Foundations of differential geometric algebra

Michael Reed (Crucial Flow Research), draft 2021

Abstract. The abbreviated foundations developed in the previous paper "Differential geometric algebra using Leibniz, Grassmann" can be used as a universal language for finite element methods based on a discrete manifold bundle, so an elaborated foundation is desired.

Tools built on these foundations enable computations based on multi-linear algebra and spin groups using the geometric algebra known as Grassmann algebra or Clifford algebra. This foundation is built on a direct-sum parametric type system for tangent bundles, vector spaces, and also projective and differential geometry. Geometric algebra is a mathematical foundation for differential geometry, which can be used to simplify the Maxwell equations to a single wave equation due to the geometric product. Introduction of geometric algebra to engineering science disciplines will be easier with programmable foundations.

In order to devise an expressive and performance oriented language for efficient discrete differential geometric algebra with the Grassmann elements, an efficient computer algebra representation was programmed. With this unifying mathematical foundation, it is possible to improve efficiency of multi-disciplinary research using geometric tensor calculus by relying on universal mathematical principles. Tools built on universal differential geometric algebra provide a natural geometric language for the Helmholtz decomposition and Hodge-DeRahm co/homology.

Using the new *Grassmann.jl* package, it is possible to compute anti-symmetric tensor products and geometric algebra in Julia's high performance computing contexts. Abstract nature of the product algebra code generation enables extension of the product operations by way of Julia's type system, with which it is possible to construct customized mixed tensor products from a vector basis along with dyadics, as well as bivector elements of Lie groups. Given this mathemaical foundation, finite element methods based on discrete manifolds can be combined with unified software tools to provide the universal foundations for complex numbers (flatland geometry), quaternions (rotation in 3D), linear algebra & bilinear, multilinear determinant, exterior forms algebra, quantum logic lattice, differential geometry, finite element methods, conformal / projection, group representation, combinatoric hypergraphs.

The *Grassmann.jl* package provides tools for computations based on multi-linear algebra and spin groups using the extended geometric algebra known as Leibniz-Grassmann-Clifford-Hestenes algebra. Combinatorial products include exterior, regressive, inner, and geometric; along with the Hodge star, adjoint, reversal, and boundary operators. The kernelized operations are built up from composite sparse tensor products and Hodge duality, with high dimensional support for up to 62 indices using staged caching and precompilation. Code generation enables concise yet highly extensible definitions. *DirectSum.jl* multivector parametric type polymorphism is based on tangent vector spaces and conformal projective geometry. Additionally, the universal interoperability between different sub-algebras is enabled by *AbstractTensors.jl*, on which the type system is built.

- **DirectSum.jl**: Abstract tangent bundle vector space types (unions, intersections, sums, etc.)
- **AbstractTensors.jl**: Tensor algebra abstract type interoperability with vector bundle parameter
- **Grassmann.jl**: ⟨Leibniz+Grassmann-Clifford-Hestenes⟩ differential geometric algebra of multivector forms
- **Leibniz.jl**: Derivation operator algebras for tensor fields
- **Reduce.jl**: Symbolic parser generator for Julia expressions using REDUCE algebra term rewriter

Mathematical foundations and some of the nuances in the definitions specific to the *Grassmann.jl* implementation are concisely described, along with the accompanying support packages that provide an extensible platform for computing with geometric algebra at high dimensions. The design is based on the `TensorAlgebra` abstract type interoperability from *AbstractTensors.jl* with a `VectorBundle` type parameter from *DirectSum.jl*. Abstract vector space type operations happen at compile-time, resulting in a differential conformal geometric algebra of hyper-dual multivector forms.

The nature of the geometric algebra code generation enables one to easily extend the abstract product operations to any specific number field type (including differential operators with *Leibniz.jl* or symbolic coefficients with *Reduce.jl*), by making use of Julia's type system. Mixed tensor products with their coefficients are constructed from these operations to work with bivector elements of Lie groups [15][24].

1. Direct sum parametric type polymorphism of Grassmann

The *DirectSum.jl* package is a work in progress providing the necessary tools to work with an arbitrary `Manifold` specified by an encoding. Due to the parametric type system for the generating `VectorBundle`, the Julia compiler can fully pre-allocate and often cache values efficiently ahead of run-time. Although intended for use with the *Grassmann.jl* package, `DirectSum` can be used independently.

Definition 1 (Vector space $\Lambda^1 V = V$ is a field's $\mathbb{K}$-module instance). Let V be a $\mathbb{K}$-module (abelian group with respect to $+$) with an element $1 \in \mathbb{K}$ such that $1V = V$ by scalar multiplication $\mathbb{K} \times V \to V$ over field $\mathbb{K}$ satisfying

- $a(x + y) = ax + ay$ distribution of vector addition,
- $(a + b)x = ax + bd$ distribution of field addition,
- $(ab)x = a(bx)$ associative compatibility.

In the software package `Grassmann`, an underlying generating vector space is also synonymous with the term `<:TensorBundle` (an abstract type).

The `AbstractTensors` package is intended for universal interoperability of the abstract `TensorAlgebra` type system. All `TensorAlgebra{V}` subtypes have type parameter *V*, used to store a `TensorBundle` value from *DirectSum.jl*. By itself, this package does not impose any specifications or structure on the `TensorAlgebra{V}` subtypes and elements, aside from requiring V to be a `TensorBundle`. This means that different packages can create tensor types having a common underlying `TensorBundle` structure. For example, this is mainly used in *Grassmann.jl* to define various `SubAlgebra`, `TensorTerm` and `TensorMixed` types, each with subtypes. Externalizing the abstract type helps extend the dispatch to other packages.

The key to making the interoperability work is that each `TensorAlgebra` subtype shares a `TensorBundle` parameter (with all `isbitstype` parameters), which contains all the info needed at compile time to make decisions about conversions. So other packages need only use the vector space information to decide on how to convert based on the implementation of a type. If external methods are needed, they can be loaded by `Requires` when making a separate package with `TensorAlgebra` interoperability.

Additionally, a universal unit volume element can be specified in terms of `LinearAlgebra.UniformScaling`, which is independent of V and has its interpretation only instantiated by the context of the `TensorAlgebra{V}` element being operated on. Universal interoperability of `LinearAlgebra.UniformScaling` as a pseudoscalar element which takes on the `TensorBundle` form of any other `TensorAlgebra` element is handled globally. This enables the usage of `I` from `LinearAlgebra` as a universal pseudoscalar element defined at every point x of a `Manifold`, which is mathematically denoted by $I = I(x)$ and specified by the $g(x)$ bilinear tensor field of TM. Utility methods such as `scalar, involute, norm, norm2, unit, even, odd` are also defined.

