Matrices, Geometry&*Mathematica*

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

Producer: Bruce Carpenter

Publisher: Math Everywhere, Inc.

MGM.05 3D Matrices

BASICS

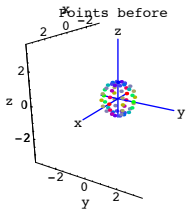
B.1) SVD analysis in 3D: Writing a 3D matrix as the product of a hanger, a stretcher and an aligner

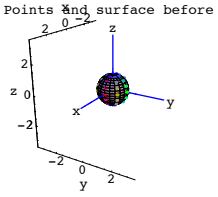
□B.1.a.i) Hits with 3D matrices

Here's the 3D unit sphere with specially marked points waiting to be hit with a 3D matrix:

This 3D sphere is plotted using spherical coordinates.
For a review of them, see C&M lesson 3.09.

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

{slow, shigh} = {0, π};
{tlow, thigh} = {0, 2 π};

ranger = 3.5;
pointcolor[s_, t_] := RGBColor[
   Normalize [{Abs[x[s, t]], Abs[y[s, t]], Abs[z[s, t]]}][[1]] ,
   Normalize[{Abs[x[s, t]], Abs[y[s, t]], Abs[z[s, t]]}][[2]],
   Normalize[{Abs[x[s, t]], Abs[y[s, t]], Abs[z[s, t]]}][[3]]];

sjump = (shigh - slow)/8;
tjump = (thigh - tlow)/8;

Clear[hitplotter, hitpointplotter, matrix3D];
hitplotter[matrix3D_] :=
  ParametricPlot3D[matrix3D.{x[s, t], y[s, t], z[s, t]},
   {s, slow, shigh}, {t, tlow, thigh}, PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False,
   ViewPoint → CMView, DisplayFunction → Identity];

hitpointplotter[matrix3D_] :=
  Show[Table[Graphics3D[{pointcolor[s, t], PointSize[0.02],
      Point[matrix3D.{x[s, t], y[s, t], z[s, t]}]}],
    {s, slow, shigh - sjump, sjump}, {t, tlow, thigh - tjump, tjump}],
   PlotRange → {{-ranger, ranger}, {-ranger, ranger},
     {-ranger, ranger}}, Axes → True, AxesLabel → {"x", "y", "z"},
   Boxed → False, ViewPoint → CMView, DisplayFunction → Identity];

pointsbefore =
  Show[hitpointplotter[IdentityMatrix[3]], Axes3D[ranger],
   PlotLabel → "Points before", DisplayFunction → $DisplayFunction];

surfacebefore = Show[hitplotter[IdentityMatrix[3]],
   hitpointplotter[IdentityMatrix[3]], Axes3D[ranger],
   PlotLabel → "Points and surface before",
   DisplayFunction → $DisplayFunction];
```

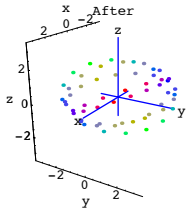


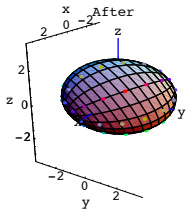


Grab both plots and animate at various speeds.

Here's what happens when you hit the everything with the 3D matrix A defined in the code below:

```
A = {{2.4, 0.5, -1.7}, {1.8, -1.6, 2.3}, {1.5, 1.9, 0.9}};

pointsafter = Show[hitpointplotter[A], Axes3D[ranger],
  PlotLabel → "After", DisplayFunction → $DisplayFunction];

surfaceafter =
 Show[hitplotter[A], hitpointplotter[A], Axes3D[ranger],
  PlotLabel → "After", DisplayFunction → $DisplayFunction];
```





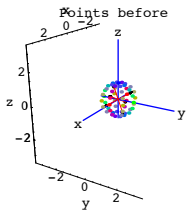
Grab all four plots and animate at various speeds.

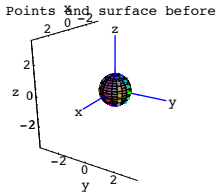
Now look at this:

```
Clear[alignerframe, k];
alignerframe[k_] := SingularValues[A][[3, k]];
Table[alignerframe[k], {k, 1, 3}]
```

```
{{0.878673, -0.0560142, 0.474126},
 {0.412109, 0.590374, -0.693992}, {0.241038, -0.805184, -0.54183}}
```

```
alignerplot =
  {Table[Arrow[alignerframe[k], Tail → {0, 0, 0}, VectorColor → Red],
    {k, 1, 3}], Table[Arrow[-alignerframe[k],
     Tail → {0, 0, 0}, VectorColor → Red], {k, 1, 3}]};

Show[pointsbefore, alignerplot];
Show[surfacebefore, alignerplot];
```

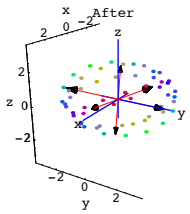


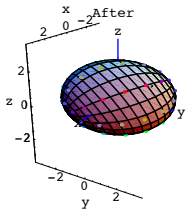


Grab both plots and animate at various speeds.

Hit the plotted frame, the points and the unit sphere with A and plot the result:

```
hangerplot = {Table[Arrow[A.alignerframe[k],
     Tail → {0, 0, 0}, VectorColor → Red], {k, 1, 3}],
   Table[Arrow[-A.alignerframe[k], Tail → {0, 0, 0},
     VectorColor → Red], {k, 1, 3}]};

Show[pointsafter, hangerplot];
Show[surfaceafter, hangerplot];
```





Grab both plots and animate at various speeds.
Then grab all four plots and animate slowly.

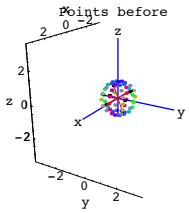
Describe what you see and try to explain why you see it.

□ Answer:

What worked so well in 2D also works pretty darn well in 3D.

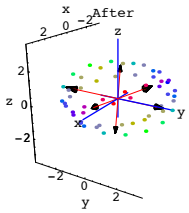
The strong evidence is that a hit with A picks up points and surfaces on the following perpendicular frame:

```
Show[pointsbefore, alignerplot];
```



And then A stretches and hangs points, curves and surfaces on this hangerframe:

```
Show[pointsafter, hangerplot];
```



The upshot:

Just as in 2D, you can do an SVD analysis of this 3D matrix A by writing

A = hanger . stretcher . aligner

as defined in the code below:

```
hanger = Transpose[SingularValues[A][[1]];
MatrixForm[hanger]
```

$$\begin{pmatrix} 0.368983 & 0.758554 & 0.537073 \\ 0.799368 & -0.553818 & 0.233017 \\ 0.474197 & 0.343339 & -0.810713 \end{pmatrix}$$

To get the stretcher matrix for A, you do this:

```
{xstretch, ystretch, zstretch} = SingularValues[A][[2]];
stretcher = ( xstretch 0 0
              0 ystretch 0
              0 0 zstretch );
MatrixForm[stretcher]
```

$$\begin{pmatrix} 3.45489 & 0 & 0 \\ 0 & 3.24833 & 0 \\ 0 & 0 & 2.04257 \end{pmatrix}$$

To get the aligner matrix for A you do this:

```
aligner = SingularValues[A][[3]];;
MatrixForm[aligner]
```

$$\begin{pmatrix} 0.878673 & -0.0560142 & 0.474126 \\ 0.412109 & 0.590374 & -0.693992 \\ 0.241038 & -0.805184 & -0.54183 \end{pmatrix}$$

Check it out:

```
MatrixForm[A]
MatrixForm[hanger.stretcher.aligner]
```

$$\begin{pmatrix} 2.4 & 0.5 & -1.7 \\ 1.8 & -1.6 & 2.3 \\ 1.5 & 1.9 & 0.9 \end{pmatrix}$$

$$\begin{pmatrix} 2.4 & 0.5 & -1.7 \\ 1.8 & -1.6 & 2.3 \\ 1.5 & 1.9 & 0.9 \end{pmatrix}$$

```
Clear[hangerframe, k];
hangerframe[k_] := SingularValues[A][[1, k]];
A.alignerframe[1] == xstretch hangerframe[1]
A.alignerframe[2] == ystretch hangerframe[2]
A.alignerframe[3] == zstretch hangerframe[3]
```

```
True
```

```
True
```

Everything checks.

Try the same thing with some random 3D matrices A:

```
A =
 ( Random[Real, {-4, 4}] Random[Real, {-4, 4}] Random[Real, {-4, 4}]
   Random[Real, {-4, 4}] Random[Real, {-4, 4}] Random[Real, {-4, 4}]
   Random[Real, {-4, 4}] Random[Real, {-4, 4}] Random[Real, {-4, 4}] );
MatrixForm[A]
```

$$\begin{pmatrix} -1.05155 & 2.20508 & 0.78952 \\ 0.955304 & -1.75508 & 0.161777 \\ 1.70016 & 3.24669 & 0.676248 \end{pmatrix}$$

```
hanger = Transpose[SingularValues[A][[1]];
MatrixForm[hanger]
```

$$\begin{pmatrix} 0.497179 & 0.58995 & 0.636217 \\ -0.357368 & -0.528942 & 0.769746 \\ 0.790634 & -0.610065 & -0.0521495 \end{pmatrix}$$

```
{xstretch, ystretch, zstretch} = SingularValues[A][[2]];
stretcher = ( xstretch 0 0
              0 ystretch 0
              0 0 zstretch );
MatrixForm[stretcher]
```

$$\begin{pmatrix} 4.4039 & 0 & 0 \\ 0 & 2.17734 & 0 \\ 0 & 0 & 0.603511 \end{pmatrix}$$

```
aligner = SingularValues[A][[3]];;
MatrixForm[aligner]
```

$$\begin{pmatrix} 0.108995 & 0.974243 & 0.197412 \\ -0.993353 & 0.114144 & -0.0148567 \\ -0.0370074 & -0.19448 & 0.980208 \end{pmatrix}$$

```
MatrixForm[A]
MatrixForm[hanger.stretcher.aligner]
```

$$\begin{pmatrix} -1.05155 & 2.20508 & 0.78952 \\ 0.955304 & -1.75508 & 0.161777 \\ 1.70016 & 3.24669 & 0.676248 \end{pmatrix}$$

$$\begin{pmatrix} -1.05155 & 2.20508 & 0.78952 \\ 0.955304 & -1.75508 & 0.161777 \\ 1.70016 & 3.24669 & 0.676248 \end{pmatrix}$$

```
Clear[hangerframe, k];
hangerframe[k_] := SingularValues[A][[1, k]];
A.alignerframe[1]
```

```
{2.18953, -1.57381, 3.48187}
```

```
xstretch hangerframe[1]
```

```
{2.18953, -1.57381, 3.48187}
```

```
A.alignerframe[2]
```

```
{1.28452, -1.15169, -1.32832}
```

```
ystretch hangerframe[2]
```

```
{1.28452, -1.15169, -1.32832}
```

```
A.alignerframe[3]
```

```
{0.383964, 0.46455, -0.0314728}
```

```
zstretch hangerframe[3]
```

```
{0.383964, 0.46455, -0.0314728}
```

Rerun until you get tired.

□ B.1.a.ii) Milking the SVD instruction for 3D matrices

Discuss how to use *Mathematica*'s SVD instruction for 3D matrices

□ Answer:

Here's how to use it on a randomly generated 3D matrix:

```
A =
 ( Random[Real, {-4, 4}] Random[Real, {-4, 4}] Random[Real, {-4, 4}]
   Random[Real, {-4, 4}] Random[Real, {-4, 4}] Random[Real, {-4, 4}]
   Random[Real, {-4, 4}] Random[Real, {-4, 4}] Random[Real, {-4, 4}] );
MatrixForm[A]
```

$$\begin{pmatrix} 2.46205 & 3.72686 & -2.98942 \\ -3.89042 & -3.21621 & 3.89705 \\ -0.502626 & -2.48997 & -1.00435 \end{pmatrix}$$

To get the hanger matrix for A, you do this:

