Matrices, Geometry&*Mathematica*

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

Producer: Bruce Carpenter

Publisher: Math Everywhere, Inc.

MGM.09 Eigensense: Diagonalizable Matrices, Matrix Exponential, Matrix Powers and Dynamical Systems *BASICS*

B.1) The kD diagonalizable matrices are those whose hits stretch or stretch and flip some basis of kD

☐B.1.a.i) Making a matrix that stretches basis vectors

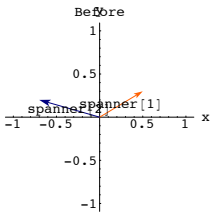
Here's a basis of 2D:

```
Clear[spanner];
spanner[1] = {0.5, 0.3};
spanner[2] = {-0.7, 0.2};
basis = {spanner[1], spanner[2]}
```

```
{{0.5, 0.3}, {-0.7, 0.2}}
```

See them:

```
ranger =
  1.5 Sqrt[Max[{spanner[1].spanner[1], spanner[2].spanner[2]}]];

basisplot =
  {Arrow[spanner[1], Tail -> {0, 0}, VectorColor -> CadmiumOrange],
                    Arrow[spanner[2],
    Tail -> {0, 0}, VectorColor -> NavyBlue]};
basislabels = {Graphics[{Black,
     Text["spanner[1]", 0.5 spanner[1]]}],
                Graphics[{Black,
     Text["spanner[2]", 0.5 spanner[2]]}]};
before = Show[basisplot, basislabels,
                      Axes -> True,
   AxesLabel -> {"x", "y"}, PlotLabel -> "Before",
                      PlotRange ->
    {{-ranger, ranger}, {-ranger, ranger}}];
```



As you can see, the vectors

{spanner[1], spanner[2]}

form a linearly independent set which spans all of 2D.

In short,

{spanner[1], spanner[2]}

is a basis of 2D.

Your job is to make a 2D matrix A so that

A.spanner[1] = 2.0 spanner[1]

and

A.spanner[2] = 1.3 spanner[2]

☐Answer:

Go with the spanner matrix corresponding to this basis:

```
SpannerMatrix = Transpose[basis];
MatrixForm[SpannerMatrix]
```

$$\begin{pmatrix} 0.5 & -0.7 \\ 0.3 & 0.2 \end{pmatrix}$$

```
spanner[1] is the first vertical column of SpannerMatrix
spanner[2] is the second vertical column of SpannerMatrix
```

The goal is to make a 2D matrix A so that

A.spanner[1] = **2.0** spanner[1]

and

A.spanner[2] = **1.3** spanner[2]

Put:

A = SpannerMatrix.$\begin{pmatrix} 2.0 & 0 \\ 0 & 1.3 \end{pmatrix}$.SpannerMatrix^{-1}:

```
A = SpannerMatrix.( {{2.0, 0}, {0, 1.3}} ).Inverse[SpannerMatrix];
MatrixForm[A]
```

$$\begin{pmatrix} 1.52581 & 0.790323 \\ 0.135484 & 1.77419 \end{pmatrix}$$

Check out whether

A.spanner[1] = 2.0 spanner[1]

and

A.spanner[2] = 1.3 spanner[2]:

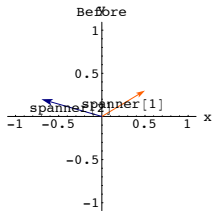
```
A.spanner[1] == 2.0 spanner[1]
A.spanner[2] == 1.3 spanner[2]
```

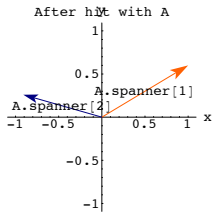
```
True
True
```

This checks.

See it:

```
Show[before];
hitbasisplot =
 {Arrow[A.spanner[1], Tail -> {0, 0}, VectorColor -> CadmiumOrange],
  Arrow[A.spanner[2], Tail -> {0, 0}, VectorColor -> NavyBlue]};
hitbasislabels =
  {Graphics[{Black, Text["A.spanner[1]", 0.5 A. spanner[1]]}],
                Graphics[
    {Black, Text["A.spanner[2]", 0.5 A.spanner[2]]}]};
afer = Show[hitbasisplot, hitbasislabels,
                      Axes -> True, AxesLabel -> {"x", "y"},
                      PlotLabel -> "After hit with A",
                      PlotRange ->
    {{-ranger, ranger}, {-ranger, ranger}}];
```



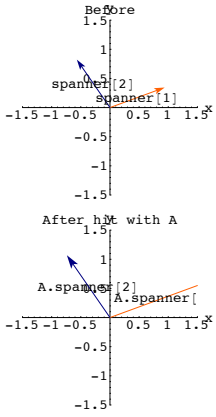


Grab both plots and animate.

Do the same thing for a random basis of 2D:

```
Clear[spanner, t];
spanner[1] = {Cos[t], Sin[t]} /. t -> Random[Real, {-π/4, π/4}];
spanner[2] = {Cos[t], Sin[t]} /. t -> Random[Real, {π/3, 1.5 π}];
basis = {spanner[1], spanner[2]};
SpannerMatrix = Transpose[basis];
A = SpannerMatrix.( {{2.0, 0}, {0, 1.3}} ).Inverse[SpannerMatrix];
ranger =
  1.5 Sqrt[Max[{spanner[1].spanner[1], spanner[2].spanner[2]}]];

basisplot =
  {Arrow[spanner[1], Tail -> {0, 0}, VectorColor -> CadmiumOrange],
                    Arrow[spanner[2],
    Tail -> {0, 0}, VectorColor -> NavyBlue]};
basislabels = {Graphics[{Black,
     Text["spanner[1]", 0.5 spanner[1]]}],
                Graphics[{Black,
     Text["spanner[2]", 0.5 spanner[2]]}]};
before = Show[basisplot, basislabels,
                      Axes -> True,
   AxesLabel -> {"x", "y"}, PlotLabel -> "Before",
                      PlotRange ->
    {{-ranger, ranger}, {-ranger, ranger}}];
hitbasisplot =
 {Arrow[A.spanner[1], Tail -> {0, 0}, VectorColor -> CadmiumOrange],
  Arrow[A.spanner[2], Tail -> {0, 0}, VectorColor -> NavyBlue]};
hitbasislabels =
  {Graphics[{Black, Text["A.spanner[1]", 0.5 A. spanner[1]]}],
                Graphics[
    {Black, Text["A.spanner[2]", 0.5 A.spanner[2]]}]};
afer = Show[hitbasisplot, hitbasislabels,
                      Axes -> True, AxesLabel -> {"x", "y"},
```

```
                    PlotLabel -> "After hit with A",
                    PlotRange ->
    {{-ranger, ranger}, {-ranger, ranger}}];
```



Grab both plots and animate and then rerun a couple of times.

□B.1.a.ii) Why that worked

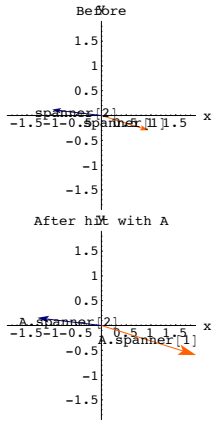
Why did that work?

□Answer:

Do another one:

```
Clear[spanner, t];
spanner[1] = {Cos[t], Sin[t]} /. t -> Random[Real, {-π/4, π/4}];
spanner[2] = {Cos[t], Sin[t]} /. t -> Random[Real, {π/3, 1.5 π}];
basis = {spanner[1], spanner[2]};
SpannerMatrix = Transpose[basis];
A = SpannerMatrix.({{2.0, 0}, {0, 1.3}}).Inverse[SpannerMatrix];
ranger =
  1.9 Sqrt[Max[{spanner[1].spanner[1], spanner[2].spanner[2]}]];

basisplot =
  {Arrow[spanner[1], Tail -> {0, 0}, VectorColor -> CadmiumOrange],
                    Arrow[spanner[2],
    Tail -> {0, 0}, VectorColor -> NavyBlue]};
```

```
basislabels = {Graphics[{Black,
     Text["spanner[1]", 0.5 spanner[1]]}],
                Graphics[{Black,
     Text["spanner[2]", 0.5 spanner[2]]}]};
before = Show[basisplot, basislabels,
                    Axes -> True,
   AxesLabel -> {"x", "y"}, PlotLabel -> "Before",
                    PlotRange ->
    {{-ranger, ranger}, {-ranger, ranger}}];
Clear[hitbasisplot]
hitbasisplot[matrix_] :=
 {Arrow[matrix.spanner[1],
   Tail -> {0, 0}, VectorColor -> CadmiumOrange],
  Arrow[matrix.spanner[2], Tail -> {0, 0}, VectorColor -> NavyBlue]};
hitbasislabels =
  {Graphics[{Black, Text["A.spanner[1]", 0.5 A. spanner[1]]}],
                Graphics[
    {Black, Text["A.spanner[2]", 0.5 A.spanner[2]]}]};
afer = Show[hitbasisplot[A], hitbasislabels,
                    Axes -> True, AxesLabel -> {"x", "y"},
                    PlotLabel -> "After hit with A",
                    PlotRange ->
    {{-ranger, ranger}, {-ranger, ranger}}];
```



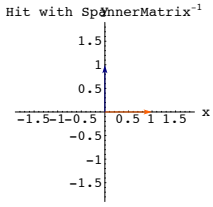
Rerun until you get a nice looking basis of 2D.

You put:

$A = \text{SpannerMatrix}.\begin{pmatrix} 2.0 & 0 \\ 0 & 1.3 \end{pmatrix}.\text{SpannerMatrix}^{-1}$:

See what the hit with $\text{SpannerMatrix}^{-1}$ does to the basis vectors:

```
stage1 = Show[hitbasisplot[Inverse[SpannerMatrix]],
                    Axes -> True,
   AxesLabel -> {"x", "y"}, PlotLabel -> "Hit with SpannerMatrix^-1",
                    PlotRange ->
    {{-ranger, ranger}, {-ranger, ranger}}];
```



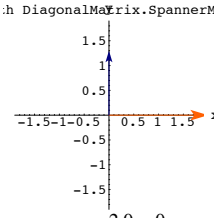
The hit with $\text{SpannerMatrix}^{-1}$ aligns spanner[1] on the x-axis and aligns spanner[2] on the y-axis.

In fact, $\text{SpannerMatrix}^{-1}$.spanner[1] = {1, 0} and $\text{SpannerMatrix}^{-1}$.spanner[2] = {0, 1}.

```
Inverse[SpannerMatrix].spanner[1]
```
```
{1., 0}
```
```
Inverse[SpannerMatrix].spanner[2]
```
```
{0, 1.}
```

See what the hit with $\begin{pmatrix} 2.0 & 0 \\ 0 & 1.3 \end{pmatrix}$. $\text{SpannerMatrix}^{-1}$ does to the basis vectors:

```
stage2 = Show[hitbasisplot[({{2.0, 0}, {0, 1.3}}).Inverse[SpannerMatrix]],
                    Axes -> True, AxesLabel -> {"x", "y"},
  PlotLabel -> "Hit with DiagonalMatrix.SpannerMatrix^-1",
                    PlotRange ->
    {{-ranger, ranger}, {-ranger, ranger}}];
```



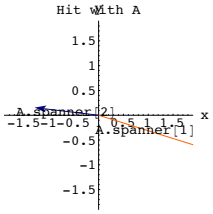
The hit with $\begin{pmatrix} 2.0 & 0 \\ 0 & 1.3 \end{pmatrix}$. $\text{SpannerMatrix}^{-1}$ stretches

by a factor of 2.5 along the x – axis and by a factor of 1.3 along the y – axis.

See what the hit with A =

$\text{SpannerMatrix}.\begin{pmatrix} 2.5 & 0 \\ 0 & 1.3 \end{pmatrix}.\text{SpannerMatrix}^{-1}$ does to the basis vectors

```
stage3 = Show[
  hitbasisplot[SpannerMatrix.({{2.5, 0}, {0, 1.3}}).Inverse[SpannerMatrix]],
          Graphics[
   {Black, Text["A.spanner[1]", 0.5 A. spanner[1]]}],
                Graphics[
   {Black, Text["A.spanner[2]", 0.5 A. spanner[2]]}],
                    Axes -> True, AxesLabel -> {"x", "y"},
  PlotLabel -> "Hit with A",
                    PlotRange ->
    {{-ranger, ranger}, {-ranger, ranger}}];
```

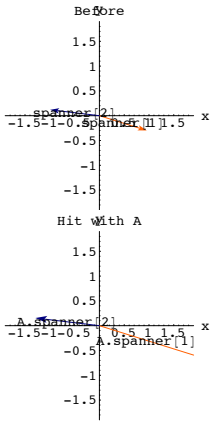


Grab all four plots and animate.

The hit with SpannerMatrix hangs the result on the given basis.

Just for the heck of it, animate these two plots:

```
Show[before];
Show[stage3];
```



A.spanner[1] = 2.0 spanner[1]

and

A.spanner[2] = 1.3 spanner[2]

You can also see why this works through formulas that correspond to the pictures above.

You go with the spanner matrix corresponding to the given basis:

spanner[1] is the first vertical column of SpannerMatrix
spanner[2] is the second vertical column of SpannerMatrix

And you put:

$$A = \text{SpannerMatrix}.\begin{pmatrix} 2.0 & 0 \\ 0 & 1.3 \end{pmatrix}.\text{SpannerMatrix}^{-1}:$$

When you hit A on spanner[1], you get

$$A.\text{spanner}[1] = \text{SpannerMatrix}.\begin{pmatrix} 2.0 & 0 \\ 0 & 1.3 \end{pmatrix}.\text{SpannerMatrix}^{-1}.\text{spanner}[1]$$

$$= \text{SpannerMatrix}.\begin{pmatrix} 2.0 & 0 \\ 0 & 1.3 \end{pmatrix}.\{1, 0\}$$

This corresponds to stage1.

$$= \text{SpannerMatrix}.\{2.0, 0\}$$

This corresponds to stage2.

$$= 2.0\ \text{SpannerMatrix}.\{1, 0\}$$

$$= 2.0\ \text{spanner}[1].$$

This corresponds to the full hit with A.

So you are guaranteed that A.spanner[1] = 2.0 spanner[1].

When you hit A on spanner[2], you get

$$A.\text{spanner}[2] = \text{SpannerMatrix}.\begin{pmatrix} 2.0 & 0 \\ 0 & 1.3 \end{pmatrix}.\text{SpannerMatrix}^{-1} .. \text{spanner}[2]$$

$$= \text{SpannerMatrix}.\begin{pmatrix} 2.0 & 0 \\ 0 & 1.3 \end{pmatrix}.\{0, 1\}$$

This corresponds to stage1.

$$= \text{SpannerMatrix}.\{0, 1.3\}$$

This corresponds to stage2.

$$= 1.3\ \text{SpannerMatrix}.\{0, 1\}$$

$$= 1.3\ \text{spanner}[2]$$

This corresponds to the full hit with A.

So you are guaranteed that

A.spanner[2] = 1.3 spanner[2].

□B.1.a.iii) Relation between aligners and hangers

Those plots hint strongly of the aligners and the hangers in earlier part of the course.
Is this an accident?
If not, what gives?

□Answer:

Get ready.

There are no accidents in mathematics.

The relationship is this:

If your given basis is a perpendicular frame (orthonormal), then

→ SpannerMatrix is the hanger matrix corresponding to the given perpendicular frame and

→ $\text{SpannerMatrix}^{-1}$ is the aligner matrix corresponding to the given perpendicular frame.

Try it out for a random 2D perpendicular frame:

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

basis = {perpframe[1], perpframe[2]}
```

```
{{0.96155, 0.274628}, {-0.274628, 0.96155}}
```

```
SpannerMatrix = Transpose[basis];
aligner = {perpframe[1], perpframe[2]};
hanger = Transpose[{perpframe[1], perpframe[2]}];

SpannerMatrix == hanger
Inverse[SpannerMatrix] == aligner
```

```
True
True
```

□B.1.a.iv) Diagonalizable matrices

What is a diagonalizable matrix?

□Answer:

The matrix made in part i) is diagonalizable.

More generally, a diagonalizable matrix kD is any matrix A you can make by going with a basis

{spanner[1], spanner[2], . . ., spanner[k]}

of kD and going with numbers (positive,or negative or 0)

s[1], s[2], ..., s[k]

and putting

$$A = \text{SpannerMatrix}.\begin{pmatrix} s[1] & 0 & 0 & \bullet\bullet\bullet & 0 \\ 0 & s[2] & 0 & \bullet\bullet\bullet & 0 \\ 0 & 0 & s[3] & \bullet\bullet\bullet & 0 \\ \bullet\bullet\bullet & \bullet\bullet\bullet & \bullet\bullet\bullet & \bullet\bullet\bullet & \bullet\bullet\bullet \\ 0 & 0 & 0 & \bullet\bullet\bullet & s[k] \end{pmatrix}\text{SpannerMatrix}^{-1}$$

Here spanner[j] is the jth vertical column of SpannerMatrix.

In this case you are guaranteed that

A.spanner[j] = s[j] spanner[j]

for all j = 1, 2, . . ., k. So that hits with A stretch or stretch and flip the basis vectors

spanner[1], spanner[2], . . ., spanner[k]

□B.1.a.v) Another sample

Here's a spanning set for a subspace S of 5D:

```
Clear[s, i, spanner];
spanner[1] = {3.43, -0.57, 8.19, -0.94, 0.78};
spanner[2] = {0.97, 1.29, 0.83, 5.23, 0.0};
spanner[3] = {-6.14, 3.08, -5.00, -1.97, -2.45};
spanner[4] = {0.0, 1.07, -2.10, 0.89, 2.51};
spanner[5] = {1.07, 0.0, 3.18, 1.73, 0.0};
spanners = Table[spanner[i], {i, 1, 5}];

ColumnForm[spanners]
```

```
{3.43, -0.57, 8.19, -0.94, 0.78}
{0.97, 1.29, 0.83, 5.23, 0}
{-6.14, 3.08, -5., -1.97, -2.45}
{0, 1.07, -2.1, 0.89, 2.51}
{1.07, 0, 3.18, 1.73, 0}
```

The dimension of S is:

```
SpannerMatrix = Transpose[spanners];
rank = Length[SingularValues[SpannerMatrix][[2]]]
```

```
5
```

S is a five dimensional subspace of 5D; so S is all of 5D:
And the spanning set consists of five vectors.
This is enough to tell you that

{spanner[1], spanner[2], . . ., spanner[5]}

is a basis for all of 5D.
Knowing this much, go ahead and make a 5D matrix A so that

A.spanner[1] = 2.5 spanner[1]
A.spanner[2] = 1.4 spanner[2]
A.spanner[3] = −2.0 spanner[3]
A.spanner[4] = −1.7 spanner[4]
A.spanner[5] = 0.0 spanner[5]

□Answer:

Full speed ahead.

You want

$$\begin{aligned} \text{A.spanner[1]} &= 2.5\ \text{spanner[1]} \\ \text{A.spanner[2]} &= 1.4\ \text{spanner[2]} \\ \text{A.spanner[3]} &= -2.0\ \text{spanner[3]} \\ \text{A.spanner[4]} &= -1.7\ \text{spanner[4]} \\ \text{A.spanner[5]} &= 0.0\ \text{spanner[5]} \end{aligned}$$

Put:

```
diagonalmatrix = DiagonalMatrix[{2.5, 1.4, -2.0, -1.7, 0.0}];
MatrixForm[diagonalmatrix]
```

$$\begin{pmatrix} 2.5 & 0 & 0 & 0 & 0 \\ 0 & 1.4 & 0 & 0 & 0 \\ 0 & 0 & -2. & 0 & 0 \\ 0 & 0 & 0 & -1.7 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Now put:

```
basis = spanners;
SpannerMatrix = Transpose[basis];
MatrixForm[SpannerMatrix]
```

$$\begin{pmatrix} 3.43 & 0.97 & -6.14 & 0 & 1.07 \\ -0.57 & 1.29 & 3.08 & 1.07 & 0 \\ 8.19 & 0.83 & -5. & -2.1 & 3.18 \\ -0.94 & 5.23 & -1.97 & 0.89 & 1.73 \\ 0.78 & 0 & -2.45 & 2.51 & 0 \end{pmatrix}$$

The matrix A you are after is:

```
A = SpannerMatrix.diagonalmatrix.Inverse[SpannerMatrix];
MatrixForm[A]
```

$$\begin{pmatrix} 3.53074 & 7.73748 & 0.0714008 & -2.31499 & -2.41785 \\ 1.50554 & -0.797645 & -0.710947 & 0.375654 & -1.11269 \\ 8.16 & 13.8174 & -0.206954 & -4.66653 & -2.98648 \\ 3.48489 & 3.77452 & -1.1778 & 0.00957989 & -3.20065 \\ 1.40694 & 1.9467 & -0.0768598 & -0.728906 & -2.33572 \end{pmatrix}$$

Check whether

$$\text{A.spanner[1]} = 2.5\ \text{spanner[1]}:$$

```
A.spanner[1]
```

```
{8.575, -1.425, 20.475, -2.35, 1.95}
```

```
2.5 spanner[1]
```

```
{8.575, -1.425, 20.475, -2.35, 1.95}
```

Check whether

$$\text{A.spanner[2]} = 1.4\ \text{spanner[2]}:$$

```
A.spanner[2]
```

```
{1.358, 1.806, 1.162, 7.322, 0}
```

```
1.4 spanner[2]
```

```
{1.358, 1.806, 1.162, 7.322, 0}
```

Check whether

$$\text{A.spanner[3]} = -2.0\ \text{spanner[3]}:$$

```
A.spanner[3]
```

```
{12.28, -6.16, 10., 3.94, 4.9}
```

```
-2.0 spanner[3]
```

```
{12.28, -6.16, 10., 3.94, 4.9}
```

Check whether

$$\text{A.spanner[4]} = -1.7\ \text{spanner[4]}:$$

```
A.spanner[4]
```

```
{0, -1.819, 3.57, -1.513, -4.267}
```

```
-1.7 spanner[4]
```

```
{0, -1.819, 3.57, -1.513, -4.267}
```

Check whether

$$\text{A.spanner[5]} = 0.0\ \text{spanner[5]}:$$

```
A.spanner[5]
```

```
{0, 0, 0, 0, 0}
```

```
0.0 spanner[5]
```

```
{0, 0, 0, 0, 0}
```

Was there ever a doubt?

□B.1.a.iv)

In the last problem (part iii immediately above) there was a big deal made about whether the given spanning set was a basis for all of 5D. Why was this such a big deal?

□Answer:

You put:

$$\text{A} = \text{SpannerMatrix}.\begin{pmatrix} 2.5 & 0 & 0 & 0 & 0 \\ 0 & 1.4 & 0 & 0 & 0 \\ 0 & 0 & -2. & 0 & 0 \\ 0 & 0 & 0 & -1.7 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.\text{SpannerMatrix}^{-1}.$$

The columns of the SpannerMatrix are the 5D vectors

{spanner[1], spanner[2], . . ., spanner[5]}.

In order to be able to be sure that SpannerMatrix is invertible,

→ You need the SpannerMatrix to hit on 5D (So that SpannerMatrix is a square matrix).This means you need five spanners.

→ You need the SpannerMatrix to be of full rank. This means the columns of SpannerMatrix must be linearly independent.

The result: You need

{spanner[1], spanner[2], . . ., spanner[5]}.

to be a basis of 5D.

B.2) Given a kD matrix A, a kD vector X is an eigenvector of A if A.X and X point in the same or opposite direction.

Using eigenvectors to find a basis that is stretched or stretched and flipped by a hit with a matrix.

Any kD diagonalizable matrix looks like this:

$$\begin{pmatrix} \text{eigenvect[1]} & \text{eigenvect[2]} & \cdots & \text{eigenvect[k]} \\ \downarrow & \downarrow & \cdots & \downarrow \\ \cdots & \cdots & \cdots & \cdots \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix}\begin{pmatrix} \text{eigenval[1]} & 0 & \cdots & 0 \\ 0 & \text{eigenval[2]} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \text{eigenval[k]} \end{pmatrix}.\begin{pmatrix} \text{eigenvect[1]} & \text{eigenvect[2]} & \cdots & \text{eigenvect[k]} \\ \downarrow & \downarrow & \cdots & \downarrow \\ \cdots & \cdots & \cdots & \cdots \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix}^{-1}$$

Using eigenvectors to recognize non-diagonalizable matrices.

The beginning of this may be old hat but the end isn't.

□B.2.a.i) Eigenvalues and eigenvectors

Here is a 2D matrix A:

```
A = {{2.1, 0.6}, {-0.1, 1.4}};
MatrixForm[A]
```

$$\begin{pmatrix} 2.1 & 0.6 \\ -0.1 & 1.4 \end{pmatrix}$$

Let *Mathematica* calculate the unit eigenvectors of A:

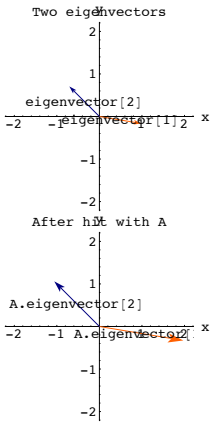
If you don't know what eigenvectors mean, don't worry. By the end of this problem you will know.

```
Clear[eigenvalue, eigenvector];
{eigenvector[1], eigenvector[2]} = Eigenvectors[A]
```

```
{{0.986394, -0.164399}, {-0.707107, 0.707107}}
```

Look at these plots which show the calculated eigenvectors of A and what you get when you hit these eigenvectors with A:

```
ranger = Max[SingularValues[A][[2]]];

eigenplot =
  {Arrow[eigenvector[1], Tail → {0, 0}, VectorColor → CadmiumOrange],
   Arrow[eigenvector[2], Tail → {0, 0}, VectorColor → NavyBlue]};

labels =
  {Graphics[{Black, Text["eigenvector[1]", 0.5 eigenvector[1]]}],
   Graphics[{Black, Text["eigenvector[2]", 0.5 eigenvector[2]]}]};

Show[eigenplot, labels, Axes → True, AxesLabel → {"x", "y"},
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  PlotLabel → "Two eigenvectors"];

hiteigenplot = {Arrow[A.eigenvector[1],
    Tail → {0, 0}, VectorColor → CadmiumOrange],
   Arrow[A.eigenvector[2], Tail → {0, 0}, VectorColor → NavyBlue]};

hitlabels = {Graphics[{Black,
     Text["A.eigenvector[1]", 0.5 A.eigenvector[1]]}], Graphics[
    {Black, Text["A.eigenvector[2]", 0.5 A.eigenvector[2]]}]};

Show[hiteigenplot, hitlabels, Axes → True, AxesLabel → {"x", "y"},
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  PlotLabel → "After hit with A"];
```



Grab both plots and animate.

Look familiar?
Try another:

```
A = {{1.7, 0.4}, {-0.2, -1.3}};
MatrixForm[A]
```

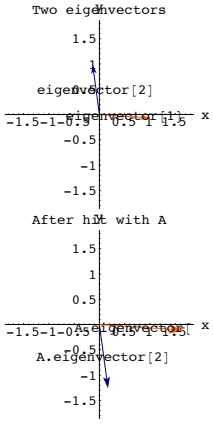
$$\begin{pmatrix} 1.7 & 0.4 \\ -0.2 & -1.3 \end{pmatrix}$$

```
Clear[eigenvalue, eigenvector];
{eigenvector[1], eigenvector[2]} = Eigenvectors[A]
```

{{0.997745, -0.0671183}, {-0.133339, 0.991071}}

```
ranger = Max[SingularValues[A][[2]]];

eigenplot =
  {Arrow[eigenvector[1], Tail → {0, 0}, VectorColor → CadmiumOrange],
   Arrow[eigenvector[2], Tail → {0, 0}, VectorColor → NavyBlue]};

labels =
  {Graphics[{Black, Text["eigenvector[1]", 0.5 eigenvector[1]]}],
   Graphics[{Black, Text["eigenvector[2]", 0.5 eigenvector[2]]}]};

Show[eigenplot, labels, Axes → True, AxesLabel → {"x", "y"},
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  PlotLabel → "Two eigenvectors"];
```

```
hiteigenplot = {Arrow[A.eigenvector[1],
    Tail → {0, 0}, VectorColor → CadmiumOrange],
   Arrow[A.eigenvector[2], Tail → {0, 0}, VectorColor → NavyBlue]};

hitlabels = {Graphics[{Black,
     Text["A.eigenvector[1]", 0.5 A.eigenvector[1]]}], Graphics[
    {Black, Text["A.eigenvector[2]", 0.5 A.eigenvector[2]]}]};

Show[hiteigenplot, hitlabels, Axes → True, AxesLabel → {"x", "y"},
  PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
  PlotLabel → "After hit with A"];
```



Grab both plots and animate.

What do these graphics tell you about eigenvectors of a matrix A?

□ Answer:

Folks all across the planet say that vector X is an eigenvector of a matrix A if

A.X is a multiple of X.

In other words X is an eigenvector of A if X and A.X point in the same or opposite directions.

This is the same as saying that a vector X is an eigenvector of a matrix A if there is a number s so that

$$A.X = s\,X.$$

In this case, the same folks say that s is the eigenvalue of A associated with the eigenvector X .

Try it out for the matrix A immediately above:

```
Clear[eigenvalue, eigenvector];
{eigenvector[1], eigenvector[2]} = Eigenvectors[A]
```

{{0.997745, -0.0671183}, {-0.133339, 0.991071}}

```
{eigenvalue[1], eigenvalue[2]} = Eigenvalues[A]
```

{1.67309, -1.27309}

```
A.eigenvector[1]
```

{1.66932, -0.112295}

```
eigenvalue[1] eigenvector[1]
```

{1.66932, -0.112295}

```
A.eigenvector[2]
```

{0.169752, -1.26172}

```
eigenvalue[2] eigenvector[2]
```

{0.169752, -1.26172}

□ B.2.a.ii) Using eigenvectors and eigenvalues to recognize diagonalizable matrices

Go with this 3D matrix A:

```
A = {{3., -0.7, 1.3}, {-0.3, 2.1, 1.8}, {0.9, 1.6, -1.8}};
MatrixForm[A]
```

$$\begin{pmatrix} 3. & -0.7 & 1.3 \\ -0.3 & 2.1 & 1.8 \\ 0.9 & 1.6 & -1.8 \end{pmatrix}$$

Calculate the eigenvectors of A:

```
Clear[eigenvector];
spanners = Eigenvectors[A]
```

{{-0.98118, -0.0405488, -0.188788},
 {0.408017, 0.833298, 0.37301}, {0.249924, 0.354922, -0.900871}}

Calculate the dimension of the subspace of 3D spanned by the eigenvectors of A:

```
SpannerMatrix = Transpose[spanners];
eigendim = Length[SingularValues[SpannerMatrix][[2]]]
```

3

This tells you that the eigenvectors of A form a basis for all of 3D and that the spanner matrix is invertible.
Explain why this signals that A is diagonalizable.

□ Answer:

To see that A is diagonalizable, you have to be able to duplicate A with a matrix of this form:

$$\text{SpannerMatrix}.\text{diagonalmatrix}.\text{SpannerMatrix}^{-1}$$

The SpannerMatrix you need is already available:

```
Clear[eigenvector];
spanners = Eigenvectors[A];
SpannerMatrix = Transpose[spanners];
MatrixForm[SpannerMatrix]
```

$$\begin{pmatrix} -0.98118 & 0.408017 & 0.249924 \\ -0.0405488 & 0.833298 & 0.354922 \\ -0.188788 & 0.37301 & -0.900871 \end{pmatrix}$$

The diagonal matrix comes from the eigenvalues of A:

```
diagonalmatrix = DiagonalMatrix[Eigenvalues[A]];
MatrixForm[diagonalmatrix]
```

$$\begin{pmatrix} 3.2212 & 0 & 0 \\ 0 & 2.75884 & 0 \\ 0 & 0 & -2.68005 \end{pmatrix}$$

Here's a look at

$$\text{SpannerMatrix}.\text{diagonalmatrix}.\text{SpannerMatrix}^{-1}:$$

```
MatrixForm[SpannerMatrix.diagonalmatrix.Inverse[SpannerMatrix]]
```

$$\begin{pmatrix} 3. & -0.7 & 1.3 \\ -0.3 & 2.1 & 1.8 \\ 0.9 & 1.6 & -1.8 \end{pmatrix}$$

Here's a look at A:

```
MatrixForm[A]
```

$$\begin{pmatrix} 3. & -0.7 & 1.3 \\ -0.3 & 2.1 & 1.8 \\ 0.9 & 1.6 & -1.8 \end{pmatrix}$$

The upshot:

$A = \text{SpannerMatrix}.\text{diagonalmatrix}.\text{SpannerMatrix}^{-1}$:

So A is diagonalizable.

□B.2.a.iii) Explanation: If you have a matrix A that hits on kD and hangs in kD so that the unit eigenvectors of A form a basis (linearly independent spanning set) for all of kD, then A is diagonalizable.

In fact Any kD diagonalizable matrix A looks like this:

$$\begin{pmatrix} \text{eigenvect}[1] & \text{eigenvect}[2] & \cdots & \text{eigenvect}[k] \\ \downarrow & \downarrow & \cdots & \downarrow \\ \cdots & \cdots & \cdots & \cdots \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix} \cdot \begin{pmatrix} \text{eigenval}[1] & 0 & \cdots & 0 \\ 0 & \text{eigenval}[2] & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \text{eigenval}[k] \end{pmatrix} \cdot \begin{pmatrix} \text{eigenvect}[1] & \text{eigenvect}[2] & \cdots & \text{eigenvect}[k] \\ \downarrow & \downarrow & \cdots & \downarrow \\ \cdots & \cdots & \cdots & \cdots \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix}^{-1}$$

Why did that work and why will it work in any dimension?
Specifically:
Go with a matrix A that hits on kD and hangs in kD so that the unit eigenvectors of A form a basis (linearly independent spanning set) for all of kD.
Explain why A is guaranteed to be diagonalizable.

□Answer:

Go with a matrix A that hits on kD and hangs in kD so that the unit eigenvectors of A form a basis (linearly independent spanning set) for all of kD.

```
This means you need k of them and that they must be linearly independent.
```

Arrange the eigenvectors into the SpannerMatrix with eigenvector[j] in the jth vertical column of SpannerMatrix.

$$\text{SpannerMatrix} = \cdot \begin{pmatrix} \text{eigenvector}[1] & \text{eigenvector}[2] & \text{eigenvector}[3] & \cdots & \text{eigenvector}[k] \\ \downarrow & \downarrow & \downarrow & \cdots & \downarrow \\ \downarrow & \downarrow & \downarrow & \cdots & \downarrow \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \downarrow & \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix}$$

And then you take the corresponding eigenvalues of A and make

$$\text{diagonalmatrix} = \text{DiagonalMatrix}[\{\text{eigenvalue}[1], \text{eigenvalue}[2], \ldots, \text{eigenvalue}[k]\}]$$

$$= \cdot \begin{pmatrix} \text{eigenvalue}[1] & 0 & 0 & \cdots & 0 \\ 0 & \text{eigenvalue}[2] & 0 & \cdots & 0 \\ 0 & 0 & \text{eigenvalue}[3] & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \text{eigenvalue}[k] \end{pmatrix}$$

You put

$B = \text{SpannerMatrix}.\text{diagonalmatrix}.\text{SpannerMatrix}^{-1}$. so that

$$B = \begin{pmatrix} \text{eigenvect}[1] & \text{eigenvect}[2] & \cdots & \text{eigenvect}[k] \\ \downarrow & \downarrow & \cdots & \downarrow \\ \cdots & \cdots & \cdots & \cdots \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix} \cdot \begin{pmatrix} \text{eigenval}[1] & 0 & \cdots & 0 \\ 0 & \text{eigenval}[2] & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \text{eigenval}[k] \end{pmatrix} \cdot \begin{pmatrix} \text{eigenvect}[1] & \text{eigenvect}[2] & \cdots & \text{eigenvect}[k] \\ \downarrow & \downarrow & \cdots & \downarrow \\ \cdots & \cdots & \cdots & \cdots \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix}^{-1}$$

If you were around in B.1) you know that this results in

$$B.\text{eigenvector}[j] = \text{eigenvalue}[j]\ \ \text{eigenvector}[j]$$

for all j = 1, 2, ..., k.

But by the very nature of eigenvalues and eigenvectors of A, you know

$$A.\text{eigenvector}[j] = \text{eigenvalue}[j]\ \ \text{eigenvector}[j]$$

for all j = 1, 2, ..., k.

And because

{eigenvector[1], eigenvector[2], . . ., eigenvector[k]}

is a basis for all of kD, this tells you that

$$A.X = B.X$$

for all X in kD. And this is enough to tell you that

$$B = A.$$

And because

$B = \text{SpannerMatrix}.\text{diagonalmatrix}.\text{SpannerMatrix}^{-1}$,

you get

$A = \text{SpannerMatrix}.\text{diagonalmatrix}.\text{SpannerMatrix}^{-1}$.

So A is guaranteed to be diagonalizable.

□B.2.a.iv) Using eigenvectors and eigenvalues to recognize matrices that are not diagonalizable

Go with this 2D matrix A:

```
A = {{1.0, 1.0}, {0.0, 1.0}};
MatrixForm[A]
```

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

Calculate the eigenvectors of A:

```
Clear[eigenvector];
spanners = Eigenvectors[A]
```

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

Calculate the dimension of the subspace of 2D spanned by the eigenvectors of A:

```
SpannerMatrix = Transpose[spanners];
eigendim = Length[SingularValues[SpannerMatrix][[2]]]
```

