Excursions in Computing Science:
Week 7c Coordinates, Angles and Reality

T. H. Merrett*
McGill University, Montreal, Canada

April 25, 2007

A. Reality

Gonna jump down, spin around, pick a bale of cotton.
Gonna jump down, spin around, pick a bale a day.
Norman Luboff, Harry Belafonte and William Attaway

1. Vectors are real.

- Independent of coordinate axes, so
- transform in a certain way when we change the axes.

Example transformations:

$$\begin{aligned} \text{rotate} \begin{pmatrix} x' \\ y' \end{pmatrix} &= \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\ \text{reflect } x \begin{pmatrix} x' \\ y' \end{pmatrix} &= \begin{pmatrix} -1 & \\ & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \end{aligned}$$

So what are *not* vectors?

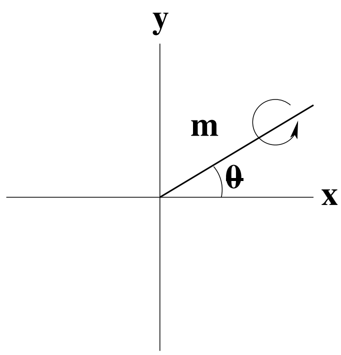
A twirl is not:

it has magnitude m and direction θ,

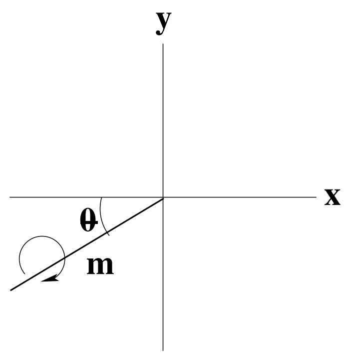
so $x = \cos\theta$ and $y = \sin\theta$

but it does not reflect the way a vector does.

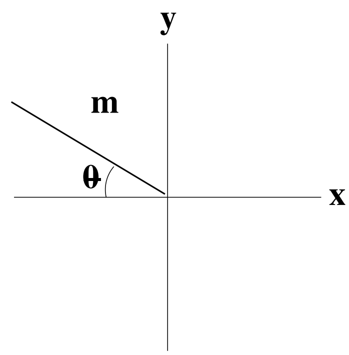
 The author gratefully acknowledges support from the taxpayers of Québec and of Canada who have paid his salary and research grants while this work was developed at McGill University, and from his students (who built the implementations and investigated the data structures and algorithms) and their funding agencies.



Twirl (m, θ)



Reflected x→ −x

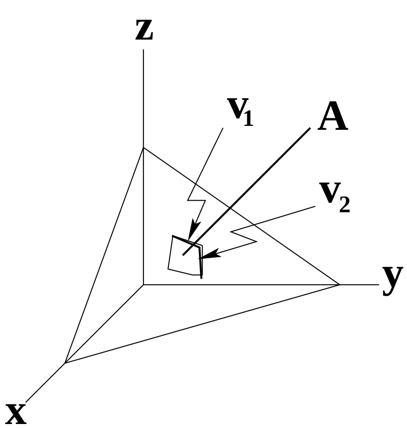


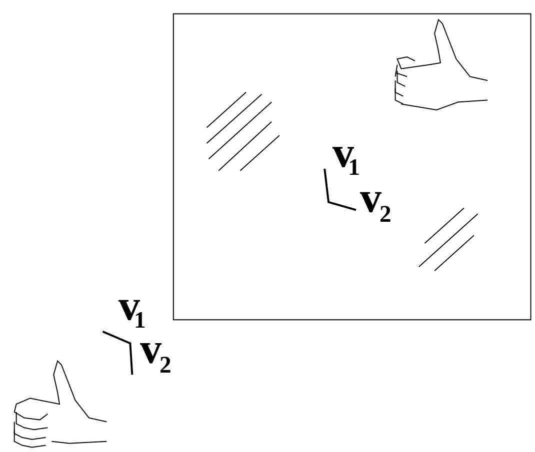
This is not the reflection

We get $m' = -m$, i.e., $x' = -x$ and $y' = -y$

instead of $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -1 & \\ & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

In 3D, an *area* is like a twirl: it can have an *orientation* to distinguish above from below.





We saw that a right-handed twirl becomes a left-handed twirl in the mirror.

Similarly the direction of turn needed to rotate v_1 into v_2 is reversed in the mirror. This direction can be taken to determine the orientation of the parallelopiped area defined by v_1 and v_2.

In some sense, $v_1 v_2 = -v_2 v_1$: the "product" is anticommutative. We'll follow up this essential insight shortly (Note 6).

2. Some pairs are not vectors: their components are not coordinates.

$$\begin{pmatrix} \text{apples}' \\ \text{oranges}' \end{pmatrix} \overset{??}{=} \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} \text{apples} \\ \text{oranges} \end{pmatrix}$$

This is not a totally hokey example. Information retrieval (I.R.) often uses "vectors" to capture the content of documents.

	around	bale	cotton	day	down	jump	pick	spin	
doc1(	1	1	1	0	1	1	1	1	)
doc2(	1	1	0	1	1	1	1	1	)

I.R. even uses dot products to detect similarity between documents:

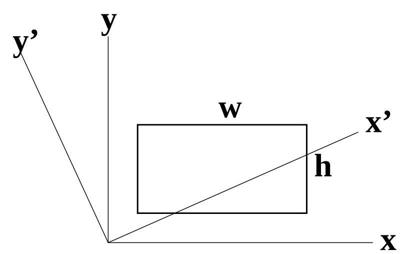
$$(\text{doc1 . doc2})/(|\text{doc1}||\text{doc2}|) = 6/(\sqrt{7}\sqrt{7}).$$

But documents are not vectors: it is not meaningful to rotate or reflect the axes.

