Multi-algebra Fluency

Dr Chris Doran

(Cambridge University, Monumo)

AGACSE, 2024

XXX.

APPLICATIONS OF GRASSMANN'S EXTENSIVE ALGEBRA*.

I PROPOSE to communicate in a brief form some applications of Grassmann's theory which it seems unlikely that I shall find time to set forth at proper length, though I have waited long for it. Until recently I was unacquainted with the *Ausdehnungslehre,* and knew only so much of it as is contained in the author's geometrical papers in *Crelle's Journal* and in *Hankel's Lectures on Complex Numbers.* I may, perhaps, therefore be permitted to express my profound admiration of that extraordinary work, and my conviction that its principles will exercise a vast influence upon the future of mathematical science.

The problem

GA is fragmenting into people advocating one algebra over another

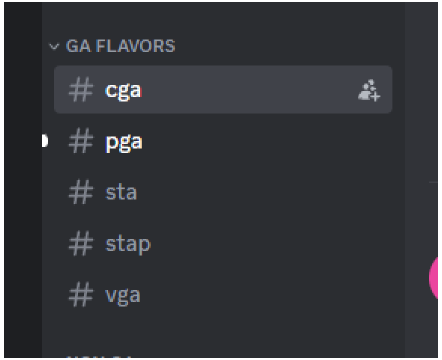
Also seeing a plethora of different products, notations etc.

(Nothing new there)

But code exacerbates the problem

Generally, flexible multi-algebra code is slow

Optimised code tends to lock you into a single algebra



$$\cancel{Q \wedge a \wedge \tilde{Q}}$$

$$\cancel{Q \vee a \vee \underset{\sim}{Q}}$$

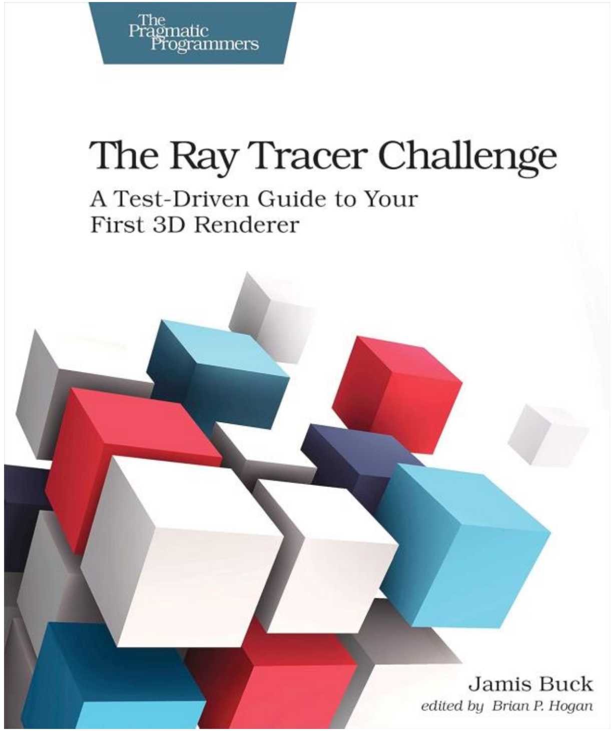
The problem

Frustrations while working through this book

Need:

- Point (x, y, z, 1)
- Vector(x, y, z)
- Plane (n_x, n_y, n_z, d)
- Sphere
- Ray …

In practice you want to move between (3,0), (4,0), (4,1) and (3, 0, 1) to be in the optimal algebra for each step.



Universal geometric algebra

There is ‘one’ algebra G(n,n)
Here n is as big as you need it to be!
It is a balanced algebra
David called this the ‘mother’ algebra
Perhaps universal is a better name!

Can just implement this, but you miss some critical optimizations.

In practice, may need to implement something more streamlined.

In: Acta Applicandae Mathematicae, Kluwer Academic Publishers 23: 65-93, (1991).

The Design of Linear Algebra and Geometry

David Hestenes

Projective Geometry with Clifford Algebra*

DAVID HESTENES and RENATUS ZIEGLER

In: *J. Math. Phys.*, **34** (8) August 1993 pp. 3642–3669.

