Matrices, Geometry&*Mathematica*

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

Producer: Bruce Carpenter

Publisher: Math Everywhere, Inc.

MGM.02 2D Matrix Action

GIVE IT A TRY!

G.1) Hitting with 2D matrices and visually assessing the result*

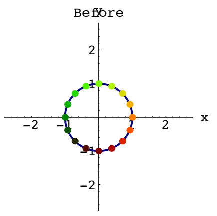
```
To get the later parts to run, you have to run part i).
```

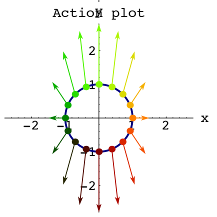
□G.1.a.i) Action Movie for $\begin{pmatrix}1.5 & 0\\ 0 & 2.8\end{pmatrix}$

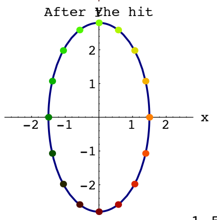
Here's an action movie depicting what you get when you hit the unit circle with the 2D matrix

$A = \begin{pmatrix}1.5 & 0\\ 0 & 2.8\end{pmatrix}$:

```
A = ( 1.5   0  );
    (  0   2.8 )
Clear[x, y, t, hitplotter,
  hitpointplotter, pointcolor, actionarrows, matrix2D];
{tlow, thigh} = {0, 2 π};

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

pointcolor[t_] = RGBColor[0.5 (Cos[t] + 1), 0.5 (Sin[t] + 1), 0];

        thigh - tlow
jump = ─────────────;
             16

hitplotter[matrix2D_] := ParametricPlot[matrix2D.{x[t], y[t]},
    {t, tlow, thigh}, PlotStyle → {{Thickness[0.01], NavyBlue}},
    PlotRange → {{-Max[{1.2, Max[SingularValues[A][[2]]]}],
       Max[{1.2, Max[SingularValues[A][[2]]]}]},
      {-Max[{1.2, Max[SingularValues[A][[2]]]}],
       Max[{1.2, Max[SingularValues[A][[2]]]}]}},
    AxesLabel → {"x", "y"}, DisplayFunction → Identity];

hitpointplotter[matrix2D_] :=
  Table[Graphics[{pointcolor[t], PointSize[0.035],
     Point[matrix2D.{x[t], y[t]}]}], {t, tlow, thigh - jump, jump}];

actionarrows[matrix2D_] :=

  Table[Arrow[matrix2D.{x[t], y[t]} - {x[t], y[t]}, Tail → {x[t], y[t]}
    VectorColor → pointcolor[t], HeadSize → 0.25],
   {t, tlow, thigh - jump, jump}];

before = Show[hitplotter[IdentityMatrix[2]],
   hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
   DisplayFunction → $DisplayFunction];

Show[before, actionarrows[A],
  PlotLabel → "Action plot", DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This matrix is" MatrixForm[A]
```







This matrix is $\begin{pmatrix}1.5 & 0\\ 0 & 2.8\end{pmatrix}$

```
Grab all three plots, align and animate slowly.
To get maximum viewing pleasure and maximum information,
look at one point and follow it.
Then pick another point and follow it.
Keep doing this until you have a good visual grasp of the action.
```

Choose from the words:

stretch, shrink, rotate, flip, shear, squash

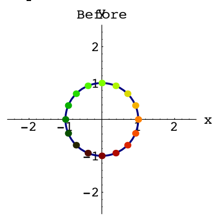
and then throw in some of your own colorful prose to describe the action of a hit with this matrix.

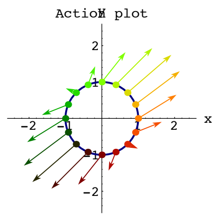
□G.1.a.ii) Action Movie for $\begin{pmatrix}2.05 & 0.65\\ 0.65 & 1.82\end{pmatrix}$

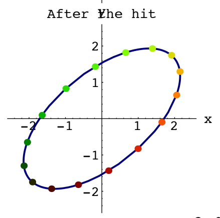
Here's an action movie depicting what you get when you hit the unit circle with the 2D matrix

$A = \begin{pmatrix}2.05 & 0.65\\ 0.65 & 1.82\end{pmatrix}$:

```
A = ( 2.05  0.65 );
    ( 0.65  1.82 )

before = Show[hitplotter[IdentityMatrix[2]],
   hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
   DisplayFunction → $DisplayFunction];

Show[before, actionarrows[A],
  PlotLabel → "Action plot", DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This matrix is" MatrixForm[A]
```







This matrix is $\begin{pmatrix}2.05 & 0.65\\ 0.65 & 1.82\end{pmatrix}$

```
Grab all three plots, align and animate slowly.
To get maximum viewing pleasure and maximum information,
look at one point and follow it.
Then pick another point and follow it.
Keep doing this until you have a good visual grasp of the action.
```

Choose from the words:

stretch, shrink, rotate, flip, shear, squash

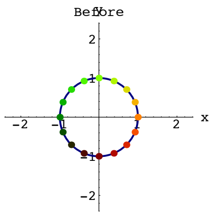
and then throw in some of your own colorful prose to describe the action of a hit with this matrix.

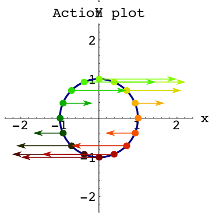
□G.1.a.iii) Action Movie for $\begin{pmatrix}1.0 & 2.0\\ 0.0 & 1.0\end{pmatrix}$

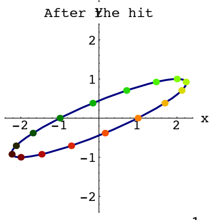
Here's an action movie depicting what you get when you hit the unit circle with the 2D matrix

$A = \begin{pmatrix}1.0 & 2.0\\ 0.0 & 1.0\end{pmatrix}$:

```
A = ( 1.0  2.0 );
    ( 0.0  1.0 )


before = Show[hitplotter[IdentityMatrix[2]],
   hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
   DisplayFunction → $DisplayFunction];

Show[before, actionarrows[A],
  PlotLabel → "Action plot", DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This matrix is" MatrixForm[A]
```







This matrix is $\begin{pmatrix} 1. & 2. \\ 0 & 1. \end{pmatrix}$

```
          Grab all three plots, align and animate slowly.
   To get maximum viewing pleasure and maximum information,
                 look at one point and follow it.
                Then pick another point and follow it.
     Keep doing this until you have a good visual grasp of the action.
```

Choose from the words:

stretch, shrink, rotate, flip, shear, squash

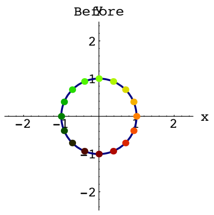
and then throw in some of your own colorful prose to describe the action of a hit with this matrix.

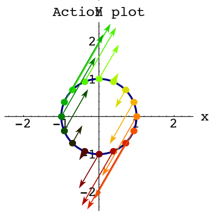
□G.1.a.iv) Action Movie for $\begin{pmatrix} 0.09 & 0.53 \\ -1.58 & 1.91 \end{pmatrix}$

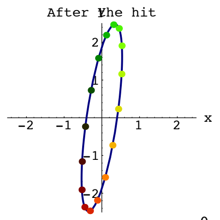
Here's an action movie depicting what you get when you hit the unit circle with the 2D matrix

$A = \begin{pmatrix} 0.09 & 0.53 \\ -1.58 & 1.91 \end{pmatrix}$

```
A = ( 0.09   0.53 );
    (-1.58   1.91 )

before = Show[hitplotter[IdentityMatrix[2]],
    hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
    DisplayFunction → $DisplayFunction];

Show[before, actionarrows[A],
  PlotLabel → "Action plot", DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This matrix is" MatrixForm[A]
```







This matrix is $\begin{pmatrix} 0.09 & 0.53 \\ -1.58 & 1.91 \end{pmatrix}$

```
          Grab all three plots, align and animate slowly.
   To get maximum viewing pleasure and maximum information,
                 look at one point and follow it.
                Then pick another point and follow it.
     Keep doing this until you have a good visual grasp of the action.
```

Choose from the words:

stretch, shrink, rotate, flip, shear, squash

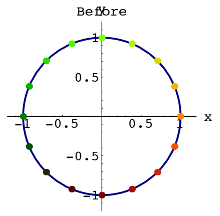
and then throw in some of your own colorful prose to describe the action of a hit with this matrix.

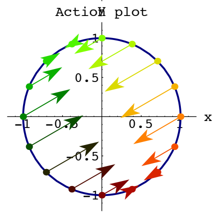
□G.1.a.v) Action Movie for $\begin{pmatrix} 0.40 & -0.35 \\ -0.35 & 0.80 \end{pmatrix}$

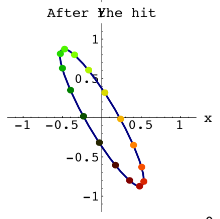
Here's an action movie depicting what you get when you hit the unit circle with the 2D matrix

$A = \begin{pmatrix} 0.40 & -0.35 \\ -0.35 & 0.80 \end{pmatrix}$:

```
A = ( 0.40   -0.35 );
    (-0.35    0.80 )
ranger = 1.2;

before = Show[hitplotter[IdentityMatrix[2]],
    hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
    DisplayFunction → $DisplayFunction];
Show[before, actionarrows[A], PlotLabel → "Action plot",
  DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This matrix is" MatrixForm[A]
```







This matrix is $\begin{pmatrix} 0.4 & -0.35 \\ -0.35 & 0.8 \end{pmatrix}$

```
          Grab all three plots, align and animate slowly.
   To get maximum viewing pleasure and maximum information,
                 look at one point and follow it.
                Then pick another point and follow it.
     Keep doing this until you have a good visual grasp of the action.
```

Choose from the words:

stretch, shrink, rotate, flip, shear, squash

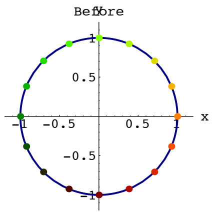
and then throw in some of your own colorful prose to describe the action of a hit with this matrix.