3. Even pairs of numbers from geometry, where rotating and reflecting *are* meaningful, are not always vectors. Let's try

$$\begin{pmatrix} \text{height} \\ \text{width} \end{pmatrix}$$

Here, no matter what the axes do, these numbers should not change.



What kind of thing remains invariant no matter what the axes do?

As with a vector, this thing, this pair of numbers, has a reality independent of the choice of coordinate axes. But the components of this one do not change if axes are rotated or reflected.

How about a matrix whose *eigenvalues* are w and h?

$$\begin{aligned} T\vec{v_1} &= w\vec{v_1} \\ T\vec{v_2} &= h\vec{v_2} \end{aligned}$$

For example, given the axes x and y shown,

$$\begin{aligned} T &= \begin{pmatrix} w & \\ & h \end{pmatrix} \\ v_1 &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ v_2 &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{aligned}$$

Then, for axes x' and y', related to x and y by rotation R,

$$\vec{v_1'} = R\vec{v_1} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

and

$$RTR^{-1}\vec{v_1'} = RTR^{-1}R\vec{v_1} = RT\vec{v_1} = Rw\vec{v_1} = wR\vec{v_1} = w\vec{v_1'}$$

This suggests that T transforms to the new axes as $T' = RTR^{-1}$.

$$\begin{aligned} \text{Hence } T'\vec{v_1'} &= w\vec{v_1'} \\ \text{Similarly } T'\vec{v_2'} &= h\vec{v_2'} \end{aligned}$$

This is called a *tensor* transformation. Height and width form a "tensor". This tensor is a diagonal matrix, $\begin{pmatrix} w & \\ & h \end{pmatrix}$, when the axes are aligned with the rectangle, as x and y are.

This tensor is not diagonal for all coordinate axes, but we can see that it is a symmetric matrix.

$$T' = RTR^{-1} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} w & \\ & h \end{pmatrix} \begin{pmatrix} c & -s \\ s & c \end{pmatrix}$$

A symmetric matrix, T, equals its own transpose, $T = T^T$.

In general we may think of a tensor loosely as a matrix describing some real thing, as opposed to an operation or transformation.

$T' = RTR^{-1}$ is symmetric because the inverse of R is the transpose of R, $R^{-1} = R^T$, which is the case for rotations, reflections and other "orthogonal" transformations of coordinate axes.

4. Maybe twirl is a tensor too.

Try $S = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and reflect in y by reversing the direction of x using the reFlection matrix F to give the tensor transformation FSF^{-1}

$$-\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} -1 & \\ & 1 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} -1 & \\ & 1 \end{pmatrix} = \begin{pmatrix} a & -b \\ -c & d \end{pmatrix}$$

(Remember, Note 1 found out that the reflection just changes the sign of the twirl, i.e., of the tensor representing it.)

So $a = 0 = d$.

Any reflection will give a similar sign change, so let's see what reflecting in the line $x = y$ gives us: $F = \begin{pmatrix} & 1 \\ 1 & \end{pmatrix}$

$$-\begin{pmatrix} & b \\ c & \end{pmatrix} = \begin{pmatrix} & 1 \\ 1 & \end{pmatrix} \begin{pmatrix} & b \\ c & \end{pmatrix} \begin{pmatrix} & 1 \\ 1 & \end{pmatrix} = \begin{pmatrix} & c \\ b & \end{pmatrix}$$

and so $c = -b$.

Unfortunately, we've gone too far. We now have only one number, b, to describe a twirl, which we saw in Note 1 requires two numbers, m and θ.

So maybe two dimensions is too small to contain a twirl. This rather makes sense now that we think of it.

Let's see if we can decribe a twirl in three dimensions.

First note that $\begin{pmatrix} & -b \\ b & \end{pmatrix}$ is an *antisymmetric* matrix: it equals the negative of its transpose.

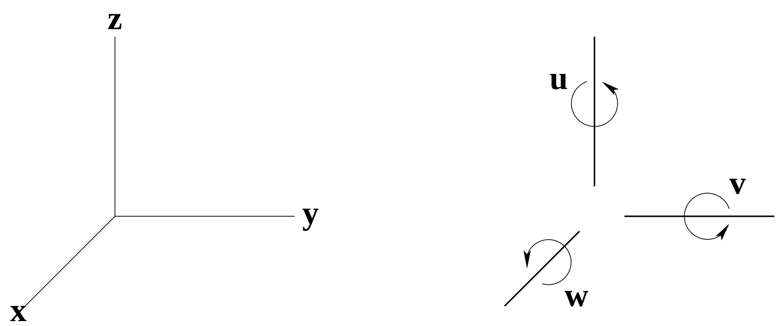
So we'll try an antisymmetric matrix in 3D. A 3×3 antisymmetric matrix has three components.

$$\begin{pmatrix} & u & v \\ -u & & w \\ -v & -w & \end{pmatrix}$$

Try reflecting in the yz plane: $x \leftrightarrow -x$

$$\begin{pmatrix} -1 & & \\ & 1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} & u & v \\ -u & & w \\ -v & -w & \end{pmatrix} \begin{pmatrix} -1 & & \\ & 1 & \\ & & 1 \end{pmatrix} = \begin{pmatrix} & -u & -v \\ u & & w \\ v & -w & \end{pmatrix}$$

This almost just changes the sign of the matrix. Is it right?