Definition 2 (Linear dependence). Let V be a vector space over field $\mathbb{K}$, then the set $\{v_i\}_i$ is linearly dependent iff $\sum_{i=1}^{n} k_i v_i = 0$ for some $0 \neq k \in \mathbb{K}^n$.

Definition 3 ($\wedge$-product annihilation). For a linearly dependent set $\{v_i\}_1^n \subset V$

$$v_1 \wedge v_2 \wedge \cdots \wedge v_n = 0.$$

Initially, it is enough to understand that $\wedge : \Lambda^n V \times \Lambda^m V \to \Lambda^{n+m} V$ is an operation which is zero for linearly dependent arguments. However, this idea comes from extending Grassmann's product $v_i \wedge v_j = -v_j \wedge v_i \implies v_i \wedge v_i = 0 = -v_i \wedge v_i$ to yield a tool for characterizing linear dependence.

Definition 4 (Dimension n-SubManifold in $\Lambda^n V$). Hence, writing the product $v_1 \wedge v_2 \wedge \cdots \wedge v_n \neq 0$ implies a linearly independent set $\{v_i\}_1^n \subseteq V$ isomorphic to an n-`SubManifold`. With the product $\Lambda^0 \Lambda^n V \times (v_1 \wedge v_2 \wedge \cdots \wedge v_n) \cong \mathbb{K}$ it is also clear that a 1-dimensional basis subspace is induced by any n-`SubManifold`.

Example 1. Therefore, $\mathbb{K} = \Lambda^0 \mathbb{K} \cong \Lambda^1 \mathbb{K}$ is a vector space or a 0-SubManifold.

Example 2. $\Lambda^n V$ is a vector space with $\Lambda^1 \Lambda^n V = \Lambda^n V$ and $\Lambda^0 \Lambda^n V = \Lambda^0 V$.

Denote $V^* = V \setminus \{0\}$ as the set V excluding the 0 element in next:

Definition 5 (Direct sum $\oplus$). To consider a set of linearly independent spaces, let $\pi_i : V \to V_i$ be projections with vector space $V_i \subset V$, define

$$V_1 \oplus V_2 \oplus \cdots \oplus V_n = V \iff \bigwedge : V_1^* \times V_2^* \times \cdots \times V_n^* \to \Lambda^n V^*.$$

DirectSum of a full non-zero product implies an n-SubManifold.

Definition 6. Grade-m projection is defined as $\langle \Lambda V \rangle_m = \Lambda^m V$ such that

$$\Lambda V = \bigoplus_{m=0}^{n} \langle \Lambda V \rangle_m = \Lambda^0 V \oplus \Lambda^1 V \oplus \cdots \oplus \Lambda^n V, \qquad \langle \Lambda V \rangle_m = \bigoplus_{m=1}^{\binom{n}{m}} \mathbb{K}.$$

Note that $\dim \langle \Lambda V \rangle_m = \binom{n}{m}$ and hence $\dim \Lambda V = \sum_{m=0}^{n} \binom{n}{m} = 2^n$.

Example 3 (Combinatorics of $\mathcal{P}(V)$ and hypergraphs $\subseteq \mathcal{P}(V) \setminus \{\emptyset\}$). Let $v_1, v_2, v_3 \in \mathbb{R}^3$, then the power set of elements is:

$$\mathcal{P}(\mathbb{R}^3) = \{\emptyset, \{v_1\}, \{v_2\}, \{v_3\}, \{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_1, v_2, v_3\}\}$$

Form a direct sum over the elements of $\mathcal{P}(V)$ with $\wedge$ to define ΛV, e.g.

$$\Lambda(\mathbb{R}^3) = \Lambda^0(\mathbb{R}^3) \oplus \Lambda^1(\mathbb{R}^3) \oplus \Lambda^2(\mathbb{R}^3) \oplus \Lambda^3(\mathbb{R}^3)$$

$$\overbrace{\hat{v_\emptyset}}^{\Lambda^0 \mathbb{R}} \oplus \overbrace{v_1 \oplus v_2 \oplus v_3}^{\Lambda^1(\mathbb{R}^3)} \oplus \overbrace{(v_1 \wedge v_2) \oplus (v_1 \wedge v_3) \oplus (v_2 \wedge v_3)}^{\Lambda^2(\mathbb{R}^3)} \oplus \overbrace{(v_1 \wedge v_2 \wedge v_3)}^{\Lambda^3(\mathbb{R}^3)}$$

Definition 7 (Vector bundle of SubManifold). Let $M = T^\mu V \in \mathrm{Vect}_{\mathbb{K}}$ be a `TensorBundle<:Manifold` of rank n,

$$T^\mu V = (n, \mathbb{P}, g, \nu, \mu), \qquad \mathbb{P} \subseteq \langle v_\infty, v_\emptyset \rangle, \qquad g : V \times V \to \mathbb{K}$$

The type `TensorBundle{n,`$\mathbb{P}$`,g,`ν, μ`}` uses *byte-encoded* data available at pre-compilation, where $\mathbb{P}$ specifies the basis for up and down projection, g is a bilinear form that specifies the metric of the space, and μ is an integer specifying the order of the tangent bundle (i.e. multiplicity limit of Leibniz-Taylor monomials). Lastly, ν is the number of tangent variables. The dual space functor $(\cdot)' : \mathrm{Vect}_{\mathbb{K}}^{\mathrm{op}} \to \mathrm{Vect}_{\mathbb{K}}$ is an involution which toggles a dual vector space with inverted signature with property $V' = \mathrm{Hom}(V, \mathbb{K})$ and having `Basis` generators

$$\langle v_1, \ldots, v_{n-\nu}, \partial_1, \ldots, \partial_\nu \rangle = M \leftrightarrow M' = \langle w_1, \ldots, w_{n-\nu}, \epsilon_1, \ldots, \epsilon_\nu \rangle$$

where v_i, w_i are a basis for the vectors and covectors, while ∂_j, ϵ_j are a basis for differential operators and tensor fields.

Example 4 (Case of 3rd order tangent bundle operator composition).