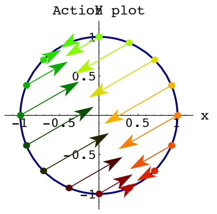
□G.1.a.vi) Action Movie for $\begin{pmatrix} 0.25 & -0.43 \\ -0.43 & 0.75 \end{pmatrix}$

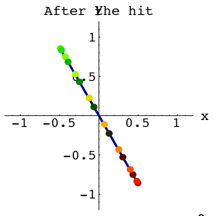
Here's an action movie depicting what you get when you hit the unit circle with the 2D matrix

$$A = \begin{pmatrix} 0.25 & -0.43 \\ -0.43 & 0.75 \end{pmatrix}:$$

```
A = ( 0.25  -0.43 );
    ( -0.43  0.75 )
ranger = 1.2;


before = Show[hitplotter[IdentityMatrix[2]],
   hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
   DisplayFunction → $DisplayFunction];
Show[before, actionarrows[A], PlotLabel → "Action plot",
  DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This matrix is" MatrixForm[A]
```







```
This matrix is ( 0.25  -0.43 )
               ( -0.43  0.75 )
```

```
                    Grab all three plots, align and animate slowly.
              To get maximum viewing pleasure and maximum information,
                           look at one point and follow it.
                        Then pick another point and follow it.
           Keep doing this until you have a good visual grasp of the action.
```

Choose from the words:

stretch, shrink, rotate, flip, shear, squash

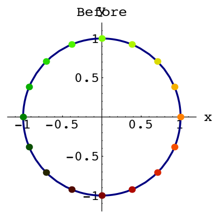
and then throw in some of your own colorful prose to describe the action of a hit with this matrix.

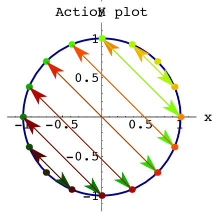
□G.1.a.vii) Action Movie for $\begin{pmatrix} 0.0 & 1.0 \\ 1.0 & 0.0 \end{pmatrix}$

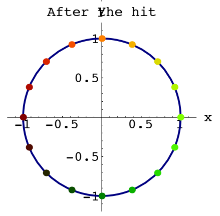
Here's an action movie depicting what you get when you hit the unit circle with the 2D matrix

$$A = \begin{pmatrix} 0.0 & 1.0 \\ 1.0 & 0.0 \end{pmatrix}:$$

```
A = ( 0.0  1.0 );
    ( 1.0  0.0 )
ranger = 1.2;
before = Show[hitplotter[IdentityMatrix[2]],
   hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
   DisplayFunction → $DisplayFunction];
Show[before, actionarrows[A], PlotLabel → "Action plot",
  DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This matrix is" MatrixForm[A]
```







```
This matrix is ( 0   1. )
               ( 1.  0  )
```

```
                    Grab all three plots, align and animate slowly.
              To get maximum viewing pleasure and maximum information,
                           look at one point and follow it.
                        Then pick another point and follow it.
           Keep doing this until you have a good visual grasp of the action.
```

Choose from the words:

stretch, shrink, rotate, flip, shear, squash

and then throw in some of your own colorful prose to describe the action of a hit with this matrix.

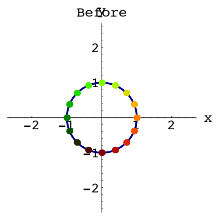
□G.1.a.viii) Action movie for $\begin{pmatrix} 1.575 & 1.949 \\ 1.949 & -0.675 \end{pmatrix}$

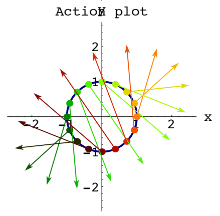
Here's an action movie depicting what you get when you hit the unit circle with the 2D matrix

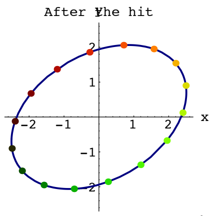
$$A = \begin{pmatrix} 1.575 & 1.949 \\ 1.949 & -0.675 \end{pmatrix}:$$

```
A = ( 1.575  1.949 );
    ( 1.949  -0.675 )
MatrixForm[A]

ranger = 3.0
before = Show[hitplotter[IdentityMatrix[2]],
   hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
   DisplayFunction → $DisplayFunction];
Show[before, actionarrows[A], PlotLabel → "Action plot",
  DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This matrix is" MatrixForm[A]
```

```
( 1.575  1.949 )
( 1.949  -0.675 )
 3.
```







```
This matrix is ( 1.575   1.949 )
               ( 1.949  -0.675 )
```

```
                 Grab all three plots, align and animate slowly.
          To get maximum viewing pleasure and maximum information,
                       look at one point and follow it.
                    Then pick another point and follow it.
      Keep doing this until you have a good visual grasp of the action.
```

Choose from the words:

stretch, shrink, rotate, flip, shear, squash

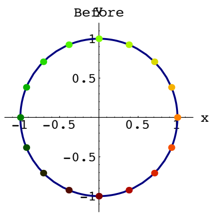
and then throw in some of your own colorful prose to describe the action of a hit with this matrix.

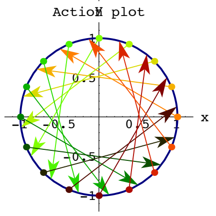
□G.1.a.ix) Action movie for $\begin{pmatrix} -0.500 & -0.866 \\ 0.866 & -0.500 \end{pmatrix}$

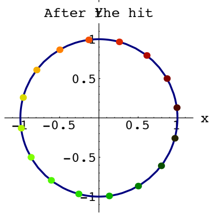
Here's an action movie depicting what you get when you hit the unit circle with the 2D matrix

$A = \begin{pmatrix} -0.500 & -0.866 \\ 0.866 & -0.500 \end{pmatrix}$:

```
A = ( -0.500  -0.866 );
    (  0.866  -0.500 )
ranger = 1.5;

before = Show[hitplotter[IdentityMatrix[2]],
   hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
   DisplayFunction → $DisplayFunction];
Show[before, actionarrows[A], PlotLabel → "Action plot",
  DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This matrix is" MatrixForm[A]
```







```
This matrix is ( -0.5   -0.866 )
               ( 0.866  -0.5   )
```

```
                 Grab all three plots, align and animate slowly.
          To get maximum viewing pleasure and maximum information,
                       look at one point and follow it.
                    Then pick another point and follow it.
      Keep doing this until you have a good visual grasp of the action.
```

Choose from the words:

stretch, shrink, rotate, flip, shear, squash

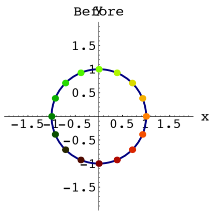
and then throw in some of your own colorful prose to describe the action of a hit with this matrix.

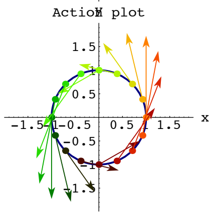
□G.1.a.x) Action movie for $\begin{pmatrix} 1.00 & -0.87 \\ 1.73 & 0.50 \end{pmatrix}$

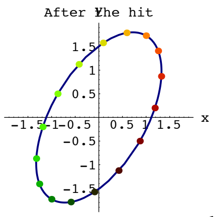
Here's an action movie depicting what you get when you hit the unit circle with the 2D matrix

$A = \begin{pmatrix} 1.00 & -0.87 \\ 1.73 & 0.50 \end{pmatrix}$:

```
A = ( 1.00  -0.87 );
    ( 1.73   0.50 )
ranger = 2.5;


before = Show[hitplotter[IdentityMatrix[2]],
   hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
   DisplayFunction → $DisplayFunction];
Show[before, actionarrows[A], PlotLabel → "Action plot",
  DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This matrix is" MatrixForm[A]
```







```
This matrix is ( 1.    -0.87 )
               ( 1.73   0.5  )
```

```
                 Grab all three plots, align and animate slowly.
          To get maximum viewing pleasure and maximum information,
                       look at one point and follow it.
                    Then pick another point and follow it.
      Keep doing this until you have a good visual grasp of the action.
```

Choose from the words:

stretch, shrink, rotate, flip, shear, squash

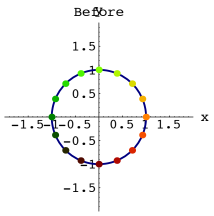
and then throw in some of your own colorful prose to describe the action of a hit with this matrix.

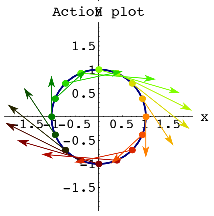
□G.1.a.xi) Action movie for $\begin{pmatrix} 1.00 & 1.73 \\ -0.87 & 0.50 \end{pmatrix}$

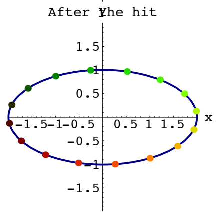
Here's an action movie depicting what you get when you hit the unit circle with the 2D matrix

$A = \begin{pmatrix} 1.00 & 1.73 \\ -0.87 & 0.50 \end{pmatrix}$:

```
A = (  1.00  1.73 );
    ( -0.87  0.50 )
ranger = 2.5;

before = Show[hitplotter[IdentityMatrix[2]],
   hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
   DisplayFunction → $DisplayFunction];
Show[before, actionarrows[A], PlotLabel → "Action plot",
  DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This matrix is" MatrixForm[A]
```







```
This matrix is ( 1.     1.73 )
               ( -0.87  0.5  )
```

```
                 Grab all three plots, align and animate slowly.
          To get maximum viewing pleasure and maximum information,
                        look at one point and follow it.
                     Then pick another point and follow it.
        Keep doing this until you have a good visual grasp of the action.
```

Choose from the words:

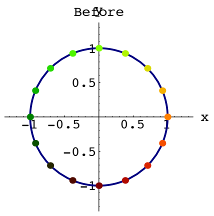
stretch, shrink, rotate, flip, shear, squash

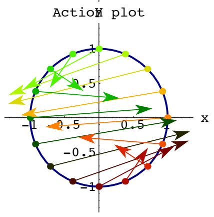
and then throw in some of your own colorful prose to describe the action of a hit with this matrix.

