Matrices, Geometry&*Mathematica*

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

Producer: Bruce Carpenter

Publisher: Math Everywhere, Inc.

MGM.05 3D Matrices

GIVE IT A TRY!

G.1) Aligners, stretchers and hangers in 3D*

□G.1.a.i) Hit with a 3D aligner

Here's the nasty formula for 3D right hand perpendicular frame corresponding to Euler angles r, s, and t:

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

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

Use this frame maker to make a matrix A as follows:

The aligner:

```
{r, s, t} = N[{0.3 π, π/4, π/8}];

{alignerframe[1], alignerframe[2], alignerframe[3]} =
  {perpframe[1, r, s, t],
   perpframe[2, r, s, t], perpframe[3, r, s, t]};

aligner = {alignerframe[1], alignerframe[2], alignerframe[3]};

MatrixForm[aligner]
```

$$\begin{pmatrix} 0.324124 & 0.753451 & 0.572061 \\ -0.906488 & 0.0743918 & 0.415627 \\ 0.270598 & -0.653281 & 0.707107 \end{pmatrix}$$

The stretcher:

```
{xstretch, ystretch, zstretch} = {2.4, 1.5, 0.8};

stretcher = ( xstretch      0          0
              0         ystretch       0
              0             0      zstretch );

MatrixForm[stretcher]
```

$$\begin{pmatrix} 2.4 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 0.8 \end{pmatrix}$$

The hanger:

```
{r, s, t} = N[{0.4, 0.2, 0.2}];

{hangerframe[1], hangerframe[2], hangerframe[3]} =
  {perpframe[1, r, s, t],
   perpframe[2, r, s, t], perpframe[3, r, s, t]};
hanger = Transpose[{hangerframe[1],
    hangerframe[2], hangerframe[3]}];

MatrixForm[hanger]
```

$$\begin{pmatrix} 0.826878 & -0.560995 & 0.0394695 \\ 0.557035 & 0.807342 & -0.194709 \\ 0.0773655 & 0.182987 & 0.980067 \end{pmatrix}$$

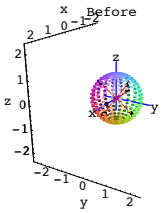
Now put

A = hanger.stretcher.aligner

and see the aligner frame and its negatives together with lots of color-coded points on the unit sphere of 3D:

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

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 = Max[{xstretch, ystretch, zstretch, 1}];
pointcolor[s_, t_] =
  RGBColor[0.5 (x[s, t] + 1), 0.5 (y[s, t] + 1), 0.5 (z[s, t] + 1)];
sjump = (shigh - slow)/18;
tjump = (thigh - tlow)/18;

Clear[hitpointplotter, hitframeplotter, matrix3D];
hitpointplotter[matrix3D_] :=
  Table[Graphics3D[{pointcolor[s, t], PointSize[0.015],
     Point[matrix3D.{x[s, t], y[s, t], z[s, t]}]}],
   {s, slow, shigh - sjump, sjump}, {t, tlow, thigh - tjump, tjump}];

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}]};

before = Show[hitpointplotter[IdentityMatrix[3]],
   hitframeplotter[IdentityMatrix[3]], Axes3D[1.5], PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False,
   ViewPoint → CMView, PlotLabel → "Before",
   DisplayFunction → $DisplayFunction];
```

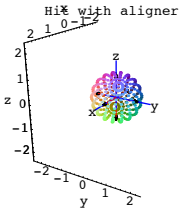
$$\begin{pmatrix} 1.41457 & 1.412 & 0.80784 \\ -0.706602 & 1.19913 & 1.15796 \\ 0.0235331 & -0.35189 & 0.774709 \end{pmatrix}$$



This shows points on the surface of the 3D unit sphere and the aligner frame defined above.

Now look at what you get when you hit with the aligner matrix:

```
Show[hitpointplotter[aligner],
  hitframeplotter[aligner], Axes3D[1.5], PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False,
  ViewPoint → CMView, PlotLabel → "Hit with aligner",
  DisplayFunction → $DisplayFunction];
```

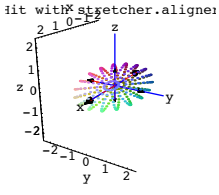


Describe qualitatively what happened to the points and the aligner frame.

□G.1.a.ii) Hitting with the aligner and then with the stretcher

Now look at what you get when you hit the unit sphere with the aligner matrix and then with the stretcher:

```
Show[hitpointplotter[stretcher.aligner],
  hitframeplotter[stretcher.aligner], Axes3D[ranger], PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False,
  ViewPoint → CMView, PlotLabel → "Hit with stretcher.aligner",
  DisplayFunction → $DisplayFunction];
```



Describe qualitatively what happened to the points and the frames.

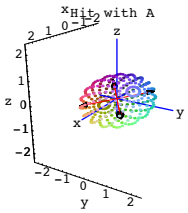
□G.1.a.iii) Hitting with A = hanger.stretcher.aligner

Now look at what you get when you hit with

A = hanger.stretcher.aligner

which is the same as hitting with the aligner, then the stretcher, and then the hanger:

```
Show[ hitpointplotter[hanger.stretcher.aligner],
  hitframeplotter[hanger.stretcher.aligner], Axes3D[ranger],
PlotRange -> {{-ranger, ranger}, {-ranger, ranger},
    {-ranger, ranger}}, Axes -> True,
    AxesLabel -> {"x", "y", "z"}, Boxed -> False, ViewPoint -> CMView,
    PlotLabel -> "Hit with A",
    DisplayFunction -> $DisplayFunction];
```



Describe qualitatively what happened to the points and the frames.

□G.1.b.i) Doing SVD for a 3D matrix

Here's a new 3D matrix:

```
A = {{-1.05, 1.53, -2.12}, {-1.56, -0.32, 1.09}, {1.31, 0.94, 0.84}};
MatrixForm[A]
```

$$\begin{pmatrix} -1.05 & 1.53 & -2.12 \\ -1.56 & -0.32 & 1.09 \\ 1.31 & 0.94 & 0.84 \end{pmatrix}$$

Do an SVD analysis of A to determine the aligner, the stretch factors and the hanger for A.

□G.1.b.ii) Actions of aligner, stretcher.aligner and hanger.stretcher.aligner

Copy and paste and edit the parts of part a) above to make plots showing the successive actions of the hit with aligner, then the hit with stretcher.aligner, and then finally the hit with

A = hanger.stretcher. aligner.

□G.1.c.i) aligner^t.stretcher.hanger^t

Here's a random 3D matrix A:

```
Clear[i, j];
dim = 3;
A = Table[Random[Real, {-8, 8}], {i, 1, dim}, {j, 1, dim}];
MatrixForm[A]
```

$$\begin{pmatrix} -6.8951 & 7.75656 & 0.317847 \\ -1.39481 & 5.05275 & 0.770465 \\ -7.83927 & -4.49302 & -1.67887 \end{pmatrix}$$

SVD analyze it:

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

$$\begin{pmatrix} -0.892497 & -0.115999 & -0.435882 \\ -0.384706 & -0.308742 & 0.869873 \\ -0.23548 & 0.944046 & 0.230926 \end{pmatrix}$$

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

$$\begin{pmatrix} 11.5706 & 0 & 0 \\ 0 & 9.29719 & 0 \\ 0 & 0 & 0.146958 \end{pmatrix}$$

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

$$\begin{pmatrix} 0.737772 & -0.674861 & -0.0159664 \\ -0.66366 & -0.720795 & -0.200026 \\ -0.123481 & -0.15817 & 0.979661 \end{pmatrix}$$

Check:

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

$$\begin{pmatrix} -6.8951 & 7.75656 & 0.317847 \\ -1.39481 & 5.05275 & 0.770465 \\ -7.83927 & -4.49302 & -1.67887 \end{pmatrix}$$

$$\begin{pmatrix} -6.8951 & 7.75656 & 0.317847 \\ -1.39481 & 5.05275 & 0.770465 \\ -7.83927 & -4.49302 & -1.67887 \end{pmatrix}$$

That checks:

Now look at this new matrix

aligner^t.stretcher.hanger^t:

```
new = Transpose[aligner].stretcher.Transpose[hanger];
MatrixForm[new]
```

$$\begin{pmatrix} -6.8951 & -1.39481 & -7.83927 \\ 7.75656 & 5.05275 & -4.49302 \\ 0.317847 & 0.770465 & -1.67887 \end{pmatrix}$$

Compare with A and A^t:

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

$$\begin{pmatrix} -6.8951 & 7.75656 & 0.317847 \\ -1.39481 & 5.05275 & 0.770465 \\ -7.83927 & -4.49302 & -1.67887 \end{pmatrix}$$

$$\begin{pmatrix} -6.8951 & -1.39481 & -7.83927 \\ 7.75656 & 5.05275 & -4.49302 \\ 0.317847 & 0.770465 & -1.67887 \end{pmatrix}$$

How is the new matrix related to A?

Put answer here.

How is the hanger frame for the new matrix related to the aligner frame for A?

Put answer here.

How is the aligner frame for the new matrix related to the hanger frame for A?

Put answer here.

How are the stretch factors for the new matrix related to the stretch factors of A?

Put answer here.

Explain why the following outputs are guaranteed to be identical:

```
Det[A]
```

```
-15.8088
```

```
Det[new]
```

```
-15.8088
```

□G.1.d.i) aligner^t.unstretcher.hanger^t

Here's a new random 3D matrix A:

```
Clear[i, j];
dim = 3;
A = Table[Random[Real, {-4, 4}], {i, 1, dim}, {j, 1, dim}];
MatrixForm[A]
```

$$\begin{pmatrix} 0.342777 & -0.809344 & -3.59074 \\ 1.24767 & -2.1414 & 0.879534 \\ 2.33674 & 3.54388 & -1.36069 \end{pmatrix}$$

SVD analyze it:

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

$$\begin{pmatrix} -0.302496 & -0.9531 & 0.00985036 \\ 0.341723 & -0.118092 & -0.932352 \\ -0.889788 & 0.278666 & -0.361418 \end{pmatrix}$$

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

$$\begin{pmatrix} 4.80454 & 0 & 0 \\ 0 & 3.56626 & 0 \\ 0 & 0 & 2.15659 \end{pmatrix}$$

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

$$\begin{pmatrix} -0.365599 & -0.757667 & 0.540627 \\ 0.0496686 & 0.564128 & 0.824192 \\ -0.929446 & 0.328176 & -0.168613 \end{pmatrix}$$

Check:

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

$$\begin{pmatrix} 0.342777 & -0.809344 & -3.59074 \\ 1.24767 & -2.1414 & 0.879534 \\ 2.33674 & 3.54388 & -1.36069 \end{pmatrix}$$

$$\begin{pmatrix} 0.342777 & -0.809344 & -3.59074 \\ 1.24767 & -2.1414 & 0.879534 \\ 2.33674 & 3.54388 & -1.36069 \end{pmatrix}$$

That checks:

Now look at this new matrix

aligner^t.unstretcher.hanger^t:

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

unstretcher = DiagonalMatrix[{1/xstretch, 1/ystretch, 1/zstretch}];
new = Transpose[aligner].unstretcher.Transpose[hanger];

MatrixForm[new]
```

$$\begin{pmatrix} 0.00549879 & 0.374177 & 0.227353 \\ -0.101564 & -0.214449 & 0.1294 \\ -0.255077 & 0.0840558 & -0.00746302 \end{pmatrix}$$

Compare with A and A^{-1}:

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

$$\begin{pmatrix} 0.342777 & -0.809344 & -3.59074 \\ 1.24767 & -2.1414 & 0.879534 \\ 2.33674 & 3.54388 & -1.36069 \end{pmatrix}$$

$$\begin{pmatrix} 0.00549879 & 0.374177 & 0.227353 \\ -0.101564 & -0.214449 & 0.1294 \\ -0.255077 & 0.0840558 & -0.00746302 \end{pmatrix}$$

How is the new matrix related to A?

Put answer here.

How is the hanger frame for the new matrix related to the aligner frame for A?
Put answer here.

How is the aligner frame for the new matrix related to the hanger frame for A?
Put answer here.

How are the stretch factors for the new matrix related to the stretch factors of A?
Put answer here.

Explain why the following outputs are guaranteed to be identical:

```
1/Det[A]
```

```
-0.0270625
```

```
Det[new]
```

```
-0.0270625
```

□G.1.e) Making the inverse matrix

Here's yet another 3D matrix:

```
A = {{3.26, -4.54, 0.00}, {2.73, 0.89, -3.81}, {9.33, 8.58, 0.82}};
MatrixForm[A]
```

$$\begin{pmatrix} 3.26 & -4.54 & 0 \\ 2.73 & 0.89 & -3.81 \\ 9.33 & 8.58 & 0.82 \end{pmatrix}$$

Do SVD analysis of A by writing A in the form
A = hanger. stretcher. aligner:

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

$$\begin{pmatrix} 0.0539764 & 0.93153 & 0.359635 \\ -0.200221 & 0.362942 & -0.910046 \\ -0.978263 & -0.0228853 & 0.206102 \end{pmatrix}$$

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

$$\begin{pmatrix} 12.9594 & 0 & 0 \\ 0 & 5.77427 & 0 \\ 0 & 0 & 3.7484 \end{pmatrix}$$

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

$$\begin{pmatrix} -0.732893 & -0.680337 & -0.00303526 \\ 0.660534 & -0.710477 & -0.242728 \\ 0.16298 & -0.179899 & 0.97009 \end{pmatrix}$$

Check:

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

$$\begin{pmatrix} 3.26 & -4.54 & 0 \\ 2.73 & 0.89 & -3.81 \\ 9.33 & 8.58 & 0.82 \end{pmatrix}$$

$$\begin{pmatrix} 3.26 & -4.54 & 0 \\ 2.73 & 0.89 & -3.81 \\ 9.33 & 8.58 & 0.82 \end{pmatrix}$$

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

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

$$\begin{pmatrix} 0.119145 & 0.0132722 & 0.0616672 \\ -0.134711 & 0.00953026 & 0.0442809 \\ 0.0539035 & -0.250731 & 0.0545306 \end{pmatrix}$$

Use what you see above to explain where *Mathematica*'s calculation of A^{-1} comes from.

G.2) Rank of a 3D Matrix*

□G.2.a.i) Hit and tell: The question of rank

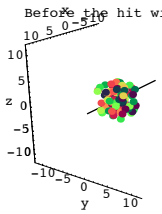
Here's a 3D matrix:

```
A = {{0.1, -0.35, 0.4}, {0.3, -1.05, 1.2}, {0.2, -0.7, 0.8}};

MatrixForm[A]
```

$$\begin{pmatrix} 0.1 & -0.35 & 0.4 \\ 0.3 & -1.05 & 1.2 \\ 0.2 & -0.7 & 0.8 \end{pmatrix}$$

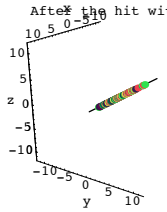
Here's a whole bunch of random points shown together with a certain line:

```
Clear[hangerframe, k];
hangerframe[k_] := SingularValues[A][[1, k]];
b = 12;
lineplot =
  Graphics3D[Line[{-b hangerframe[1], +b hangerframe[1]}]];
a = 3;
points = Table[{Random[Real, {-a, a}],
    Random[Real, {-a, a}], Random[Real, {-a, a}]}, {k, 1, 100}];

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]}];
ranger = b;

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



Here's what happens to those same points after you hit them with A:

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

after = Show[hitpointplot, lineplot, 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"];
```



Describe what you see and explain why you see it. Explain in full why all the points landed on the plotted line.
What is the rank of A?