Lie groups as spin groups

C. Doran
Department of Applied Mathematics and Theoretical Physics, Silver Street, Cambridge CB3 9EW, United Kingdom

D. Hestenes
Department of Physics and Astronomy, Arizona State University, Tempe, Arizona 85287

F. Sommen[a)] and N. Van Acker
Department of Mathematical Analysis, University of Gent, Galgaan 2, 9000 Gent, Belgium

Outline

How do we achieve multi-algebra fluency?

1. Understand the different ways larger algebras encompass smaller ones
2. (optional) Know the matrix representations

We will review these relationships with a view to:

1. Helping us write more efficient code.
2. Designing base units to move smoothly between algebras.

Matrices? Really?

Suppose we are in CGA and have two even rotors representing some conformal transformation.

Each has 16 elements, so a naïve implementation of their product would involve 256 multiplication operations.

Representing these as 2x2 matrices of quaternions reduces that down to 128.

So worthwhile.

Generally, in the algebras between 3D and 6D there are factors of 2 (or sometimes 4) to be found.

This all supposes we want to be in ‘geometric product first’ setting.

Why should I care?

This claim is not only dead wrong, but completely absurd. The authors made an error in their calculation of the number of operations needed to compose two rotors. The correct number is $2^{n-1} \cdot 2^{n-1} = 2^{2n-2}$, but they thought the number was just 2^n. The same mistake is made later on the same page, where they explicitly write $2^{n-1} \times n \times 2^{n-1} = n2^n$. This second instance is listed in the errata, but it doesn't look like anybody noticed the first one. It's clear that while one of the authors was writing this section, he had it stuck in his head that multiplying 2^{n-1} by itself gave 2^n. These kinds of errors happen, and the fact that something was simply miscalculated is not my chief complaint. The real problem here is that none of the three authors paused for a second and thought to themselves, "Wait a minute. It can't be *that* good because it would tell us there's something fundamentally inefficient about linear algebra that we don't understand." If the conclusion was correct, then it would open new avenues of research into exactly why orthogonal matrix transformations are so bad. But that didn't happen, of course, because the authors are wrong. The correct numbers are listed in the following table up through five dimensions.

n	**Matrix Composition** n^3 **operations**	**Rotor Composition** 2^{2n-2} **operations**
2	4 *	4
3	24 *	16
4	64	64
5	125	256

Wrong!

Entries with an asterisk under matrix composition have been reduced by exploiting the orthogonality of the matrices. Some GA enthusiasts have cried foul when I do this, claiming that it's somehow not fair that I use the most efficient method of calculation available, but they don't hesitate to pat themselves on the back whenever they're able to make use of a similar type of optimization that happens to benefit GA. On multiple occasions, I've seen GA authors purposely compare the best possible implementation of a GA method against the worst possible implementation of the equivalent matrix method in an attempt to demonstrate superiority that doesn't actually exist. An example is highlighted in PGA4CS below.

Matrix Representations of G(p,q)

$p+q$ \ $p-q$	8	7	6	5	4	3	2	1	0	−1	−2	−3	−4	−5	−6	−7	−8
0									$\mathbf{R}$								
1								$\mathbf{R}^2$		$\mathbf{C}$							
2							$M_2(\mathbf{R})$		$M_2(\mathbf{R})$		$\mathbf{H}$						
3						$M_2(\mathbf{C})$		$M_2^2(\mathbf{R})$		$M_2(\mathbf{C})$		$\mathbf{H}^2$					
4					$M_2(\mathbf{H})$		$M_4(\mathbf{R})$		$M_4(\mathbf{R})$		$M_2(\mathbf{H})$		$M_2(\mathbf{H})$				
5				$M_2^2(\mathbf{H})$		$M_4(\mathbf{C})$		$M_4^2(\mathbf{R})$		$M_4(\mathbf{C})$		$M_2^2(\mathbf{H})$		$M_4(\mathbf{C})$			
6			$M_4(\mathbf{H})$		$M_4(\mathbf{H})$		$M_8(\mathbf{R})$		$M_8(\mathbf{R})$		$M_4(\mathbf{H})$		$M_4(\mathbf{H})$		$M_8(\mathbf{R})$		
7		$M_8(\mathbf{C})$		$M_4^2(\mathbf{H})$		$M_8(\mathbf{C})$		$M_8^2(\mathbf{R})$		$M_8(\mathbf{C})$		$M_4^2(\mathbf{H})$		$M_8(\mathbf{C})$		$M_8^2(\mathbf{R})$	
8	$M_{16}(\mathbf{R})$		$M_8(\mathbf{H})$		$M_8(\mathbf{H})$		$M_{16}(\mathbf{R})$		$M_{16}(\mathbf{R})$		$M_8(\mathbf{H})$		$M_8(\mathbf{H})$		$M_{16}(\mathbf{R})$		$M_{16}(\mathbf{R})$

