Geometric Algebra (GA)

Daniel Hug

2025

1 Documentation

Overview (high-level initial intros on youtube):

- *https://www.thestrangeloop.com*
- *https://www.youtube.com/@sudgylacmoe*
- *https://www.youtube.com/@bivector*

Sources:

- https://projectivegeometricalgebra.org
- https://terathon.com/blog/poor-foundations-ga.html
- https://bivector.net
- https://www.youtube.com/@bivector
- https://www.youtube.com/@sudgylacmoe
- https://www.youtube.com/watch?v=ItGlUbFBFfc
- https://www.youtube.com/watch?v=v-WGO2ILMXA&t=1476s

1.1 Introduction

The ga library is intended to make Geometric Algebra (GA) more accessible by providing functionality for numerical calculations required for simulation of physical systems.

It implements functionality for two- and three-dimensional spaces with Euclidean geometry. Currently it provides data types and operations to handle the algebras of regular Euclidean geometry ($ega2d, ega3d$) and projective Euclidean geometry ($pga2dp, pga3dp$). The library handles fully populated multivectors, even and odd grade multivectors as well as their components like scalars, vectors, bivectors, trivectors, and pseudoscalars of the corresponding linear spaces efficiently. It also provides operations to work with these objects. It was designed with extensibiliy in mind and could be easily extended to provide functionality for handling additional algebras, e.g. spacetime algebra (STA), with limited effort.

The main library is a header only library in the ga sub-folder of the library which can be used by including either

```
#include ga/ga_ega.hpp // or
#include ga/ga_pga.hpp
```

and by making the corresponding namespaces accessible by stating

```
using namespace hd::ga;      // and either
using namespace hd::ga::ega; // or
using namespace hd::ga::pga;
```

in your application code. For a simple usage example please have a look either at `ga_test/src/ga_ega_test.cpp` or at `ga_test/src/ga_pga_test.cpp`.

All product expressions for inner, outer, interior and geometric products and their regressive counterparts are generated via the file `ga_prdxpr/ga_prdxpr_main.cpp` and user input provided in the header files of the corresponding algebra in that folder. The generated coefficient expressions are used in the source code to implement dot products, wedge products, contractions, expansions, geometric products or their regressive counterparts, as well as sandwich products describing rotors and motors of the corresponding algebras. If certain products are required for your applicaton, but not yet implemented in the ga library they can easily be generated within `ga_prdxpr/src_prdxpr` as needed and added with minimal effort.

As mentioned before, we distinguish inner, interior, outer and geometric products.

- Inner and interior products are grade reducing operations.
- The inner product combines arguments of the same type and grade, while the interior products involve complements/duals in at least one of their arguments, combining and linking arguments of different grades. Inner products are linked to the metric of the modelled space.
- The outer product is a grade-increasing operation adding the grades of its arguments. Outer products, like the wedge product, connect subspaces represented by their operands and create new subspaces defined by the span of the subspaces of the operands.
- Geometric products are the combination of both types of products in one expression. They typically result in multivectors consisting of parts with different grades as a result of the operation. Their main advantage is invertibility under certain circumstances.
- Regressive products use a complement operation on their arguments before and/or after applying the operation linking the arguments of the product.
- Sandwich products are used to orthogonally transform objects implementation operators for rotation or general motion including rotation and translation. This can be achieved by multiplying the object symmetrically from the left and right side.

The complement operation used in this library is defined in [Lengyel, 2024b]. It is uniquely determined with respect to the outer product (not with respect to the geometric product, as commonly used for the dualization operation by many autors working with GA). This helps to generate consistent signs of expressions, operators, complements and duals. The left complement $\underline{u}$ and the right complement $\bar{u}$ are defined as

$$\underline{u} \wedge u = I_n \tag{1a}$$
$$u \wedge \bar{u} = I_n \tag{1b}$$

with I_n as the pseudoscalar of the n-dimensional algebra, and the wedge symbol ($\wedge$) as the operator symbol of the outer product. The effect of the complement is to turn basis elements not contained in the object u into the constituting basis elements of the complement of the object $\underline{u}$ or $\bar{u}$. Connecting the object and its complement with the wedge products creates an object

that fills the available space in the sense of requiring all basis elements to represent it. The full space is represented by the pseudoscalar of that space, since the pseudoscalar I_n is formed formed by the product of all n basis vectors of the space.