Yes, if we interpret w as the x-component of the twirl, v as the y-component and u as the z-component. Check the diagram carefully!

Let's see what happens if we rotate in the xy plane.

$$\begin{pmatrix} c & s & \\ -s & c & \\ & & 1 \end{pmatrix} \begin{pmatrix} & u & v \\ -u & & w \\ -v & -w & \end{pmatrix} \begin{pmatrix} c & -s & \\ s & c & \\ & & 1 \end{pmatrix} = \begin{pmatrix} & u & sw+cv \\ -u & & cw-sv \\ -(sw+cv) & -(cw-sv) & \end{pmatrix}$$

This should be, and is, the same result we would get with $\begin{pmatrix} w \\ v \\ u \end{pmatrix}$ being just a vector, transformed in the usual vector way,

$$\begin{pmatrix} c & -s & \\ s & c & \\ & & 1 \end{pmatrix} \begin{pmatrix} w \\ v \\ u \end{pmatrix}$$

So a twirl, while transforming like a vector under rotation, is in general a tensor; for instance, it does not transform like a vector under reflection.

(Even though "twirl" is in one sense a rotation, we are here looking at it as a "real thing" so the matrix representing it is a tensor—as opposed to the quite different matrix that describes the *operator*, rotation.)

5. Twirl and area are "pseudovectors" or "axial vectors" in Willard Gibbs' vector analysis (which is widely used in spatial science). We now know that they are really tensors. It is just a coincidence that 3×3 antisymmetric tensors have 3 components, like a vector. This does not happen in two dimensions (1 component) or four dimensions (6 components).

Vector analysis generates pseudovectors by a "cross product" of two vectors: $A = v_1 \times v_2 = -v_2 \times v_1$, to use the area example from Note 1.

Vector analysis is unsatisfactory because

a) it is not a closed system: operating on vectors we get things that are not vectors (and, worse, they *look like* vectors);

b) it only works in three dimensions and does not generalize to more, or fewer, dimensions.

Can we make better abstractions for spatial entities, instead of vectors?

We need a formalism

- which is independent of coordinate axes;
- which captures the notion of area being the anticommutative combination of two vectors;
- which does not depend on the number of dimensions of the space.

B. Angle Algebra

6. Vectors and Areas and .. All Together

- Parts of space are lines, areas, volumes, ..
- We'll ignore absolute position and consider only direction and magnitude.
- We'll take the basis elements to be orthonormal and anticommutative.

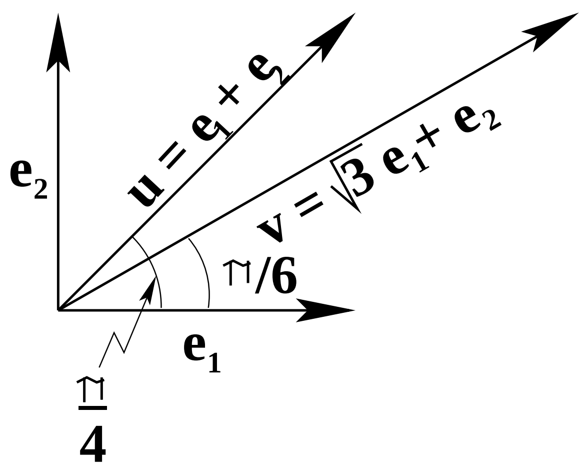
(We'll use the word "elements" instead of "vectors": some but not all elements can be thought of as vectors.)

1. The basis elements are e_1 and e_2, which are defined to have the following properties.

$$\begin{aligned} e_1 e_1 &\stackrel{\text{def}}{=} 1 \\ e_2 e_2 &\stackrel{\text{def}}{=} 1 \\ e_{12} &\stackrel{\text{def}}{=} e_1 e_2 \stackrel{\text{def}}{=} -e_2 e_1 \end{aligned}$$

2. An arbitrary element can be a linear combination of basis elements. Its product with itself is the square of its length or magnitude.

$$\begin{aligned} u &= e_1 + e_2 \\ uu &= (e_1 + e_2)(e_1 + e_2) = 1 + 1 = 2 \\ v &= \sqrt{3} e_1 + e_2 \\ vv &= (\sqrt{3} e_1 + e_2)(\sqrt{3} e_1 + e_2) = 3 + 1 = 2^2 \end{aligned}$$



3. The product of two different elements gives their magnitudes times the cosine and sine of the angle between them.

$$\begin{aligned} uv &= (e_1 + e_2)(\sqrt{3} e_1 + e_2) \\ &= \sqrt{3} + 1 + (1 - \sqrt{3}) e_{12} \\ &= 2\sqrt{2}\left(\frac{\sqrt{3}+1}{2\sqrt{2}} + \frac{1-\sqrt{3}}{2\sqrt{2}} e_{12}\right) \\ &= 2\sqrt{2}(\cos(\pi/6 - \pi/4) + \sin(\pi/6 - \pi/4) e_{12}) \\ (ce_1 + se_2)(c'e_1 + s'e_2) &= (cc' + ss') + (cs' - c's) e_{12} \\ &= \cos(-) + \sin(-) e_{12} \end{aligned}$$

where cos(−) and sin(−) are respectively the cosine and sine of the angle between the two unit elements in this second example.

