Matrices, Geometry&*Mathematica*
Authors: Bruce Carpenter, Bill Davis and Jerry Uhl ©2001
Producer: Bruce Carpenter
Publisher: Math Everywhere, Inc.

MGM.08 Subspaces, Spans , Dimension, Linear Independence, Basis, Orthonormal Basis
TUTORIALS

T.1) Orthonormal bases are bases consisting of mutually perpendicular unit vectors - they are perpendicular frames.
The advantages and pleasures of having an orthonormal basis.
Getting orthonormal bases via SVD hanger frames for the spanner matrix

Most folks say that a basis for a subspace S of nD is any linearly independent spanning set
{spanner[1], spanner[2], . . ., spanner[k]}
for S.
The same folks say that a basis {spanner[1], spanner[2], . . ., spanner[k]} for a subspace S of nD is an **orthonormal basis** for S if
{spanner[1], spanner[2], . . ., spanner[k]}
is also a perpendicular frame (This is the same as saying.
spanner[i].spanner[i] = 1 for i = 1, 2, ... k
and
spanner[i].spanner[j] = 0 for i ≠ j.)

□T.1.a.i) Life in the orthonormal world is good

List some of the advantages you get when have an orthonormal basis for a subspace S of kD.

□**Answer:**

Everything gets really easy.

Some examples:

□Example 1: Easy calculation of basis coefficients

If {spanner[1], spanner[2], . . ., spanner[k]} is an orthonormal basis of a subspace S of nD, then every X in S is of the form

$$X = \sum_{i=1}^{k} x[i]\,\text{spanner}[i]$$

with x[i] = X.spanner[i].

```
No need to fool around with PseudoInverse of the Spanner Matrix..
```

□Example 2: Easy calculation closest vectors

If {spanner[1], spanner[2], . . ., spanner[k]} is an orthonormal basis of a subspace S of nD and X is a given vector in nD, then the vector in S closest to X is

$$\text{XSclosest} = \sum_{i=1}^{k} x[i]\,\text{spanner}[i]$$

with x[i] = X.spanner[i].

```
No need to fool around with the Spanner Matrix or the Sprojection matrix.
```

□Example 3: Easy calculation Sprojection matrix

If {spanner[1], spanner[2], . . ., spanner[k]} is an orthonormal basis of a subspace S of nD then you can go directly to the Sprojection matrix

Sprojection = hanger.aligner

where

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

and

hanger = $\{\text{spanner}[1], \text{spanner}[2], \ldots, \text{spanner}[k]\}^t$

so that

spanner[i] is the ith horizontal row of aligner

and

spanner[j] is the jth vertical column of hanger.

```
No need to fool around with the Spanner Matrix or the Sprojection matrix.
```

Life in the orthonormal world is about as good as it gets.

□T.1.a.ii) What happens when you don't have an orthonormal basis

Do those nice clean formulas discussed in part i) work for a given basis that is not an orthonormal basis?

□**Answer:**

No way.

□T.1.b) Using SVD analysis of the Spanner Matrix to get an orthonormal basis

A subspace S of 6D is defined by the following given spanning set:

```
Clear[i, spanner];
spanner[1] = {3.43, -0.57, 2.19, -0.94, 1.28};
spanner[2] = {0.97, 1.29, 0.83, 5.23, 1.26};
spanner[3] = {-6.14, 3.08, -5.00, -1.97, -0.96};
spanner[4] = {-3.68, 1.22, -3.64, -8.14, -0.94};
spanner[5] = {2.71, -2.51, 2.81, 2.91, -0.32};
spanners = Table[spanner[i], {i, 1, 5}]
```

```
{{3.43, -0.57, 2.19, -0.94, 1.28},
 {0.97, 1.29, 0.83, 5.23, 1.26}, {-6.14, 3.08, -5., -1.97, -0.96},
 {-3.68, 1.22, -3.64, -8.14, -0.94}, {2.71, -2.51, 2.81, 2.91, -0.32}}
```

Use SVD analysis of the corresponding spanner matrix to come up with an orthonormal basis for S.
Use your orthonormal basis to build the Sprojection matrix.

□**Answer:**

The hangerframe for the Spanner Matrix is a perpendicular frame (orthonormal basis) that spans S.

This hangerframe is guaranteed to be an orthonormal basis of S.

Here it is:

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

Clear[ortho, k];
ortho[k_] := SingularValues[SpannerMatrix][[1, k]]
Table[ortho[k], {k, 1, Sdim}]
```

```
{{-0.550517, 0.225396, -0.486973, -0.628875, -0.116196},
 {-0.458778, 0.348461, -0.301171, 0.759864, -0.000785019},
 {0.334709, 0.705309, -0.0275415, -0.131643, 0.610263}}
```

You can use this to make the Sprojection very quickly from rank one matrices

```
rankOne[k_] := Transpose[{ortho[k]}].{ortho[k]};
```

$$\text{sum} = \sum_{k=1}^{\text{Sdim}} \text{rankOne}[k];$$

```
MatrixForm[sum]
```

```
 0.625576   -0.047877   0.397039   -0.0464648    0.268588
-0.047877    0.669689  -0.234133    0.0301879    0.40396
 0.397039   -0.234133   0.328605    0.0810221    0.040013
-0.0464648   0.0301879  0.0810221   0.990207    -0.00786101
 0.268588    0.40396    0.040013   -0.00786101   0.385923
```

Or you can make the Sprojection matrix directly from SVD:

```
aligner = {ortho[1], ortho[2], ortho[3]};
hanger = Transpose[{ortho[1], ortho[2], ortho[3]}];
SVDSprojection = hanger.aligner;
        MatrixForm[SVDSprojection]
```

```
 0.625576   -0.047877   0.397039   -0.0464648    0.268588
-0.047877    0.669689  -0.234133    0.0301879    0.40396
 0.397039   -0.234133   0.328605    0.0810221    0.040013
-0.0464648   0.0301879  0.0810221   0.990207    -0.00786101
 0.268588    0.40396    0.040013   -0.00786101   0.385923
```

Compare:

```
Sprojection = SpannerMatrix.PseudoInverse[SpannerMatrix];
MatrixForm[Sprojection]
```

```
 0.625576   -0.047877   0.397039   -0.0464648    0.268588
-0.047877    0.669689  -0.234133    0.0301879    0.40396
 0.397039   -0.234133   0.328605    0.0810221    0.040013
-0.0464648   0.0301879  0.0810221   0.990207    -0.00786101
 0.268588    0.40396    0.040013   -0.00786101   0.385923
```

Nailed it.

T.2) Vivid Graphics:
Alien plots coming from projections of highD surfaces onto three dimensional subspaces

```
If the plots you are about to see strike your fancy in a big way, get the book
        "Beyond the Third Dimension" by Thomas F. Banchoff
           (Scientific American Library, 1996).
```

□T.2.a.i) Looking at a three dimensional photo of a 4D surface

Here's a function f[s,t]:

```
Clear[f, s, t];
f[s_, t_] = {Cos[s], Sin[s], Cos[t], Sin[t]}
```

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

```
Note the four slots.
```

If you could plot this function, the result would be a surface in 4D which some advanced folks call by the name "Clifford Torus."
Unfortunately, you can't plot this surface, but you can go with any three dimensional subspace S of 4D and plot the projection of this 4D surface onto S.
Just how do you go about doing this?

□**Answer:**

You do this by going with a spanning set for your three dimensional subspace S or 4D..

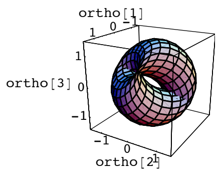
Here's a sample:

```
Clear[s, i, spanner];
spanner[1] = {0.327337, -0.121305, 0.39909, 0.397503};
```

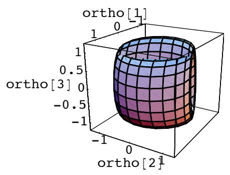
```
spanner[2] = {0.871239, 0.659994, -0.659799, 0.497028};
spanner[3] = {0.028996, 0.684016, 0.746671, -0.391136};
spanners = Table[spanner[i], {i, 1, 3}]
```
```
{{0.327337, -0.121305, 0.39909, 0.397503},
 {0.871239, 0.659994, -0.659799, 0.497028},
 {0.028996, 0.684016, 0.746671, -0.391136}}
```
Next you go from this spanning set for S to an orthonormal basis for S
```
SpannerMatrix = Transpose[spanners];
Clear[ortho, k];
ortho[k_] := SingularValues[SpannerMatrix][[1, k]];
Table[ortho[k], {k, 1, 3}]
```
```
{{0.614538, 0.322678, -0.571578, 0.43763},
 {0.270615, 0.750688, 0.573952, -0.183889},
 {-0.34538, 0.426903, -0.586325, -0.595558}}
```
To plot the projection of the surface onto S, you plot

{f[s, t].ortho[1], f[s, t].ortho[2], f[s, t].ortho[3]}

and label the corresponding axes accordingly.

Here you go:
```
Clear[surfaceprojectionplotter, f, s, t];
f[s_, t_] = {Cos[s], Sin[s], Cos[t], Sin[t]};

surfaceprojectionplotter[s_, t_] =
    {f[s, t].ortho[1], f[s, t].ortho[2], f[s, t].ortho[3]};

{slow, shigh} = {0, 2 π};
{tlow, thigh} = {0, 2 π};
ParametricPlot3D[surfaceprojectionplotter[s, t],
  {s, slow, shigh}, {t, tlow, thigh}, Axes -> True,
  AxesLabel -> {"ortho[1]", "ortho[2]", "ortho[3]"},
ViewPoint -> CMView];
```


How does that grab you, sports fans

See what happens when you project onto random three dimensional subspaces of 4D:
```
Clear[s, i, spanner];
spanner[1] = Table[Random[Real, {-1, 1}], {i, 1, 4}];
spanner[2] = Table[Random[Real, {-1, 1}], {i, 1, 4}];
spanner[3] = Table[Random[Real, {-1, 1}], {i, 1, 4}];
spanners = Table[spanner[i], {i, 1, 3}];
SpannerMatrix = Transpose[spanners];
Clear[ortho, k];
ortho[k_] := SingularValues[SpannerMatrix][[1, k]];

Clear[surfaceprojectionplotter, f, s, t];
f[s_, t_] = {Cos[s], Sin[s], Cos[t], Sin[t]};

surfaceprojectionplotter[s_, t_] =
    {f[s, t].ortho[1], f[s, t].ortho[2], f[s, t].ortho[3]};

{slow, shigh} = {0, 2 π};
{tlow, thigh} = {0, 2 π};
ParametricPlot3D[surfaceprojectionplotter[s, t],
  {s, slow, shigh}, {t, tlow, thigh}, Axes -> True,
  AxesLabel -> {"ortho[1]", "ortho[2]", "ortho[3]"},
ViewPoint -> CMView];
```


```
Rerun several times and if you're on a fast machine rerun many, many times.
```
Each of these views is a three dimensional slice of a 4D surface.

Almost as good as the Hubble telescope.

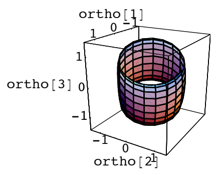
□ T.2.a.ii) Why it's just a little bit interesting

Take the function f[s,t] used to generate the plots in parts i and ii):
```
f[s, t]
```
```
{Cos[s], Sin[s], Cos[t], Sin[t]}
```
Look at:
```
f[s, t].f[s, t]
```
```
Cos[s]^2 + Cos[t]^2 + Sin[s]^2 + Sin[t]^2
```
Apply trig identities:
```
TrigExpand[f[s, t].f[s, t]]
```
```
2
```
This tells you that no matter what s and t are, you are guaranteed that f[s,t] is on the surface of the 4D sphere of radius $\sqrt{2}$ centered at {0,0,0,0}.
Why is this fact just a little bit mind-boggling?

□ Answer:

This tells you that all the plots produced above are projections of pieces of the 4D sphere of radius $\sqrt{2}$ centered at {0,0,0,0} onto three dimensional subspace of 4D.

This might make you expect that these projections would look like pieces of spheres.-after all projections of pieces of spheres in 3D onto two dimensional subspaces of 3D look like pieces of circles.

But they don't:
```
Clear[s, i, spanner];
spanner[1] = Table[Random[Real, {-1, 1}], {i, 1, 4}];
spanner[2] = Table[Random[Real, {-1, 1}], {i, 1, 4}];
spanner[3] = Table[Random[Real, {-1, 1}], {i, 1, 4}];
spanners = Table[spanner[i], {i, 1, 3}];
SpannerMatrix = Transpose[spanners];
Clear[ortho, k];
ortho[k_] := SingularValues[SpannerMatrix][[1, k]];

Clear[surfaceprojectionplotter, f, s, t];
f[s_, t_] = {Cos[s], Sin[s], Cos[t], Sin[t]};

surfaceprojectionplotter[s_, t_] =
    {f[s, t].ortho[1], f[s, t].ortho[2], f[s, t].ortho[3]};

{slow, shigh} = {0, 2 π};
{tlow, thigh} = {0, 2 π};
ParametricPlot3D[surfaceprojectionplotter[s, t],
  {s, slow, shigh}, {t, tlow, thigh}, Axes -> True,
  AxesLabel -> {"ortho[1]", "ortho[2]", "ortho[3]"},
ViewPoint -> CMView];
```


Moral: 4D spheres are alien beasts but they are real.

T.3) The perpendicular complement $S^{\perp}$ of a subspace S of kD is the subspace of kD consisting of all of all Y that are perpendicular to everything in S.

The perpendicular complement $S^{\perp}$ of a subspace S is also the subspace of kD consisting of all hits with Identity - Sprojection.

How to take a kD vector Y and put it in the form $Y = YS + YS^{\perp}$ with YS in S , $YS^{\perp}$ in $S^{\perp}$ and

$$|| Y || = \sqrt{|| YS ||^2 + || YS^{\perp} ||^2}$$

Extending a basis of subspace of kD to a basis of all of kD

□ T.3.a.i) The perpendicular complement $S^{\perp}$ of a subspace S of kD is the subspace of kD consisting of all hits with

Identity - Sprojection

The perpendicular complement $S^{\perp}$ of a subspace of kD is the subspace of kD consisting of all hits with

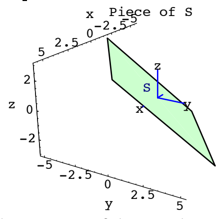
Identity - Sprojection.

A handy spanning set for $S^{\perp}$ is the set of vertical columns of

Identity - Sprojection.

Here's a spanning set for a subspace of 3D:
```
Clear[s, i, spanner];
spanner[1] = {1.2, 0.9, 0};
spanner[2] = {-1.0, 1.2, -1.5};
spanners = Table[spanner[i], {i, 1, 2}];
```
And a plot of part of this subspace:

```
h = 3;
k = 2;
Clear[s, t];
planeplot = ParametricPlot3D[s spanner[1] + t spanner[2], {s, -h, h},
   {t, -k, k}, PlotPoints → {2, 2}, DisplayFunction → Identity];

Slabel = Graphics3D[{NavyBlue,
    Text["S", 0.5 spanner[1] - 0.5 spanner[2] + {0, 0, 0.3}]}];

Splot = Show[planeplot, Slabel,
   Axes3D[2], ViewPoint → CMView, PlotRange → All,
   Axes → True, AxesLabel → {"x", "y", "z"}, Boxed → False,
   PlotLabel → "Piece of S", DisplayFunction → $DisplayFunction];
```



Throw some of the members of $S^{\perp}$ into this plot and describe what you see.

Answer:

Set up the Sprojection matrix:

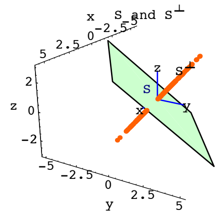
```
SpannerMatrix = Transpose[spanners];
Sprojection = SpannerMatrix.PseudoInverse[SpannerMatrix];
MatrixForm[Sprojection]
```

$$\begin{pmatrix} 0.827056 & 0.230592 & 0.299769 \\ 0.230592 & 0.692544 & -0.399693 \\ 0.299769 & -0.399693 & 0.4804 \end{pmatrix}$$

$S^{\perp}$ consists of all hits with Identity - Sprojection.

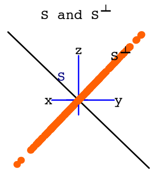
See some::

