Matrices, Geometry&*Mathematica*

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

Producer: Bruce Carpenter

Publisher: Math Everywhere, Inc.

MGM.02 2D Matrix Action

TUTORIALS

T.1) Hitting with a 2D matrix and visual assessment of the result

□T.1.a.i) A hit with $\begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}$ stretches:

Go with

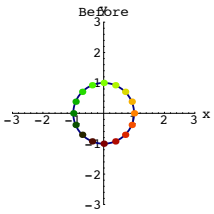
$$A = \begin{pmatrix} 2.0 & 0 \\ 0 & 3.0 \end{pmatrix}$$

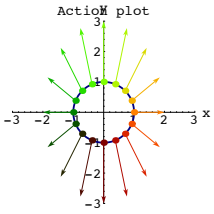
and use an action movie to analyze the action resulting from hitting the unit circle with A.

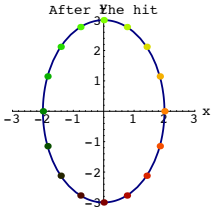
□Answer:

Hit it!

```
A = {{2.0, 0}, {0, 3.0}};
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];

jump = (thigh - tlow)/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];
```







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.

This matrix is a pure stretcher along the x and y axes.

□T.1.a.ii) A hit with $\begin{pmatrix} 0.5 & 0.0 \\ 0.0 & 0.7 \end{pmatrix}$ shrinks

Go with

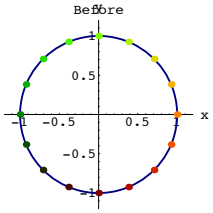
$$A = \begin{pmatrix} 0.5 & 0.0 \\ 0.0 & 0.7 \end{pmatrix}$$

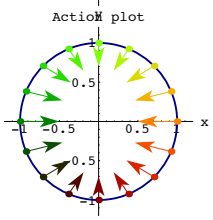
and use an action movie to analyze the action resulting from hitting the unit circle with A.

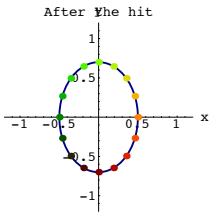
□Answer:

Hit it!

```
A = {{0.5, 0.0}, {0.0, 0.7}};

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.
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.

This matrix shrinks along the x and y axes.

□T.1.a.iii) A hit with $\begin{pmatrix} 2.0 & 0.0 \\ 0.0 & 0.3 \end{pmatrix}$ stretches and shrinks

Go with

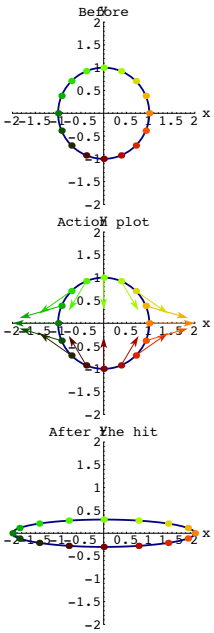
$$A = \begin{pmatrix} 2.0 & 0.0 \\ 0.0 & 0.3 \end{pmatrix}$$

and use an action movie to analyze the action resulting from hitting the unit circle with A.

□Answer:

Hit it!

```
A = {{2.0, 0.0}, {0.0, 0.3}};
ranger = 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];
```



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.

A hit with this matrix shrinks along the y-axis and stretches along the x-axis.

□T.1.a.iv) A hit with $\begin{pmatrix} 1.18 & -0.56 \\ 1.61 & 0.41 \end{pmatrix}$ stretches, shrinks and rotates

Go with

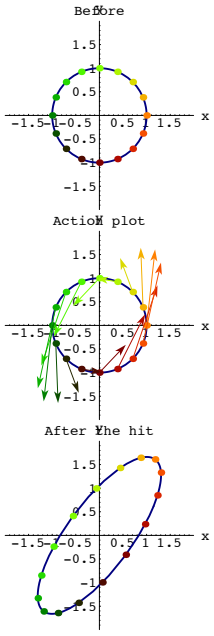
$$A = \begin{pmatrix} 1.18 & -0.56 \\ 1.61 & 0.41 \end{pmatrix}$$

and use an action movie to analyze the action resulting from hitting the unit circle with A.

□Answer:

Hit it!

```
A = {{1.18, -0.56}, {1.61, 0.41}};

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.
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.

A hit with this matrix shrinks, stretches and rotates.

□T.1.a.v) A hit with $\begin{pmatrix} 0 & 1.0 \\ 1.0 & 0 \end{pmatrix}$ flips

Go with

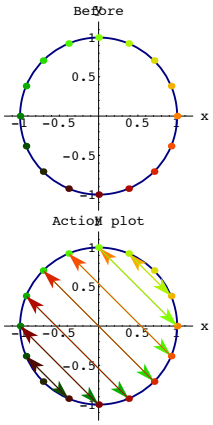
$$A = \begin{pmatrix} 0 & 1.0 \\ 1.0 & 0 \end{pmatrix}$$

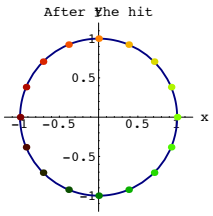
and use an action movie to analyze the action resulting from hitting the unit circle with A.