H = quaternions

Null generators

First up, lets dispense with the 'null' algebras:

$$G(p,q,r) \in G(p+r, q+r)$$

Easy to do. For each null direction introduce a pair of vectors of opposite signature.

If you want a totally null algebra, you get back the 'balanced' algebra

(Also get to a balanced algebra if you combine a vector space and its dual space. See Lie groups as spin groups …)

The containment implied above is usually via upper / lower triangular matrices (see PGA later)

Pseudoscalar

In odd-dimensional algebras the pseudoscalar commutes with every element in the algebra.

Not the case in even dimensions

The sign of the square of the pseudoscalar is critical.

$$I^2 = -1$$

Defines a complex structure

$$E^2 = 1$$

Defines a direct sum structure

$$\tfrac{1}{2}(1+E), \quad \tfrac{1}{2}(1-E)$$

Idempotents. Split the algebra in two.

$$(1+E)(1-E) = 0$$

Matrix Representations of G(p,q)

$p+q$ \ $p-q$	8	7	6	5	4	3	2	1	0	−1	−2	−3	−4	−5	−6	−7	−8
0									$\mathbf{R}$								
1								$\mathbf{R}^2$		$\mathbf{C}$							
2							$M_2(\mathbf{R})$		$M_2(\mathbf{R})$		$\mathbf{H}$						
3						$M_2(\mathbf{C})$		$M_2{}^2(\mathbf{R})$		$M_2(\mathbf{C})$		$\mathbf{H}^2$					
4					$M_2(\mathbf{H})$		$M_4(\mathbf{R})$		$M_4(\mathbf{R})$		$M_2(\mathbf{H})$		$M_2(\mathbf{H})$				
5				$M_2{}^2(\mathbf{H})$		$M_4(\mathbf{C})$		$M_4{}^2(\mathbf{R})$		$M_4(\mathbf{C})$		$M_2{}^2(\mathbf{H})$		$M_4(\mathbf{C})$			
6			$M_4(\mathbf{H})$		$M_4(\mathbf{H})$		$M_8(\mathbf{R})$		$M_8(\mathbf{R})$		$M_4(\mathbf{H})$		$M_4(\mathbf{H})$		$M_8(\mathbf{R})$		
7		$M_8(\mathbf{C})$		$M_4{}^2(\mathbf{H})$		$M_8(\mathbf{C})$		$M_8{}^2(\mathbf{R})$		$M_8(\mathbf{C})$		$M_4{}^2(\mathbf{H})$		$M_8(\mathbf{C})$		$M_8{}^2(\mathbf{R})$	
8	$M_{16}(\mathbf{R})$		$M_8(\mathbf{H})$		$M_8(\mathbf{H})$		$M_{16}(\mathbf{R})$		$M_{16}(\mathbf{R})$		$M_8(\mathbf{H})$		$M_8(\mathbf{H})$		$M_{16}(\mathbf{R})$		$M_{16}(\mathbf{R})$

Division algebras

Odd dimensions are either complex or a direct sum (alternate)

Efficiency is driven by size of spinor space

Triality / octonions

Key relations 1

$$G(p+1,q+1) = G(p,q) \otimes G(1,1)$$

$$G(1,1) \cong M_2(\boldsymbol{R})$$

Use these relations together to reduce algebra down to remaining Euclidean or anti-Euclidean bits.

$$e_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad f_1 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