$T^3(\mathbb{R}^0) = \partial_\emptyset \oplus \partial_1 \oplus \partial_2 \oplus \partial_3 \oplus (\partial_1 \circ \partial_2) \oplus (\partial_1 \circ \partial_3) \oplus (\partial_2 \circ \partial_3) \oplus (\partial_1 \circ \partial_2 \circ \partial_3)$

In order to shorten the notation, the operation symbol is left out:

$$\{v_1, v_2, v_3, v_{12}, v_{13}, v_{23}, v_{123}\}, \{\partial_1, \partial_2, \partial_3, \partial_{12}, \partial_{13}, \partial_{23}, \partial_{123}\}$$

The direct sum operator $\oplus$ can be used to join spaces (alternatively $+$), and the dual space functor $'$ is an involution which toggles a dual vector space with inverted signature. In addition to the direct-sum operation, several other operations are supported, such as $\cup, \cap, \subseteq, \supseteq$ for set operations. Due to the design of the `TensorBundle` dispatch, these operations enable code optimizations at compile-time provided by the bit parameters.

$$\bigcup T^{\mu_i} V_i = \left(|\mathbb{P}| + \max\left\{n_i - |\mathbb{P}_i|\right\}_i, \bigcup \mathbb{P}_i, \cup g_i, \max\left\{\nu_i\right\}_i, \max\left\{\mu_i\right\}_i\right)$$

$$\bigoplus T^{\mu_i} V_i = \left(|\mathbb{P}| + \sum (n_i - |\mathbb{P}_i|), \bigcup \mathbb{P}_i, \oplus_i g_i, \max\left\{\nu_i\right\}_i, \max\left\{\mu_i\right\}_i\right)$$

These are roughly the formulas used for those operations. Note differences between the operations $\bigcup$ and $\bigoplus$, which are similar

Calling manifolds with sets of indices constructs the subspace representations. Given `M(s::Int...)` one can encode `SubManifold{length(s),M,s}` with induced orthogonal space Z, such that computing unions of submanifolds is done by inspecting the parameter $s \in V \subseteq W$ and $s \notin Z$. Here, calling a `Manifold` with a set of indices produces a `SubManifold` representation.

$$T^e V \subset T^\mu W \iff \exists Z \in \mathrm{Vect}_{\mathbb{K}}(T^e(V \oplus Z) = T^{e \leq \mu} W,\ V \perp Z).$$

Operations on `Manifold` types is automatically handled at compile time.

Definition 8 (Shirokov's permutations). Consider $\sigma_j(\omega) = \sum_{k=0}^{n} (-1)^{\binom{k}{2^{j-1}}} \langle\omega\rangle_k$,

$$\sigma_1(\omega) \equiv \overline{\omega}, \qquad \sigma_2(\omega) \equiv \tilde{\omega}, \qquad \sigma_{12} = \sigma_2(\sigma_1(\omega)) \equiv \tilde{\overline{\omega}}$$

Proposition 1 ($\mathfrak{S}_j = \langle \sigma_1, \sigma_2, \ldots, \sigma_j \rangle$ is a group). *Group $\mathfrak{S}_2 = \{1, \sigma_1, \sigma_2, \sigma_{12}\}$ is a set of automorphisms: grade involution $\overline{\omega} = \sigma_1(\omega) = \sum_{k=0}^{n} (-1)^{\binom{k}{1}} \langle\omega\rangle_k$, reverse $\tilde{\omega} = \sigma_2(\omega) = \sum_{k=0}^{n} (-1)^{\binom{k}{2}} \langle\omega\rangle_k = \sum_{k=0}^{n} (-1)^{(k-1)k/2} \langle\omega\rangle_k$ is an anti-automorphism with $\sigma_2(v_i \wedge v_j) = \sigma_2(v_j) \wedge \sigma_2(v_i)$, and Clifford conjugate $\tilde{\overline{\omega}}$ is the composition of grade involution and reverse anti-automorphism.*

Definition 9 (Real $\widetilde{\Re}\omega = (\omega + \tilde{\omega})/2$ and imaginary $\widetilde{\Im}\omega = (\omega - \tilde{\omega})/2$). Real and imaginary define $\mathbb{Z}_2$-grading projections such that $\Lambda V = \widetilde{\Re}\Lambda V \oplus \widetilde{\Im}\Lambda V$; where $\widetilde{\Re}\Lambda V$ is real part and $\widetilde{\Im}\Lambda V$ is imaginary part.

Definition 10 (Even $\overline{\Re}\omega = (\omega + \overline{\omega})/2$ and odd $\overline{\Im}\omega = (\omega - \overline{\omega})/2$). The projection $\overline{\Re}\Lambda V$ is even grade and $\overline{\Im}\Lambda V$ is odd grade with $\mathbb{Z}_2$-grading $\Lambda V = \overline{\Re}\Lambda V \oplus \overline{\Im}\Lambda V$.

$\mathbb{Z}_2$-grading projections: $\sigma_j(\Re)\omega = (\omega + \sigma_j(\omega))/2$ and $\sigma_j(\Im)\omega = (\omega - \sigma_j(\omega))/2$.

The direct sum of a `TensorBundle` and its dual $V \oplus V'$ represents the full mother space, which is yet another $\mathbb{Z}_2$-grading [23] of a vector space.

```
collect(V) # all SubManifold vector basis elements
collect(SubManifold(V')) # all covector basis elements
collect(SubManifold(V+V')) # all mixed basis elements
```

Since `TensorBundle` choices are fundamental to `TensorAlgebra` operations, the universal interoperability between `TensorAlgebra{V}` elements with different associated `TensorBundle` choices is naturally realized by applying the `union` morphism to operations, e.g. $\bigwedge : \Lambda^{p_1} V_1 \times \cdots \times \Lambda^{p_g} V_g \to \Lambda^{\sum_k p_k} \bigcup_k V_k$. Some of the method names like $+, -, *, \otimes, \circledast, \odot, \boxtimes, \star$ for `TensorAlgebra` elements are shared across different packages, with interoperability.

```
function op(::TensorAlgebra{V},::TensorAlgebra{V}) where V
    # well defined operations if V is shared
end # but what if V ≠ W in the input types?

function op(a::TensorAlgebra{V},b::TensorAlgebra{W}) where {V,W}
    VW = V ∪ W        # VectorSpace type union
    op(VW(a),VW(b))   # makes call well-defined
end # this option is automatic with interop(a,b)
```

Suppose we are dealing with a new subtype in another project. To define additional specialized interoperability for further methods, it is necessary to define dispatch that catches well-defined operations for equal `TensorBundle` choices and a fallback method for interoperability, along with a `TensorBundle` morphism. Thus, interoperability is a situation of defining one additional fallback method for the operation and also a new `TensorBundle` compatibility morphism, satisfying a union morphism law

```
op(a,b) |> Manifold == Manifold(a) ∪ Manifold(b)
b(a) |> Manifold == Manifold(a) ∪ Manifold(b)
```

The metric signature of the `SubManifold{V,1}` elements of a vector space V can be specified with the `V"..."` constructor by using + and - to specify whether the `SubManifold{V,1}` element of the corresponding index squares to +1 or -1. For example, `S"+++"` constructs a positive definite 3-dimensional `TensorBundle`, so constructors such as `S"..."` and `D"..."` are convenient.

```
julia> ℝ^3 == S"+++" == Manifold(3)
true
```

It is possible to specify an arbitrary `DiagonalForm` having numerical values for the basis including degeneracy `D"1,1,1,0"`, although the `Signature` format has a more compact representation if limited to +1 and -1. It is also possible to change the diagonal scaling, such as with `D"0.3,2.4,1"`. Further development will result in more metric types, including non-diagonal metric tensors.