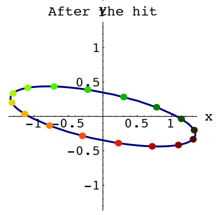
□G.1.a.xii) Action movie for a random matrix

Here's an action movie depicting what you get when you hit the unit circle with a random 2D matrix:

```
A = ( Random[Real, {-2, 2}]  Random[Real, {-2, 2}] );
    ( Random[Real, {-2, 2}]  Random[Real, {-2, 2}] )
ranger = 2.5;

before = Show[hitplotter[IdentityMatrix[2]],
    hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
    DisplayFunction → $DisplayFunction];
Show[before, actionarrows[A], PlotLabel → "Action plot",
  DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This random matrix is" MatrixForm[A]
```







```
This random matrix is ( -0.77885   -1.09544 )
                      ( -0.133716  0.416051 )
```

Rerun **many** times observing the actions each time.

No specific response is asked for, but if you would like to write a few lines on what you have noticed, please do.

□G.1.a.xiii) Action movie for a matrix of your own choice

Replace the numbers a, b, c and d with **decimal** numbers of your own choice and run the action movie:

```
A = ( a  b );
    ( c  d )
ranger = 2.5;

before = Show[hitplotter[IdentityMatrix[2]],
    hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
    DisplayFunction → $DisplayFunction];
Show[before, actionarrows[A], PlotLabel → "Action plot",
  DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
"This comes from" MatrixForm[A]
```

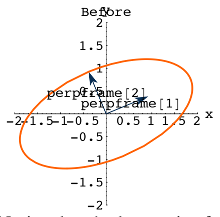
Play with the numbers and rerun until you get a matrix that you really like.

G.2) Aligners, hangers and stretchers*

□G.2.a) Aligners

Here's a tilted ellipse shown with a new perpendicular frame:

```
s = π/8;
Clear[perpframe];
{perpframe[1], perpframe[2]} =
  N[{{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}}];

frameplot = {Table[Arrow[perpframe[k], Tail → {0, 0},
      VectorColor → Indigo, HeadSize → 0.3], {k, 1, 2}],
    Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
    Graphics[Text["perpframe[2]", 0.5 perpframe[2]]]};

Clear[x, y, t];
{tlow, thigh} = {0, 2 π};
ranger = 2;

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

ellipseplot = ParametricPlot[{x[t], y[t]}, {t, tlow, thigh},
  PlotStyle → {{Thickness[0.01], CadmiumOrange}},
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  AxesLabel → {"x", "y"}, DisplayFunction → Identity];

Show[ellipseplot, frameplot,
  PlotLabel → "Before", DisplayFunction → $DisplayFunction];
```



Notice that the long axis of the plotted ellipse is aligned on perpframe[1] and the short axis of the ellipse is aligned on perpframe[2].

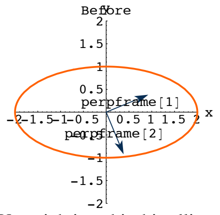
Your job is to hit this ellipse with an aligner matrix so that the long axis of the ellipse is aligned the x-axis and the short axis of the ellipse is aligned with the y-axis.

Show off your matrix with a decisive plot.

□G.2.b) Hangers

Here's an ellipse shown with a perpendicular frame:

```
s = π/8;
{perpframe[1], perpframe[2]} =
  N[{{Cos[s], Sin[s]}, -{Cos[s + π/2], Sin[s + π/2]}}];

frameplot = {Table[Arrow[perpframe[k], Tail → {0, 0},
      VectorColor → Indigo, HeadSize → 0.3], {k, 1, 2}],
    Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
    Graphics[Text["perpframe[2]", 0.5 perpframe[2]]]};

Clear[x, y, t, hitplotter, matrix2D];
{tlow, thigh} = {0, 2 π};
ranger = 2;
{x[t_], y[t_]} = {2 Cos[t], Sin[t]};

pointcolor[t_] = RGBColor[0.5 (Cos[t] + 1), 0.5 (Sin[t] + 1), 0];

jump = (thigh - tlow)/8;

ellipseplot  = ParametricPlot[{x[t], y[t]}, {t, tlow, thigh},
  PlotStyle → {{Thickness[0.01], CadmiumOrange}},
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  AxesLabel → {"x", "y"}, DisplayFunction → Identity];


Show[ellipseplot, frameplot,
  PlotLabel → "Before", DisplayFunction → $DisplayFunction];
```

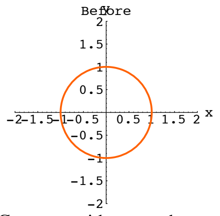


Your job is to hit this ellipse with a hanger matrix so that the long axis of the ellipse is aligned with perpframe[1] and the short axis of the ellipse is aligned with perpframe[2]. Show off your matrix with a decisive plot.

□G.2.c) Stretchers

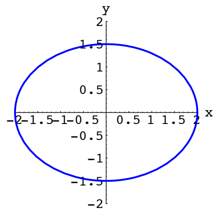
Here's the unit circle:

```
Clear[x, y, t];
{tlow, thigh} = {0, 2 π};
ranger = 2;
{x[t_], y[t_]} = {Cos[t], Sin[t]};


circleplot = ParametricPlot[{x[t], y[t]}, {t, tlow, thigh},
  PlotStyle → {{Thickness[0.01], CadmiumOrange}},
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  AxesLabel → {"x", "y"}, DisplayFunction → Identity];


Show[circleplot, PlotLabel → "Before",
  DisplayFunction → $DisplayFunction];
```



Come up with a stretcher matrix A so that when you hit the unit circle with A you get this ellipse:

```
ParametricPlot[{2 Cos[t], 1.5 Sin[t]}, {t, tlow, thigh},
    PlotStyle -> {{Thickness[0.01], Blue}},
    PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
    AxesLabel -> {"x", "y"}];
```

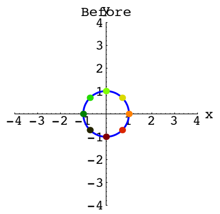


G.3) Area measurements and estimates*

□G.3.a.i) Hitting the unit circle with a stretcher

Here's the unit circle waiting to be hit with a matrix:

```
Clear[x, y, t, hitplotter, hitpointplotter, pointcolor, matrix2D];
{tlow, thigh} = {0, 2 π};
ranger = 4;
{x[t_], y[t_]} = {Cos[t], Sin[t]};

pointcolor[t_] = RGBColor[0.5 (Cos[t] + 1), 0.5 (Sin[t] + 1), 0];

       thigh - tlow
jump = ────────────;
            8

hitplotter[matrix2D_] := ParametricPlot[matrix2D.{x[t], y[t]},
   {t, tlow, thigh}, PlotStyle → {{Thickness[0.01], Blue}},
   PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
   AxesLabel → {"x", "y"}, DisplayFunction → Identity];

hitpointplotter[matrix2D_] :=
  Table[Graphics[{pointcolor[t], PointSize[0.035],
     Point[matrix2D.{x[t], y[t]}]}], {t, tlow, thigh - jump, jump}];

Show[hitplotter[IdentityMatrix[2]],
  hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
  DisplayFunction → $DisplayFunction];
```

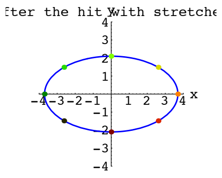


Here's what you get when you hit the unit circle with the xystretcher

$$\text{stretcher} = \begin{pmatrix}3.7 & 0\\ 0 & 2.1\end{pmatrix}:$$

```
stretcher = DiagonalMatrix[{3.7, 2.1}];
MatrixForm[stretcher]

Show[hitplotter[stretcher], hitpointplotter[stretcher],
  PlotLabel → "After the hit with stretcher",
  DisplayFunction → $DisplayFunction];
```

```
( 3.7   0  )
(  0   2.1 )
```



Given that the area of the unit circle measures out to π square units, what does the area of the ellipse plotted above measure out to?

□G.3.a.ii) Hitting the unit circle with a stretcher and then with a hanger

Here's a hanger matrix:

```
s = 0.1 π;
{perpframe[1], perpframe[2]} =
  N[{{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}}];
hanger = Transpose[{perpframe[1], perpframe[2]}];

MatrixForm[hanger]
```

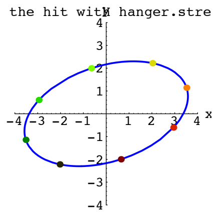
```
( 0.951057  -0.309017 )
( 0.309017   0.951057 )
```

Here's what you get when you hit the unit circle with the same xystretcher

$$\text{stretcher} = \begin{pmatrix}3.7 & 0\\ 0 & 2.1\end{pmatrix}$$

as in part i) and then hit it with hanger matrix defined immediately above::

```
Show[hitplotter[hanger.stretcher],
  hitpointplotter[hanger.stretcher],
  PlotLabel → "After the hit with hanger.stretcher",
  DisplayFunction → $DisplayFunction];
```

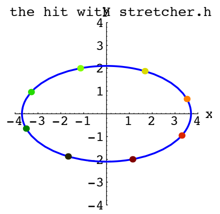


Given that the area of the unit circle measures out to π square units, what does the area of the ellipse plotted above measure out to?

□G.3.a.iii) Hitting the unit circle with a hanger and then with a stretcher

Here's what you get when you hit the unit circle with the hanger in part ii) above and then hit the result with the same xystretcher as above:

```
Show[hitplotter[stretcher.hanger],
  hitpointplotter[stretcher.hanger],
  PlotLabel → "After the hit with stretcher.hanger",
  DisplayFunction → $DisplayFunction];
```

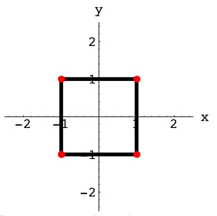


This ellipse is not tilted. Does this surprise you? Why or why not?
Given that the area of the unit circle measures out to π square units, what does the area of the ellipse plotted above measure out to?