Basis representation for G(1,1)

$$G(n,n) = G(1,1) \otimes G(1,1) \cdots \otimes G(1,1)$$

These all commute!
That should be a surprise

Example

Construct this explicitly with G(2,2)

$$\{e_1, f_1, e_2, f_2\}, \quad e_1 e_1 = e_2 e_2 = 1 \quad f_1 f_1 = f_2 f_2 = -1$$

$$1 \qquad \begin{matrix} e_1, e_2 \\ f_1, f_2 \end{matrix} \qquad \begin{matrix} e_1 e_2, e_1 f_2, e_1 f_1 = N_1 \\ f_1 f_2, f_1 e_2, e_2 f_2 = N_2 \end{matrix} \qquad \begin{matrix} e_1 N_2, f_1 N_2 \\ e_2 N_1, f_2 N_1 \end{matrix} \qquad E = N_1 N_2$$

4 vectors

6 bivectors

4 trivectors

$$\{1, e_1, f_1, N_1\} \otimes \{1, e_2 N_1, f_2 N_1, N_2\}$$

$$e_2 N_1 f_2 N_1 = e_2 f_2 N_1 N_1 = e_2 f_2$$

Closed

These generators are trivectors in the base space

Grade is a relative concept.

Balanced Algebra

Basic bit-wise implementation goes as (8-bit representation)

$$00000000 \leftrightarrow 1$$

Scalar

$$00000001 \leftrightarrow e_1$$

$$00000010 \leftrightarrow f_1$$

$$00000100 \leftrightarrow e_2 N_1$$

trivector

$$00001000 \leftrightarrow f_2 N_1$$

$$00010000 \leftrightarrow e_3 N_1 N_2$$

$$00100000 \leftrightarrow f_3 N_1 N_2$$

5-vectors

So, for example

$$f_3 \leftrightarrow 00101111$$

N_2 N_1

Balanced Algebra

Result of blade multiplication is still a bitwise XOR.

Due to commutativity, Just need to get the sign correct for each 2x2 block.

	1	e_1	f_1	$e_1 f_1$
1	1	e_1	f_1	$e_1 f_1$
e_1	e_1	1	$e_1 f_1$	f_1
f_1	f_1	$-e_1 f_1$	-1	e_1
$e_1 f_1$	$e_1 f_1$	$-f_1$	$-e_1$	1

So, introduce a factor of -1 if:

1. The first entry contains an f AND
2. The second entry is a vector

Balanced Algebra

First term contains *f*: n AND 10101010

Second term is a vector:

$$
\begin{array}{l}
01001110 \\
10011100 \quad \text{Shift left} \\
\hline
11010010 \quad \text{XOR}
\end{array}
$$

Keep these terms (AND with vector above)

Shift left, XOR and AND are all primitive bit-wise operations.

All available in C++, Julia etc.

3 lines of Julia code!

Issues

This is slow!

You end up performing m*n multiplications for each product.

- OK if the arguments are homogenous
- Inefficient for general elements (such as rotors) – misses the idempotent idea

After multiplying there is a simplification stage, which involves sorting the m*n products, and then combining ones with the same blade.

- This can be slow, and not very hardware friendly.
- Some progress with JIT compilers (Brandon Flores https://github.com/brainandforce)

For small algebras matrix implementations are more efficient for most operations. If speed not an issue, then just work in G(n,n).

Matrix Representations of G(p,q)

$p+q$ \ $p-q$ →	8	7	6	5	4	3	2	1	0	−1	−2	−3	−4	−5	−6	−7	−8
0									$\mathbf{R}$								
1								$\mathbf{R}^2$		$\mathbf{C}$							
2							$M_2(\mathbf{R})$		$M_2(\mathbf{R})$		$\mathbf{H}$						
3						$M_2(\mathbf{C})$		$M_2^2(\mathbf{R})$		$M_2(\mathbf{C})$		$\mathbf{H}^2$					
4					$M_2(\mathbf{H})$		$M_4(\mathbf{R})$		$M_4(\mathbf{R})$		$M_2(\mathbf{H})$		$M_2(\mathbf{H})$				
5				$M_2^2(\mathbf{H})$		$M_4(\mathbf{C})$		$M_4^2(\mathbf{R})$		$M_4(\mathbf{C})$		$M_2^2(\mathbf{H})$		$M_4(\mathbf{C})$			
6			$M_4(\mathbf{H})$		$M_4(\mathbf{H})$		$M_8(\mathbf{R})$		$M_8(\mathbf{R})$		$M_4(\mathbf{H})$		$M_4(\mathbf{H})$		$M_8(\mathbf{R})$		
7		$M_8(\mathbf{C})$		$M_4^2(\mathbf{H})$		$M_8(\mathbf{C})$		$M_8^2(\mathbf{R})$		$M_8(\mathbf{C})$		$M_4^2(\mathbf{H})$		$M_8(\mathbf{C})$		$M_8^2(\mathbf{R})$	
8	$M_{16}(\mathbf{R})$		$M_8(\mathbf{H})$		$M_8(\mathbf{H})$		$M_{16}(\mathbf{R})$		$M_{16}(\mathbf{R})$		$M_8(\mathbf{H})$		$M_8(\mathbf{H})$		$M_{16}(\mathbf{R})$		$M_{16}(\mathbf{R})$