7. Rotation

Let's have a magnitude operator,

$$| v |= \sqrt{vv} = \text{length of } v$$

and a normalizing operator,

$${}^{n}v = v/ \mid v \mid:\ {}^{n}v^{n}v = 1;\ v^{n}v =\mid v \mid$$

and ${}^{n}v^{n}uu = {}^{n}v \mid u \mid$, which rotates u into the direction of v.

$$\begin{aligned}
\text{Try } {}^{n}v &= ce_1 + se_2 \\
u &= \mid u \mid (c'e_1 + s'e_2) = xe_1 + ye_2 \\
{}^{n}v^{n}u &= (cc' + ss') - (c's - cs')e_{12} \\
&= C - Se_{12}
\end{aligned}$$

where $C = \cos(-)$ and $S = \sin(-)$ as in Note 6. Compare this with 2-numbers, $C - iS$.

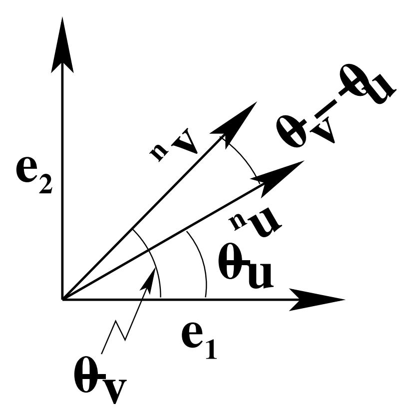
If we note that $e_{12}e_{12} = e_1e_2e_1e_2 = -e_1e_2e_2e_1 = -1$, we seem to find that e_{12} is the square root of -1. It's better to think of e_{12} as a $\pi/2$ rotation:

$$\begin{aligned}
e_{12}e_2 &= e_1 \\
e_{12}e_1 &= -e_2
\end{aligned}$$

So what is the meaning of $C - Se_{12}$?

$$\begin{aligned}
(C - Se_{12})u &= (C - Se_{12})(xe_1 + ye_2) \\
&= (Cx - Sy)e_1 + (Sx + Cy)e_2 \\
&= (e_1\ e_2)\begin{pmatrix} C & -S \\ S & C \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}
\end{aligned}$$

It's the rotation that rotates u onto v.



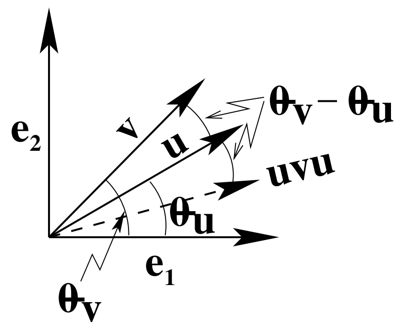
8. Reflection

If uuv and vuu rotate $u \to v$ what is uvu?

Let's try it with u and v normalized.

$$u = c'e_1 + s'e_2$$

$$
\begin{aligned}
v &= ce_1 + se_2 \\
uvu &= (c'e_1 + s'e_2)(ce_1 + se_2)(c'e_1 + s'e_2) \\
&= Ce_1 + Se_2 \text{ where} \\
C &= \cos(\theta_u - \theta_v + \theta_u) = \cos(\theta_u - (\theta_v - \theta_u)) \\
S &= \sin(\theta_u - \theta_v + \theta_u) = \sin(\theta_u - (\theta_v - \theta_u))
\end{aligned}
$$



uvu is the *reflection* of v in u.

(Another viewpoint: since $w(vu)$ rotates w by the angle between v and u, so (vu) is the reflection of v in u.)

Note that the *projection* of v in u is $(uvu + v)/2$, which can be written as a relationship among the reflection operator, F, the identity operator, I, and the projection operator, P: $P = (F + I)/2$.

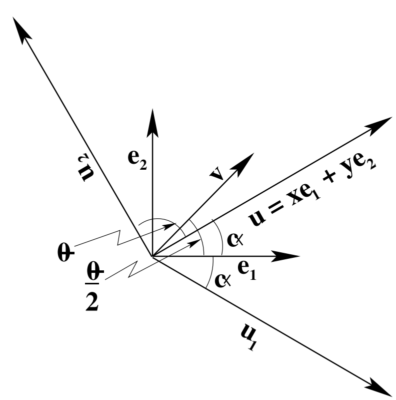
Note finally that a rotation is two reflections:

1. in e_1;
2. in "half-v", an element whose angle with e_1 is half the angle we wish to rotate through.

(We'll use the subscript J to indicate half-angles, since J sort of looks like 2 upside-down.)

$$
\begin{aligned}
u &= xe_1 + ye_2 & c = \cos\theta \quad c_J = \cos\theta/2 \\
v_J &= c_J e_1 + s_J e_2 & s = \sin\theta \quad s_J = \sin\theta/2 \\
v_J e_1 u e_1 v_J &= (c_J - s_J e_{12}) u (c_J + s_J e_{12}) \\
&= (e_1 \; e_2) \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}
\end{aligned}
$$

which is the rotation. (Recall that $c = c_J^2 - s_J^2$ and $s = 2c_J s_J$.) $\theta/2 + \theta/2 + \alpha - \alpha = \theta$:



9. 3D rotations

Outside of a 2-D plane we can't use $C + Se_{12}$ in 3-D:

$$e_3(C + Se_{12}) = Ce_3 + Se_{123}$$

(Note the extension of the rule for combining basis elements:

$$e_3e_{12} = e_3e_1e_2 = -e_1e_3e_2 = e_1e_2e_3 \stackrel{\text{def}}{=} e_{123})$$

So let's try two reflections:

$$\begin{aligned} \text{rotate } u &= xe_1 + ye_2 + ze_3 \\ \text{in plane } P &= re_{12} + pe_{23} + qe_{31} \end{aligned}$$

with P normalized: $p^2 + q^2 + r^2 = 1$.