```
a = 3;
point[k_] := {Random[Real, {-a, a}],
   Random[Real, {-a, a}], Random[Real, {-a, a}]};
randomhits = Table[Graphics3D[
   {PointSize[0.02], CadmiumOrange, Point[
     (IdentityMatrix[3] - Sprojection).point[k]]}], {k, 1, 300}];
label = Graphics3D[{Black, Text["S⊥",
    (IdentityMatrix[3] - Sprojection).(1.1 {a, a, a})]}];
both = Show[Splot, randomhits, label, PlotLabel -> "S and S⊥"];
```



Look at everything from a view point parallel to S:

```
Show[both, ViewPoint -> 12 spanner[1], Axes -> False];
```



The vectors in $S^{\perp}$ are perpendicular to everything in S.

If you want to see why this is automatic, then go on.

T.3.a.ii) If Y is in $S^{\perp}$, then Y is perpendicular to everything in S

Explain this:
Given a subspace S of kD and a vector X in $S^{\perp}$, then it's automatic that X is perpendicular to everything in S.

Answer:

Go with an X in $S^{\perp}$. This means X is a hit with Identity - Sprojection.

In other words, there is guaranteed to be a Y in kD with

X = (Identity - Sprojection).Y = Y - Sprojection.Y

Remember this: Sprojection.Y is the vector in S that is **closest** to Y.

And because perpendicular distance is the closest distance,

X = Y - Sprojection.Y

is perpendicular to everything in S.

T.3.a.iii) You can decompose $Y = YS + YS^{\perp}$ with YS in S, $YS^{\perp}$ in $S^{\perp}$ and

$$\| Y \| = \sqrt{\| YS \|^2 + \| YS^{\perp} \|^2}$$

Explain this:
Given a subspace S of kD and given any kD vector Y, you can decompose

$Y = YS + YS^{\perp}$

with YS in S and with $YS^{\perp}$ in $S^{\perp}$ and with

$$\| Y \| = \sqrt{\| YS \|^2 + \| YS^{\perp} \|^2}.$$

Answer:

Go with your Y and set

YS = Sprojection. Y

and set

$YS^{\perp}$ = (Identity - Sprojection). Y = Y - Sprojection. Y.

It' s now automatic that

$Y = YS + YS^{\perp}$

and that YS is in S and $YS^{\perp}$ is in $S^{\perp}$.

And:

$$\| Y.Y \|^2 = Y.Y = (YS + YS^{\perp}).(YS + YS^{\perp})$$
$$= (YS.YS) + (YS.YS^{\perp}) + (YS^{\perp}.YS) + (YS^{\perp}.YS^{\perp})$$
$$= (YS.YS) + 0 + 0 + (YS^{\perp}.YS^{\perp})$$
$$= \| YS \|^2 + \| YS^{\perp} \|^2.$$

So

$$\| Y \| = \sqrt{\| YS \|^2 + \| YS^{\perp} \|^2}$$

T.3.a.iv) If Y is perpendicular to everything in S, then Y is in $S^{\perp}$

Explain this:
Go with a subspace S of kD.
If a kD vector Y is perpendicular to everything in S, then it's automatic that Y is in $S^{\perp}$.

Answer:

Go with a Y that is perpendicular to everything in S.

Decompose

$Y = YS + YS^{\perp}$

with YS in S and $YS^{\perp}$ in $S^{\perp}$.

The goal is to explain why YS = {0,0,. . .,0}, forcing $Y = YS^{\perp}$ and putting Y into $S^{\perp}$.

To this end, take the equation

$Y = YS + YS^{\perp}$.

Dot YS through this equation to get

$YS.Y = YS.YS + YS.YS^{\perp}$.

This is the same as

$0 = YS.YS + 0$

```
                           Reasons:
 YS.Y = 0 because YS is in S and Y is perpendicular to everything in S.
YS.YS⊥ = 0 because YS is in S and YS⊥ is perpendicular to everything in S.
```

This results in

$0 = YS.YS = \| YS \|^2$

and leads to the inescapable conclusion that

YS = {0,0,. . . ,0}.

T.3.a.v) $S^{\perp}$ consists exactly of all Y that are perpendicular to everything in S

Explain this:
Go with a subspace S of kD. You are guaranteed that $S^{\perp}$ consists exactly of all Y that are perpendicular to everything in S.

Answer:

Put PS equal to the set of vectors in kD that are perpendicular to everything in S.

Part ii) above says that if Y is in $S^{\perp}$, then Y is perpendicular to everything in S.

So $S^{\perp} \subset PS$.

Part iv) above says that if Y is perpendicular to everything in S, then Y is in $S^{\perp}$.

So $PS \subset S^{\perp}$.

The conclusion:

$PS = S^{\perp}$.

T.3.a.vi) The projection matrix for Perp[S] is (Identity - Sprojection)

Given a subspace S of kD,you can express the projection matrix for Perp[S] as
Perp[S] projection = IdentityMatrix − Sprojection.
Explain why this works.

Answer:

Take any vector X in kD
Put

Y = (IdentityMatrix – Sprojection).X

To explain why

Perp[S] projection = (IdentityMatrix – Sprojection),

you've got to explain why
->Y is in Perp[S]
and why
-> Y is closer to X than any other member of Perp[S].

□ Explanation of why Y is in Perp[S]

Because Sprojection.X is the closest member of S to X and because perpendicular distance is the shortest distance.

Y = (IdentityMatrix – Sprojection).X = X – Sprojection.X

is perpendicular to everything in S. This tells you that Y is Perp[S].

□ Explanation of why Y is closer to X than any other member of Perp[S].

Because perpendicular distance is the shortest distance, this is the same as explaining why X - Y is perpendicular to everything in Perp[S].
To do this, take any Z in Perp[S] and look at this little calculation:

Z.(X - Y) = Z.(X - (X - Sprojection.X)) = Z.(Sprojection.X) = 0

because Z is In Perp[S] and Sprojection.X is in S and everything in Perp[S] is perpendicular to everything in S.

This tells you that if you take any member of Perp[S], then (X - Y) is perpendicular to it. In other words,
X - Y is perpendicular to everything in Perp[S].
Done.

□ T.3.b.i) Using the $S^\perp$ idea to extend a given basis for a subspace S of kD to a basis for all of kD

Here's a basis (linearly independent spanning set) for a subspace S of 5D

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

```
spanner[2] = Table[Random[Real, {-2, 2}], {j, 1, 5}];
spanner[3] = Table[Random[Real, {-2, 2}], {j, 1, 5}];
spanners = Table[spanner[i], {i, 1, 3}]
```

```
{{-1.49252, 1.02121, 1.10839, -0.373076, -1.44557},
 {-0.579212, -1.49713, -0.164263, 1.17257, -0.171403},
 {0.0261164, -1.84871, -0.229586, 1.03194, 0.375104}}
```

Check to see that these spanners are a basis of S:

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

```
3
If the output of the last cell is not 3, go back to the beginning and rerun.
```

Come up with two 5D vectors X[1] and X[2] so that
when you put

spanner[4] = X[1] and spanner[5] = X[2],

then

{spanner[1], spanner[2], spanner[3], spanner[4], spanner[5]}

is a basis (linearly independent spanning set) for all of 5D.

□ Answer:

Make the Sprojection matrix:

```
SpannerMatrix = Transpose[spanners];
Sprojection = SpannerMatrix.PseudoInverse[SpannerMatrix];
MatrixForm[Sprojection]
```

```
 0.530983   -0.0263806  -0.117017   -0.218562    0.432298
-0.0263806   0.844447   -0.190262   -0.261823   -0.160964
-0.117017   -0.190262    0.751823   -0.352017   -0.113032
-0.218562   -0.261823   -0.352017    0.494012   -0.0986265
 0.432298   -0.160964   -0.113032   -0.0986265   0.378734
```

You can extend the original spanning set of S by throwing in a basis of $S^\perp$.

Because $S^\perp$ is all hits with Identity - Sprojection, you can get a basis for $S^\perp$ by looking at stretch factors and the hanger frame for Identity - Sprojection:

```
A = IdentityMatrix[5] - Sprojection;
stretches = SingularValues[A][[2]]
```

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

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