□ G.3.b.i) Hitting a square centered on {0,0} with a stretcher and then with a hanger

Here's a square centered on {0,0}:

```
corners = {{-1, -1}, {1, -1}, {1, 1}, {-1, 1}, {-1, -1}};
ranger = 2.5;

Clear[hitplotter, matrix];
hitplotter[matrix_] :=
  {Graphics[{Black, Thickness[0.02], Line[Table[
       matrix.corners[[k]], {k, 1, Length[corners]}]]}], Table[
    Graphics[{PointSize[0.035], Red, Point[matrix.corners[[k]]]}],
    {k, 1, Length[corners]}]};

Show[hitplotter[IdentityMatrix[2]],
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"}];
```



Set up a random hanger matrix as follows:

```
s = Random[Real, {0.1 π, 0.45 π}];
{perpframe[1], perpframe[2]} =
  N[{{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}}];
hanger = Transpose[{perpframe[1], perpframe[2]}];

MatrixForm[hanger]
```

$$\begin{pmatrix} 0.273386 & -0.961904 \\ 0.961904 & 0.273386 \end{pmatrix}$$

Set up a stretcher matrix as follows:

```
xstretch = 1.5;
ystretch = 0.8;
stretcher = DiagonalMatrix[{xstretch, ystretch}];

MatrixForm[stretcher]
```

$$\begin{pmatrix} 1.5 & 0 \\ 0 & 0.8 \end{pmatrix}$$

Put A = stretcher.hanger:

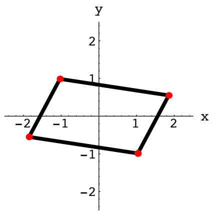
```
 A = stretcher.hanger;
MatrixForm[A]
```

$$\begin{pmatrix} 0.410079 & -1.44286 \\ 0.769524 & 0.218709 \end{pmatrix}$$

A is a rotation followed by a stretch.

Hit the square with A:

```
Show[hitplotter[A],
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"}];
```



Why did you get a parallelogram instead of an ordinary rectangle?
Use the information in the code defining A to help to measure the area inside this parallelogram.

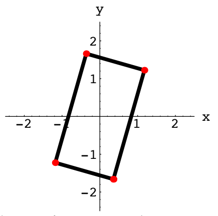
For a friendly tip, click on the right.

If you are doing a lot of unpleasant calculation, then you are not doing this problem the easiest way.

□ G.3.b.ii) Hitting a square centered on {0,0} with a hanger and then with a stretcher

In the last problem, you hit the square with
A= stretcher.hanger
and got a parallelogram.
This time reverse the order of the product and hit the square with
B = hanger.stretcher
and see what you get:

```
B = hanger.stretcher;
Show[hitplotter[B],
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"}];
```

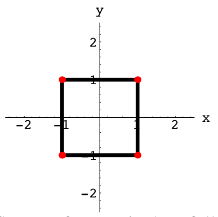


A genuine rectangle.
Why did you get a rectangle?
What does the area of this rectangle measure out to?

□ G.3.b.iii) Getting a square

Here's the same set up as in part i) immediately above:

```
corners = {{-1, -1}, {1, -1}, {1, 1}, {-1, 1}, {-1, -1}};
ranger = 2.5;

Clear[hitplotter, matrix];
hitplotter[matrix_] :=
  {Graphics[{Black, Thickness[0.02], Line[Table[
       matrix.corners[[k]], {k, 1, Length[corners]}]]}], Table[
    Graphics[{PointSize[0.035], Red, Point[matrix.corners[[k]]]}],
    {k, 1, Length[corners]}]};

Show[hitplotter[IdentityMatrix[2]],
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"}];
```



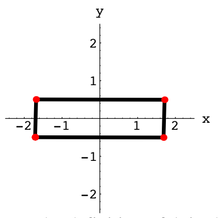
Set up a 2D matrix A as follows:

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

hanger = Transpose[{perpframe[1], perpframe[2]}];
A = DiagonalMatrix[{1.7, 0.5}].hanger;

MatrixForm[A]
```

$$\begin{pmatrix} 1.69995 & 0.0133754 \\ -0.00393395 & 0.499985 \end{pmatrix}$$

Hit the square with A:

```
Show[hitplotter[A],
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"}];
```



Note the definition of A in the code above:
A = DiagonalMatrix[**{1.7,0.5}**].hanger
Change the number 0.5 to a new number so that the result is a square.
Show off your square with a decisive plot.
Try to explain why your response was guaranteed to work.
What does the area enclosed by this tilted square measure out to?

G.4) Rows, columns and the transpose of a 2D matrix*

□G.4.a.ii) Columns are vertical

Here is a cleared 2D matrix:

```
Clear[a, b, c, d];
A = (a  b
     c  d);
MatrixForm[A]
```

```
(a  b
 c  d)
```

Most folks say that the first column of A is
column1 = {a, c},
and the second column of A is
column2 = {b, d}.
Look at these:

```
Clear[x, y];
A.{x, y}
```

```
{a x + b y, c x + d y}
```

```
column1 = {a, c};
column2 = {b, d};
x column1 + y column2
```

```
{a x + b y, c x + d y}
```

The upshot:

```
A.{x, y} == x column1 + y column2
```

```
True
```

No dot products here.

It works every time.
To hit {x,y} with A, you can just calculate:
A.{x,y} = x column1 + y column2
This gives you another easy way of calculating matrix hits by hand without any dot products.
To illustrate, go with:

```
A = (-2   1
      0  -3);
MatrixForm[A]
```

```
(-2   1
  0  -3)
```

And use the two columns of A to give a hand calculation of
A.{x,y}.
Check yourself with *Mathematica* if you like.

□G.4.a.ii) A.{1,0} = column1 and A.{0,1} = column2

Here's a random 2D matrix:

```
Clear[i, j];
A = Table[Random[Real, {-5, 5}], {i, 1, 2}, {j, 1, 2}];
MatrixForm[A]
```

```
( 1.41388   3.50339
 0.299555   1.53808)
```

You can get the first vertical column of A by hitting {1,0} with A.
Try it:

```
A.{1, 0}
```

```
{1.41388, 0.299555}
```

And you can get the second vertical column of A by hitting {0,1} with A.
Try it:

```
A.{0, 1}
```

```
{3.50339, 1.53808}
```

Use the fact that
A.{x,y} = x column1 + y column2
to explain why
A.{1,0} = column1
and
A.{0,1} = column2

□G.4.a.iii) What happens when column2 is a multiple of column1

Here's how you make a random matrix so that the second vertical column is a multiple of the first vertical column:

```
column1 = {Random[Real, {-2, 2}], Random[Real, {-2, 2}]}
```

```
{-0.133994, 0.0557171}
```

```
column2 = Random[Real, {-1, 1}] column1
```

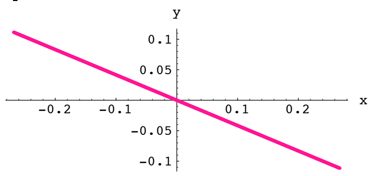
```
{0.00414284, -0.00172267}
```

```
A = Transpose[{column1, column2}];
MatrixForm[A]
```

```
(-0.133994   0.00414284
 0.0557171  -0.00172267)
```

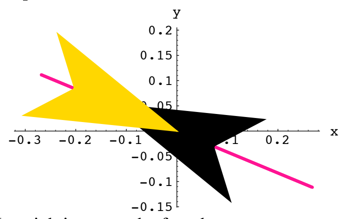
Look at what happens when you hit A on the circle of radius 2 centered at {0,0}:

```
hitplot = ParametricPlot[A.{2 Cos[t], 2 Sin[t]},
  {t, 0, 2 π}, PlotStyle -> {{Thickness[0.01], DeepPink}},
  AxesLabel -> {"x", "y"}];
```



The hit with A squashed the circle onto a line.
Throw in plots of the column vectors:

```
Show[hitplot, Arrow[column1,
   Tail -> {0, 0}, VectorColor -> Black, HeadSize -> 0.3],
  Arrow[column2, Tail -> {0, 0}, VectorColor -> Gold,
   HeadSize -> 0.3]];
```



Your job is to use the fact that
A{x,y} = x column1 + y column2
to explain why this turned out the way it did.

□G.4.b.i) Rows are horizontal

Here is a cleared 2D matrix:

```
Clear[a, b, c, d];
A = (a  b
     c  d);
MatrixForm[A]
```

```
(a  b
 c  d)
```

Most folks say that the first row of A is
row1 = {a,b},
and the second row of A is
row2 = {c,d}.

```
row1 = {a, b};
row2 = {c, d};
A == {row1, row2}
```

```
True
```

To hit {x,y} with A using the rows of A, you just calculate two dot products this way
A.{x,y} = {row1.{x,y}, row2.{x,y}.
Try it out:

```
Clear[x, y];
A.{x, y}
{row1.{x, y}, row2.{x, y}}
```

```
{a x + b y, c x + d y}
```

```
{a x + b y, c x + d y}
```

It works every time.
This gives you an easy way of calculating matrix hits by hand.
To illustrate, go with:

```
A = (2.  -1.
     0    3.);
MatrixForm[A]
```

```
(2.  -1.
 0    3.)
```

And give a hand calculation of
A.{x,y}.
Check yourself with *Mathematica* if you like.

□G.4.b.ii) What happens when row2 is a multiple of row1

Here's how you make a random matrix so that the second horizontal row is a multiple of the first horizontal row:

```
row1 = {Random[Real, {-2, 2}], Random[Real, {-2, 2}]}
```

```
{-0.997562, 0.920857}
```

Calculate a number s:

```
s = Random[Real, {-1, 1}]
```

```
0.636212
```

Set
row2 = s row1:

```
row2 = s row1
```

```
{-0.634661, 0.585861}
```

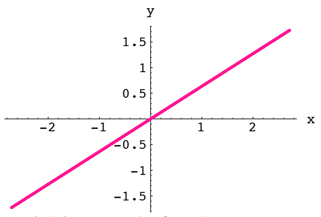
Enter row1 and row2 into a matrix A:

```
A = {row1, row2};
MatrixForm[A]
```

```
( -0.997562  0.920857 )
( -0.634661  0.585861 )
```

Look at what happens when you hit A on the circle of radius 2 centered at {0,0}:

```
hitplot = ParametricPlot[A.{2 Cos[t], 2 Sin[t]},
  {t, 0, 2 π}, PlotStyle -> {{Thickness[0.01], DeepPink}},
  AxesLabel -> {"x", "y"}];
```



Your job is to use the fact that

A.{x,y} = {row1.{x,y}, row2.{x,y} = {row1.{x,y}, s row1.{x,y}.

to explain why this turned out the way it did.

How is the slope of this line related to:

```
s
```

```
0.636212
```

□G.4.c.i) The transpose A^t of a 2D matrix A

Here is a cleared 2D matrix:

```
Clear[a, b, c, d];
A = ( a  b );
    ( c  d )
MatrixForm[A]
```

```
( a  b )
( c  d )
```

Here's *Mathematica*'s calculation of A^t, the transpose of A:

```
MatrixForm[Transpose[A]]
```

```
( a  c )
( b  d )
```

How are the rows and the columns of A related to the rows and columns of A^t, the transpose of A?

□G.4.c.ii) Using the transpose to enter a matrix via columns

Entering a matrix through rows is duck soup. For instance if you want the matrix A whose

first row is {-2.9,7.2}

and whose

second row is {4.7,6.1},

all you have to do is to put:

```
row1 = {-2.9, 7.2};
row2 = {4.7, 6.1};
A = {row1, row2};

MatrixForm[A]
```

```
( -2.9  7.2 )
(  4.7  6.1 )
```

If you want to enter a matrix through its columns quickly, then the transpose is just your ticket.

For instance if you want the matrix A whose

first column is {7.1,-3.2}

and whose

second column is {-4.9 ,-0.5},

all you have to do is to enter these columns as rows of a matrix B:

```
column1 = {7.1, -3.2};
column2 = {-4.9, -0.5};
B = {column1, column2};

MatrixForm[B]
```

```
(  7.1  -3.2 )
( -4.9  -0.5 )
```

The matrix A whose

first column is {7.1,-3.2}

and whose

second column is {-4.9 ,-0.5} is:

```
A = Transpose[B];
MatrixForm[A]
```

```
(  7.1  -4.9 )
( -3.2  -0.5 )
```

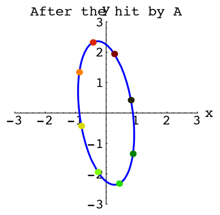
Got any idea why that worked?

□G.4.d) Hits with A and A^t

Here is a random matrix A shown with the ellipse you get when you hit the unit circle with A:

```
A = {{Random[Real, {-2, 2}], Random[Real, {-2, 2}]},
    {Random[Real, {-2, 2}], Random[Real, {-2, 2}]}};
MatrixForm[A]

Clear[x, y, t, hitplotter, hitpointplotter, pointcolor, matrix2D];
{tlow, thigh} = {0, 2 π};
ranger = 3;
{x[t_], y[t_]} = {Cos[t], Sin[t]};

pointcolor[t_] = RGBColor[0.5 (Cos[t] + 1), 0.5 (Sin[t] + 1), 0];

       thigh - tlow
jump = ------------;
            8

hitplotter[matrix2D_] := ParametricPlot[matrix2D.{x[t], y[t]},
    {t, tlow, thigh}, PlotStyle → {{Thickness[0.01], Blue}},
    PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
    AxesLabel → {"x", "y"}, DisplayFunction → Identity];

hitpointplotter[matrix2D_] :=
  Table[Graphics[{pointcolor[t], PointSize[0.035],
     Point[matrix2D.{x[t], y[t]}]}], {t, tlow, thigh - jump, jump}];

Ahit = Show[hitplotter[A],
    hitpointplotter[A], PlotLabel → "After the hit by A",
    DisplayFunction → $DisplayFunction];
```

```
( -0.880416  -0.271138 )
(  1.34279   -1.943     )
```



```
If you don't get a nice fat, inviting ellipse, then rerun.
```

Look at A:

```
MatrixForm[A]
```

```
( -0.880416  -0.271138 )
(  1.34279   -1.943     )
```

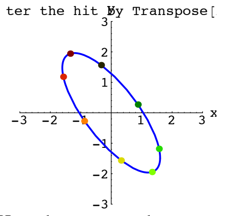
Here's how A^t, the transpose of A, looks:

```
MatrixForm[Transpose[A]]
```

```
( -0.880416   1.34279 )
( -0.271138  -1.943   )
```

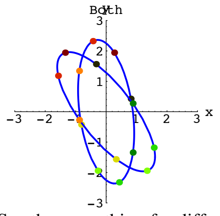
Here's the ellipse you get when you hit the unit circle with A^t:

```
transposeAhit =
  Show[hitplotter[Transpose[A]], hitpointplotter[Transpose[A]],
   PlotLabel → "After the hit by Transpose[A]",
   DisplayFunction → $DisplayFunction];
```



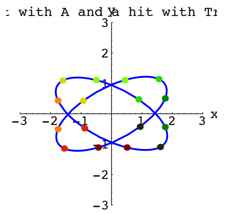
Here they are together:

```
Show[Ahit, transposeAhit,
  PlotLabel → "Both", AspectRatio → Automatic];
```



See the same thing for different random matrices A:

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

Show[hitplotter[A], hitpointplotter[A], hitplotter[Transpose[A]],
  hitpointplotter[Transpose[A]], AspectRatio → Automatic,
  PlotLabel → "After a hit with A and a hit with Transpose[A]",
  DisplayFunction → $DisplayFunction];
```



`Rerun at least five times.`

And then use your visual evidence to describe:
In what ways do the ellipses seem to be different?
In what ways do the ellipses seem to be similar?

G.5) Calculating A.B through hitting A on the columns of B and through dot products of rows of A and columns of B*

□G.5.a.i) To get the first column of A.B , you hit A on the first column of B. To get the second column of A.B , you hit A on the second column of B

Here are two cleared 2D matrices A and B together with *Mathematica*'s calculation of A.B:

```
Clear[x, y, z, w, a, b, c, d];
A = (a  b
     c  d);

B = (x  y
     z  w);

MatrixForm[A.B]
```

$$\begin{pmatrix} a\,x+b\,z & b\,w+a\,y \\ c\,x+d\,z & d\,w+c\,y \end{pmatrix}$$

The first vertical column of B is:

```
firstcolB = {x, z}
```

```
{x, z}
```

Here's what you get when you hit A on the first vertical column of B:

```
A.firstcolB
```

```
{a x + b z, c x + d z}
```

▪ How is this related to the first vertical column of A.B?
Put answer here.

▪ What vector do you hit A on to get the second vertical column of A.B?
Put answer here.

□G.5.b.i) Calculating A.B through dot products of rows of A and columns of B

Here are two cleared 2D matrices A and B together with *Mathematica*'s calculation of A.B:

```
Clear[x, y, z, w, a, b, c, d];
A = (a  b
     c  d);

B = (x  y
     z  w);

MatrixForm[A.B]
```

$$\begin{pmatrix} a\,x+b\,z & b\,w+a\,y \\ c\,x+d\,z & d\,w+c\,y \end{pmatrix}$$

Study the pattern and then describe the how the entries in A.B are related to dot products of rows of A and columns of B:

`Click on the right for a healthy tip.`

Look again

```
Clear[x, y, z, w, a, b, c, d];
A = (a  b
     c  d);

B = (x  y
     z  w);

MatrixForm[A.B]
```

$$\begin{pmatrix} a\,x+b\,z & b\,w+a\,y \\ c\,x+d\,z & d\,w+c\,y \end{pmatrix}$$

The first row of A is:

```
row1A = {a, b}
```

```
{a, b}
```

The first column of B is:

```
col1B = {x, z}
```

```
{x, z}
```

Their dot product is:

```
row1A.col1B
```

```
a x + b z
```

That's one of the entries in:

```
MatrixForm[A.B]
```

$$\begin{pmatrix} a\,x+b\,z & b\,w+a\,y \\ c\,x+d\,z & d\,w+c\,y \end{pmatrix}$$

□G.5.b.ii) $\begin{pmatrix} \text{prepframe[1]} \rightarrow \\ \text{perpframe[2]} \rightarrow \end{pmatrix} . \begin{pmatrix} \text{prepframe[1]} & \text{perpframe[2]} \\ \downarrow & \downarrow \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Go with any perpendicular frame {perpframe[1],perpframe[2]}.
You make the corresponding aligner matrix by putting the perpendicular frames into the rows of the aligner.

$$\text{aligner} = \begin{pmatrix} \text{prepframe[1]} \rightarrow \\ \text{perpframe[2]} \rightarrow \end{pmatrix}$$

.You make the corresponding hanger matrix by putting the perpendicular frames into the columns of the hanger.

$$\text{hanger} = \begin{pmatrix} \text{prepframe[1]} & \text{perpframe[2]} \\ \downarrow & \downarrow \end{pmatrix}$$

Use the dot product method you developed in part i) above to explain why

$$\text{aligner.hanger} = \begin{pmatrix} \text{prepframe[1]} \rightarrow \\ \text{perpframe[2]} \rightarrow \end{pmatrix} . \begin{pmatrix} \text{prepframe[1]} & \text{perpframe[2]} \\ \downarrow & \downarrow \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

□G.5.c) A.B and B.A

Here are two random matrices A and B the products A.B and B.A:

```
A = {{Random[Real, {-8, 8}], Random[Real, {-8, 8}]},
     {Random[Real, {-8, 8}], Random[Real, {-8, 8}]}};

B = {{Random[Real, {-8, 8}], Random[Real, {-8, 8}]},
     {Random[Real, {-8, 8}], Random[Real, {-8, 8}]}};

MatrixForm[A.B]
```

```
( 23.6192    25.841   )
( -33.3844   0.962608 )
```

```
MatrixForm[B.A]
```

```
( 9.88476  -45.4761 )
( 16.2755   14.697  )
```

```
A.B == B.A
```

```
False
```

Rerun all parts several times.
Do the outcomes surprise you?
Why or why not?