$(c_J - s_JP)u(c_J + s_JP) =$

$$(e_1\ e_2\ e_3)\left(\begin{pmatrix} c & -sr & sq \\ sr & c & -sp \\ -sq & sp & c \end{pmatrix} + (1-c)\begin{pmatrix} p \\ q \\ r \end{pmatrix}(p, q, r)\right)\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Note that $pe_1 + qe_2 + re_3 \perp P = re_{12} + pe_{23} + qe_{31}$.

Note also that $\begin{pmatrix} p \\ q \\ r \end{pmatrix}$ is an eigenvector of the rotation matrix: what is the significance of that?

Now *two* rotations:

$$\begin{aligned} \text{by } (c, s) \text{ about} \qquad & pe_1 + qe_2 + re3 \\ \text{then by } (c', s') \text{ about} \qquad & p'e_1 + q'e_2 + r'e3 \\ & \Downarrow \\ \text{a rotation by } (c'', s'') \text{ about} \qquad & p''e_1 + q''e_2 + r''e3 \end{aligned}$$

$$\begin{aligned} &(c_J + s_J(re_{12} + pe_{23} + qe_{31}))(c'_J + s'_J(r'e_{12} + p'e_{23} + q'e_{31})) \\ &= c''_J + s''_J(r''e_{12} + p''e_{23} + q''e_{31}) \end{aligned}$$

where

$$\begin{aligned} c''_J &= c_Jc'_J - s_Js'_J(rr' + pp' + qq') \\ s''_Jr'' &= s_Jc'_Jr + c_Js'_Jr' + s_Js'_J(qp' - pq') \\ s''_Jp'' &= s_Jc'_Jp + c_Js'_Jp' + s_Js'_J(rq' - qr') \\ s''_Jq'' &= s_Jc'_Jq + c_Js'_Jq' + s_Js'_J(pr' - rp') \end{aligned}$$

Note that in 3-D all the angles are half angles.

Note that 3-D rotations do not commute.

10. Summary

(These notes show the trees. Try to see the forest!)

- Vectors are real things, independent of coordinates.

- So where they are written in terms of coordinates, these coordinates must transform correctly under rotation, reflection, projection and inversion: $X\vec{v}$.
- *Some* real things are not vectors, but tensors, and so tensor elements must also transform correctly: XTX^{-1}.
- Clifford or geometric or angle algebra:
  - parts of space: lines, areas, volumes, ..;
  - ignore position, consider only magnitude, direction;
  - basic elements are orthonormal and commutative.
- 2-D rotation from u to v is uuv or vuu.
- Reflection of v in u is uvu.
- 3-D rotation by (c, s) about $re_{12} + pe_{23} + qe_{31}$ ..
- Two 3-D rotations need half angles and are not commutative.

NB In 2-D: $1, e_1, e_2, e_3, e_{12}$. In 3-D: $1, e_1, e_2, e_3, e_{12}, e_{23}, e_{31}, e_{123}$.

11.Appendix: Summary of vector and matrix operations

$+$

$$\begin{aligned} \vec{u} + \vec{v} &= \begin{pmatrix} u_1 + v_1 \\ u_2 + v_2 \end{pmatrix} \\ A + B &= \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{pmatrix} \end{aligned}$$

$\bullet$

$$\begin{aligned} \vec{u}.\vec{v} &= (u_1\ u_2) \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \\ &= u_1 v_1 + u_2 v_2 \\ &= \mid \vec{u} \mid\mid \vec{v} \mid \cos(\angle(\vec{u}, \vec{v})) \\ A\vec{u} &= \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \\ &= \begin{pmatrix} a_{11}u_1 + a_{12}u_2 \\ a_{21}u_1 + a_{22}u_2 \end{pmatrix} \\ \vec{u}A &= (u_1\ u_2) \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \\ &= (u_1 a_{11} + u_2 a_{21}\ \ u_1 a_{12} + u_2 a_{22}) \\ AB &= \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix} \end{aligned}$$

$\otimes$

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{pmatrix}$$

Clifford algebra

$$\begin{aligned} uv &= (u_1e_1 + u_2e_2)(v_1e_1 + v_2e_2) \\ &= u_1v_1 + u_2v_2 + (u_1v_2 - u_2v_1)e_{12} \\ &= \vec{u}.\vec{v} + \mid \vec{u} \times \vec{v} \mid e_{12} \\ &= \mid u \mid\mid v \mid (\cos(\angle(\vec{u},\vec{v})) + \sin(\angle(\vec{u},\vec{v}))e_{12}) \end{aligned}$$

Compare $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(v_1, v_2) = \begin{pmatrix} u_1v_1 & u_1v_2 \\ u_2v_1 & u_2v_2 \end{pmatrix}$

Finally, compare these with 2-numbers (Week 4):