Need to understand the outer reps

$$G(p+1, q+1) = G(p,q) \otimes M_2(\boldsymbol{R})$$

Balanced algebras

Key relations 2

We have removed all mixed-signature contributions.

Next consider the identities

$$G(q+2, p) = G(p, q) \otimes G(2, 0)$$
$$G(q, p+2) = G(p, q) \otimes G(0, 2)$$

Proof is similar to G(1,1) case.
Main difference is that the pseudoscalars have negative square.

$$G(2, 0) \cong M_2(\mathbf{R})$$

Easy to establish.

$$G(0, 2) \cong \mathbf{Q}$$

First appearance of quaternions.

Key relations 3

Build on the previous identities to establish

$$G(p+4,0) = G(p,0) \otimes M_2(\mathbf{Q})$$
$$G(0,p+4) = G(0,p) \otimes M_2(\mathbf{Q})$$

Generators in smaller space are 5-vectors in larger space

Need 2 side relations

$$\mathbf{Q} \otimes \mathbf{Q} = \mathbf{M_4(R)}$$

Not totally obvious.

$$\mathbf{Q} \otimes \mathbf{C} = \mathbf{M_2(C)}$$

Simple from the matrix rep of quaternions.

$$G(p+8,0) = G(p,0) \otimes M_8(\mathbf{R})$$
$$G(0,q+8) = G(0,q) \otimes M_8(\mathbf{R})$$

These ensure everything loops round with periodicity 8

Projective Splits and the ESA

$$G(p,q)^{+} = G(q,p-1) = G(p,q-1)$$

Key relationship 4

How does this work?

Suppose we have a basis: $\{e_1, e_2, \ldots, e_n\}$

Define bivectors : $E_i = e_i e_n \quad i = 1 \ldots n-1$

These generate a GA:

$$E_i E_j + E_j E_i = e_i e_n e_j e_n + e_j e_n e_i e_n = -(e_i e_j + e_j e_i) = -2\delta_{ij}$$

G(n,0) -> G(0,n-1)

(Different variations hold with mixed signatures).

Defines an embedding of G(0,n-1) inside G(n,0).

Different from the 'obvious' embedding.

Again 'grade' is a subjective concept here. The E_i can be grade 2 or 1.

