Preferred Canonical Form for 3D Euclidean Geometric Algebra

Kurt Nalty

July 2, 2015

Abstract

Having gained more experience with geometric algebra, I have formed an opinion about my preferred canonical form. I prefer the cyclic bivector format, as opposed to the ascending numerical basis format, as it clearly shows scalar product terms in the products of vectors and bivectors.

Geometric Algebra Canonical Choices

When first introduced to geometric algebra, I noticed differences in canonical basis between various authors dealing with the bivector terms. One camp uses xy, xz, yz formats, while the other camp uses xy, yz, and zx.

Initially, I used the xy, yz, zx format, due to the sense of beauty and braid. However, as I developed software to automatically create multiplication tables, the xy, xz, yz format made sense as a simple canonical form well suited for bubble sort counting of inversions in higher order blades.

Example Problem with Ascending Numerical Form

Using the ascending canonical form given the following gives an associative algebra. My multivector format is

Multivector: (u.q, u.x, u.y, u.z, u.xy, u.xz, u.yz, u.xyz)

```
u = (a, b, c, d, e, f, g, h) * (A, B, C, D, E, F, G, H)

Note Ginac Hash Order
u.q   = C*c+B*b-h*H-E*e-F*f-g*G+a*A+d*D
u.x   = -d*F-g*H+D*f+b*A-E*c+B*a-h*G+C*e
u.y   = -B*e+E*b+g*D+A*c+F*h-d*G+a*C+f*H
u.z   = -g*C+d*A+G*c+a*D+F*b-E*h-e*H-B*f
u.xy  = -G*f+d*H+g*F+C*b+e*A+E*a-B*c+h*D
u.xz  = e*G-E*g-C*h-c*H-B*d+a*F+b*D+A*f
u.yz  = b*H-F*e+g*A-d*C+D*c+a*G+E*f+B*h
u.xyz = E*d+h*A+e*D+B*g-C*f+b*G+a*H-F*c

Clifford Product is Associative
```

However, we have a warped vector bivector product.

```
u = (0, b, c, d, 0, 0, 0, 0) * (0, 0, 0, 0, E, F, G, 0)

Ginac Hash Order
u.q   = 0
u.x   = -d*F-E*c
u.y   = E*b-d*G
u.z   = G*c+F*b
u.xy  = 0
u.xz  = 0
u.yz  = 0
u.xyz = E*d+b*G-F*c

u = (0, 0, 0, 0, E, F, G, 0) * (0, b, c, d, 0, 0, 0, 0)
Ginac Hash Order
u.q   = 0
u.x   = d*F+E*c
u.y   = -E*b+d*G
u.z   = -G*c-F*b
u.xy  = 0
u.xz  = 0
u.yz  = 0
u.xyz = E*d+b*G-F*c
```

This warped product looks to have a sign error on F. The F term corresponds to the xz field. We fix the error by changing our basis from xz to zx.

Having gained more experience, I have now returned to the xy, yz, zx camp, due to ease of identifying scalar product sequences in the vector/bivector produces. The use of the xz base rather than the zx base introduces a sign change, which spoils the obvious dot products terms in the vector/bivector products.

Preferred Structure for Euclidean 3D

My preferred structure for Euclidean 3D geometric multivectors is

```
struct G3E3{
ex q,x,y,z,xy,zx,yz,xyz;
G3E3() {q = 0; x = 0; y = 0; z = 0; xy = 0; zx = 0; yz = 0; xyz = 0;}
G3E3(ex qq, ex xx, ex yy, ex zz, ex xxyy, ex zzxx, ex yyzz, ex xxyyzz)
{q = qq; x = xx; y = yy; z = zz; xy = xxyy; zx = zzxx; yz = yyzz; xyz = xxyyzz;}
};
```

My preferred product for Euclidean 3D geometric multivectors is

```
G3E3 operator*(const G3E3 &a, const G3E3 &b) {
G3E3 c;
c.q   =   a.q*b.q    + a.x*b.x    + a.y*b.y    + a.z*b.z
        - a.xy*b.xy  - a.zx*b.zx  - a.yz*b.yz  - a.xyz*b.xyz;
c.x   =   a.q*b.x    + a.x*b.q    - a.y*b.xy   + a.z*b.zx
        + a.xy*b.y   - a.zx*b.z   - a.yz*b.xyz - a.xyz*b.yz ;
c.y   =   a.q*b.y    + a.x*b.xy   + a.y*b.q    - a.z*b.yz
        - a.xy*b.x   - a.zx*b.xyz + a.yz*b.z   - a.xyz*b.zx ;
c.z   =   a.q*b.z    - a.x*b.zx   + a.y*b.yz   + a.z*b.q
        - a.xy*b.xyz + a.zx*b.x   - a.yz*b.y   - a.xyz*b.xy ;
c.xy  =   a.q*b.xy   + a.x*b.y    - a.y*b.x    + a.z*b.xyz
        + a.xy*b.q   + a.zx*b.yz  - a.yz*b.zx  + a.xyz*b.z  ;
c.zx  =   a.q*b.zx   - a.x*b.z    + a.y*b.xyz  + a.z*b.x
       - a.xy*b.yz   + a.zx*b.q   + a.yz*b.xy  + a.xyz*b.y  ;
c.yz  =   a.q*b.yz   + a.x*b.xyz  + a.y*b.z    - a.z*b.y
        + a.xy*b.zx  - a.zx*b.xy  + a.yz*b.q   + a.xyz*b.x  ;
c.xyz =   a.q*b.xyz  + a.x*b.yz   + a.y*b.zx   + a.z*b.xy
```

```
          + a.xy*b.z   + a.zx*b.y   + a.yz*b.x   + a.xyz*b.q  ;
return c;
}
```