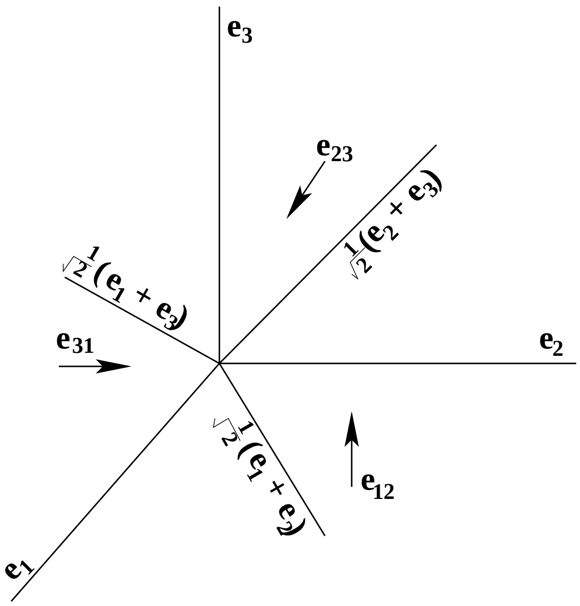
$$\begin{aligned} u + v &= u_1 + v_1 + i(u_2 + v_2) \\ uv &= (u_1 + iu_2)(v_1 + iv_2) \\ &= u_1v_1 - u_2v_2 + i(u_1v_2 + u_2v_1) \\ &= \mid u \mid e^{i\angle u} \mid v \mid e^{i\angle v} \\ &= \mid u \mid\mid v \mid e^{i(\angle u + \angle v)} \end{aligned}$$

12. **Excursions for Friday and beyond.**
You've seen lots of ideas. Now *do* something with them!

1. The dot product of two normalized vectors in any number of dimensions equals the cosine of the angle between the vectors. Show this: a) use $(Xu)^TXv = u^Tv$ to discover that the dot product is invariant under any axis transformation, X, whose transpose is its inverse; and b) use this invariance to reduce any two d-dimensional vectors, $\vec{u}$ and $\vec{v}$, to the two dimensions of their common plane.
   What is the angle between doc1 and doc2 in Note 2?
2. Calculate the reflections in the yz plane of twirls pointing along each of the x, y and z axes, and explain why what you get is right.
3. Confirm that w, u and v in the 3D twirl tensor must refer to the x, y and z components, respectively.
4. Is there a way to use 2-numbers to represent 3D twirl as a 2×2 tensor?
5. What is the matrix for the reflection of $u = xe_1 + ye_2$ in $v = ce_1 + se_2$ (c and s are cosine and sine, respectively, so v is normalized)?
6. Why is $u(vu)$ the reflection of v in u? Explain in terms of the rotation, (uv). (Take u and v to be normalized.)
7. Explain why the projection of v on u is $(uvu + v)/2$. For $u = c'e_1 + s'e_2$ and $v = ce_1 + se_2$, give the matrices F (reflection) and P (projection). What is the significance of $P - I$, where I is the identity matrix?
8. Show that 3D rotation by angle (c, s) about $re_{12} + pe_{23} + qe_{31}$ is the matrix given in Note 9. Show that $(p, q, r)^T$ is an eigenvector (Note 1 of Week 8), find the corresponding eigenvalue, and explain what these mean.
9. Check the derivation of the expression for double rotation in 3D. How would we find p'', q'' and r''?
10. Compare rotating by $\pi/2$ about (1,0,0) then $\pi/2$ about (0,1,0) with rotating $\pi/2$ about (0,1,0) then $\pi/2$ about (1,0,0). Use both angle algebra and your hands and some physical object such as a book.

11. Using rotations (and other operations) in the angle algebra and a starting edge, e_1, find the other two edges of an equilateral triangle. How would this help you draw it with a graphics program?

12. Using rotations (and other operations) in the angle algebra and the equilateral triangle of the previous Excursion, calculate the three edges needed to build it into an equilateral tetrahedron. How would you find the angles between the planes in the tetrahedron?



13. a) (Warmup and check.) What is the plane formed by the edges $\mathbf{e}_1$ and $(\mathbf{e}_2 + \mathbf{e}_3)/\sqrt{2}$? What is the angle between these two edges? What angle does the plane make with $\mathbf{e}_{12}$? (Keep all edges and planes normalized! Be careful about signs, and check what they mean!)
b) Answer the questions from (a) for the edges $(\mathbf{e}_1 + \mathbf{e}_3)/\sqrt{2}$ and $(\mathbf{e}_2 + \mathbf{e}_3)/\sqrt{2}$.

c) Examine and test the MATLAB function

```
% function [cos12,sin12,face12] = product(edge1,edge2)
% THM 070410    in file: product.m
% edge1: normalized 3-vector, e.g. [p1,q1,r1]
% edge2: normalized 3-vector, e.g. [p2,q2,r2]
% cos12 = p1p2+q1q2+r1r2
% sin12 = +sqrt(1-cos^2)
% face12: normalized 3-vector,
%  [(q1r2-r1q2)/sin12,(r1p2-p1r2)/sin12,(p1q2-q1p2)/sin12]
% (Works for planes as input, but use -cos12, -sin12)
function [cos12,sin12,face12] = product(edge1,edge2)
p1 = edge1(1); q1 = edge1(2); r1 = edge1(3);
p2 = edge2(1); q2 = edge2(2); r2 = edge2(3);
cos12 = p1*p2+q1*q2+r1*r2;
sin12 = sqrt(1-cos12^2):  % when might this be 0?
if abs(sin12)<10^-8 face12 = [0,0,0]; else
face12 = [(q1*r2-r1*q2)/sin12,(r1*p2-p1*r2)/sin12,(p1*q2-q1*p2)/sin12];
end
```

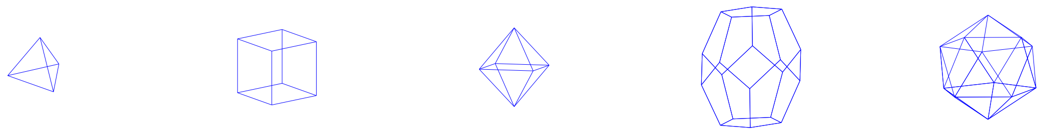
Why must we change the sign if `edge1` and `edge2` represent faces rather than edges on input?