□G.2.a.ii) Hit and tell: SVD analysis

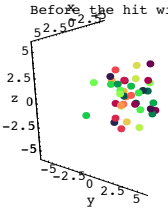
Here's a new 3D matrix:

```
A = {{0.50, 0.60, -1.20}, {-0.20, -0.24, 0.48}, {-0.55, -0.66, 1.32}};

MatrixForm[A]
```

$$\begin{pmatrix} 0.5 & 0.6 & -1.2 \\ -0.2 & -0.24 & 0.48 \\ -0.55 & -0.66 & 1.32 \end{pmatrix}$$

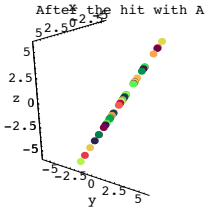
Here are a whole bunch of random points:

```
a = 3;
b = 6;
points = Table[{Random[Real, {-a, a}],
    Random[Real, {-a, a}], Random[Real, {-a, a}]}, {k, 1, 40}];

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]}];

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



Here's what happens to those same points after you hit them with A:

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

after = Show[hitpointplot, 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"];
```



Evidently the hit with A sent all the points onto a line through {0,0,0}. SVD analyze A to determine why this happened.

Use your analysis to express this line in terms of one of the hanger frames of A and then to add the plot of this line to the plot immediately above.

□G.2.b.i) Hit and tell: All hits land on a plane

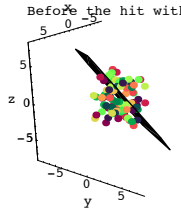
Here's a 3D matrix:

```
A = {{1.5, -0.35, 0.4}, {0.5, -1.05, 1.2}, {1.1, 0.49, -0.56}};

MatrixForm[A]
```

$$\begin{pmatrix} 1.5 & -0.35 & 0.4 \\ 0.5 & -1.05 & 1.2 \\ 1.1 & 0.49 & -0.56 \end{pmatrix}$$

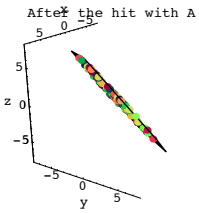
Here are a whole bunch of random points shown together with a certain plane:

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

b = 8;
{slow, shigh} = {-b, b};
{tlow, thigh} = {-b, b};

Clear[planeplotter, s, t];

planeplotter[s_, t_] = s hangerframe[1] + t hangerframe[2];
planeplot = ParametricPlot3D[planeplotter[s, t],
   {s, slow, shigh}, {t, tlow, thigh}, PlotRange → All,
   PlotPoints → {2, 2}, DisplayFunction → Identity];

a = 3;
points = Table[{Random[Real, {-a, a}],
    Random[Real, {-a, a}], Random[Real, {-a, a}]}, {k, 1, 100}];
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]}];

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



Here's what happens to those same points after you hit them with A:

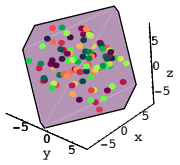
```
hitpointplot = Table[Graphics3D[{PointSize[0.04], pointcolor[k],
     Point[A.points[[k]]]}], {k, 1, Length[points]}];

after = Show[hitpointplot, planeplot, 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 hit points from a different view point:

```
Show[after, ViewPoint → 10 hangerframe[1] × hangerframe[2]];
```

After the hit with A



Describe what you see and explain why you see it. Use SVD analysis to help to explain in full why all the points landed on the plotted plane.
What is the rank of A?

□G.2.b.ii) Hit and tell: All hits land on a plane

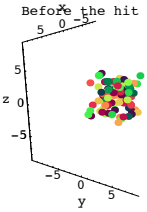
Here's a new 3D matrix:

```
A = {{0.5, -0.7, -1.3}, {1.6, -0.8, -0.2}, {1.3, -1.1, -1.4}};

MatrixForm[A]
```

$$\begin{pmatrix} 0.5 & -0.7 & -1.3 \\ 1.6 & -0.8 & -0.2 \\ 1.3 & -1.1 & -1.4 \end{pmatrix}$$

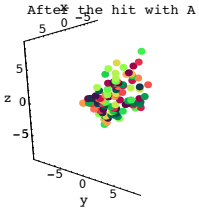
Here are a whole bunch of random points:

```
a = 3;
b = 9;
points = Table[{Random[Real, {-a, a}],
    Random[Real, {-a, a}], Random[Real, {-a, a}]}, {k, 1, 100}];
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]}];

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



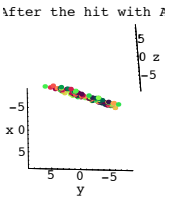
Here's what happens to those same points after you hit them with A:

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

after = Show[hitpointplot, 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 them from a different viewpoint:

```
Show[after, ViewPoint → 8 {-0.65, 0.02, -0.76}];
```



Evidently the hit with A sent all the points onto a plane through {0,0,0}. SVD analyze A to determine why this happened.

Use your analysis to express this plane in terms of two of the hanger frames of A and then to add the plot of this plane to the plot immediately above.

□G.2.b.iii) Hit and tell

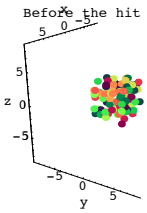
Here's a new 3D matrix:

```
A = {{0.5, 1.7, -1.3}, {-1.6, -0.8, -0.2}, {1.3, -1.1, -1.7}};

MatrixForm[A]
```

$$\begin{pmatrix} 0.5 & 1.7 & -1.3 \\ -1.6 & -0.8 & -0.2 \\ 1.3 & -1.1 & -1.7 \end{pmatrix}$$

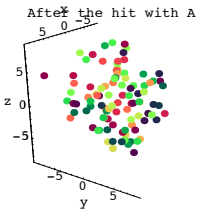
Here are a whole bunch of random points:

```
a = 3;
b = 9;
points = Table[{Random[Real, {-a, a}],
    Random[Real, {-a, a}], Random[Real, {-a, a}]}, {k, 1, 100}];
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]}];

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



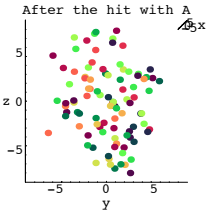
Here's what happens to those same points after you hit them with A:

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

after = Show[hitpointplot, 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"];
```



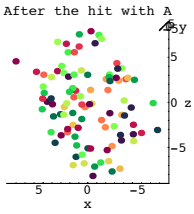
See them from a three different viewpoints:
Down the x-axis:

```
Show[after, ViewPoint → 8 {1, 0, 0}];
```



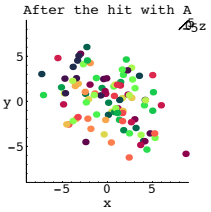
Down the y-axis:

```
Show[after, ViewPoint → 8 {0, 1, 0}];
```



Down the z-axis:

```
Show[after, ViewPoint → 8 {0, 0, 1}];
```



Evidently the hit with A did not send the points onto a line or a plane. SVD analyze A to determine why this happened. What is the rank of A?

□G.2.c) Determine the rank

Here's a 3D matrix A:

```
A = {{0.4, -0.8, -1.2}, {-0.5, 1.0, 1.5}, {-0.2, 0.4, 0.6}};

MatrixForm[A]
```

$$\begin{pmatrix} 0.4 & -0.8 & -1.2 \\ -0.5 & 1. & 1.5 \\ -0.2 & 0.4 & 0.6 \end{pmatrix}$$

Determine the rank of A.
Put answer here.

Here's another 3D matrix A:

```
A = {{1.359, 1.57, 0.705}, {1.9, -3.7, 1.0}, {2.718, 3.14, 1.41}};

MatrixForm[A]
```

$$\begin{pmatrix} 1.359 & 1.57 & 0.705 \\ 1.9 & -3.7 & 1. \\ 2.718 & 3.14 & 1.41 \end{pmatrix}$$

Determine the rank of A.
Put answer here.

Here's yet another 3D matrix A:

```
A = {{1.24, -1.16, 1.30}, {1.91, 1.77, 1.95}, {2.31, -2.52, 0.58}};

MatrixForm[A]
```

$$\begin{pmatrix} 1.24 & -1.16 & 1.3 \\ 1.91 & 1.77 & 1.95 \\ 2.31 & -2.52 & 0.58 \end{pmatrix}$$

Determine the rank of A.

Put answer here.

□G.2.d.i) Saying that the rank of a 3D matrix A is 3 is the same as saying that Det[A] ≠ 0.

Explain this:

Saying that the rank of a 3D matrix A is 3 is the same as saying that Det[A] ≠ 0.

□G.2.d.ii) Can you determine the rank of a 3D matrix A merely by looking at Det[A]?

Can you determine the rank of a 3D matrix A merely by looking at Det[A]?

Why or why not?

□G.2.d.iii) A rank 0 matrix

There is exactly one 3D matrix whose rank is 0.

Which 3D matrix is this sorry excuse for a matrix?

G.3) Linear Algebra: Using 3D matrices to try to solve linear equations*

□G.3.a.i) Using a matrix hit to solve a linear system

You are given the linear system

$$1.3\,x - 0.4\,y + 2.7\,z = 0.8$$
$$0.6\,x + 0.5\,y - 1.3\,z = 1.6$$
$$2.1\,y + 4.7\,z = -0.9$$

to solve for x, y and z.

You enter the coefficient matrix:

```
A = {{1.3, -0.4, 2.7}, {0.6, 0.5, -1.3}, {0.0, 2.1, 4.7}};
MatrixForm[A]
```

$$\begin{pmatrix} 1.3 & -0.4 & 2.7 \\ 0.6 & 0.5 & -1.3 \\ 0 & 2.1 & 4.7 \end{pmatrix}$$

And then you check to see whether A is invertible:

```
Det[A]
```

```
11.134
```

Why does the result tell you that the rank of A is 3?

Why does this tell you that A is invertible?

Why does this news make you very happy?

Use a matrix hit to come up with the x, the y, and the z that solve the given linear system.

□G.3.a.ii) Using a matrix hit to solve a different linear system

Use a matrix hit to come up with the x, the y, and the z that solve the linear system

$$-x - 0.25\,z = 1.13$$
$$0.42\,x + 0.43\,y - 0.34\,z = 1.68$$
$$2.25\,x - 1.54\,y - 1.66\,z = -1.31$$

Check your work.

□G.3.b.i) A not invertible

Here is a new 3D matrix:

```
A = {{-1.65, 1.95, -1.10}, {2.32, 0.98, -1.68}, {0.50, 1.27, -1.28}};
MatrixForm[A]
```

$$\begin{pmatrix} -1.65 & 1.95 & -1.1 \\ 2.32 & 0.98 & -1.68 \\ 0.5 & 1.27 & -1.28 \end{pmatrix}$$

You check to see whether A is invertible:

```
Det[A]
```

```
0
```

Explain why this tells you that the rank of A is either 1 or 2 and that this matrix is not invertible?

□G.3.b.ii) Determine the location of the {u,v,w}'s for which the linear system

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

has at least one solution

Stay with the same non-invertible 3D matrix:

```
A = {{-1.65, 1.95, -1.10}, {2.32, 0.98, -1.68}, {0.50, 1.27, -1.28}};
MatrixForm[A]
```

$$\begin{pmatrix} -1.65 & 1.95 & -1.1 \\ 2.32 & 0.98 & -1.68 \\ 0.5 & 1.27 & -1.28 \end{pmatrix}$$

In spite of the fact that A is not invertible, the linear system:

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

either has many solutions or no solutions depending on where {u,v,w} is located.

Determine the location of the {u,v,w}'s for which the linear system

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

has at least one solution.

□G.3.b.iii) Any solutions?

Stay with the same non-invertible 3D matrix:

```
A = {{-1.65, 1.95, -1.10}, {2.32, 0.98, -1.68}, {0.50, 1.27, -1.28}};
MatrixForm[A]
```

$$\begin{pmatrix} -1.65 & 1.95 & -1.1 \\ 2.32 & 0.98 & -1.68 \\ 0.5 & 1.27 & -1.28 \end{pmatrix}$$

Determine whether the linear system

A.{x, y, z} = {−1.315, −3.86, −2.456}

has any solutions at all.

If it does, then give a sample solution..

□G.3.b.iv) Any solutions?

Stay with the same non-invertible 3D matrix:

```
A = {{-1.65, 1.95, -1.10}, {2.32, 0.98, -1.68}, {0.50, 1.27, -1.28}};
MatrixForm[A]
```

$$\begin{pmatrix} -1.65 & 1.95 & -1.1 \\ 2.32 & 0.98 & -1.68 \\ 0.5 & 1.27 & -1.28 \end{pmatrix}$$

Determine whether the linear system

A.{x, y, z} = {0.336, 0.421, −0.842}

has any solutions at all.

If it does, then give a sample solution..

□G.3.c.i) A new one

Here is a new 3D matrix:

```
A = {{1.86, 2.95, -5.67}, {5.32, 2.98, -2.53}, {0.50, 1.27, -1.28}};
MatrixForm[A]
```

$$\begin{pmatrix} 1.86 & 2.95 & -5.67 \\ 5.32 & 2.98 & -2.53 \\ 0.5 & 1.27 & -1.28 \end{pmatrix}$$

You check:

```
Det[A]
```

```
-14.6223
```

Smiling, you say that for each given {u,v,w} the linear system:

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

has exactly one solution. Are you right?

□G.3.c.ii) Another new one

Here is a new 3D matrix:

```
A = {{0.56, 6.01, -5.82}, {0.00, 1.34, -2.79}, {4.70, -5.39, -1.28}};
MatrixForm[A]
```

$$\begin{pmatrix} 0.56 & 6.01 & -5.82 \\ 0 & 1.34 & -2.79 \\ 4.7 & -5.39 & -1.28 \end{pmatrix}$$

You check:

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

```
{9.57282, 6.34889, 0.847966}
```

Smiling, you say that for each given {u,v,w} the linear system

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

has exactly one solution. You are right.

Why are you?

G.4) Volume and area:Ellipsoids in 3D, 2D ellipses in 3D, and 3D boxes*

□G.4.a.i) Coming up with a perpendicular frame on which an ellipsoid is hung

Here's a 3D matrix:

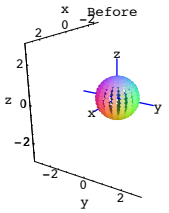
```
A = {{2.0, -0.4, 1.2}, {1.5, -1.5, 0.5}, {-2.3, 0.8, -1.6}};
MatrixForm[A]
```

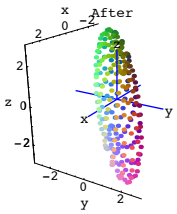
$$\begin{pmatrix} 2. & -0.4 & 1.2 \\ 1.5 & -1.5 & 0.5 \\ -2.3 & 0.8 & -1.6 \end{pmatrix}$$

Watch this matrix hit points on the 3D unit sphere and hang the result in 3D:

```
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 = 3.0;
sjump = (shigh - slow)/16;
tjump = (thigh - tlow)/16;

pointcolor[s_, t_] =
  RGBColor[0.5 (x[s, t] + 1), 0.5 (y[s, t] + 1), 0.5 (z[s, t] + 1)];
threeDpointplot = Table[Graphics3D[{pointcolor[s, t],
     PointSize[0.025], Point[{x[s, t], y[s, t], z[s, t]}]}],
   {s, slow, shigh - sjump, sjump}, {t, tlow, thigh - tjump, tjump}];

before = Show[threeDpointplot,
   ThreeAxes[2], PlotLabel → "Before", PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   Axes → True, Boxed → False, AxesLabel → {"x", "y", "z"},
   ViewPoint → CMView, DisplayFunction → $DisplayFunction];

hitpointplot = Table[Graphics3D[{pointcolor[s, t],
     PointSize[0.025], Point[A.{x[s, t], y[s, t], z[s, t]}]}],
   {s, slow, shigh - sjump, sjump}, {t, tlow, thigh - tjump, tjump}];

after = Show[hitpointplot, ThreeAxes[2.5], PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   Axes → True, AxesLabel → {"x", "y", "z"},
   PlotLabel → "After", ViewPoint → CMView,
   Boxed → False, DisplayFunction → $DisplayFunction];
```





After the hit, the points on the 3D unit sphere have been deposited on the skin of an ellipsoid. In search of the perpendicular frame on which this ellipsoid is hung, you look at an SVD analysis of A:

```
Clear[alignerframe, hangerframe, k];
{alignerframe[1], alignerframe[2], alignerframe[3]} =
 SingularValues[A][[3]]
```

```
{{-0.806735, 0.350922, -0.475429},
 {-0.114603, -0.882206, -0.456705}, {0.579694, 0.313954, -0.751923}}
```

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

```
{4.20621, 1.05611, 0.18009}
```

```
{hangerframe[1], hangerframe[2], hangerframe[3]} =
 SingularValues[A][[1]]
```

```
{{-0.552601, -0.469354, 0.688723},
 {-0.401822, 0.874008, 0.273219}, {0.730186, 0.125763, 0.671574}}
```

At this point, you have enough information identify the perpendicular frame on which this ellipsoid is hung.
Do it.
Make a good plots showing off what you know.