```
{{0, -0.392517, -0.467954, -0.635718, -0.472029},
 {-0.684848, -0.0385204, -0.170866, -0.319139, 0.631232}}
```

Put spanner[4] = hangerframe[1] and spanner[5] = hangerframe[2] to get your basis of all of 5D:

```
spanner[4] = hangerframe[1];
spanner[5] = hangerframe[2];
extendedspanners = Table[spanner[k], {k, 1, 5}]
```

```
{{-1.49252, 1.02121, 1.10839, -0.373076, -1.44557},
 {-0.579212, -1.49713, -0.164263, 1.17257, -0.171403},
 {0.0261164, -1.84871, -0.229586, 1.03194, 0.375104},
 {0, -0.392517, -0.467954, -0.635718, -0.472029},
 {-0.684848, -0.0385204, -0.170866, -0.319139, 0.631232}}
```

Check to see that {spanner[1],spanner[2],spanner[3],spanner[4],spanner[5]} spans all five dimensions of 5D:

```
extendedSpannerMatrix = Transpose[extendedspanners];
rank = Length[SingularValues[extendedSpannerMatrix][[2]]]
```

```
5
```

Yep, {spanner[1],spanner[2],spanner[3],spanner[4],spanner[5]} spans all five dimensions of 5D.
And you're out of here.

T.4) The null space of a matrix A is the subspace of hitdimD of vectors hit onto {0,0,,...,0} by A. The SVD aligner frame for A spans a subspace S of hitdimD. The null space of A is the perpendicular complement $S^\perp$ of this subspace. This fact tells you how to come up with an orthonormal basis for the null space of A

□ T.4.a.i) The null space of A is the subspace of hitdimD of vectors hit onto {0,0,,...,0} by A

Here's a matrix:

```
Clear[a];
a[i_, j_] := Random[Real, {-5, 5}]
A = Table[a[i, j], {i, 1, 3}, {j, 1, 7}];
MatrixForm[A]
```

```
 3.74849  -0.662407   3.60943   -1.56621   1.72004    0.823738  -4.7405
-1.93859  -2.02499   -0.444953  -2.29357   0.290423   3.9077    -1.8310
 4.15446  -0.966742  -0.681643   0.23754  -0.417035   3.96797   -1.0598
```

When you ask Mathematica for the null space of A, you get:

```
NullSpace[A]
```

```
{{-0.319571, 0.258227, 0.279332,
   0.747602, -0.0849843, 0.311735, -0.299761},
  {0, 0.0749981, -0.228006, 0.172414, 0.942748, 0.0994647, 0.118301},
  {0, 0.75125, 0.287154, -0.385375, 0.00673947, 0.336794, 0.301955},
  {0, -0.482643, 0.633566, 0.107473, 0.0849562, 0.146141, 0.570548}}
```

What are these vectors?

□ Answer:

Look again:

```
Clear[spanner];
{spanner[1], spanner[2], spanner[3], spanner[4]} = NullSpace[A]
```

```
{{-0.319571, 0.258227, 0.279332,
   0.747602, -0.0849843, 0.311735, -0.299761},
  {0, 0.0749981, -0.228006, 0.172414, 0.942748, 0.0994647, 0.118301},
  {0, 0.75125, 0.287154, -0.385375, 0.00673947, 0.336794, 0.301955},
  {0, -0.482643, 0.633566, 0.107473, 0.0849562, 0.146141, 0.570548}}
```

Mathematica delivers a basis for the subspace of 7D consisting of the Y's in 7D that are squashed to {0,0,. . .,0} by a hit with A.
Try this out for cleared coefficients x[j]:

$$\text{Expand}\left[A.\left(\sum_{j=1}^{4} x[j]\ \text{spanner}[j]\right)\right]$$

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

This subspace of 7D is called the Null Space of A.

□ T.4.a.ii) The SVD aligner frame for A spans a subspace S of hitdimD. The null space of A is the perpendicular complement $S^\perp$ of this subspace. This fact tells you how to come up with an orthonormal basis for the null space of A

Here's a new matrix A:

```
hangdim = 4;
hitdim = 7;
Clear[a];
a[i_, j_] := Random[Real, {-5, 5}]
A = Table[a[i, j], {i, 1, hangdim}, {j, 1, hitdim}];
MatrixForm[A]
```

```
-4.18849   2.00311  -1.96979   0.191643   1.47391    3.39368   4.59642
 3.4716   -4.34983   3.13421   1.53501    0.496594   1.09513   0.427785
-3.75541   1.58889  -2.07384   1.27333    2.21133   -2.72946   2.68862
-3.30964   3.24336   3.33041   1.87711   -0.312743   0.213159 -1.86124
```

Use SVD and perpendicular complements to come up with an orthonormal basis for the null space of A.

Answer:

Calculate the SVD alignerframe consisting of vectors with corresponding positive stretch factors:

```
rankA = Length[SingularValues[A][[2]]];
Clear[alignerframe];
alignerframe[k_] := SingularValues[A][[3, k]];
Sspanners = Table[alignerframe[k], {k, 1, rankA}]
```

```
{{-0.699413, 0.492867, -0.317343, 0.0407122,
  0.175461, 0.0206534, 0.366505}, {0.255912, -0.469264,
  -0.468648, -0.148536, 0.198659, 0.211905, 0.623086},
 {-0.142503, -0.0872588, 0.578129, 0.301075, 0.0663367,
  0.696024, 0.241557}, {0.10397, 0.29804, -0.260627,
  -0.509787, -0.393721, 0.608563, -0.21723}}
```

The set {alignerframe[1],alignerframe[2],. . ., alignerframe[rankA]} spans a subspace S of hitdimD.

The truth is that the null space of A is the perpendicular complement $S^\perp$ of S.

`For a detailed explanation of this, click on the right.`

On way of explaining why the null space of A **is** the perpendicular complement $S^\perp$ of S is to explain the following two facts:

$\boxed{\text{Fact 1}}$ **Any X in $S^\perp$ is also in the null space of A.**

`For a detailed explanation of Fact 1, click on the right.`

Go with an X in $S^\perp$ and look at

$$A.X = \sum_{k=1}^{rankA} \text{stretch}[k]\,(X.\text{alignerframe}[k])\,\text{hangerframe}[k] \qquad (\text{ with stretch}[k] > 0)$$

$$= \sum_{k=1}^{rankA} \text{stretch}[k]\,(0)\,\text{hangerframe}[k]$$

`Reason: X is perpendicular to everything in S.`

$$= \sum_{k=1}^{rankA} \{0, 0, \ldots, 0\}$$

$$= \{0, 0, \ldots, 0\}.$$

This explains why any X in $S^\perp$ is also in the null space of A.

$\boxed{\text{Fact 2}}$ **Any X in the null space of A is also in $S^\perp$.**

`For a detailed explanation of Fact 2, click on the right.`

Go with an X in the null space of A.

This gives:

$$\{0,0,\ldots,0\} = A.X = \sum_{k=1}^{rankA} \text{stretch}[k]\,(X.\text{alignerframe}[k])\,\text{hangerframe}[k].$$

So

$$\sum_{k=1}^{rankA} \text{stretch}[k]\,(X.\text{alignerframe}[k])\,\text{hangerframe}[k] = \{0,0,\ldots,0\}$$

Take a p with $1 \le p \le rankA$ and dot both sides with hangerframe[p] to get

$$\sum_{k=1}^{rankA} \text{stretch}[k]\,(X.\text{alignerframe}[k])\,\textbf{hangerframe}[\mathbf{p}].\text{hangerframe}[k] =$$
$$\textbf{hangerframe}[\mathbf{p}].\{0,0,\ldots,0\} = 0.$$

This gives:

$$\text{stretch}[p]\,(X.\text{alignerframe}[p]) = 0.$$

```
Reason: hangerframe[p].hangerframe[k] = 0 for k ≠ p.
        hangerframe[p].hangerframe[k] = 1 for k = k.
```

The result:

$$(X.\text{alignerframe}[p]) = 0.$$

`Reason: stretch[p] >0`

And because p is any integer with $1 \le p \le rankA$, this tells you that X is perpendicular to each of the spanning vectors

{alignerframe[1],alignerframe[2],. . .,alignerframe[rankA]} of S.

Consequently X is perpendicular to everything in S.

This signals that X is in $S^\perp$.

This explains why any X in the null space of A is also in $S^\perp$

To go after an orthonormal basis for S (perpendicular frame spanning S) remember that $S^\perp$ consists of all hits with Identity - Sprojection:

```
SpannerMatrix = Transpose[Sspanners];
SProjection = SpannerMatrix.PseudoInverse[SpannerMatrix];
B = IdentityMatrix[hitdim] - SProjection;
MatrixForm[B]
```

```
( 0.414214     0.421386    0.00746086   0.162393    0.122269    -0.0038707
  0.421386     0.440431    0.0646123    0.0884399   0.129878    -0.0313817
  0.00746086   0.0646123   0.277503    -0.363616    0.00781753  -0.137921
  0.162393     0.0884399  -0.363616     0.625751   -0.198321     0.131316
  0.122269     0.129878    0.00781753  -0.198321    0.770331     0.147711
 -0.00387077  -0.0313817  -0.137921     0.131316    0.147711     0.0998718
  0.153891     0.197575    0.212049    -0.105838   -0.289642    -0.175536
```

So you can get an orthonormal basis for the null space of A by going with the SVD hangerframe of

B = Identity - Sprojection.

Here you go:

```
rankB = Length[SingularValues[B][[2]]];
Clear[hangerframe];
hangerframe[k_] := SingularValues[B][[1, k]];
nullspacespanners = Table[hangerframe[k], {k, 1, rankB}]
```

```
{{0, -0.0731689, -0.373279, 0.296644, 0.596872, 0.310372, -0.561003},
 {-0.643594, -0.654739, -0.0115925, -0.252323,
  -0.189978, 0.00601431, -0.239112}, {0, 0.0799671,
  0.371526, -0.688539, 0.614803, -0.0592035, -0.000358305}}
```

There you go.

Try it out:

```
Clear[x, j];
```

$$\textbf{Expand}\Big[\mathbf{A}.\sum_{k=1}^{rankB} \mathbf{x}[\mathbf{k}]\ \textbf{hangerframe}[\mathbf{k}]\Big]$$

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

T.5) The sum S1 + S2 of two subspaces of kD is a subspace of kD.

The intersection S1 ⋂ S2 of two subspaces of kD is a subspace of kD.

How to find orthonormal bases for them.

No matter what Y in kD you go with, you are guaranteed that

||Sprojection.Y|| ≤ ||Y||.

Saying ||Sprojection.Y|| = ||Y|| for a given Y in kD is the same as saying Y is in S.

Saying ||Sprojection.Y|| < ||Y|| for a given Y in kD is the same as saying Y is not in S.

Saying that a given Y in kD is in S1 ⋂ S2 is the same as saying

||S2Projection.S1Projection.Y|| = ||Y||

T.5.a.i) Adding two subspaces of kD

Given two subspaces S1 and S2 of kD, most folks calculate

S1 + S2

by taking everything in S1 and adding this to everything in S2.

Confirm that S1 + S2 is a subspace of kD by explaining how to get a spanning set for

S1 + S2

from a given spanning sets S1 and a given spanning set for S2.

Answer:

To do this you've got to start with a spanning sets for S1 and S2 and exhibit a spanning set for S1 + S2.

The subspace S1 is specified by a spanning set

spanners1 = {spanner1[1], spanner1[2], . . ., spanner1[p]}.

The members of S1 look like this:

$$\sum_{i=1}^{p} x[i]\ \text{spanner1}[i]$$

where the x[i]'s are real numbers.

The subspace S1 is specified by a spanning set

spanners2 = {spanner2[1], spanner2[2], . . ., spanner2[q]}.

The members of S2 look like this:

$$\sum_{j=1}^{q} y[j]\ \text{spanner2}[j]$$

where the y[j]'s are real numbers.

You get S1 + S2 by taking everything in S1 and adding this to everything in S2.

In other words S1 + S2 is everything of this form:

$$\sum_{i=1}^{p} x[i]\ \text{spanner1}[i] + \sum_{j=1}^{q} y[j]\ \text{spanner2}[j].$$

This tells you that a spanning set for S1 + S2 is the spanning set you get by throwing spanners1 and spanners2 into one big spanning set.

In fancier terms, a spanning set for S1 + S2 is:

spanners1 ⋃ spanners2

□T.5.a.ii) Calculating the dimension of S1 + S2 and getting an orthonormal basis for S1 + S2

A subspace S1 of 8D is specified by the following spanning set:

```
Clear[i, spanner1];
spanner1[1] = {0.0, -0.3, -0.2, 0.2, 0.4, 0.4, 0.5, 0.9};
spanner1[2] = {0.9, 0.8, 0.7, 0.4, 0.0, -0.1, 0.0, -0.7};
spanner1[3] = {-0.5, -0.3, 0.9, 0.4, -0.3, 0.2, -0.5, 0.4};
spanner1[4] = {1.4, 0.8, -0.4, 0.2, 0.7, 0.1, 1., -0.2};
spanners1 = Table[spanner1[i], {i, 1, 4}]
```

```
{{0, -0.3, -0.2, 0.2, 0.4, 0.4, 0.5, 0.9},
 {0.9, 0.8, 0.7, 0.4, 0, -0.1, 0, -0.7},
 {-0.5, -0.3, 0.9, 0.4, -0.3, 0.2, -0.5, 0.4},
 {1.4, 0.8, -0.4, 0.2, 0.7, 0.1, 1., -0.2}}
```

Another subspace S2 of 8D is specified by the following spanning set:

```
Clear[j, spanner2];
spanner2[1] = {0.2, -0.5, -0.3, 0.4, 0.4, -0.8, 0, 0};
spanner2[2] = {0.4, 0.2, 1.4, 1., 0.1, 0.5, 0, 0.6};
spanners2 = Table[spanner2[j], {j, 1, 2}]
```

```
{{0.2, -0.5, -0.3, 0.4, 0.4, -0.8, 0, 0},
 {0.4, 0.2, 1.4, 1., 0.1, 0.5, 0, 0.6}}
```

Calculate:
- the dimension of S1,
- the dimension of S2 and
- the dimension of S1 + S2.

Exhibit an orthonormal basis of S1 + S2

□Answer:

The dimension of S1 is:

```
SpannerMatrix1 = Transpose[spanners1];
rank = Length[SingularValues[SpannerMatrix1][[2]]];
S1dim = rank
```

```
3
```

The dimension of S2 is:

```
SpannerMatrix2 = Transpose[spanners2];
rank = Length[SingularValues[SpannerMatrix2][[2]]];
S2dim = rank
```

```
2
```

To calculate the dimension of S1 + S2, you have to come up with a spanning set. According to part i) immediately above, you just take spanners1 and throw in spanners2 to get

spanners1 ⋃ spanners2

as a spanning set for S1 + S2:

```
S1plusS2spanners = spanners1 ⋃ spanners2
```

```
{{-0.5, -0.3, 0.9, 0.4, -0.3, 0.2, -0.5, 0.4},
 {0, -0.3, -0.2, 0.2, 0.4, 0.4, 0.5, 0.9},
 {0.2, -0.5, -0.3, 0.4, 0.4, -0.8, 0, 0},
 {0.4, 0.2, 1.4, 1., 0.1, 0.5, 0, 0.6},
 {0.9, 0.8, 0.7, 0.4, 0, -0.1, 0, -0.7},
 {1.4, 0.8, -0.4, 0.2, 0.7, 0.1, 1., -0.2}}
```

Now you can make the spanner matrix for S1 + S2 and use it to calculate the dimension of S1 + S2:

```
SpannerMatrix12 = Transpose[S1plusS2spanners ];
rank = Length[SingularValues[SpannerMatrix12][[2]]];
S1plusS2dim = rank
```

```
4
```

The dimension of S1 + S2 is 4.

An orthonormal basis for S1 + S2 comes from the hanger frame that SpannerMatrix12 uses to hang its hits:

```
hangerframe = SingularValues[SpannerMatrix12][[1]]
```

```
{{-0.710266, -0.449488, 0.0262666, -0.159291,
  -0.276357, -0.00968883, -0.386125, 0.204589},
 {-0.0541454, -0.0857098, -0.799162, -0.448589,
  0.0848737, -0.254626, 0.151882, -0.233934},
 {0.00559216, 0.299563, 0.228723, -0.164433, -0.349389,
  -0.232407, -0.39408, -0.706749}, {0.142732, -0.409137,
  0.0187944, 0.457823, 0.244355, -0.714688, -0.16556, -0.0661886}}
```

There you go.

□T.5.b.i) Using the S1 Projection and the S2 Projection matrices to come up with an orthonormal basis of S1⋂ S2 and to get the S1⋂ S2 Projection matrix

A subspace S1 of 7D is specified by the following spanning set:

```
Clear[i, spanner1];
spanner1[1] = {0.0, -0.3, -0.2, 0.2, 0.4, 0.4, 0.5};
spanner1[2] = {0.9, 0.8, 0.7, 0.4, 0.0, -0.1, 0.0};
spanner1[3] = {-0.5, -0.3, 0.9, 0.4, -0.3, 0.2, -0.5};
spanner1[4] = {1.4, 0.8, -0.4, 0.2, 0.7, 0.1, 1.};
spanners1 = Table[spanner1[i], {i, 1, 4}]
```

```
{{0, -0.3, -0.2, 0.2, 0.4, 0.4, 0.5}, {0.9, 0.8, 0.7, 0.4, 0, -0.1, 0},
 {-0.5, -0.3, 0.9, 0.4, -0.3, 0.2, -0.5},
 {1.4, 0.8, -0.4, 0.2, 0.7, 0.1, 1.}}
```

Another subspace S2 of 7D is specified by the following spanning set:

```
Clear[j, spanner2];
spanner2[1] = {0.2, -0.5, -0.3, 0.4, 0.4, -0.8, 0};
spanner2[2] = {0.4, 0.2, 1.4, 1., 0.1, 0.5, 0};
spanner2[3] = {-1.4, -1.1, 0.2, 0, -0.3, 0.3, -0.5};
spanners2 = Table[spanner2[j], {j, 1, 3}]
```

```
{{0.2, -0.5, -0.3, 0.4, 0.4, -0.8, 0}, {0.4, 0.2, 1.4, 1., 0.1, 0.5, 0},
 {-1.4, -1.1, 0.2, 0, -0.3, 0.3, -0.5}}
```

Come up with an orthonormal basis for S1 ⋂ S2 and use the result to calculate the dimension of S1 ⋂ S2.
Come up with the S1⋂ S2Projection matrix.

□Answer:

Calculate the S1projection matrix:

```
SpannerMatrix1 = Transpose[spanners1];
S1projection = SpannerMatrix1.PseudoInverse[SpannerMatrix1];
MatrixForm[S1projection]
```

```
⎛ 0.557602    0.426489    0.0272519   0.0768324    0.124662   -0.0715796
⎜ 0.426489    0.449198    0.0724158  -0.0415587   -0.0659013  -0.227337
⎜0.0272519   0.0724158    0.690413    0.36382     -0.121633    0.112763
⎜0.0768324  -0.0415587    0.36382     0.3347       0.112492    0.244563
⎜ 0.124662  -0.0659013   -0.121633    0.112492     0.244693    0.193282
⎜-0.0715796  -0.227337    0.112763    0.244563     0.193282    0.30468
⎝ 0.193595  -0.0503005   -0.218817    0.101069     0.316429    0.212508
```

Calculate the S2projection matrix:

```
SpannerMatrix2 = Transpose[spanners2];
S2projection = SpannerMatrix2.PseudoInverse[SpannerMatrix2];
MatrixForm[S2projection]
```

```
⎛ 0.580998   0.357118   0.0139449    0.14584     0.174123   -0.173197
⎜ 0.357118   0.557586   0.0241616   -0.192869   -0.072767    0.22391
⎜0.0139449  0.0241616   0.618917     0.372619   -0.0237549   0.305724  -
⎜ 0.14584   -0.192869   0.372619     0.509043    0.181802   -0.137306  -
⎜ 0.174123  -0.072767  -0.0237549    0.181802    0.148426   -0.236968
⎜-0.173197   0.22391    0.305724    -0.137306   -0.236968    0.514974  -
⎝ 0.184732   0.161052  -0.0469475   -0.0264589   0.0318039  -0.0341964
```

Multiply out S1projection.S2projection

```
product = S2projection.S1projection;
MatrixForm[product]
```

```
⎛ 0.557725    0.421762    0.0232497    0.079585     0.131189   -0.065391
⎜ 0.428854    0.35833    -0.00451917   0.0113547    0.059572   -0.108385
⎜0.0296426  -0.0194408    0.612641     0.417309    0.00520443   0.233009
⎜0.0756999   0.00195599   0.400662     0.309361     0.0524059    0.1876
⎜  0.121     0.0747895   -0.00251562   0.0305667    0.0504235   0.0091096
⎜-0.0763205  -0.0451771   0.266991     0.13849     -0.058249    0.0662222
⎝ 0.188357    0.150985   -0.0483966   -0.0161409    0.0384902  -0.050985
```

This is not necessarily the projection matrix for S1 ⋂ S2, but you can extract the projection matrix for S1 ⋂ S2 from S2projection.S1projection.

Here's how:

Look at the stretch factors of S2projection.S1projection:

```
stretches = SingularValues[product][[2]]
```

```
{1., 1., 0.14864}
```

Two of them equal 1. This is your signal that the dimension of S1 ⋂ S2 is 2.

To get an orthonormal basis for S1 ⋂ S2, you go with the first two alignerframe vectors for

product = S2projection.S1projection:

```
Clear[alignerframe];
alignerframe[k_] :=
 SingularValues[S2projection.S1projection][[3, k]] ;
orthonormal = Table[alignerframe[k], {k, 1, 2}]
```

```
{{-0.372009, -0.339533, 0.655455, 0.413351,
  -0.0738112, 0.321822, -0.19191}, {-0.647405, -0.468274,
  -0.422631, -0.353512, -0.14378, -0.0683272, -0.180714}}
```

The projection matrix for S1⋂ S2 is:

```
aligner = {alignerframe[1], alignerframe[2]};
hanger = Transpose[ aligner];
S1intersectS2Projection = hanger.aligner;
MatrixForm[S1intersectS2Projection]
```