□Answer:

Hit it!

```
A = {{0, 1.0}, {1.0, 0}};
ranger = 1.4;

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.
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.

A hit with this matrix flips.

□T.1.a.vi) A hit with $\begin{pmatrix} 0.35 & 1.65 \\ 1.65 & 0.35 \end{pmatrix}$ flips and stretches

Go with

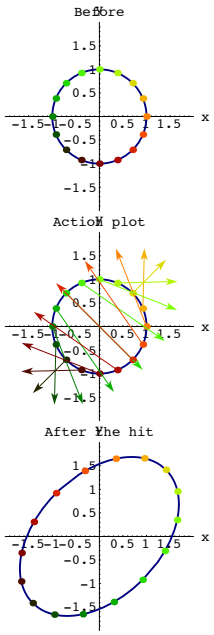
$$A = \begin{pmatrix} 0.35 & 1.65 \\ 1.65 & 0.35 \end{pmatrix}$$

and use an action movie to analyze the action resulting from hitting the unit circle with A.

□Answer:

Hit it!

```
A = {{0.35, 1.65}, {1.65, 0.35}};
ranger = 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];
```



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.

A hit with this matrix stretches and flips.

□ T.1.a.vii) A hit with this $\begin{pmatrix} 1.2 & 0.8 \\ 0.6 & 0.4 \end{pmatrix}$ rotates, stretches and smashes

Go with

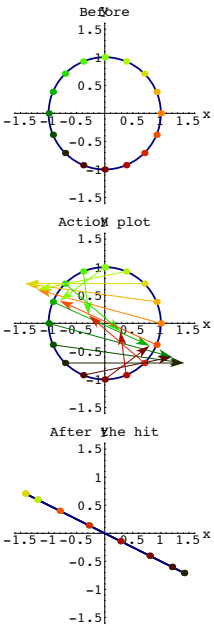
$$A = \begin{pmatrix} 1.2 & 0.8 \\ 0.6 & 0.4 \end{pmatrix}$$

and use an action movie to analyze the action resulting from hitting the unit circle with A.

□ Answer:

Hit it!

```
A = {{-1.2, -0.8}, {0.6, 0.4}};
ranger = 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];
```



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.

A hit with this matrix rotates, stretches and smashes.

T.2) Making your own 2D matrices: Rotations

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

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

□ T.2.a.i) Making a rotation matrix

Here is a cleared 2D matrix:

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

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

Look at what happens when you hit {1,0} with A:

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

{a, c}

A.{1,0} is the first vertical column of A

And look at look at what happens when you hit {0,1} with A:

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

{b, d}

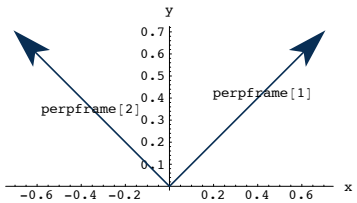
A.{0,1} is the second vertical column of A

How do you use this information to make a matrix that rotates everything s counterclockwise radians about {0,0}?

□ Answer:

To go after a matrix A whose hits rotate everything by s = $\frac{\pi}{4}$ counterclockwise radians about {0,0}, look at this RIGHT HAND perpendicular frame:

```
s = 0.25 π;
{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],
    {k, 1, 2}], Graphics[Text["perpframe[1]", 0.6 perpframe[1]]],
   Graphics[Text["perpframe[2]", 0.5 perpframe[2]]]};

Show[frameplot, Axes → True,
  AxesLabel → {"x", "y"}, AspectRatio → Automatic];
```



Remembering that {1,0} points in the direction of the positive x- axis and that {0,1} points in the direction of the positive y-axis, you want

A.{1,0} = perpframe[1]

and

A.{0,1} = perpframe[2].

Now look again at:

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

{a, c}

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

{b, d}

This tells you to put:

```
{a, c} = perpframe[1]
```

{0.707107, 0.707107}

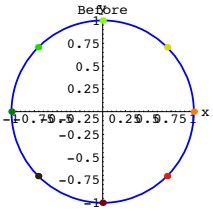
```
{b, d} = perpframe[2]
```

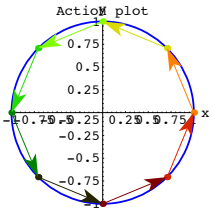
{-0.707107, 0.707107}

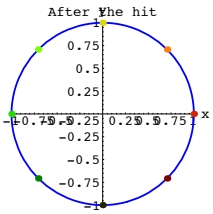
Watch hits with A rotate the unit circle $\frac{\pi}{4}$ counterclockwise radians about {0,0}:

```
A = {{a, b}, {c, d}};
Clear[x, y, t, hitplotter,
  hitpointplotter, pointcolor, actionarrows, matrix2D];
{tlow, thigh} = {0, 2 π};
ranger = 1;
{x[t_], y[t_]} = {Cos[t], Sin[t]};

pointcolor[t_] = RGBColor[0.5 (Cos[t] + 1), 0.5 (Sin[t] + 1), 0];
jump = (thigh - tlow)/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}];

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];
```



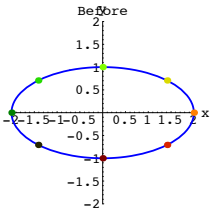


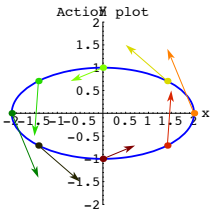


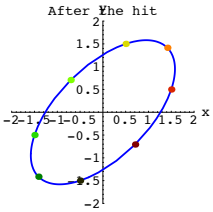
Grab all three plots, align and animate slowly.
To get maximum viewing pleasure and maximum information,
look at one point and follow it.
Then pick another point and follow it.
Keep doing this until you have a good visual grasp of the action.

Watch hits with A rotate this ellipse $\frac{\pi}{4}$ counterclockwise radians about {0,0}:

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

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];
```



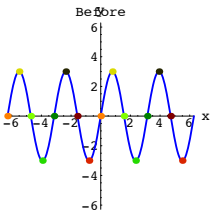


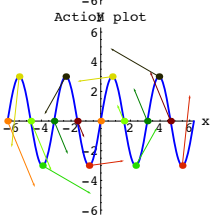


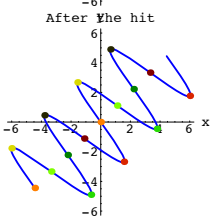
Grab all three plots, align and animate slowly.
To get maximum viewing pleasure and maximum information,
look at one point and follow it.
Then pick another point and follow it.
Keep doing this until you have a good visual grasp of the action.

Watch hits with A rotate this Sine wave $\frac{\pi}{4}$ counterclockwise radians about {0,0}:

```
Clear[x, y, t];
{tlow, thigh} = {-2 π, 2 π};
ranger = 2 π;
{x[t_], y[t_]} = {t, 3 Sin[2 t]};

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.
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.

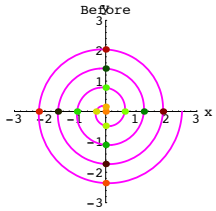
Play.

□T.2.a.ii) Whirling counterclockwise about {0,0}

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

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

ranger = 3;
{x[t_], y[t_]} = t {0.4 Cos[4 t], 0.4 Sin[4 t]};
{tlow, thigh} = {0, 2 π};
```

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

jump = (thigh - tlow)/16;

hitplotter[matrix2D_] := ParametricPlot[matrix2D.{x[t], y[t]},
   {t, tlow, thigh}, PlotStyle → {{Thickness[0.01], Magenta}},
   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}];

before = Show[hitplotter[IdentityMatrix[2]],
   hitpointplotter[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}.

□ Answer:

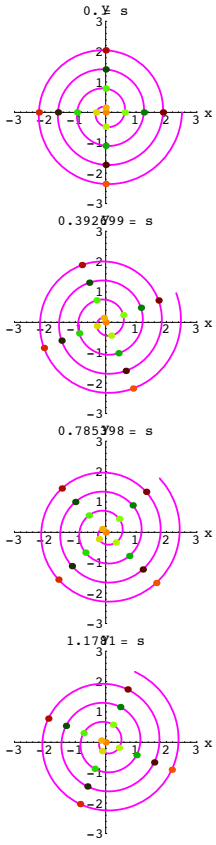
Here's the 2D matrix that rotates s counterclockwise radians about {0,0}:

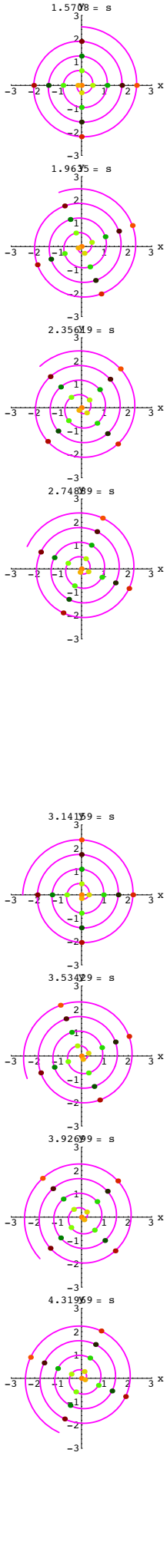
```
Clear[A, a, b, c, d, s];
{perpframe[1], perpframe[2]} =
  N[{{Cos[s], Sin[s]}, {Cos[s + π/2], Sin[s + π/2]}}];

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

A[s_] = {{a, b}, {c, d}};
MatrixForm[A[s]]
```