□G.4.a.ii) Measuring the volume of the ellipsoid

Stay with the same setup as in part i) immediately above. Measure the volume enclosed by your ellipsoid.

□G.4.b.i) A matrix that hits on 3D and hangs in 2D

Here's a matrix that hits on 3D and hangs in 2D:

```
A = {{1.8, 0.5, -1.3}, {-0.9, 1.2, 0.8}};
MatrixForm[A]
```

$$\begin{pmatrix} 1.8 & 0.5 & -1.3 \\ -0.9 & 1.2 & 0.8 \end{pmatrix}$$

When you hit this matrix on a 3D Point {x,y,z}, you get a 2D point:

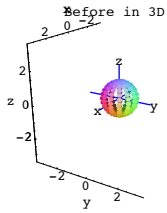
```
Clear[x, y, z];
A.{x, y, z}
```

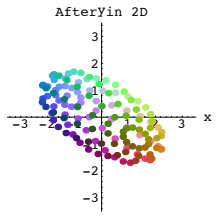
```
{1.8 x + 0.5 y - 1.3 z, -0.9 x + 1.2 y + 0.8 z}
```

Note the two slots in A.{x,y,z}.

Watch this matrix hit points on the 3D unit sphere and hang the result in 2D:

```
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;
sjump = (shigh - slow)/12;
tjump = (thigh - tlow)/12;

pointcolor[s_, t_] =
  RGBColor[0.5 (x[s, t] + 1), 0.5 (y[s, t] + 1), 0.5 (z[s, t] + 1)];
threeDpointplot = Table[Graphics3D[{pointcolor[s, t],
     PointSize[0.025], Point[{x[s, t], y[s, t], z[s, t]}]}],
   {s, slow, shigh - sjump, sjump}, {t, tlow, thigh - tjump, tjump}];

before = Show[threeDpointplot,
   ThreeAxes[2], PlotLabel → "Before in 3D", PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   Axes → True, Boxed → False, AxesLabel → {"x", "y", "z"},
   ViewPoint → CMView, DisplayFunction → $DisplayFunction];

twoDhitpointplot =
  Table[Graphics[{pointcolor[s, t], PointSize[0.035],
     Point[A.{x[s, t], y[s, t], z[s, t]}]}],
   {s, slow, shigh - sjump, sjump}, {t, tlow, thigh - tjump, tjump}];

after = Show[twoDhitpointplot,
   PlotRange → {{-ranger, ranger}, {-ranger, ranger}},
   Axes → True, AxesLabel → {"x", "y"}, PlotLabel → "After in 2D",
   DisplayFunction → $DisplayFunction];
```





After the hit, the points on the 3D unit sphere seem to have been deposited on and inside an ellipse in 2D. Try to identify and plot this ellipse. Frame it up and measure the area enclosed by the ellipse.

□G.4.c.i) Det[A] = 0 tells you that A puts all its hits on a line or on a plane

Here's a 3D matrix:

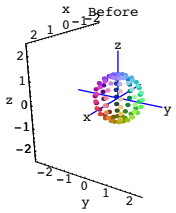
```
A = {{-2.0, 0.4, -1.2}, {-1.5, -1.5, -0.5}, {-1.6, -0.4, -0.8}};
MatrixForm[A]
```

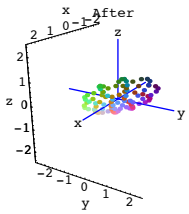
$$\begin{pmatrix} -2. & 0.4 & -1.2 \\ -1.5 & -1.5 & -0.5 \\ -1.6 & -0.4 & -0.8 \end{pmatrix}$$

See this matrix hit points on the 3D unit sphere and hang the result in 3D:

```
A = {{-2.0, 0.4, -1.2}, {-1.5, -1.5, -0.5}, {-1.6, -0.4, -0.8}};
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 = 2.5;
sjump = (shigh - slow)/10;
tjump = (thigh - tlow)/10;

pointcolor[s_, t_] =
  RGBColor[0.5 (x[s, t] + 1), 0.5 (y[s, t] + 1), 0.5 (z[s, t] + 1)];
threeDpointplot = Table[Graphics3D[{pointcolor[s, t],
     PointSize[0.025], Point[{x[s, t], y[s, t], z[s, t]}]}],
```

```
    {s, slow, shigh - sjump, sjump}, {t, tlow, thigh - tjump, tjump}];

before = Show[threeDpointplot,
   ThreeAxes[2], PlotLabel → "Before", PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   Axes → True, Boxed → False, AxesLabel → {"x", "y", "z"},
   ViewPoint → CMView, DisplayFunction → $DisplayFunction];

hitpointplot = Table[Graphics3D[{pointcolor[s, t],
     PointSize[0.025], Point[A.{x[s, t], y[s, t], z[s, t]}]}],
   {s, slow, shigh - sjump, sjump}, {t, tlow, thigh - tjump, tjump}];

after = Show[hitpointplot, ThreeAxes[2.5], PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   Axes → True, AxesLabel → {"x", "y", "z"},
   PlotLabel → "After", ViewPoint → CMView,
   Boxed → False, DisplayFunction → $DisplayFunction];
```





Upon seeing this you check:

```
Det[A]
```

0

Seeing that

Det[A] = 0,

you announce that all the points in the second plot are all on the same plane.
You are right. Explain why you are right.

□G.4.c.ii) SVD gives extra information

Stay with the same matrix A as in part i) immediately above and look at this SVD analysis of A:

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

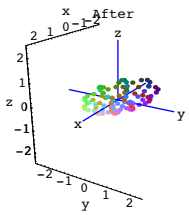
{3.40059, 1.46491}

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

{{-0.63808, -0.549863, -0.538985}, {0.644631, -0.764314, 0.0165902}}

At this point, you have enough information to add a plot of the border of the ellipse to the plot below:

```
Show[after];
```

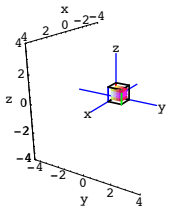


Do it and while you're at it measure the planar area enclosed by this ellipse.

□G.4.d.i) Hitting a 3D matrix A on a cube and measuring the volume of the result

Here's the cube with corners at {0,0,1},{1,0,0},{1,1,0} , etc:

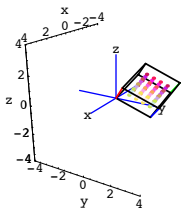
```
jump = 0.2;
Clear[boxplotter, basepoint,
  r, s, t, egde1, edge2, edge3, pointcolor];
ranger = 4;
pointcolor[r_, s_, t_] =
  RGBColor[0.5 (Sin[π r] + 1), 0.5 (Cos[π s] + 1), Sin[π t]^2];

a = 0.01;
boxplotter[basepoint_, edge1_, edge2_, edge3_] :=
  {Table[Graphics3D[{PointSize[0.02], pointcolor[r, s, t],
      Point[basepoint + r edge1 + s edge2 + t edge3]}], {s, jump, 1 - jump,
     jump}, {t, jump, 1 - jump, jump}, {r, jump, 1 - jump, jump}],
   Graphics3D[{Thickness[a], Line[{basepoint, basepoint + edge1,
      basepoint + edge1 + edge2, basepoint + edge2, basepoint}]}],
   Graphics3D[{Thickness[a], Line[{basepoint + edge3,
      basepoint + edge3 + edge1, basepoint + edge3 + edge1 + edge2,
      basepoint + edge3 + edge2, basepoint + edge3}]}], Graphics3D[
    {Thickness[a], Red, Line[{basepoint, basepoint + edge3}]}],
   Graphics3D[{Thickness[a], Blue,
     Line[{basepoint + edge1, basepoint + edge1 + edge3}]}],
   Graphics3D[{Thickness[a], Green, Line[
      {basepoint + edge1 + edge2, basepoint + edge1 + edge2 + edge3}]}],
   Graphics3D[{Thickness[a], Line[{basepoint + edge2,
      basepoint + edge2 + edge3}]}]};

basepoint = {0, 0, 0};
edge1 = {1, 0, 0};
edge2 = {0, 1, 0};
edge3 = {0, 0, 1};

Show[boxplotter[basepoint, edge1, edge2, edge3],
  ThreeAxes[3], PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Boxed → False, ViewPoint → CMView, Axes → True,
  AxesLabel → {"x", "y", "z"}];
```



The volume of this box measures out to 1 cubic unit.
Here's the box and the points after they have been hit with this 3D matrix:

```
A = {{-0.81, -2.73, 1.00},
     {2.21, 0.23, 1.06},
     {-1.22, 0.95, 1.22}};
MatrixForm[A]

Show[boxplotter[A.basepoint, A.edge1, A.edge2, A.edge3],
 ThreeAxes[3], PlotRange →
  {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
 Boxed → False, ViewPoint → CMView, Axes → True,
 AxesLabel → {"x", "y", "z"}];
```

$$\begin{pmatrix} -0.81 & -2.73 & 1. \\ 2.21 & 0.23 & 1.06 \\ -1.22 & 0.95 & 1.22 \end{pmatrix}$$

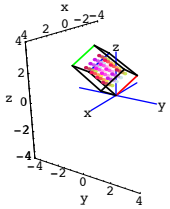


Measure the volume of this box parallelogram.

□G.4.d.ii) Hitting the same cube with A^t and measuring the volume of the result

Here is what you get when you hit the original box in part i) above with A^t, the transpose of A:

```
B = Transpose[A];

Show[boxplotter[B.basepoint, B.edge1, B.edge2, B.edge3],
  ThreeAxes[3], PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Boxed → False, ViewPoint → CMView, Axes → True,
  AxesLabel → {"x", "y", "z"}];
```



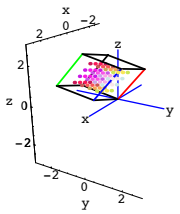
Although this is not the same box as in part i), clued-in matrix folks know that the volume of this box is guaranteed to measure out to the same value as the volume of the box in part i). How do the clued-in matrix folks know this?

□G.4.e.i) Putting the edges of a box parallelogram into the columns of a 3D matrix

Here's a new box with lots of points inside:

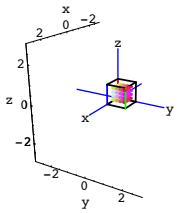
```
ranger = 3.0;
boxbasepoint = {0, 0, 0};
boxedge1 = {1.4, -0.5, 0.3};
boxedge2 = {0.6, -1.7, 0.6};
boxedge3 = {-0.5, 1.2, 1.6};

orginalbox =
  Show[boxplotter[boxbasepoint, boxedge1, boxedge2, boxedge3],
```

```
ThreeAxes[2.5], PlotRange → {{-ranger, ranger},
  {-ranger, ranger}, {-ranger, ranger}}, Boxed → False,
ViewPoint → CMView, Axes → True, AxesLabel → {"x", "y", "z"}];
```



Note that {0,0,0} is one of the corners of this parallelogram box.
Here's the cube with corners at
{0,0,0}, {1,0,0}, {1,1,0}, {1,1,1}, {0,1,1} and {0,0,1}:

```
cubebasepoint = {0, 0, 0};
cubeedge1 = {1, 0, 0};
cubeedge2 = {0, 1, 0};
cubeedge3 = {0, 0, 1};

Show[boxplotter[cubebasepoint, cubeedge1, cubeedge2, cubeedge3],
  ThreeAxes[2.5], PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Boxed → False, ViewPoint → CMView, Axes → True,
  AxesLabel → {"x", "y", "z"}];
```



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

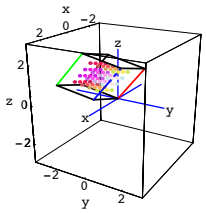
```
A = Transpose[{boxedge1, boxedge2, boxedge3}];
MatrixForm[A]
```

$$\begin{pmatrix} 1.4 & 0.6 & -0.5 \\ -0.5 & -1.7 & 1.2 \\ 0.3 & 0.6 & 1.6 \end{pmatrix}$$

The edges of the box are the vertical columns of A.

Here's what you get when you hit the cube with A:

```
Show[boxplotter[A.cubebasepoint, A.cubeedge1,
   A.cubeedge2, A.cubeedge3], ThreeAxes[2.5], PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  ViewPoint → CMView, Axes → True, AxesLabel → {"x", "y", "z"}];
```



Determine the relationship between this parallelogram box and the original parallelogram box.

Put answer here.

Stay with the same setup as immediately above and look at:

```
Det[A]
```

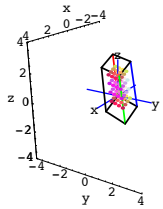
```
-4.225
```

Use the result to measure the volume enclosed by the parallelogram box.

□G.4.e.ii) Make a matrix giving a box

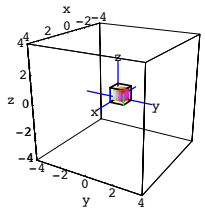
Here's a new parallelogram box:

```
boxbasepoint = {0, 0, 0};
boxedge1 = {1, 2, 0};
boxedge2 = {1.5, 0, -0.6};
boxedge3 = {-2.0, -1.6, 2.2};
ranger = 4;

Show[boxplotter[boxbasepoint, boxedge1, boxedge2, boxedge3],
  ThreeAxes[2.5], PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Boxed → False, ViewPoint → CMView, Axes → True,
  AxesLabel → {"x", "y", "z"}];
```



Note that {0,0,0} is one of the corners of this box.
Here's the cube with corners at
{0,0,0}, {1,0,0}, {1,1,0}, {1,1,1}, {0,1,1} and {0,0,1}:

```
cubebasepoint = {0, 0, 0};
cubeedge1 = {1, 0, 0};
cubeedge2 = {0, 1, 0};
cubeedge3 = {0, 0, 1};

Show[boxplotter[cubebasepoint, cubeedge1, cubeedge2, cubeedge3],
  ThreeAxes[2.5], PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  ViewPoint → CMView, Axes → True, AxesLabel → {"x", "y", "z"}];
```



Make a matrix A so that hitting this cube with A gives the parallelogram box.
Measure the volume of the parallelogram box.

□G.4.e.iii) Defining a box with the columns of a matrix

Given a 3D matrix A, you can use, as above, the columns of A to define edges of a 3D box.
How is Det[A] related to the volume of this box?