Declaring an additional point at infinity is done by specifying it in the string constructor with ∞ at the first index (i.e. Riemann sphere `S"∞+++"`). The hyperbolic geometry can be declared by $\emptyset$ subsequently (i.e. hyperbolic projection `S"∅+++"`). Additionally, the *null-basis* based on the projective split for confromal geometric algebra would be specified with `∞∅` initially (i.e. 5D CGA `S"∞∅+++"`). These two declared basis elements are interpreted in the type system. The `tangent(V,μ,ν)` map can be used to specify μ and ν.

The index number n of the `TensorBundle` corresponds to the total number of generator elements. However, `V"∞∅+++"` is of type `TensorBundle{5,3}` with 5 generator elements, it can be internally recognized in the direct sum algebra as being an embedding of a 3-index `TensorBundle{3,0}` with additional encoding of the null-basis (origin and point at infinity) in the parameter $\mathbb{P}$ of the `TensorBundle{n,ℙ}` type. The `tangent` map takes V to its tangent space and can be applied repeatedly for higher orders, such that `tangent(V,μ,ν)` can be used to specify μ and ν.

```
julia> V = tangent(ℝ^3)
T¹(+++₁)

julia> tangent(V')
T²(----¹)'

julia> V+V'
T¹(+++---₁¹)*
```

The elements of the `Basis` can be generated in many ways using the `SubManifold` elements created by the `@basis` macro,

```
julia> using Grassmann; @basis ℝ'⊕ℝ^3 #' equivalent to basis"-+++"
((-+++), v, v1, v2, v3, v4, v12, v13, v14, v23, v24, v34, v123, v124, v134, v234, v1234)
```

The macro `@basis V` declares a local basis in Julia. As a result of this macro, all of the `SubManifold{V,G}` elements generated by that `TensorBundle` become available in the local workspace with the specified naming. The first argument provides signature specifications, the second argument is the variable name for the `TensorBundle`, and the third and fourth argument are prefixes of the `SubManifold` vector names (and covector basis names). By default, V is assigned the `TensorBundle` and v is the prefix for the `SubManifold` elements.

It is entirely possible to assign multiple different bases with different signatures without any problems. The `@basis` macro arguments are used to assign the vector space name to V and basis elements to v, but other assigned names can be chosen so that their local names do not interfere: Alternatively, if you do not wish to assign these variables to your local workspace, the versatile constructors of `DirectSum.Basis{V}` can be used to contain them, which is exported to the user as the method `Λ(V)`.

The Grassmann `SubManifold` elements $v_k \in \Lambda^1 V$ and $w^k \in \Lambda^1 V'$ are linearly independent vector and covector elements of V, while the Leibniz `Operator` elements $\partial_k \in L^1 V$ are partial tangent derivations and $\epsilon_k \in L^1 V'$ are dependent functions of the `tangent` manifold. An element of a mixed-symmetry `TensorAlgebra{V}` is a multilinear mapping that is formally constructed by taking the tensor products of linear and multilinear maps,

$$(\bigotimes_k \omega_k)(v_1, \dots, v_{\sum_k p_k}) = \prod_k \omega_k(v_1, \dots, v_{p_k}).$$

Higher `grade` elements correspond to `SubManifold` subspaces, while higher `order` function elements become homogenous polynomials and Taylor series.

To help provide a commonly shared and readable indexing to the user, some extended dual index print methods with full alphanumeric characters:

```
julia> DirectSum.printindices(stdout,DirectSum.indices(UInt(2^62-1)),false,"v")
v1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
julia> DirectSum.printindices(stdout,DirectSum.indices(UInt(2^62-1)),false,"w")
w1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
```

An application of this is in the `Grassmann` package, where dual indexing is used.

Definition 11 (Mixed-symmetry basis). Combining the linear basis generating elements with each other using the multilinear tensor product yields a graded (decomposable) `SubManifold` $\langle v_{p_1} \otimes \cdots \otimes v_{p_k} \rangle_k$, where `rank` k is determined by the sum of basis index multiplicities in the tensor product decomposition. The Grassmann anti-symmetric exterior basis is denoted by $v_{i_1 \dots i_g} \in \Lambda_g V$ with its dual $w^{i_1 \cdots i_g} \in \Lambda^g V$, while the Leibniz symmetric basis are $\partial_{i_1}^{\mu_1} \dots \partial_{i_g}^{\mu_g} \in L_g V$ with corresponding $\epsilon_{i_1}^{\mu_1} \dots \epsilon_{i_g}^{\mu_g} \in L^g V$ adjoint elements. The algebra partitions into symmetric and anti-symmetric tensor equivalence classes. For any pair,

$$\underbrace{\omega \otimes \eta = -\eta \otimes \omega}_{\text{anti-symmetric}} \qquad \text{or} \qquad \underbrace{\omega \otimes \eta = \eta \otimes \omega}_{\text{symmetric}}.$$

Combined, this space produces the full Leibniz tangent algebra and the Grassmann exterior algebra with 2^n elements. The mixed index algebra is partitioned into both symmetric and anti-symmetric tensor equivalence classes. Any mixed tensor `SubManifold` pair ω, η satisfies either

```
julia> Λ(ℝ^3)
DirectSum.Basis{(+++),8}(v, v1, v2, v3, v12, v13, v23, v123)

julia> Λ(tangent(ℝ^2))
DirectSum.Basis{T¹(++₁),8}(v, v1, v2, ∂₁, v12, ∂₁v1, ∂₁v2, ∂₁v12)

julia> Λ(tangent((ℝ^0)',3,3))
DirectSum.Basis{T³(123)',8}(w, ϵ₁,ϵ₂ ,ϵ₃ ,ϵ₁₂ ,ϵ₁₃ ,ϵ₂₃ ,ϵ₁₂₃ )
```

Typically the k in a product $\left(\partial_{p_1} \otimes \cdots \otimes \partial_{p_k}\right)^{(k)}$ is referred to as the `order` of the element if it is fully symmetric, which is overall tracked separately from the `grade` such that $\partial_k \left\langle w_j \right\rangle_r = \left\langle \partial_k w_j \right\rangle_r$ and $(\partial_k)^{(r)} \omega_j = (\partial_k w_j)^{(r)}$.