It can be shown that the left and right complements fulfill

$$\underline{\bar{a}} = a \tag{2}$$

which shows that the left and right complement operations are inverses to each other. This is valid independent of the sequence of application, independent of the grade of the argument a, and independent of the dimension of the space modelled in the algebra. Based on this it is possible to define so-called regressive products using rules analog to deMorgan's rules in boolean algebra as follows (definitions can be found in [Lengyel, 2024b] and in [Browne, 2012]):

$$a \vee b \equiv \underline{\bar{a} \wedge \bar{b}} \tag{3a}$$
$$a \vee b \equiv \overline{\underline{a} \wedge \underline{b}} \tag{3b}$$

The same complement is applied to both arguments first, the resulting expressions are combined by the wedge product, and afterwards the result uses the inverse complement operation to transform the outcome of the wedge product into the final result of the regressive product. The inverted wedge symbol $(\vee)$ is used for the regressive wedge product. Both definitions (3a) and (3b) are equivalent, but (3a) is used within the library to derive expressions for regressive products.

There are some additional formulas for complements which are useful for working with GA expressions and thus are provided here for reference. Taking the right and left complements of (3a) and (3b) we end up with

$$\overline{a \vee b} = \overline{\underline{\bar{a} \wedge \bar{b}}} = \bar{a} \wedge \bar{b} \tag{4a}$$
$$\underline{a \vee b} = \underline{\underline{a} \wedge \underline{b}} = \underline{a} \wedge \underline{b} \tag{4b}$$

Substituting $a = \underline{u}$ and $b = \underline{v}$ into (4a) and $a = \bar{u}$ and $b = \bar{v}$ into (4b) we end up with

$$u \wedge v = \overline{\underline{u} \vee \underline{v}} \tag{5a}$$
$$u \wedge v = \underline{\bar{u} \vee \bar{v}} \tag{5b}$$

This shows that the outer product and the regressive outer product can be expressed by each other.

The complement operation is used to define a unique dualization operation that works for cases when the metric is non-degenerate and even for cases when it is degenerate, like in projective geometric algebra PGA. This is the main advantage compared to the usual definition used by GA practitioners by multiplying the object with the pseudoscalar (or its inverse), since this operation breaks down for spaces with degenerate metrics. The right dual $A^\star$ is defined as $A^\star = \overline{GA}$, where A is an arbitrary multivector and G the extended metric and GA a matrix-vector-product. There is also a corresponding left dualization operation defined as $A_\star = \underline{GA}$. $\star$ is used as dualization operator (Hodge star). Relations to the geometric product are $A^\star = A^\dagger \vee I_n$ and $A_\star = I_n \vee A^\dagger$, where $\dagger$ is the reversion operation.

Following bi-argument products $op(a, b) = a\ op\ b$ and their respective regressive variants $rop(a, b) = \bar{a}\ \underline{op}\ \bar{b}$ are implemented (with `op(a,b)` or `rop(a,b)` as the name of the implemented function). Using A vs. a implies $gr(A) \geq gr(a)$, where $gr(arg)$ returns the grade of the argument arg:

- inner product, delivering a scalar: $a \bullet b$, `dot(a,b)`
- outer product of a $j-$ and $k-$vector, delivering a j+k-vector in $span(a, b)$: $a \wedge b$, `wdg(a,b)`
- geometric product: $a \vee b$, `operator*(a,b)`
- commutator product (the asymmetric part of the geometric product): $cmt(a, b) = \frac{1}{2}(a \vee b - b \vee a)$, `cmt(a,b)`
- regressive inner product, delivering a pseudoscalar: $a \circ b$, `rdot(a,b)`
- regressive outer product: $a \vee b$, `rwdg(a,b)`
- regressive geometric product: $a \barwedge b$, `rgpr(a,b)`
- left (bulk) contraction: $a \lfloor B = a \ll B = a_\star \vee B$, `operator<<(a,B)`, `lbulk_contract(a,B)`
- right (bulk) contraction: $B \lfloor a = B \gg a = B \vee a^\star$, `operator>>(B,a)`, `rbulk_contract(B,a)`
- left weight contraction: $a_\star \vee B$, `lweight_contract(a,B)`
- right weight contraction: $B \vee a^\star$, `rweight_contract(B,a)`
- left bulk expansion: $A_\star \wedge b$, `lbulk_expand(A,b)`