G.6) Inverse Matrices*

□G.6.a) Inverting stretcher matrices

For a given 2D matrix A, the inverse matrix A^{-1} is the matrix you hit with to undo whatever a hit with A did.
For instance, when you hit with this stretcher matrix:

```
{xstretch, ystretch} = {2.0, 0.25};
stretcher = DiagonalMatrix[{xstretch, ystretch}];

MatrixForm[stretcher]
```

```
( 2.   0    )
( 0    0.25 )
```

You stretch everything by a factor of 2.0 units along the x-axis and by a factor of 0.25 units long the y-axis. To undo this, you put:

$$\texttt{unstretcher = DiagonalMatrix}\left[\left\{\frac{1}{\texttt{xstretch}}, \frac{1}{\texttt{ystretch}}\right\}\right];$$

```
MatrixForm[unstretcher]
```

```
( 0.5  0  )
( 0    4. )
```

This is the inverse matrix A^{-1} of A.
Confirm with *Mathematica*:

```
MatrixForm[Inverse[stretcher]]
```

```
( 0.5  0  )
( 0    4. )
```

You take over any explain in terms of stretches why the product
unstretcher.stretcher
gives you the identity matrix, the matrix that does nothing.

```
MatrixForm[unstretcher.stretcher]
```

```
( 1.  0  )
( 0   1. )
```

□G.6.b.i) Inverting hanger matrices

For a given 2D matrix A, the inverse matrix A^{-1} is the matrix you hit with to undo whatever a hit with A did.
For instance, when you hit with this hanger matrix:

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

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

hanger = Transpose[{perpframe[1], perpframe[2]}];

MatrixForm[hanger]
```

```
( 0.866778   0.498693 )
(-0.498693   0.866778 )
             perpframe[1] is the first vertical column of hanger.
            perpframe[2] is the second vertical column of hanger.
```

A hit with hanger hangs everything on the given perpendicular frame with
perpframe[1] playing the former role of {1, 0}pointing out the positive x-axis
and with
perpframe[2] playing the former role {0, 1}pointing out the positive y-axis.
To undo this, you put:

```
aligner = {perpframe[1], perpframe[2]};
MatrixForm[aligner]
```

```
(0.866778  -0.498693)
(0.498693   0.866778)
             perpframe[1] is the first horizontal row of aligner.
            perpframe[2] is the second horizontal row of aligner.
```

This is the inverse matrix hanger^{-1} of hanger.
Confirm with *Mathematica*:

```
MatrixForm[Inverse[hanger]]
```

```
(0.866778  -0.498693)
(0.498693   0.866778)
```

Remembering that a hit with aligner aligns everything on the 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],
You take over any explain in terms of hanging and aligning why the it is totally natural that
aligner = hanger^{-1}.

□ G.6.b.ii) Inverting rotation matrices

Hits with the following matrix rotate everything about {0,0} by s counterclockwise radians:

```
Clear[s];
{perpframe[1], perpframe[2]} =


  N[{{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}}];

{a, c} = perpframe[1];
{b, d} = perpframe[2];
rotator[s_] = {{a, b}, {c, d}};

MatrixForm[rotator[s]]
```

```
(Cos[s]  -1. Sin[s])
(Sin[s]     Cos[s] )
```

Explain, in terms of the meaning of the value of s, why it is totally natural that rotator[s]$^{-1}$
is given rotator[– s]

```
MatrixForm[rotator[-s]]
```

```
( Cos[s]   1. Sin[s])
(-Sin[s]     Cos[s] )
```

Confirm with *Mathematica*:

```
MatrixForm[Simplify[Inverse[rotator[s]]]]
```

```
( 1. Cos[s]   1. Sin[s])
(-1. Sin[s]   1. Cos[s])
```

□ G.6.c) Inverting aligner matrices

For a given 2D matrix A, the inverse matrix A^{-1}is the matrix you hit with to undo
whatever a hit with A did.
For instance, when you hit with this aligner matrix:

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

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

aligner = {perpframe[1], perpframe[2]};

MatrixForm[aligner]
```

```
(0.00793194   0.999969  )
(-0.999969   0.00793194 )
             perpframe[1] is  the first horizontal row of aligner.
            perpframe[2] is  the second horizontal row of aligner.
```

A hit with aligner aligns everything on the x and y axes with
{1, 0}pointing along of the positive x-axis playing the former role of perpframe[1]
and with
{0, 1}pointing along of the positive x-axis playing the former role of perpframe[2].

Here is *Mathematica*'s calculation of the inverse of this aligner matrix:

```
MatrixForm[Inverse[aligner]]
```

```
(0.00793194  -0.999969 )
( 0.999969   0.00793194)
```

Duplicate this inverse matrix by calculating a hanger matrix.

□ G.6.d.i) Inverting products

Here is a random hanger matrix:

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

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

hanger = Transpose[{perpframe[1], perpframe[2]}];
MatrixForm[hanger]
```

```
( 0.744573  0.667541)
(-0.667541  0.744573)
```

Here is a random stretcher matrix:

```
xstretch = Random[Real, {2.0, 4.0}];
ystretch = Random[Real, {0.4, 1.9}];
stretcher = {{xstretch, 0}, {0, ystretch}};

MatrixForm[stretcher]
```

```
(3.61809     0    )
(   0     1.67391 )
```

Here is a the matrix product
hanger.stretcher

```
product = hanger.stretcher;
MatrixForm[product]
```

```
( 2.69393   1.1174 )
(-2.41523  1.24635 )
```

A hit with this matrix stretches and then hangs on the
Look at this calculation of
(hanger. stretcher)$^{-1}$,
the inverse of the product (hanger. stretcher) :

```
MatrixForm[Expand[Simplify[Inverse[product]]]]
```

```
(0.205792  -0.184501)
(0.398792   0.444811)
```

Compare with this calculation of
stretcher^{-1}.hanger^{-1}:

```
MatrixForm[Inverse[stretcher].Inverse[hanger]]
```

```
(0.205792  -0.184501)
(0.398792   0.444811)
```

Stare at both outputs to see that
(hanger. stretcher)$^{-1}$= stretcher^{-1}.hanger^{-1}.
This cannot be an accident. Explain why you think that is not an accident.

Click on the right for a tip suggested by an MGM student.

In the morning, you first put your socks on and then you put your shoes on.

At night, you first take your shoes off and then you take your socks off.

□ G.6.d.ii) $(A.B)^{-1} = B^{-1}.A^{-1}$

Here are two random matrices A and B:

```
A = ( Random[Real, {-2, 2}]  Random[Real, {-2, 2}] );
    ( Random[Real, {-2, 2}]  Random[Real, {-2, 2}] )
MatrixForm[A]
```

```
(0.987463  -0.782083)
( 1.96376   0.277506)
```

```
B = ( Random[Real, {-2, 2}]  Random[Real, {-2, 2}] );
    ( Random[Real, {-2, 2}]  Random[Real, {-2, 2}] )
MatrixForm[A]
```

```
(0.987463  -0.782083)
( 1.96376   0.277506)
```

Look at these calculations of
$(A.B)^{-1}$ and $B^{-1}.A^{-1}$:

```
MatrixForm[Inverse[A.B]]
```

```
(-0.616514  0.400009)
(  2.4324   -5.83526)
```

```
MatrixForm[Inverse[B].Inverse[A]]
```

```
(-0.616514  0.400009)
(  2.4324   -5.83526)
```

The upshot:

$(A.B)^{-1} = B^{-1}.A^{-1}$:

Part of the job of mathematics is to explain why calculations turn out the way they do.

Remembering that hitting with the inverse A^{-1} of a given matrix A undoes whatever a hit with A does, and remembering that hitting with the inverse B^{-1} of a given matrix B undoes whatever a hit with B does, try to explain the formula

$(A.B)^{-1} = B^{-1}.A^{-1}$

Click on the right for a tip.

In real life:

In the morning, you first put your **socks** on and then you put your **shoes** on.

At night, you first take your **shoes** off and then you take your **socks** off.

In matrices:

When you hit with **A**.B **,** you first hit with B and then you hit with **A**.

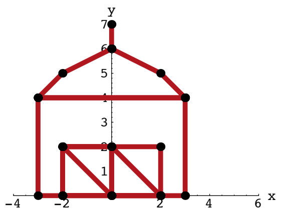
So the first phase of undoing a hit with **A**.B is to undo the hit with **A**. This you do by hitting first with A^{-1}.

The next phase is to . . . (you take over here).

G.6.e) Inverting shear matrices

Take a look at the old barn owned by Farmer Ben:

```
barn = {{0, 7}, {0, 6}, {2, 5}, {3, 4}, {-3, 4}, {3, 4},
    {3, 0}, {2, 0}, {0, 0}, {2, 0}, {2, 2}, {-2, 2}, {-2, 0},
    {2, 0}, {0, 2}, {0, 0}, {-2, 2}, {-2, 0}, {0, 0}, {0, 2},
    {-2, 2}, {-2, 0}, {-3, 0}, {-3, 4}, {-2, 5}, {0, 6}};

Clear[hitplotter, matrix];
hitplotter[matrix_] := {Graphics[{IndianRed, Thickness[0.02],
     Line[Table[matrix.barn[[k]], {k, 1, Length[barn]}]]}],
   Table[Graphics[{PointSize[0.035], Point[matrix.barn[[k]]]}],
    {k, 1, Length[barn]}]};

Show[hitplotter[IdentityMatrix[2]], PlotRange → {{-4, 6}, {0, 7}},
  Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"}];
```

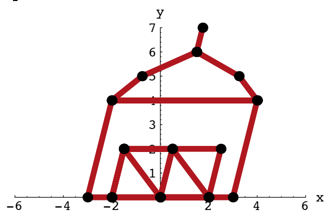


See what happens when you hit this barn with the shear matrix

$$\text{shear}[b] = \begin{pmatrix} 1.0 & b \\ 0.0 & 1.0 \end{pmatrix}$$

with b = 0.25:

```
Clear[shear, b];
shear[b_] = {{1, b}, {0, 1.0}};

Show[hitplotter[shear[0.25]], PlotRange → {{-6, 6}, {0, 7}},
  Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"}];
```

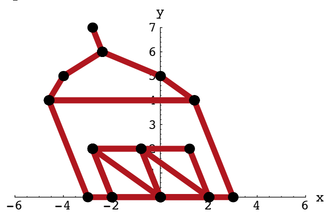


See what happens when you hit this barn with the shear matrix

shear[b] = {{1.0, b},{0,1.0}}

with b = -0.4:

```
Show[hitplotter[shear[-0.4]], PlotRange → {{-6, 6}, {0, 7}},
  Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"}];
```



For a cleared b, *Mathematica* calculates the inverse of

$$\text{shear}[b] = \begin{pmatrix} 1.0 & b \\ 0 & 1.0 \end{pmatrix}$$

as follows.