A higher-order composite tensor element is an oriented-multi-set X such that $v_X = \bigotimes_k v_{i_k}^{\otimes \mu_k}$ with the indices $X = \left((i_1, \mu_1), \ldots, (i_g, \mu_g)\right)$ and $|X| = \sum_k \mu_k$ is tensor `rank`. Anti-symmetric indices $\Lambda X \subseteq \Lambda V$ have two orientations and higher multiplicities of them result in zero values, so the only interesting multiplicity is $\mu_k \equiv 1$. The Leibniz-Taylor algebra is a quotient polynomial ring $LV \cong R[x_1, \ldots, x_n] / \{\prod_{k=1}^{\mu+1} x_{p_k}\}$ so that $\partial_k^{\mu+1}$ is zero. Typically the k in a product $\left(\partial_{p_1} \otimes \cdots \otimes \partial_{p_k}\right)^{(k)}$ is referred to as the `order` of the element if it is fully symmetric, which is overall tracked separately from the `grade` such that $\partial_k \langle v_j \rangle_r = \langle \partial_k v_j \rangle_r$ and $(\partial_k)^{(r)} \omega_j = (\partial_k v_j)^{(r)}$.

Grassmann's exterior product is an anti-symmetric tensor product

$$v_i \wedge v_j = -v_j \wedge v_i \implies v_i \wedge v_i = 0 = -v_i \wedge v_i,$$

which generalizes the multilinear determinant transposition property

$$v_{\omega_1} \wedge \cdots \wedge v_{\omega_m} \wedge v_{\eta_1} \wedge \cdots \wedge v_{\eta_n} = (-1)^{mn} v_{\eta_1} \wedge \cdots \wedge v_{\eta_n} \wedge v_{\omega_1} \wedge \cdots \wedge v_{\omega_m}.$$

Hence for graded elements it is possible to deduce that

$$\omega \in \Lambda^m V, \eta \in \Lambda^n V : \qquad \omega \wedge \eta = (-1)^{mn} \eta \wedge \omega.$$

Remark. Observe that the anti-symmetric property implies that $\omega \otimes \omega = 0$, while the symmetric property neither implies nor denies such a property. Grassmann remarked [13] in 1862 that the symmetric algebra of functions is by far more complicated than his anti-symmetric exterior algebra. The first part of the book focused on anti-symmetric exterior algebra, while the more complex symmetric function algebra of Leibniz was subject of the second multivariable part of the book. Elements ω_k in the space ΛV of anti-symmetric algebra are often studied as unit quantum state vectors in a unitary probability space, where $\sum_k \omega_k \neq \bigotimes_k \omega_k$ is entanglement.

```
julia> indices(Λ(3).v12)
2-element Array{Int64,1}:
 1
 2
```

Grassmann's exterior algebra doesn't invoke the properties of multi-sets, as it is related to the algebra of oriented sets; while the Leibniz symmetric algebra is that of unoriented multi-sets. Combined, the mixed-symmetry algebra yields a multi-linear propositional lattice. The formal sum of equal `grade` elements is an oriented `Chain` and with mixed `grade` it is a `MultiVector` simplicial complex. Thus, various standard operations on the oriented multi-sets are possible including $\cup, \cap, \oplus$ and the index operation $X \ominus Y = (X \cup Y) \backslash (X \cap Y)$, which is symmetric difference operation $\veebar$.

```
julia> typeof(V) # dispatch by vector space
SubManifold{(-+++),4,0x0000000000000000f}

julia> typeof(v13) # extensive type info
SubManifold{(-+++),2,0x0000000000000005}

julia> 2v1 + v3 # vector Chain{V,1} element
2v1 + 0v2 + 1v3 + 0v4

julia> 5 + v2 + v234 # MultiVector{V} element
5 + 1v2 + 1v234
```

Definition 12 (Simplex{V,G,B,T}, e.g. $5v_1, 7v_{34}, 8v_{123}$). Value of type T associated with a grade G Submanifold B of V element of $\Lambda^G V$.

```
struct Simplex{V,G,B,T} <: TensorTerm{V,G}
    v::T end
```

Definition 13 (Chain{V,G,T}, e.g. $3v_1 + 4v_2$ or $1v_{12} + 1v_{13} + 1v_{23}$). Grade G subspaces of dimension $\binom{n}{G}$ with type T coefficients are elements of $\Lambda^G V$.

```
@computed struct Chain{V,G,K} <: TensorGraded{V,G}
    v::Values{binomial(mdims(V),G),K} end
```

Definition 14 (MultiVector{V,T}). Full 2^n dimensional representation of ΛV over type T are a generalization of (mixed) oriented hypergraphs.

```
@computed struct MultiVector{V,K} <: TensorMixed{V}
    v::Values{1<<mdims(V),K} end
```

The parametric type formalism in `Grassmann` syntax design is highly expressive to enable the pre-allocation of geometric algebra computations for specific sparse-subalgebras, including the representation of rotational groups, Lie bivector algebras, and affine projective geometry.

In order to work with a `TensorAlgebra{V}`, it is necessary for some computations to be cached. This is usually done automatically when accessed. Staging of precompilation and caching is designed so that a user can smoothly transition between very high dimensional and low dimensional algebras in a single session, with varying levels of extra caching and optimizations.

It is possible to reach `Simplex` elements with up to $n = 62$ vertices, requiring full alpha-numeric labeling with lower-case and capital letters.

```
julia> Λ(62).v32a87Ng
-1v2378agN
```

The 62 indices require full alpha-numeric labeling with lower-case and capital letters. This now allows users to reach up to $4,611,686,018,427,387,904$ dimensions with Julia `using Grassmann`. Then the volume element is

```
v1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Full `MultiVector` allocations are only possible for $N \leq 22$, sparse operations are available at higher dimensions. While `DirectSum.Basis{V}` is a container for the `TensorAlgebra` generators of V, the `Basis` is only cached for $N \leq 8$. For the range of dimensions $8 < N \leq 22$, the `SparseBasis` type is used.