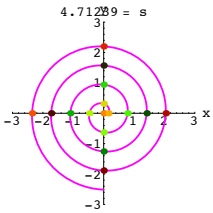
$$\begin{pmatrix} \text{Cos}[s] & -1.\,\text{Sin}[s] \\ \text{Sin}[s] & \text{Cos}[s] \end{pmatrix}$$

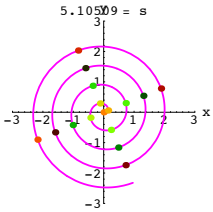
To make the spiral whirl about {0,0}, just do this:

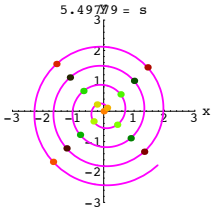
```
jump = π/8;
Table[Show[hitplotter[A[s]],
   hitpointplotter[A[s]], PlotLabel → N[s] "= s",
   DisplayFunction → $DisplayFunction], {s, 0, 2 π - jump, jump}];
```

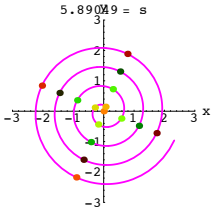


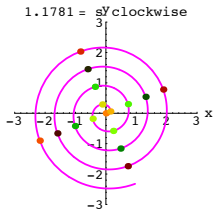


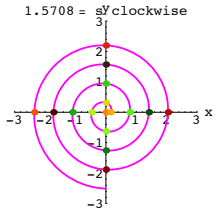


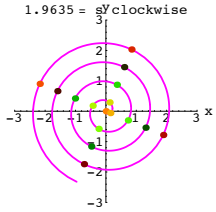


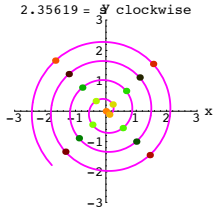












Grab the plots, align and animate, running at various speeds - slow and fast.

Whirl, whirl, whirl.

It might not be a good idea to look at this movie at fast speeds immediately after returning from a long visit to a campus hangout.

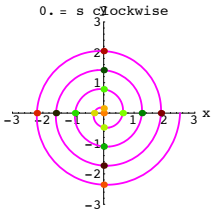
□ T.2.a.iii) Whirling clockwise about {0,0}

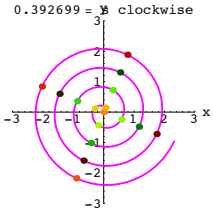
Make a movie showing the spiral in part i) whirl in the counterclockwise way around {0,0}.

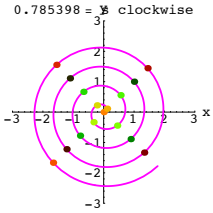
□ Answer:

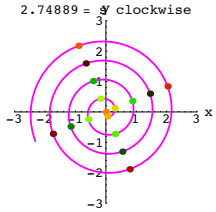
Copy paste, edit and watch that spiral whirl in the clockwise direction:

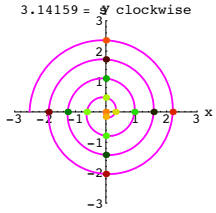
```
Table[Show[hitplotter[A[-s]],
  hitpointplotter[A[-s]], PlotLabel → N[s] "= s clockwise",
  DisplayFunction → $DisplayFunction], {s, 0, 2 π - jump, jump}];
```

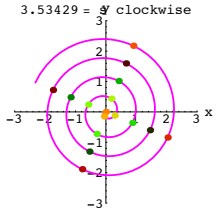


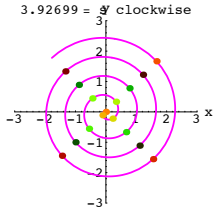


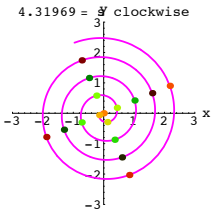


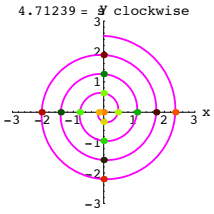


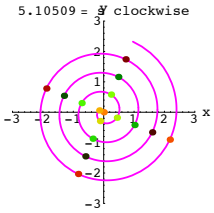


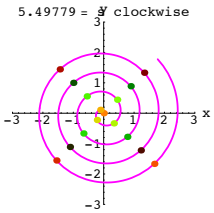


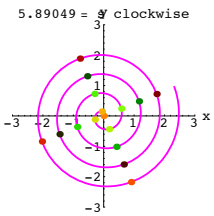












There you go.

□T.2.b.i) Rotation matrices are hanger matrices

Explain this:
Rotations are hanger matrices.