```
hanger = Transpose[SingularValues[A][[1]];
MatrixForm[hanger]
```

$$\begin{pmatrix} -0.638283 & 0.148443 & 0.755354 \\ 0.755072 & 0.311763 & 0.576776 \\ 0.149873 & -0.938493 & 0.311078 \end{pmatrix}$$

To get the stretcher matrix for A, you do this:

```
stretcher = DiagonalMatrix[SingularValues[A][[2]]];
MatrixForm[stretcher]
```

$$\begin{pmatrix} 8.36328 & 0 & 0 \\ 0 & 2.57687 & 0 \\ 0 & 0 & 0.656337 \end{pmatrix}$$

To get the aligner matrix for A you do this:

```
aligner = SingularValues[A][[3]];
MatrixForm[aligner]
```

$$\begin{pmatrix} -0.548154 & -0.619427 & 0.561995 \\ -0.145799 & 0.732419 & 0.66506 \\ -0.823572 & 0.282617 & -0.49179 \end{pmatrix}$$

Check it out:

```
MatrixForm[A]
MatrixForm[hanger.stretcher.aligner]
```

$$\begin{pmatrix} 2.46205 & 3.72686 & -2.98942 \\ -3.89042 & -3.21621 & 3.89705 \\ -0.502626 & -2.48997 & -1.00435 \end{pmatrix}$$

$$\begin{pmatrix} 2.46205 & 3.72686 & -2.98942 \\ -3.89042 & -3.21621 & 3.89705 \\ -0.502626 & -2.48997 & -1.00435 \end{pmatrix}$$

Yep.

If you want the individual vectors in the aligner frame, do this:

```
Clear[alignerframe, k];
alignerframe[k_] := SingularValues[A][[3, k]];
Table[alignerframe[k], {k, 1, 3}]
```

```
{{-0.548154, -0.619427, 0.561995},
 {-0.145799, 0.732419, 0.66506}, {-0.823572, 0.282617, -0.49179}}
```

If you want {xstretch,ystretch,zstretch}, here they are:

```
{xstretch, ystretch, zstretch} = SingularValues[A][[2]]
```

```
{8.36328, 2.57687, 0.656337}
```

If you want the individual vectors in the hanger frame, do this:

```
Clear[hangerframe, k];
hangerframe[k_] := SingularValues[A][[1, k]];
Table[hangerframe[k], {k, 1, 3}]
```

```
{{-0.638283, 0.755072, 0.149873},
 {0.148443, 0.311763, -0.938493}, {0.755354, 0.576776, 0.311078}}
```

Check them out:

```
MatrixForm[A]
```

$$\begin{pmatrix} 2.46205 & 3.72686 & -2.98942 \\ -3.89042 & -3.21621 & 3.89705 \\ -0.502626 & -2.48997 & -1.00435 \end{pmatrix}$$

```
MatrixForm[hanger.stretcher.aligner]
```

$$\begin{pmatrix} 2.46205 & 3.72686 & -2.98942 \\ -3.89042 & -3.21621 & 3.89705 \\ -0.502626 & -2.48997 & -1.00435 \end{pmatrix}$$

```
{A.alignerframe[1], xstretch hangerframe[1]}
```

```
{{-5.33814, 6.31488, 1.25343}, {-5.33814, 6.31488, 1.25343}}
```

```
{A.alignerframe[2], ystretch hangerframe[2]}
```

```
{{0.382519, 0.803373, -2.41837}, {0.382519, 0.803373, -2.41837}}
```

```
{A.alignerframe[3], zstretch hangerframe[3]}
```

```
{{0.495767, 0.378559, 0.204172}, {0.495767, 0.378559, 0.204172}}
```

Everything checks.

Another way for *Mathematica* to serve you.

□B.1.a.iii) The SVD instruction for 3D matrices - how it works

How does *Mathematica*'s SVD instruction work for 3D matrices?

□Answer:

It's the same as in 2D.

Mathematica tries to find a 3D perpendicular frame

{alignerframe[1], aligerframe[2], alignerframe[3]}

so that

(A.alignerframe[1]).(A.alignerframe[2]) = 0,

(A.alignerframe[1]).(A.alignerframe[3]) = 0,

and

(A.alignerframe[2]).(A.alignerframe[3]) = 0

It's an impossible job by hand, but it's easy for *Mathematica*.

Later in the course you will learn why this is guaranteed to work.

□B.1.a.iv) Descriptions of the actions of the hanger, the stretcher and the aligner for a 3D matrix

Here's a sample 3D matrix A together with the SVD calculations of the aligner the stretcher and the hanger for

$$A = \begin{pmatrix} -1.49 & 0.55 & -0.59 \\ -0.89 & -1.38 & -1.45 \\ 0.42 & -0.02 & -1.03 \end{pmatrix}:$$

```
A = {{-1.49, 0.55, -0.59}, {-0.89, -1.38, -1.45}, {0.42, -0.02, -1.03}};
MatrixForm[A]
```

$$\begin{pmatrix} -1.49 & 0.55 & -0.59 \\ -0.89 & -1.38 & -1.45 \\ 0.42 & -0.02 & -1.03 \end{pmatrix}$$

```
hanger = Transpose[SingularValues[A][[1]]];
MatrixForm[hanger]
```

$$\begin{pmatrix} -0.427099 & 0.868978 & 0.249927 \\ -0.877295 & -0.331313 & -0.347255 \\ -0.218953 & -0.367572 & 0.903853 \end{pmatrix}$$

```
stretcher = DiagonalMatrix[SingularValues[A][[2]]];
MatrixForm[stretcher]
```

$$\begin{pmatrix} 2.40373 & 0 & 0 \\ 0 & 1.52991 & 0 \\ 0 & 0 & 0.888182 \end{pmatrix}$$

```
aligner = SingularValues[A][[3]];
MatrixForm[aligner]
```

$$\begin{pmatrix} 0.551314 & 0.407759 & 0.727864 \\ -0.754483 & 0.616051 & 0.226357 \\ 0.356102 & 0.673954 & -0.647284 \end{pmatrix}$$

Check it out:

```
MatrixForm[A]
MatrixForm[hanger.stretcher.aligner]
```

$$\begin{pmatrix} -1.49 & 0.55 & -0.59 \\ -0.89 & -1.38 & -1.45 \\ 0.42 & -0.02 & -1.03 \end{pmatrix}$$

$$\begin{pmatrix} -1.49 & 0.55 & -0.59 \\ -0.89 & -1.38 & -1.45 \\ 0.42 & -0.02 & -1.03 \end{pmatrix}$$

```
Clear[alignerframe, k];
alignerframe[k_] := SingularValues[A][[3, k]];
Table[alignerframe[k], {k, 1, 3}]
```

```
{{0.551314, 0.407759, 0.727864},
 {-0.754483, 0.616051, 0.226357}, {0.356102, 0.673954, -0.647284}}
```

```
{xstretch, ystretch, zstretch} = SingularValues[A][[2]]
```

```
{2.40373, 1.52991, 0.888182}
```

```
Clear[hangerframe, k];
hangerframe[k_] := SingularValues[A][[1, k]];
Table[hangerframe[k], {k, 1, 3}]
```

```
{{-0.427099, -0.877295, -0.218953},
 {0.868978, -0.331313, -0.367572}, {0.249927, -0.347255, 0.903853}}
```

Check them out:

```
{A.alignerframe[1], xstretch hangerframe[1]}
```

```
{{-1.02663, -2.10878, -0.526303}, {-1.02663, -2.10878, -0.526303}}
```

```
{A.alignerframe[2], ystretch hangerframe[2]}
```

```
{{1.32946, -0.506878, -0.562352}, {1.32946, -0.506878, -0.562352}}
```

```
{A.alignerframe[3], zstretch hangerframe[3]}
```

```
{{0.221981, -0.308426, 0.802786}, {0.221981, -0.308426, 0.802786}}
```

This checks.

Illustrate the actions of aligner, stretcher and hanger.

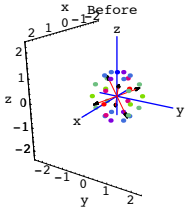
□Answer:

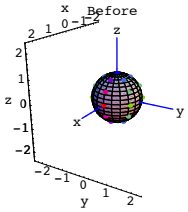
Here you go:

Stage O: Before the hit:

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

{slow, shigh} = {0, π};
{tlow, thigh} = {0, 2 π};

ranger = Max[1.2, Max[SingularValues[A][[2]]]];
pointcolor[s_, t_] := RGBColor[
   Normalize [{Abs[x[s, t]], Abs[y[s, t]], Abs[z[s, t]]}][[1]] ,
   Normalize[{Abs[x[s, t]], Abs[y[s, t]], Abs[z[s, t]]}][[2]],
   Normalize[{Abs[x[s, t]], Abs[y[s, t]], Abs[z[s, t]]}][[3]]];
```

```
sjump = (shigh - slow)/6;
tjump = (thigh - tlow)/6;

Clear[hitplotter, hitpointplotter, matrix3D];
hitplotter[matrix3D_] :=
 ParametricPlot3D[matrix3D.{x[s, t], y[s, t], z[s, t]},
  {s, slow, shigh}, {t, tlow, thigh}, PlotRange ->
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Axes -> True, AxesLabel -> {"x", "y", "z"}, Boxed -> False,
  ViewPoint -> CMView, DisplayFunction -> Identity]

hitpointplotter[matrix3D_] :=
  Show[Table[Graphics3D[{pointcolor[s, t], PointSize[0.025],
      Point[matrix3D.{x[s, t], y[s, t], z[s, t]}]}],
    {s, slow, shigh - sjump, sjump}, {t, tlow, thigh - tjump, tjump}],
   PlotRange -> {{-ranger, ranger}, {-ranger, ranger},
     {-ranger, ranger}}, Axes -> True, AxesLabel -> {"x", "y", "z"},
   Boxed -> False, ViewPoint -> CMView, DisplayFunction -> Identity];

hitframeplotter[matrix3D_] :=
  {Table[Arrow[matrix3D.alignerframe[k],
     Tail -> {0, 0, 0}, VectorColor -> Red], {k, 1, 3}],
   Table[Arrow[-matrix3D.alignerframe[k], Tail -> {0, 0, 0},
     VectorColor -> Red], {k, 1, 3}]};

Show[hitpointplotter[IdentityMatrix[3]],
  hitframeplotter[IdentityMatrix[3]], Axes3D[ranger],
  PlotLabel -> "Before", DisplayFunction -> $DisplayFunction];

Show[hitplotter[IdentityMatrix[3]],
  hitpointplotter[IdentityMatrix[3]],
  hitframeplotter[IdentityMatrix[3]], Axes3D[ranger],
  PlotLabel -> "Before", DisplayFunction -> $DisplayFunction];
```



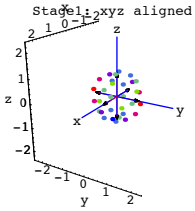


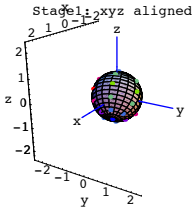
Note the aligner frame.
Grab these two plots and animate.