```
julia> Λ(22)
DirectSum.SparseBasis{(++++++++++++++++++++++),4194304}(v, ..., v1234567890abcdefghijkl)
```

This is the largest `SparseBasis` that can be generated with Julia, due to array size limitations.

To reach higher dimensions with $N > 22$, the `DirectSum.ExtendedBasis` type is used. It is suficient to work with a 64-bit representation (which is the default). And it turns out that with 62 standard keyboard characters, this fits. At 22 dimensions and lower there is better caching, with further extra caching for 8 dimensions or less. Thus, the largest Hilbert space that is fully reachable has 4,194,304 dimensions, but we can still reach out to 4,611,686,018,427,387,904 dimensions with the `ExtendedBasis` built in. It is still feasible to extend to a further super-extended 128-bit representation using the `UInt128` type (but this will require further modifications of internals and helper functions). To reach into infinity even further, it is theoretically possible to construct ultra-extensions also using dictionaries. Full `MultiVector` elements are not representable when `ExtendedBasis` is used, but the performance of the `Basis` and sparse elements should be just as fast as for lower dimensions for the current `SubAlgebra` and `TensorAlgebra` types. The sparse representations are a work in progress to be improved with time.

2. Geometric algebraic product structure

For the oriented sets of the Grassmann exterior algebra, the parity of $(-1)^{\Pi}$ is factored into transposition compositions when interchanging ordering of the tensor product argument permutations [3]. The symmetrical algebra does not need to track this parity, but has higher multiplicities in its indices. Symmetric differential function algebra of Leibniz trivializes orientation into a single class of index multi-sets, while Grassmann's exterior algebra is partitioned into two oriented equivalence classes by anti-symmetry. Full tensor algebra can be sub-partitioned into equivalence classes in multiple ways based on the element symmetry, grade, and metric signature composite properties. Both symmetry classes can be characterized by the same geometric product.

Definition 15. The *geometric algebraic product* is the Π oriented symmetric difference operator $\ominus$ (weighted by the bilinear form g) and multi-set sum $\oplus$ applied to multilinear tensor products $\otimes$ in a single operation: $\Lambda V \times \Lambda V \to \Lambda V$

$$\underbrace{\overbrace{(-1)^{\Pi(X,Y)}}^{\text{orient parity}}\overbrace{\det\left[g_{\Lambda(X\cap Y)}\right]}^{\text{intersect metric}}(\overbrace{\bigotimes_{k\in\Lambda(X\ominus Y)} w^{i_k}}^{(X\cup Y)\setminus(X\cap Y)})}_{\Lambda^1\text{-anti-symmetric, }\Lambda^g\text{-mixed-symmetry}} \otimes \underbrace{(\overbrace{\bigotimes_{k\in L(X\oplus Y)} \epsilon_{i_k}^{\otimes\mu_k}}^{\text{multi-set sum}})}_{L^g\text{-symmetric}}$$

For any $v_i \in \Lambda^1 V$, we define $v_i^2 = v_i v_i = g_{ii}$, so typically the diagonal metric g of the algebra is often defined by relations like these.

Example 5 (Squaring a vector in S"+–"). $v_1^2 = v_1 v_1 = 1$, $v_2^2 = v_2 v_2 = -1$

If $v_i, v_j \in \Lambda^1 V$ are orthogonal, then $v_i v_j = -v_j v_i$ for the basis, and then geometric product and exterior product are interchangeable.

Example 6. S"++" : $v_{12}^2 = v_{12}v_{12} = v_1v_2v_1v_2 = -v_1v_1v_2v_2 = -1 \cdot 1 = -1$
S"+–" : $v_{12}^2 = v_{12}v_{12} = v_1v_2v_1v_2 = -v_1v_1v_2v_2 = -1 \cdot -1 = 1$

Example 7. S"+++" : $(v_1 + v_{123})v_{12} = (v_1 + v_1 \wedge v_2 \wedge v_3)(v_1 \wedge v_2) = v_2 - v_3$

Definition 16 (Null-basis of projective split). Let $v_\pm^2 = \pm 1$ be a basis with $v_\infty = v_+ + v_-$ and $v_\emptyset = (v_- - v_+)/2$ An embedding space $\mathbb{R}^{p+1,q+1}$ carrying the action from the group $O(p+1, q+1)$ then has $v_\infty^2 = 0$, $v_\emptyset^2 = 0$, $v_\infty \cdot v_\emptyset = -1$, and $v_{\infty\emptyset}^2 = 1$ with Minkowski plane $v_{\infty\emptyset}$ having these product properties

$$\begin{aligned} v_{\infty\emptyset}v_\infty &= -v_\infty, & v_{\infty\emptyset}v_\emptyset &= v_\emptyset, \\ v_\infty v_\emptyset &= -1 + v_{\infty\emptyset}, & v_\emptyset v_\infty &= -1 - v_{\infty\emptyset} \end{aligned}$$

Definition 17. Symmetry properties of the tensor algebra can be characterized in terms of the geometric product by two averaging operations, which are the symmetrization $\odot$ and anti-symmetrization $\boxtimes$ operators:

$$\bigodot_{k=1}^{j} \omega_k = \frac{1}{j!}\sum_{\sigma\in S_P}\prod_k \omega_{\sigma(k)}, \qquad \bigboxtimes_{k=1}^{j} \omega_k = \sum_{\sigma\in S_P}\frac{(-1)^{\Pi(\sigma)}}{j!}\prod_k \omega_{\sigma(k)}$$

These products satisfy various `MultiVector` properties, including the associative and distributive laws.

Definition 18 (Exterior product). Let $w_k \in \Lambda^{p_k} V$, then for all $\sigma \in S_{\sum p_k}$ define an equivalence relation $\sim$ such that

$$\bigwedge_k \omega_k(v_1, \dots, v_{p_k}) \sim (-1)^{\Pi(\sigma)}(\bigotimes_k \omega_k)(v_{\sigma(1)}, \dots, v_{\sigma(\sum p_k)})$$

if and only if $\prod_k \omega_k = \boxtimes_k \omega_k$ holds. It has become typical to use the $\wedge$ product symbol to denote products of such elements as $\bigwedge \Lambda V \equiv \bigotimes \Lambda V / \sim$ modulo anti-symmetrization.

Definition 19 (Symmetric Leibniz differentials). Let $\partial_k = \frac{\partial}{\partial x_k} \in L_g V$ be Leibnizian symmetric tensors, then there is an equivalence relation $\asymp$ which holds for each $\sigma \in S_p$

$$(\partial_p \circ ... \circ \partial_1)\omega \asymp (\bigotimes_k \partial_{\sigma(k)})\omega \iff \prod_k \partial_k = \bigodot_k \partial_k,$$

along with each derivation $\partial_k(\omega\eta) = \partial_k(\omega)\eta + \omega\partial_k(\eta)$.

Multiplication with an ϵ_i element is used help signify tensor fields so that differential operators are automatically applied in the `Basis` algebra as

$$\partial_j(\omega \otimes \epsilon_i) \neq (\partial_j \otimes \omega)\epsilon_i.$$

```
julia> using Reduce, Grassmann; @mixedbasis tangent(ℝ^2,3,2);
julia> (∂1+∂12) * (:(x1^2*x2^2)*e1 + :(sin(x1))*e2)
0.0 + (2 * x1 * x2 ^ 2)∂₁₂e¹ + (cos(x1))∂₁₂e² + (4 * x1 * x2)∂₁₂e¹
```

Definition 20 (Reversed product). A commonly occuring product is $\langle\tilde{\omega}\omega\rangle$.

$$|\omega|^2 = \langle\tilde{\omega}\omega\rangle\,, \qquad |\omega| = \sqrt{\langle\tilde{\omega}\omega\rangle}, \qquad ||\omega|| = \text{Euclidean } |\omega|.$$

Remark. In general $\sqrt{\omega} = e^{(\log\omega)/2}$ is valid for invertible ω.

Definition 21 (Inverse). A simple way to calculate inverses is $\omega^{-1} = \tilde{\omega}(\tilde{\omega}\omega)^{-1} = \tilde{\omega}/|\omega|^2$, with $\eta/\omega = \eta\omega^{-1}$ and $\eta\backslash\omega = \eta^{-1}\omega$. Shirokov [25] recently developed a more sophisticated theory for calculating inverses based on $\mathfrak{S}_j$ groups.