The test suite for this product is
`http://www.kurtnalty.com/CL3R3.ginac.cp`,
with results:

```
u = (a, b, c, d, e, f, g, h) * (A, B, C, D, E, F, G, H)
Ginac Hash Order
u.q   = -g*G+d*D+A*a-f*F-E*e+c*C+B*b-H*h
u.x   = -c*E+C*e-h*G-H*g+B*a+A*b-f*D+d*F
u.y   = -f*H+a*C+g*D+A*c-d*G-F*h+E*b-B*e
u.z   = -H*e+c*G-E*h-F*b+A*d+f*B+a*D-g*C
u.xy  = C*b+h*D-c*B+E*a+H*d-g*F+A*e+f*G
u.zx  = A*f+a*F-e*G+B*d-b*D+C*h+c*H+E*g
u.yz  = B*h+H*b+c*D-d*C+F*e+a*G+A*g-f*E
u.xyz = f*C+A*h+H*a+c*F+B*g+e*D+E*d+b*G

Clifford Product is Associative

u = (0, b, c, d, 0, 0, 0, 0) * (0, B, C, D, 0, 0, 0, 0)
Ginac Hash Order
u.q   = d*D+c*C+B*b
u.x   = 0
u.y   = 0
u.z   = 0
u.xy  = C*b-c*B
u.zx  = B*d-b*D
u.yz  = c*D-d*C
u.xyz = 0

u = (0, B, C, D, 0, 0, 0, 0) * (0, b, c, d, 0, 0, 0, 0)
Ginac Hash Order
u.q   = d*D+c*C+B*b
u.x   = 0
u.y   = 0
u.z   = 0
```

```
u.xy  = -C*b+c*B
u.zx  = -B*d+b*D
u.yz  = -c*D+d*C
u.xyz = 0

u = (0, b, c, d, 0, 0, 0, 0) * (0, b, c, d, 0, 0, 0, 0)
Ginac Hash Order
u.q   = c^2+d^2+b^2
u.x   = 0
u.y   = 0
u.z   = 0
u.xy  = 0
u.zx  = 0
u.yz  = 0
u.xyz = 0

u = (0, b, c, d, 0, 0, 0, 0) * (0, 0, 0, 0, E, F, G, 0)
Ginac Hash Order
u.q   = 0
u.x   = -c*E+d*F
u.y   = -d*G+E*b
u.z   = c*G-F*b
u.xy  = 0
u.zx  = 0
u.yz  = 0
u.xyz = c*F+E*d+b*G

u = (0, 0, 0, 0, E, F, G, 0) * (0, b, c, d, 0, 0, 0, 0)
Ginac Hash Order
u.q   = 0
u.x   = c*E-d*F
u.y   = d*G-E*b
u.z   = -c*G+F*b
u.xy  = 0
u.zx  = 0
u.yz  = 0
u.xyz = c*F+E*d+b*G
```

```
u = (0, b, c, d, 0, 0, 0, 0) * (0, 0, 0, 0, 0, 0, 0, H)
Ginac Hash Order
u.q   = 0
u.x   = 0
u.y   = 0
u.z   = 0
u.xy  = H*d
u.zx  = c*H
u.yz  = H*b
u.xyz = 0

u = (0, 0, 0, 0, 0, 0, 0, H) * (0, b, c, d, 0, 0, 0, 0)
Ginac Hash Order
u.q   = 0
u.x   = 0
u.y   = 0
u.z   = 0
u.xy  = H*d
u.zx  = c*H
u.yz  = H*b
u.xyz = 0

u = (0, 0, 0, 0, e, f, g, 0) * (0, 0, 0, 0, E, F, G, 0)
Ginac Hash Order
u.q   = -g*G-f*F-E*e
u.x   = 0
u.y   = 0
u.z   = 0
u.xy  = -g*F+f*G
u.zx  = -e*G+E*g
u.yz  = F*e-f*E
u.xyz = 0

u = (0, 0, 0, 0, E, F, G, 0) * (0, 0, 0, 0, e, f, g, 0)
Ginac Hash Order
u.q   = -g*G-f*F-E*e
u.x   = 0
u.y   = 0
```

```
u.z   = 0
u.xy  = g*F-f*G
u.zx  = e*G-E*g
u.yz  = -F*e+f*E
u.xyz = 0

u = (0, 0, 0, 0, e, f, g, 0) * (0, 0, 0, 0, e, f, g, 0)
Ginac Hash Order
u.q   = -f^2-g^2-e^2
u.x   = 0
u.y   = 0
u.z   = 0
u.xy  = 0
u.zx  = 0
u.yz  = 0
u.xyz = 0

