A MODERN TOOL FOR TEACHING PROJECTIVE GEOMETRIC ALGEBRA

Zachary Leger   Stephen Mann

University of Waterloo, Canada

From GABLE to PGABLE

In the late 90s, GABLE, a MATLAB package for OGA was created for a tutorial for 3 dimensional geometric algebras [4]. The project has largely been abandoned for more than 20 years, as the work shifted to Gaigen/GAViewer [2] and then Ganja. Although still compatible with present-day MATLAB, GABLE does not take advantage of the features of modern MATLAB. Thus, we have created PGABLE, a modernized version of GABLE capable of performing computations and visualizations for OGA, and have extended it to support PGA as a basis for a tutorial for learning PGA.

Background: Projective Geometric Algebra

Ordinary Geometric Algebra (OGA)

- 3D geometric algebra
- Euclidean basis vectors e_1, e_2 and e_3 where $e_1^2 = e_2^2 = e_3^2 = 1$
- Points (scalars), lines (vectors) and planes (bivectors) as well as volumes (trivectors) Centered at the origin

Projective Geometric Algebra (PGA)

- Also called Plane-based Geometric Algebra
- 4D geometric algebra representing Euclidean motion in 3D space
- Basis vectors e_0, e_1, e_2 and e_3, with $e_1^2 = e_2^2 = e_3^2 = 1$
- Null vector e_0 ($e_0^2 = 0$)
- Can represent points, lines and planes offset from the origin

Planes in PGA:

$$\vec{n} - \delta e_0$$

where $\vec{n}$ is a Euclidean vector representing the normal of the plane. Lines are constructed via the outer product of two planes. Points are formed from the outer product of three planes.

Computational Features

To reduce confusion, PGABLE keeps the OGA and PGA models separate. Users cannot perform operations using elements from distinct models. Different operations are available for each model.
Operations available to both models:

- **Basic operations**: addition (`+`), subtraction (`-`), the geometric product (`product, *`), the outer product (`outer,^`), the inner product (`inner, .*`)
- **Contractions**: left contraction (`lcont`), right contraction (`rcont`)
- **Set-like operations**: join (`join`), meet (`meet`)
- **Sign management**: conjugate (`conj`), grade involution (`gi`), reverse (`reverse`)
- **Basic norm operations**: norm (`norm`), normalize (`normalize`)
- **Advanced operations**: inverse (`inverse`), geometric exponentiation (`gexp`), the geometric logarithm (`glog`), square root (`sqrt`)

Since the pseudoscalar of PGA is not invertible, the operations of dual `dual` and inverse dual `invdual` are only available in OGA. There are also operations unique to PGA:

- The Hodge dual ($\star$) and its inverse are provided as `hd` and `ihd`.
- The Poincaré dual map, also called the JMap, is provided by `jmap` [5].
- In addition to `norm`, PGA is also equipped with `vnorm`, which computes the vanishing norm defined as

$$||A||_\infty = ||\star A||$$

Visualizations of OGA elements in PGABLE

PGABLE vectors can depict vectors, bivectors and trivectors just as GABLE can. However, the internal computations have been rewritten to use PGA geometric operations.

- **Vectors**: represented by an arrow with its tail at the origin.
- **Bivectors**: represented by a disk with hairs indicating its direction of spin.
- **Trivectors**: represented by a sphere whose radius is the magnitude with hairs that point outward if the pseudoscalar is positive and inwards if it is negative.

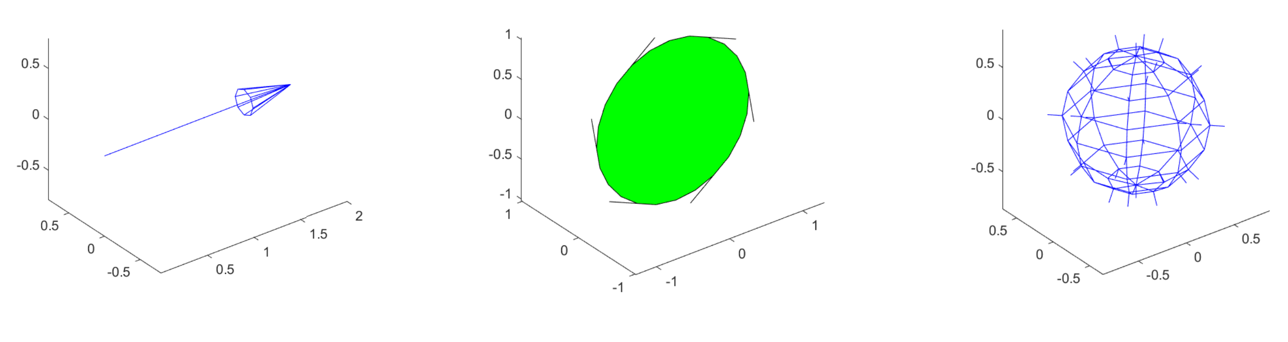


Figure 1. Visualizations of a vector, bivector and trivector in PGABLE.

Visualizations of PGA elements in PGABLE

- **Points**: represented by an octahedron at the point's position.
- **Lines**: represented by line with fletching. The fletching points in the direction of the line and spins around the line L corresponding to using the versor $exp(-\phi L/2)$.
- **Planes**: represented by a square with with 9 arrows pointing in the direction of the plane's normal.