□Answer:

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]}}
```

This matrix rotates by s radians

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

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

Then you need:

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

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

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

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

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

And this tells you that A is the hanger matrix for {perpframe[1],perpframe[2]}

□T.2.b.ii) When you hang on a right hand perpendicular frame, you are rotating.

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

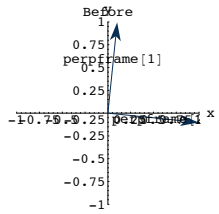
Are hanger matrices corresponding to left hand frames also rotation matrices?

□Answer:

Try it and see:

Go with random left hand perpendicular frame:

```
s = Random[Real, {-Pi/2, Pi/2}];
{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 frame is:

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

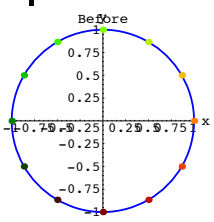
```
hanger = Transpose[{perpframe[1], perpframe[2]}];
MatrixForm[hanger]
```

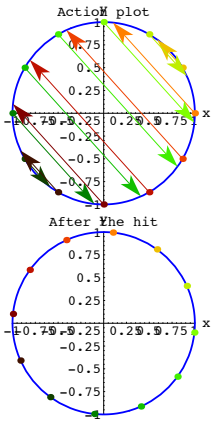
$$\begin{pmatrix} 0.101905 & 0.994794 \\ 0.994794 & -0.101905 \end{pmatrix}$$

Here comes the action movie:

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

```
Clear[x, y, t, hitplotter,
  hitpointplotter, pointcolor, actionarrows, matrix2D];
{tlow, thigh} = {0, 2 π};
ranger = 1;
{x[t_], y[t_]} = {Cos[t], Sin[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], 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}];

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];
```





That's no rotation!

A flip, yes.

A rotation no way.

The moral:

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

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

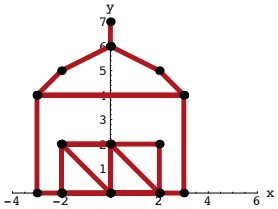
T.3) Making your own 2D matrices: Shears

□ T.3.a) Gil Strang's barn

This graphic is adapted (with permission) from
Gilbert Strang's text book Linear Algebra,
Wellesley-Cambridge Publishing

Look at this:

```
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"}];
```

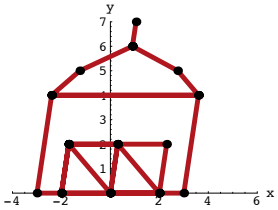


See what happens when you hit this barn with the shear matrix

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

with b = 0.15:

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

Show[hitplotter[A[0.15]], PlotRange → {{-4, 6}, {0, 7}},
 Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"}];
```



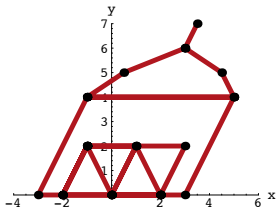
A tilt.

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

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

with b = 0.5:

```
Show[hitplotter[A[0.5]], PlotRange → {{-4, 6}, {0, 7}},
 Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"}];
```



A bigger tilt.

Describe what you see and try to explain why you see it.

□ Answer:

Look at the matrix you are hitting with:

```
Clear[b];
MatrixForm[A[b]]
```

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

Evidently, the bigger positive b you go with, the bigger tilt you get.

To try to explain why this happens, remember {1,0} points in the horizontal direction and look at:

```
A[b].{1, 0}
```

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

This tells you that when you hit horizontal lines with A, you get a horizontal line.

Next, remember {0,1} points in the vertical direction and look at:

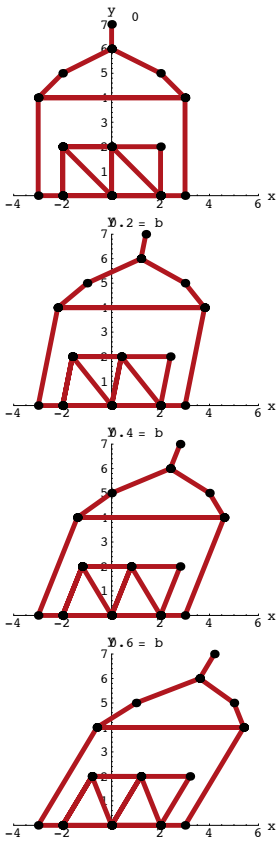
```
A[b].{0, 1}
```

```
{b, 1.}
```

This tells you that when you hit vertical lines with A, you get a line whose slope is 1/b.

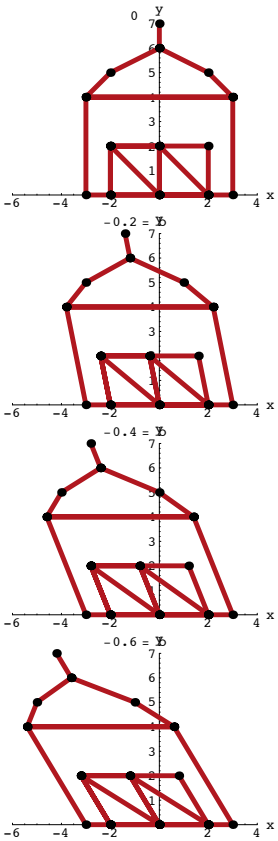
This is why bigger positive b you go with, the bigger tilt you get:

```
Table[Show[hitplotter[A[b]], PlotRange → {{-4, 6}, {0, 7}},
   Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"},
   PlotLabel → b "= b"], {b, 0, 0.6, 0.2}];
```



Knowing that when you hit vertical lines with A, you get a line whose slope is 1/b, you also know what will happen when you go with negative b's:

```
Table[Show[hitplotter[A[b]], PlotRange → {{-6, 4}, {0, 7}},
   Axes → True, AspectRatio → Automatic, AxesLabel → {"x", "y"},
   PlotLabel → b "= b"], {b, 0, -0.6, -0.2}];
```



To complete the explanation, go with a cleared b and look at this:

```
MatrixForm[A[b]]
```

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

And look at what happens when you hit {x,y} with A[b]:

```
A[b].{x, y}
```

```
{1. x + b y, 1. y}
```

This tells you that the y-slot before the hit is the same as the y-slot after the hit.

Consequently:

-> A hit by A[b] does not move horizontal lines up or down.

-> A hit by A[b] does move horizontal lines to the right (for b>0 and y > 0) or to the left (for b < 0 and y > 0)

T.4) Linearity of matrix hits:

A.(r {x,y}) = r A.{x,y} and

A.({x,y} + {u,v}) = A.{x,y} + A.{u,v}.

Consequences:

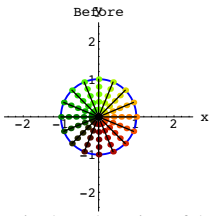
Hitting a line with a matrix gives another line and hitting a parallelogram with a matrix gives another parallelogram

□ T.4.a.i) One aspect of linearity:

A.(r {x,y}) = r A.{x,y}

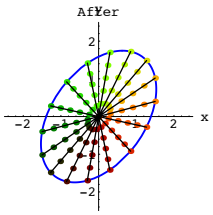
Here's the unit circle chock-full of points:

```
Clear[x, y, t, perpframe, s];
{tlow, thigh} = {0, 2 π};
ranger = 2.5;
{x[t_], y[t_]} = {Cos[t], Sin[t]};

Clear[hitplotter, hitpointplotter,
  pointcolor, hitspokesplotter, matrix2D, r];
pointcolor[t_] = RGBColor[0.5 (Cos[t] + 1), 0.5 (Sin[t] + 1), 0];

tjump = (thigh - tlow)/16;

rjump = 0.2;
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[0.03],
     Point[matrix2D.{r x[t], r y[t]}]}],
   {t, tlow, thigh - tjump, tjump}, {r, 0, 1, rjump}];

hitspokesplotter[matrix2D_] =
  Table[Graphics[Line[{{0, 0}, matrix2D.{x[t], y[t]}}]],
   {t, tlow, thigh - tjump, tjump}];

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



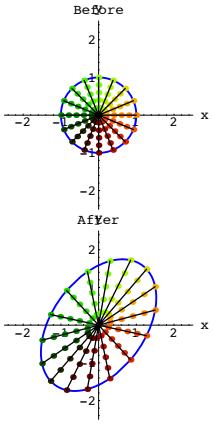
Notice how the points of the same color are lined up on the plotted spokes.
Here's what happens when you hit everything in sight with this matrix:

```
A = {{1.5, 0.3}, {0.4, 1.7}};

after = Show[hitplotter[A], hitpointplotter[A], hitspokesplotter[A],
  PlotLabel → "After", DisplayFunction → $DisplayFunction];
```



Review both plots:

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



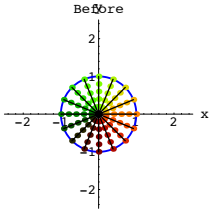
Grab and animate.

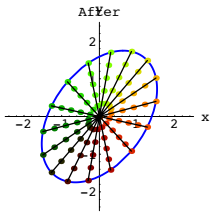
Describe what you see and explain why you see it.

□ Answer:

Take another look:

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



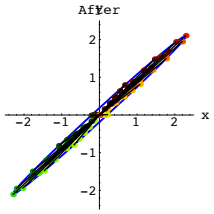


After the hit, points of the same color are still lined up on the plotted spokes.

This outcome has nothing to do with the specific matrix you go with.