$$G(p,q)^+ = G(q,p-1) = G(p,q-1)$$

$p+q$ \ $p-q$	8	7	6	5	4	3	2	1	0	−1	−2	−3	−4	−5	−6	−7	−8
0									$\mathbf{R}$								
1								$\mathbf{R}^2$		$\mathbf{C}$							
2							$M_2(\mathbf{R})$		$M_2(\mathbf{R})$		$\mathbf{H}$						
3						$M_2(\mathbf{C})$		$M_2^2(\mathbf{R})$		$M_2(\mathbf{C})$		$\mathbf{H}^2$					
4					$M_2(\mathbf{H})$		$M_4(\mathbf{R})$		$M_4(\mathbf{R})$		$\boxed{M_2(\mathbf{H})}$		$M_2(\mathbf{H})$				
5				$M_2^2(\mathbf{H})$		$\boxed{M_4(\mathbf{C})}$		$M_4^2(\mathbf{R})$		$M_4(\mathbf{C})$		$M_2^2(\mathbf{H})$		$M_4(\mathbf{C})$			
6			$M_4(\mathbf{H})$		$M_4(\mathbf{H})$		$M_8(\mathbf{R})$		$M_8(\mathbf{R})$		$\boxed{M_4(\mathbf{H})}$		$M_4(\mathbf{H})$		$M_8(\mathbf{R})$		
7		$M_8(\mathbf{C})$		$M_4^2(\mathbf{H})$		$M_8(\mathbf{C})$		$M_8^2(\mathbf{R})$		$M_8(\mathbf{C})$		$M_4^2(\mathbf{H})$		$M_8(\mathbf{C})$		$M_8^2(\mathbf{R})$	
8	$M_{16}(\mathbf{R})$		$M_8(\mathbf{H})$		$M_8(\mathbf{H})$		$M_{16}(\mathbf{R})$		$M_{16}(\mathbf{R})$		$M_8(\mathbf{H})$		$M_8(\mathbf{H})$		$M_{16}(\mathbf{R})$		$M_{16}(\mathbf{R})$

CGA

STA

CSTA

Two families

‘Geometry’ family

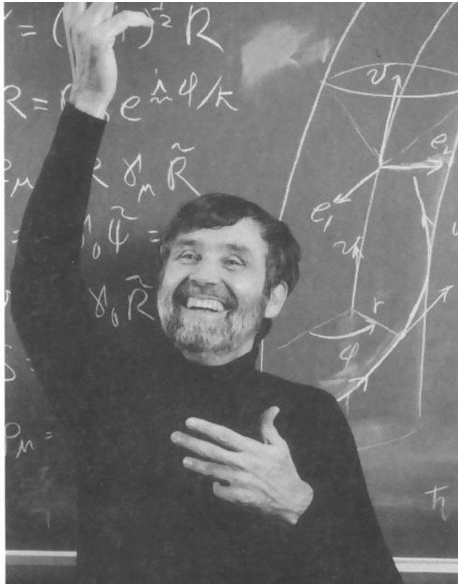
$$G(3,0)^+ = q$$
$$G(4,0)^+ = (q, q)$$
$$G(3,0,1)^+ = (q, q)$$
$$G(4,1)^+ = (q, q, q, q)$$

3D Geometry is quaternionic

‘Relativistic’ family

$$G^+(2,0) = c$$
$$G^+(1,3) = M_2(c)$$
$$G^+(2,4) = M_4(c)$$

Spacetime physics is complex



Quaternion blocks

Q1 Can we use quaternions as basic units to move between the 'geometry' algebras?

Q2 Do quaternion building blocks help us produce an efficient implementation?

Quickly find you need two primitives:

Quaternion vector `Qvec(x, y, z)`

Full quaternion `Quat(x, y, z, w)`

And 2 products: $\boldsymbol{q}_1 \times \boldsymbol{q}_2,\qquad q_1 q_2$

Clearly going to use Qvecs for vectors, and Quats for points.

Also need addition, multiplication by a scalar…

Projective Geometry G(4,0)

This is the algebra for 'proper' projective geometry.

Even sub-algebra (ESA) is a quaternion pair:

$$\{1, e_i e_j, E_4\}, \qquad E_4 = e_1 e_2 e_3 e_4$$

$$(E_4)^2 = (e_1 e_2 e_1 e_2)(e_3 e_4 e_3 e_4) = +1$$

$$E_+ = \tfrac{1}{2}(1 + E_4), \qquad E_- = \tfrac{1}{2}(1 - E_4)$$

Pseudoscalar has positive square and commutes with ESA. Define split: $M = ME_+ + Me_-$

Project each half down to a quaternion

Product: $MN = M_+N_+E_+ + M_-N_-E_-$

Product is now simply 2 quaternion products

Map from odd to even is usual 'projective split'.

$$M = ue_4$$

Odd element u

Projective Geometry G(4,0)

Implementation in terms of Quats is obvious:

$$q(a) = a_w + a_x i + a_y j + a_z k$$

$$a \mapsto ae_4 \mapsto (q(a), \tilde{q}(a))$$

Our generic 'point' to drop into any algebra

Lines through the origin become bivectors

$$\boldsymbol{a} \mapsto (\boldsymbol{q}(\boldsymbol{a}), -\boldsymbol{q}(\boldsymbol{a}))$$