```
1
```

This tells you that the eigenvectors of A do NOT form a basis for all of 2D.
Explain why this signals that A is not diagonalizable.

□Answer:

For a matrix A to diagonalizable, you have to be able to duplicate A with a matrix of this form:

$$A = \begin{pmatrix} \text{eigenvector}[1] & \text{eigenvector}[2] \\ \downarrow & \downarrow \end{pmatrix} \cdot \begin{pmatrix} \text{eigenvalue}[1] & 0 \\ 0 & \text{eigenvalue}[2] \end{pmatrix} \cdot \begin{pmatrix} \text{eigenvector}[1] & \text{eigenvector}[2] \\ \downarrow & \downarrow \end{pmatrix}^{-1}$$

The trouble here is that $\begin{pmatrix} \text{eigenvector}[1] & \text{eigenvector}[2] \\ \downarrow & \downarrow \end{pmatrix}$ is not invertible because its columns are not linearly independent.

B.3) Functions of diagonalizable matrices.

If

A =

$$\begin{pmatrix} \text{eigenvect}[1] & \text{eigenvect}[2] & \cdots & \text{eigenvect}[k] \\ \downarrow & \downarrow & \cdots & \downarrow \\ \cdots & \cdots & \cdots & \cdots \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix} \cdot \begin{pmatrix} \text{eigenval}[1] & 0 & \cdots & 0 \\ 0 & \text{eigenval}[2] & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \text{eigenval}[k] \end{pmatrix} \cdot \begin{pmatrix} \text{eigenvect}[1] & \text{eigenvect}[2] & \cdots & \text{eigenvect}[k] \\ \downarrow & \downarrow & \cdots & \downarrow \\ \cdots & \cdots & \cdots & \cdots \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix}^{-1},$$

then

f[A] =

$$\begin{pmatrix} \text{eigenvect}[1] & \text{eigenvect}[2] & \cdots & \text{eigenvect}[k] \\ \downarrow & \downarrow & \cdots & \downarrow \\ \cdots & \cdots & \cdots & \cdots \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix} \cdot \begin{pmatrix} f[\text{eigenval}[1]] & 0 & \cdots & 0 \\ 0 & f[\text{eigenval}[2]] & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & f[\text{eigenval}[k]] \end{pmatrix} \cdot \begin{pmatrix} \text{eigenvect}[1] & \text{eigenvect}[2] & \cdots & \text{eigenvect}[k] \\ \downarrow & \downarrow & \cdots & \downarrow \\ \cdots & \cdots & \cdots & \cdots \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix}^{-1}.$$

The matrix exponential E^{At} for a diagonalizable matrix A.

The derivative with respect to t of E^{At} is $A.E^{At}$

□B.3.a.i) Hitting a diagonalizable matrix with a function f[x] .

How do you apply a function f[x] such as

$f[x] = x^2$
$f[x] = x^6$
$f[x] = 1/x$
$f[x] = \text{Sin}[x]$

to a diagonalizable matrix?

□Answer:

Here's a 3D matrix A:

```
A = {{1.40, 0.66, -1.10}, {0.57, 0.68, -3.06}, {-0.81, -3.08, -2.53}};
MatrixForm[A]
```

$$\begin{pmatrix} 1.4 & 0.66 & -1.1 \\ 0.57 & 0.68 & -3.06 \\ -0.81 & -3.08 & -2.53 \end{pmatrix}$$

Check whether A is diagonalizable:

```
spanners = Eigenvectors[A];
SpannerMatrix = Transpose[spanners];
eigendim = Length[SingularValues[SpannerMatrix][[2]]]
```

Good. A has three linearly independent eigenvectors and this signals that A is diagonalizable.

The diagonal matrix comes from the eigenvalues of A:

```
diagonalmatrix = DiagonalMatrix[Eigenvalues[A]];
MatrixForm[diagonalmatrix]
```

$$\begin{pmatrix} -4.4311 & 0 & 0 \\ 0 & 3.14122 & 0 \\ 0 & 0 & 0.83988 \end{pmatrix}$$

Here's a look at

SpannerMatrix.diagonalmatrix.SpannerMatrix^{-1}:

```
MatrixForm[SpannerMatrix.diagonalmatrix.Inverse[SpannerMatrix]]
```

$$\begin{pmatrix} 1.4 & 0.66 & -1.1 \\ 0.57 & 0.68 & -3.06 \\ -0.81 & -3.08 & -2.53 \end{pmatrix}$$

Here's a look at A:

```
MatrixForm[A]
```

$$\begin{pmatrix} 1.4 & 0.66 & -1.1 \\ 0.57 & 0.68 & -3.06 \\ -0.81 & -3.08 & -2.53 \end{pmatrix}$$

Good. A is diagonalizable.

To calculate A, you multiply out

$$\text{SpannerMatrix}.\begin{pmatrix} -4.45083 & 0 & 0 \\ 0 & 3.09764 & 0 \\ 0 & 0 & 0.823183 \end{pmatrix}.\text{SpannerMatrix}^{-1}:$$

To calculate f[A], you multiply out

$$\text{SpannerMatrix}.\begin{pmatrix} f[-4.45083] & 0 & 0 \\ 0 & f[3.09764] & 0 \\ 0 & 0 & f[0.823183] \end{pmatrix}.\text{SpannerMatrix}^{-1}:$$

That's all there is to it.

The advantage of working with diagonalizable matrices is:

You can apply functions to diagonalizable matrices with almost no work.

Try it out by calculating A.A this way:

```
Clear[f, x];
f[x_] = x^2;
fdiagonalmatrix = DiagonalMatrix[Map[f, Eigenvalues[A]]];
fA = SpannerMatrix.fdiagonalmatrix.Inverse[SpannerMatrix];
MatrixForm[fA]
```

$$\begin{pmatrix} 3.2272 & 4.7608 & -0.7766 \\ 3.6642 & 10.2634 & 5.034 \\ -0.8403 & 5.1634 & 16.7167 \end{pmatrix}$$

Compare:

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

$$\begin{pmatrix} 3.2272 & 4.7608 & -0.7766 \\ 3.6642 & 10.2634 & 5.034 \\ -0.8403 & 5.1634 & 16.7167 \end{pmatrix}$$

Nailed it.

Try it out by calculating A.A.A.A.A.A this way:

```
Clear[f, x];
f[x_] = x^6;
fdiagonalmatrix = DiagonalMatrix[Map[f, Eigenvalues[A]]];
fA = SpannerMatrix.fdiagonalmatrix.Inverse[SpannerMatrix];
MatrixForm[fA]
```

$$\begin{pmatrix} 305.534 & 797.534 & 422.7 \\ 579.279 & 2428.72 & 2936.68 \\ 245.561 & 2979.49 & 5796.33 \end{pmatrix}$$

Compare:

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

$$\begin{pmatrix} 305.534 & 797.534 & 422.7 \\ 579.279 & 2428.72 & 2936.68 \\ 245.561 & 2979.49 & 5796.33 \end{pmatrix}$$

Nailed it again.

Try it out by calculating A^{-1} this way:

```
Clear[f, x];
f[x_] = 1/x;
fdiagonalmatrix = DiagonalMatrix[Map[f, Eigenvalues[A]]];
fA = SpannerMatrix.fdiagonalmatrix.Inverse[SpannerMatrix];
MatrixForm[fA]
```

$$\begin{pmatrix} 0.953368 & -0.432648 & 0.108774 \\ -0.33538 & 0.379202 & -0.312822 \\ 0.103059 & -0.323122 & -0.0492544 \end{pmatrix}$$

Compare:

```
MatrixForm[Inverse[A]]
```

$$\begin{pmatrix} 0.953368 & -0.432648 & 0.108774 \\ -0.33538 & 0.379202 & -0.312822 \\ 0.103059 & -0.323122 & -0.0492544 \end{pmatrix}$$

It looks like you're onto something.

Hell, you can even calculate the Sine of A:

```
Clear[f, x];
f[x_] = Sin[x];
fdiagonalmatrix = DiagonalMatrix[Map[f, Eigenvalues[A]]];
fA = SpannerMatrix.fdiagonalmatrix.Inverse[SpannerMatrix];
MatrixForm[fA]
```

$$\begin{pmatrix} 0.552551 & -0.292975 & 0.22137 \\ -0.23634 & 0.416116 & 0.347538 \\ 0.173019 & 0.346211 & 0.736968 \end{pmatrix}$$

Nothing can stop you.

□B.3.a.ii) The matrix exponential E^{At}

Here's a 2D matrix A:

```
A = {{2.0, -0.3}, {-0.5, 1.7}};
MatrixForm[A]
```

$$\begin{pmatrix} 2. & -0.3 \\ -0.5 & 1.7 \end{pmatrix}$$

The diagonal matrix comes from the eigenvalues of A:

```
diagonalmatrix = DiagonalMatrix[Eigenvalues[A]];
MatrixForm[diagonalmatrix]
```

$$\begin{pmatrix} 2.26533 & 0 \\ 0 & 1.43467 \end{pmatrix}$$

Here's a look at

SpannerMatrix.diagonalmatrix.SpannerMatrix^{-1}:

```
spanners = Eigenvectors[A];
SpannerMatrix = Transpose[spanners];
MatrixForm[SpannerMatrix.diagonalmatrix.Inverse[SpannerMatrix]]
```

$$\begin{pmatrix} 2. & -0.3 \\ -0.5 & 1.7 \end{pmatrix}$$

Compare:

```
MatrixForm[A]
```

$$\begin{pmatrix} 2. & -0.3 \\ -0.5 & 1.7 \end{pmatrix}$$

Good. A is diagonalizable.
Use what you see to calculate
E^{At} and Sin[At].

□Answer:

To calculate A, you multiply out

$$\text{SpannerMatrix}.\begin{pmatrix} 2.26533 & 0 \\ 0 & 1.43467 \end{pmatrix}.\text{SpannerMatrix}^{-1}:$$

To calculate E^{At}, you multiply out

$$\text{SpannerMatrix}.\begin{pmatrix} E^{2.26533\,t} & 0 \\ 0 & E^{1.43467\,t} \end{pmatrix}.\text{SpannerMatrix}^{-1}$$

Here's some code:

```
Clear[f, x, t];
f[x_] = E^(x t);
fdiagonalmatrix = DiagonalMatrix[Map[f, Eigenvalues[A]]];
fA = SpannerMatrix.fdiagonalmatrix.Inverse[SpannerMatrix];
MatrixForm[fA]
```

$$\begin{pmatrix} 0.319421\, e^{1.43467\,t} + 0.680579\, e^{2.26533\,t} & 0.361158\, e^{1.43467\,t} - 0.361158\, e^{2.26533\,t} \\ 0.601929\, e^{1.43467\,t} - 0.601929\, e^{2.26533\,t} & 0.680579\, e^{1.43467\,t} + 0.319421\, e^{2.26533\,t} \end{pmatrix}$$

Check with *Mathematica*:

```
MatrixForm[MatrixExp[A t]]
```

$$\begin{pmatrix} 0.319421\, e^{1.43467\, t} + 0.680579\, e^{2.26533\, t} & 0.361158\, e^{1.43467\, t} - 0.361158\, e^{2.26533\, t} \\ 0.601929\, e^{1.43467\, t} - 0.601929\, e^{2.26533\, t} & 0.680579\, e^{1.43467\, t} + 0.319421\, e^{2.26533\, t} \end{pmatrix}$$

Nailed it.

Calculate Sin[A t] this way:

```
Clear[f, x];
f[x_] = Sin[x t];
fdiagonalmatrix = DiagonalMatrix[Map[f, Eigenvalues[A]]];
fA = SpannerMatrix.fdiagonalmatrix.Inverse[SpannerMatrix];
MatrixForm[fA]
```

$$\begin{pmatrix} 0.319421\, \text{Sin}[1.43467\, t] + 0.680579\, \text{Sin}[2.26533\, t] & 0.361158\, \text{Sin}[1.43467\, t] \\ 0.601929\, \text{Sin}[1.43467\, t] - 0.601929\, \text{Sin}[2.26533\, t] & 0.680579\, \text{Sin}[1.43467\, t] \end{pmatrix}$$

Mathematica has no built in function that calculates Sin[A t]; so you can't check whether this one is correct.

You can take this on faith or you can go onto the next part to see why this procedure is guaranteed to work for you.:

□B.3.a.iii) Why it works

Why is the procedure used above guaranteed to work on all diagonalizable matrices?

□Answer:

Here's the idea in 2D: The same idea (with more writing) works in all dimensions.

A 2D diagonalizable matrix A is of the form

$$A = \text{SpannerMatrix}.\begin{pmatrix} \text{eigenvalue}[1] & 0 \\ 0 & \text{eigenvalue}[2] \end{pmatrix}.\text{SpannerMatrix}^{-1}$$

Just to keep everything from bouncing off the screen, use this shorthand:

SM = SpannerMatrix

e[1] = eigenvalue[1]

e[2] = eigenvalue[2].

In this shorthand,a 2D diagonalizable matrix A is of the form

$$A = \text{SM}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\text{SM}^{-1}.$$

This gives:

$$A.A = \text{SM}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\text{SM}^{-1}.\text{SM}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\text{SM}^{-1}$$

$$= \text{SM}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\text{SM}^{-1}.$$

Reason: $\text{SM}^{-1}.\text{SM} = \text{IdentityMatrix}$, the matrix whose hits do nothing.

So

$$A.A = \text{SM}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\text{SM}^{-1}$$

$$= \text{SM}.\begin{pmatrix} e[1]^2 & 0 \\ 0 & e[2]^2 \end{pmatrix}.\text{SM}^{-1}$$

Reason: $\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix} = \begin{pmatrix} e[1]^2 & 0 \\ 0 & e[2]^2 \end{pmatrix}$.

This explains why when you go with $f[x] = x^2$, then you are guaranteed that

$$A.A = \text{SM}.\begin{pmatrix} f[e[1]] & 0 \\ 0 & f[e[2]] \end{pmatrix}.\text{SM}^{-1}.$$

To see why when you go with $f[x] = x^3$, then you are guaranteed that

$$A.A.A = \text{SM}.\begin{pmatrix} f[e[1]] & 0 \\ 0 & f[e[2]] \end{pmatrix}.\text{SM}^{-1},$$

calculate

$$A.A.A = A.(A.A) = \text{SM}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\text{SM}^{-1}\,\text{SM}.\begin{pmatrix} e[1]^2 & 0 \\ 0 & e[2]^2 \end{pmatrix}.\text{SM}^{-1}$$

$$= \text{SM}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\begin{pmatrix} e[1]^2 & 0 \\ 0 & e[2]^2 \end{pmatrix}.\text{SM}^{-1}$$

$$= \text{SM}.\begin{pmatrix} e[1]^3 & 0 \\ 0 & e[2]^3 \end{pmatrix}.\text{SM}^{-1}$$

$$= \text{SM}.\begin{pmatrix} f[e[1]] & 0 \\ 0 & f[e[2]] \end{pmatrix}.\text{SM}^{-1}$$

The same ideas (but more writing) explain why the method performs as it should for all power functions $f[x] = x^k$. And because other functions normally used in science and engineering such as

E^x, Sin[x], Cos[x], $\frac{1}{x}$, etc.

can be built from combinations of power functions (expansions in powers of x, Taylor series, etc.), the method is guaranteed to work for these functions as well.

□B.3.a.iv) The derivative with respect to t of $E^{(A\,t)}$ is $A.E^{(A\,t)}$.

Given a diagonalizable matrix A, the derivative with respect to t of

$E^{(A\,t)}$

is

$A.E^{(A\,t)}$.

Explain this juicy fact.

□Answer:

Here's the idea in 2D: The same idea (with more writing) works in all dimensions.

A 2D diagonalizable matrix A is of the form

$$A = \text{SpannerMatrix}.\begin{pmatrix} \text{eigenvalue}[1] & 0 \\ 0 & \text{eigenvalue}[2] \end{pmatrix}.\text{SpannerMatrix}^{-1}$$

Just to keep everything from bouncing off the screen, use this shorthand:

SM = SpannerMatrix

e[1] = eigenvalue[1]

e[2] = eigenvalue[2].

In this shorthand, a 2D diagonalizable matrix A is of the form

$$A = \text{SM}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\text{SM}^{-1}.$$

This gives:

$$A\,t = t\,\text{SM}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\text{SM}^{-1} = \text{SM}.\begin{pmatrix} t\,e[1] & 0 \\ 0 & t\,e[2] \end{pmatrix}.\text{SM}^{-1}$$

So.

$$E^{A\,t} = \text{SM}.\begin{pmatrix} E^{t\,e[1]} & 0 \\ 0 & E^{t\,e[2]} \end{pmatrix}.\text{SM}^{-1}.$$

Differentiate with respect to t to get

$$D[E^{A\,t}, t] = \text{SM}.\begin{pmatrix} D[E^{t\,e[1]}, t] & 0 \\ 0 & D[E^{t\,e[2]}, t] \end{pmatrix}.\text{SM}^{-1}$$

$$= \text{SM}.\begin{pmatrix} e[1]\,E^{t\,e[1]} & 0 \\ 0 & e[2]\,E^{t\,e[2]} \end{pmatrix}.\text{SM}^{-1}$$

$$= \text{SM}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\begin{pmatrix} E^{t\,e[1]} & 0 \\ 0 & E^{t\,e[2]} \end{pmatrix}.\text{SM}^{-1}$$

$$= \text{SM}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.(\text{SM})^{-1}.\text{SM}.\begin{pmatrix} E^{t\,e[1]} & 0 \\ 0 & E^{t\,e[2]} \end{pmatrix}\text{SM}^{-1}$$

$$= \text{SM}.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.\text{SM}^{-1}.\text{SM}.\begin{pmatrix} E^{t\,e[1]} & 0 \\ 0 & E^{t\,e[2]} \end{pmatrix}\text{SM}^{-1}$$

$$= A.E^{A\,t}$$

$$A = SM.\begin{pmatrix} e[1] & 0 \\ 0 & e[2] \end{pmatrix}.SM^{-1}$$

Reasons:

and

$$E^{At} = .SM.\begin{pmatrix} E^{te[1]} & 0 \\ 0 & E^{te[2]} \end{pmatrix}SM^{-1}$$

And you're out of here.

B.4) The *Mathematica* instruction MatrixExp[A] for calculating E^A quickly without going to the trouble of calculating eigenvalues and eigenvectors.

How you can calculate E^A via series in the case that A is not diagonalizable

□B.4.a) Using MatrixExp[A] to calculating E^A without going to the trouble of calculating eigenvalues and eigenvectors

Here's a random 2D matrix A:

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

$$\begin{pmatrix} 0.0535191 & 0.303907 \\ 0.523795 & 1.76773 \end{pmatrix}$$

Now look at this:

```
Clear[B, t];
B = MatrixExp[A t];
MatrixForm[B]
```

$$\begin{pmatrix} 0.953295\, e^{-0.0347929\, t} + 0.0467052\, e^{1.85605\, t} & -0.160726\, e^{-0.0347929\, t} + 0.160726\, e \\ -0.277017\, e^{-0.0347929\, t} + 0.277017\, e^{1.85605\, t} & 0.0467052\, e^{-0.0347929\, t} + 0.953295\, e \end{pmatrix}$$

What is this new matrix?

□Answer:

It is E^{At}.

If you feed a matrix A t into the *Mathematica* instruction MatrixExp[], *Mathematica* tries to calculate E^{At}.

This is all to your advantage because you can use this instruction to calculate E^{At} without going to the trouble of calculating the eigenvalues and eigenvectors of A.

□B.4.a.ii) Sometimes MatrixExp[ ] balks

Does the *Mathematica* MatrixExp instruction ever fail?

□Answer:

Not usually, but you can get strange output when you enter the matrix using exact fractions or integers. See this happen:

```
A = {{1   -1},
     {3/2  -3/4}};
MatrixForm[MatrixExp[A]]
```

$$\begin{pmatrix} \frac{i\,(7-i\sqrt{47})\,e^{\frac{1}{8}-\frac{i\sqrt{47}}{8}t}}{2\sqrt{47}} - \frac{i\,(7+i\sqrt{47})\,e^{\frac{1}{8}+\frac{i\sqrt{47}}{8}t}}{2\sqrt{47}} & -\frac{4\,i\,e^{\frac{1}{8}-\frac{i\sqrt{47}}{8}t}}{\sqrt{47}} + \frac{4\,i\,e^{\frac{1}{8}+\frac{i\sqrt{47}}{8}t}}{\sqrt{47}} \\ \frac{6\,i\,e^{\frac{1}{8}-\frac{i\sqrt{47}}{8}t}}{\sqrt{47}} - \frac{6\,i\,e^{\frac{1}{8}+\frac{i\sqrt{47}}{8}t}}{\sqrt{47}} & \frac{(-7\,i+\sqrt{47})\,e^{\frac{1}{8}-\frac{i\sqrt{47}}{8}t}}{2\sqrt{47}} + \frac{(7\,i+\sqrt{47})\,e^{\frac{1}{8}+\frac{i\sqrt{47}}{8}t}}{2\sqrt{47}} \end{pmatrix}$$

Yuck. Looks bad and feels worse.

You will get agreeable results when you enter the matrix through decimal entries like this:

```
decimalA = N[{{1   -1},
              {3/2  -3/4}}];
MatrixForm[decimalA]
```

$$\begin{pmatrix} 1. & -1. \\ 1.5 & -0.75 \end{pmatrix}$$

Now look at the calculation of e^A:

```
MatrixForm[MatrixExp[decimalA]]
```

$$\begin{pmatrix} 1.61645 & -0.99946 \\ 1.49919 & -0.132609 \end{pmatrix}$$

Sweet and nice.

□B.4.b.i) In the rare situations in which A is not diagonalizable, you can use a series

If A is not diagonalizable, how can you do an approximate calculation of E^A?

□Answer:

You can perturb the matrix A a bit or you can go to series approximations.

It's not a lot of fun, but it can be done.

Take a non-diagonalizable A:

```
A = {{0.5, 0.5}, {-0.5, 1.5}};
MatrixForm[A]
```

$$\begin{pmatrix} 0.5 & 0.5 \\ -0.5 & 1.5 \end{pmatrix}$$

This matrix is not diagonalizable because its eigenvectors are not linearly independent:

```
Eigenvectors[A]
```

```
{{-0.707107, -0.707107}, {-0.707107, -0.707107}}
```

To try to calculate E^A, start with a series approximation of E^x:

```
khigh = 5;
Clear[x, k];
    Eseries = Normal[Series[E^x, {x, 0, khigh}]]
```

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120}$$

This is the same as:

```
1 + Sum[x^k/k!, {k, 1, khigh}]
```

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120}$$

To approximate E^A, change 1 to the identity matrix and change each x^k to MatrixPower[A,k]:

```
approx = IdentityMatrix[2] + Sum[MatrixPower[A, k]/k!, {k, 1, khigh}];
MatrixForm[approx]
```

$$\begin{pmatrix} 1.3625 & 1.35417 \\ -1.35417 & 4.07083 \end{pmatrix}$$

Compare:

```
MatrixForm[MatrixExp[A]]
```

$$\begin{pmatrix} 1.35914 & 1.35914 \\ -1.35914 & 4.07742 \end{pmatrix}$$

Not too bad. If you want a better approximation, just raise khigh:

```
khigh = 25;
approx = IdentityMatrix[2] + Sum[MatrixPower[A, k]/k!, {k, 1, khigh}];
MatrixForm[approx]
```

$$\begin{pmatrix} 1.35914 & 1.35914 \\ -1.35914 & 4.07742 \end{pmatrix}$$

Compare:

```
MatrixForm[MatrixExp[A]]
```

$$\begin{pmatrix} 1.35914 & 1.35914 \\ -1.35914 & 4.07742 \end{pmatrix}$$

□B.4.b.ii) The advantage you get when you work with a diagonalizable matrix

What advantage do you get when you work with a diagonalizable matrix ?

□Answer:

You don't have to fool around with perturbations or with series approximations.

B.5) Real eigenvalues of A reveal how $E^{At}.X$ plots out as t gets large

Eigenvalues and eigenvectors of A reveal the ultimate direction of $E^{At}.X$ as t gets large

□B.5.a.i) If A has two negative eigenvalues, then $E^{At}.X \longrightarrow \{0,0\}$ as t gets large

Here's a 2D matrix A:

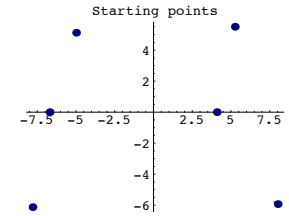
```
A = {{-1.76  1.11},
     {0.90  -0.75}};
MatrixForm[A]
```

$$\begin{pmatrix} -1.76 & 1.11 \\ 0.9 & -0.75 \end{pmatrix}$$

Here are six random points
{starter[1], starter[2], starter[3], starter[4], starter[5], starter[6]}
in 2D :

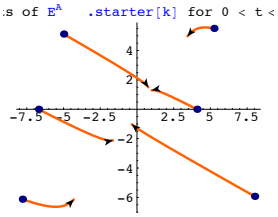
```
Clear[pointer, trajectoryplot, eigenplot,
  scaler, eigenvector, starter, thigh, k, j];

trajectoryplot[thigh_, starter_] :=
  ParametricPlot[MatrixExp[A t].starter, {t, 0, thigh},
   PlotStyle -> {{Thickness[0.01], CadmiumOrange}},
   DisplayFunction -> Identity];


pointer[thigh_, starter_] :=
  ArrowHead[MatrixExp[A thigh] .starter,
   (D[MatrixExp[A t], t] /. t -> thigh) .starter,
              HeadSize -> 0.6, VectorColor -> Black,
   Aperture -> 0.4];

starterplots = Table[Graphics[
    {NavyBlue, PointSize[0.03], Point[starter[j]]}], {j, 1, 6}];
```

```
starter[1] = {Random[Real, {4, 10}], Random[Real, {4, 10}]};
starter[2] = {Random[Real, {-10, -4}], Random[Real, {-10, -4}]};
starter[3] = {Random[Real, {-10, -4}], Random[Real, {4, 10}]};
starter[4] = {Random[Real, {4, 10}], Random[Real, {-10, -4}]};
starter[5] = {Random[Real, {-10, -4}], 0};
starter[6] = {Random[Real, {4, 10}], 0};

thigh = 0.01;
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
  starterplots, PlotLabel -> "Starting points",
  DisplayFunction -> $DisplayFunction];
```



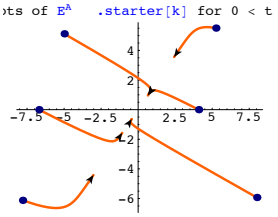
Here is how the curves
E^{At}.starter[1], E^{At}.starter[2], E^{At}.starter[3],
E^{At}.starter[4], E^{At}.starter[5], E^{At}.starter[6]
plot out as t advances from 0 to 1.5

```
thigh = 1.5;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



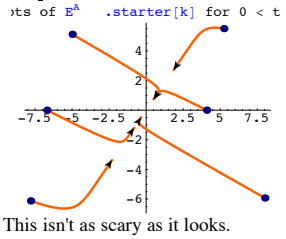
Here's what happens as t advances from 0 to 4:

```
thigh = 4;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



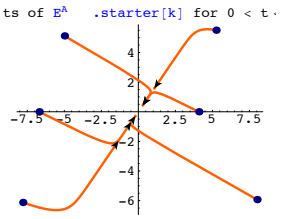
And here's what happens as t advances from 0 to 6:

```
thigh = 6;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



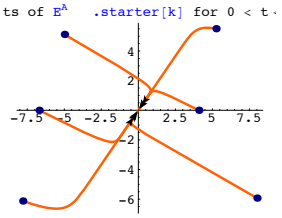
This isn't as scary as it looks.
See what happens as t advances from 0 to 10:

```
thigh = 10;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



Here's what happens as t advances from 0 to 20:

```
thigh = 20;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



Grab the plots, align and animate.

Eventually all the plots head to {0,0} and stall at {0,0}.
Now look at the eigenvalues of A:

```
Eigenvalues[A]
```

{-2.37483, -0.135167}

Both negative.
Explain this statement:
If A is any diagonalizable matrix and both its eigenvalues are **negative**, then no matter what X is, you are guaranteed that the plot of
E^{At}.X
heads to {0,0} and stalls at {0,0}.

□ **Answer:**

Arrange the eigenvectors of A into the SpannerMatrix with eigenvect[1]in the leftmost vertical column and eigenvect[2] in the second vertical column of SpannerMatrix.

So that

$$A = \text{SpannerMatrix}.\begin{pmatrix}\text{eigval}[1] & 0\\ 0 & \text{eigval}[2]\end{pmatrix}.\text{SpannerMatrix}^{-1}.$$

This gives

$$E^{At} = \text{SpannerMatrix}.\begin{pmatrix}E^{\text{eigval}[1]t} & 0\\ 0 & E^{\text{eigval}[2]t}\end{pmatrix}.\text{SpannerMatrix}^{-1}$$

And

$$E^{At}.X = \text{SpannerMatrix}.\begin{pmatrix}E^{\text{eigval}[1]t} & 0\\ 0 & E^{\text{eigval}[2]t}\end{pmatrix}.\text{SpannerMatrix}^{-1}.X.$$

Because eigenval[1] < 0 and eigenval[2] < 0, the functions involved ($E^{\text{eigval}[1]t}$ and $E^{\text{eigval}[2]t}$) both go to 0 as t gets large. The result:

$$E^{At}.X \text{ --> } \text{SpannerMatrix}.\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}.\text{SpannerMatrix}^{-1}.X = \{0, 0\}$$

as t gets large.

□ **B.5.a.ii) If A has two positive eigenvalues, then $\| E^{At}.X \| \longrightarrow \infty$ as t gets large**

Here's a new 2D matrix A:

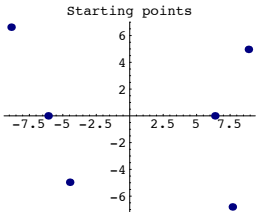
```
A = ( 0.84  -0.10
     -0.20   0.40 );
MatrixForm[A]
```

$$\begin{pmatrix}0.84 & -0.1\\ -0.2 & 0.4\end{pmatrix}$$

Here are six random points
{starter[1], starter[2], starter[3], starter[4], starter[5], starter[6]}
in 2D :

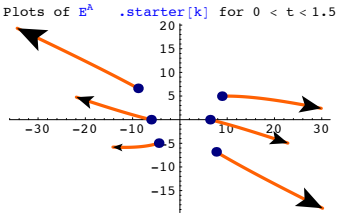
```
Clear[pointer, trajectoryplot, eigenplot,
  scaler, eigenvector, starter, thigh, k, j];

trajectoryplot[thigh_, starter_] :=
  ParametricPlot[MatrixExp[A t].starter, {t, 0, thigh},
   PlotStyle -> {{Thickness[0.01], CadmiumOrange}},
   DisplayFunction -> Identity];


pointer[thigh_, starter_] :=
  ArrowHead[MatrixExp[A thigh] .starter,
   (D[MatrixExp[A t], t] /. t -> thigh) .starter,
              VectorColor -> Black, Aperture -> 0.4];

starterplots = Table[Graphics[
    {NavyBlue, PointSize[0.03], Point[starter[j]]}], {j, 1, 6}];
```

```
starter[1] = {Random[Real, {4, 10}], Random[Real, {4, 10}]};
starter[2] = {Random[Real, {-10, -4}], Random[Real, {-10, -4}]};
starter[3] = {Random[Real, {-10, -4}], Random[Real, {4, 10}]};
starter[4] = {Random[Real, {4, 10}], Random[Real, {-10, -4}]};
starter[5] = {Random[Real, {-10, -4}], 0};
starter[6] = {Random[Real, {4, 10}], 0};

thigh = 0.01;
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
  starterplots, PlotLabel -> "Starting points",
  DisplayFunction -> $DisplayFunction];
```



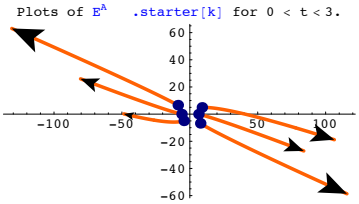
Here is how the curves
E^{At}.starter[1], E^{At}.starter[2], E^{At}.starter[3],
E^{At}.starter[4], E^{At}.starter[5], E^{At}.starter[6]
plot out as t advances from 0 to 1.5:

```
thigh = 1.5;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



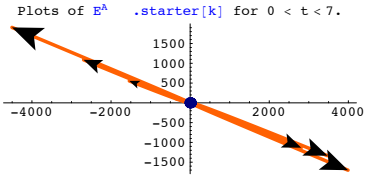
Here's what happens as t advances from 0 to 3.0:

```
thigh = 3.0;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



And here's what happens as t advances from 0 to 7.0:

```
thigh = 7.0;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



Eventually all the plots head away from {0,0} and try to run off the screen.
Now look at the eigenvalues of A:

```
Eigenvalues[A]
```

```
{0.881534, 0.358466}
```

Both positive.
Explain this statement:
If A is any diagonalizable matrix and both its eigenvalues are positive, then as long as $X \neq \{0,0\}$, you are guaranteed that, as t gets large,

$$\| E^{At}.X \| \longrightarrow \infty.$$

□ **Answer:**

Arrange the eigenvectors of A into the SpannerMatrix with eigenvect[1]in the leftmost vertical column and eigenvect[2] in the second vertical column of SpannerMatrix.

So that

$$A = \text{SpannerMatrix}.\begin{pmatrix} \text{eigval[1]} & 0 \\ 0 & \text{eigval[2]} \end{pmatrix}.\text{SpannerMatrix}^{-1}.$$

This gives

$$E^{At} = \text{SpannerMatrix}.\begin{pmatrix} E^{\text{eigval[1]}\,t} & 0 \\ 0 & E^{\text{eigval[2]}\,t} \end{pmatrix}.\text{SpannerMatrix}^{-1}$$

And

$$E^{At}.X = \text{SpannerMatrix}.\begin{pmatrix} E^{\text{eigval[1]}\,t} & 0 \\ 0 & E^{\text{eigval[2]}\,t} \end{pmatrix}.\text{SpannerMatrix}^{-1}.X.$$

Because eigenval[1] > 0 and eigenval[2] > 0, the functions involved ($E^{\text{eigval[1]}\,t}$ and $E^{\text{eigval[2]}\,t}$) both become huge as t gets large. The result:
Unless X = {0,0},

$$\| E^{At}.X \| \longrightarrow \infty$$

as t gets large.

□ **B.5.b.i) Eigenvalues and eigenvectors of A reveal the ultimate direction of $E^{At}.X$ as t gets large**

Here's a 2D matrix A:

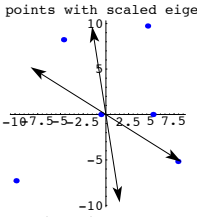
```
A = ( 1.8   0.1
     -0.4   1.1 );
      MatrixForm[A]
```

$$\begin{pmatrix} 1.8 & 0.1 \\ -0.4 & 1.1 \end{pmatrix}$$

Here are six random points
{starter[1], starter[2], starter[3], starter[4], starter[5], starter[6]}
in 2D shown with scaled versions of the eigenvectors of A:

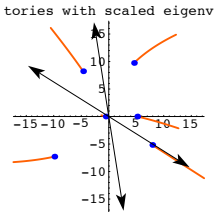
```
Clear[trajectoryplot, eigenplot,
  scaler, eigenvector, starter, thigh, k, j];
```

```
trajectoryplot[thigh_, starter_] :=
  ParametricPlot[MatrixExp[A t].starter, {t, 0, thigh},
   PlotStyle -> {{Thickness[0.01], CadmiumOrange}},
   DisplayFunction -> Identity];
starterplots = Table[Graphics[
    {Blue, PointSize[0.03], Point[starter[j]]}], {j, 1, 6}];

scaler[thigh_] :=
    0.9 Sqrt[
   (MatrixExp[A thigh].starter[1]).(MatrixExp[A thigh].starter[1])]

{eigenvector[1], eigenvector[2]} = Eigenvectors[A];
eigenplot[khigh_, j_] :=
    {Arrow[scaler[khigh] eigenvector[j],
    Tail -> {0, 0}, VectorColor -> Black],
     Arrow[-scaler[khigh] eigenvector[j],
    Tail -> {0, 0}, VectorColor -> Black]};

starter[1] = {Random[Real, {4, 10}], Random[Real, {4, 10}]};
starter[2] = {Random[Real, {-10, -4}], Random[Real, {-10, -4}]};
starter[3] = {Random[Real, {-10, -4}], Random[Real, {4, 10}]};
starter[4] = {Random[Real, {4, 10}], Random[Real, {-10, -4}]};
starter[5] = {Random[Real, {-10, 0}], 0};
starter[6] = {Random[Real, {0, 10}], 0};

thigh = 0.01;
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
  starterplots, eigenplot[thigh, 1], eigenplot[thigh, 2],
  PlotLabel -> "Starting points with scaled eigenvectors",
  DisplayFunction -> $DisplayFunction];
```



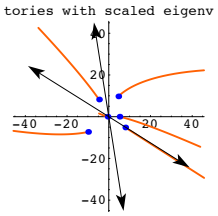
Here's how the curves
E^{At}.starter[1], E^{At}.starter[2], E^{At}.starter[3],
E^{At}.starter[4], E^{At}.starter[5], E^{At}.starter[6]
plot out as t advances from 0 to 0.5 shown with scaled eigenvectors of A:

```
thigh = 0.5;
ranger = scaler[thigh];
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
  starterplots, eigenplot[thigh, 1], eigenplot[thigh, 2],
  PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
  PlotLabel -> "Trajectories with scaled eigenvectors",
  DisplayFunction -> $DisplayFunction];
```



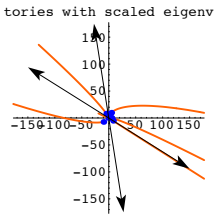
Here's what happens as t advances from 0 to 1.2:

```
thigh = 1.2;
ranger = scaler[thigh];
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
  starterplots, eigenplot[thigh, 1], eigenplot[thigh, 2],
  PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
  PlotLabel -> "Trajectories with scaled eigenvectors",
  DisplayFunction -> $DisplayFunction];
```



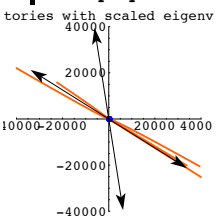
Here's what happens as t advances from 0 to 2.0:

```
thigh = 2.0;
ranger = scaler[thigh];
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
  starterplots, eigenplot[thigh, 1], eigenplot[thigh, 2],
  PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
  PlotLabel -> "Trajectories with scaled eigenvectors",
  DisplayFunction -> $DisplayFunction];
```



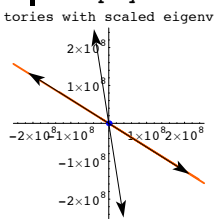
Here's what happens as t advances from 0 to 5.0:

```
thigh = 5.0;
ranger = scaler[thigh];
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
  starterplots, eigenplot[thigh, 1], eigenplot[thigh, 2],
  PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
  PlotLabel -> "Trajectories with scaled eigenvectors",
  DisplayFunction -> $DisplayFunction];
```



Here's what happens as t advances from 0 to 10.0:

```
thigh = 10.0;
ranger = scaler[thigh];
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
  starterplots, eigenplot[thigh, 1], eigenplot[thigh, 2],
  PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
  PlotLabel -> "Trajectories with scaled eigenvectors",
  DisplayFunction -> $DisplayFunction];
```



As t gets large, the curves

E^(A t).starter[j]

seem to shy away from one of the eigenvectors and to gravitate to the line through {0,0} defined by the ultimate eigenvector.
How do you identify the ultimate eigenvector?

□ Answer:

Look at the eigenvectors of A:

```
Clear[eigenvector];
{eigenvector[1], eigenvector[2]} = Eigenvectors[A]
```

```
{{0.846962, -0.531654}, {-0.155032, 0.987909}}
```

Look at the eigenvalues:

```
Clear[eigenvalue];
{eigenvalue[1], eigenvalue[2]} = Eigenvalues[A]
```

```
{1.73723, 1.16277}
```

Note that

eigenvalue[1] > eigenvalue[2].

This signals that eigenvector[1] is ultimate.

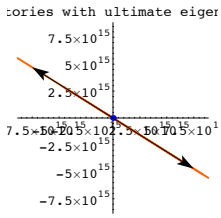
This means that as t gets large the plots of

$E^{A\,t}$.starter[j]

are eventually attracted to the line through {0, 0} defined by eigenvector[1] .

See it happen in this plot which displays only the line through {0, 0} defined by eigenvector[1].