Stage1: Hit with aligner to align on the x, y and z-axes

This plots the surface with
{1,0,0} pointing out the positive x-axis playing the role of alignerframe[1],
{0,1,0} pointing out the positive y-axis playing the role of alignerframe[2],
and {0,0,1} pointing out the positive 3-axis playing the role of alignerframe[3],.

```
stage1hitter = aligner;

Show[hitpointplotter[stage1hitter], hitframeplotter[stage1hitter],
  Axes3D[ranger], PlotLabel -> "Stage1: xyz aligned",
  DisplayFunction -> $DisplayFunction];

Show[hitplotter[stage1hitter],
  hitframeplotter[stage1hitter], Axes3D[ranger],
  hitpointplotter[stage1hitter], PlotLabel -> "Stage1: xyz aligned",
  DisplayFunction -> $DisplayFunction];
```

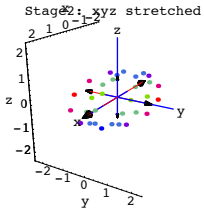


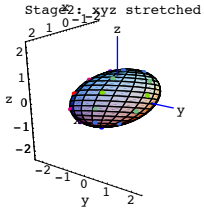


Grab these two plots and animate at various speeds.
Then grab all four plots above and animate slowly.

Stage2: Stretch along the x, y and z -axes:

```
stage2hitter = stretcher.aligner;

Show[hitpointplotter[stage2hitter], hitframeplotter[stage2hitter],
  Axes3D[ranger], PlotLabel -> "Stage2: xyz stretched",
  DisplayFunction -> $DisplayFunction];

Show[hitplotter[stage2hitter], hitframeplotter[stage2hitter],
  Axes3D[ranger], hitpointplotter[stage2hitter],
  PlotLabel -> "Stage2: xyz stretched",
  DisplayFunction -> $DisplayFunction];
```

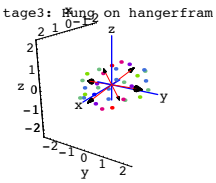


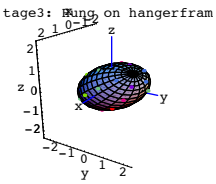


Grab these two plots and animate at various speeds.
Then grab all six plots above and animate slowly.

Stage3: Hang on the hanger frame.

```
stage3hitter = hanger.stretcher.aligner;

Show[hitpointplotter[stage3hitter], hitframeplotter[stage3hitter],
  Axes3D[ranger], PlotLabel -> "Stage3: Hung on hangerframe",
  DisplayFunction -> $DisplayFunction];

Show[hitplotter[stage3hitter], hitframeplotter[stage3hitter],
  Axes3D[ranger], hitpointplotter[stage3hitter],
  PlotLabel -> "Stage3: Hung on hangerframe",
  DisplayFunction -> $DisplayFunction];
```





Grab these two plots and animate at various speeds.
Then grab all eight plots above and animate slowly.

Done.

B.2) Using SVD analysis in 3D to try to invert 3D matrices

□B.2.a.i) Success at inverting

Here's a sample 3D matrix A:

```
A = {{1.3, -3.5, 1.3}, {0.5, 2.1, -0.9}, {-0.4, 1.8, 4.6}};
MatrixForm[A]
```

$$\begin{pmatrix} 1.3 & -3.5 & 1.3 \\ 0.5 & 2.1 & -0.9 \\ -0.4 & 1.8 & 4.6 \end{pmatrix}$$

Here's *Mathematica*'s calculation of A^{-1}:

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

$$\begin{pmatrix} 0.475628 & 0.777534 & 0.0177096 \\ -0.0818013 & 0.274077 & 0.0767414 \\ 0.0733682 & -0.0396357 & 0.188902 \end{pmatrix}$$

Use SVD analysis to explain where this calculation comes from.

□ Answer:

Do your SVD analysis in the form

A = hanger. stretcher.aligner:

```
hanger = Transpose[SingularValues[A][[1]]];
MatrixForm[hanger]
```

$$\begin{pmatrix} 0.104962 & -0.871593 & -0.478862 \\ -0.0140423 & 0.480174 & -0.877061 \\ -0.994377 & -0.0987826 & -0.038161 \end{pmatrix}$$

```
{xstretch, ystretch, zstretch} = SingularValues[A][[2]];
stretcher = DiagonalMatrix[{xstretch, ystretch, zstretch}];
MatrixForm[stretcher]
```

$$\begin{pmatrix} 4.96395 & 0 & 0 \\ 0 & 4.45763 & 0 \\ 0 & 0 & 1.07179 \end{pmatrix}$$

```
aligner = SingularValues[A][[3]];
MatrixForm[aligner]
```

$$\begin{pmatrix} 0.106202 & -0.440524 & -0.891437 \\ -0.191463 & 0.870671 & -0.453072 \\ -0.975737 & -0.218794 & -0.0081231 \end{pmatrix}$$

Check it out:

```
MatrixForm[A]
MatrixForm[hanger.stretcher.aligner]
```

$$\begin{pmatrix} 1.3 & -3.5 & 1.3 \\ 0.5 & 2.1 & -0.9 \\ -0.4 & 1.8 & 4.6 \end{pmatrix}$$

$$\begin{pmatrix} 1.3 & -3.5 & 1.3 \\ 0.5 & 2.1 & -0.9 \\ -0.4 & 1.8 & 4.6 \end{pmatrix}$$

Good.

This tells you that

A = hanger.stretcher.aligner.

Now remembering that A^{-1} undoes everything A does, you build

A^{-1}= aligner^{-1}.stretcher^{-1}.hanger^{-1}.

To do this, note that just as in 2D:

- aligner^{-1} = aligner^{t}:

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

$$\begin{pmatrix} 0.106202 & -0.191463 & -0.975737 \\ -0.440524 & 0.870671 & -0.218794 \\ -0.891437 & -0.453072 & -0.0081231 \end{pmatrix}$$

```
MatrixForm[Transpose[aligner]]
```

$$\begin{pmatrix} 0.106202 & -0.191463 & -0.975737 \\ -0.440524 & 0.870671 & -0.218794 \\ -0.891437 & -0.453072 & -0.0081231 \end{pmatrix}$$

- stretcher^{-1} $= \begin{pmatrix} \frac{1}{\text{xstretch}} & 0 & 0 \\ 0 & \frac{1}{\text{ystretch}} & 0 \\ 0 & 0 & \frac{1}{\text{zstretch}} \end{pmatrix}$:

```
unstretcher = DiagonalMatrix[{1/xstretch, 1/ystretch, 1/zstretch}];
MatrixForm[unstretcher]
```

$$\begin{pmatrix} 0.201453 & 0 & 0 \\ 0 & 0.224334 & 0 \\ 0 & 0 & 0.933018 \end{pmatrix}$$

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

$$\begin{pmatrix} 0.201453 & 0 & 0 \\ 0 & 0.224334 & 0 \\ 0 & 0 & 0.933018 \end{pmatrix}$$

And

- hanger^{-1} = hanger^{t}:

```
Inverse[hanger]
```

```
{{0.104962, -0.0140423, -0.994377},
 {-0.871593, 0.480174, -0.0987826},
 {-0.478862, -0.877061, -0.038161}}
```

```
Transpose[hanger]
```

```
{{0.104962, -0.0140423, -0.994377},
 {-0.871593, 0.480174, -0.0987826},
 {-0.478862, -0.877061, -0.038161}}
```

The upshot: A^{-1} is given by

```
SVDinverse = Transpose[aligner].unstretcher.Transpose[hanger];

MatrixForm[SVDinverse]
```

$$\begin{pmatrix} 0.475628 & 0.777534 & 0.0177096 \\ -0.0818013 & 0.274077 & 0.0767414 \\ 0.0733682 & -0.0396357 & 0.188902 \end{pmatrix}$$

Compare with *Mathematica*'s direct calculation:

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

$$\begin{pmatrix} 0.475628 & 0.777534 & 0.0177096 \\ -0.0818013 & 0.274077 & 0.0767414 \\ 0.0733682 & -0.0396357 & 0.188902 \end{pmatrix}$$

Had it all the way.

□ B.1.b.ii) Failure

Here's another sample 3D matrix A

```
A = {{2.4, -7.2, 0.3}, {0.8, -2.4, 0.1}, {1.0, -1.0, 1.2}};
MatrixForm[A]
```

$$\begin{pmatrix} 2.4 & -7.2 & 0.3 \\ 0.8 & -2.4 & 0.1 \\ 1. & -1. & 1.2 \end{pmatrix}$$

Here's *Mathematica*'s attempt at a calculation of A^{-1}:

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

```
Inverse::luc :
 Result for Inverse of badly conditioned matrix {{2.4, -7.2, 0.3}, {0.8, -2.4, 0.1},
    {1., -1., 1.2}} may contain significant numerical errors.
```

$$\begin{pmatrix} -2.57826 \times 10^{15} & 7.73477 \times 10^{15} & -0.102965 \\ -7.9759 \times 10^{14} & 2.39277 \times 10^{15} & 0.00411862 \\ 1.48389 \times 10^{15} & -4.45166 \times 10^{15} & 0.92257 \end{pmatrix}$$

Garbage.

Explain what happened.

□ Answer:

Do an SVD analysis of the stretch factors of A:

```
SingularValues[A][[2]]
```

```
{8.11577, 1.29395}
```

Ouch!

No need to go further!

The fact that *Mathematica* reports only two stretch factors is *Mathematica*'s way of telling you that the third stretch factor is 0.

This also signals that a hit by this matrix squashes everything onto the plane through {0,0,0} determined by hangerframe[1] and hangerframe[2]:

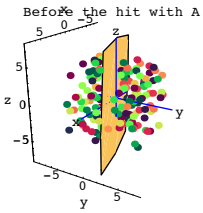
```
Clear[hangerframe];
{hangerframe[1], hangerframe[2]} = SingularValues[A][[1]]
```

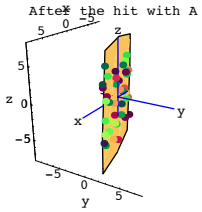
```
{{-0.935545, -0.311848, -0.165851}, {-0.15734, -0.0524467, 0.986151}}
```

Watch it happen in this action movie:

```
Clear[hangerframe];
{hangerframe[1], hangerframe[2]} = SingularValues[A][[1]];
a = 5;
points = Table[{Random[Real, {-a, a}],
    Random[Real, {-a, a}], Random[Real, {-a, a}]}, {k, 1, 200}];
pointcolor[k_] = RGBColor[0.5 (Sin[2 π k / Length[points]] + 1),
   0.5 (Cos[2 π k / Length[points]] + 1), 0.3];

pointplot = Table[
   Graphics3D[{PointSize[0.04], pointcolor[k], Point[points[[k]]]}],
   {k, 1, Length[points]}];

hitpointplot = Table[Graphics3D[{PointSize[0.04], pointcolor[k],
     Point[A.points[[k]]]}], {k, 1, Length[points]}];

b = 8;
planeplot = Graphics3D[
   Polygon[{-b hangerframe[1] - b hangerframe[2], -b hangerframe[1] +
      b hangerframe[2], b hangerframe[1] + b hangerframe[2],
     b hangerframe[1] - b hangerframe[2]}]];

ranger = b;
```

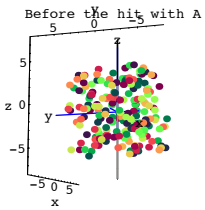
```
before = Show[pointplot, planeplot, Axes3D[ranger],
   Axes → True, AxesLabel → {"x", "y", "z"}, PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   Boxed → False, ViewPoint → CMView,
   PlotLabel → "Before the hit with A"];

after = Show[hitpointplot, planeplot, Axes3D[ranger], Axes → True,
   Axes → True, AxesLabel → {"x", "y", "z"}, PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   Boxed → False, ViewPoint → CMView,
   PlotLabel → "After the hit with A"];
```

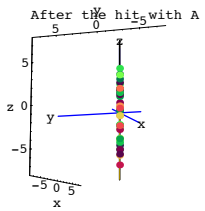




See the same thing from the view point of 6 hangerframe[1].

```
Show[before, ViewPoint → 6 hangerframe[1]];

Show[after, ViewPoint → 6 hangerframe[1]];
```

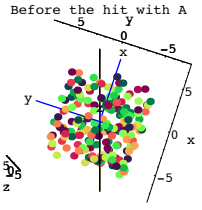


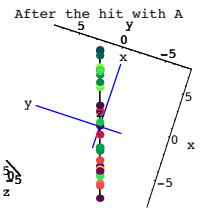


Grab and animate.