```
Clear[b];
MatrixForm[Inverse[shear[b]]]
```

$$\begin{pmatrix} 1. & -1.\,b \\ 0 & 1. \end{pmatrix}$$

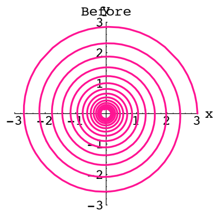
Remembering that the inverse matrix ,$\text{shear}[b]^{-1}$,is the matrix you hit with to undo whatever a hit with shear[b] did, use the meaning of the value of b to help give a short explanation about why *Mathematica*'s calculation of $\text{shear}[b]^{-1}$ is right on the money.

G.7) Rotations versus flips: Right versus left

G.7.a) Whirl

Here's a spiral ready to be hit with a 2D matrix:

```
Clear[x, y, t, hitplotter,
  hitpointplotter, pointcolor, actionarrows, matrix2D];
{tlow, thigh} = {0, 15};
ranger = 3;
{x[t_], y[t_]} = 3 E^(-0.2 t) {Cos[6 t], Sin[6 t]};

hitplotter[matrix2D_] := ParametricPlot[matrix2D.{x[t], y[t]},
   {t, tlow, thigh}, PlotStyle → {{Thickness[0.01], DeepPink}},
   PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
   AxesLabel → {"x", "y"}, DisplayFunction → Identity];

before = Show[hitplotter[IdentityMatrix[2]],
   PlotLabel → "Before", DisplayFunction → $DisplayFunction];
```



Use hits with rotation matrices to make a movie showing this spiral whirl in the counterclockwise way around {0,0}.

G.7.b.i) Rotation matrices versus hanger matrices

To get a matrix A whose hits rotate by s radians , you go with this right hand perpendicular frame:

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

```
{{Cos[s], Sin[s]}, {-1. Sin[s], Cos[s]}}
```

To get the rotation matrix whose hits rotate by s radians, go with cleared entries in a matrix A:

```
Clear[a, b, c, d];
rotator = {{a, b}, {c, d}};
MatrixForm[rotator]
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Then you need:

```
rotator.{1, 0} == perpframe[1]
```

```
{a, c} == {Cos[s], Sin[s]}
```

```
rotator.{0, 1} == perpframe[2]
```

```
{b, d} == {-1. Sin[s], Cos[s]}
```

This tells you that to make the first vertical column of rotator is perpframe[1] and the second vertical column of rotator is perpframe[2].

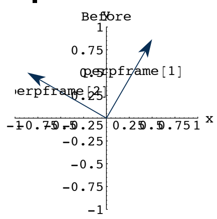
How does this information reveal the relation between this rotation matrix and the hanger matrix for the given right hand perpendicular frame?

G.7.b.ii) Hangers: Right versus left

According to the part immediately above, when you go with a right hand perpendicular frame, the corresponding hanger is a rotation matrix.

See it happen for this right hand perpendicular frame:

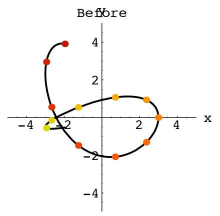
```
s = π/3;
{perpframe[1], perpframe[2]} =
    N[{{Cos[s], Sin[s]}, {Cos[s + Pi / 2], Sin[s + Pi / 2]}}];

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]]]};

ranger = 1;
 Show[frameplot,
      PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
      Axes -> True, AxesLabel -> {"x", "y"}, PlotLabel -> "Before"];
```

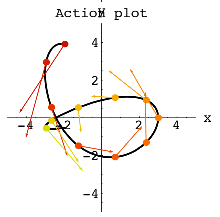


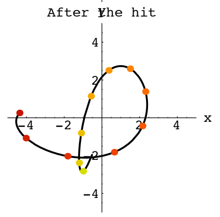
See a hit with hanger rotate a curve around {0,0}:

```
hanger = Transpose[{perpframe[1], perpframe[2]}];
A = hanger;

Clear[x, y, t, hitplotter,
  hitpointplotter, pointcolor, actionarrows, matrix2D];
{tlow, thigh} = {-1, 1};
ranger = 5;
{x[t_], y[t_]} = {3 Cos[4 t], 1.5 E^-t Sin[5 t]};

pointcolor[t_] = RGBColor[0.5 (Cos[t] + 1), 0.5 (Sin[t] + 1), 0];

jump = (thigh - tlow)/12;

hitplotter[matrix2D_] := ParametricPlot[matrix2D.{x[t], y[t]},
    {t, tlow, thigh}, PlotStyle → {{Thickness[0.01], Black}},
    PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
    AxesLabel → {"x", "y"}, DisplayFunction → Identity];

hitpointplotter[matrix2D_] :=
  Table[Graphics[{pointcolor[t], PointSize[0.035],
      Point[matrix2D.{x[t], y[t]}]}], {t, tlow, thigh - jump, jump}];

actionarrows[matrix2D_] :=
  Table[Arrow[matrix2D.{x[t], y[t]} - {x[t], y[t]}, Tail → {x[t], y[t]
    VectorColor → pointcolor[t], HeadSize → 0.25],
   {t, tlow, thigh - jump, jump}];

before = Show[hitplotter[IdentityMatrix[2]],
    hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
    DisplayFunction → $DisplayFunction];
```

```
Show[before, actionarrows[A],
  PlotLabel → "Action plot", DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
```



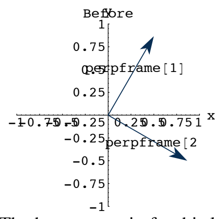




Grab all three plots, align and animate slowly.

Now go with this left hand perpendicular frame:

```
s = π/3;
{perpframe[1], perpframe[2]} =
    N[{{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]]]};

ranger = 1;
Show[frameplot,
     PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
     Axes -> True, AxesLabel -> {"x", "y"}, PlotLabel -> "Before"];
```



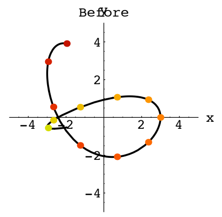
The hanger matrix for this left hand perpendicular frame is:

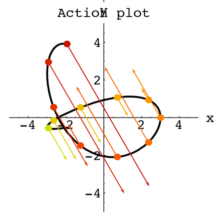
```
hanger = Transpose[{perpframe[1], perpframe[2]}];
MatrixForm[hanger]
```

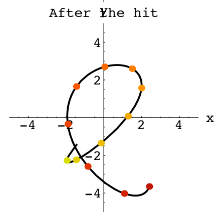
$$\begin{pmatrix} 0.5 & 0.866025 \\ 0.866025 & -0.5 \end{pmatrix}$$

See an action movie showing what a hit with this hanger matrix does to the same curve plotted above:

```
A = hanger;
ranger = 5;
before = Show[hitplotter[IdentityMatrix[2]],
     hitpointplotter[IdentityMatrix[2]], PlotLabel -> "Before",
     DisplayFunction -> $DisplayFunction];

 Show[before,
    actionarrows[A], PlotLabel -> "Action plot",
    DisplayFunction -> $DisplayFunction];

 Show[hitplotter[A],
    hitpointplotter[A],
    PlotLabel -> "After the hit",
    DisplayFunction -> $DisplayFunction];
```







Grab all three plots, align and animate slowly.

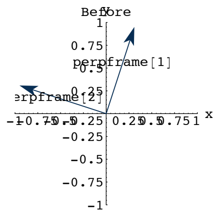
You make the call
Is this hanger matrix a rotation matrix?

□G.7.b.iii) Aligners: Right versus left

When you go with a right hand perpendicular frame, the corresponding aligner is a rotation matrix.
Reason. When you go with a right hand frame, then the corresponding hanger is a rotation and because the corresponding aligner is the inverse of the hanger, the corresponding aligner is also a rotation.
See it happen for this right hand perpendicular frame:

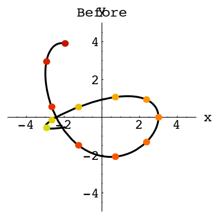
```
s = 0.4 π;
{perpframe[1], perpframe[2]} =
     N[{{Cos[s], Sin[s]}, {Cos[s + Pi/2], Sin[s + Pi/2]}}];
```

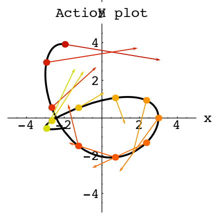
```
frameplot = {Table[
   Arrow[perpframe[k], Tail -> {0, 0},
      VectorColor -> Indigo, HeadSize -> 0.2], {k, 1, 2}],
   Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
   Graphics[Text["perpframe[2]", 0.6 perpframe[2]]]};

ranger = 1;
 Show[frameplot,
      PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
      Axes -> True, AxesLabel -> {"x", "y"}, PlotLabel -> "Before"];
```

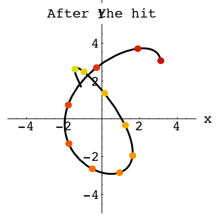


See a hit with the aligner corresponding to this frame rotate a curve around {0,0}:

```
aligner = {perpframe[1], perpframe[2]};
A = aligner;

Clear[x, y, t, hitplotter,
  hitpointplotter, pointcolor, actionarrows, matrix2D];
{tlow, thigh} = {-1, 1};
ranger = 5;
{x[t_], y[t_]} = {3 Cos[4 t], 1.5 E^-t Sin[5 t]};

pointcolor[t_] = RGBColor[0.5 (Cos[t] + 1), 0.5 (Sin[t] + 1), 0];

jump = (thigh - tlow)/12;

hitplotter[matrix2D_] := ParametricPlot[matrix2D.{x[t], y[t]},
   {t, tlow, thigh}, PlotStyle → {{Thickness[0.01], Black}},
   PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
   AxesLabel → {"x", "y"}, DisplayFunction → Identity];

hitpointplotter[matrix2D_] :=
  Table[Graphics[{pointcolor[t], PointSize[0.035],
     Point[matrix2D.{x[t], y[t]}]}], {t, tlow, thigh - jump, jump}];

actionarrows[matrix2D_] :=
  Table[Arrow[matrix2D.{x[t], y[t]} - {x[t], y[t]}, Tail → {x[t], y[t]}
    VectorColor → pointcolor[t], HeadSize → 0.25],
   {t, tlow, thigh - jump, jump}];

before = Show[hitplotter[IdentityMatrix[2]],
   hitpointplotter[IdentityMatrix[2]], PlotLabel → "Before",
   DisplayFunction → $DisplayFunction];

Show[before, actionarrows[A],
  PlotLabel → "Action plot", DisplayFunction → $DisplayFunction];

Show[hitplotter[A], hitpointplotter[A],
  PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
```



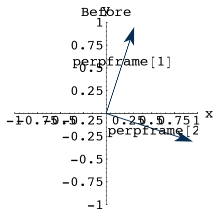




Grab all three plots, align and animate slowly.