```
thigh = 20.0;
ranger = scaler[thigh];
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
 starterplots, eigenplot[thigh, 1],
 PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
 PlotLabel -> "Trajectories with ultimate eigenvector",
 DisplayFunction -> $DisplayFunction];
```

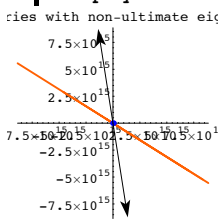


As t gets large, the plots of

$E^{A\,t}$.starter[j]

just laugh at the line through {0,0} defined by eigenvector[2]:

```
thigh = 20.0;
ranger = scaler[thigh];
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
 starterplots, eigenplot[thigh, 2],
 PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
 PlotLabel -> "Trajectories with non-ultimate eigenvector",
 DisplayFunction -> $DisplayFunction];
```



This happens because

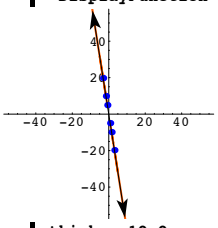
eigenvalue[1] > eigenvalue[2].

There is one theoretical exception: If starter[j] is an exact multiple of eigenvector[2], then the curves $E^{A\,t}$. starter[j] stay on the line through {0,0} defined by eigenvector[2].

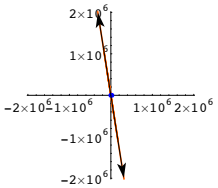
See this happen:

```
starter[1] = 20 eigenvector[2];
starter[2] = -20 eigenvector[2];
starter[3] = 10 eigenvector[2];
starter[4] = -10 eigenvector[2];
starter[5] = 5 eigenvector[2];
```

```
starter[6] = -5 eigenvector[2];
thigh = 1.0;
ranger = scaler[thigh];
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
 starterplots, eigenplot[thigh, 2],
 PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
 DisplayFunction -> $DisplayFunction];
```




```
thigh = 10.0;
ranger = scaler[thigh];
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
 starterplots, eigenplot[thigh, 2],
 PlotRange -> {{-ranger, ranger}, {-ranger, ranger}},
 DisplayFunction -> $DisplayFunction];
```



The reason for this is that starter[1], starter[2], starter[3], starter[4], startere[5] and starter[6] are all multiples of eigenvector[2].

◻B.5.b.ii) Why ultimate eigenvectors work they way they do - preliminary step

As a preliminary step toward explaining why ultimate eigenvectors work the way they do, explain this:

When you go with a diagonalizable 2D matrix A and calculate its eigenvalues, eigenval[1], eigenval[2] and its eigenvectors eigenvect[1] and eigenvect[2],
then you are guaranteed that
$E^{A\,t}$.(u eigenvect[1] + v eigenvect[2])

$$= E^{\text{eigenval}[1]\,t}\,(u\,\text{eigvect}[1] + v\,E^{(\text{eigenval}[2]-\text{eigenval}[1])\,t}\,\text{eigvect}[2])$$

◻Answer

Arrange the eigenvectors of A into the SpannerMatrix with eigenvect[1]in the leftmost vertical column and eigenvect[2] in the second vertical column of SpannerMatrix.

So that

$$A = \text{SpannerMatrix}.\begin{pmatrix}\text{eigval}[1] & 0\\ 0 & \text{eigval}[2]\end{pmatrix}.\text{SpannerMatrix}^{-1}.$$

and

$$A.\text{eigvect}[j] = \text{eigval}[j]\,\text{eigvect}[j]$$

for j = 1,2.

This gives

$$E^{A\,t} = \text{SpannerMatrix}.\begin{pmatrix}E^{\text{eigval}[1]\,t} & 0\\ 0 & E^{\text{eigval}[2]\,t}\end{pmatrix}.\text{SpannerMatrix}^{-1}$$

This tells you that

$$E^{A\,t}.\text{eigvect}[j] = E^{\text{eigval}[j]}\,\text{eigvect}[j]$$

for j = 1, 2.

So

$$E^{A\,t}.(u\,\text{eigvect}[1] + v\,\text{eigvect}[2])$$

$$= u\,E^{\text{eigenval}[1]\,t}\,\text{eigvect}[1] + v\,E^{\text{eigval}[2]\,t}\,\text{eigvect}[2].$$

This is the same as:

$$E^{\text{eigenval}[1]\,t}\,(u\,\text{eigvect}[1] + v\,E^{\text{eigval}[2]\,t}\;E^{-\text{eigenval}[1]\,t}\,\text{eigvect}[2]).$$

This is the same as

$$E^{\text{eigenval}[1]\,t}\,(u\,\text{eigvect}[1] + v\,E^{\text{eigval}[2]\,t-\text{eigval}[1]\,t}\,\text{eigvect}[2]).$$

And this is the same as

$$E^{\text{eigenval}[1]\,t}\,(u\,\text{eigvect}[1] + v\,E^{(\text{eigval}[2]-\text{eigval}[1])\,t}\,\text{eigvect}[2]).$$

This explains why

$$E^{A\,t}.(u\,\text{eigvect}[1] + v\,\text{eigvect}[2])$$

is the same as:

$$E^{\text{eigenval}[1]\,t}\,(u\,\text{eigvect}[1] + v\,E^{(\text{eigval}[2]-\text{eigval}[1])\,t}\,\text{eigvect}[2]).$$

◻B.5.b.iii) Why ultimate eigenvectors work they way they do -full explanation

Explain this:
If A is a diagonalizable 2D matrix , if
eigval[1] > eigval[2],
and if X is a given point NOT ON the line through {0,0} defined by eigenvect[2],
then as t gets large
$E^{A\,t}.X$
gravitates to the line through {0, 0} defined by eigenvect[1].

◻Answer:

Because {eigenvect[1], eigenvect[2]} is a basis of 2D,you are guaranteed that X is of the form

$$X = u\,\text{eigenvect}[1] + v\,\text{eigenvect}[2].$$

So

$$E^{A\,t}.X = E^{A\,t}.(u\,\text{eigenvect}[1] + v\,\text{eigenvect}[2]).$$

The part immediately above tells you that

$$E^{At}.X = E^{At} .. (u\,\text{eigenvect}[1] + v\,\text{eigenvect}[2]) =$$

$$= E^{\text{eigenval}[1]\,t}\,(u\,\text{eigvect}[1] + v\,E^{(\text{eigenval}[2]-\text{eigenval}[1])\,t}\,\text{eigvect}[2])$$

And because X is not on the line through {0,0} defined by eigenvect[2], you are guaranteed that $u \neq 0$.

So $E^{At}.X$ points in the same direction as

$$(u\,\text{eigvect}[1] + v\,E^{(\text{eigval}[2]-\text{eigval}[1])\,t}\,\text{eigvect}[2]) ..$$

Now because

$$\text{eigval}[1] > \text{eigval}[2],$$

it is certain that

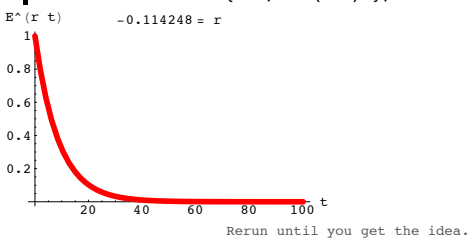
$$E^{(\text{eigval}[2]-\text{eigval}[1])\,t} \to 0$$

as t gets large.

Click on the right for a detailed explanation of this.

Because
eigval[1] > eigval[2],
you know that
$\text{eigval}[2] - \text{eigval}[1] < 0$,
so
$E^{(\text{eigval}[2]-\text{eigval}[1])\,t} \to 0$ as t gets large.
See $E^{r\,t} \to 0$as t gets large for some random negative numbers r:

```
r = Random[Real, {-1, -0.1}];

Cleart[t];
    Plot[E^(r t), {t, 0, 100},
      PlotStyle -> {{Thickness[0.02], Red}}, PlotRange -> All,
  AspectRatio -> 1 / GoldenRatio, AxesOrigin -> {0, 0},
     AxesLabel -> {"t", "E^(r t)"}, PlotLabel -> r "= r"];
```



Rerun until you get the idea.

The upshot: As t gets large the limiting direction of

$E^{At}.X$

is

(u eigvect[1] + v (0) eigvect[2])

= u eigenvect[1].

And now because u ≠ 0 this tells you that the limiting direction of E^(A t).X is on the line through {0,0} defined by eigenvect[1].

B.6) Eigenvalues of A tell how A^k.X plots out as k gets large

Eigenvalues and eigenvectors of A reveal the ultimate direction of A^k.X as k gets large

□B.6.a.i) If A is any diagonalizable matrix and the absolute values of all the eigenvalues of A are less than 1, then no matter what X is, you are guaranteed that

$A^k.X \to \{0, 0\}$.

Here's a 2D matrix A:

```
A = {{0.8, 0.1}, {-0.4, -0.3}};
MatrixForm[A]
```

$$\begin{pmatrix} 0.8 & 0.1 \\ -0.4 & -0.3 \end{pmatrix}$$

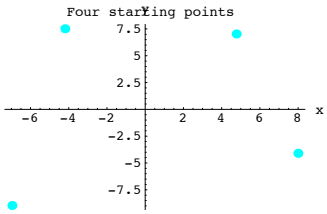
Here are four random points

{starter[1], starter[2], starter[3], starter[4]}

in 2D:

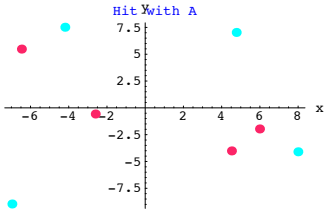
```
A = {{0.8, 0.1}, {-0.4, -0.3}};
Clear[iterationplot, interationplotter, eigenplot, scaler, point,
  pointcolor, eigenvector, starter, khigh, k, j, iterationpoints];
point[k_, starter_] := MatrixPower[A, k].starter;
pointcolor[k_, khigh_] := RGBColor[Sin[(Pi/2) k/(khigh + 0.1)],
  Cos[(Pi/2) k/(khigh + 0.1)], E^(-k/(khigh + 0.1))]

iterationpoints[starter_, khigh_] :=
Table[Graphics[{pointcolor[k, khigh], PointSize[0.03],
    Point[MatrixPower[A, k].starter]}], {k, 0, khigh}];
```

```
iterationplotter[khigh_] :=
 Show[Table[iterationpoints[starter[j], khigh], {j, 1, 4}],
  Axes -> True, AxesLabel -> {"x", "y"},
  PlotLabel -> "Hits with A,A^2,..,A^khigh",
      AspectRatio -> 1/GoldenRatio, DisplayFunction -> Identity];

starter[1] = {Random[Real, {4, 10}], Random[Real, {4, 10}]};
starter[2] = {Random[Real, {-10, -4}], Random[Real, {-10, -4}]};
starter[3] = {Random[Real, {-10, -4}], Random[Real, {4, 10}]};
starter[4] = {Random[Real, {4, 10}], Random[Real, {-10, -4}]};

khigh = 0;
Show[iterationplotter[khigh],
  PlotLabel -> "Four starting points ",
  DisplayFunction -> $DisplayFunction];
```



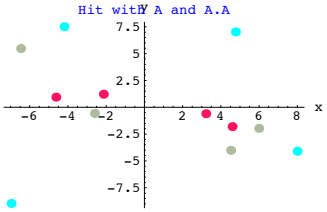
Here are the same four starter points together with the points you get when you hit each of the original starting points with A:

```
khigh = 1;
Show[iterationplotter[khigh],
      PlotLabel -> "Hit with A",
 DisplayFunction -> $DisplayFunction];
```



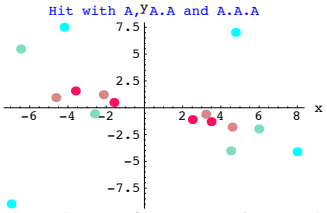
Here are the same four starter points together with the points you get when you hit each of the original starting points with A and with A.A:

```
khigh = 2;
Show[iterationplotter[khigh],
      PlotLabel -> "Hit with A and A.A",
 DisplayFunction -> $DisplayFunction];
```



Here are the same four starter points together with the points you get when you hit each of the original starting points with A, A.A and A.A.A:

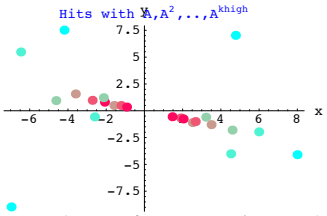
```
khigh = 3;
Show[iterationplotter[khigh],
     PlotLabel -> "Hit with A, A.A and A.A.A",
     DisplayFunction -> $DisplayFunction];
```



Here are the same four starter points together with the points you get when you hit each of the original starting points with A, A.A, A^3, A^4, and A^5:

All of these powers are done with matrix multiplication.

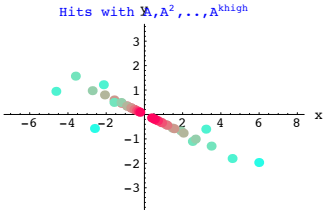
```
khigh = 5;
Show[iterationplotter[khigh],
  DisplayFunction -> $DisplayFunction];
```



Here are the same four starter points together with the points you get when you hit each of the original starting points with A, A.A, A^3, A^4, . . ., A^9, A^{10}:

All of these powers are done with matrix multiplication.

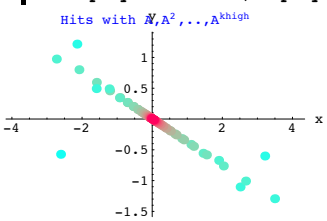
```
khigh = 10;
Show[iterationplotter[khigh], DisplayFunction -> $DisplayFunction];
```



Here are the same four starter points together with the points you get when you hit each of the original starting points with
A, A.A, A^3, A^4, . . ., A^{19}, A^{20}:

All of these powers are done with matrix multiplication.

```
khigh = 20;
Show[iterationplotter[khigh],
  DisplayFunction -> $DisplayFunction];
```



Grab the plots, align and animate.

As k gets large,
A^k.starter[j] → {0, 0}.
Now look at the eigenvalues of A:

```
Clear[eigenvalue];
{eigenval[1], eigenval[2]} = Eigenvalues[A]
```

```
{0.762348, -0.262348}
```

Take note that
|eigenval[1]| < 1 and |eigenval[2]| < 1.
The absolute values of the eigenvalues of A are both less than 1.
Explain this statement:
If A is any diagonalizable matrix and the absolute values of all the eigenvalues of A are less than 1, then no matter what X is, you are guaranteed that
A^k.X → {0, 0}.

□ **Answer:**

Arrange the eigenvectors of A into the SpannerMatrix with eigenvect[1]in the leftmost vertical column and eigenvect[2] in the second vertical column of SpannerMatrix.

So that

$$A = \text{SpannerMatrix}.\begin{pmatrix} \text{eigval[1]} & 0 \\ 0 & \text{eigval[2]} \end{pmatrix}.\text{SpannerMatrix}^{-1}.$$

This gives

$$A^k = \text{SpannerMatrix}.\begin{pmatrix} \text{eigval[1]}^k & 0 \\ 0 & \text{eigval[2]}^k \end{pmatrix}.\text{SpannerMatrix}^{-1}$$

So

$$A^k.X = \text{SpannerMatrix}.\begin{pmatrix} \text{eigval[1]}^k & 0 \\ 0 & \text{eigval[2]}^k \end{pmatrix}.\text{SpannerMatrix}^{-1}.X$$

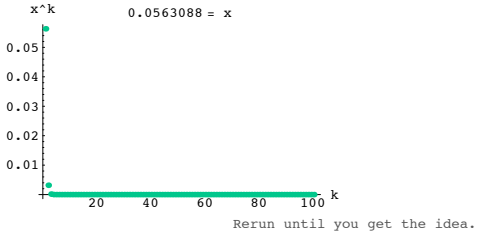
Because |eigenval[1]| < 1 and |eigenval[2]| < 1, you are guaranteed that
$\text{eigenval[1]}^k \dashrightarrow 0$ and $\text{eigenval[2]}^k \longrightarrow 0$
as k gets large .

Click on the right for an explanation of this.

Go with a random x with -1 < x < 1 and plot the powers
x, x^2, x^3, ..., x^{99}, x^{100}:

```
x = Random[Real, {-1, 1}];
xpowers = Table[{k, x^k}, {k, 1, 100}];
```

```
ListPlot[xpowers,
       PlotStyle -> {PointSize[0.02], TurquoiseBlue},
  PlotRange -> All,
      AxesLabel -> {"k", "x^k"}, PlotLabel -> x "= x"];
```



Rerun until you get the idea.

So as k gets large

$$A^k.X \dashrightarrow \text{SpannerMatrix}.\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.\text{SpannerMatrix}^{-1}.X = \{0, 0\}.$$

□ **B.6.a.ii) If A is any diagonalizable matrix and the absolute values of both eigenvalues of A are both bigger than 1, then no matter what X is (as long as X ≠ {0,0}), you are guaranteed that**

|| A^k.X || → ∞.

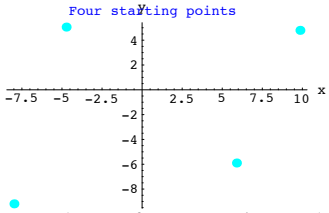
Here's a new 2D matrix A:

```
A = ( 1.8   0.2
     -0.3  -1.4 );
MatrixForm[A]
```

$$\begin{pmatrix} 1.8 & 0.2 \\ -0.3 & -1.4 \end{pmatrix}$$

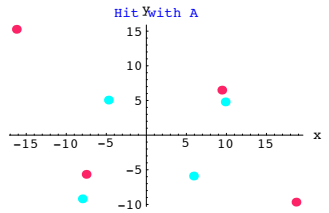
Here are four random points
{starter[1], starter[2], starter[3], starter[4]}
in 2D:

```
A = ( 1.8   0.2
     -0.3  -1.4 );
Clear[iterationplot, interationplotter, eigenplot, scaler, point,
  pointcolor, eigenvector, starter, khigh, k, j, iterationpoints];
point[k_, starter_] := MatrixPower[A, k] .starter ;
pointcolor[k_, khigh_] := RGBColor[Sin[(Pi / 2) k / (khigh + 0.1)],
  Cos[(Pi / 2) k / (khigh + 0.1)], E^(-k / (khigh + 0.1))]

iterationpoints[starter_, khigh_] :=
Table[Graphics[ {pointcolor[k, khigh], PointSize[0.03],
     Point[MatrixPower[A, k] .starter]}], {k, 0, khigh}];

iterationplotter[khigh_] :=
 Show[Table[iterationpoints[starter[j], khigh], {j, 1, 4}],
  Axes -> True, AxesLabel -> {"x", "y"},
  PlotLabel -> "Hits with A,A^2,..,A^khigh",
       AspectRatio -> 1 / GoldenRatio, DisplayFunction -> Identity];
starter[1] = {Random[Real, {4, 10}], Random[Real, {4, 10}]};
starter[2] = {Random[Real, {-10, -4}], Random[Real, {-10, -4}]};
starter[3] = {Random[Real, {-10, -4}], Random[Real, {4, 10}]};
starter[4] = {Random[Real, {4, 10}], Random[Real, {-10, -4}]};

khigh = 0;
Show[iterationplotter[khigh], PlotLabel -> "Four starting points ",
 DisplayFunction -> $DisplayFunction];
```



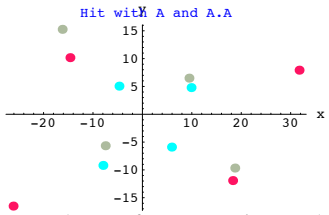
Here are the same four starter points together with the points you get when you hit each of the original starting points with A:

```
khigh = 1;
Show[iterationplotter[khigh],
     PlotLabel -> "Hit with A",
 DisplayFunction -> $DisplayFunction];
```



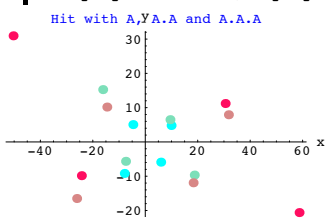
Here are the same four starter points together with the points you get when you hit each of the original starting points with A and A.A:

```
khigh = 2;
Show[iterationplotter[khigh],
 PlotLabel -> "Hit with A and A.A",
 DisplayFunction -> $DisplayFunction];
```



Here are the same four starter points together with the points you get when you hit each of the original starting points with A, A.A and A.A.A:

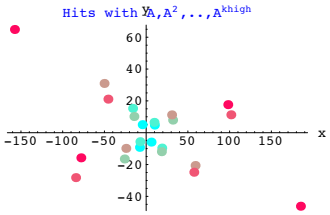
```
khigh = 3;
Show[iterationplotter[khigh],
 PlotLabel -> "Hit with A, A.A and A.A.A",
 DisplayFunction -> $DisplayFunction];
```



Here are the same four starter points together with the points you get when you hit each of the original four starting points with A, A.A, A^3, A^4, and A^5:

All of these powers are done with matrix multiplication.

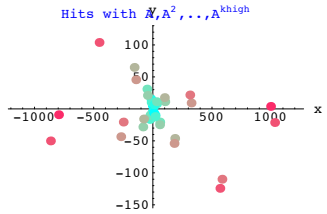
```
khigh = 5;
Show[iterationplotter[khigh],
  DisplayFunction -> $DisplayFunction];
```



Here are the same four starter points together with the points you get when you hit each of the original starting points with A, A.A, A^3, A^4, . . ., A^9, A^{10}:

All of these powers are done with matrix multiplication.

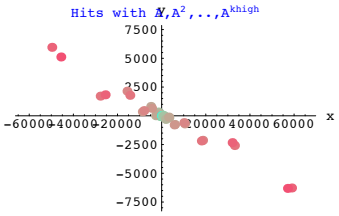
```
khigh = 10;
Show[iterationplotter[khigh],
  DisplayFunction -> $DisplayFunction];
```



Here are the same four starter points together with the points you get when you hit each of the original starting points with
A, A.A, A^3, A^4, . . ., A^{19}, A^{20}:

All of these powers are done with matrix multiplication.