□G.4.f.i) Shuffling the columns of a 3D matrix

Here's a random 3D matrix entered by vertical columns:

```
Clear[col];
   col[1] = {Random[Real, {-10, 10}],
  Random[Real, {-10, 10}], Random[Real, {-10, 10}]}
```

```
{-1.07348, -9.84926, -8.18245}
```

```
col[2] = {Random[Real, {-10, 10}],
  Random[Real, {-10, 10}], Random[Real, {-10, 10}]}
```

```
{-3.58601, -8.97089, 9.22757}
```

```
col[3] = {Random[Real, {-10, 10}],
  Random[Real, {-10, 10}], Random[Real, {-10, 10}]}
```

```
{-1.4358, -7.62089, 1.13983}
```

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

$$\begin{pmatrix} -1.07348 & -3.58601 & -1.4358 \\ -9.84926 & -8.97089 & -7.62089 \\ -8.18245 & 9.22757 & 1.13983 \end{pmatrix}$$

Here's the matrix you get by interchanging col[1] and col[2]:

```
shuffledA = Transpose[{col[2], col[1], col[3]}];
MatrixForm[shuffledA]
```

$$\begin{pmatrix} -3.58601 & -1.07348 & -1.4358 \\ -8.97089 & -9.84926 & -7.62089 \\ 9.22757 & -8.18245 & 1.13983 \end{pmatrix}$$

Now look at these calculations of
Det[A] and Det[shuffledA]:

```
Det[A]
```

```
-92.5001
```

```
Det[shuffledA]
```

```
92.5001
```

Got any idea why that happened?

Click on the right for a tip.

The columns of A define a box. So do the columns of shuffledA.

Also look at:

```
MatrixForm[A]
```

$$\begin{pmatrix} -1.07348 & -3.58601 & -1.4358 \\ -9.84926 & -8.97089 & -7.62089 \\ -8.18245 & 9.22757 & 1.13983 \end{pmatrix}$$

```
MatrixForm[shuffledA]
```

$$\begin{pmatrix} -3.58601 & -1.07348 & -1.4358 \\ -8.97089 & -9.84926 & -7.62089 \\ 9.22757 & -8.18245 & 1.13983 \end{pmatrix}$$

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

$$\begin{pmatrix} -3.58601 & -1.07348 & -1.4358 \\ -8.97089 & -9.84926 & -7.62089 \\ 9.22757 & -8.18245 & 1.13983 \end{pmatrix}$$

The columns of A define a box.

So do the columns of shuffledA.

□G.4.f.ii) Shuffling the rows of a 3D matrix

Here's a random 3D matrix entered by horizontal rows:

```
Clear[row];
row[1] = {Random[Real, {-10, 10}],
  Random[Real, {-10, 10}], Random[Real, {-10, 10}]}
```

```
{6.27927, 2.86216, -0.532739}
```

```
row[2] = {Random[Real, {-10, 10}],
  Random[Real, {-10, 10}], Random[Real, {-10, 10}]}
```

```
{-7.13647, -3.19236, 2.18697}
```

```
row[3] = {Random[Real, {-10, 10}],
  Random[Real, {-10, 10}], Random[Real, {-10, 10}]}
```

```
{-6.66372, 3.32166, -3.81677}
```

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

$$\begin{pmatrix} 6.27927 & 2.86216 & -0.532739 \\ -7.13647 & -3.19236 & 2.18697 \\ -6.66372 & 3.32166 & -3.81677 \end{pmatrix}$$

Here's the matrix you get by interchanging row[1] and row[2]:

```
shuffledA = {row[2], row[1], row[3]};
MatrixForm[shuffledA]
```

$$\begin{pmatrix} -7.13647 & -3.19236 & 2.18697 \\ 6.27927 & 2.86216 & -0.532739 \\ -6.66372 & 3.32166 & -3.81677 \end{pmatrix}$$

Now look at these calculations of
Det[A] and Det[shuffledA]:

```
Det[A]
```

```
-64.8154
```

```
Det[shuffledA]
```

```
64.8154
```

Got any idea why that happened?

Click on the right for a tip.

You can use your explanation from the last part.

Just remember

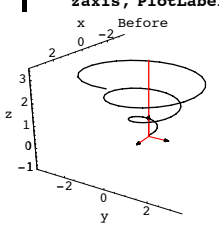
col[k] of A^t is row[k] of A.

G.5) Making matrices to get the job done:
Spinning in 3D; frame stretchers, projections and flips;
bouncing light rays off 3D surfaces; moving frames and wings

□G.5.a.i) Spinning about the z-axis

Here's a helix shown with the usual perpendicular frames pointing out the x, y and z-axes:

```
Clear[perpframe];
{perpframe[1], perpframe[2], perpframe[3]} =
  {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};

frameplot = Table[Arrow[perpframe[k],
    Tail → {0, 0, 0}, VectorColor → Red], {k, 1, 3}];

Clear[x, y, z, s, t];
{x[t_], y[t_], z[t_]} = 0.5 {t Cos[3 t], t Sin[3 t], t};

{tlow, thigh} = {0, 2 π};
ranger = 3.5;
zaxis = Graphics3D[{Red, Line[{{0, 0, 0}, ranger perpframe[3]}]}];

Clear[hitplotter, matrix];
hitplotter[matrix3D_] :=
  ParametricPlot3D[matrix3D.{x[t], y[t], z[t]}, {t, tlow, thigh},
   PlotRange → {{-ranger, ranger}, {-ranger, ranger}, {-1, ranger}},
   BoxRatios → Automatic, Axes → True, AxesLabel → {"x", "y", "z"},
   Boxed → False, ViewPoint → CMView, DisplayFunction → Identity];

surfacebefore = Show[hitplotter[IdentityMatrix[3]], frameplot,
   zaxis, PlotLabel → "Before", DisplayFunction → $DisplayFunction];
```



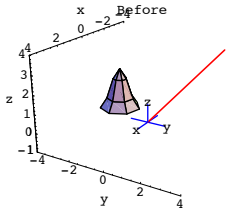
That line is pointing out the z-axis.

Use 3D rotator matrix hits to make a movie showing this helix rotating about the z-axis.

□G.5.a.ii) Spinning about a given line

Here's a bell shown with a line through {0,0,0} in 3D:

```
linedirectionvector = {-0.5, 1.2, 1.4};
newline = Graphics3D[{Red, Thickness[0.007],
    Line[{{0, 0, 0}, 4 linedirectionvector}]}];

Clear[x, y, z, s, t];
{x[s_, t_], y[s_, t_], z[s_, t_]} =
  {1, -1, 2} + {s Cos[t], s Sin[t], Cos[π s]};

{slow, shigh} = {0, 1};
{tlow, thigh} = {0, 2 π};

ranger = 4;
Clear[hitplotter, matrix];
hitplotter[matrix3D_] :=
  ParametricPlot3D[matrix3D.{x[s, t], y[s, t], z[s, t]},
   {s, slow, shigh}, {t, tlow, thigh}, PlotPoints → {4, 8},
   PlotRange → {{-ranger, ranger}, {-ranger, ranger}, {-1, ranger}},
   BoxRatios → Automatic, Axes → True, AxesLabel → {"x", "y", "z"},
   Boxed → False, ViewPoint → CMView, DisplayFunction → Identity];

surfacebefore =
  Show[hitplotter[IdentityMatrix[3]], ThreeAxes[1], newline,
   PlotLabel → "Before", DisplayFunction → $DisplayFunction];
```

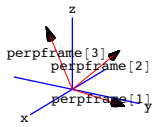


Use 3D rotator matrix hits to make a movie showing this bell rotating the plotted line.

□G.5.b.i) Make a positive definite matrix

Here's a 3D perpendicular frame:

```
{r, s, t} = N[{π/16, π/8, π/3}];

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

ranger = 1;
Show[
  Table[Arrow[perpframe[k], Tail → {0, 0, 0}, VectorColor → Red],
   {k, 1, 3}], Graphics3D[Text["perpframe[1]", 0.6 perpframe[1]]],
  Graphics3D[Text["perpframe[2]", 0.6 perpframe[2]]],
  Graphics3D[Text["perpframe[3]", 0.6 perpframe[3]]],
  ThreeAxes[1], ViewPoint → CMView, PlotRange → {{-ranger, ranger},
    {-ranger, ranger}, {-ranger, ranger}}, Boxed → False];
```

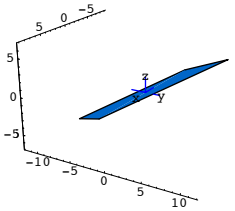


Make a 3D positive definite matrix (a frame stretcher) A whose hits stretch
vectors in the direction of perpframe[1] by a factor of 2.0,
vectors in the direction of perpframe[2] by a factor of 2.0, and
vectors in the direction of perpframe[3] by a factor of 0.8.
Illustrate with plots showing what a hit with this matrix does to points scattered on the surface of the unit sphere. Show the plots of the hit points from the viewpoints of perpframe[1], perpframe[2], and perpframe[3].
Say why you know in advance that the hit points shown from the viewpoint of perpframe[3] are guaranteed show up within a circle. What is the radius of this circle?

□G.5.b.ii) Make a projection

Here's a bit of the plane determined by the two vectors defined below:

```
Clear[planevector];
planevector[1] = {1.23, -2.07, -1.19};
planevector[2] = {1.75, 1.78, 1.03};

b = 3;
planeplot = Graphics3D[
   Polygon[{-b planevector[1] - b planevector[2], -b planevector[1] +
       b planevector[2], b planevector[1] + b planevector[2],
     b planevector[1] - b planevector[2]}]];

Show[planeplot, ThreeAxes[2], PlotRange → All,
  Axes → True, Boxed → False, ViewPoint → CMView];
```



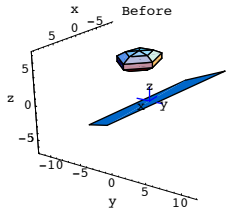
The question here is to come up with a matrix P so that when you hit a point {x,y,z} with your matrix P, you get the point on the plane that is closest to {x,y,z}.
Illustrate the action of your matrix with decisive plots.

□G.5.b.iii) Make a flipper

Here's the same plane as in part i) shown with an alien solid:

```
Clear[planevector];
planevector[1] = {1.23, -2.07, -1.19};
planevector[2] = {1.75, 1.78, 1.03};

b = 3;
planeplot = Graphics3D[
   Polygon[{-b planevector[1] - b planevector[2], -b planevector[1] +
       b planevector[2], b planevector[1] + b planevector[2],
     b planevector[1] - b planevector[2]}]];

Clear[x, y, z, s, t];
{x[s_, t_], y[s_, t_], z[s_, t_]} =
  {1, -1, 6} + {4.0 Sin[s] Cos[t], 3.0 Sin[s] Sin[t], 1.2 Cos[s]};

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

ranger = 8;
```

```
Clear[hitplotter, matrix];
hitplotter[matrix3D_] :=
  ParametricPlot3D[matrix3D.{x[s, t], y[s, t], z[s, t]},
   {s, slow, shigh}, {t, tlow, thigh}, PlotPoints → {6, 6}, PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   BoxRatios → Automatic, Axes → True, AxesLabel → {"x", "y", "z"},
   Boxed → False, ViewPoint → CMView, DisplayFunction → Identity];

before =
  Show[hitplotter[IdentityMatrix[3]], planeplot, ThreeAxes[2],
   PlotRange → All, Axes → True, Boxed → False, ViewPoint → CMView;
   PlotLabel → "Before", DisplayFunction → $DisplayFunction];
```



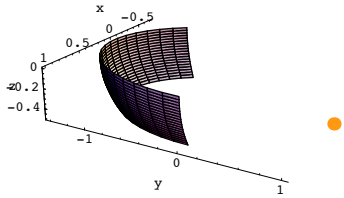
Use a 3D matrix hit to flip this surface underneath the plane.

□G.5.c) Bouncing light rays off surfaces in 3D

Here's a spherical surface and a point above the curve all plotted in true scale:

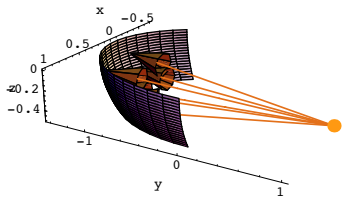
```
Clear[x, y, z, r, s, t];
{x[s_, t_], y[s_, t_], z[s_, t_]} =
  {Sin[s] Cos[t], 1.4 Sin[s] Sin[t], Cos[s]};


{slow, shigh} = {π/2, 2 π/3};

{tlow, thigh} = {π + π/4, 2 π};

point = {0, 1, -0.4};
Clear[surfaceplotter];
surfaceplotter[s_, t_] = {x[s, t], y[s, t], z[s, t]};

surfaceplot = ParametricPlot3D[Evaluate[surfaceplotter[s, t]],
   {s, slow, shigh}, {t, tlow, thigh}, Boxed → False,
   BoxRatios → Automatic, ViewPoint → CMView,
   AxesLabel → {"x", "y", "z"}, DisplayFunction → Identity];

pointplot =
  Graphics3D[{CadmiumYellow, PointSize[0.04], Point[point]}];
```

```
Show[surfaceplot, pointplot, Boxed → False,
  BoxRatios → Automatic, ViewPoint → CMView, PlotRange → All,
  AxesLabel → {"x", "y", "z"}, DisplayFunction → $DisplayFunction];
```



Light rays emanate from the point and hit the surface:

```
Clear[ray];
ray[s_, t_] = surfaceplotter[s, t] - point;

sjump = (shigh - slow)/3;
tjump = (thigh - tlow)/4;

rayplots =
  Table[Arrow[ray[s, t], Tail → point, VectorColor → MarsYellow],
   {s, slow + sjump, shigh - sjump, sjump},
   {t, tlow + tjump, thigh - tjump, tjump}];

setup = Show[surfaceplot, pointplot, rayplots, Boxed → False,
   BoxRatios → Automatic, ViewPoint → CMView, PlotRange → All,
   AxesLabel → {"x", "y", "z"}, DisplayFunction → $DisplayFunction];
```



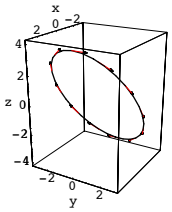
Your job is to plot the reflected light rays. Do it.

□G.5.d) Moving frames and wings

Here is a curve in three dimensions with some of its unit tangents plotted in true scale:

```
Clear[P, x, y, z, t];
x[t_] = 3 Cos[t];
```

```
y[t_] = 3 Sin[t];
z[t_] = 4 Cos[t];
P[t_] = {x[t], y[t], z[t]};

{tlow, thigh} = {0, 2 π};
tjump = π/3;

curveplot = ParametricPlot3D[Evaluate[P[t]],
  {t, tlow, thigh}, PlotPoints → 30, DisplayFunction → Identity];

Clear[unittan];
unittan[t_] = P'[t]/Sqrt[P'[t].P'[t]];
unittanvectors = Table[
  Arrow[unittan[t], Tail → P[t], VectorColor → Red], {t, 0, 2 π, π/6}];

curveandtans = Show[curveplot, unittanvectors,
  ViewPoint → CMView, PlotRange → All, BoxRatios → Automatic,
  AxesLabel → {"x", "y", "z"}, DisplayFunction → $DisplayFunction];
```

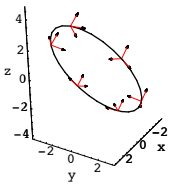


If you have had a good course in vector calculus, then you know that each point on the curve grows its own right hand perpendicular frame:

If you did not know this, that's not so bad because you saw it here first.
If you want more background, jump into C&M Lesson 3.02

```
Clear[perpframe, perpframeplotter, k];
perpframe[1, t_] := N[unittan[t]]
perpframe[2, t_] := N[unittan'[t]/Sqrt[unittan'[t].unittan'[t]]]
perpframe[3, t_] := perpframe[1, t] × perpframe[2, t]

perpframeplot[t_] := Table[
   Arrow[perpframe[k, t], Tail → P[t], VectorColor → Red], {k, 1, 3}];
perpframes = Table[perpframeplot[t], {t, tlow, thigh, tjump}];
```

```
setup = Show[curveplot, perpframes, ViewPoint → CMView,
   PlotRange → All, BoxRatios → Automatic, AxesLabel → {"x", "y", "z"},
   Boxed → False, DisplayFunction → $DisplayFunction];
```

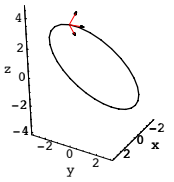


Most folks like to say that
perpframe[1,t] is the unit tangent at P[t].
perpframe[2,t] is the main unit normal at P[t]; it points into the curvature.
perpframe[3,t[is the binormal at P[t].