(Hint. Multiplying by $\mathbf{e}_{12}$ in 2D gives a quarter-rotation. Does multiplying by $\mathbf{e}_{123}$ in 3D also do this? What does a "quarter rotation" mean in this case for an edge? For a face? What is $\mathbf{e}_{123}\mathbf{e}_{123}$?)
d) (Warmup and check.) Rotate the edges $\mathbf{e}_1$ and $(\mathbf{e}_2 + \mathbf{e}_3)/\sqrt{2}$ through the angle you found in (a) so as to put them both in $\mathbf{e}_{12}$: this should give $\mathbf{e}_1$ itself and $\mathbf{e}_2$, respectively.
e) Rotate the edges from (b) so as to put them both in $\mathbf{e}_{12}$. Check that they have the same angle with each other that they did before rotating.
f) Find two additional normalized edges that share with each of the new edges from (e) the same angle you found in (b) that they have with each other. (Note that the solution is direct if the input edges are in $\mathbf{e}_{12}$ but would require iteration if the $\mathbf{e}_3$ components of the edges are nonzero: try it!)
g) Write a MATLAB function, `e12equiAngle()`, for (f), i.e., which given two edges in $\mathbf{e}_{12}$ finds an edge sharing with those two edges the angle that is between the input edges.
Write a MATLAB function, `equiAngle()`, which given *any* two edges finds an edge sharing with those two edges the angle that is between the input edges: find the plane of the given edges, rotate it into the $\mathbf{e}_{12}$ plane, use `e12equiAngle()` to find the new edge, and rotate this back again. (The next excursion gives a possible `rotate3D()` function interface.)
h) Rotate the edge from (f) that has the negative $\mathbf{e}_3$ component inversely to the rotation in (e). What is the resulting combination of this edge and the two original edges in (b)?

14. Inspect and run the following MATLAB function.

```
% function [pentcoords,pentedges,pentface] = pentagon(startcoords,startedge,pentface)
% THM 070409    in file: pentagon.m
% Makes pentagon of unit edges, given 3D coords for 1 vertex, 1 edge, 1 plane
% startcoords 3-vector, e.g. [0,0,0]
% startedge 3-vector, e.g. [1,0,0]
% pentface 3-vector, e.g.[0,0,1]      The plane in which the pentagon is made
% pentcoords 5*3 array, e.g. [0,0,0;1,0,0;..]
% pentedges 5*3 array, e.g. [1,0,0;..]
% uses rotate3D
function [pentcoords,pentedges,pentface] = pentagon(startcoords,startedge,pentface)
angle = 2*pi/5;
edgesIN = startedge';
planesIN = pentface';
pentedges = edgesIN;
pentcoords = startcoords';
[edgesOUT,planesOUT] = rotate3D(pentface,angle,edgesIN,planesIN);
for k = 1:4
   pentedges = [pentedges,edgesOUT];
   coords = pentcoords(k,:) + pentedges(k,:)
   pentcoords = [pentcoords,coords];
   [edgesOUT,planesOUT] = rotate3D(pentface,angle,edgesOUT,planesIN);
end
```

Write the function `rotate3D(plane,angle,edgesIN,planesIN)`, which rotates arbitrary sets of `edgesIN` and `planesIN` about `angle` in `plane`.
Write a program which calls `pentagon()` and uses `quiver3` to draw the resulting pentagon.

15. 

Above are the five "Platonic solids": the tetrahedron (4 faces), the cube (hexahedron, 6 faces), octahedron (8 faces), dodecahedron (12 faces) and icosahedron (20 faces). Use the techniques of the previous excursions to build them in MATLAB.
(The cube and octahedron do not need angle algebra machinery and their edges can be written down straight from pairs of coordinates. They make a good place to start. The tetrahedron can also be written down directly from coordinates, or it can be made from an equilateral triangle and an additional vertex out of the plane and equidistant from each vertex of the triangle; but it is good exercise to use angle algebra for this, following the next-to-previous excursion or the notes on Clifford Algebra available from the course home page.)
By finding a way to draw the octahedron inside the cube and the icosahedron inside the dodecahedron, show that these are two pairs of "duals"—the faces of one of each pair correspond to the vertices of the other, and vice-versa. What is the dual of the tetrahedron?

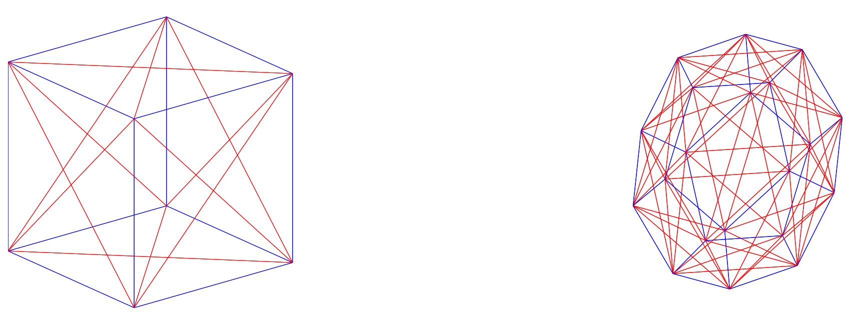
16. How many colours are needed to colour the vertices of each of the Platonic solids, if no two vertices of the same colour may be joined by an edge? How many colours for the faces, if no two faces separated by an edge as a boundary may have the same colour? What about colouring vertices of polygons in 2D?