```
khigh = 20;
Show[iterationplotter[khigh],
            DisplayFunction -> $DisplayFunction];
```



As k gets large,
$\| A)^k.\text{starter[j]} \| \to \infty$.
Now look at the eigenvalues of A:

```
Clear[eigenvalue];
{eigenval[1], eigenval[2]} = Eigenvalues[A]
```

{1.78114, -1.38114}

|eigenval[1]| > 1 and |eigenval[2]| > 1.
The absolute values of the eigenvalues of A are both bigger than 1.
Explain this statement:
If A is any diagonalizable matrix and the absolute values of the of both eigenvalues of A are both bigger than 1, then no matter what as long as $X \neq \{0,0\}$, you are guaranteed that
$\| A^k.X \| \to \infty$.

□ **Answer:**

Arrange the eigenvectors of A into the SpannerMatrix with eigenvect[1] in the leftmost vertical column and eigenvect[2] in the second vertical column of SpannerMatrix.

So that

$$A = \text{SpannerMatrix}.\begin{pmatrix} \text{eigval}[1] & 0 \\ 0 & \text{eigval}[2] \end{pmatrix}.\text{SpannerMatrix}^{-1}.$$

This gives

$$A^k = \text{SpannerMatrix}.\begin{pmatrix} \text{eigval}[1]^k & 0 \\ 0 & \text{eigval}[2]^k \end{pmatrix}.\text{SpannerMatrix}^{-1}$$

So

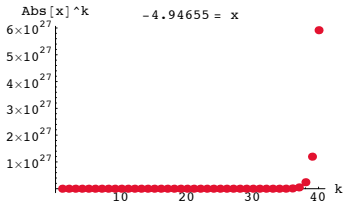
$$A^k.X = \text{SpannerMatrix}.\begin{pmatrix} \text{eigval}[1]^k & 0 \\ 0 & \text{eigval}[2]^k \end{pmatrix}.\text{SpannerMatrix}^{-1}.X$$

Because |eigenval[1]| > 1 and |eigenval[2]| > 1, you are guaranteed that
$|\text{eigenval}[1]^k| \to \infty$ and $|\text{eigenval}[2]^k| \to \infty$
as k gets large .

Click on the right for an explanation of this.

Go with a random x with |x| > 1 and plot the powers
$|x|, |x|^2, |x|^3, \ldots, |x|^{39}, |x|^{40}$:

```
x = (-1)^Random[Integer, {0, 1}] Random[Real, {1, 5}];
xpowers = Table[{k, Abs[x]^k}, {k, 1, 40}];
ListPlot[xpowers,
       PlotStyle -> {PointSize[0.03], GeraniumLake},
  PlotRange -> All,
      AxesLabel -> {"k", "Abs[x]^k"}, PlotLabel -> x "= x"];
```



Rerun until you get the idea.

The result:

Unless X = {0,0},

$\| A^k.X \| \to \infty$.

as k gets large.

□ **B.6.b.i) If |eigenvalue[1]| > |eigenvalue[2]|, then the ultimate direction of $A^k.X$ as k gets large is in the direction of eigenvector[1] (unless X is a multiple of eigenvector[2])**

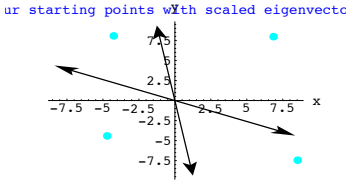
Here's a 2D matrix A:

```
A = {{1.20, 0.11}, {-0.43, 0.32}};
MatrixForm[A]
```

$$\begin{pmatrix} 1.2 & 0.11 \\ -0.43 & 0.32 \end{pmatrix}$$

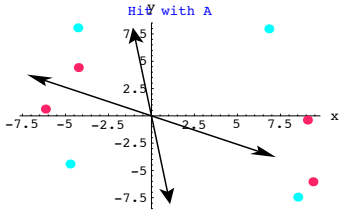
Here are four random points
{starter[1], starter[2], starter[3], starter[4]}
in 2D shown with scaled versions of the eigenvectors of A:

```
A = {{1.20, 0.11}, {-0.43, 0.32}};
Clear[iterationplot, interationplotter, eigenplot, point, scaler,
  pointcolor, eigenvector, starter, khigh, k, j, iterationpoints];
point[k_, starter_] := MatrixPower[A, k].starter;
pointcolor[k_, khigh_] := RGBColor[Sin[(Pi/2) k/(khigh + 0.1)],
  Cos[(Pi/2) k/(khigh + 0.1)], E^(-k/(khigh + 0.1))]

iterationpoints[starter_, khigh_] :=
Table[Graphics[ {pointcolor[k, khigh], PointSize[0.03],
     Point[MatrixPower[A, k].starter]}], {k, 0, khigh}];

iterationplotter[khigh_] :=
 Show[Table[iterationpoints[starter[j], khigh], {j, 1, 4}],

  Axes -> True, AxesLabel -> {"x", "y"},
  PlotLabel -> "Hits with A,A^2,..,A^khigh",
       AspectRatio -> 1/GoldenRatio, DisplayFunction -> Identity];


scaler[khigh_] :=
  0.9 Sqrt[point[khigh, starter[1]].point[khigh, starter[1]]];

{eigenvector[1], eigenvector[2]} = Eigenvectors[A];
eigenplot[khigh_, j_] :=
    {Arrow[scaler[khigh] eigenvector[j],
    Tail -> {0, 0}, VectorColor -> Black],
     Arrow[-scaler[khigh] eigenvector[j],
    Tail -> {0, 0}, VectorColor -> Black]};




starter[1] = {Random[Real, {4, 10}], Random[Real, {4, 10}]};
starter[2] = {Random[Real, {-10, -4}], Random[Real, {-10, -4}]};
starter[3] = {Random[Real, {-10, -4}], Random[Real, {4, 10}]};
starter[4] = {Random[Real, {4, 10}], Random[Real, {-10, -4}]};

khigh = 0;
Show[iterationplotter[khigh],
 eigenplot[khigh, 1], eigenplot[khigh, 2],
 PlotLabel -> "Four starting points with scaled eigenvectors",
 DisplayFunction -> $DisplayFunction];
```



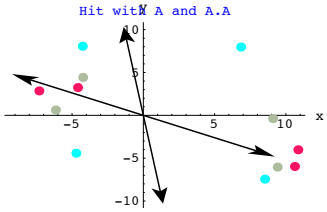
Here are the same four starter points together with the points you get when you hit each of the original starting points with A:

```
khigh = 1;
Show[iterationplotter[khigh],
 eigenplot[khigh, 1], eigenplot[khigh, 2],
              PlotLabel -> "Hit with A",
 DisplayFunction -> $DisplayFunction];
```



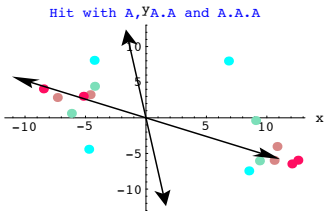
Here are the same four starter points together with the points you get when you hit each of the original starting points with A and A.A:

```
khigh = 2;
Show[iterationplotter[khigh],
 eigenplot[khigh, 1], eigenplot[khigh, 2],
     PlotLabel -> "Hit with A and A.A",
 DisplayFunction -> $DisplayFunction];
```



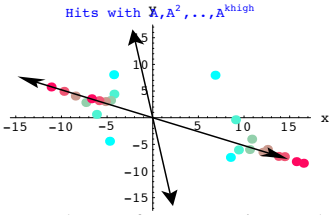
Here are the same four starter points together with the points you get when you hit each of the original starting points with A, A.A and A.A.A:

```
khigh = 3;
Show[iterationplotter[khigh],
 eigenplot[khigh, 1], eigenplot[khigh, 2],
             PlotLabel -> "Hit with A, A.A and A.A.A",
 DisplayFunction -> $DisplayFunction];
```



Here are the same four starter points together with the points you get when you hit each of the original starting points with A, A.A, A^3, A^4, and A^5:

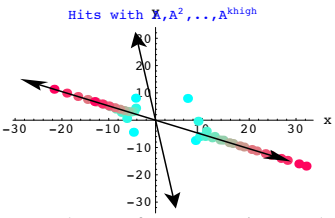
```
khigh = 5;
Show[iterationplotter[khigh],
  eigenplot[khigh, 1], eigenplot[khigh, 2],
              DisplayFunction -> $DisplayFunction];
```



Here are the same four starter points together with the points you get when you hit each of the original starting points with A, A.A, A^3, A^4, . . ., A^9 , A^{10}:

All of these powers are done with matrix multiplication.

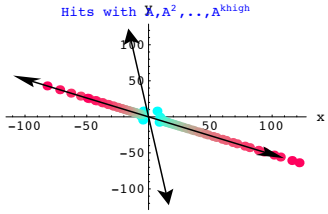
```
khigh = 10;
Show[iterationplotter[khigh],
  eigenplot[khigh, 1], eigenplot[khigh, 2],
              DisplayFunction -> $DisplayFunction];
```



Here are the same four starter points together with the points you get when you hit each of the original starting points with
A, A.A, A^3, A^4, . . ., A^{19} , A^{20}:

All of these powers are done with matrix multiplication.

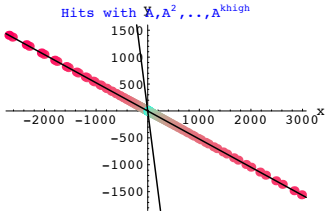
```
khigh = 20;
Show[iterationplotter[khigh],
  eigenplot[khigh, 1], eigenplot[khigh, 2],
              DisplayFunction -> $DisplayFunction];
```



Here are the same four starter points together with the points you get when you hit each of the original starting points with
A, A.A, A^3, A^4, . . ., A^{49} , A^{50}:

All of these powers are done with matrix multiplication.

```
khigh = 50;
Show[iterationplotter[khigh],
  eigenplot[khigh, 1], eigenplot[khigh, 2],
              DisplayFunction -> $DisplayFunction];
```



As k gets large, the points

A^k.starter[j]

seem to shy away from one of the eigenvectors and to gravitate to the line through {0,0} defined by the ultimate eigenvector.

How do you identify the ultimate eigenvector?

□ **Answer:**

Look at the eigenvectors of A:

```
Clear[eigenvector];
{eigenvector[1], eigenvector[2]} = Eigenvectors[A]
```

```
{{0.886198, -0.463306}, {-0.13256, 0.991175}}
```

Look at the eigenvalues:

```
Clear[eigenvalue];
{eigenvalue[1], eigenvalue[2]} = Eigenvalues[A]
```

```
{1.14249, 0.377508}
```

Note that

| eigenvalue[1] | > | eigenvalue[2] |.

This signals that eigenvector[1] is ultimate.

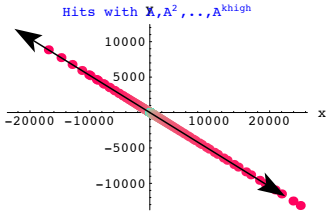
This means that as k gets large, the plotted points

A^k.starter[j]

are attracted to the line through {0, 0} defined by eigenvector[1] .

See it happen in this plot which displays only the line through {0, 0} defined by eigenvector[1].

```
khigh = 60;
Show[iterationplotter[khigh],
  eigenplot[khigh, 1], PlotRange -> All,
              DisplayFunction -> $DisplayFunction];
```

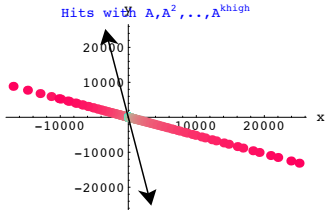


As k gets large, the points

A^k.starter[j]

just laugh at the line through {0,0} defined by eigenvector[2]:

```
khigh = 60;
Show[iterationplotter[khigh],
  eigenplot[khigh, 2], PlotRange -> All,
              DisplayFunction -> $DisplayFunction];
```



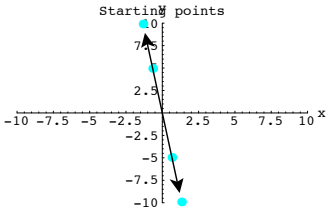
This happens because

$|\text{eigenvalue}[1]| > |\text{eigenvalue}[2]|$.

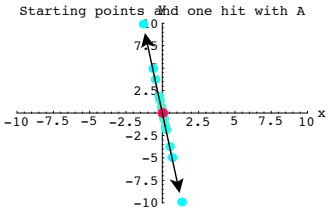
The is one theoretical exception: If starter is an exact multiple of eigenvector[2], then the plotted points A^k.starter[j]stay on the line through {0,0} defined by eigenvector[2].

See this happen:

```
starter[1] = 10 eigenvector[2];
starter[2] = -10 eigenvector[2];
starter[3] = 5 eigenvector[2];
starter[4] = -5 eigenvector[2];
khigh = 0;
Show[iterationplotter[khigh],
  eigenplot[0, 2], PlotRange -> {{-10, 10}, {-10, 10}},
              PlotLabel -> "Starting points",
  DisplayFunction -> $DisplayFunction];
```




```
khigh = 12;
Show[iterationplotter[khigh],
  eigenplot[0, 2], PlotRange -> {{-10, 10}, {-10, 10}},
              PlotLabel -> "Starting points and one hit with A",
  DisplayFunction -> $DisplayFunction];
```



The reason for this is that starter[1], starter[2], starter[3] and starter[4] are all multiples of eigenvector[2].

☐B.6.b.ii) Explanation-Preliminary step

As a preliminary step toward explaining why ultimate eigenvectors work the way they do, explain this:
When you go with a diagonalizable 2D matrix A and calculate its eigenvalues,
eigenval[1] and eigenval[2]
and its corresponding eigenvectors
eigenvect[1] and eigenvect[2],
then

$A^k.(s\,\text{eigenvect}[1] + t\,\text{eigenvect}[2])$

$$= \text{eigval}[1]^k \left(s\ \text{eigvect}[1] + t \left(\frac{\text{eigval}[2]}{\text{eigval}[1]}\right)^k \text{eigvect}[2]\right)$$

☐Answer

Arrange the eigenvectors of A into the SpannerMatrix with eigenvect[1]in the leftmost vertical column and eigenvect[2] in the second vertical column of SpannerMatrix.

So that

$$A = \text{SpannerMatrix}.\begin{pmatrix} \text{eigval}[1] & 0 \\ 0 & \text{eigval}[2] \end{pmatrix}.\text{SpannerMatrix}^{-1}.$$

This gives

$$A^k = \text{SpannerMatrix}.\begin{pmatrix} \text{eigval}[1]^k & 0 \\ 0 & \text{eigval}[2]^k \end{pmatrix}.\text{SpannerMatrix}^{-1}$$

This tells you that

$A^k.\text{eigvect}[j] = \text{eigval}[j]^k\ \text{eigvect}[j]$

for j = 1, 2.

So

$A^k.(s\,\text{eigvect}[1] + t\,\text{eigvect}[2])$

$= s\,\text{eigval}[1]^k\ \text{eigvect}[1] + t\,\text{eigval}[2]^k\ \text{eigvect}[2].$

Factor out $\text{eigval}[1]^k$to get::

$A^k.(s\,\text{eigvect}[1] + t\,\text{eigvect}[2])$

$$= \text{eigval}[1]^k \left(s\ \text{eigvect}[1] + t\ \frac{\text{eigval}[2]^k}{\text{eigval}[1]^k}\ \text{eigvect}[2]\right)$$

$$= \text{eigval}[1]^k \left(s\ \text{eigvect}[1] + t \left(\frac{\text{eigval}[2]}{\text{eigval}[1]}\right)^k \text{eigvect}[2]\right).$$

Explanation complete.

☐B.6.b.iii) Full explanation

Explain this:
If A is a diagonalizable 2D matrix ,if
$|\text{eigval}[1]| > |\text{eigval}[2]|$,
and if X is a given point NOT ON the line through {0,0} defined by eigenvect[2],
then as k gets large
$A^k.X$
gravitates to the line through {0, 0} defined by eigenvect[1].

☐Answer:

Because {eigenvect[1], eigenvect[2]} is a basis of 2D,you are guaranteed that X is of the form

$X = s\,\text{eigenvect}[1] + t\,\text{eigenvect}[2].$

So

$A^k.X = A^k.(s\,\text{eigenvect}[1] + t\,\text{eigenvect}[2]).$

The part immediately above tells you that

$A^k.X = A^k.(s\,\text{eigenvect}[1] + t\,\text{eigenvect}[2])$

$$= \text{eigval}[1]^k \left(s\ \text{eigvect}[1] + t \left(\frac{\text{eigval}[2]}{\text{eigval}[1]}\right)^k \text{eigvect}[2]\right).$$

And because X is not on the line through {0,0} defined by eigenvect[2], you are guaranteed that $s \neq 0$.

So $A^k.X$ points in the same direction as

$$\pm\left(s\ \text{eigvect}[1] + t \left(\frac{\text{eigval}[2]}{\text{eigval}[1]}\right)^k \text{eigvect}[2]\right).$$

Now because

$|\text{eigval}[1]| > |\text{eigval}[2]|$,

it's certain that

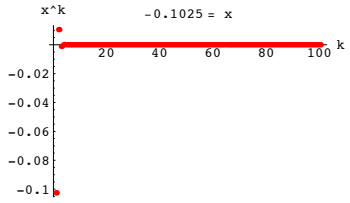
$$-1 < \frac{\text{eigenval}[2]}{\text{eigenval}[1]} < 1\ .$$

So as k gets large $\left(\frac{\text{eigval}[2]}{\text{eigval}[1]}\right)^k$ closes in on 0..

Click on the right for an explanation of this.

Go with a random x with -1 < x < 1 and plot the powers
$x, x^2, x^3, \ldots, x^{99}, x^{100}$:

```
x = Random[Real, {-1, 1}];
xpowers = Table[{k, x^k}, {k, 1, 100}];
ListPlot[xpowers,
       PlotStyle -> {PointSize[0.02], Red}, PlotRange -> All,
      AxesLabel -> {"k", "x^k"}, PlotLabel -> x "= x"];
```



Rerun until you get the idea.

The upshot: As k gets large the limiting direction of

$A^k.X$

is

$\pm(s\ \text{eigvect}[1] + t\,(0)\,\text{eigvect}[2])$

$= \pm\, s\,\text{eigenvect}[1].$

And now because $s \neq 0$ this tells you that the limiting direction of

$A^k.X$ is on the line through {0,0} defined by eigenvect[1].

☐B.6.b.iv) Staying on a line

Explain this:
If A is a diagonalizable 2D matrix ,if
and if X is ON the line through {0,0} defined by eigenvect[j],
then as k gets large
$A^k.X$
stays on to the line through {0, 0} defined by eigenvect[j].

☐Answer:

Because X is on the line through {0,0} defined by eigenvector[j] , you know that

$X = t\,\text{eigenvect}[j]$.

So

$A^k.X = A^k.t\,\text{eigenvect}[2] = t\,\text{eigenval}[j]^k\,\text{eigenvect}[j]$.

The upshot:

No matter what k is, $A^k.X$ is on the line through {0,0} defined by eigenvect[j]

B.7) Sometimes you have to go to complex kD to diagonalize a matrix.

The matrix exponential for matrices diagonalizable with complex numbers

How $E^{At}.X$ swirls when A is diagonalizable with complex numbers.

□ Background on Complex kD

For usual kD, you go with vectors whose slots are real numbers.
For complex kD, you go with vectors whose slots are complex numbers a + *i* b where $i = \sqrt{-1}$.
Here is a random vector in complex3D:

```
dim = 3;
Clear[k];
complexX =
 Table[Random[Real, {-10, 10}] Random[Complex], {k, 1, dim}]
```

```
{0.0792902 + 0.0848301 i, 2.96246 + 1.46079 i, 4.76229 + 1.36851 i}
```

Here is a random member of complex5D:

```
dim = 5;
Clear[k];
complexX =
 Table[Random[Real, {-10, 10}] Random[Complex], {k, 1, dim}]
```

```
{-7.46804 - 3.59636 i, 3.27019 + 0.0129144 i,
 -2.08443 - 5.55286 i, -0.447458 - 4.57564 i, 3.99551 + 3.74049 i}
```

You can't plot them but you can add and subtract them and you can multiply them by complex numbers.

□ B.7.a.i) Complex eigenvectors can happen, but they don't block your path

Go with this 2D matrix A:

```
A = {{0.2, 1.2}, {-1.5, 0.4}};
MatrixForm[A]
```

$$\begin{pmatrix} 0.2 & 1.2 \\ -1.5 & 0.4 \end{pmatrix}$$

Calculate the eigenvectors of A:

```
Clear[eigenvector];
spanners = Eigenvectors[A]
```

```
{{0.0496904 - 0.664812 i, 0.745356}, {0.0496904 + 0.664812 i, 0.745356}}
```

Note the presence of $i = \sqrt{-1}$.
In spite of this, calculate the dimension of the subspace of complex 2D spanned by the eigenvectors of A:

```
SpannerMatrix = Transpose[spanners];
eigendim = Length[SingularValues[SpannerMatrix][[2]]]
```

```
2
```

This tells you that the eigenvectors of A form a basis for all of complex2D and that the spanner matrix is invertible.
Explain why this signals that A is diagonalizable with complex numbers.

□ Answer:

To see that A is diagonalizable with complex numbers, you have to be able to duplicate A with a matrix of this form:

SpannerMatrix.diagonalmatrix.SpannerMatrix^{-1}

The SpannerMatrix you need is already available:

```
Clear[eigenvector];
spanners = Eigenvectors[A];
SpannerMatrix = Transpose[spanners];
MatrixForm[SpannerMatrix]
```

$$\begin{pmatrix} 0.0496904 - 0.664812\,i & 0.0496904 + 0.664812\,i \\ 0.745356 & 0.745356 \end{pmatrix}$$

The diagonal matrix comes from the eigenvalues of A:

```
diagonalmatrix = DiagonalMatrix[Eigenvalues[A]];
MatrixForm[diagonalmatrix]
```

$$\begin{pmatrix} 0.3 + 1.33791\,i & 0 \\ 0 & 0.3 - 1.33791\,i \end{pmatrix}$$

Here's a look at

SpannerMatrix.diagonalmatrix.SpannerMatrix^{-1}:

```
MatrixForm[SpannerMatrix.diagonalmatrix.Inverse[SpannerMatrix]]
```

$$\begin{pmatrix} 0.2 & 1.2 \\ -1.5 & 0.4 \end{pmatrix}$$

Here's a look at A:

```
MatrixForm[A]
```

$$\begin{pmatrix} 0.2 & 1.2 \\ -1.5 & 0.4 \end{pmatrix}$$

The upshot:

A = SpannerMatrix.diagonalmatrix.SpannerMatrix^{-1}:

So A is diagonalizable with complex numbers..

□ B.7.a.ii) 3D sample

Do one in 3D.

□ Answer:

Happy to oblige.

Go with:

```
A = {{0.2, 1.2, 1.2}, {-1.5, 0.4, 0.7}, {0.5, 1.2, -0.3}};
MatrixForm[A]
```

$$\begin{pmatrix} 0.2 & 1.2 & 1.2 \\ -1.5 & 0.4 & 0.7 \\ 0.5 & 1.2 & -0.3 \end{pmatrix}$$

Calculate the eigenvectors of A:

```
Clear[eigenvector];
spanners = Eigenvectors[A]
```

```
{{0.690044, -0.0822972 + 0.569322 i, 0.411176 + 0.154506 i},
 {0.690044, -0.0822972 - 0.569322 i, 0.411176 - 0.154506 i},
 {-0.219603, -0.544963, 0.809191}}
```

Note the presence of $i = \sqrt{-1}$.

In spite of this, calculate the dimension of the subspace of complex3D spanned by the eigenvectors of A:

```
SpannerMatrix = Transpose[spanners];
eigendim = Length[SingularValues[SpannerMatrix][[2]]]
```

```
3
```

This tells you that the eigenvectors of A form a basis for all of complex3D and that the spanner matrix is invertible.

To diagonalize A with complex numbers, you have to be able to duplicate A with a matrix of this form:

SpannerMatrix.diagonalmatrix.SpannerMatrix^{-1}

The SpannerMatrix you need is already available:

```
Clear[eigenvector];
spanners = Eigenvectors[A];
SpannerMatrix = Transpose[spanners];
MatrixForm[SpannerMatrix]
```

$$\begin{pmatrix} 0.690044 & 0.690044 & -0.219603 \\ -0.0822972 + 0.569322\,i & -0.0822972 - 0.569322\,i & -0.544963 \\ 0.411176 + 0.154506\,i & 0.411176 - 0.154506\,i & 0.809191 \end{pmatrix}$$

The diagonal matrix comes from the eigenvalues of A:

```
diagonalmatrix = DiagonalMatrix[Eigenvalues[A]];
MatrixForm[diagonalmatrix]
```

$$\begin{pmatrix} 0.771926 + 1.25875\,i & 0 & 0 \\ 0 & 0.771926 - 1.25875\,i & 0 \\ 0 & 0 & -1.24385 \end{pmatrix}$$

Here's a look at

SpannerMatrix.diagonalmatrix.SpannerMatrix^{-1}:

```
MatrixForm[SpannerMatrix.diagonalmatrix.Inverse[SpannerMatrix]]
```

$$\begin{pmatrix} 0.2 & 1.2 & 1.2 \\ -1.5 & 0.4 & 0.7 \\ 0.5 & 1.2 & -0.3 \end{pmatrix}$$

Here's a look at A:

```
MatrixForm[A]
```

$$\begin{pmatrix} 0.2 & 1.2 & 1.2 \\ -1.5 & 0.4 & 0.7 \\ 0.5 & 1.2 & -0.3 \end{pmatrix}$$

The upshot:

A = SpannerMatrix.diagonalmatrix.SpannerMatrix^{-1}:

So A is diagonalizable with complex numbers..

B.7.b.i) The matrix exponential E^{At} for matrices that are diagonalizable with complex numbers

Here's a new 2D matrix A:

```
A = {{0.3, 1.2}, {-1.5, -0.4}};
MatrixForm[A]
```

$$\begin{pmatrix} 0.3 & 1.2 \\ -1.5 & -0.4 \end{pmatrix}$$

Calculate the eigenvectors of A:

```
Clear[eigenvector];
spanners = Eigenvectors[A]
```

```
{{-0.173916 - 0.643582 i, 0.745356}, {-0.173916 + 0.643582 i, 0.745356}}
```

Note the presence of $i = \sqrt{-1}$.
In spite of this, calculate the matrix exponential E^{At}.

Answer:

Check:

```
SpannerMatrix = Transpose[spanners];
eigendim = Length[SingularValues[SpannerMatrix][[2]]]
```

```
2
```

This tells you that the eigenvectors of A form a basis for all of complex2D and that the spanner matrix is invertible.

To diagonalize A with complex numbers, you have to be able to duplicate A with a matrix of this form:

SpannerMatrix.diagonalmatrix.SpannerMatrix^{-1}

The SpannerMatrix you need is already available:

```
Clear[eigenvector];
spanners = Eigenvectors[A];
SpannerMatrix = Transpose[spanners];
MatrixForm[SpannerMatrix]
```

$$\begin{pmatrix} -0.173916 - 0.643582\,i & -0.173916 + 0.643582\,i \\ 0.745356 & 0.745356 \end{pmatrix}$$

The diagonal matrix comes from the eigenvalues of A:

```
diagonalmatrix = DiagonalMatrix[Eigenvalues[A]];
MatrixForm[diagonalmatrix]
```

$$\begin{pmatrix} -0.05 + 1.29518\,i & 0 \\ 0 & -0.05 - 1.29518\,i \end{pmatrix}$$

Here's a look at

SpannerMatrix.diagonalmatrix.SpannerMatrix^{-1}:

```
MatrixForm[SpannerMatrix.diagonalmatrix.Inverse[SpannerMatrix]]
```

$$\begin{pmatrix} 0.3 & 1.2 \\ -1.5 & -0.4 \end{pmatrix}$$

Here's a look at A:

```
MatrixForm[A]
```

$$\begin{pmatrix} 0.3 & 1.2 \\ -1.5 & -0.4 \end{pmatrix}$$

The upshot:

A = SpannerMatrix.diagonalmatrix.SpannerMatrix^{-1}:

So A is diagonalizable with complex numbers.

To calculate E^{At}, look at::

```
diagonalmatrix = DiagonalMatrix[Eigenvalues[A]];
MatrixForm[diagonalmatrix]
```

$$\begin{pmatrix} -0.05 + 1.29518\,i & 0 \\ 0 & -0.05 - 1.29518\,i \end{pmatrix}$$

Replace this diagonal matrix with the new diagonal matrix you get by hitting the diagonal terms with the function

f[x] = E^{xt}:

```
Clear[f, x, t];
f[x_] = E^(x t);
fdiagonalmatrix = DiagonalMatrix[Map[f, Eigenvalues[A]]];
MatrixForm[fdiagonalmatrix]
```

$$\begin{pmatrix} e^{(-0.05+1.29518\,i)\,t} & 0 \\ 0 & e^{(-0.05-1.29518\,i)\,t} \end{pmatrix}$$

Here is E^{At} in raw form:

```
rawform =
 Simplify[ SpannerMatrix.fdiagonalmatrix.Inverse[SpannerMatrix]]
```

```
{{(0.5 + 0.135116 i) e^((-0.05-1.29518 i) t) + (0.5 - 0.135116 i) e^((-0.05+1.29518 i) t),
  0.463255 i e^((-0.05-1.29518 i) t) - 0.463255 i e^((-0.05+1.29518 i) t)},
 {-0.579069 i e^((-0.05-1.29518 i) t) + 0.579069 i e^((-0.05+1.29518 i) t),
  (0.5 - 0.135116 i) e^((-0.05-1.29518 i) t) +
   (0.5 + 0.135116 i) e^((-0.05+1.29518 i) t)}}
```

Check with *Mathematica*'s built-in instruction:

```
MatrixExp[A t]
```

```
{{(0.5 + 0.135116 i) e^((-0.05-1.29518 i) t) + (0.5 - 0.135116 i) e^((-0.05+1.29518 i) t),
  0.463255 i e^((-0.05-1.29518 i) t) - 0.463255 i e^((-0.05+1.29518 i) t)},
 {-0.579069 i e^((-0.05-1.29518 i) t) + 0.579069 i e^((-0.05+1.29518 i) t),
  (0.5 - 0.135116 i) e^((-0.05-1.29518 i) t) +
   (0.5 + 0.135116 i) e^((-0.05+1.29518 i) t)}}
```

The calculation checks.

That's sort of a mess.

Here's a simplified version of E^{At}:

```
cleanversion = ComplexExpand[
  SpannerMatrix.fdiagonalmatrix.Inverse[SpannerMatrix]]
```

```
{{1. e^(-0.05 t) Cos[1.29518 t] + 0.270232 e^(-0.05 t) Sin[1.29518 t],
  0.92651 e^(-0.05 t) Sin[1.29518 t]}, {-1.15814 e^(-0.05 t) Sin[1.29518 t],
  1. e^(-0.05 t) Cos[1.29518 t] - 0.270232 e^(-0.05 t) Sin[1.29518 t]}}
```

That's still quite a pill. But notice that

$i = \sqrt{-1}$

has disappeared.

If you want to know where the Sines and Cosines come from, go on to the next part.

B.7.b.ii) Euler's Formula: $E^{(a+i\,b)\,t} = E^{a\,t}(\text{Cos}[b\,t] + i\,\text{Sin}[b\,t])$

Keep everything from part i) immediately above alive.
When E^{At} was calculated the first time the result was:

```
rawform =
 Simplify[ SpannerMatrix.fdiagonalmatrix.Inverse[SpannerMatrix]]
```

```
{{(0.5 + 0.135116 i) e^((-0.05-1.29518 i) t) + (0.5 - 0.135116 i) e^((-0.05+1.29518 i) t),
  0.463255 i e^((-0.05-1.29518 i) t) - 0.463255 i e^((-0.05+1.29518 i) t)},
 {-0.579069 i e^((-0.05-1.29518 i) t) + 0.579069 i e^((-0.05+1.29518 i) t),
  (0.5 - 0.135116 i) e^((-0.05-1.29518 i) t) +
   (0.5 + 0.135116 i) e^((-0.05+1.29518 i) t)}}
```

Then a cleaner version of E^{At} without $i = \sqrt{-1}$ was calculated:

```
cleanversion = ComplexExpand[
  SpannerMatrix.fdiagonalmatrix.Inverse[SpannerMatrix]]
```

```
{{1. e^(-0.05 t) Cos[1.29518 t] + 0.270232 e^(-0.05 t) Sin[1.29518 t],
  0.92651 e^(-0.05 t) Sin[1.29518 t]}, {-1.15814 e^(-0.05 t) Sin[1.29518 t],
  1. e^(-0.05 t) Cos[1.29518 t] - 0.270232 e^(-0.05 t) Sin[1.29518 t]}}
```

How was the cleaner version derived from the original version?

Answer:

Mathematica took the original version and replaced all terms of the form

$$: E^{(a+i\,b)\,t}$$

with the equivalent form

$$E^{a\,t}(\text{Cos}[b\,t] + i\,\text{Sin}[b\,t])$$

This equivalence is so important that it's even programmed into *Mathematica*:

```
Clear[a, b, t];
ComplexExpand[E^((a+i b) t)]
```

```
e^(a t) Cos[b t] + i e^(a t) Sin[b t]
```

And it even has a name: "Euler's formula."

If you've never seen this equivalence before and want more on it, click on the right.

This is from MEI's Calculus&*Mathematica* course.

The complex exponential and the Euler's formula

Background

Look at these partial expansions of
Cos[x], Sin[x] and E^x
in powers of x:

```
Clear[x]
cosexpansion = Normal[Series[Cos[x], {x, 0, 8}]]
```

$$1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \frac{x^8}{40320}$$

```
sinexpansion = Normal[Series[Sin[x], {x, 0, 8}]]
```

$$x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040}$$

```
eexpansion = Normal[Series[E^x, {x, 0, 8}]]
```

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \frac{x^7}{5040} + \frac{x^8}{40320}$$

Notice that the terms in the expansion of
E^x
seem to have been lifted from the expansions of
Cos[x] and Sin[x].
Complex numbers explain this phenom.
Here's how.

Activate the complex (imaginary) number $i = \sqrt{-1}$:

```
i
```

$$i$$

```
i^2
```

$$-1$$

Now replace x by i x in the expansion of E^x:

```
Normal[Series[E^x, {x, 0, 10}]] /. x → i x
```

$$1 + i x - \frac{x^2}{2} - \frac{i x^3}{6} + \frac{x^4}{24} + \frac{i x^5}{120} - \frac{x^6}{720} - \frac{i x^7}{5040} + \frac{x^8}{40320} + \frac{i x^9}{362880} - \frac{x^{10}}{3628800}$$

Embedded in this is the expansion of Cos[x]:

```
Normal[Series[Cos[x], {x, 0, 10}]]
```

$$1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \frac{x^8}{40320} - \frac{x^{10}}{3628800}$$

You can also see i times the expansion of Sin[x]:

```
Expand[i Normal[Series[Sin[x], {x, 0, 10}]]]
```

$$i x - \frac{i x^3}{6} + \frac{i x^5}{120} - \frac{i x^7}{5040} + \frac{i x^9}{362880}$$

In fact:

```
k = Random[Integer, {15, 30}];
Normal[Series[E^x, {x, 0, k}]] /. x → i x
```

$$1 + i x - \frac{x^2}{2} - \frac{i x^3}{6} + \frac{x^4}{24} + \frac{i x^5}{120} - \frac{x^6}{720} - \frac{i x^7}{5040} + \frac{x^8}{40320} + \frac{i x^9}{362880} - \frac{x^{10}}{3628800} - \frac{i x^{11}}{39916800} + \frac{x^{12}}{479001600} + \frac{i x^{13}}{6227020800} - \frac{x^{14}}{87178291200} - \frac{i x^{15}}{1307674368000} + \frac{x^{16}}{20922789888000} + \frac{i x^{17}}{355687428096000} - \frac{x^{18}}{6402373705728000} - \frac{i x^{19}}{121645100408832000} + \frac{x^{20}}{2432902008176640000} + \frac{i x^{21}}{51090942171709440000} - \frac{x^{22}}{1124000727777607680000}$$