- right bulk expansion: $a \wedge B^{\star}$, `rbulk_expand(a,B)`
- left weight expansion: $A_{\star} \wedge b$, `lweight_expand(A,b)`
- right bulk expansion: $a \wedge B^{*}$, `rweight_expand(a,B)`

where the items from the left contraction downwards are all considered interior products. They operate on subspaces after applying a dualization operation to one operand.

TODO: *fill in further introductory notes here*

1.2 Basic formulas

The following is provided for Euclidean algebra of two-dimensional space (*ega2d*) using an orthormal basis $\boldsymbol{e}_1$, $\boldsymbol{e}_2$:

$$(\boldsymbol{e}_1)^2 = \boldsymbol{e}_1^2 = +1, \text{ and } (\boldsymbol{e}_2)^2 = \boldsymbol{e}_2^2 = +1 \tag{6a}$$

$$s_{2d} = s\mathbf{1} \tag{6b}$$

$$v_{2d} = v_1\boldsymbol{e}_1 + v_2\boldsymbol{e}_2 \tag{6c}$$

$$ps_{2d} = ps\boldsymbol{e}_{12} = ps\mathbb{1} \tag{6d}$$

equation (6b) is the scalar part s_{2d}, equation (6c) contains the vector part v_{2d}, and equation (6d) contains the pseudoscalar part ps_{2d}. The index $2d$ is typically omitted when clear from context. The basis elements are $\{\mathbf{1}, \boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_{12}\}$. Using these components a multivector M of $2d$ space is defined as

$$M = s\mathbf{1} + v_1\boldsymbol{e}_1 + v_2\boldsymbol{e}_2 + ps\boldsymbol{e}_{12} \tag{7}$$

where equation (7) contains three parts: the scalar part $s\mathbf{1}$ (basis element is the scalar $\mathbf{1}$; if $\mathbf{1}$ is not shown, it is implicitly assumed for scalar values), the vector part v (basis elements $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$) and the pseudoscalar part $ps\boldsymbol{e}_{12} = ps\mathbb{1}$ (basis element is $\boldsymbol{e}_{12}$, which is sometimes written as $\mathbb{1}$ to show its character as pseudoscalar of this space. It is a bivector in $2d$-Euclidean space).

For Euclidean algebra of three-dimensional space (*ega3d*) using an orthonormal basis $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_3$ there are:

$$(\boldsymbol{e}_1)^2 = \boldsymbol{e}_1^2 = +1, (\boldsymbol{e}_2)^2 = \boldsymbol{e}_2^2 = +1, \text{ and } (\boldsymbol{e}_3)^2 = \boldsymbol{e}_3^2 = +1 \tag{8a}$$

$$s_{3d} = s\mathbf{1} \tag{8b}$$

$$v_{3d} = v_1\boldsymbol{e}_1 + v_2\boldsymbol{e}_2 + v_3\boldsymbol{e}_3 \tag{8c}$$

$$b_{3d} = b_1\boldsymbol{e}_{23} + b_2\boldsymbol{e}_{31} + b_3\boldsymbol{e}_{12} \tag{8d}$$