```
julia> 1/v34, inv(v34) == ~v34/abs2(v34)
(-1.0v34, true)
```

Definition 22 (Sandwich product). The sandwich is defined as $\eta \oslash \omega = \overline{\omega}\backslash\eta\omega$. Alternatively, the reversed definition is $\eta \oslash \omega = \eta\omega/\overline{\eta}$ or in Julia $\eta >>> \omega$.

```
julia> (2v3+5v4) ⊘ v3 == inv(v3)*(2v3+5v4)*involute(v3)
true
```

Since $\langle(\tilde{\omega}+\omega)(\omega+\tilde{\omega})\rangle = (\omega+\tilde{\omega})^2$, it follows $|\Re\omega|^2 = (\Re\omega)^2$. Similarly, $\langle(\tilde{\omega}-\omega)(\omega-\tilde{\omega})\rangle = -(\omega+\tilde{\omega})^2$ implies $|\Im\omega|^2 = -(\Im\omega)^2$. Due to the $\mathbb{Z}_2$-grading induced by $\omega = \Re\omega + \Im\omega$, it is possible to partition real and imaginary by

$$\langle\tilde{\omega}\rangle_r / \left|\langle\omega\rangle_r\right| = \sqrt{\langle\tilde{\omega}\rangle_r^2 / \left|\langle\omega\rangle_r\right|^2} = \sqrt{\langle\tilde{\omega}\rangle_r / \langle\omega\rangle_r} = \sqrt{(-1)^{(r-1)r/2}} \in \left\{1, \sqrt{-1}\right\},$$

which is a unique partitioning completely independent of the metric space and manifold of the algebra [18].

$$\tilde{\omega}\omega = |\omega|^2 = |\Re\omega + \Im\omega|^2 = |\Re\omega|^2 + |\Im\omega|^2 + 2\Re(\Re\omega\Im\omega)$$

Similar to `real`, `imag`, `even`, and `odd`; the `radial` and `angular` components in a multivector exponential are partitioned but with parity of their metric.

Definition 23 (Poincare-Hodge complement $\star$). Let $\omega = w_{i_1} \wedge \cdots \wedge w_{i_p}$ and $\star\omega = \tilde{\omega} I$, then $\star : \Lambda^p V \to \Lambda^{n-p} V$.

Remark. While $\star\omega$ is `complementright` of ω, the `complementleft` would be $I\tilde{\omega}$ and $!\omega$ denotes the non-metric variant the complement. The $\star$ symbol was added to the Julia language as unary operator on Julia's v1.2 release.

John Browne has discussed Grassmann duality principle in book [6], stating that every theorem (involving either of the exterior and regressive products) can be translated into its dual theorem by replacing the $\wedge$ and $\vee$ operations and applying *Poincare duality* (homology). First applying this Grassmann duality principle to the $\wedge$ product alone, let $\{\omega_k\}_k \in \Lambda^{p_k} V, P = \sum_k p_k$, then it is possible to obtain the co-product $\bigvee : \Lambda^{p_1} V_1 \times \cdots \times \Lambda^{p_g} V_g \to \Lambda^{P-(g-1)\#V} \bigcup_k V_k$. Grassmann's original notation implicitly combined $\wedge, \vee, \star$.

Join $\wedge \mapsto \cup$ union, meet $\vee \mapsto \cap$ intersection, complement $\star \mapsto {}^{\perp}$ yield an orthocomplementary propositional lattice in quantum logic:

$$(\star \bigvee_k \omega_k)(v_1, \dots, v_n) = (\bigwedge_k \star\omega_k)(v_1, \dots, v_n) \quad \textit{DeMorgan's Law.}$$

However, this is only completely true for Euclidean algebras. In general, the original Grassmann (OG) complement must be used in DeMorgan's Law, while tensor contractions utilize the Hodge complement's metric.

Definition 24 (Original Grassmann complement $|$). This operation is the same as $\star$ but is always Euclidean ($g \equiv 1$). In Julia it is also the `!` method.

Interior contractions $\eta \cdot \omega = \eta \vee \star\omega$ need both $\star$ and $|$ complements. Of fundamental importance is the complement of a complement axiom:

Theorem 1. *Let $\omega \in \Lambda^m V$, then $\star\star\omega = (-1)^{m(n-m)}\omega |I|^2$.*

Proof. Let $\omega \in \Lambda^m V$, then rewrite the expressions

$$\begin{aligned}
\star\star\omega &= \widetilde{\tilde{\omega} I} I = (-1)^{m(m-1)/2} \widetilde{\omega I} I \\
&= (-1)^{m(m-1)/2} (-1)^{(n-m-1)(n-m)/2} \omega I^2 \\
&= (-1)^{m(n-m)} (-1)^{n(n-1)/2} \omega I^2 \\
&= (-1)^{m(n-m)} \omega \tilde{I} I = (-1)^{m(n-m)} \omega |I|^2
\end{aligned}$$

Hence, the result follows since $(-1)^{n(n-1)/2} I I = \tilde{I} I = I \cdot I = |I|^2$. □

Corollary 1 (Euclidean complement of a complement axiom). Let $\omega \in \Lambda^m(\mathbb{R}^n)$, then $\star\star\omega = (-1)^{m(n-m)}\omega$ since $|I|^2 = 1$.

Lemma. Let $\omega \in \Lambda^m V$, then $I \vee \omega = \omega$.

Proof. $I \vee \omega = |^{-1}((|I) \wedge |\omega) = |^{-1}(1 \wedge |\omega) = |^{-1}(|\omega) = \omega$. □

Corollary 2. Obviously, $\tilde{\omega} I = I \cdot \omega$ since $I \cdot \omega = I \vee \star\omega = \star\omega = \tilde{\omega} I$.

Theorem 2. *Let $\omega \in \Lambda^m V$, then $(\omega \vee \star\omega) I = \omega \wedge \star\omega$.*

Proof. It is straight forward to check based on properties of $|, \star, \wedge, \vee$ that

$$\begin{aligned}\omega \vee \star\omega &= \omega \vee (\tilde{\omega} I) = |^{-1}((|\omega) \wedge |(\tilde{\omega} I)) \\ &= g(\omega, I)(-1)^{m(n-m)}|^{-1}((|\omega) \wedge \omega) \\ &= g(\omega, I)|^{-1}(\omega \wedge (|\omega)) = (\omega \wedge \star\omega)/I\end{aligned}$$

From this the result follows, after multiplying by I pseudoscalar. □

Theorem 3. $\eta \wedge \star\omega = (\tilde{\omega} \vee \star\tilde{\eta})I = (\tilde{\omega} \cdot \tilde{\eta})I \iff \eta \cdot \omega = \eta \vee \star\omega = (\tilde{\omega} \wedge \star\tilde{\eta})/I.$

Theorem 4. *Let $\eta, \omega \in \Lambda^m V$, then $\tilde{\eta} \cdot \tilde{\omega} = \eta \cdot \omega$.*

Proof. Let $\eta, \omega \in \Lambda^m V$, then $\tilde{\eta} \cdot \tilde{\omega} = ((-1)^{m(n-m)})^2(\eta \cdot \omega) = \eta \cdot \omega$. □

Corollary 3 (Absolute value $|\omega|^2 = \omega \cdot \omega$).

$$(\omega \cdot \omega)I = \tilde{\omega} \wedge \star\tilde{\omega} = \tilde{\omega} \star \tilde{\omega} = \tilde{\omega}\omega I = |\omega|^2 I \iff \omega \cdot \omega = \tilde{\omega}\omega$$

The expressions can also be reversed: $\omega \wedge \star\omega = \omega \star \omega = \omega\tilde{\omega}I = |\omega|^2 I$. However, when $\eta \in \Lambda^r V$ and $\omega \in \Lambda^s V$ are of unequal grade, then there exist several possible variations of graded contraction operations. Of course, the most natural option for the interior contraction is Grassmann's right contraction also written $\eta|\omega = \eta \vee \star\omega$. However, many authors such as Dorst [8] prefer the Conventional contraction, which is one of the other variations.

Contraction	left(η, ω)	right(η, ω)
Grassmann	$\langle\omega\rangle_s \vee \star\langle\eta\rangle_r = \langle\tilde{\eta}\omega\rangle_{s-r}$	$\langle\eta\rangle_r \vee \star\langle\omega\rangle_s = \langle\tilde{\eta}\omega\rangle_{r-s}$
Reversed	$\langle\tilde{\omega}\rangle_s \vee \star\langle\tilde{\eta}\rangle_r = \langle\eta\tilde{\omega}\rangle_{s-r}$	$\langle\tilde{\eta}\rangle_r \vee \star\langle\tilde{\omega}\rangle_s = \langle\eta\tilde{\omega}\rangle_{r-s}$
Conventional	$\langle\omega\rangle_s \vee \star\langle\tilde{\eta}\rangle_r = \langle\eta\omega\rangle_{s-r}$	$\langle\tilde{\eta}\rangle_r \vee \star\langle\omega\rangle_s = \langle\eta\omega\rangle_{r-s}$
Unconventional	$\langle\tilde{\omega}\rangle_s \vee \star\langle\eta\rangle_r = \langle\tilde{\eta}\tilde{\omega}\rangle_{s-r}$	$\langle\eta\rangle_r \vee \star\langle\tilde{\omega}\rangle_s = \langle\tilde{\eta}\tilde{\omega}\rangle_{r-s}$