The geometric product of two vectors involves

$$\begin{aligned} ab &= ae_4e_4be_4 \\ &= (q(a), \tilde{q}(a)) \times (\tilde{q}(b), q(b) \\ &= (q(a)\tilde{q}(b), \tilde{q}(a)q(b)) \end{aligned}$$

Only need these 12 products

$$q_0(a)\boldsymbol{q}(b)$$

$$q_0(b)\boldsymbol{q}(a)$$

$$\boldsymbol{q}(a) \times \boldsymbol{q}(b)$$

Speed

All speed tests carried out Lenovo laptop with Ryzen 6000 mobile CPU.
Timings in Julia using BenchmarkTools package.

`project(a*b, 2)` 14ns — Base result for two full quaternion products

$$q_0(a)\boldsymbol{q}(b)$$
$$q_0(b)\boldsymbol{q}(a)$$
$$\boldsymbol{q}(a) \times \boldsymbol{q}(b)$$

→ 5ns — Just using Qvec products

Similar holds for a.B and a^B.
Clearly worth implementing dot and wedge products separately. Particularly for G(4,0) (projective) geometry, as we hardly ever use a full geometric product (rotors not much use).

Speed

Still a role to play for actual thinking!

Consider the determinant of a 4x4 matrix:

Baseline LinearAlgebra package: `det(M)` 240ns

SimpleGA: `project(a*b*c*4,4)` 40ns

`dot(project(a*b, 2),project(E_4*c*d, 2))` 40ns

Using a direct implementation of a^b and Qvecs:

$$\langle (a \wedge b) E_4 (c \wedge d) \rangle$$ 10ns

Projective and conformal geometry

Projective and conformal geometries are different!

Projective geometry

- Any two planes meet in a single line
- No metric information
- Invariance group is GL(n+1).
- Affine plane + hyperplane at infinity
- Duality (1825)

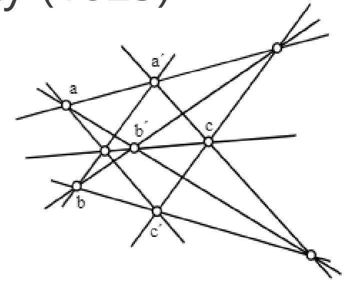


Fig. 2a Pappus configuration

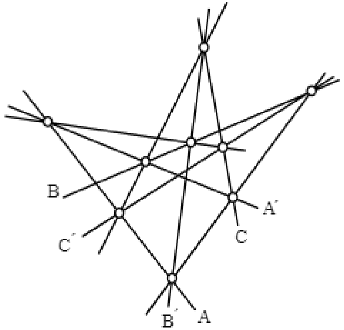
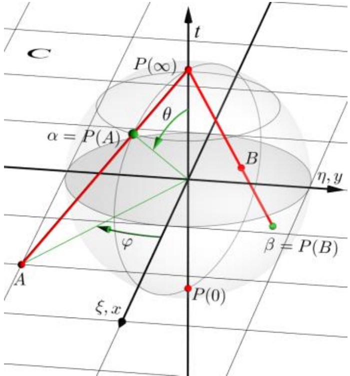


Fig. 2b Pappus configuration

Conformal geometry

- Distance encoded via dot product
- Invariance group SO(p+1, q+1)
- Single point at infinity
- Inversion
- Lines and circles u
- Algebraic duality



CGA G(4,1)

Even / odd swap is on pseudoscalar

Point $q(x) \mapsto \begin{pmatrix} \boldsymbol{q}(x) & -x_0 \\ \boldsymbol{x}^2/x_0 & -\boldsymbol{q}(x) \end{pmatrix}$

Null, but not normalised

Line through origin $\boldsymbol{q} \mapsto \begin{pmatrix} \boldsymbol{q} & 0 \\ 0 & \boldsymbol{q} \end{pmatrix}$

These are trivectors.
See how 3D drops out of 5D

Bivector $\begin{pmatrix} q_1 & \boldsymbol{q}_2 \\ \boldsymbol{q}_3 & -\tilde{q}_1 \end{pmatrix}$