$$ps_{3d} = ps\boldsymbol{e}_{123} = ps\mathbb{1} \tag{8e}$$

equation (8b) is the scalar part s_{3d}, equation (8c) is the vector part v_{3d}, equation (8d) is the bivector part b_{3d}, and equation (8e) is the pseudoscalar part ps_{3d}. The index $3d$ is typically omitted when clear from context. Comparing to the $2d$-case it becomes obvious, that all parts depend on context, specifically on the dimensionality of the modeled space, and thus need to be defined and treated accordingly (*hint*: since C++ is a statically typed language those types need to be well-defined and distinguishable from each other). The basis elements are $\{\mathbf{1}, \boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_{23}, \boldsymbol{e}_{31}, \boldsymbol{e}_{12}, \boldsymbol{e}_{123}\}$. Using these components a

multivector M of $3d$ space is defined as

$$M = s\mathbf{1} + v_1\boldsymbol{e}_1 + v_2\boldsymbol{e}_2 + v_3\boldsymbol{e}_3 + b_1\boldsymbol{e}_{23} + b_2\boldsymbol{e}_{31} + b_3\boldsymbol{e}_{12} + ps\boldsymbol{e}_{123} \tag{9}$$

where equation (9) contains four parts: the scalar component $s\mathbf{1}$ (basis element is the scalar $\mathbf{1}$, the vector part v (basis elements $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$), the bivector part b (basis elements $\boldsymbol{e}_{23}$, $\boldsymbol{e}_{31}$ and $\boldsymbol{e}_{12}$) and the pseudoscalar part $ps\boldsymbol{e}_{123} = ps\mathbb{1}$ (basis element is $\boldsymbol{e}_{123}$, which is sometimes written as $\mathbb{1}$ to show its character as pseudoscalar of this space. It is a trivector in $3d$-Euclidean space).

TODO: *fill in pga2d and pga3d definitions here*

The inner product between multivectors A and B is defined as

$$A \bullet B = A^T G B \tag{10}$$

with G as the extended metric and matrix multiplication on the right hand side. It satisfies the identity

$$A \bullet B = \langle A \wedge \tilde{B} \rangle_0 = \langle B \wedge \tilde{A} \rangle_0 \tag{11}$$

where $\wedge$ stands for the geometric product within the grade projection operator $\langle\rangle_k$ for grade k. The norm of a multivector A is defined as

$$\|A\| = \sqrt{A \bullet A} \tag{12}$$

with a positive argument under the square root due to the definition of the inner product.

The contraction is explicitly defined as

$$A \rfloor B = A \ll B = A_\star \vee B \tag{13a}$$
$$B \lfloor A = B \gg A = B \vee A^\star \tag{13b}$$

with $\star$ denoting the Hodge dual which is formed by the respective complement operation (in spaces of even dimension the left or right complement respectively, and in spaces of odd dimension the complement function regardless of the side of the operand). For blades A and B, they satisfy

$$A \rfloor B = A \ll B = \langle B \wedge \tilde{A} \rangle_{gr(B)-gr(A)} \tag{14a}$$
$$B \lfloor A = B \gg A = \langle \tilde{A} \wedge B \rangle_{gr(B)-gr(A)} \tag{14b}$$

and fulfill the requirement that $A \rfloor B = A \lfloor B = A \bullet B$ whenever A and B have the same grade. In this case the contractions reduce to the inner product.

The geometric product between and vector v and a blade B is defined in terms of interior and exterior products, i.e. the right contraction $\lfloor$, the left contraction $\rfloor$ and wedge product $\wedge$ as follows:

$$a \barwedge B = B \lfloor a + a \wedge B \tag{15a}$$
$$B \barwedge a = a \rfloor B + B \wedge a \tag{15b}$$

or sometimes using the right shift ($\gg$) or left shift ($\ll$) operators directly

$$a \barwedge B = B \gg a + a \wedge B \tag{16a}$$
$$B \barwedge a = a \ll B + B \wedge a \tag{16b}$$

as an alternative.

Every vector a can be decomposed into parts $a_\parallel$ parallel to the k-blade B and $a_\perp$ perpendicular to B, such that $a = a_\parallel + a_\perp$. For equation (16a): $B \gg a$ is a $(k-1)$-blade. If $a_\parallel \neq 0$ then $B \gg a$ represents a subspace of B. $a \wedge B$ is a $(k+1)$-blade. If $a_\perp \neq 0$ then $a \wedge B$ represents $span(a, B)$.
For arguments of equal grade the contractions reduce to the dot product $\bullet$, so that one can write specifically for two vectors a and b:

$$a \barwedge b = a \bullet b + a \wedge b \tag{17}$$

TODO: *fill in further basic formulas here*

Text with a norm $\|x\|$ and an indexed norm as $\|u\|_\bullet$ and $\|u\|_\circ$.