u = (0, 0, 0, 0, e, f, g, 0) * (0, 0, 0, 0, 0, 0, 0, H)
Ginac Hash Order
u.q   = 0
u.x   = -H*g
u.y   = -f*H
u.z   = -H*e
u.xy  = 0
u.zx  = 0
u.yz  = 0
u.xyz = 0

u = (0, 0, 0, 0, 0, 0, 0, H) * (0, 0, 0, 0, e, f, g, 0)
u.q   = 0
u.x   = -H*g
u.y   = -f*H
u.z   = -H*e
u.xy  = 0
u.zx  = 0
u.yz  = 0
u.xyz = 0
```

```
u = (0, 0, 0, 0, 0, 0, 0, h) * (0, 0, 0, 0, 0, 0, 0, H)
u.q   = -H*h
u.x   = 0
u.y   = 0
u.z   = 0
u.xy  = 0
u.zx  = 0
u.yz  = 0
u.xyz = 0

u = (0, 0, 0, 0, 0, 0, 0, H) * (0, 0, 0, 0, 0, 0, 0, h)
Ginac Hash Order
u.q   = -H*h
u.x   = 0
u.y   = 0
u.z   = 0
u.xy  = 0
u.zx  = 0
u.yz  = 0
u.xyz = 0

u = (0, 0, 0, 0, 0, 0, 0, h) * (0, 0, 0, 0, 0, 0, 0, h)
Ginac Hash Order
u.q   = -h^2
u.x   = 0
u.y   = 0
u.z   = 0
u.xy  = 0
u.zx  = 0
u.yz  = 0
u.xyz = 0

r = (a, b, c, d, e, f, g, h) u = Dual(r) = (-h, -g, -f, -e, d, c, b, a)
Ginac Hash Order
u.q   = -h
u.x   = -g
u.y   = -f
u.z   = -e
```

```
u.xy  = d
u.zx  = c
u.yz  = b
u.xyz = a

r = (a, b, c, d, e, f, g, h) u = RDual(r) = (-h, -g, -f, -e, d, c, b, a)
Ginac Hash Order
u.q   = -h
u.x   = -g
u.y   = -f
u.z   = -e
u.xy  = d
u.zx  = c
u.yz  = b
u.xyz = a

u = RegressViaWedge(r,s)
Ginac Hash Order
u.q   = f*C+c*F+B*g+e*D+E*d+b*G
u.x   = F*e-f*E
u.y   = -e*G+E*g
u.z   = -g*F+f*G
u.xy  = 0
u.zx  = 0
u.yz  = 0
u.xyz = 0

u = Regressive(r,s)
Ginac Hash Order
u.q   = -g*G+d*D+A*a-f*F-E*e+c*C+B*b-H*h
u.x   = -c*E+C*e-h*G-H*g+B*a+A*b-f*D+d*F
u.y   = -f*H+a*C+g*D+A*c-d*G-F*h+E*b-B*e
u.z   = -H*e+c*G-E*h-F*b+A*d+f*B+a*D-g*C
u.xy  = h*D+E*a+H*d-g*F+A*e+f*G
u.zx  = A*f+a*F-e*G+C*h+c*H+E*g
u.yz  = B*h+H*b+F*e+a*G+A*g-f*E
u.xyz = A*h+H*a
```

```
u = Wedge(r,s)
r = (a, b, c, d, e, f, g, h)
s = (A, B, C, D, E, F, G, H)
Ginac Hash Order
u.q   = 0
u.x   = 0
u.y   = 0
u.z   = 0
u.xy  = C*b-c*B
u.zx  = B*d-b*D
u.yz  = c*D-d*C
u.xyz = f*C+c*F+B*g+e*D+E*d+b*G
```

References

[1] Chris Doran and Anthony Lasenby, *Geometric Algebra for Physicists* Cambridge University Press, ISBN 978-0-521-71595-9

[2] Leo Dorst, Daniel Fontune and Stephen Mann, *Geometric Algebra for Computer Science* Morgan Kaufmann Publishers, ISBN 978-0-12-374942-0

[3] David Hestenes and Garret Sobczyk, *Clifford Algebra to Geometric Calculus* D. Reidal Publishing Company, ISBN 978-90-277-2581-5

[4] Anthony Lasenby and Chris Doran, *Lectures and Handouts 1999* `www.mrao.cam.ac.uk/clifford/ptIIIcourse/`