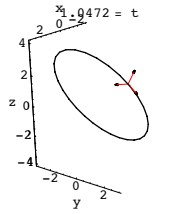
Watch these frames dance around the curve:

```
Table[Show[curveplot, perpframeplot[t],
  ViewPoint → CMView, PlotRange → All, BoxRatios → Automatic,
  PlotRange → {{-3, 3}, {-3, 3}, {-5, 5}}, Boxed → False,
  AxesLabel → {"x", "y", "z"}, PlotLabel → N[t] "= t",
  DisplayFunction → $DisplayFunction], {t, tlow, thigh, tjump}]
```

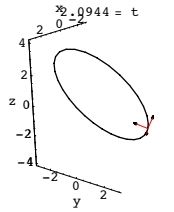
0. = t



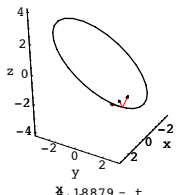
1.0472 = t



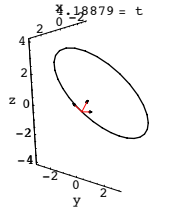
2.0944 = t



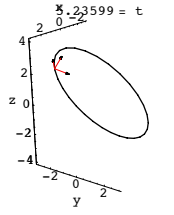
3.14159 = t



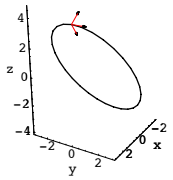
4.18879 = t



5.23599 = t



6.28319 = t

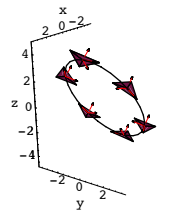


{- Graphics3D -, - Graphics3D -, - Graphics3D -,
 - Graphics3D -, - Graphics3D -, - Graphics3D -, - Graphics3D -}

Grab and animate.

Associate to each perpendicular frame a wing:

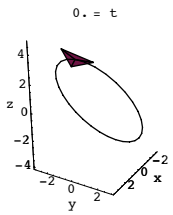
```
Clear[r, s, matrix3D, wingplotter, wingplots];
wingplotter[t_, matrix3D_] := ParametricPlot3D[P[t] + matrix3D.
     (r 1.3 Cos[s] perpframe[1, t] + r 1.0 Sin[s] perpframe[2, t]),
   {s, 0, 2 π}, {r, 0, 1}, PlotPoints → {4, 2},
   DisplayFunction → Identity];

Show[setup, Table[wingplotter[t, IdentityMatrix[3]],
   {t, tlow, thigh, tjump}], ViewPoint → CMView, PlotRange → All,
  BoxRatios → Automatic, AxesLabel → {"x", "y", "z"}];
```



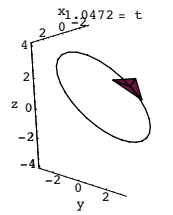
Note how each wing is hung on perpframe[1,t] and perpframe[2,t].

```
Table[Show[curveplot, wingplotter[t, IdentityMatrix[3]],
  ViewPoint → CMView, PlotRange → All, BoxRatios → Automatic,
  PlotRange → {{-3, 3}, {-3, 3}, {-5, 5}}, Boxed → False,
  AxesLabel → {"x", "y", "z"}, PlotLabel → N[t] "= t",
  DisplayFunction → $DisplayFunction], {t, tlow, thigh, tjump}]
```

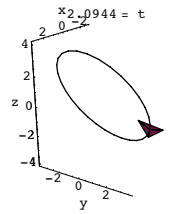
0. = t



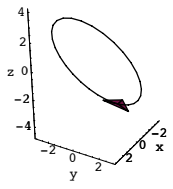
1.0472 = t

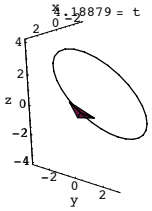


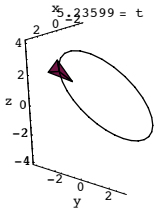
2.0944 = t

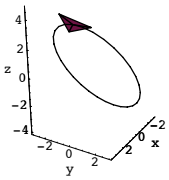


3.14159 = t









{- Graphics3D -, - Graphics3D -, - Graphics3D -,
 - Graphics3D -, - Graphics3D -, - Graphics3D -, - Graphics3D -}

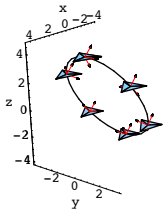
Your job is to define a matrix function A[t] so that when you execute:

```
Clear[A];
A[t_] = ?? ?? ??????????????

Show[setup, Table[wingplotter[t, A[t]], {t, tlow, thigh, tjump}],
 ViewPoint -> CMView, PlotRange -> All,
  BoxRatios -> Automatic,
  AxesLabel -> {"x", "y", "z"}];
```

You get the same output as you get from:

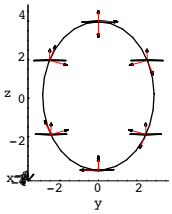
```
Clear[r, s, matrix3D, alignedwingplotter, alignedwingplots];
alignedwingplotter[t_, matrix3D_] := ParametricPlot3D[
   P[t] + matrix3D.(r 1.3 Cos[s] {1, 0, 0} + r 1.0 Sin[s] {0, 1, 0}),
   {s, 0, 2 π}, {r, 0, 1}, PlotPoints → {4, 2},
   DisplayFunction → Identity];
```

```
alignedwings =
  Show[setup, Table[alignedwingplotter[t, IdentityMatrix[3]],
    {t, tlow, thigh, tjump}], ViewPoint → CMView, PlotRange → All,
   BoxRatios → Automatic, AxesLabel → {"x", "y", "z"}];
```



In this picture the aligned wings are parallel to the xy-plane:

```
Show[alignedwings, ViewPoint → 12 {1, 0, 0}];
```

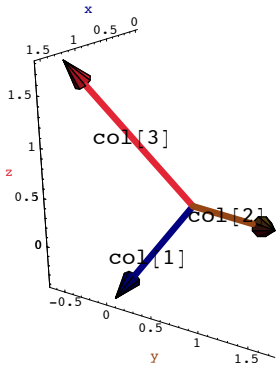


Make a movie showing your aligned wings dance around the curve.

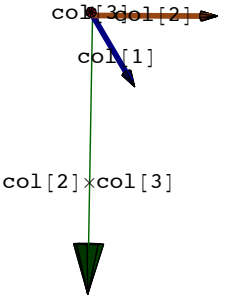
G.6) 3D determinant fundamentals

□G.6.a.i) Columns and the sign of the 3D determinant

Here is a plot the columns of a certain 3D matrix A:



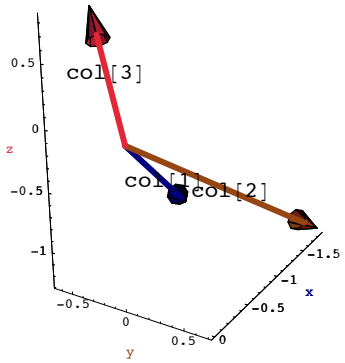
Here are the same columns of A shown with col[2] × col[3] shown a viewpoint along col[3]:



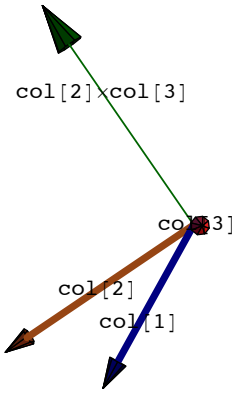
You make the call:
Det[A] is
Positive. . . . .Zero. . . . Negative. . . .

□G.6.a.ii) Columns and the sign of the 3D determinant

Here is a plot the columns of a certain 3D matrix A:



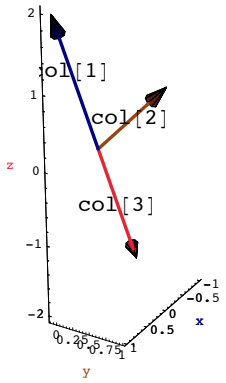
Here are the same columns of A shown with col[2] × col[3] shown a viewpoint along col[3]:



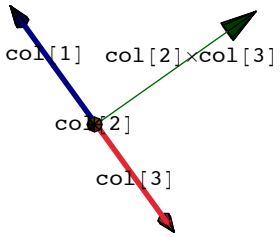
You make the call:
Det[A] is
Positive. . . . .Zero. . . . Negative. . . .

□G.6.a.iii) Columns and the sign of the 3D determinant

Here is a plot the columns of a certain 3D matrix A:



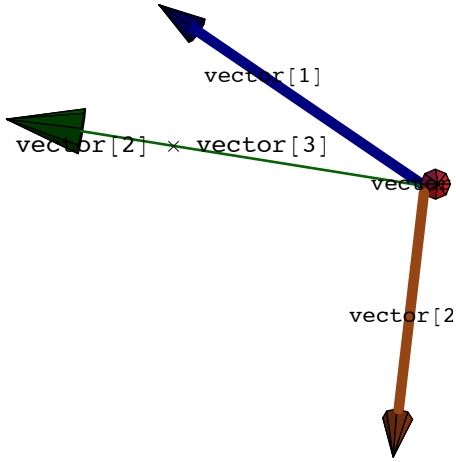
Here are the same columns of A shown with col[2] × col[3] shown a viewpoint along col[2]:



You make the call:
Det[A] is
Positive. . . . .Zero. . . . Negative. . . .

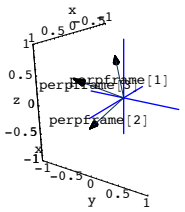
□G.6.a.iv) Sign of the 3D determinant revealed by a matrix hit

Here is a plot of three vectors {vector[1],vector[2],vector[3]} along with vector[2]×vector[3] shown from a viewpoint along vector[3]:



Here is a plot {A.vector[1],A.vector[2],A.vector[3]} along with (A.vector[2])×(A.vector[3]) shown from a viewpoint along A.vector[3] for a certain matrix A

```
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   Boxed → False, Axes → True, ViewPoint → CMView,
   AxesLabel → {"x", "y", "z"}];

hanger = Transpose[{perpframe[1], perpframe[2], perpframe[3]}];
MatrixForm[hanger]
```



$$\begin{pmatrix} -0.64678 & 0.0442213 & 0.761393 \\ -0.570618 & -0.690439 & -0.444623 \\ 0.506033 & -0.722038 & 0.471796 \end{pmatrix}$$

The determinant of this matrix is:

```
Det[hanger]
```
```
1.
```

Rerun until you get a left hand perpendicular frame.

□G.6.a.vi) The aligner and hanger frames set the sign of the determinant

From the Tutorials:

- If A is a 3D hanger or aligner based on a right hand frame, then Det[A] = 1.
- If A is a 3D hanger or aligner based on a left hand frame, then Det[A] = —1.
- If A is a 3D stretcher, then Det[A] = product of stretch factors.

So if A = hanger.stretcher.aligner, then
Det[A] = Det[hanger] xstretch ystretch Det[aligner].

Use this good information to help to answer these questions:

- How do you know that saying Det[A] < 0 is the same as saying that either
  the aligner frame is a right hand frame and the hangerframe is a left hand frame
  or
  the aligner frame is a left hand frame and the hangerframe is a right hand frame.

Put answer here.

- How do you know that if Det[A] < 0, then a hit with A does not preserve orientation?

Put answer here.

- How do you know that saying Det[A] > 0 is the same as saying that either
  the aligner frame is a right hand frame and the hangerframe is a right hand frame
  or
  the aligner frame is a left hand frame and the hangerframe is a left hand frame.

Put answer here.

- How do you know that if Det[A] > 0, then a hit with A incorporates no flip?

Put answer here.

- How do you know that if Det[A] > 0, then a hit with A preserves orientation?

Put answer here.

□G.6.a.vii) What sets the absolute value of the determinant?

Here's a random 3D matrix A:

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

$$\begin{pmatrix} -1.14522 & -0.233979 & 1.18528 \\ 1.54174 & -1.17537 & -0.597489 \\ -0.0975219 & 1.33591 & -1.02088 \end{pmatrix}$$

Here are its SVD stretch factors:

```
{xstretch, ystretch, zstretch} = SingularValues[A][[2]]
```
```
{2.39663, 1.99331, 0.0763632}
```

How do you use these three numbers to calculate |Det[A]|?

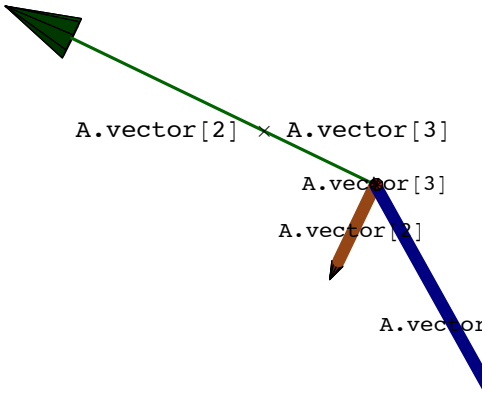
□G.6.b.i) Products

Here's a plot of the vertical columns of

$$A = \begin{pmatrix} 0.40 & 0.89 & 0.80 \\ -1.31 & -0.56 & 1.11 \\ 1.73 & -0.07 & -0.78 \end{pmatrix};$$

shown with the plot of col[2] × col[3] all shown from a viewpoint along col[3].

```
A = ( 0.40   0.89   0.80
     -1.31  -0.56   1.11
      1.73  -0.07  -0.78 );
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},
```



You make the call:
Is Det[A] > 0 or is Det[A] < 0?
Give your reasons.

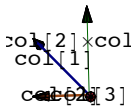
□G.6.a.v) Left or right hand perpendicular frame?

Here's a random perpendicular frame and corresponding hanger matrix:

```
Clear[perpframe];
{r, s, t} =
  {Random[Real, {0, π}], Random[Real, {0, π}], Random[Real, {0, π}]};
{perpframe[1], perpframe[2], perpframe[3]} =
  {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
    Cos[s] Cos[t] Sin[r] + Cos[r] Sin[t], Sin[r] Sin[s]},
   ((-1)^Random[Integer,{0,1}]) {-Cos[t] Sin[r] - Cos[r] Cos[s] Sin[t],
     Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[r] Sin[s]},
   {Sin[s] Sin[t], -Cos[t] Sin[s], Cos[s]}};
ranger = 1.0;
frameplot = Show[
   Table[Arrow[perpframe[k], Tail → {0, 0, 0}, VectorColor → Indigo],
    {k, 1, 3}], Graphics3D[Text["perpframe[1]", 0.4 perpframe[1]]],
   Graphics3D[Text["perpframe[2]", 0.7 perpframe[2]]],
   Graphics3D[Text["perpframe[3]", 0.7 perpframe[3]]],
   Axes3D[2, 0.1], PlotRange →
```

```
    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[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 columns",
 DisplayFunction -> $DisplayFunction];
```

A columns



The columns of A are positively oriented.

