Show[frameplot,
      hungcurveplot,
      PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
      AspectRatio -> Automatic,
      Axes -> True, AxesLabel -> {"x", "y"},
  PlotLabel -> "Hung on left hand frame"];
```



Here they are one after another:

```
Show[before];
Show[after];
```



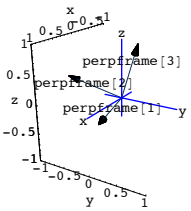
Grab, align and animate both plots and then you make the call:
▪Do you say that the act of hanging the curve on the plotted perpendicular frame is a rotation of the curve or is it a rotation and a flip?

G.8) Plotting on a plane

□G.8.a) Plotting on a plane

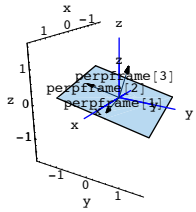
Here is a plot of a 3D perpendicular frame:

```
Clear[perpframe];
r = 0;
s = -0.3;
t = 0.2;
Clear[perpframe];
{perpframe[1], perpframe[2], perpframe[3]} =
  {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
     Cos[t] Sin[r] + Cos[r] Cos[s] Sin[t], Sin[s] Sin[t]},
    (-1)^Random[Integer,{0,1}] {-Cos[s] Cos[t] Sin[r] - Cos[r] Sin[t],
      Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[t] Sin[s]},
    {Sin[r] Sin[s], -Cos[r] Sin[s], Cos[s]}};

ranger = 1.0;
frameplot = Show[
   Table[Arrow[perpframe[k], Tail → {0, 0, 0}, VectorColor → Indigo],
    {k, 1, 3}], Graphics3D[Text["perpframe[1]", 0.4 perpframe[1]]],
   Graphics3D[Text["perpframe[2]", 0.7 perpframe[2]]],
   Graphics3D[Text["perpframe[3]", 0.7 perpframe[3]]],
   Axes3D[1, 0.1], PlotRange → {{-ranger, ranger},
     {-ranger, ranger}, {-ranger, ranger}}, Boxed → False,
   Axes → True, ViewPoint → CMView, AxesLabel → {"x", "y", "z"}];
```



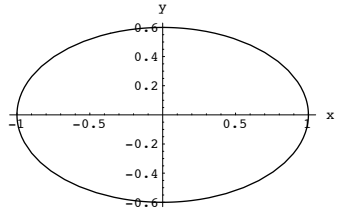
Here is a piece of the plane through {0,0,0} that is framed by perpframe[1] and perpframe[2]:

```
Clear[s, t];
{slow, shigh} = {-1.3, 1.3};
{tlow, thigh} = {-1.3, 1.3};
ranger = 1.7;

planeplot = ParametricPlot3D[ s perpframe[1] + t perpframe[2],
       {s, slow, shigh}, {t, tlow, thigh}, PlotPoints -> {2, 2},
     DisplayFunction -> Identity];
setup = Show[planeplot, frameplot, Axes3D[2, 0.1],
  PlotRange ->
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Boxed -> False,
  Axes -> True, ViewPoint -> CMView,
   AxesLabel -> {"x", "y", "z"}, DisplayFunction -> $DisplayFunction];
```



Here is an ellipse plotted directly in 2D:

```
Clear[x, y, t];
{x[t_], y[t_]} = {1.0 Cos[t], 0.6 Sin[t]};
{tlow, thigh} = {0, 2 Pi};
ParametricPlot[{x[t], y[t]},
  {t, tlow, thigh}, AxesLabel -> {"x", "y"}];
```



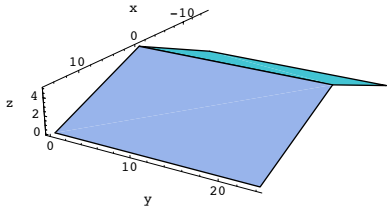
You job is to plot an exact copy of this ellipse on the plane plotted above.
Measure the area on the plane enclosed by your ellipse.

G.9) Stovepipes and roofs

□G.9.a) Stovepipes and roofs

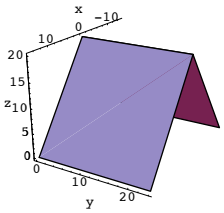
Here is a roof top:

```
Clear[roof, x, y, z];
    roof[x_, y_, z_] := {Graphics3D[
    Polygon[{{x, 0, 0}, {x, y, 0}, {0, y, z}, {0, 0, z}, {x, 0, 0}}]],
       Graphics3D[Polygon[{{-x, 0, 0}, {-x, y, 0},
     {0, y, z}, {0, 0, z}, {-x, 0, 0}}]]};
Show[roof[15, 24, 5], ViewPoint -> CMView, Axes -> True,
  Boxed -> False, AxesLabel -> {"x", "y", "z"}];
```

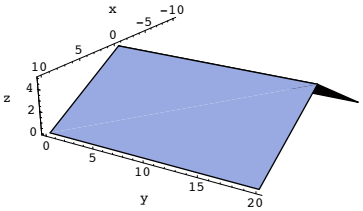


The meaning of the variables x,y and z are given in the plot.
See some more:

```
Show[roof[15, 24, 20], ViewPoint -> CMView,
  Axes -> True, Boxed -> False, AxesLabel -> {"x", "y", "z"}];
```

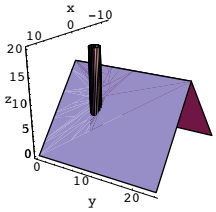


```
Show[roof[10, 20, 5], ViewPoint -> CMView,
  Axes -> True, Boxed -> False, AxesLabel -> {"x", "y", "z"}];
```



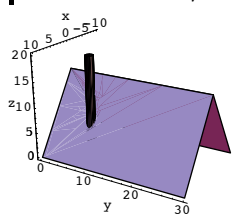
Here comes a stove pipe through one of these roofs:

```
Clear[stovepipe, x, y, z, radius];

stovepipe[x_, y_, z_, radius_] :=
  ParametricPlot3D[s ({x, y, 0} + radius {Cos[t], Sin[t], 0}) +
    (1 - s) ({x, y, z} + radius {Cos[t], Sin[t], 0}) ,
      {s, 0, 1}, {t, 0, 2 Pi}, PlotPoints -> {2, Automatic},
   DisplayFunction -> Identity];
x = 12;
y = 24;
z = 15;
xstovepipecenter = 5;
ystovepipecenter = 8;
stovepipeheight = 20;
stovepiperadius = 1;

roofplot = Show[roof[x, y, z], stovepipe[xstovepipecenter,
    ystovepipecenter, stovepipeheight , stovepiperadius],
   PlotRange -> All, ViewPoint -> CMView, Axes -> True,
   Boxed -> False, AxesLabel -> {"x", "y", "z"}];
```



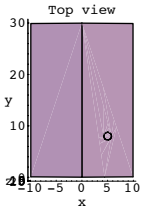
Another:

```
x = 10;
y = 30;
z = 15;
xstovepipecenter = 5;
ystovepipecenter = 8;
stovepipeheight = 20;
stovepiperadius = 0.75;

roofplot = Show[roof[x, y, z], stovepipe[xstovepipecenter,
    ystovepipecenter, stovepipeheight , stovepiperadius],
   PlotRange -> All, ViewPoint -> CMView, Axes -> True,
   Boxed -> False, AxesLabel -> {"x", "y", "z"}];
```

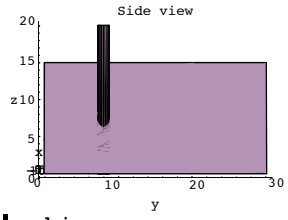


Se the top view, the side view and the end view:

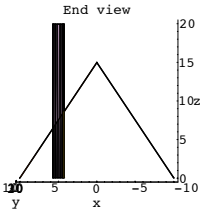
```
topview =
  Show[roofplot, ViewPoint -> 100 {0, 0, 1}, PlotLabel -> "Top view"];
```




```
sideview =
  Show[roofplot, ViewPoint -> 100 {1, 0, 0}, PlotLabel -> "Side view"];
```

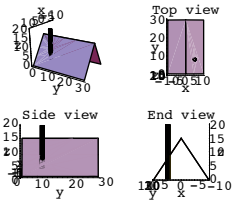



```
endview =
  Show[roofplot, ViewPoint -> 100 {0, 1, 0}, PlotLabel -> "End view"];
```



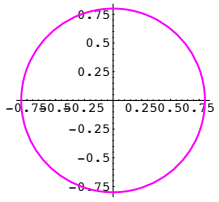
See everything:

```
Show[GraphicsArray[{{roofplot, topview}, {sideview, endview}}]];
```



Your job is to cut a hole in the metal roof to let the stovepipe come through.
That lab pest the dorky Calculus Cal says to cut a circle whose radius is the same as the radius of the stovepipe.
For instance in the plot above, the radius is 0.8; so Cal's suggestion is to cut this circle out of the metal roof sheeting:

```
Calholeplot = ParametricPlot[{0.8 Cos[t], 0.8 Sin[t]},
   {t, 0, 2 Pi}, PlotStyle -> {{Thickness[0.01], Magenta}}];
```



What do you say to Cal?
What curve do you cut out of the metal roof sheeting to let this pipe pass through snugly?