Now go with this left hand perpendicular frame:

```
s = 0.4 Pi;
{perpframe[1], perpframe[2]} =
    N[{{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]]]};

ranger = 1;
Show[frameplot,
      PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
      Axes -> True, AxesLabel -> {"x", "y"}, PlotLabel -> "Before"];
```



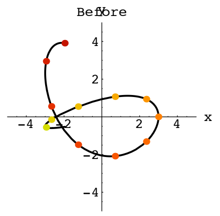
The aligner matrix for this left hand perpendicular frame is:

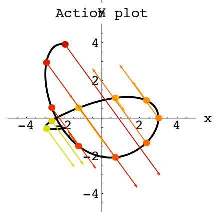
```
aligner = {perpframe[1], perpframe[2]};
MatrixForm[hanger]
```

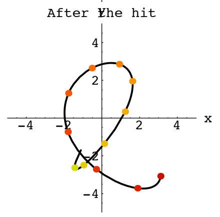
$$\begin{pmatrix} 0.5 & 0.866025 \\ 0.866025 & -0.5 \end{pmatrix}$$

See an action movie showing what a hit with this hanger matrix does to the same curve plotted above:

```
A = aligner;
ranger = 5;
before = Show[hitplotter[IdentityMatrix[2]],
     hitpointplotter[IdentityMatrix[2]], PlotLabel -> "Before",
     DisplayFunction -> $DisplayFunction];

 Show[before,
    actionarrows[A], PlotLabel -> "Action plot",
     DisplayFunction -> $DisplayFunction];

 Show[hitplotter[A],
     hitpointplotter[A],
     PlotLabel -> "After the hit",
     DisplayFunction -> $DisplayFunction];
```







Grab all three plots, align and animate slowly.

You make the call
Is this hanger matrix a rotation matrix?

□ G.7.b.iii) Rotations versus flips: Right versus left

Based on your experience, agree or disagree with each of the following statements.

When you hang on a right hand perpendicular frame, you are rotating.
Agree........ Disagree.........

When you hang on a left hand perpendicular frame, you are flipping.
Agree........ Disagree.........

When you align with respect to a right hand perpendicular frame, you are rotating.
Agree........ Disagree.........

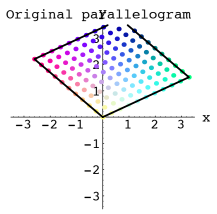
When you align with respect to a left hand perpendicular frame, you are flipping.
Agree........ Disagree.........

G.8) Matrices and parallelograms

□ G.8.a.i) Sides of the parallelogram in the columns of the matrix

Here's a parallelogram with lots of points inside:

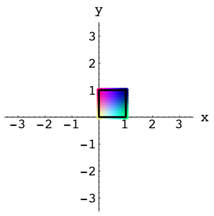
```
jump = 0.1;
Clear[parallelogramplotter, basepoint, side1, side2, pointcolor];
ranger = 3.5;
pointcolor[r_, t_] =
  RGBColor[0.5 (Cos[π t] + 1), 0.5 (Cos[π r] + 1), 0.5 (Sin[π t] + 1)];

parallelogramplotter[basepoint_, side1_, side2_] :=
  {Table[Graphics[{PointSize[0.025],
      pointcolor[r, t], Point[basepoint + t side1 + r side2]}],
    {t, 0, 1, jump}, {r, 0, 1, jump}], Graphics[
    {Thickness[0.01], Black, Line[{basepoint, basepoint + side1,
       basepoint + side1 + side2, basepoint + side2, basepoint}]}]};

basepoint = {0, 0};
paraside1 =
  {2.4 + Random[Real, {-1, 1}], 0.8 + Random[Real, {-1, 1}]};
paraside2 = {-2.0 + Random[Real, {-1, 1}],
   1.7 + Random[Real, {-1, 1}]};

paraplot =
  Show[parallelogramplotter[basepoint, paraside1, paraside2],
   PlotRange → {{-ranger, ranger}, {-ranger, ranger}}, Axes → True,
   AxesLabel → {"x", "y"}, PlotLabel → "Original parallelogram "];
```



Rerun until you get a nice one.

Here's the square with corners at {0,0}, {1,0},{1,1} and {0,1}

```
basepoint = {0, 0};
squareside1 = {1, 0};
squareside2 = {0, 1};

Show[parallelogramplotter[basepoint, squareside1, squareside2],
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y"}];
```



Use the two defining sides of the parallelogram to define a matrix A this way:

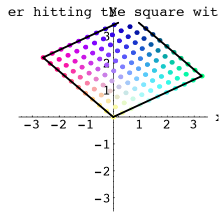
```
A = Transpose[{paraside1, paraside2}];
MatrixForm[A]
```

```
( 3.28197  -2.60127 )
( 1.51925   2.20903 )
```

paraside1 is the first vertical column of A
and paraside2 is the second vertical column of A

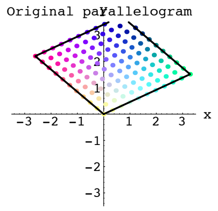
Here's what you get when you hit the square and the points inside with A:

```
Show[
  parallelogramplotter[A.basepoint, A.squareside1, A.squareside2],
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y"},
  PlotLabel → "After hitting the square with A"];
```



Here's the original parallelogram:

```
Show[paraplot];
```



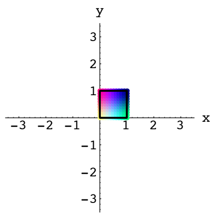
Grab the last two plots and animate.

Visually determine the relationship between this parallelogram and the original parallelogram. No calculations are asked for.

□ G.8.a.ii) Using a matrix to define the parallelogram

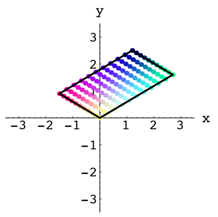
Here's the square with corners at {0,0}, {1,0}, {1,1} and {0,1}:

```
basepoint = {0, 0};
squareside1 = {1, 0};
squareside2 = {0, 1};

Show[parallelogramplotter[basepoint, squareside1, squareside2],
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y"}];
```



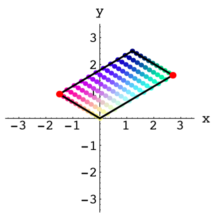
Here's what you get when you hit this square and the points inside with the matrix
$A = \begin{pmatrix} 2.7 & -1.5 \\ 1.6 & 0.9 \end{pmatrix}$:

```
A = ( 2.7  -1.5 );
    ( 1.6   0.9 )
basepoint = {0, 0};
squareside1 = {1, 0};
squareside2 = {0, 1};

hitsquare = Show[
   parallelogramplotter[A.basepoint, A.squareside1, A.squareside2],
   PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
   Axes → True, AxesLabel → {"x", "y"}];
```



Look at these specially marked points:

```
Show[hitsquare,
  Graphics[{PointSize[0.035], Red, Point[A.squareside1]}],
  Graphics[{PointSize[0.035], Red, Point[A.squareside2]}]];
```



When you look at A:

```
MatrixForm[A]
```

```
( 2.7  -1.5 )
( 1.6   0.9 )
```

You have enough information to give the exact locations at which these two points are centered.
Do it.

□G.8.b) Matrices hit parallel lines into parallel lines

The explanation of why hitting a parallelogram with a matrix gives another parallelogram in the Tutorials is not complete.
That explanation told you why you get a line when you hit a line with a matrix. It did not quite tell you why you get parallel lines when you hit two parallel lines with the same matrix.
Explain why this is guaranteed.

G.9) Using rotation matrices to explain why

Cos[s + t] = Cos[s] Cos[t] - Sin[s] Sin[t]

Sin[s + t] = Sin[s] Cos[t] + Sin[t] Cos[s].

□G.9.a) Using rotation matrices to explain why

Cos[s + t] = Cos[s] Cos[t] - Sin[s] Sin[t]

Sin[s + t] = Sin[s] Cos[t] + Sin[t] Cos[s].

When you use a rotation matrix to rotate about {0,0} by an angle s, you hit with the hanger corresponding to the right hand perpendicular frame coming from
{perpframe[1], perpframe[2]} :
= {{Cos[s], Sin[s]}, {Cos[s + $\frac{\pi}{2}$], Sin[s + $\frac{\pi}{2}$]}} :

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

rotator[s_] = Transpose[{perpframe[1], perpframe[2]}];

MatrixForm[rotator[s]]
```

```
( Cos[s]  -Sin[s] )
( Sin[s]   Cos[s] )
```

When you rotate by an angle s and then follow it up with a rotation by an angle t, you hit with:

```
MatrixForm[rotator[t].rotator[s]]
```

```
( Cos[s] Cos[t] - Sin[s] Sin[t]  -Cos[t] Sin[s] - Cos[s] Sin[t] )
( Cos[t] Sin[s] + Cos[s] Sin[t]   Cos[s] Cos[t] - Sin[s] Sin[t] )
```

But you can achieve a rotation by an angle s followed by a rotation by an angle t by a direct hit with a rotation by the angle (s + t)

```
MatrixForm[rotator[s + t]]
```

```
( Cos[s + t]  -Sin[s + t] )
( Sin[s + t]   Cos[s + t] )
```

Stare at the last two outputs for a minute and then answer this question:

How does this give you a clean matrix explanation of the trig identities
Cos[s + t] = Cos[s] Cos[t] − Sin[s] Sin[t]
Sin[s + t] = Cos[t] Sin[s] + Cos[s] Sin[t]?