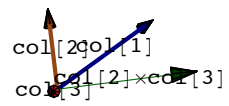
Here's a plot of the vertical columns of

$$B = \begin{pmatrix} 1.87 & 0.07 & -0.32 \\ 1.33 & 0.97 & -1.33 \\ 0.06 & 0.84 & 1.04 \end{pmatrix};$$

shown with the plot of col[2]×col[3] all shown from a viewpoint along col[3].

```
B = ( 1.87  0.07  -0.32
      1.33  0.97  -1.33
      0.06  0.84   1.04 );

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

B columns



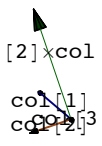
The columns of B are positively oriented.
Here's a plot of the vertical columns of A.B shown with the plot of col[2]×col[3] all shown from a viewpoint along col[3].

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

$$\begin{pmatrix} 1.9797 & 1.5633 & -0.4797 \\ -3.1279 & 0.2975 & 2.3184 \\ 3.0952 & -0.602 & -1.2717 \end{pmatrix}$$

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

A.B columns



The columns of A.B are positively oriented.
Here you took two matrices A and B each with positively oriented columns and found that the columns of the product A.B are also positively oriented.

Was this just a fluke?
Or is it true that when you go with any two 3D matrices A and B each with positively oriented columns, then the columns of the product A.B are guaranteed to be positively oriented?

On what facts do you base your answer?

□G.6.b.ii) When does A.B preserve orientation?

Explain these factoids for given 3D matrices A and B:.

- If hits with A **do** preserve orientation and hits with B **do** preserve orientation, then hits with A.B **do** preserve orientation
  Put answer here.
- If hits with A **do** preserve orientation but hits with B **do not** preserve orientation, then hits with A.B **do not** preserve orientation.
  Put answer here.
- If hits with A **do not** preserve orientation but hits with B **do** preserve orientation, then hits with A.B **do not** preserve orientation.
  Put answer here.
- If hits with A **do not** preserve orientation and hits with B **do not** preserve orientation, then hits with A.B **do** preserve orientation.
  Put answer here.

□G.6.c.i) Rows and columns

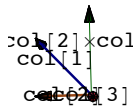
Here's a plot of the vertical columns of

$$A = \begin{pmatrix} 0.40 & 0.89 & 0.80 \\ -1.31 & -0.56 & 1.11 \\ 1.73 & -0.07 & -0.78 \end{pmatrix};$$

shown with the plot of col[2] × col[3] all shown from a viewpoint along col[3].

```
A = ( 0.40   0.89   0.80
     -1.31  -0.56   1.11
      1.73  -0.07  -0.78 );
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[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 columns",
 DisplayFunction -> $DisplayFunction];
```

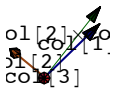
A columns



The columns of A are positively oriented.
Now look at a plot of the rows of A which are the columns of A^t

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

A rows



In this plot , col[j] is the jth column of A^t; so col[j] is row[j] of A.
The orientation of the rows of A is positive.

Here you took a matrix 3D A with positively oriented columns and found that the rows of A are also positively oriented.

Was this just a fluke?
Or is it true that when you go with a matrix 3D A with positively oriented columns, then the rows of A are guaranteed to be positively oriented?

On what facts do you base your answer?

G.6.c.ii) Interchanging two columns; interchanging two rows

One day, one of your friends comes by, holds out a dusty book and says," This book I checked out of the library says that:
When you interchange two columns of a 3D matrix, you get the determinant of the resulting matrix by multiplying the old determinant by -1."
You say that, for 3D matrices, that's easy to explain. First look at this cleared 3D matrix.

```
Clear[a, b, c, r, s, t, x, y , z];
A = {{a, r, x}, {b, s, y}, {c, t, z}};
```

To interchange the first and second columns of A, you do this:

```
interchange12A = A. {{0, 1, 0}, {1, 0, 0}, {0, 0, 1}};
MatrixForm[interchange12A]
```

$$\begin{pmatrix} r & a & x \\ s & b & y \\ t & c & z \end{pmatrix}$$

Use the fact that

$$\text{Det[interhange 12A]} = \text{Det[A.} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}\text{]} = \text{Det[A] Det[} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}\text{]}$$

to explain why

Det[interchange12A] = - Det[A].

Put answer here.

Now go back to

```
Clear[a, b, c, r, s, t, x, y , z];
A = {{a, r, x}, {b, s, y}, {c, t, z}};
```

To interchange the first and third columns of A, you do this:

```
interchange13A = A. {{0, 0, 1}, {0, 1, 0}, {1, 0, 0}};
MatrixForm[interchange13A]
```

$$\begin{pmatrix} x & r & a \\ y & s & b \\ z & t & c \end{pmatrix}$$

Use the fact that

$$\text{Det[interhange 13A]} = \text{Det[A.} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}\text{]} = \text{Det[A] Det[} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}\text{]}$$

to explain why

Det[interchange13A] = - Det[A].

Put answer here.

Now you take over and build an explanation of why interchanging the second and third column of A reverses the sign of the determinant.
Put answer here.

What happens to the determinant when you interchange two rows of a 3D matrix?
Put answer here.

G.6.d.i) Using the determinant formula Det[A] = col[1].(col[2] × col[3])

Go with $A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 2 & -2 & -1 \end{pmatrix}$

The determinant of A is:

```
A = {{1, 2, 3}, {0, 1, 2}, {2, -2, -1}};
Det[A]
```

5

The formula for the determinant is:

Det[A] = col[1] . (col[2] × col[3])

Use the formula to duplicate this calculation of Det[A].

G.7) A 3D vector X ≠ {0,0,0} is an eigenvector of a 3D matrix A if A.X points in the same or opposite direction as X.

In other words A.X = s X for a number s.

In this case, s is called the eigenvalue corresponding to X. Eigenvalues as roots of the characteristic polynomial Det[A - s Identity]

Using eigenvectors to learn what a 3D reflection matrix does and to learn that the product of two 3D rotations is one 3D rotation*

G.7.a.i) Eigenvectors and eigenvalues

All clued in folks say that a 3D vector

X ≠ {0,0,0}

is an eigenvector of a 3D matrix A if **A.X points in the same or opposite direction as X**.
In other words

A.X = s X for a number s.

In this case, s is called the eigenvalue corresponding to the eigenvector X.
Here's a 3D matrix A:

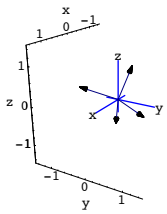
```
A = {{1.35, 0.08, -0.033}, {0.08, 1.286, 0.088}, {-0.033, 0.089, 1.46}};

MatrixForm[A]
```

$$\begin{pmatrix} 1.35 & 0.08 & -0.033 \\ 0.08 & 1.286 & 0.088 \\ -0.033 & 0.089 & 1.46 \end{pmatrix}$$

Here's a look at its unit eigenvectors shown with another unit vector:

Just as you don't have to know how a fine meal was cooked in order to enjoy it,you don't have to know how these special vectors were calculated in order to enjoy them.Later on in the course,you will learn all about them.

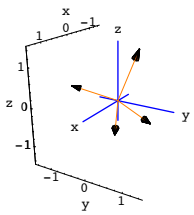
```
Clear[eigenvector];
{eigenvector[1], eigenvector[2], eigenvector[3]} =
  Eigenvectors[A];
othervector = Normalize[
   {Random[Real, {-1, 1}], 2, Random[Real, {-1, 1}]}];
ranger = 1.5;

before = Show[
   Arrow[eigenvector[1], Tail → {0, 0, 0}, VectorColor → NavyBlue],
   Arrow[eigenvector[2], Tail → {0, 0, 0}, VectorColor → NavyBlue],
   Arrow[eigenvector[3], Tail → {0, 0, 0}, VectorColor → NavyBlue],
   Arrow[othervector, Tail → {0, 0, 0}, VectorColor → NavyBlue],
   Axes3D[1.0], PlotRange → {{-ranger, ranger},
     {-ranger, ranger}, {-ranger, ranger}}, Axes → True,
   AxesLabel → {"x", "y", "z"}, ViewPoint → CMView, Boxed → False];
```



If you don't see four distinct clear vectors, then rerun.

Here's what you get when you hit all four plotted vectors with the matrix A:

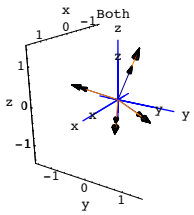
```
after = Show[
  Arrow[A.eigenvector[1], Tail -> {0, 0, 0}, VectorColor -> Orange],
      Arrow[A.eigenvector[2], Tail -> {0, 0, 0},
   VectorColor -> Orange],
      Arrow[A.eigenvector[3], Tail -> {0, 0, 0},
   VectorColor -> Orange],
       Arrow[A.othervector, Tail -> {0, 0, 0},
   VectorColor -> Orange],
      Axes3D[1.6],
       PlotRange ->
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
      Axes -> True, AxesLabel -> {"x", "y", "z"},
      ViewPoint -> CMView, Boxed -> False];
```



Grab both plots, align and animate.

Here are both plots:

```
Show[before, after, PlotLabel → "Both", PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  Axes → True, AxesLabel → {"x", "y", "z"},
  ViewPoint → CMView, Boxed → False];
```



If you don't like what you see, go back and rerun.

Describe how this plot reveals which three of the original vectors are the eigenvectors of A.
Describe how this plot reveals that all the eigenvalues of A are positive.

□G.7.a.ii) Positive definite matrices (frame stretchers)

Here's a 3D perpendicular frame and the frame stretcher matrix A based on this perpendicular frame:

```
{r, s, t} = N[{0.1 Pi, Pi / 4, 0.3 Pi}];
Clear[perpframe];
{perpframe[1], perpframe[2], perpframe[3]} =
{{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
   Cos[s] Cos[t] Sin[r] + Cos[r] Sin[t],
   Sin[r] Sin[s]}, {-Cos[t] Sin[r] - Cos[r] Cos[s] Sin[t],
   Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[r] Sin[s]}, {
   Sin[s] Sin[t], -Cos[t] Sin[s], Cos[s]}};
aligner = {perpframe[1], perpframe[2], perpframe[3]};

{xstretch, ystretch, zstretch} = {1.5, 2.1, 0.4};
```

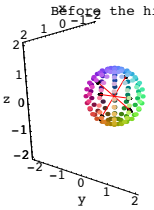
```
stretcher = DiagonalMatrix[{xstretch, ystretch, zstretch}];

hanger = Transpose[{perpframe[1], perpframe[2], perpframe[3]}];

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

$$\begin{pmatrix} 1.456 & 0.198281 & -0.737778 \\ 0.198281 & 1.32264 & 0.381904 \\ -0.737778 & 0.381904 & 1.22135 \end{pmatrix}$$

Here are points on the 3D unit sphere shown with the given perpendicular frame:

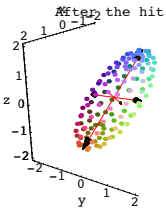
```
Clear[x, y, s, t, pointcolor];
{x[s_, t_], y[s_, t_], z[s_, t_]} =
  {Sin[s] Cos[t], Sin[s] Sin[t], Cos[s]};

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

ranger = Max[{xstretch, ystretch, zstretch, 1.2}];
pointcolor[s_, t_] =
  RGBColor[0.5 (x[s, t] + 1), 0.5 (y[s, t] + 1), 0.5 (z[s, t] + 1)];
sjump = (shigh - slow) / 12;
tjump = (thigh - tlow) / 12;

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}],
    Table[Arrow[matrix3D.perpframe[k], Tail → {0, 0, 0},
      VectorColor → Red], {k, 1, 3}],
    Table[Arrow[-matrix3D.perpframe[k], Tail → {0, 0, 0},
      VectorColor → Red], {k, 1, 3}]}, PlotRange →
    {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
   Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False,
   ViewPoint → CMView, DisplayFunction → Identity];
```

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

pointsbefore = Show[hitpointplotter[IdentityMatrix[3]],
   hitframeplotter[IdentityMatrix[3]], PlotLabel → "Before the hit",
   DisplayFunction → $DisplayFunction];
```



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

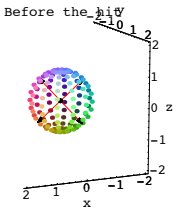
```
pointsafter = Show[hitpointplotter[A], hitframeplotter[A],
   PlotLabel → "After the hit", DisplayFunction → $DisplayFunction];
```

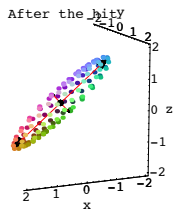


Grab both plots and animate at various speeds.

See the same thing from the view points of each of the given perpendicular frame vectors:

```
Show[pointsbefore, ViewPoint → 10 perpframe[1]];
Show[pointsafter, ViewPoint → 10 perpframe[1]];
```

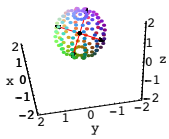




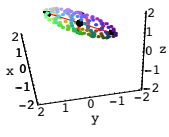
Grab both plots and animate at various speeds.

```
Show[pointsbefore, ViewPoint → 10 perpframe[2]];
Show[pointsafter, ViewPoint → 10 perpframe[2]];
```

Before the hit

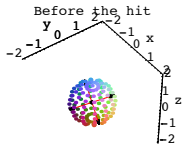


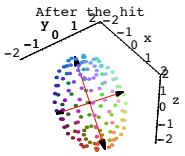
After the hit



Grab both plots and animate at various speeds.

```
Show[pointsbefore, ViewPoint → 10 perpframe[3]];
Show[pointsafter, ViewPoint → 10 perpframe[3]];
```





Grab both plots and animate at various speeds.

Grab, align and animate..

This frame stretcher A matrix was made so that

A.perpframe[1] = xstretch perpframe[1]

and

A.perpframe[2] = ystretch perpframe[2]:

and

A.perpframe[3] = zstretch perpframe[3]

and

```
A.perpframe[1] == xstretch perpframe[1]
A.perpframe[2] == ystretch perpframe[2]
A.perpframe[3] == zstretch perpframe[3]
```

```
True
True
True
```

Remembering that a 3D vector

X ≠ {0,0,0}

is an eigenvector A if **A.X points in the same or opposite direction as X**.
In other words

A.X = s X for a number s.

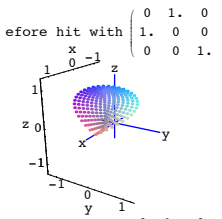
In this case, s is called the eigenvalue corresponding to the eigenvector X.

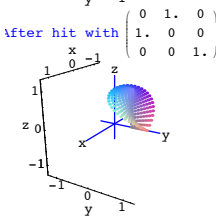
Use the output above to read off three guaranteed eigenvectors of A and give the associated eigenvalues.

□**G.7.a.iii)** $\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{1.0} & \mathbf{0} \\ \mathbf{1.0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1.0} \end{pmatrix}$

The matrix A = $\begin{pmatrix} 0 & 1.0 & 0 \\ 1.0 & 0 & 0 \\ 0 & 0 & 1.0 \end{pmatrix}$ is a hanger corresponding to the perpendicular frame {{0,1,0},{1,0,0},{0,0,1}}.
See it do its work:

```
A = {{0, 1.0, 0}, {1.0, 0, 0}, {0, 0, 1.0}};
Clear[x, y, z, r, s, t, pointcolor];
a = 0.5;
{x[r_, t_], y[r_, t_], z[r_, t_]} =
  {a r Cos[t] + a, r Sin[t], Sin[0.5 t]};

{rlow, rhigh} = {0, 1};
{tlow, thigh} = {0, 2 π};

ranger = 1.3;
pointcolor[s_, t_] :=
  RGBColor[x[r, t] / (2 a) , (y[r, t] + 1) / 2, (z[r, t] + 1) / 2 ];

rjump = (rhigh - rlow) / 8;
tjump = (thigh - tlow) / 36;