```
  0.557524    0.429472    0.029778    0.0750951   0.120542   -0.0754852
  0.429472    0.334563   -0.0246416   0.0251942   0.0923897  -0.0772732
  0.029778   -0.0246416   0.608238    0.420337    0.0123858   0.239817
 0.0750951    0.0251942   0.420337    0.295829    0.0203179   0.15718
  0.120542    0.0923897   0.0123858   0.0203179   0.0261207  -0.01393
-0.0754852   -0.0772732   0.239817    0.15718    -0.01393     0.108238
  0.188388    0.149784   -0.0494132  -0.0154416   0.0401482  -0.0494132
```

□T.5.b.ii) Why did that work?

Thanks go to Timothy "Timbo" Braun for helpful pointers on this.

In the last problem,you calculated an SVD aligner frame for the product S2Projection.S1Projection.
And then you got your orthonormal basis for S1⋂ S2 by taking all the aligner frame vectors with corresponding SVD stretch factors equal to 1.
Why did that do the job?
Why will the same thing work in other situations?

□Answer:

Go with subspaces S1 and S2 of kD and a given kD vector Y.

It so happens that saying that Y is in S1⋂ S2 is the same as saying that

||S2Projection.S1Projection.Y|| = ||Y||.

For an explanation of this delicious fact, go to the next part.

That's why you go with the aligner frame vectors for the product S2Projection.S1Projection that have corresponding stretch factors equal to 1.

□T.5.b.iii) Explanation of the logical sequence that says for subspaces S. S1 and S2 of kD:

$\boxed{1}$ No matter what Y in kD you go with, you are guaranteed that ||Sprojection.Y|| ≤ ||Y||.

$\boxed{2}$ Saying ||Sprojection.Y|| = ||Y|| for a given Y in kD is the same as saying Y is in S.

$\boxed{3}$ Saying ||Sprojection.Y|| < ||Y|| for a given Y in kD is the same as saying Y is not in S.

↦ $\boxed{4}$ Saying that a given Y in kD is in S1⋂ S2 is the same as saying

||S2Projection.S1Projection.Y|| = ||Y||

Here is the logical sequence for an explanation of why when you are given two subspaces S1 and S2 of kD, then saying that a given Y in kD is in S1⋂ S2 is the same as saying
||S2Projection.S1Projection.Y|| = ||Y||.

For subspaces S, S1 and S2 of kD:

$\boxed{1}$ No matter what Y in kD you go with, you are guaranteed that ||Sprojection.Y|| ≤ ||Y||.

Click on the right for a detailed explanation

Explanation:

Take Y break it into Y = Y1 + Y2 with Y1 in S and Y2 perpendicular to S. (So that SProjectionY = Y1).

Now note that

$$\|\text{SProjection}Y\| = \|Y1\| \leq \sqrt{\|Y1\|^2 + \|Y2\|^2} = \|Y\|.$$

$\boxed{2}$ Saying ||Sprojection.Y|| = ||Y|| for a given Y in kD is the same as saying Y is in S.

Click on the right for a detailed explanation

Explanation:

Take Y break it into Y = Y1 + Y2 with Y1 in S and Y2 perpendicular to S. (So that SProjectionY = Y1).

It's automatic that ||SProjectionY|| = ||Y1||.

$$\|\text{SProjection}Y\| = \|Y1\| = \|Y\| = \sqrt{\|Y1\|^2 + \|Y2\|^2} \iff Y2 = \{0,0,\ldots,0\} \iff Y \text{ is in } S.$$

Read ⟺ as "is the same as"

$\boxed{3}$ Saying ||Sprojection.Y|| < ||Y|| for a given Y in kD is the same as saying Y is not in S.

Click on the right for a detailed explanation

According to $\boxed{1}$, you know that ||Sprojection.Y|| ≤ ||Y||.

$\boxed{2}$ tells you that saying ||Sprojection.Y|| = ||Y|| for a given Y in kD is the same as saying Y is in S.

So saying ||Sprojection.Y|| < ||Y|| is the same as saying Y is not in S.

↦ $\boxed{4}$ Given two subspaces S1 and S2 of kD, then saying that a given Y in kD is in S1⋂ S2 is the same as saying
||S2Projection.S1Projection.Y|| = ||Y||.

Click on the right for a detailed explanation

First half of the explanation:

Assume Y is in S1 ⋂ S2 and explain why ||S2Projection.S1Projection.Y|| = ||Y||.

The assumption that Y is in S1⋂ S2, guarantees that S1Projection.Y = Y and S2Projection.Y = Y.

So if Y is in both S1⋂ S2, then

S2Projection.S1Projection.Y = S2Projection.Y = Y.

And so

||S2Projection.S1Projection.Y|| = ||Y||.

Second half of the explanation::

Assume ||S2Projection.S1Projection.Y|| = ||Y|| and explain why Y is in S1⋂ S2.

According to $\boxed{3}$, if Y is not in S1, then ||S1projection.Y|| < ||Y||.

But according to $\boxed{1}$

||Y|| = ||S2Projection.S1Projection.Y|| ≤ ||S1Projection.Y|| < || Y ||.

The upshot:

The assumption that Y is not in S1 leads to the contradiction ||Y|| < ||Y||.

This explains why Y is in S1.

The fact that Y is in S1 guarantees that S1Projection.Y = Y.

So according to $\boxed{1}$,

||Y|| = ||S2Projection.S1Projection.Y|| = ||S2Projection.Y||.

This gives

||S2Projection.Y|| = ||Y||.

Now $\boxed{2}$ steps in to tell you that Y is also in S2.

The conclusion:

Saying **||S2Projection.S1Projection.Y|| = ||Y||** is the same as saying **Y is in S1⋂ S2.**

There you go.

T.6) Getting an orthonormal basis from a given basis via the Gram-Schmidt process

□T.6.a.i) Using the GramSchmidt instruction

A subspace S of 6D is defined by the following given spanning set:

```
Clear[i, spanner];
spanner[1] = {0.43, -0.57, 2.19, -0.94, 1.28};
spanner[2] = {0.97, -3.29, -0.83, 5.23, 1.26};
spanner[3] = {-6.14, 3.08, 0.00, -1.97, -0.96};
spanner[4] = {-3.68, 1.22, -3.64, -2.14, -0.94};
spanners = Table[spanner[i], {i, 1, 4}]
```

```
{{0.43, -0.57, 2.19, -0.94, 1.28}, {0.97, -3.29, -0.83, 5.23, 1.26},
 {-6.14, 3.08, 0, -1.97, -0.96}, {-3.68, 1.22, -3.64, -2.14, -0.94}}
```

Check the dimension of S:

```
SpannerMatrix = Transpose[spanners];

Sdim = Length[SingularValues[SpannerMatrix][[2]]]
```

```
4
```

This tells you that the dimension of S is the same as the number of vectors in the spanning set.
And this signals that the given spanning set for S is also a basis (linearly independent spanning set) for S.
Old matrix hands know that you can get an orthonormal basis (perpendicular frame spanning S) for S by applying what lots of folks like to call the "Gram-Schmidt Orthogonalization Process:"

```
Needs["LinearAlgebra`Orthogonalization`"];
orthospanners = GramSchmidt[spanners]
```

```
{{0.15369, -0.203729, 0.782747, -0.335974, 0.457496},
 {0.177117, -0.550208, -0.00607795, 0.769655, 0.271099},
 {-0.94543, 0.0548018, 0.18959, 0.200235, 0.16468},
 {-0.169181, -0.51427, -0.550278, -0.479896, 0.416892}}
```

What are these vectors?

□Answer:

Take another look:

```
orthospanners = GramSchmidt[spanners]
```

```
{{0.15369, -0.203729, 0.782747, -0.335974, 0.457496},
 {0.177117, -0.550208, -0.00607795, 0.769655, 0.271099},
 {-0.94543, 0.0548018, 0.18959, 0.200235, 0.16468},
 {-0.169181, -0.51427, -0.550278, -0.479896, 0.416892}}
```

These four vectors give you an orthonormal basis for the subspace S spanned by the original basis. They were produced through a process widely known as the Gram-Schmidt process.

If you want to know what the Gram-Schmidt process is, go on.

□T.6.a.ii) How the Gram-Schmidt process works

How does the Gram-Schmidt process take in a basis (linearly independent spanning set)
{spanner[1], spanner[2], spanner[3], . . ., spanner[k]}
for a subspace S of kD and spit out an orthonormal basis (perpendicular frame) for the same subspace S?

□Answer:

Here's how it goes:

Go with the given basis

{spanner[1], spanner[2], spanner[3], . . ., spanner[k]}

for a subspace S of nD.

□Gram-Schmidt Step1:

Put Y[1] = spanner[1].

and put

$$X[1] = \frac{Y[1]}{\sqrt{Y[1].Y[1]}}.$$

□Gram-Schmidt Step2:

Put

Y[2] = spanner[2] − (component of spanner[2] in the direction of X[1]).

Put

$$X[2] = \frac{Y[2]}{\sqrt{Y[2].Y[2]}}\ .$$

Note:

X[1] and X[2] are unit vectors,

X[1].X[2] = 0 ,

and

{X[1], X[2]} spans the subspace spanned by {spanner[1], spanner[2]}

□Gram-Schmidt Step3:

Put

Y[3] = spanner[3] .
− (component of spanner[3] in the direction of X[1])
− (component of spanner[3] in the direction of X[2]).

and put

$$X[3] = \frac{Y[3]}{\sqrt{Y[3].Y[3]}}.$$

Note:Note:

X[1], X[2] and X[3] are all unit vectors,

X[1].X[2] = 0

X[1].X[3] = 0

X[2].X[3] = 0

and

{X[1], X[2], X[3]} spans the subspace spanned by {spanner[1], spanner[2], spanner[3]}.

□Gram-Schmidt Step j:

Put

$$Y[j] = spanner[j] - \sum_{i=1}^{j-1} \text{component of spanner[j] in the direction of X[i].}$$

$$X[j] = \frac{Y[j]}{\sqrt{Y[j].Y[j]}}\ .$$

Note

X[1], X[2],. . . , X[j] are all unit vectors,

X[p].X[i] = 0 for i,p = 1,2,...,j-1 with i ≠ p.

and

{X[1], X[2], X[3], . . ., X[j]}

spans the subspace spanned by

{spanner[1], spanner[2], spanner[3], ..., spanner[j]}

Starting with

given spanning set

{spanner[1], spanner[2], spanner[3], . . ., spanner[k]},

the process stops when j = k and produces vectors

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

with the guarantees that:

1) Each X[i] is a unit vector .

2) And for each stage j,

{X[1], X[2], ..., X[j]}

spans the same subspace as

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

This tells you that

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

spans the same subspace as

{spanner[1], spanner[2], spanner[3], . . ., spanner[k]}

So {X[1], X[2], . . ., X[k]} is guaranteed to be an orthonormal basis of S (a perpendicular frame that frames up S).

□T.6.a.iii) Trying it out