In fact if you go with a random matrix, you get:

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


Show[hitplotter[A], hitpointplotter[A], hitspokesplotter[A],
  PlotLabel → "After", DisplayFunction → $DisplayFunction];
"In this graphic, A is" MatrixForm[A]
```



In this graphic, A is $\begin{pmatrix} 1.77254 & -1.46583 \\ 1.49565 & -1.48119 \end{pmatrix}$

Rerun five or more times.

The reason this works for any 2D matrix A has to do with something folks call "linearity of matrix hits."

One aspect of this property is that for any number r

A.(r{ x, y}) = r A.{x,y}

Check it out for a cleared A, r, x, and y:

```
Clear[a, b, c, d, x, y, r];
A = {{a, b}, {c, d}};
linearityproperty = A.(r {x, y}) == Expand[r A.{x, y}]
```

```
True
```

This property explains why the like-colored points are lined up on straight spokes after the hit by a 2D matrix A. Here it goes:

Before the hit, the like-colored points are of the form

r {x[t],y[t]} with r running from 0 to 1.

After these points are hit with A, they become

A. (r {x[t],y[t]}) with r running from 0 to 1.

The linearity property tells you that these are the same as the points

r A.{x[t],y[t]} with r running from 0 to 1.

Theses points all lie on the line from {0,0} to A.{x[t],y[t]}. And that's why points of like color are lined up in a straight line after the hit by A.

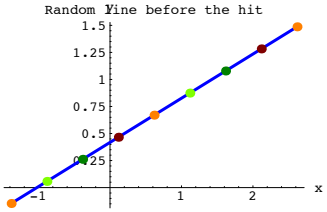
□T.4.a.ii) Another aspect of linearity:

A.({x,y} + {u,v}) = A.{x,y} + A.{u,v}.

When you hit a line with a matrix, you get a line

Here's a random line shown with color-coded points on the line.

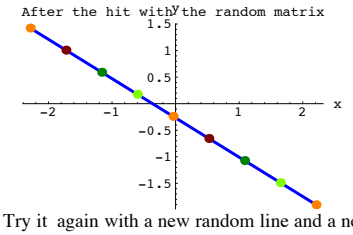
```
Clear[x, y, t, s];
{tlow, thigh} = {-2, 2};
{x[t_], y[t_]} = {Random[Real, {-2, 2}], Random[Real, {-2, 2}]} +
   t {1, Random[Real, {-1, 1}]};

Clear[hitplotter, hitpointplotter,
  pointcolor, actionarrows, matrix2D];
pointcolor[t_] = RGBColor[0.5 (Cos[π t] + 1), 0.5 (Sin[π t] + 1), 0];

tjump = (thigh - tlow)/8;

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

hitpointplotter[matrix2D_] :=
  Table[Graphics[{pointcolor[t], PointSize[0.03],
     Point[matrix2D.{x[t], y[t]}]}], {t, tlow, thigh, tjump}];

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

before = Show[hitplotter[IdentityMatrix[2]],
   hitpointplotter[IdentityMatrix[2]],
   PlotLabel → "Random line before the hit",
   AspectRatio → 1/GoldenRatio, DisplayFunction → $DisplayFunction];
```



Here's what you get when you hit everything on this line with this random 2D matrix:

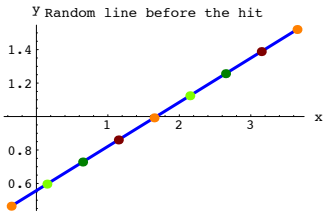
```
A = {{Random[Real, {-2, 2}], Random[Real, {-2, 2}]},
   {Random[Real, {-2, 2}], Random[Real, {-2, 2}]}};
MatrixForm[A]

Show[hitplotter[A], hitpointplotter[A], PlotRange → All,
  PlotLabel → "After the hit with the random matrix",
  AspectRatio → 1/GoldenRatio, DisplayFunction → $DisplayFunction];
```

$\begin{pmatrix} -1.76491 & 1.56805 \\ 1.56724 & -1.801 \end{pmatrix}$

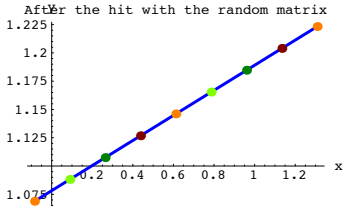


Try it again with a new random line and a new random matrix A:

```
Clear[x, y, t];
    {x[t_], y[t_]} = {Random[Real, {-2, 2}], Random[Real, {-2, 2}]} +
          t {1, Random[Real, {-1, 1}]};
Show[hitplotter[IdentityMatrix[2]],
    hitpointplotter[IdentityMatrix[2]],
    PlotLabel -> "Random line before the hit",
    AspectRatio -> 1 / GoldenRatio,
    DisplayFunction -> $DisplayFunction];

A = {{Random[Real, {-2, 2}], Random[Real, {-2, 2}]},
   {Random[Real, {-2, 2}], Random[Real, {-2, 2}]}};
MatrixForm[A]
Show[hitplotter[A],
    hitpointplotter[A], PlotRange -> All,
    PlotLabel -> "After the hit with the random matrix",
    AspectRatio -> 1 / GoldenRatio,
    DisplayFunction -> $DisplayFunction];
```



$\begin{pmatrix} 0.3301 & 0.0688647 \\ -0.474073 & 1.94188 \end{pmatrix}$



Rerun many times.

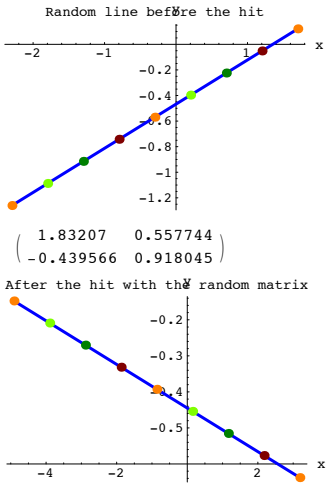
Describe what you see and explain why you see it.

□Answer:

See another:

```
Clear[x, y, t];
    {x[t_], y[t_]} = {Random[Real, {-2, 2}], Random[Real, {-2, 2}]} +
          t {1, Random[Real, {-1, 1}]};
```

```
Show[hitplotter[IdentityMatrix[2]],
    hitpointplotter[IdentityMatrix[2]],
     PlotLabel -> "Random line before the hit",
     AspectRatio -> 1 / GoldenRatio,
    DisplayFunction -> $DisplayFunction];

A = {{Random[Real, {-2, 2}], Random[Real, {-2, 2}]},
   {Random[Real, {-2, 2}], Random[Real, {-2, 2}]}};
MatrixForm[A]
 Show[hitplotter[A],
    hitpointplotter[A], PlotRange -> All,
    PlotLabel -> "After the hit with the random matrix",
    AspectRatio -> 1 / GoldenRatio,
    DisplayFunction -> $DisplayFunction];
```



$$\begin{pmatrix} 1.83207 & 0.557744 \\ -0.439566 & 0.918045 \end{pmatrix}$$

When you hit the random line with the random matrix, you get another line. It works this way every time. The reason this works has to do with another aspect of what folks call "linearity of matrix hits."

One aspect of this property was highlighted in part i) above. The other aspect of the linearity property is that

A.({x,y} + {u,v}) = A.{x,y} + A.{u,v}.

Check it out for a cleared A, x,y , u and v:

```
Clear[a, b, c, d, p, q, u, v];
A = {{a, b}, {c, d}};

linearityproperty =
 Expand[A.({x, y} + {u, v})] == Expand[A.{x, y} + A.{u, v}]
```

```
True
```

This property explains why you get another line when you hit a line with a 2D matrix.

Here's how it goes:

You start with a point {x,y} on the line and a direction vector {u,v} and parameterize the resulting line:

line[t] = {x,y} + t {u,v}

When you hit line[t] with a 2D matrix A, you get

A. line[t] = A.({x,y} + t {u,v})

The linearity property immediately above tells you that this the same as

A.{x,y} + A.(t {u,v}).

And the linearity property in part i) above tells you that this is the same as

A.{x,y} + t A.({u,v}).

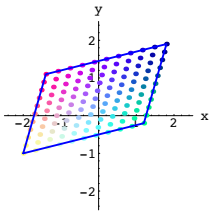
This is a parametrization of the line passing through A.{x,y} with direction vector A.{u,v}.

That's why you get a line when you hit a line with a matrix.

□ T.4.b.i) Parallelograms

Here's a parallelogram:

```
jump = 0.1;
Clear[parallelogramplotter, basepoint, side1, side2, pointcolor];
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], Blue, Line[{basepoint, basepoint + side1,
       basepoint + side1 + side2, basepoint + side2, basepoint}]}]};

basepoint = {-2, -1};
side1 = {3.2, 0.8};
side2 = {0.6, 2.1};

Show[parallelogramplotter[basepoint, side1, side2],
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y"}];
```



What do you get when you hit this parallelogram with a matrix?

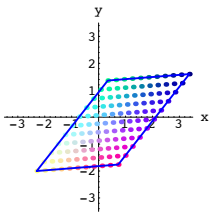
□ Answer:

When you hit a line with a matrix, you get another line.

So,when you hit a parallelogram with a matrix, you get another parallelogram.

See it happen for A = $\begin{pmatrix} 0.5 & 1.3 \\ 1.1 & -0.2 \end{pmatrix}$:

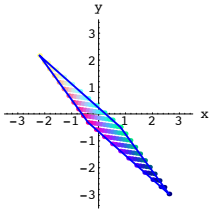
```
A = {{0.5, 1.3}, {1.1, -0.2}};

ranger = 3.5;

Show[parallelogramplotter[A.basepoint, A.side1, A.side2],
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y"}];
```



Hit the original parallelogram with a random matrix:

```
a = 1.2;
A = {{Random[Real, {-a, a}], Random[Real, {-a, a}]},
   {Random[Real, {-a, a}], Random[Real, {-a, a}]}};
MatrixForm[A]

Show[parallelogramplotter[A.basepoint, A.side1, A.side2],
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y"}];
```

$$\begin{pmatrix} 0.77874 & 0.637259 \\ -0.575028 & -1.01926 \end{pmatrix}$$



Rerun five times or more.

The reason these plots turn out the way they do is that when you hit a line with a matrix, you get another line.