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

before = Show[hitpointplotter[IdentityMatrix[3]],
   Axes3D[ranger], PlotLabel → "Before hit with" MatrixForm[A],
   DisplayFunction → $DisplayFunction];

after = Show[hitpointplotter[A], Axes3D[ranger],
  PlotLabel → "After hit with" MatrixForm[A],
  DisplayFunction → $DisplayFunction];
```

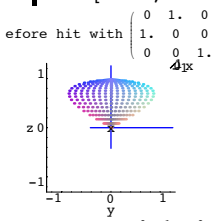


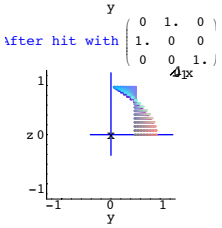


Grab and animate slowly.

Here's the same thing from a viewpoint out on the positive x axis:

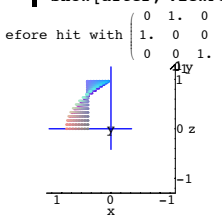
```
Show[before, ViewPoint -> 13 {1, 0, 0}];
Show[after, ViewPoint -> 13 {1, 0, 0}];
```

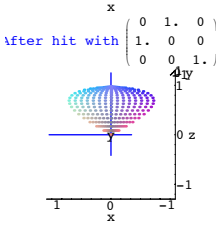




Here's the same thing from a viewpoint out on the positive y axis:

```
Show[before, ViewPoint -> 13 {0, 1, 0}];
Show[after, ViewPoint -> 13 {0, 1, 0}];
```

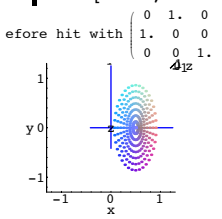


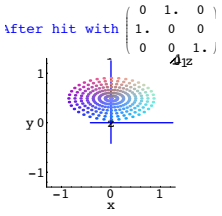


Grab and animate slowly.

Here's the same thing from a viewpoint out on the positive z axis:

```
Show[before, ViewPoint -> 13 {0, 0, 1}];
Show[after, ViewPoint -> 13 {0, 0, 1}];
```





Grab and animate slowly.

Now look at this:

```
A = {{0, 1.0, 0}, {1.0, 0, 0}, {0, 0, 1.0}};
Clear[eigenvalue, eigenvector];
{eigenvalue[1], eigenvalue[2], eigenvalue[3]} = Eigenvalues[A]
```

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

```
{eigenvector[1], eigenvector[2], eigenvector[3]} = Eigenvectors[A]
```

```
{{0.707107, 0.707107, 0}, {-0.707107, 0.707107, 0}, {0, 0, 1.}}
```

```
A.eigenvector[1] == eigenvalue[1] eigenvector[1]
```

```
True
```

```
A.eigenvector[2] == eigenvalue[2] eigenvector[2]
```

```
True
```

```
A.eigenvector[3] == eigenvalue[3] eigenvector[3]
```

```
True
```

Stare at the output above and come up with two vectors that define the plane about which hits with A flip.

□G.7.b.i) Eigenvalues as roots of the characteristic polynomial

Saying that a 3D vector X ≠ {0,0,0} is an eigenvector of a 3D matrix A is the same as saying

A.X = s X for a number s.

In this case, s is called the eigenvalue corresponding to the eigenvector X.

Saying A.X = s X for a number s is the same as saying

A.X - s X = {0,0,0}

And this is the same as saying

(A - s Identity).X = {0,0,0}.

This is the same as saying that the matrix (A - s Identity) squashes a non-zero vector onto {0,0,0}.

This is the same as saying that the matrix (A - s Identity) has at least one zero SVD stretch factor.

And because the absolute value of the determinant is the product of the SVD stretch factors,

saying that a number s is an eigenvalue for A (i.e. A.X = s X for some X ≠ {0,0,0}) is the same as saying that

Det[A - s Identity] = 0.

Try this out on this 3D matrix:

```
A = {{2.9, 0.4, 0.2}, {-0.4, 5.8, -0.1}, {0.2, 0.4, -3.7}};

MatrixForm[A]
```

$$\begin{pmatrix} 2.9 & 0.4 & 0.2 \\ -0.4 & 5.8 & -0.1 \\ 0.2 & 0.4 & -3.7 \end{pmatrix}$$

The characteristic polynomial of A is:

```
Clear[charpolynomial, s];
charpolynomial[s_] = Det[A - s IdentityMatrix[3]]
```

```
-62.982 + 15.21 s + 5. s^2 - s^3
```

The roots of the characteristic polynomial are:

```
Solve[charpolynomial[s] == 0, s]
```

```
{{s → -3.70248}, {s → 2.96469}, {s → 5.73779}}
```

Eigenvalues of A are:

```
Eigenvalues[A]
```

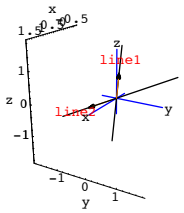
```
{5.73779, -3.70248, 2.96469}
```

Any surprises here?

□G.7.c) Two 3D rotations result in one 3D rotation

Here are two lines in 3D:

```
{s1, t1} = {Random[Real, {0, π/2}], Random[Real, {0, π/2}]};

unitvector1 = {Sin[s1] Cos[t1], Sin[s1] Sin[t1], Cos[s1]};
line1 = Graphics3D[Line[{-2 unitvector1, 2 unitvector1}]];
label1 = Graphics3D[{Red, Text["line1", 1.5 unitvector1]}];
{s2, t2} = {Random[Real, {π/2, π}], Random[Real, {0, -π/2}]};

unitvector2 = {Sin[s2] Cos[t2], Sin[s2] Sin[t2], Cos[s2]};
line2 = Graphics3D[Line[{-2 unitvector2, 2 unitvector2}]];
label2 = Graphics3D[{Red, Text["line2", 1.5 unitvector2]}];

setup = Show[line1, line2, label1, label2,
   Arrow[unitvector1, Tail → {0, 0, 0}, VectorColor → Orange],
   Arrow[unitvector2, Tail → {0, 0, 0}, VectorColor → Orange],
   Axes3D[1.5], Axes → True, AxesLabel → {"x", "y", "z"},
   ViewPoint → CMView, Boxed → False, PlotRange → All,
   DisplayFunction → $DisplayFunction];
```



Make a matrix rotate1 whose hits rotate everything by $\frac{\pi}{4}$ radians about line1:

```
perpframe[3] = unitvector1;
throwawayvector = {Random[Real, {-1, 1}],
   Random[Real, {-1, 1}], Random[Real, {-1, 1}]};
planevector = throwawayvector × perpframe[3];
perpframe[1] = planevector / Norm[planevector];
perpframe[2] = perpframe[3] × perpframe[1];

Clear[zrotater3D, s];
zrotater3D[s_] = Transpose[
   {{Cos[s], Sin[s], 0}, {Cos[s + π/2], Sin[s + π/2], 0}, {0, 0, 1}}];
aligner = {perpframe[1], perpframe[2], perpframe[3]};
hanger = Transpose[aligner];
rotate1 = Expand[hanger.zrotater3D[π/4].aligner];
MatrixForm[rotate1]
```

$$\begin{pmatrix} 0.738734 & -0.615822 & 0.273925 \\ 0.666632 & 0.727513 & -0.162252 \\ -0.0993655 & 0.302468 & 0.947966 \end{pmatrix}$$

Make a matrix rotate2 whose hits rotate everything by 0.4 π radians about line2:

```
perpframe[3] = unitvector2;
throwawayvector = {Random[Real, {-1, 1}],
   Random[Real, {-1, 1}], Random[Real, {-1, 1}]};
planevector = throwawayvector × perpframe[3];
perpframe[1] = planevector / Norm[planevector];
perpframe[2] = perpframe[3] × perpframe[1];
aligner = {perpframe[1], perpframe[2], perpframe[3]};
hanger = Transpose[aligner];
rotate2 = Expand[hanger.zrotater3D[0.4 π].aligner];
MatrixForm[rotate2]
```

$$\begin{pmatrix} 0.327834 & 0.302509 & -0.894993 \\ -0.505504 & 0.856493 & 0.104331 \\ 0.798117 & 0.418219 & 0.433707 \end{pmatrix}$$

Now put

A = rotate2.rotate1:

```
A = rotate2.rotate1;
MatrixForm[A]
```

$$\begin{pmatrix} 0.532776 & -0.252515 & -0.807704 \\ 0.187166 & 0.965968 & -0.178535 \\ 0.825299 & -0.0560556 & 0.561907 \end{pmatrix}$$

Look at this:

```
Clear[eigenvalue, eigenvector];
{eigenvalue[1], eigenvalue[2], eigenvalue[3]} = Eigenvalues[A]
```

```
{0.530325 + 0.847794 i, 0.530325 - 0.847794 i, 1.}
```

```
{eigenvector[1], eigenvector[2], eigenvector[3]} = Eigenvectors[A]
```

```
{{0.70526, 0.0493209 - 0.183839 i, -0.0132795 - 0.682791 i},
 {0.70526, 0.0493209 + 0.183839 i, -0.0132795 + 0.682791 i},
 {0.0722344, -0.963089, 0.259309}}
```

This gives you the extra information that:

```
A.eigenvector[1] == eigenvalue[1] eigenvector[1]
A.eigenvector[2] == eigenvalue[2] eigenvector[2]
A.eigenvector[3] == eigenvalue[3] eigenvector[3]
```

```
True
True
True
```

Old matrix hands know that because A is one rotation followed by another,the matrix A is itself a rotation matrix. Use the eigenvalue-eigenvector information above to come up with the 3D line hits with A rotate about.

G.8) Plotting

□G.8.a) Hitting the sphere with A^{-1} and with A^t

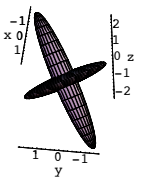
Here is a 3D matrix A shown together what you get when you hit the 3D unit sphere with A^t and with A^{-1}:

```
This matrix uses a random aligner frame and a random hanger frame.
The stretch factors are {2,1.4,0.3}.
```

```
Clear[alignerframe, hangerframe];
{r, s, t} = {Random[Real, {0, π}],
   Random[Real, {0, 2 π}], Random[Real, {0, 2 π}]};
```

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

        {xstretch, ystretch, zstretch} = {2, 1.4, 0.3};
stretcher = DiagonalMatrix[{xstretch, ystretch, zstretch}];

            {r, s, t} =
  {Random[Real, {0, Pi}], Random[Real, {0, 2 Pi}],
  Random[Real, {0, 2 Pi}]};
        {hangerframe[1], hangerframe[2],
  hangerframe[3]} = {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
     Cos[s] Cos[t] Sin[r] + Cos[r] Sin[t], Sin[r] Sin[s]},
    {-Cos[t] Sin[r] - Cos[r] Cos[s] Sin[t],
     Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[r] Sin[s]},
    {Sin[s] Sin[t], -Cos[t] Sin[s], Cos[s]}};
        hanger = Transpose[
   {hangerframe[1], hangerframe[2], hangerframe[3]}];

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

Clear[x, y, z, s, t];
{x[s_, t_], y[s_, t_], z[s_, t_]} =
 Sin[s] Cos[t] hangerframe[1] + Sin[s] Sin[t] hangerframe[2] +
  Cos[s] hangerframe[3];
{slow, shigh} = {0, Pi};
{tlow, thigh} = {0, 2 Pi};
B = Transpose[A];
tranposeplot =
 ParametricPlot3D[B.{x[s, t], y[s, t], z[s, t]},
   {s, slow, shigh}, {t, tlow, thigh},
  DisplayFunction -> Identity];
BB = Inverse[A];
inverseplot =
 ParametricPlot3D[BB.{x[s, t], y[s, t], z[s, t]},
   {s, slow, shigh}, {t, tlow, thigh},
  DisplayFunction -> Identity];


Show[tranposeplot, inverseplot, Axes -> True, Boxed -> False,
 AxesLabel -> {"x", "y", "z"}, ViewPoint -> 10 alignerframe[2],
 DisplayFunction -> $DisplayFunction];
```

$$\begin{pmatrix} -0.873033 & 0.546086 & -0.951935 \\ -0.893488 & -1.53062 & 0.163118 \\ -0.386913 & -0.685877 & 0.543664 \end{pmatrix}$$



Rerun a couple of times.

One of those ellipsoids is what you get by hitting the 3D unit sphere with A^{-1} .
The other ellipsoid is what you get by hitting the 3D unit sphere with A^t.
The stretch factors for A have been set to make the graphics revealing. Everything else about A is random.
The puzzle here is:
Why do they turn out the way they do?

Click for a tip.

Ask yourself :

What perpendicular frame does A^t use to hang its hits?

What perpendicular frame does A^{-1} use to hang its hits?

□G.8.b) Min[xstretch,ystretch,zstretch] ||{x,y,z}|| ≤ ||A.{x,y,z}|| .

If xstretch,ystretch and zstretch are all positive, is it possible for A.{x,y,z} = {0,0,0} when {x,y,z}≠ {0,0,0}?

Here's a random matrix 3D A:

```
Clear[a, i, j];
a[i_, j_] := Random[Real, {-2, 2}]
A = Table[a[i, j], {i, 1, 3}, {j, 1, 3}];
MatrixForm[A]
```

$$\begin{pmatrix} -0.997866 & -0.554323 & -1.71037 \\ -1.59623 & -0.333778 & -1.53344 \\ -1.40004 & 0.977112 & -1.44173 \end{pmatrix}$$

Here are the stretch factors for A:

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

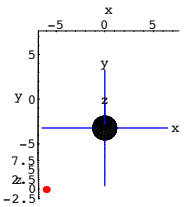
{3.55997, 1.19209, 0.371647}

Now look at this plot showing:
- A.{x,y,z} for a random point {x,y,z}.
- The sphere of
  radius = Min[{xstretch,ystretch,zstretch}] ||{x,y,z}||

centered at {0,0,0}.

Here ||{x,y,z}|| = Sqrt[{x,y,z}.{x,y,z}]

```
{x, y, z} = {Random[Real, {-3, 3}],
   Random[Real, {-3, 3}], Random[Real, {-3, 3}]};
xyznorm = Sqrt[{x, y, z}.{x, y, z}];

littleradius = xyznorm Min[{xstretch, ystretch, zstretch}];
bigradius = xyznorm Max[{xstretch, ystretch, zstretch}];
ranger = bigradius;
hitxyzplot = Graphics3D[{Red, PointSize[0.05], Point[A.{x, y, z}]}];
hitxyzlabel =
  Graphics3D[{Red, Text["A.{x,y,z}", A.{x, y, z}, {0, 2}]}];
littlesphereplot = Graphics3D[Sphere[littleradius, 20, 20]];
bigsphereplot = Graphics3D[Sphere[bigradius, 20, 20]];

Show[hitxyzplot, hitxyzlabel, littlesphereplot,
  ThreeAxes[0.5 (littleradius + bigradius)],
  Axes → True, Boxed → False, AxesLabel → {"x", "y", "z"},
  PlotRange → All, ViewPoint → 10 (A.{x, y, z}) × {1, 0, 0}];
```



Explain how the plot reflects the fact that
Min[xstretch,ystretch,zstretch] ||{x,y,z}|| ≤ ||A.{x,y,z}|| .
If xstretch,ystretch and zstretch are all positive, is it possible for A.{x,y,z} = {0,0,0} when {x,y,z}≠ {0,0,0}?