See the same thing from the view point of 6 hangerframe[2].

```
Show[before, ViewPoint → 6 hangerframe[2]];

Show[after, ViewPoint → 6 hangerframe[2]];
```





The plotted plane is the plane through {0,0,0} defined by hangerframe[1] and hangerframe[2].

The hit with A squashes everything onto this plane and the hit with A deposits many different points onto the same point on the plane.

There is no way of going from a given point {a,b,c} on the plane to a single point {x,y,z} with

A.{x,y,z} = {a,b,c}.

The upshot: There is no way to undo a hit with A.

That's why A is not invertible.

B.3) Rank of a 3D Matrix

□B.3.a.i) The rank of a matrix is the number of dimensions that matrix uses to hang its hits

How do you determine the rank of a given 3D matrix A?

□Answer:

Do an SVD analysis of the SVD stretch factors of A.

If A has three non-zero stretch factors, then the rank of A is 3.
In this case A uses all of 3D to hang its hits.

If A has exactly two non-zero stretch factors, then the rank of A is 2.
In this case, A hangs all of its hits on a two dimensional plane within 3D.

If A has exactly one non-zero stretch factor, then the rank of A is 1.
In this case, A hangs all of its hits on a one dimensional line within 3D.

If A has no non-zero stretch factors, then the rank of A is 0.
In this case, A hits all {x,y,z}'s onto {0,0,0}.
The only 3D matrix A whose rank is 0 is

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

□B.3.a.ii) Determine the rank and plot

Here's a 3D matrix A:

```
A = {{1.0, -1.0, 0.5}, {1.7, -0.5, -1.3}, {0.4, -0.4, 0.2}};
MatrixForm[A]
```

$$\begin{pmatrix} 1. & -1. & 0.5 \\ 1.7 & -0.5 & -1.3 \\ 0.4 & -0.4 & 0.2 \end{pmatrix}$$

Determine the rank of A.
If the rank of A is 1, then give the line on which A hangs all its hits.
If the rank of A is 2, then give the plane on which A hangs all its hits.
Illustrate with decisive plots.

□Answer:

SVD analyze the stretch factors of A:

```
stretches = SingularValues[A][[2]]
```
```
{2.39264, 1.30968}
```

Exactly two non-zero stretch factors.

This tells you that the rank of A is 2.

This matrix A hangs all its hits on the plane through {0, 0, 0} framed by

{hangerframe[1], hangerframe[2]}.

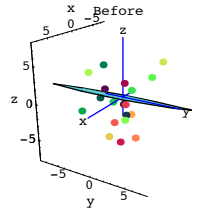
This plane consists of all points of the form:

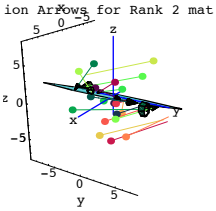
```
Clear[hangerframe, s, t];
{hangerframe[1], hangerframe[2]} = SingularValues[A][[1]];

s hangerframe[1] + t hangerframe[2]
```
```
{-0.438607 s + 0.818347 t,
 -0.881387 s - 0.472395 t, -0.175443 s + 0.327339 t}
```

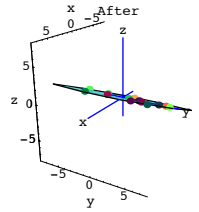
See it happen:

```
a = 5;
points = Table[{Random[Real, {-a, a}],
    Random[Real, {-a, a}], Random[Real, {-a, a}]}, {k, 1, 20}];
pointcolor[k_] = RGBColor[0.5 (Sin[2 π k / Length[points]] + 1),
   0.5 (Cos[2 π k / Length[points]] + 1), 0.3];

pointplot = Table[
   Graphics3D[{PointSize[0.04], pointcolor[k], Point[points[[k]]]}],
   {k, 1, Length[points]}];
```

```
hitpointplot = Table[Graphics3D[{PointSize[0.04], pointcolor[k],
     Point[A.points[[k]]]}], {k, 1, Length[points]}];

actionarrows = Table[Arrow[A.points[[k]] - points[[k]], Tail → points[[k]],
    VectorColor → pointcolor[k]], {k, 1, Length[points]}];

b = 8;
planeplot = Graphics3D[
   Polygon[{-b hangerframe[1] - b hangerframe[2], -b hangerframe[1] +
      b hangerframe[2], b hangerframe[1] + b hangerframe[2],
     b hangerframe[1] - b hangerframe[2]}]];

ranger = b;
before = Show[pointplot, planeplot, Axes3D[ranger], Axes → True,
   AxesLabel → {"x", "y", "z"}, Boxed → False, PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   ViewPoint → CMView, PlotLabel → "Before"];

action = Show[before, actionarrows, PlotRange → {{-ranger, ranger},
     {-ranger, ranger}, {-ranger, ranger}}, ViewPoint → CMView,
   PlotLabel → "Action Arrows for Rank 2 matrix"];

after = Show[hitpointplot, planeplot, Axes3D[ranger], Axes → True,
   Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False, PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   ViewPoint → CMView, PlotLabel → "After"];
```



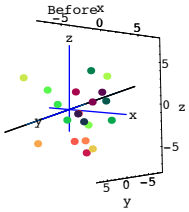


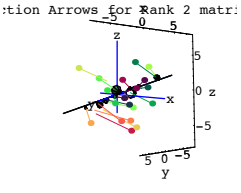


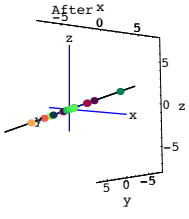
Grab and animate.

See the same thing from the view point of 6 hangerframe[1].

```
Show[before, ViewPoint → 6 hangerframe[1]];
Show[action, ViewPoint → 6 hangerframe[1]];
Show[after, ViewPoint → 6 hangerframe[1]];
```







This matrix is not invertible.

□B.3.a.iii) Determine the rank and then plot

Here's yet another 3D matrix A:

```
A = {{1.24, -1.18, 1.3}, {1.86, -1.77, 1.95}, {0.496, -0.472, 0.52}};
MatrixForm[A]
```

$$\begin{pmatrix} 1.24 & -1.18 & 1.3 \\ 1.86 & -1.77 & 1.95 \\ 0.496 & -0.472 & 0.52 \end{pmatrix}$$

Determine the rank of A.
If the rank of A is 1, then give the line on which A hangs all its hits.
If the rank of A is 2, then give the plane on which A hangs all its hits.
Illustrate with decisive plots.

□Answer:

SVD analyze the stretch factors of A:

```
stretches = SingularValues[A][[2]]
```

```
{3.96916}
```

Exactly one non-zero stretch factor.

This tells you that the rank of A is 1.

This matrix A hangs all its hits on the line through {0,0,0} consisting of all the multiples of hangerframe[1].

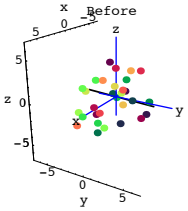
```
Clear[hangerframe, t];
{hangerframe[1]} = SingularValues[A][[1]];

t hangerframe[1]
```

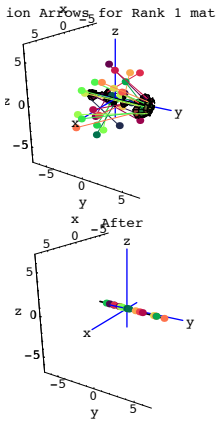
```
{-0.54153 t, -0.812296 t, -0.216612 t}
```

See it happen:

```
a = 4;
points = Table[{Random[Real, {-a, a}],
    Random[Real, {-a, a}], Random[Real, {-a, a}]}, {k, 1, 30}];
pointcolor[k_] = RGBColor[0.5 (Sin[2 π k / Length[points]] + 1),
   0.5 (Cos[2 π k / Length[points]] + 1), 0.3];
```

```
pointplot = Table[
   Graphics3D[{PointSize[0.04], pointcolor[k], Point[points[[k]]]}],
   {k, 1, Length[points]}];

hitpointplot = Table[Graphics3D[{PointSize[0.04], pointcolor[k],
     Point[A.points[[k]]]}], {k, 1, Length[points]}];

actionarrows = Table[Arrow[A.points[[k]] - points[[k]], Tail → points[[k]],
    VectorColor → pointcolor[k]], {k, 1, Length[points]}];

b = 12;
lineplot = Graphics3D[
   {Thickness[0.01], Line[{-b hangerframe[1], b hangerframe[1]}]}];

ranger = 7;
before = Show[pointplot, lineplot, Axes3D[ranger], Axes → True,
   AxesLabel → {"x", "y", "z"}, Boxed → False, PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   ViewPoint → CMView, PlotLabel → "Before"];

action = Show[before, actionarrows, PlotRange → {{-ranger, ranger},
     {-ranger, ranger}, {-ranger, ranger}}, ViewPoint → CMView,
   PlotLabel → "Action Arrows for Rank 1 matrix"];

after = Show[hitpointplot, lineplot, Axes3D[ranger], Axes → True,
   AxesLabel → {"x", "y", "z"}, Boxed → False, PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   ViewPoint → CMView, PlotLabel → "After"];
```

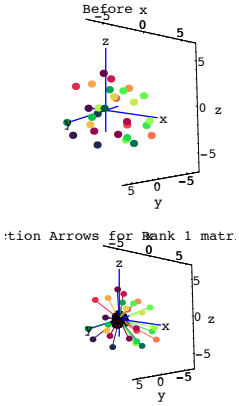




Grab and animate.

See the same thing from the view point of 6 hangerframe[1].

```
Show[before, ViewPoint → 6 hangerframe[1]];
Show[action, ViewPoint → 6 hangerframe[1]];
Show[after, ViewPoint → 6 hangerframe[1]];
```



Grab and animate.

This matrix is not invertible.

☐B.3.a.iv) Determine the rank

Here's another 3D matrix A:

```
A = {{2.247, 3.581, -2.334},
     {0.863, -4.776, 3.453},
     {1.770, -3.142, 2.183}};
MatrixForm[A]
```

```
{{2.247, 3.581, -2.334},
 {0.863, -4.776, 3.453}, {1.77, -3.142, 2.183}}
```

$$\begin{pmatrix} 2.247 & 3.581 & -2.334 \\ 0.863 & -4.776 & 3.453 \\ 1.77 & -3.142 & 2.183 \end{pmatrix}$$

Determine the rank of A.
If the rank of A is 1, then give the line on which A hangs all its hits.
If the rank of A is 2, then give the plane on which A hangs all its hits.

☐Answer:

SVD analyze the stretch factors of A:

```
stretches = SingularValues[A][[2]]
```

```
{8.22805, 2.9759, 0.109838}
```

Three non-zero stretch factors.

This tells you that the rank of A is 3.

This matrix A uses all of 3D to hang its hits. It is invertible.

In fact, saying that a 3D matrix A is invertible is the same as saying that the rank of A is 3.

B.4) Linear Algebra: Using the idea of rank and 3D SVD analysis to try to solve systems of linear equations

☐B.4.a) Success when the coefficient matrix is invertible

Use what you know about 3D matrices to try come up with the x and the y and the z that solve the simultaneous linear equations:

$$\begin{aligned} 2.1\ x - 1.7\ y + 2.1\ z &= 7.1 \\ 1.8\ x + 0.6\ y - 3.2\ z &= -1.5 \\ -2.1\ x \quad\quad + 0.7\ z &= 2.3. \end{aligned}$$

☐Answer:

Go with this matrix:

```
                    Folks like to call this the "coefficient matrix."
A = {{2.1, -1.7, 2.1},
     {1.8, 0.6, -3.2},
     {-2.1, 0, 0.7}};
MatrixForm[A]
```

$$\begin{pmatrix} 2.1 & -1.7 & 2.1 \\ 1.8 & 0.6 & -3.2 \\ -2.1 & 0 & 0.7 \end{pmatrix}$$

Check the rank of A:

```
stretches = SingularValues[A][[2]]
```

```
{4.22645, 3.52677, 0.386026}
```

Three non-zero stretch factors. The rank of A is 3 and this means A is invertible.