Euclidean transform generator $\begin{pmatrix} \boldsymbol{q} & 0 \\ \boldsymbol{a} & \boldsymbol{q} \end{pmatrix}$

Speed:
Generic product takes 40ns
A^B for vectors takes 6ns
(Dependent on division by 2)

PGA G(3,0,1)

Even elements are again a quaternion pair.
See this from the embedding in CGA:

$$\psi = q_1 + q_2 I_3 n \longleftrightarrow \begin{pmatrix} q_1 & 0 \\ q_2 & q_1 \end{pmatrix}$$

This triangular structure is typical of null algebras

Even / odd map is performed with $I_3 = e_1 e_2 e_3$

Point $q(a) \mapsto (q_0, \boldsymbol{q}(a))$ Mixed representation. Bit unusual

Qvec $\boldsymbol{q}(\boldsymbol{a}) \mapsto (\boldsymbol{q}(\boldsymbol{a}), 0)$ The dual plane through the origin

Even product involves 3 quaternion multiplies. Definitely worth implementing a^b directly.

$$\psi\phi = (q_1 r_1) + (q_1 r_2 + q_2 r_1) I_3 n$$

PGA – Points at infinity

A puzzle with PGA is how it finds room for all the extra structure at infinity. One way to think about this is via the embedding:

$$q(a) \mapsto \begin{pmatrix} q_0 & 0 \\ \boldsymbol{q} & q_0 \end{pmatrix} = q_0 \begin{pmatrix} 1 & 0 \\ \boldsymbol{q}/q_0 & 1 \end{pmatrix}$$

Up in CGA this is an un-normalized translator, so

$$q(a) \mapsto q_0 T(\boldsymbol{q}/q_0)$$

So, this is how the extra points at infinity are smuggled in. As a limiting case of un-normalized rotors. Can even think of the point representations as spinors for the translation group!

Grade

Seen the same quaternion vector / line through origin in multiple places now:

- As grade-1 generators of G(3,0)
- As grade-2 bivectors in G(4,0)
- As grade-1 vectors in PGA (planes through the origin)
- As spinors (grade 0 + grade 2) for translators in PGA / CGA
- As grade-3 lines in CGA

Do not think too rigidly about grade!

Conclusions

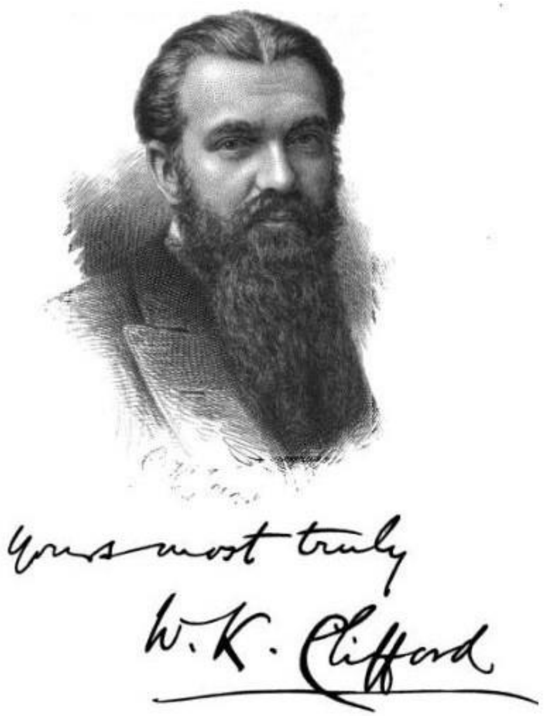
- Everyone should know how to jump between algebras
- Balanced algebras have some neat advantages
- Grade is a fluid concept
- Quaternion-like primitives can help us move between the algebras most routinely encountered in practice
- Enables the construction of efficient inner, outer and geometric products
- Use matrices for full geometric products to gain some factors of 2

- ToDo: Implement this is SimpleGA (https://github.com/MonumoLtd/SimpleGA.jl)

Final words and thanks

There is no way that one can assess the contributions that this extraordinary man would have made, had he lived to a reasonable age. One has to make do with what he actually achieved. His heritage is, indeed, quite stunning, and we are all his beneficiaries.

Roger Penrose



monumo