□G.8.c) Making a nicer plot by changing parameterizations

Here's a random 3D matrix A:

```
Clear[alignerframe, hangerframe];
{r, s, t} = {Random[Real, {0, Pi}],
   Random[Real, {0, 2 Pi}], Random[Real, {0, 2 Pi}]};

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

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

{xstretch, ystretch, zstretch} = {Random[Real, {0.5, 2}],
   Random[Real, {0.5, 2}], Random[Real, {0.5, 2}]};
stretcher = DiagonalMatrix[{xstretch, ystretch, zstretch}];

{r, s, t} = {Random[Real, {0, Pi}],
   Random[Real, {0, 2 Pi}], Random[Real, {0, 2 Pi}]};
        {hangerframe[1], hangerframe[2],
  hangerframe[3]} = {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
    Cos[s] Cos[t] Sin[r] + Cos[r] Sin[t], Sin[r] Sin[s]},
   {-Cos[t] Sin[r] - Cos[r] Cos[s] Sin[t],
    Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[r] Sin[s]},
   {Sin[s] Sin[t], -Cos[t] Sin[s], Cos[s]}};
hanger = Transpose[{hangerframe[1],
    hangerframe[2], hangerframe[3]}];

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

$$\begin{pmatrix} 1.10762 & -0.724646 & 0.32062 \\ 0.222818 & 0.0839392 & 1.55896 \\ -0.820013 & -0.374454 & 0.17348 \end{pmatrix}$$

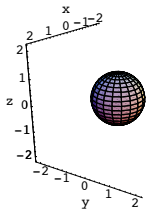
One usual way of paramterizing the 3D unit sphere is to go with the spherical coordinate parametrization:

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

{Cos[t] Sin[s], Sin[s] Sin[t], Cos[s]}

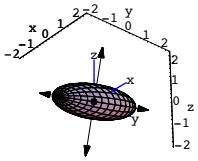
And plot:

```
{slow, shigh} = {0, π};
{tlow, thigh} = {0, 2 π};
ranger = 1.3 Max[1, Max[SingularValues[A][[2]]]];
ParametricPlot3D[{x[s, t], y[s, t], z[s, t]},
  {s, slow, shigh}, {t, tlow, thigh}, AxesLabel → {"x", "y", "z"},
  Boxed → False, PlotRange → {{-ranger, ranger},
    {-ranger, ranger}, {-ranger, ranger}}, ViewPoint → CMView];
```



To see what a hit with A does to the 3D unit sphere, you hit {x[s,t],y[st],z[s,t]} with A and plot:

```
Clear[hangerframe];
{hangerframe[1], hangerframe[2], hangerframe[3]} =
  SingularValues[A][[1]];

scalefactor = ranger;
hangerframeplot =
  {Arrow[scalefactor hangerframe[1], Tail → {0, 0, 0},
    VectorColor → Black], Arrow[scalefactor hangerframe[2],
    Tail → {0, 0, 0}, VectorColor → Black],
   Arrow[scalefactor hangerframe[3], Tail → {0, 0, 0},
    VectorColor → Black], Arrow[-scalefactor hangerframe[1],
    Tail → {0, 0, 0}, VectorColor → Black],
   Arrow[-scalefactor hangerframe[2], Tail → {0, 0, 0},
    VectorColor → Black], Arrow[-scalefactor hangerframe[3],
    Tail → {0, 0, 0}, VectorColor → Black]};

hitplot = ParametricPlot3D[A.{x[s, t], y[s, t], z[s, t]},
   {s, slow, shigh}, {t, tlow, thigh}, DisplayFunction → Identity];

Show[hitplot, Axes3D[2], hangerframeplot,
  Boxed → False, AxesLabel → {"x", "y", "z"}, PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  ViewPoint → 10 hangerframe[2], DisplayFunction → $DisplayFunction];
```



That's nice. The ellipsoid is skewed right on the plotted hangerframe as it should be.

But the plotting lines on the ellipsoid leave something to be desired because they don't merge at any of the places at which the hanger frame vectors pierce the skin of the ellipsoid.

To remedy this situation, go with this new parametrization of the 3D unit sphere:

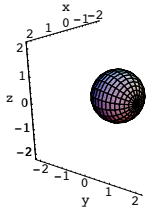
```
This parametrization incorporates the SVD aligner frame of A.
```

```
Clear[newx, newy, newz, s, t, alignerframe];
{alignerframe[1], alignerframe[2], alignerframe[3]} =
  SingularValues[A][[3]];

{newx[s_, t_], newy[s_, t_], newz[s_, t_]} =
    Sin[s] Cos[t] alignerframe[1] +
      Sin[s] Sin[t] alignerframe[2] +
        Cos[s] alignerframe[3]
```

```
{-0.31889 Cos[s] - 0.537279 Cos[t] Sin[s] + 0.780795 Sin[s] Sin[t],
 -0.947737 Cos[s] + 0.171891 Cos[t] Sin[s] - 0.268791 Sin[s] Sin[t],
 0.0102043 Cos[s] - 0.825703 Cos[t] Sin[s] - 0.564013 Sin[s] Sin[t]}
```

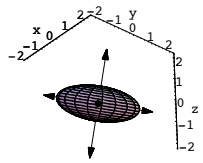
And plot:

```
{slow, shigh} = {0, π};
{tlow, thigh} = {0, 2 π};
ranger = 1.3 Max[1, Max[SingularValues[A][[2]]]];
ParametricPlot3D[{newx[s, t], newy[s, t], newz[s, t]},
  {s, slow, shigh}, {t, tlow, thigh}, AxesLabel → {"x", "y", "z"},
  Boxed → False, PlotRange → {{-ranger, ranger},
    {-ranger, ranger}, {-ranger, ranger}}, ViewPoint → CMView];
```



Now hit this new parameterization with A and plot:

```
Clear[hangerframe];
{hangerframe[1], hangerframe[2], hangerframe[3]} =
  SingularValues[A][[1]];

scalefactor = ranger;
hangerframeplot =
  {Arrow[scalefactor hangerframe[1], Tail → {0, 0, 0},
    VectorColor → Black], Arrow[scalefactor hangerframe[2],
    Tail → {0, 0, 0}, VectorColor → Black],
   Arrow[scalefactor hangerframe[3], Tail → {0, 0, 0},
    VectorColor → Black], Arrow[-scalefactor hangerframe[1],
    Tail → {0, 0, 0}, VectorColor → Black],
   Arrow[-scalefactor hangerframe[2], Tail → {0, 0, 0},
    VectorColor → Black], Arrow[-scalefactor hangerframe[3],
    Tail → {0, 0, 0}, VectorColor → Black]};

newhitplot = ParametricPlot3D[A.{newx[s, t], newy[s, t], newz[s, t]},
   {s, slow, shigh}, {t, tlow, thigh}, DisplayFunction → Identity];

Show[newhitplot, hangerframeplot,
  Boxed → False, AxesLabel → {"x", "y", "z"}, PlotRange →
   {{-ranger, ranger}, {-ranger, ranger}, {-ranger, ranger}},
  ViewPoint → 10 hangerframe[2], DisplayFunction → $DisplayFunction];
```



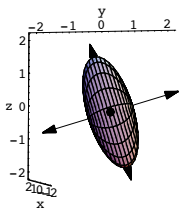
Neato.
Try it for a new matrix:

```
Clear[alignerframe, hangerframe];

{r, s, t} = {Random[Real, {0, π}],
   Random[Real, {0, 2 π}], Random[Real, {0, 2 π}]};

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

            {xstretch, ystretch, zstretch} =
  {Random[Real, {0.5, 2}],
   Random[Real, {0.5, 2}], Random[Real, {0.5, 2}]};
stretcher = DiagonalMatrix[{xstretch, ystretch, zstretch}];
```

```
      {r, s, t} = {Random[Real, {0, Pi}],
   Random[Real, {0, 2 Pi}], Random[Real, {0, 2 Pi}]};
      {hangerframe[1], hangerframe[2],
  hangerframe[3]} = {{Cos[r] Cos[t] - Cos[s] Sin[r] Sin[t],
    Cos[s] Cos[t] Sin[r] + Cos[r] Sin[t], Sin[r] Sin[s]},
   {-Cos[t] Sin[r] - Cos[r] Cos[s] Sin[t],
    Cos[r] Cos[s] Cos[t] - Sin[r] Sin[t], Cos[r] Sin[s]},
   {Sin[s] Sin[t], -Cos[t] Sin[s], Cos[s]}};
      hanger = Transpose[
   {hangerframe[1], hangerframe[2], hangerframe[3]}];

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

Clear[newx, newy, newz, s, t, alignerframe];
{alignerframe[1], alignerframe[2], alignerframe[3]} =
  SingularValues[A][[3]];
{newx[s_, t_], newy[s_, t_], newz[s_, t_]} =
    Sin[s] Cos[t] alignerframe[1] +
   Sin[s] Sin[t] alignerframe[2] + Cos[s] alignerframe[3];

Clear[hangerframe];
{hangerframe[1], hangerframe[2], hangerframe[3]} =
  SingularValues[A][[1]];

scalefactor = ranger;
hangerframeplot =
   {Arrow[scalefactor hangerframe[1],
    Tail -> {0, 0, 0}, VectorColor -> Black],
   Arrow[scalefactor hangerframe[2],
    Tail -> {0, 0, 0}, VectorColor -> Black],
   Arrow[scalefactor hangerframe[3],
    Tail -> {0, 0, 0}, VectorColor -> Black],
   Arrow[-scalefactor hangerframe[1],
    Tail -> {0, 0, 0}, VectorColor -> Black],
   Arrow[-scalefactor hangerframe[2],
    Tail -> {0, 0, 0}, VectorColor -> Black],
   Arrow[-scalefactor hangerframe[3],
    Tail -> {0, 0, 0}, VectorColor -> Black]};

newhitplot = ParametricPlot3D[A.{newx[s, t], newy[s, t], newz[s, t]},
   {s, slow, shigh}, {t, tlow, thigh},
 DisplayFunction -> Identity];

 Show[newhitplot, hangerframeplot,
  Boxed -> False, AxesLabel -> {"x", "y", "z"},
 PlotRange -> {{-ranger, ranger},
```

```
    {-ranger, ranger}, {-ranger, ranger}},
  ViewPoint -> 10 hangerframe[2],
   DisplayFunction -> $DisplayFunction];
```

$$\begin{pmatrix} -0.687094 & 0.460826 & 1.55007 \\ -0.290678 & -0.801034 & 0.23619 \\ 1.56137 & 0.571848 & 0.58042 \end{pmatrix}$$



Rerun a couple of times.

Speculate on why the new parametrization of the 3D unit sphere:
{newx[s, t], newy[s, t], newz[s, t]} =
Sin[s] Cos[t] alignerframe[1] + Sin[s] Sin[t] alignerframe[2] + Cos[s] alignerframe[3]
gives such a pleasing plot.

G.9) Isometries and rigid body motions

□G.9.a.i) Matrices whose hits do not change volume measurements

If A is a 3D matrix and hits with A do not change volume measurements, then what is |Det[A]| guaranteed to be?

□G.9.a.ii) Make a matrix whose hits do not change volume measurements but do change some length measurements

Make a 3D matrix A whose hits do not change any volume measurements but do change some length measurements.

□G.9.b.i) SVD stretch factors of isometries in 3D

Many folks say that a 3D matrix A is an isometry if
$\|A.X\| = \|X\|$
for all 3D vectors X.
Saying that a 3D matrix A is an isometry is the same as saying that hits with A do not change lengths.
Examples of isometry matrices are: aligners, hangers, rotations, frame flippers (reflection matrices).

If A is a 3D isometry matrix (i.e. hits with A do not change lengths), then what are the three SVD stretch factors of A guaranteed to be?

□G.9.b.ii) If A is a matrix all three SVD stretch factors equal to 1, the A is an isometry

When you make a 3D matrix A with all three SVD stretch factors equal to 1, then you are guaranteed that
A.alignerframe[1] = hangerframe[1]
A.alignerframe[2] = hangerframe[2]
A.alignerframe[3] = hangerframe[3]
When you take any 3D X and resolve it into components, you get

$$X = \sum_{j=1}^{3} (X.\text{alignerframe}[j])\,\text{alignerframe}[j]$$

and

$$\|X\| = \sqrt{\sum_{j=1}^{3} (X.\text{alignerframe}[j])^2}\,.$$

When you hit the same 3D X with A, you get

$$A.X = \sum_{j=1}^{3} (X.\text{alignerframe}[j])\,\text{hangerframe}[j]$$

and

$$\|A.X\| = \sqrt{\sum_{j=1}^{3} (X.\text{alignerframe}[j])^2}\,.$$

Is this enough to tell you that saying that a 3D matrix is an isometry is the same as saying that all three SVD stretch factors of A are equal to 1?

□G.9.b.iii) Agree or disagree

Agree or disagree with each of the following statements. Explain yourself:
▪ If A is an isometry matrix, then |Det[A]| = 1.
Agree..... Disagree.....
▪ There are matrices B other than isometry matrices with |Det[B]| = 1.
Agree..... Disagree.....

□G.9.c.i) Rigid body motion matrices

Rigid body motion matrices are isometry matrices whose hits do not change orientations. A hit with a rigid body motion matrix represents what get when you physically pick up a surface and move it without changing it in any other way.

Aligners and hangers coming from right hand frames are rigid body motion matrices.
Rotation matrices are rigid body motion matrices.

Aligners and hangers coming from left hand frames are not rigid body motion matrices.
Reflection matrices are not rigid body motion matrices.
Explain these:
▪ If A is a 3D isometry matrix and Det[A] = 1, then A is a rigid body motion matrix.
Put answer here.

▪ If A is a rigid body motion matrix, then A is a 3D isometry matrix and Det[A] = 1.
Put answer here.

▪ If A is a 3D isometry matrix and Det[A] = -1, then A is not a rigid body motion matrix.
Put answer here.

▪ If A is a 3D plane flipper matrix, then A is not a rigid body motion matrix.
Put answer here.

□G.9.c.ii) Calculus Cal

That silly pest Calculus Cal came bounding into the lab and announced that a matrix is a rigid body motion matrix is just any matrix whose determinant is 1.
Tell Cal why he is wrong.

□G.9.d) FYI Only: A pearl from Euler

The great Euler established once and for all that all rigid body motion matrices are in fact rotations about lines.
His argument depends on this fact.
Fact: If A is any 3D matrix and charpolynomial[s] = Det[A - s Identity], then you are guaranteed that
charpolynomial[s]→ -∞
as s→ +∞.
Here's why this is true:
Go with any cleared 3D matrix A:

```
Clear[a, b, c, d, e, f, g, h , i];
A = {{a, b, c}, {d, e, f}, {g, h, i}};
MatrixForm[A]
```

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

The characteristic polynomial of A is:

```
Clear[charpolynomial];
    charpolynomial[s_] = Det[A - s IdentityMatrix[3]]
```

$$-ceg + bfg + cdh - afh - bdi + aei + bds - aes + cgs + fhs - ais - eis + as^2 + es^2 + is^2 - s^3$$

As s→ +∞, the dominant term $-s^3$ pulls charpolynomial[s] to -∞

Now here is the gist of Euler's explanation:
Go with a rigid body motion matrix A.
Note:
▪ charpolynomial[0] = Det[A - 0 Identity] = Det[A] = 1 (because A is a rigid body motion matrix)

▪ For large s, charpolynomial[s] is negative (because charpolynomial[s]->-infinity as s->infinity.)

This tells you that charpolynomial[s] = 0 for some positive s = ss. (Because the graph of charpolynomial[s] goes from positive to negative as s advances from 0.)

This tells you that charpolynomial[ss] = Det[A - ss Identity] = 0.

This tells you that there is a non-zero vector X so that
(A - ss Identity).X = {0,0,0}.
This is the same as
A.X = ss .X.
This tells you that ss = 1 because hits with A do not change lengths.
So
A.X = X.
It turns out that this vector X defines the line about which hits with A rotate.