The linear system of equations is

$$\begin{aligned} 2.1\ x - 1.7\ y + 2.1\ z &= 7.1 \\ 1.8\ x + 0.6\ y - 3.2\ z &= -1.5 \\ -2.1\ x \quad\quad + 0.7\ z &= 2.3. \end{aligned}$$

This is the same as

A.{x, y, z} = {7.1, −1.5, 2.3}:

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

ColumnForm[Thread[A.{x, y, z} == {7.1, -1.5, 2.3}]]
```

```
2.1 x - 1.7 y + 2.1 z == 7.1
1.8 x + 0.6 y - 3.2 z == -1.5
-2.1 x + 0.7 z == 2.3
```

To come up with the x, the y and the z that solve this system, start with

A.{x, y, z} = {7.1, −1.5, 2.3}:

and hit both sides with A^{-1} to get

{x, y, z} = A^{-1}.A.{x, y, z} = A^{-1} {7.1, −1.5, 2.3}:

```
{x, y, z} = Inverse[A].{7.1, -1.5, 2.3}
```

```
{-1.87887, -9.40146, -2.35089}
```

Check:

```
A.{x, y, z} == {7.1, -1.5, 2.3}
```

```
True
```

Check again:

```
Clear[x, y, z];
Solve[A.{x, y, z} == {7.1, -1.5, 2.3}]
```

```
{{x → -1.87887, y → -9.40146, z → -2.35089}}
```

Got it.

☐B.4.b.i) What to try when the coefficient matrix is not invertible

Here's a new 3D matrix

```
A = {{1.288, -0.166, -0.148},
     {-0.162, 2.008, -0.592},
     {-0.563, -0.921, 0.370}};
MatrixForm[A]
```

$$\begin{pmatrix} 1.288 & -0.166 & -0.148 \\ -0.162 & 2.008 & -0.592 \\ -0.563 & -0.921 & 0.37 \end{pmatrix}$$

Check the rank of A:

```
stretches = SingularValues[A][[2]]
```

```
{2.31839, 1.42981}
```

Exactly two non-zero stretch factors so the rank of A is two.
This signals that A is not invertible:

```
Inverse[A]
```

```
Inverse::sing :
 Matrix {{1.288, -0.166, -0.148}, {-0.162, <<6>>, -0.592}, {-0.563, -0.921, 0.37}} is singular.
  Inverse[{{1.288, -0.166, -0.148},
    {-0.162, 2.008, -0.592}, {-0.563, -0.921, 0.37}}]
```

Garbage.
In spite of this for a given u, v and w, the corresponding linear system,

$$\begin{aligned} 1.288\ x - 0.166\ y - 0.148\ z &= u \\ -0.162\ x + 2.008\ y - 0.592\ z &= v \end{aligned}$$

$-0.563\,x - 0.921\,y + 0.370\,z = w$,

might have many or might have no solutions for x, y and z, depending on where the point {u, v, w} is located.

How do you make this determination?

□ Answer:

The fact that the rank of A is 2 tells you that you that A hangs all its hits on a 2D plane in 3D.

In other words the rank of A is 2.

This plane is framed by:

```
Clear[hangerframe];
{hangerframe[1], hangerframe[2]} = SingularValues[A][[1]]
```

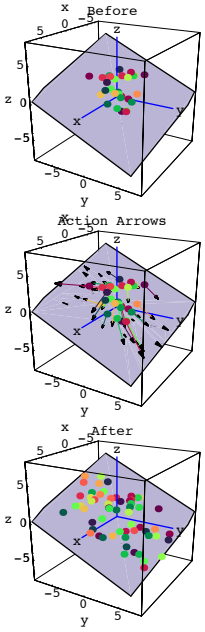
```
{{0.0378603, -0.90123, 0.431685}, {-0.912085, 0.145322, 0.383382}}
```

In other words, this plane consists of all points of the form

s hangerframe[1] + t hangerframe[2]

What it happen in this action movie:

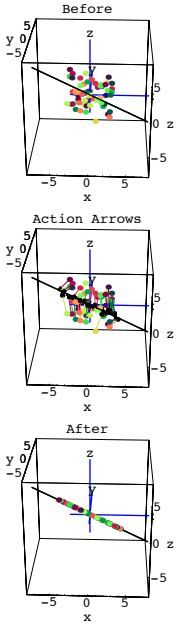
```
a = 3;
points = Table[{Random[Real, {-a, a}],
    Random[Real, {-a, a}], Random[Real, {-a, a}]}, {k, 1, 60}];
pointcolor[k_] = RGBColor[0.5 (Sin[2 π k / Length[points]] + 1),
   0.5 (Cos[2 π k / Length[points]] + 1), 0.3];

pointplot = Table[
   Graphics3D[{PointSize[0.04], pointcolor[k], Point[points[[k]]]}],
   {k, 1, Length[points]}];

hitpointplot = Table[Graphics3D[{PointSize[0.04], pointcolor[k],
     Point[A.points[[k]]]}], {k, 1, Length[points]}];

actionarrows = Table[Arrow[A.points[[k]] - points[[k]], Tail → points[[k]],
    VectorColor → pointcolor[k]], {k, 1, Length[points]}];

b = 8;
planeplot = Graphics3D[
   Polygon[{-b hangerframe[1] - b hangerframe[2], -b hangerframe[1] +
      b hangerframe[2], b hangerframe[1] + b hangerframe[2],
     b hangerframe[1] - b hangerframe[2]}]];

ranger = b;
before = Show[pointplot, planeplot, Axes3D[ranger],
   Axes → True, AxesLabel → {"x", "y", "z"}, PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   ViewPoint → CMView, PlotLabel → "Before"];

action = Show[before, actionarrows, PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   ViewPoint → CMView, PlotLabel → "Action Arrows"];

after = Show[hitpointplot, planeplot, Axes3D[ranger], Axes → True,
   Axes → True, AxesLabel → {"x", "y", "z"}, PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   ViewPoint → CMView, PlotLabel → "After"];
```



Grab and animate.

See the same thing from the view point of 6 hangerframe[1].

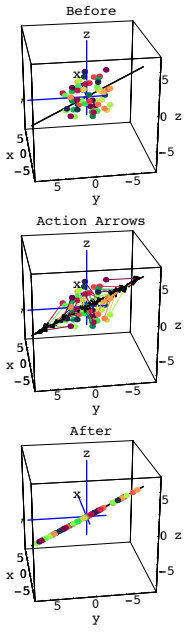
```
Show[before, ViewPoint → 6 hangerframe[1]];
Show[action, ViewPoint → 6 hangerframe[1]];
Show[after, ViewPoint → 6 hangerframe[1]];
```



Grab and animate.

See the same thing from the view point of 6 hangerframe[2].

```
Show[before, ViewPoint → 6 hangerframe[2]];
Show[action, ViewPoint → 6 hangerframe[2]];
Show[after, ViewPoint → 6 hangerframe[2]];
```



Grab and animate.

The plotted plane is the plane through {0,0,0} defined by hangerframe[1] and hangerframe[2].

A hit with A squashes everything onto this plane.

The upshot:

If you are going to have any chance of solving

A.{x, y, z} = {u, v, w}

for {x, y, z}, then {u, v, w} MUST be on the plane through {0,0,0} defined by hangerframe[1] and hangerframe[2].

□ B.4.b.ii) A solution if

{u,v,w} = ({u,v,w} .hangerframe[1])hangerframe[1] +

({u,v,w} .hangerframe[2])hangerframe[2].

Otherwise no solution

Stay with the same matrix A as in part i) above.
You now know that if you are going to have any chance of solving

A.{x, y, z} = {u, v, w}

for {x, y, z}, then {u, v, w} MUST be on the plane through {0,0,0} defined by hangerframe[1] and hangerframe[2].
How do you make this determination?

□ Answer:

Try to resolve {u,v,w} into components in the directions of hangerframe[1] and hangerframe[2] by calculating

soltest = ({u,v,w} .hangerframe[1])hangerframe[1] +
({u,v,w} .hangerframe[2])hangerframe[2] .

If soltest = {u,v,w} then {u,v,w} is on the plane through {0,0,0} defined by hangerframe[1] and hangerframe[2] and a solution {x,y,z} is guaranteed.

If soltest ≠ {u,v,w} then {u,v,w} is not on the plane through {0,0,0} defined by hangerframe[1] and hangerframe[2] there is no possible solution {x,y,z}.

An example:

```
{u, v, w} = {1.00, -1.35, 0.20}
{1., -1.35, 0.2}
soltest = ({u, v, w} .hangerframe[1] ) hangerframe[1] +
  ({u, v, w} .hangerframe[2] ) hangerframe[2]
{0.991667, -1.35833, 0.183333}
```

Close but no cigar; soltest ≠ {u,v,w} and this signals that there is no possible solution {x,y,z} of A.{x,y,z} = {u,v,w}.

Another example:

```
{u, v, w} = {1.0134, 4.3508, -2.6821}
{1.0134, 4.3508, -2.6821}
soltest = ({u, v, w} .hangerframe[1] ) hangerframe[1] +
  ({u, v, w} .hangerframe[2] ) hangerframe[2]
{1.0134, 4.3508, -2.6821}
```

Bingo. This time soltest = {u,v,w} and this signals that there is a solution{x,y,z} of A.{x,y,z} = {u,v,w}.

□ B.4.b.iii) Using SVD to come up with a solution

Stay with the same matrix A as in part i) above.
When you look at:

```
{u, v, w} = {1.0134, 4.3508, -2.6821}
{1.0134, 4.3508, -2.6821}
soltest = ({u, v, w} .hangerframe[1] ) hangerframe[1] +
  ({u, v, w} .hangerframe[2] ) hangerframe[2]
{1.0134, 4.3508, -2.6821}
```

You see there is a guaranteed solution {x,y,z} of A.{x,y,z} = {u,v,w}.
Use SVD analysis to come up with a solution.

□ Answer:

Calculate the aligner frame and the stretch factors for A:

```
Clear[alignerframe];
{alignerframe[1], alignerframe[2]} = SingularValues[A][[3]]
{{-0.0208227, -0.954773, 0.296606}, {-0.989049, 0.0630276, 0.133451}}
{xstretch, ystretch} = SingularValues[A][[2]]
{2.31839, 1.42981}
```

One solution is:

$$\{x, y, z\} = \frac{(\{u, v, w\} .\text{hangerframe}[1])}{\text{xstretch}}\ \text{alignerframe}[1] + \frac{(\{u, v, w\} .\text{hangerframe}[2])}{\text{ystretch}}\ \text{alignerframe}[2]$$

```
{0.958576, 2.01762, -0.768096}
```

Check this out:

```
A.{x, y, z}
{1.0134, 4.3508, -2.6821}
{u, v, w}
{1.0134, 4.3508, -2.6821}
```

Bagged it.

□ B.4.b.iv) Explanation of the solution

formula

$$\{x, y, z\} = \frac{(\{u,v,w\} .\text{hangerframe}[1])}{\text{xstretch}}\ \text{alignerframe}[1] + \frac{(\{u,v,w\} .\text{hangerframe}[2])}{\text{ystretch}}\ \text{alignerframe}[2]$$

Why did that solution formula

$$\{x, y, z\} = \frac{(\{u,v,w\} .\text{hangerframe}[1])}{\text{xstretch}}\ \text{alignerframe}[1] + \frac{(\{u,v,w\} .\text{hangerframe}[2])}{\text{ystretch}}\ \text{alignerframe}[2]$$

work?
And why will it work for any {u,v,w} with

{u,v,w} = ({u,v,w} .hangerframe[1])hangerframe[1]
({u,v,w} .hangerframe[2])hangerframe[2]?

□ Answer:

Start with

{x,y,z} =

$$\frac{(\{u,v,w\} .\text{hangerframe}[1])}{\text{xstretch}}\ \text{alignerframe}[1] + \frac{(\{u,v,w\} .\text{hangerframe}[2])}{\text{ystretch}}\ \text{alignerframe}[2]$$

Hit both sides with A to get

A.{x,y,z} =

$$\frac{(\{u,v,w\} .\text{hangerframe}[1])}{\text{xstretch}}\ \text{A.alignerframe}[1] + \frac{(\{u,v,w\} .\text{hangerframe}[2])}{\text{ystretch}}\ \text{A.alignerframe}[2]$$

This is the same as

A.{x,y,z} =

$$\frac{(\{u,v,w\} .\text{hangerframe}[1])}{\text{xstretch}}\ \text{xstretch hangerframe}[1] + \frac{(\{u,v,w\} .\text{hangerframe}[2])}{\text{ystretch}}\ \text{ystretch hangerframe}[2]$$

Reasons:
A.alignerframe[1] = xstretch hangerframe[1]
A.alignerframe[2] = ystretch hangerframe[2]

This is the same as

A.{x,y,z} =

({u, v, w} .hangerframe[1]) hangerframe[1] +
({u, v, w} .hangerframe[2]) hangerframe[2]

Reason:
Cancellation

And if {u,v,w} =

({u, v, w} .hangerframe[1]) hangerframe[1] +
({u, v, w} .hangerframe[2]) hangerframe[2]

This is the same as

A.{x,y,z} = {u,v,w}.

That's why the formula will work any time you have

{u,v,w} =

({u, v, w} .hangerframe[1]) hangerframe[1] +
({u, v, w} .hangerframe[2]) hangerframe[2]

□ B.4.b.v) Even more success

Stay with the same 3D matrix A as in part iii) above. And stay with the same {x,y,z} and {u,v.w} as in part iii) so that

A.{x,y,z} = {u,v,w}

Are there solutions other than the {x,y,z} you found in part iii)?
If so, indicate with a plot where they are.

□ Answer:

Yes.

There are infinitely many.

To see where they come from, remember that zstretch = 0. This tells you that

A.(t alignerframe[3]) = {0,0,0},

The SVD instruction will not give you alignerframe[3], but you can get it with:

```
alignerframe[3] = NullSpace[A][[1]]
{0.14611, 0.290579, 0.94563}
```

Check:

```
Clear[t];
A.(t alignerframe[3])
{0, 0, 0}
```

When you go with the Xsol you got in part iii) and add to it any multiple of alignerframe[3], you get another solution:

```
Clear[t];
Xsol = {x, y, z};

Chop[Expand[A.(Xsol + t alignerframe[3])]] == {u, v, w}
True
```

The upshot:

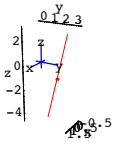
All the solutions of the linear system

A.{x,y,z} = {u,v,w}

are located on the line through Xsol with direction vector alignerframe[3].

Here's a look at part of it:

```
basepoint = Xsol;
directionvector = alignerframe[3];
basepointplot =
  Graphics3D[{Red, PointSize[0.05], Point[basepoint]}];

a = 4;
solutionlineplot = Graphics3D[{Red, Thickness[0.01], Line[
     {basepoint - a directionvector, basepoint + a directionvector}]}];

solutions = Show[basepointplot,
   solutionlineplot, Axes3D[1.5], ViewPoint → CMView,
   Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False];
```



Think of it:

All points on the plotted line are solutions of A.{x,y,z} = {u,v,w}.

B.5) Determinants of 3D matrices.

Positive versus negative orientation of the columns of a 3D matrix.

If the columns of A are positively oriented, then Det[A] = + xstretch ystretch zstretch.

If the columns of A are negatively oriented, then Det[A] = — xstretch ystretch zstretch.

So you can use |Det[A]| = xstretch ystretch zstretch as the volume conversion factor.

If you interchange the vertical columns of a 3D matrix, you change the sign but not the absolute value of the determinant.

The short, sweet formula Det[A] = col[1].(col[2] × col[3])

□ **B.5.a.i) If col[1].(col[2]×col[3]) > 0 , then the columns of A are positively oriented.**

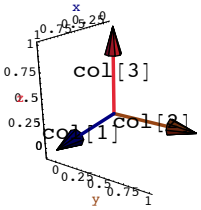
If col[1].(col[2]×col[3]) < 0 , then the columns of A are negatively oriented

Talk about orientation of the columns of a 3D matrix.

□ **Answer:**

Here are the columns of the 3D Identity matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$:

```
Clear[colplotter, matrix];
colplotter[matrix_] :=
 Show[Arrow[matrix.{1, 0, 0}, Tail → {0, 0, 0},
   VectorColor → NavyBlue, HeadSize → 0.3, ShaftWidth -> 0.02],
  Arrow[matrix.{0, 1, 0}, Tail → {0, 0, 0}, VectorColor → MarsOrange,
   HeadSize → 0.3, ShaftWidth -> 0.02],
  Arrow[matrix.{0, 0, 1}, Tail → {0, 0, 0},
   VectorColor → AlizarinCrimson, HeadSize → 0.3, ShaftWidth -> 0.02],
  Graphics3D[{Text["col[1]", 0.6 matrix.{1, 0, 0}]}],
  Graphics3D[{Text["col[2]", 0.5 matrix.{0, 1, 0}]}],
  Graphics3D[{Text["col[3]", 0.5 matrix.{0, 0, 1}]}], Axes -> True,
  AxesLabel -> {"x", "y", "z"}, Boxed -> False, PlotRange -> All,
  ViewPoint -> CMView, DisplayFunction -> Identity];
Identitycols = Show[colplotter[IdentityMatrix[3]],
   DisplayFunction -> $DisplayFunction];
```



These are the x,y ,z axes unit vectors.

Folks all over the globe have agreed that their orientation is positive because

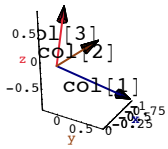
col[1].(col[2]× col[3]) = {1,0,0}•({0,1,0}×{0,0,1}) = {1,0,0}.{1,0,0} = 1 > 0:

```
{1, 0, 0}.({0, 1, 0} × {0, 0, 1})
```

```
1
```

Here's a plot of the columns of the following matrix A:

```
A = {{-1.11, -0.61, -0.70},
     {0.89, 0.46, -0.30},
     {-0.88, 0.36, 0.86}};
Acols = Show[colplotter[A], DisplayFunction -> $DisplayFunction];
```



Here
col[1] = A.{1,0,0}, col[2] = A.{1,0,0} and col[3] = A.{0,0,1}

Check their orientation:

col[1].(col[2]× col[3]) = (A.{1,0,0}).((A.{0,1,0})× (A.{0,0,1})):

```
(A.{1, 0, 0}).((A.{0, 1, 0}) × (A.{0, 0, 1}))
```

```
-0.760782
```

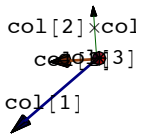
Negative.

The columns of A are negatively oriented.

See what this means by looking at the columns and .(col[2]×col[3]) from the viewpoint of col[3]:

```
Show[colplotter[A], Arrow[(A.{0, 1, 0}) × (A.{0, 0, 1}),
  Tail -> {0, 0, 0}, VectorColor -> GosiaGreen],
 Graphics3D[Text["col[2]×col[3]",
   0.6 ((A.{0, 1, 0}) × (A.{0, 0, 1}))]], ViewPoint -> 5 (A.{0, 0, 1}),
 Axes -> False, PlotLabel -> "Negatively oriented columns",
 DisplayFunction -> $DisplayFunction];
```

:ively oriented co:



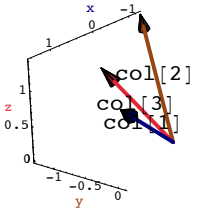
You can see that col[1].(col[2]× col[3]) < 0.

The columns of A are negatively oriented.

See another:

Here's a plot of the columns of the following matrix A:

```
A = {{1.4, -1.3, 0},
     {0, -1.3, -1.2},
     {1.0, 1.4, 0.9}};
Acols = Show[colplotter[A], DisplayFunction -> $DisplayFunction];
```



Here
col[1] = A.{1,0,0}, col[2] = A.{1,0,0} and col[3] = A.{0,0,1}

Check their orientation:

col[1].(col[2]×col[3]) = (A.{1,0,0}).((A.{0,1,0})× (A.{0,0,1})):

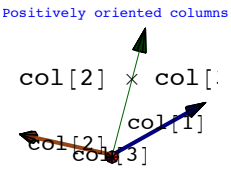
```
(A.{1, 0, 0}).((A.{0, 1, 0}) × (A.{0, 0, 1}))
```

```
2.274
```

Positive.

See what this means by looking at the columns and col[2]×col[3] from the viewpoint of col[3]:

```
Show[colplotter[A], Arrow[(A.{0, 1, 0}) × (A.{0, 0, 1}),
  Tail -> {0, 0, 0}, VectorColor -> GosiaGreen],
 Graphics3D[Text["col[2] × col[3]",
   0.6 ((A.{0, 1, 0}) × (A.{0, 0, 1}))]], ViewPoint -> 5 (A.{0, 0, 1}),
 Axes -> False, PlotLabel -> "Positively oriented columns",
 DisplayFunction -> $DisplayFunction];
```



You can see that col[1].(col[2]× col[3]) > 0.

The columns of A are positively oriented.

That's all there is to orientation in 3D.

□B.5.b.i) Definition of determinant first part:

If any given 3D matrix A has positively oriented columns,

Det[A] is defined to be the volume measurement of the parallelogram box defined by the columns of A.

So if the columns of A are positively oriented, then Det[A] = + xstretch ystretch zstretch

Here's a 3D matrix A together with *Mathematica*'s calculation of the determinant of A:

```
A = {{1.67, -1.38, -0.09}, {-0.47, -0.83, 0.16}, {-0.44, -1.69, -1.78}};
Det[A]
```

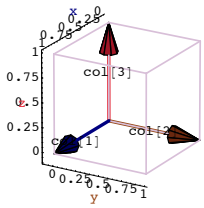
```
4.13187
```

What does this calculation of Det[$\begin{pmatrix} 1.67 & -1.38 & -0.09 \\ -0.47 & -0.83 & 0.16 \\ -0.44 & -1.69 & -1.78 \end{pmatrix}$] measure?

□Answer:

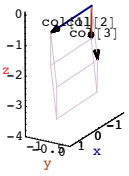
Go with the unit cube in 3D:

```
Clear[hitboxplotter, r, s, t, pointcolor, matrix];
ranger = 4;

thickness = 0.01;
color = Thistle;
hitboxplotter[matrix_] :=
 Show[Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0}, {0, 0, 0} + matrix.{1, 0, 0},
      {0, 0, 0} + matrix.{1, 0, 0} + matrix.{0, 1, 0},
      {0, 0, 0} + matrix.{0, 1, 0}, {0, 0, 0}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{1, 0, 0},
      {0, 0, 0} + matrix.{1, 0, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{0, 1, 0}, {0, 0, 0} +
       matrix.{1, 0, 0} + matrix.{0, 1, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{0, 0, 1}, {0, 0, 0} + matrix.{0, 0, 1} +
       matrix.{1, 0, 0}, {0, 0, 0} + matrix.{0, 0, 1} + matrix.{1, 0, 0} +
       matrix.{0, 1, 0}, {0, 0, 0} + matrix.{0, 0, 1} + matrix.{0, 1, 0},
      {0, 0, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0}, {0, 0, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{1, 0, 0},
      {0, 0, 0} + matrix.{1, 0, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{1, 0, 0} + matrix.{0, 1, 0}, {0, 0, 0} +
       matrix.{1, 0, 0} + matrix.{0, 1, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{0, 1, 0},
      {0, 0, 0} + matrix.{0, 1, 0} + matrix.{0, 0, 1}}]}],
  Axes -> True, AxesLabel -> {"x", "y", "z"},
  Boxed -> False,
  PlotRange -> All,
  ViewPoint -> 12 CMView,
  DisplayFunction -> Identity];
Clear[colplotter, matrix];
colplotter[matrix_] :=
 Show[Arrow[matrix.{1, 0, 0}, Tail → {0, 0, 0},
   VectorColor → NavyBlue, HeadSize → 0.3, ShaftWidth -> 0.02],
  Arrow[matrix.{0, 1, 0}, Tail → {0, 0, 0}, VectorColor → MarsOrange,
   HeadSize → 0.3, ShaftWidth -> 0.02],
  Arrow[matrix.{0, 0, 1}, Tail → {0, 0, 0},
   VectorColor → AlizarinCrimson, HeadSize → 0.3, ShaftWidth -> 0.02],
  Graphics3D[{Text["col[1]", 0.6 matrix.{1, 0, 0}]}],
  Graphics3D[{Text["col[2]", 0.5 matrix.{0, 1, 0}]}],
  Graphics3D[{Text["col[3]", 0.5 matrix.{0, 0, 1}]}], Axes -> True,
  AxesLabel -> {"x", "y", "z"}, Boxed -> False, PlotRange -> All,
  ViewPoint -> CMView, DisplayFunction -> Identity];

Show[hitboxplotter[IdentityMatrix[3]],
  colplotter[IdentityMatrix[3]],
  DisplayFunction -> $DisplayFunction];
```



Here's what you get when you hit everything with A:

```
A = {{1.67, -1.38, -0.09}, {-0.47, -0.83, 0.16}, {-0.44, -1.69, -1.78}};
Show[hitboxplotter[A], colplotter[A],
  DisplayFunction -> $DisplayFunction];
```



That's the parallelogram box defined by the columns of A.

Check the orientation of the columns of A:

col[1].(col[2]×col[3]) = (A.{1,0,0}).((A.{0,1,0})× (A.{0,0,1})):

```
(A.{1, 0, 0}).((A.{0, 1, 0}) × (A.{0, 0, 1}))
```

```
4.13187
```

The columns of A have positive orientation.

If any given 3D matrix A has positively oriented columns,

Det[A] is defined to be the volume measurement of the parallelogram box defined by the columns of A.

Because the volume of the unit cube measures out to 1 and you hit A on the unit cube to get the parallelogram box, Det[A] is given by:

```
{xstretch, ystretch, zstretch} = SingularValues[A][[2]];
detA = xstretch ystretch zstretch
```

```
4.13187
```

Check:

```
Det[A]
```

```
4.13187
```

On the money.

□B.5.b.ii) Definition of determinant second part:

If any given 3D matrix A has negatively oriented columns,

-Det[A] is defined to be the volume measurement of the parallelogram box defined by the columns of A.

So if the columns of A are negatively oriented, then Det[A] = - xstretch ystretch zstretch

Here's a 3D matrix A together with *Mathematica*'s calculation of the determinant of A:

```
A = {{-1.78, 1.69, 1.95}, {-1.30, -1.04, -0.50}, {0.93, -0.20, -1.74}};
Det[A]
```

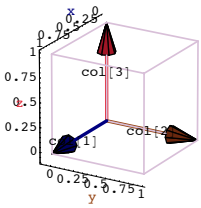
```
-5.25868
```

What does this calculation of Det[$\begin{pmatrix} -1.42 & 1.56 & 1.67 \\ 1.02 & -1.64 & -0.30 \\ -1.71 & 0.25 & 0.19 \end{pmatrix}$] measure?

□Answer:

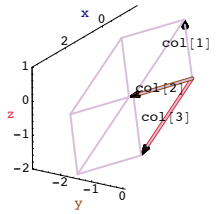
Go with the unit cube in 3D:

```
Show[hitboxplotter[IdentityMatrix[3]],
  colplotter[IdentityMatrix[3]],
  DisplayFunction -> $DisplayFunction];
```



Here's what you get when you hit everything with A:

```
A = {{-1.78  1.69   1.95 },
     {-1.30  -1.04  -0.50},
     {0.93   -0.20  -1.74}};

Show[hitboxplotter[A], colplotter[A],
  DisplayFunction -> $DisplayFunction];
```



That's the parallelogram box defined by the columns of A.

Check the orientation of the columns of A:

col[1].(col[2]×col[3]) = (A.{1,0,0}).((A.{0,1,0})× (A.{0,0,1})):

```
(A.{1, 0, 0}).((A.{0, 1, 0}) × (A.{0, 0, 1}))
-5.25868
```

The columns of A have negative orientation.

If any given 3D matrix A has negatively oriented columns,

- Det[A] is defined to be the volume measurement of the parallelogram box defined by the columns of A.

Because the volume of the unit cube measures out to 1 and you hit A on the unit cube to get the parallelogram box,

Det[A] is given by:

```
{xstretch, ystretch, zstretch} = SingularValues[A][[2]];
detA = -xstretch ystretch zstretch
-5.25868
```

Check:

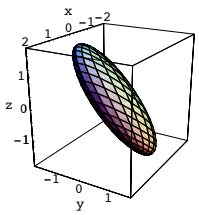
```
Det[A]
-5.25868
```

On the money.

□B.5.b.iii) Using the 3D determinant to measure volumes

Here's a random 2D matrix A together with the ellipsoid you get when you hit the unit sphere with A:

```
Clear[a];
a[i_, j_] := ((-1)^Random[Integer,{0,1}]) Random[Real, {0.5, 1.5}]

A = {{a[1, 1]  a[1, 2]  a[1, 3]},
     {a[2, 1]  a[2, 2]  a[2, 3]},
     {a[3, 1]  a[3, 2]  a[3, 3]}};
ellipsoidplot = ParametricPlot3D[
   A.{Sin[s] Cos[t], Sin[s] Sin[t], Cos[s]}, {s, 0, π}, {t, 0, 2 π},
   ViewPoint -> CMView, Axes -> True, AxesLabel -> {"x", "y", "z"}];
```



Rerun a couple of times until you get a nice inviting ellipse.

The area enclosed by this ellipse measures out to:

```
unitspherevolume = 4 π / 3;

{xstretch, ystretch, zstretch} = SingularValues[A][[2]];
xstretch ystretch zstretch unitspherevolume
11.5489
```

Use the determinant of A to duplicate this calculation.

□Answer:

Just go with |Det[A]| $\frac{4\pi}{3}$:

```
Abs[Det[A]] 4 π / 3
11.5489
```

The reasons:

If the columns of A are positively oriented,then Det[A] = + xstretch ystretch zstretch.

If the columns of A are negatively oriented,then Det[A] = - xstretch ystretch zstretch.

Either way: |Det[A]| = xstretch ystretch zstretch.

□B.5.b.iv) If you interchange the vertical columns of a 3D matrix, you change the sign but not the absolute value of the determinant

Here's a random 3D matrix A:

```
Clear[a];
a[i_, j_] := ((-1)^Random[Integer,{0,1}]) Random[Real, {0.5, 1.5}]

A = {{a[1, 1]  a[1, 2]  a[1, 3]},
     {a[2, 1]  a[2, 2]  a[2, 3]},
     {a[3, 1]  a[3, 2]  a[3, 3]}};
MatrixForm[A]
```

$$\begin{pmatrix} 1.46898 & 1.44417 & 1.10583 \\ -1.34907 & -0.516976 & -1.04389 \\ -1.34446 & 1.24295 & -1.01325 \end{pmatrix}$$

Here's the matrix you get when you interchange the first two columns of A:

```
interchangedA = A.{{0, 1, 0},
                   {1, 0, 0},
                   {0, 0, 1}};
MatrixForm[interchangedA]
```

$$\begin{pmatrix} 1.44417 & 1.46898 & 1.10583 \\ -0.516976 & -1.34907 & -1.04389 \\ 1.24295 & -1.34446 & -1.01325 \end{pmatrix}$$

Calculate the determinants of each:

```
{Det[A], Det[interchangedA]}
{0.105329, -0.105329}
```

The result:

Det[interchangedA] = - Det[A].

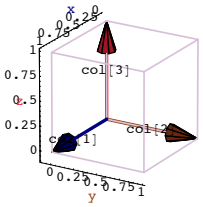
Explain why it had to turn out this way

□Answer:

Go with the unit cube in 3D:

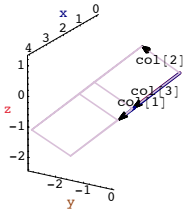
```
Clear[hitboxplotter, r, s, t, pointcolor, matrix];
ranger = 4;

thickness = 0.01;
color = Thistle;
hitboxplotter[matrix_] :=
 Show[Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0}, {0, 0, 0} + matrix.{1, 0, 0},
      {0, 0, 0} + matrix.{1, 0, 0} + matrix.{0, 1, 0},
      {0, 0, 0} + matrix.{0, 1, 0}, {0, 0, 0}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{0, 0, 1}, {0, 0, 0} + matrix.{0, 0, 1} +
       matrix.{1, 0, 0}, {0, 0, 0} + matrix.{0, 0, 1} + matrix.{1, 0, 0} +
       matrix.{0, 1, 0}, {0, 0, 0} + matrix.{0, 0, 1} + matrix.{0, 1, 0},
      {0, 0, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0}, {0, 0, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{1, 0, 0},
      {0, 0, 0} + matrix.{1, 0, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{1, 0, 0} + matrix.{0, 1, 0}, {0, 0, 0} +
       matrix.{1, 0, 0} + matrix.{0, 1, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{0, 1, 0},
      {0, 0, 0} + matrix.{0, 1, 0} + matrix.{0, 0, 1}}]}],
  Axes -> True, AxesLabel -> {"x", "y", "z"},
  Boxed -> False,
  PlotRange -> All,
  ViewPoint -> 12 CMView,
  DisplayFunction -> Identity];
Clear[colplotter, matrix];
colplotter[matrix_] :=
 Show[Arrow[matrix.{1, 0, 0}, Tail → {0, 0, 0},
   VectorColor → NavyBlue, HeadSize → 0.3, ShaftWidth -> 0.02],
  Arrow[matrix.{0, 1, 0}, Tail → {0, 0, 0}, VectorColor → MarsOrange,
   HeadSize → 0.3, ShaftWidth -> 0.02],
  Arrow[matrix.{0, 0, 1}, Tail → {0, 0, 0},
   VectorColor → AlizarinCrimson, HeadSize → 0.3, ShaftWidth -> 0.02],
  Graphics3D[{Text["col[1]", 0.6 matrix.{1, 0, 0}]}],
  Graphics3D[{Text["col[2]", 0.5 matrix.{0, 1, 0}]}],
```

```
  Graphics3D[{Text["col[3]", 0.5 matrix.{0, 0, 1}]}], Axes -> True,
  AxesLabel -> {"x", "y", "z"}, Boxed -> False, PlotRange -> All,
  ViewPoint -> CMView, DisplayFunction -> Identity];

Show[hitboxplotter[IdentityMatrix[3]],
  colplotter[IdentityMatrix[3]],
  DisplayFunction -> $DisplayFunction];
```



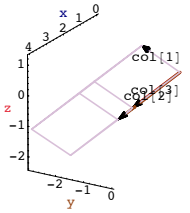
Here's the parallelogram box you get when you hit everything with A:

```
Ahit = Show[hitboxplotter[A],
   colplotter[A], DisplayFunction -> $DisplayFunction];
```



Here's the parallelogram box you get when you hit everything with interchangedA:

```
interchangedAhit = Show[hitboxplotter[interchangedA],
   colplotter[interchangedA], DisplayFunction -> $DisplayFunction];
```

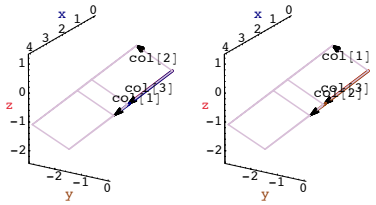


Grab and animate both plots.

That's the parallelogram box defined by the columns of interchangedA.

Notice that both parallelograms are the same physical parallelograms.