```
Normal[Series[Cos[x], {x, 0, k}]] +
 Expand[i Normal[Series[Sin[x], {x, 0, k}]]]
```

$$1 + i x - \frac{x^2}{2} - \frac{i x^3}{6} + \frac{x^4}{24} + \frac{i x^5}{120} - \frac{x^6}{720} - \frac{i x^7}{5040} + \frac{x^8}{40320} + \frac{i x^9}{362880} - \frac{x^{10}}{3628800} - \frac{i x^{11}}{39916800} + \frac{x^{12}}{479001600} + \frac{i x^{13}}{6227020800} - \frac{x^{14}}{87178291200} - \frac{i x^{15}}{1307674368000} + \frac{x^{16}}{20922789888000} + \frac{i x^{17}}{355687428096000} - \frac{x^{18}}{6402373705728000} - \frac{i x^{19}}{121645100408832000} + \frac{x^{20}}{2432902008176640000} + \frac{i x^{21}}{51090942171709440000} - \frac{x^{22}}{1124000727777607680000}$$

That's why folks all agree that

$$E^{i x} = \text{Cos}[x] + i\ \text{Sin}[x]$$

□B.7.b.iii) Eigenvalues $p + q\sqrt{-1}$ and $p - q\sqrt{-1}$ with $p < 0$ and $q \neq 0$ signal that $E^{A t}.X$ swirls toward {0, 0}

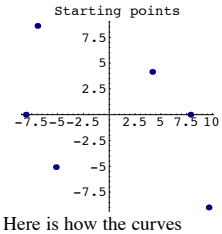
Go with this 2D matrix A:

```
A = {{-0.5, -1.1}, {1.6, -0.8}};
MatrixForm[A]
```

$$\begin{pmatrix} -0.5 & -1.1 \\ 1.6 & -0.8 \end{pmatrix}$$

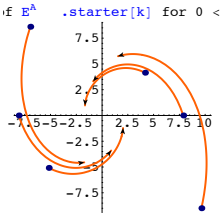
Here are six random points
{starter[1], starter[2], starter[3], starter[4], starter[5], starter[6]}
in 2D :

```
Clear[pointer, trajectoryplot, eigenplot,
  scaler, eigenvector, starter, thigh, k, j];

trajectoryplot[thigh_, starter_] :=
  ParametricPlot[MatrixExp[A t].starter, {t, 0, thigh},
   PlotStyle -> {{Thickness[0.01], CadmiumOrange}},
   DisplayFunction -> Identity];


pointer[thigh_, starter_] :=
  ArrowHead[MatrixExp[A thigh] .starter,
   (D[MatrixExp[A t], t] /. t -> thigh) .starter,
                HeadSize -> 0.6, VectorColor -> Black,
   Aperture -> 0.4];

starterplots = Table[Graphics[
    {NavyBlue, PointSize[0.03], Point[starter[j]]}], {j, 1, 6}];

starter[1] = {Random[Real, {4, 10}], Random[Real, {4, 10}]};
starter[2] = {Random[Real, {-10, -4}], Random[Real, {-10, -4}]};
starter[3] = {Random[Real, {-10, -4}], Random[Real, {4, 10}]};
starter[4] = {Random[Real, {4, 10}], Random[Real, {-10, -4}]};
starter[5] = {Random[Real, {-10, -4}], 0};
starter[6] = {Random[Real, {4, 10}], 0};

thigh = 0.01;
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
  starterplots, PlotLabel -> "Starting points",
  DisplayFunction -> $DisplayFunction];
```



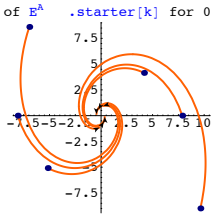
Here is how the curves
$E^{A t}$.starter[1], $E^{A t}$.starter[2], $E^{A t}$.starter[3],
$E^{A t}$.starter[4], $E^{A t}$.starter[5], $E^{A t}$.starter[6]
plot out as t advances from 0 to 1.5

```
thigh = 1.5;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



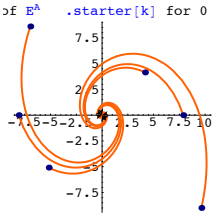
Swirling.
Here's what happens as t advances from 0 to 4:

```
thigh = 4;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



Here's what happens as t advances from 0 to 10:

```
thigh = 10;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



Grab the plots, align and animate.

Eventually all the plots swirl to {0,0} and stall at {0,0}.
Now look at the eigenvalues of A:

```
Eigenvalues[A]
```

$$\{-0.65 + 1.31814\ i,\ -0.65 - 1.31814\ i\}$$

Negative real part, nonzero imaginary part.

Explain this bold statement:
If A is any 2D matrix that is diagonalizable matrix with complex numbers and its complex eigenvalues are of the form

$p + I\,q$ and $p - I\,q$ with $p < 0$ and $q \neq 0$,
then no matter what X is, you are guaranteed that the plot of
$E^{A\,t}.X$
swirls to {0,0} and stalls at {0,0}.

□ **Answer:**

Arrange the complex eigenvectors of A into the SpannerMatrix with the eigenvector corresponding to p + I q in the leftmost vertical column and the eigenvector corresponding to p - I q in the second vertical column of SpannerMatrix.

So that

$$A = \text{SpannerMatrix}.\begin{pmatrix} p + I\,q & 0 \\ 0 & p - I\,q \end{pmatrix}.\text{SpannerMatrix}^{-1}.$$

Use shorthand notation by putting

SpannerMatrix = SM

so that

$$A = SM.\begin{pmatrix} p + I\,q & 0 \\ 0 & p - I\,q \end{pmatrix}.SM^{-1}$$

This gives

$$E^{A\,t} = SM.\begin{pmatrix} E^{(p+I\,q)\,t} & 0 \\ 0 & E^{(p-I\,q)\,t} \end{pmatrix}.SM^{-1}$$

And

$$E^{A\,t}.X = SM.\begin{pmatrix} E^{(p+I\,q)\,t} & 0 \\ 0 & E^{(p-I\,q)\,t} \end{pmatrix}.SM^{-1}.X.$$

This is the same as

$$E^{A\,t}.X = SM.\begin{pmatrix} E^{pt} & 0 \\ 0 & E^{pt} \end{pmatrix}\begin{pmatrix} E^{Iqt} & 0 \\ 0 & E^{-Iqt} \end{pmatrix}.SM^{-1}.X$$

Because $p < 0$, you know that $E^{p\,t} \to 0$ as t gets large. The result:

$$E^{A\,t}.X \to SM.\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.\begin{pmatrix} E^{Iqt} & 0 \\ 0 & E^{-Iqt} \end{pmatrix}.SM^{-1}.X = \{0, 0\}$$

as t gets large.

This explains why the plots head to {0,0} and stall there.

To see why the plots swirl, look at

$$E^{A\,t}.X = SM.\begin{pmatrix} E^{pt} & 0 \\ 0 & E^{pt} \end{pmatrix}\begin{pmatrix} E^{Iqt} & 0 \\ 0 & E^{-Iqt} \end{pmatrix}.SM^{-1}.X$$

and remember that

$E^{I\,q\,t} = \text{Cos}[q\,t] + I\,\text{Sin}[q\,t]$ and $E^{-I\,q\,t} = \text{Cos}[q\,t] - I\,\text{Sin}[q\,t]$.

The presence of the oscillatory functions

Sin[q t] and Cos[q t]

explains why the plots swirl.

In fact the bigger q is, the more vivid the swirl.

Reason: The bigger q is, the faster Sin[q t] and Cos[q t] oscillate.

□ B.7.b.iv) Eigenvalues $p + q\sqrt{-1}$ and $p - q\sqrt{-1}$ with $p > 0$ and $q \neq 0$ signal that $E^{A\,t}.X$ swirls away from {0, 0}

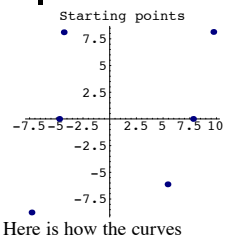
Go with this new 2D matrix A:

```
A = {{0.5, 1.2}, {-1.5, 0.8}};
MatrixForm[A]
```

$$\begin{pmatrix} 0.5 & 1.2 \\ -1.5 & 0.8 \end{pmatrix}$$

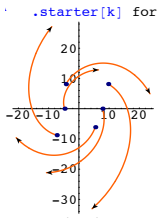
Here are six random points
{starter[1], starter[2], starter[3], starter[4], starter[5], starter[6]}
in 2D :

```
Clear[trajectoryplot, eigenplot, scaler,
  pointer, eigenvector, starter, thigh, k, headsize, j];

trajectoryplot[thigh_, starter_] :=
  ParametricPlot[MatrixExp[A t].starter, {t, 0, thigh},
   PlotStyle -> {{Thickness[0.01], CadmiumOrange}},
   DisplayFunction -> Identity];


pointer[thigh_, starter_] :=
  ArrowHead[MatrixExp[A thigh].starter,
   (D[MatrixExp[A t], t] /. t -> thigh).starter,
              HeadSize -> headsize,
   VectorColor -> Black, Aperture -> 0.4];

starterplots = Table[Graphics[
    {NavyBlue, PointSize[0.03], Point[starter[j]]}], {j, 1, 6}];

starter[1] = {Random[Real, {4, 10}], Random[Real, {4, 10}]};
starter[2] = {Random[Real, {-10, -4}], Random[Real, {-10, -4}]};
starter[3] = {Random[Real, {-10, -4}], Random[Real, {4, 10}]};
starter[4] = {Random[Real, {4, 10}], Random[Real, {-10, -4}]};
starter[5] = {Random[Real, {-10, -4}], 0};
starter[6] = {Random[Real, {4, 10}], 0};

thigh = 0.01;
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
  starterplots, PlotLabel -> "Starting points",
  DisplayFunction -> $DisplayFunction];
```



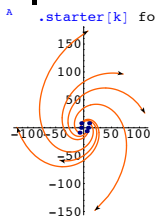
Here is how the curves
$E^{A\,t}.\text{starter}[1]$, $E^{A\,t}.\text{starter}[2]$, $E^{A\,t}.\text{starter}[3]$,
$E^{A\,t}.\text{starter}[4]$, $E^{A\,t}.\text{starter}[5]$, $E^{A\,t}.\text{starter}[6]$
plot out as t advances from 0 to 1.5

```
thigh = 1.5;
headsize = 2;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



Here's what happens as t advances from 0 to 4:

```
thigh = 4;
headsize = 10;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^(A\t).starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



Eventually all the plots swirl away from to {0,0} .
Now look at the eigenvalues of A:

```
Eigenvalues[A]
```

{0.65 + 1.33323 i, 0.65 - 1.33323 i}

Positive real part, nonzero imaginary part.
Explain this statement:

If A is any 2D matrix that is diagonalizable matrix with complex numbers and its complex eigenvalues are of the form
$p + I\,q$ and $p - I\,q$ with $p > 0$ and $q \neq 0$,
then no matter what X is, you are guaranteed that the plot of
$E^{A\,t}.X$
swirls away from {0,0}..

□ **Answer:**

Arrange the complex eigenvectors of A into the SpannerMatrix with the eigenvector corresponding to p + I q in the leftmost vertical column and the eigenvector corresponding to p - I q in the second vertical column of SpannerMatrix.

So that

$$A = \text{SpannerMatrix}.\begin{pmatrix} p + I q & 0 \\ 0 & p - I q \end{pmatrix}.\text{SpannerMatrix}^{-1}.$$

Use shorthand notation by putting

SpannerMatrix = SM

so that

$$A = SM.\begin{pmatrix} p + I q & 0 \\ 0 & p - I q \end{pmatrix}.SM^{-1}$$

This gives

$$E^{A t} = SM.\begin{pmatrix} E^{(p+I q) t} & 0 \\ 0 & E^{(p-I q) t} \end{pmatrix}.SM^{-1}$$

And

$$E^{A t}.X = SM.\begin{pmatrix} E^{(p+I q) t} & 0 \\ 0 & E^{(p-I q) t} \end{pmatrix}.SM^{-1}.X.$$

This is the same as

$$E^{A t}.X = SM.\begin{pmatrix} E^{pt} & 0 \\ 0 & E^{pt} \end{pmatrix}\begin{pmatrix} E^{Iqt} & 0 \\ 0 & E^{-Iqt} \end{pmatrix}.SM^{-1}.X$$

Because $p > 0$, you know that $E^{pt} \to \infty$ as t gets large. The result:

$$E^{A t}.X \to SM.\begin{pmatrix} \infty & 0 \\ 0 & \infty \end{pmatrix}.\begin{pmatrix} E^{Iqt} & 0 \\ 0 & E^{-Iqt} \end{pmatrix}.SM^{-1}.X$$

as t gets large. So the plots try to leave the screen.

To see why the plots swirl, look at

$$E^{A t}.X = SM.\begin{pmatrix} E^{pt} & 0 \\ 0 & E^{pt} \end{pmatrix}\begin{pmatrix} E^{Iqt} & 0 \\ 0 & E^{-Iqt} \end{pmatrix}.SM^{-1}.X$$

and remember that

$E^{I q t} = \text{Cos}[q t] + I\,\text{Sin}[q t]$ and $E^{-I q t} = \text{Cos}[q t] - I\,\text{Sin}[q t]$.

The presence of the oscillatory functions

Sin[q t] and Cos[q t]

explains why the plots swirl.

In fact the bigger q is, the more vivid the swirl.

Reason: The bigger q is, the faster Sin[q t] and Cos[q t] oscillate.

□B.7.b.v) Eigenvalues $p + q\sqrt{-1}$ and $p - q\sqrt{-1}$ with p = 0 and q ≠ 0 signal that $E^{A t}.X$ oscillates around {0, 0} on closed curves

Go with this new 2D matrix A:

```
A = {{-0.2, 1.2}, {-1.5, 0.2}};
MatrixForm[A]
```

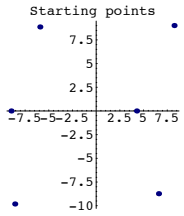
$$\begin{pmatrix} -0.2 & 1.2 \\ -1.5 & 0.2 \end{pmatrix}$$

Here are six random points

{starter[1], starter[2], starter[3], starter[4], starter[5], starter[6]}

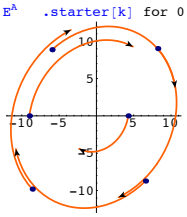
in 2D :

```
Clear[trajectoryplot, eigenplot, scaler,
  pointer, eigenvector, starter, thigh, k, headsize, j];

trajectoryplot[thigh_, starter_] :=
  ParametricPlot[MatrixExp[A t].starter, {t, 0, thigh},
   PlotStyle -> {{Thickness[0.01], CadmiumOrange}},
   DisplayFunction -> Identity];


pointer[thigh_, starter_] :=
  ArrowHead[MatrixExp[A thigh] .starter,
   (D[MatrixExp[A t], t] /. t -> thigh) .starter,
                HeadSize -> 1, VectorColor -> Black, Aperture -> 0.4];

starterplots = Table[Graphics[
    {NavyBlue, PointSize[0.03], Point[starter[j]]}], {j, 1, 6}];

starter[1] = {Random[Real, {4, 10}], Random[Real, {4, 10}]};
starter[2] = {Random[Real, {-10, -4}], Random[Real, {-10, -4}]};
starter[3] = {Random[Real, {-10, -4}], Random[Real, {4, 10}]};
starter[4] = {Random[Real, {4, 10}], Random[Real, {-10, -4}]};
starter[5] = {Random[Real, {-10, -4}], 0};
starter[6] = {Random[Real, {4, 10}], 0};

thigh = 0.01;
Show[Table[trajectoryplot[thigh, starter[j]], {j, 1, 6}] ,
  starterplots, PlotLabel -> "Starting points",
  DisplayFunction -> $DisplayFunction];
```



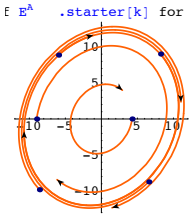
Here is how the curves
$E^{A t}$.starter[1], $E^{A t}$.starter[2], $E^{A t}$.starter[3],
$E^{A t}$.starter[4], $E^{A t}$.starter[5], $E^{A t}$.starter[6]
plot out as t advances from 0 to 1.5

```
thigh = 1.5;
headsize = 2;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^A\t.starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



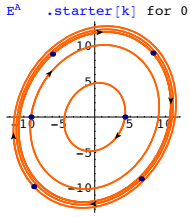
Here's what happens as t advances from 0 to 4:

```
thigh = 4;
headsize = 2;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^A\t.starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



Here's what happens as t advances from 0 to 10:

```
thigh = 10;
headsize = 2;
Show[Table[trajectoryplot[thigh, starter[k]], {k, 1, 6}] ,
    Table[pointer[thigh, starter[k]], {k, 1, 6}],
 starterplots, PlotRange -> All,
 PlotLabel -> "Plots of E^A\t.starter[k] for 0 < t" < thigh,
 DisplayFunction -> $DisplayFunction];
```



The plots oscillate on closed curves centered at {0,0} .
Now look at the eigenvalues of A:

```
Eigenvalues[A]
```

{1.32665 i, -1.32665 i}

Zero real part, nonzero imaginary part.
Explain this statement:

If A is any 2D matrix that is diagonalizable matrix with complex numbers and its complex

eigenvalues are of the form

$p + I\,q$ and $p - I\,q$ with $\mathbf{p = 0}$ and $q \neq 0$,

then no matter what X is, you are guaranteed that the plot of

$E^{A\,t}.X$

oscillate on closed curves centered at {0,0} .

□ **Answer:**

Arrange the complex eigenvectors of A into the SpannerMatrix with the eigenvector corresponding to $I\,q$ in the leftmost vertical column and the eigenvector corresponding to $-I\,q$ in the second vertical column of SpannerMatrix.

So that

$$A = \text{SpannerMatrix}.\begin{pmatrix} I\,q & 0 \\ 0 & -I\,q \end{pmatrix}.\text{SpannerMatrix}^{-1}.$$

Use shorthand notation by putting

SpannerMatrix = SM

so that

$$A = SM.\begin{pmatrix} i\,q & 0 \\ 0 & -i\,q \end{pmatrix}.SM^{-1}$$

This gives

$$E^{A\,t} = SM.\begin{pmatrix} E^{i\,q\,t} & 0 \\ 0 & E^{-i\,q\,t} \end{pmatrix}.SM^{-1}$$

And

$$E^{A\,t}.X = SM.\begin{pmatrix} E^{i\,q\,t} & 0 \\ 0 & E^{-i\,q\,t} \end{pmatrix}.SM^{-1}.X.$$

This is the same as

$$E^{A\,t}.X = SM.\begin{pmatrix} \text{Cos}[q\,t] + i\,\text{Sin}[q\,t] & 0 \\ 0 & \text{Cos}[q\,t] - i\,\text{Sin}[q\,t] \end{pmatrix}.SM^{-1}.X.$$

To see why the oscillate on closed curves centered at {0,0} ,remember that

$E^{I\,q\,t} = \text{Cos}[q\,t] + I\,\text{Sin}[q\,t]$ and $E^{-I\,q\,t} = \text{Cos}[q\,t] - I\,\text{Sin}[q\,t]$.

The presence of the periodic oscillatory functions

Sin[q t] and Cos[q t]

explains why the plots repeat themselves over and over on closed curves centered at {0,0}.

In fact the bigger q is, the faster the oscillation.

Reason: The bigger q is, the faster Sin[q t] and Cos[q t] oscillate.