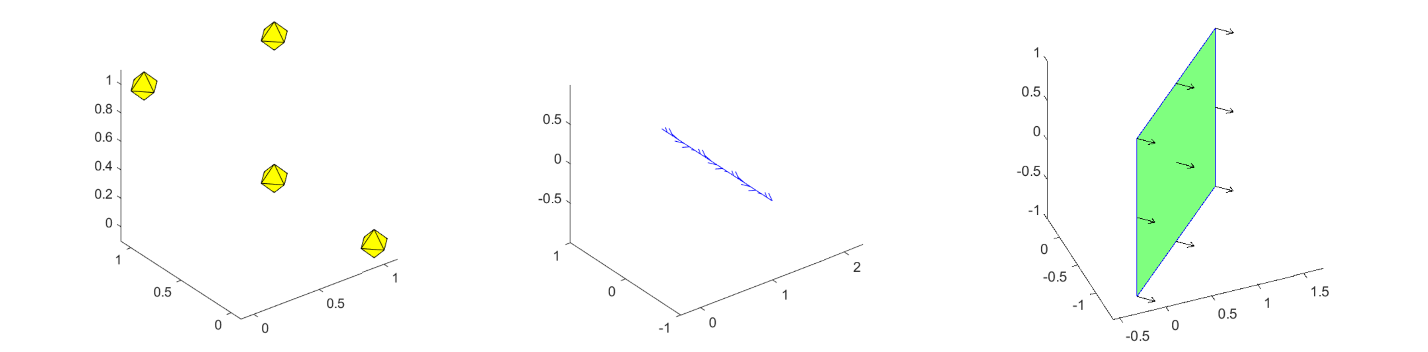


Figure 2. On the left, the 4 points (1, 0, 0) (1, 1, 0) (1, 1, 1) and (0, 1, 1).
In the middle, the visualization of the line $-e_{03} + e_{13}$.
On the right, a visualization of the plane $e_1 - e_2 - e_0$.

There are also points and lines at infinity, called *vanishing points* and *vanishing lines* respectively. Since we cannot draw points and lines at infinity, we draw these elements on MATLAB's bounding box.

- **Vanishing points:** represented by stars placed on the bounding box. A hair protrudes from the center of the star toward the center of the bounding box indicating the direction of the vanishing point.
- **Vanishing lines:** represented by a dashed quadrilateral which loops around the bounding box. Small triangles on the loop point towards the direction in which objects translate when the vanishing line L is used as a versor in $exp(-\phi L/2)$.

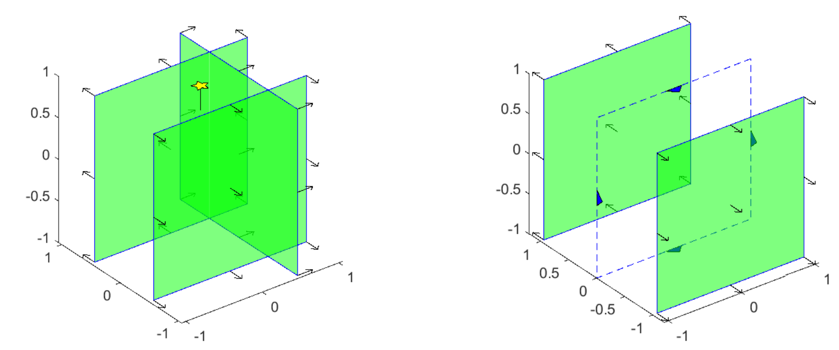
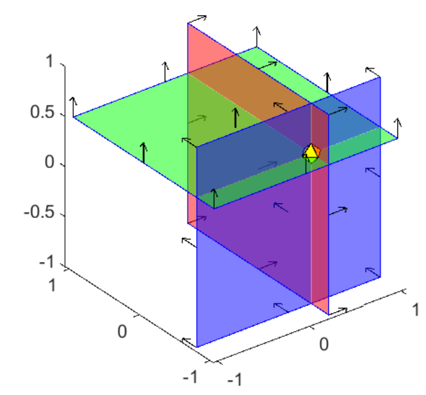


Figure 3. On the left, a visualization of three planes, $e_2 - 0.5e_0$ and $-e_2 - 0.5e_0$ which are parallel, and $-e_2 - 0.5e_0$ which intersects them. The outer product of these three planes gives the vanishing point $-e_0 \wedge e_1 \wedge e_2$, which is represented by a star.
On the right, the planes $-e_2 - e_0$ and $e_2 - e_0$ and the vanishing line $-2e_0 \wedge e_2$.

Example

```
>> n1 = e3 - 0.5*e0;
>> n2 = e1 - 0.25*e0;
>> n3 = e2 + 0.75*e0;
>> draw(n1); draw(n2,'r'); draw(n3,'b');
>> p = n1^n2^n3

p = -0.5*e0^e1^e2 + -0.75*e0^e1^e3 + -
0.25*e0^e2^e3 + e1^e2^e3
>> draw(p)
```



Getting PGABLE

PGABLE is still under development.

Please sign up if you would like to be notified when it is ready for download.

References

[1] Leo Dorst and Steven De Keninck.
A guided tour to the plane-based geometric algebra pga.
http://bivector.net/PGA4CS.html, March 2022.

[2] Daniel Fontijne.
Gaigen 2: a geometric algebra implementation generator.
In *Proceedings of the 5th International Conference on Generative Programming and Component Engineering*, GPCE '06, page 141–150, New York, NY, USA, 2006. Association for Computing Machinery.

[3] Charles Gunn.
Geometry, kinematics, and rigid body mechanics in Cayley-Klein geometries.
PhD thesis, TUBerlin, 2011.

[4] Stephen Mann, Leo Dorst, and Timaeus Bouma.
The making of GABLE: a geometric algebra learning environment in Matlab.
2001.

[5] Jeremy Ong.
Geometry potpourri.
https://www.jeremyong.com/klein/geometry-potpourri/#the-poincare-dual-map.