17. Confirm that the Platonic solids satisfy

$$2 + E = F + V$$

where E is the number of edges, F is the number of faces and V is the number of vertices. Does this hold for any other figure?

18. How many spheres can be packed around a sphere of the same radius? (Hint: start with 2D and show that six circles pack a centre circle. What angle does each circle subtend at the centre? Approximately what proportion of the spherical surface area, $4\pi r$, is inside one of the packing spheres centred at distance r? Must the centres of the packing spheres form the vertices of one of the Platonic solids?)

19. 

The red additions to the cube and the dodecahedron above are the paths of length 2. That is, since the cube has a blue edge (0,0,0)–(1,0,0) and a blue edge (1,0,0)–(1,1,0), then (0,0,0)–(1,1,0) will be a red edge.

Here are all the coordinate pairs for the cube, in two different orders: the set on the left is sorted by columns 4, 5 and 6; the set on the right is sorted by columns 1, 2 and 3.

0	0	1	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	0	0
0	1	1	0	0	1	0	0	1	0	1	1
1	0	1	0	0	1	0	0	1	1	0	1
0	0	0	0	1	0	0	1	0	0	0	0
0	1	1	0	1	0	0	1	0	0	1	1
1	1	0	0	1	0	0	1	0	1	1	0
0	0	1	0	1	1	0	1	1	0	0	1
0	1	0	0	1	1	0	1	1	0	1	0
1	1	1	0	1	1	0	1	1	1	1	1
0	0	0	1	0	0	1	0	0	0	0	0
1	0	1	1	0	0	1	0	0	1	0	1
1	1	0	1	0	0	1	0	0	1	1	0
0	0	1	1	0	1	1	0	1	0	0	1
1	0	0	1	0	1	1	0	1	1	0	0
1	1	1	1	0	1	1	0	1	1	1	1
0	1	0	1	1	0	1	1	0	0	1	0
1	0	0	1	1	0	1	1	0	1	0	0
1	1	1	1	1	0	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	0	1	1
1	0	1	1	1	1	1	1	1	1	0	1
1	1	0	1	1	1	1	1	1	1	1	0

a) Confirm that these coordinate pairs link up so as to give the red edges shown with the cube.
b) Examine the following MATLAB code which will make the links you checked in (a). It implements a simplified *natural composition* operator of the *relational algebra*. It is built in terms of three other relational algebra operators, *natural join, projection* and a family of operators that treat relations as sets of rows and produce set difference (−), union (`u`), intersection (`n`) and symmetric difference (+). (Note that this last operator is here applied to the set of *columns* of the relations being put together.)
Look up [Mer99, Database programming], implement these operators, and show that `relationCompos()` applied to the coordinate pairs for the cube produces the red figures shown.
c) Run your `relationCompos()` on the coordinate pairs you got for the dodecahedron in an earlier excursion.

```
% function joinOut = relationCompos(joinIndices,joinIn1,joinIn2)
% THM 070420 in file relationCompos.m
% joinIndices 2*m array giving indices to be joined
% joinIn1 n1*m1 array
% joinIn2 n2*m2 array
% joinOut n*(m1-m+m2) array
% joinOut rows will be unduplicated if joinIn1 and joinIn2 rows are
% Uses relationSetOp(), relationJoin(), relationProject()
function joinOut = relationCompos(joinIndices,joinIn1,joinIn2)
sizIn1 = size(joinIn1);
sizIn2 = size(joinIn2);
```

```
sizInd = size(joinIndices(1,:));
%all = zeros(sizInd);              % indices for compareRows(): all columns
for k = 1:sizIn1(2) - sizInd(2) + sizIn2(2) all(k) = k; end
projIndices = relationSetOp('-',all',joinIndices(1,:)')
joinOut = relationProject(projIndices',relationJoin(joinIndices,joinIn1,joinIn2));
```

20. Look up H. S. M. Coxeter's *Regular Polytopes* [Cox63] and use the angle algebra to construct higher-dimensional versions of the tetrahedron, cube and octahedron.

21. How might we use the angle algebra to describe a shear operation?

22. Look up William Kingdon Clifford, 1845–1879, and describe his role in creating the angle algebra. (It is really called the Clifford algebra, or sometimes the geometric algebra.)

23. Look up Josiah Willard Gibbs, 1839–1903, and his vector analysis.

24. Look up Sir William Rowan Hamilton, 1805–1865, and his "quaternions". What mental block stumped him for a long time? How did he misinterpret what he invented, and how do quaternions relate to 3D angle algebra? (see [Alt92].)

25. How do the Pauli matrices (Week 6) relate to 3D angle algebra?

26. Why is the number of basic elements of d-dimensional angle algebra equal to 2^d?

27. Survey the usage of the phrase "real world" and distinguish a legitimate usage from a put-down of academics.

28. Any part of the lecture that needs working through.

References

[Alt92] Simon L. Altmann. *Icons and Symmetries*. Clarendon Press, Oxford, 1992.

[Cox63] H. S. M. Coxeter. *Regular Polytopes*. The MacMillan Company, Collier-MacMillan Ltd, New York, London, 1963. 2nd ed.

[Mer84] T. H. Merrett. *Relational Information Systems*. Reston Publishing Co., Reston, VA., 1984.

[Mer99] T. H. Merrett. Relational information systems. (revisions of [Mer84]):
Data structures for secondary storage http://www.cs.mcgill.ca/∼cs420
Database programming:
http://www.cs.mcgill.ca/∼cs612, 1999.