```
Show[GraphicsArray[{Ahit, interchangedAhit}]];
```



Reason: They are both defined by the same vectors.

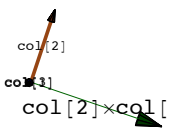
So

enclosed volume = |Det[A]| = |Det[interchangedA]|.

But the orientation of the columns of interchangedA is opposite to the orientation of the columns of A:

```
Show[colplotter[A], Arrow[(A.{0, 1, 0}) × (A.{0, 0, 1}),
  Tail -> {0, 0, 0}, VectorColor -> GosiaGreen],
 Graphics3D[Text["col[2]×col[3]",
   0.6 ((A.{0, 1, 0}) × (A.{0, 0, 1}))]],
 ViewPoint -> 5 (A.{0, 0, 1}), Axes -> False,
 PlotLabel -> "A cols and col[2]×col[3] ",
 DisplayFunction -> $DisplayFunction];
```

ls and col[2]×col[

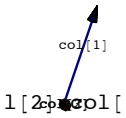



```
(A.{1, 0, 0}).((A.{0, 1, 0}) × (A.{0, 0, 1}))
```

0.105329

```
Show[colplotter[interchangedA],
 Arrow[(interchangedA.{0, 1, 0}) × (interchangedA.{0, 0, 1}),
  Tail -> {0, 0, 0}, VectorColor -> GosiaGreen],
 Graphics3D[Text["col[2]×col[3]",
   0.6 ((interchangedA.{0, 1, 0}) × (interchangedA.{0, 0, 1}))]],
  ViewPoint -> 5 (A.{0, 0, 1}), Axes -> False,
  PlotLabel -> "interchangedA cols and col[2]×col[3] ",
  DisplayFunction -> $DisplayFunction];
```

cols and col[2



l[2]×col[

```
(interchangedA.{1, 0, 0}).
 ((interchangedA.{0, 1, 0}) × (interchangedA.{0, 0, 1}))
```

-0.105329

So Det[A] and Det[interchangedA] have opposite signs but the same absolute value.

□B.5.c) If A is a 3D matrix, then

Det[A] = col[1].(col[2]×col[3])= (A.{1,0,0}).((A.{0,1,0})× (A.{0,0,1}))

Explain this formula known world wide:

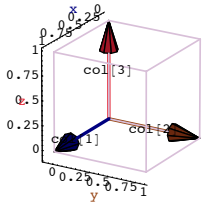
If A is a 3D matrix, then

Det[A] = col[1].(col[2]×col[3]) = (A.{1,0,0}).((A.{0,1,0})× (A.{0,0,1})).

□Answer:

Go with the unit cube in 3D:

```
Clear[hitboxplotter, r, s, t, pointcolor, matrix];
ranger = 4;

thickness = 0.01;
color = Thistle;
hitboxplotter[matrix_] :=
 Show[Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0}, {0, 0, 0} + matrix.{1, 0, 0},
      {0, 0, 0} + matrix.{1, 0, 0} + matrix.{0, 1, 0},
      {0, 0, 0} + matrix.{0, 1, 0}, {0, 0, 0}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{0, 0, 1}, {0, 0, 0} + matrix.{0, 0, 1} +
       matrix.{1, 0, 0}, {0, 0, 0} + matrix.{0, 0, 1} + matrix.{1, 0, 0} +
       matrix.{0, 1, 0}, {0, 0, 0} + matrix.{0, 0, 1} + matrix.{0, 1, 0},
      {0, 0, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0}, {0, 0, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{1, 0, 0},
      {0, 0, 0} + matrix.{1, 0, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{1, 0, 0} + matrix.{0, 1, 0}, {0, 0, 0} +
       matrix.{1, 0, 0} + matrix.{0, 1, 0} + matrix.{0, 0, 1}}]}],
  Graphics3D[{Thickness[thickness], color,
    Line[{{0, 0, 0} + matrix.{0, 1, 0},
      {0, 0, 0} + matrix.{0, 1, 0} + matrix.{0, 0, 1}}]}],
  Axes -> True, AxesLabel -> {"x", "y", "z"},
  Boxed -> False,
  PlotRange -> All,
  ViewPoint -> 12 CMView,
  DisplayFunction -> Identity];
Clear[colplotter, matrix];
colplotter[matrix_] :=
 Show[Arrow[matrix.{1, 0, 0}, Tail → {0, 0, 0},
   VectorColor → NavyBlue, HeadSize → 0.3, ShaftWidth -> 0.02],
  Arrow[matrix.{0, 1, 0}, Tail → {0, 0, 0}, VectorColor → MarsOrange,
   HeadSize → 0.3, ShaftWidth -> 0.02],
  Arrow[matrix.{0, 0, 1}, Tail → {0, 0, 0},
   VectorColor → AlizarinCrimson, HeadSize → 0.3, ShaftWidth -> 0.02],
  Graphics3D[{Text["col[1]", 0.6 matrix.{1, 0, 0}]}],
  Graphics3D[{Text["col[2]", 0.5 matrix.{0, 1, 0}]}],
  Graphics3D[{Text["col[3]", 0.5 matrix.{0, 0, 1}]}], Axes -> True,
  AxesLabel -> {"x", "y", "z"}, Boxed -> False, PlotRange -> All,
  ViewPoint -> CMView, DisplayFunction -> Identity];

Show[hitboxplotter[IdentityMatrix[3]],
  colplotter[IdentityMatrix[3]],
  DisplayFunction -> $DisplayFunction];
```



The volume of this cube is 1.

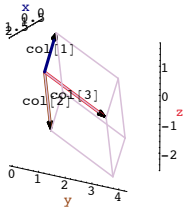
When you hit this with a matrix A, then the resulting box has volume Det[A] if hits with the columns of A are positively oriented, but the resulting box has volume -Det[A] if A are

positively oriented.

Case1: The columns of A are positively oriented

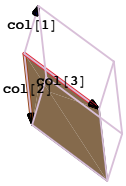
Hit unit cube with the matrix A defined in the code below:

```
A = {{0.6, 1.1, 0.3}, {0.8, 0.9, 2.5}, {1.7, -1.5, -1.0}};
hitcube = Show[hitboxplotter[A],
  colplotter[A], DisplayFunction -> $DisplayFunction];
```



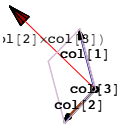
Shade in the base defined by col[2] = A.{0,1,0} and col[3] = A.{0,0,1} :

```
shadedbase = Show[hitcube, Graphics3D[{SurfaceColor[DarkKhaki],
     Polygon[{{0, 0, 0}, A.{0, 1, 0}, A.{0, 1, 0} + A.{0, 0, 1},
       A.{0, 0, 1} }]}], hitcube, Axes -> False];
```



Now look at the same thing from a view point perpendicular to the shaded base and together with a plot of (col[2]×col[3]) :

```
Show[shadedbase, Arrow[(A.{0, 1, 0}) × (A.{0, 0, 1}),
  Tail -> {0, 0, 0}, VectorColor -> Red],
 Graphics3D[Text["(col[2]×col[3])",
   0.6 ((A.{0, 1, 0}) × (A.{0, 0, 1}))]],
 ViewPoint -> 5 (A.{0, 0, 1}), Axes -> False];
```



As you can see, col[1] .(col[2]×col[3]) > 0, so the columns of A are positively oriented and so

Det[A] = volume of the box.

To see why

Det[A] = col[1] .(col[2]×col[3]),

all you have to do is to see why

col[1] .(col[2]×col[3]) = volume of the box.

To this end, look at

col[1] .(col[2]×col[3])

= ||col[1]|| ||.(col[2]×col[3])|| Cos[angle between col[1] and col[2]×col[3]]

= ||col[1]|| Cos[angle between col[1] and col[2]×col[3]] ||.(col[2]×col[3])||

= (height of box) ||(col[2]×col[3]||

= (height of box) ||col[2]|| ||col[3])|| |Sin[angle from co1[2] to col[3]]|

= (height of box) (area of base)

= volume of box.

This explains why

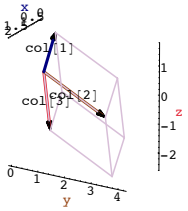
Det[A] = col[1] .(col[2]×col[3])

in the case that the columns of A are positively oriented.

Case2: The columns of A are negatively oriented

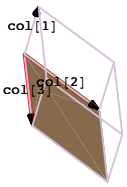
Hit unit cube with the matrix A defined in the code below:

```
A = {{0.6, 0.3, 1.1}, {0.8, 2.5, 0.9}, {1.7, -1.0, -1.5}};
hitcube = Show[hitboxplotter[A],
  colplotter[A], DisplayFunction -> $DisplayFunction];
```



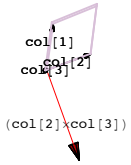
Shade in the base defined by col[2] = A.{0,1,0} and col[3] = A.{0,0,1} :

```
shadedbase = Show[hitcube, Graphics3D[{SurfaceColor[DarkKhaki],
     Polygon[{{0, 0, 0}, A.{0, 1, 0}, A.{0, 1, 0} + A.{0, 0, 1},
       A.{0, 0, 1} }]}], hitcube, Axes -> False];
```



Now look at the same thing from a view point perpendicular to the shaded base and together with a plot of (col[2]×col[3]) :

```
Show[shadedbase, Arrow[(A.{0, 1, 0}) × (A.{0, 0, 1}),
  Tail -> {0, 0, 0}, VectorColor -> Red],
 Graphics3D[Text["(col[2]×col[3])",
   0.6 ((A.{0, 1, 0}) × (A.{0, 0, 1}))]],
 ViewPoint -> 5 (A.{0, 0, 1}), Axes -> False];
```



As you can see, col[1] .(col[2]×col[3]) < 0, the columns of A are negatively oriented and

Det[A] = – volume of the box.

To see why

Det[A] = col[1] .(col[2]×col[3]),

all you have to do is to see why

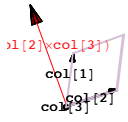
– col[1] .(col[2]×col[3]) = volume of the box.

To this end, notice that

– col[1] .(col[2]×col[3]) = col[1] .(–(col[2] × col[3]))

and take a look at the box from a view point perpendicular to the shaded base and together with a plot of –(col[2] × col[3]) :

```
Show[shadedbase, Arrow[-((A.{0, 1, 0}) × (A.{0, 0, 1}))],
  Tail -> {0, 0, 0}, VectorColor -> Red],
 Graphics3D[Text["-(col[2]×col[3])",
   -0.6 ((A.{0, 1, 0}) × (A.{0, 0, 1}))]],
 ViewPoint -> 5 (A.{0, 0, 1}), Axes -> False];
```



And now it all unravels:

– col[1] .(col[2]×col[3]) = col[1] .(–(col[2] × col[3]))

= ||col[1]|| ||.-(col[2]×col[3])|| Cos[angle between col[1] and -(col[2]×col[3])]

= ‖col[1]‖ Cos[angle between col[1] and -col[2]×col[3]] ‖-(col[2]×col[3])‖

= (height of box) ‖-(col[2]×col[3])‖

= (height of box) ‖(col[2]×col[3])‖

= (height of box) ‖col[2]‖ ‖col[3])‖ |Sin[angle from co1[2] to col[3]]|

= (height of box) (area of base)

= volume of box.

This explains why

Det[A] = col[1] .(col[2]×col[3])

the columns of A are negatively oriented.