$\boldsymbol{e}_1$, $\boldsymbol{e}_1$, $\boldsymbol{e}_{123}$, $\bigstar$, $\star$, $\bigstar$

1.3 Physics modelling

The key idea is to represent physical entities and statements using objects that are represented as directly as possible in the underlying algebra and have a geometric interpretation.

EGA is limited to objects that contain the origin. It can be used to directly model vectors (i.e. directions), lines and planes through the orgin. In addition, it can model complex numbers, quaternions and thus rotations in $2d$ and $3d$ well. To realize them it uses sandwich products for calculating the transformation resulting from two consequtive reflections either across lines ($2d$-case) or across planes ($3d$-case). The resulting transformations are rotations. The advantage of using geometric algebra to model those rotations is the intuitive modelling, due to geometric meaning of the objects used and the fact that the sandwich products can be used to transform different objects using the same formulas regardless of the type of object to be transformed.

In additon, *PGA* is capable to model objects that do not contain the origin. Thus it can be used to directly model points, directions, lines, and planes that may or may not contain the origin. The additional degree of freedom is possible due to embedding the Euclidean space of dimension n into a projective space of dimension $n + 1$, e.g. $2d$ Euclidean algebra is modelled by embedding it in a $3d$ projective space, and $3d$ Euclidean algebra is modelled in a $4d$ projective space. In a projective space the objects must be projected into the modelled space. Thus the objects of the embedding space typically have one dimension more than the corresponding objects in modelled space. E.g. a projective point (a scalar or $0d$ object) is a vector in the embedding space (a $1d$ object), and a bivector in projective space (a $2d$ object) is used to model a line (a $1d$ object). The projection reduces the object dimension by one. Projective objects can be multiplied by a scalar without changing the geometrical meaninig of the projective object in the embedding space.

Due to its extended expressiveness *PGA* will be used to model physics of mass points and rigid bodies. *PGA* will be applied to model mass points, as well as velocity, acceleration, force, torque, momentum, angular momentum, etc. to describe kinematics and dynamics of mass points and rigid bodies. The approach used here is inspired by [Dorst and De Keninck, 2022] and [Dorst and De Keninck, 2023], but it does not use plane-based *PGA* with planes as vectors, which are based on dual representations of the objects to be modelled. It rather uses the approach as described by [Lengyel, 2024b] using projective geometric algebra based on direct representations of points,

lines and planes. The reasons for choosing the latter approach are explained in [Lengyel, 2024a]. Further hints on inconsistencies in GA application up-to-now can be found as well in [Kritchevsky, 2024]. The goal of the this work is to be as consistent in terms of mathematical application of GA and as intuitively applicable as possible.

1.4 Literature

References

[Browne, 2012] Browne, J. (2012). *Grassmann Algebra*, volume 1: Foundations. Barnard Publishing.

[Dorst and De Keninck, 2023] Dorst, L. and De Keninck, S. (August 15, 2023). May the forque be with you - dynamics in pga. *University of Amsterdam, The Netherlands*, V2.6.

[Dorst and De Keninck, 2022] Dorst, L. and De Keninck, S. (March 14, 2022). A guided tour to the plane-based geometric algebra pga. *University of Amsterdam, The Netherlands*, V2.0.

[Kritchevsky, 2024] Kritchevsky, A. (February 28, 2024). The case against geometric algebra. *Alex Kritchevsky's blog*.

[Lengyel, 2024b] Lengyel, E. (2024b). *Projective Geometric Algebra Illuminated*. Terathon Software LLC.

[Lengyel, 2024a] Lengyel, E. (August 23, 2024a). Poor Foundations in Geometric Algebra. *Eric Lengyel's blog*.