```
This is the way students in book courses generate an orthonormal basis from
                          a non-orthonormal basis.
```

Here is the linearly independent spanning set from part i)

```
Clear[i, spanner];
spanner[1] = {0.43, -0.57, 2.19, -0.94, 1.28};
spanner[2] = {0.97, -3.29, -0.83, 5.23, 1.26};
spanner[3] = {-6.14, 3.08, 0.00, -1.97, -0.96};
spanner[4] = {-3.68, 1.22, -3.64, -2.14, -0.94};
spanners = Table[spanner[i], {i, 1, 4}]
```

```
{{0.43, -0.57, 2.19, -0.94, 1.28}, {0.97, -3.29, -0.83, 5.23, 1.26},
 {-6.14, 3.08, 0, -1.97, -0.96}, {-3.68, 1.22, -3.64, -2.14, -0.94}}
```

An orthonormal basis for the same subspace is:

```
orthospanners = GramSchmidt[ spanners]
```

```
{{0.15369, -0.203729, 0.782747, -0.335974, 0.457496},
 {0.177117, -0.550208, -0.00607795, 0.769655, 0.271099},
 {-0.94543, 0.0548018, 0.18959, 0.200235, 0.16468},
 {-0.169181, -0.51427, -0.550278, -0.479896, 0.416892}}
```

Illustrate how the Gram-Schmidt process produces this orthonormal basis for S:

□Answer:

Copy, paste and edit:

Go with the given basis

{spanner[1], spanner[2], spanner[3], sapnner[4]}

for a subspace S of 6D.

□Gram-Schmidt Step1:

Put Y[1] = spanner[1].

and put

$$X[1] = \frac{Y[1]}{\sqrt{Y[1].Y[1]}}:$$

```
Clear[X, Y];
Y[1] = spanner[1];
X[1] = Y[1] / Sqrt[Y[1].Y[1]]
```

```
{0.15369, -0.203729, 0.782747, -0.335974, 0.457496}
```

□ Gram-Schmidt Step2:

Put

Y[2] = spanner[2] – (component of spanner[2] in the direction of X[1]).

Put

$$X[2] = \frac{Y[2]}{\sqrt{Y[2].Y[2]}}$$

```
Y[2] = spanner[2] - (spanner[2].X[1]) X[1];
            Y[2]
X[2] = ──────────────
        √Y[2].Y[2]
```

```
{0.177117, -0.550208, -0.00607795, 0.769655, 0.271099}
```

□ Gram-Schmidt Step3:

Put

Y[3] = spanner[3] .
– (component of spanner[3] in the direction of X[1])
– (component of spanner[3] in the direction of X[2]).

and put

$$X[3] = \frac{Y[3]}{\sqrt{Y[3].Y[3]}}.$$

```
Y[3] = spanner[3] - (spanner[3].X[1] ) X[1] - (spanner[3].X[2]) X[2];
            Y[3]
X[3] = ──────────────
        √Y[3].Y[3]
```

```
{-0.94543, 0.0548018, 0.18959, 0.200235, 0.16468}
```

□ Gram-Schmidt Step 4:

Put

Y[4] = spanner[4] .
– (component of spanner[4] in the direction of X[1])
– (component of spanner[4] in the direction of X[2]).
– (component of spanner[4] in the direction of X[3]).

and put

$$X[4] = \frac{Y[4]}{\sqrt{Y[4].Y[4]}}.$$

```
Clear[i];
                    3
Y[4] = spanner[4] - ∑ (spanner[4].X[i]) X[i];
                   i=1
            Y[4]
X[4] = ──────────────
        √Y[4].Y[4]
```

```
{-0.169181, -0.51427, -0.550278, -0.479896, 0.416892}
```

Compare:

```
{X[1], X[2], X[3], X[4]}
```

```
{{0.15369, -0.203729, 0.782747, -0.335974, 0.457496},
 {0.177117, -0.550208, -0.00607795, 0.769655, 0.271099},
 {-0.94543, 0.0548018, 0.18959, 0.200235, 0.16468},
 {-0.169181, -0.51427, -0.550278, -0.479896, 0.416892}}
```

```
orthospanners = GramSchmidt[ spanners]
```

```
{{0.15369, -0.203729, 0.782747, -0.335974, 0.457496},
 {0.177117, -0.550208, -0.00607795, 0.769655, 0.271099},
 {-0.94543, 0.0548018, 0.18959, 0.200235, 0.16468},
 {-0.169181, -0.51427, -0.550278, -0.479896, 0.416892}}
```

It will work this way every time.

□ T.6.b.i) SVD versus Gram-Schmidt: They both give you an orthonormal basis.

Usually they don't give the same basis

Given a linearly independent spanning set for a subspace S of nD,
you have your choice of an orthonormal basis S via
1) By going with the GramSchmidt process (as above)
or
2) By going with the SVD hanger frame for the associated SpannerMatrix (as in the Basics)
Does it matter which you use?.

□ Answer:

Try out the two on this spanning set for a subspace S of 6D:

```
Clear[i, spanner];
spanner[1] = {0.43, -0.57, 2.19, -0.94, 1.28, 1.00};
spanner[2] = {0.97, -3.29, -0.83, 5.23, 1.26, 0.98};
spanner[3] = {-6.14, 3.08, 0.00, -1.97, -0.96, 0.75};
spanner[4] = {-3.68, 1.22, -3.64, -2.14, -0.94, -1.26};
spanner[5] = {1.23, 1.78, 6.90, -3.77, -0.86, 2.53};
spanners = Table[spanner[i], {i, 1, 5}]
```

```
{{0.43, -0.57, 2.19, -0.94, 1.28, 1.},
 {0.97, -3.29, -0.83, 5.23, 1.26, 0.98},
 {-6.14, 3.08, 0, -1.97, -0.96, 0.75},
 {-3.68, 1.22, -3.64, -2.14, -0.94, -1.26},
 {1.23, 1.78, 6.9, -3.77, -0.86, 2.53}}
```

```
SpannerMatrix = Transpose[spanners];
SVDorthonormal = SingularValues[SpannerMatrix ][[1]]
```

```
{{-0.309737, 0.451242, 0.462024, -0.663247, -0.168356, 0.136884},
 {-0.647711, 0.19277, -0.687577, -0.0662828, -0.10748, -0.23367},
 {0.614371, 0.0788091, -0.327883, -0.525292, -0.127839, -0.465356},
 {-0.170243, -0.48737, -0.0660962, -0.464989, 0.712909, 0.068307},
 {-0.0318953, -0.587729, -0.155685, -0.247929, -0.625779, 0.419823}}
```

```
GramSchmidtorthonormal = GramSchmidt[ spanners]
```

```
{{0.144724, -0.191843, 0.737081, -0.316373, 0.430806, 0.336567},
 {0.163632, -0.526279, -0.0573272, 0.776931, 0.235874, 0.183602},
 {-0.91543, 0.112447, 0.0875209, 0.173846, 0.08286, 0.323413},
 {-0.192827, -0.525872, -0.517754, -0.467223, 0.439113, -0.0842036},
 {-0.00194202, -0.476431, -0.00615876,
  -0.213709, -0.717747, 0.460586}}
```

The results are different but each gives a fully legitimate orthonormal basis for S.

If you are away from the computer, you might prefer Gram-Schmidt because, in principle, it can be done by hand.

□ T.6.b.ii) Down side of Gram Schmidt

What's the down side of Gram-Schmidt?

□ Answer:

The down side of Gram-Schmidt is that before you use it, you have to know that the given spanning set is linearly independent (is a basis). There is no such problem when you go with SVD.