```
julia> (v1 + v2) ⋅ (1.5v2 + v3)
1.5v
```

For the null-basis, complement operations are different:

$$\begin{aligned}\star v_\infty &= \star(v_+ + v_-) = (v_- + v_+)v_{1\ldots n} = v_{\infty 1\ldots n} \\ \star 2v_\emptyset &= \star(v_- - v_+) = (v_+ - v_-)v_{1\ldots n} = -2v_{\emptyset 1\ldots n}\end{aligned}$$

This Hodge complement satisfies $\langle\omega \vee \star\omega\rangle I = \omega \wedge \star\omega$, which is naturally a result of using the geometric product in the definition, but with the usage of the null-basis applied. If null-basis are used as indices, then complementing them is a more tricky exercise than for standard indices. Not only is the Hodge complement of the null-basis different, but the metric independent complement of the null-basis is yet again different than normal indices,

$$\begin{aligned}!v_\infty &= !(v_+ + v_-) = (v_- - v_+)v_{1\ldots n} = 2v_{\emptyset 1\ldots n} \\ !2v_\emptyset &= !(v_- - v_+) = (v_+ + v_-)v_{1\ldots n} = -v_{\infty 1\ldots n}\end{aligned}$$

For that variation of complement, $||\omega||^2 I = \omega \wedge !\omega$ holds.

Definition 25. Let $\nabla = \sum_k \partial_k v_k$ be a vector field and $\epsilon = \sum_k \epsilon_k(x) w_k \in \Omega^1 V$ be unit sums of the mixed-symmetry basis. Elements of $\Omega^p V$ are known as *differential p-forms* and both ∇ and ϵ are *tensor fields* dependent on $x \in W$. A differential form is $dx_k = \epsilon_k(x) w_k$, so that $\epsilon_k = dx_k / w_k$ and $\partial_k \omega(x) = \omega'(x)$. The space W does not have to equal V, as $\Omega^p V$ could have $\mathbb{K} = LW$ coefficients.

Definition 26. Differential $d : \Omega^p V \to \Omega^{p+1} V$, co-differential $\delta : \Omega^p V \to \Omega^{p-1} V$

$$\star d\omega = \star(\nabla \wedge \omega) = \nabla \times \omega, \qquad \omega \cdot \nabla = \omega \vee \star\nabla = \partial\omega = -\delta\omega.$$

These two maps have the special properties $d \circ d = 0$ and $\partial \circ \partial = 0$ for any form ω and vector field ∇. In topology there is *boundary* operator ∂ defined by $\partial\epsilon = \epsilon \cdot \nabla = \sum_k \partial_k \epsilon_k$ and is commonly discussed in terms the limit $\epsilon(x) \cdot \nabla\omega(x) = \lim_{h \to 0} \frac{\omega(x + h\epsilon) - \omega(x)}{h}$, which is the directional derivative [27].

Example 8 (Vorticity curl of vector-field). In 3-dimensions, curl is defined as

$$\star d(dx_1 + dx_2 + dx_3) = (\partial_2 - \partial_3) dx_1 + (\partial_3 - \partial_1) dx_2 + (\partial_1 - \partial_2) dx_3.$$

Example 9 (Boundary of 3-simplex). Faces of simplex (oriented):

$$\partial(w_{1234}) = -\partial_4 w_{123} + \partial_3 w_{124} - \partial_2 w_{134} + \partial_1 w_{234}.$$

```
julia> tangent(ℝ^3)(∇)
0v12 + 0v13 +∂₁ 1v1 + 0v23 +∂₁ 1v2 +∂₁ 1v3

julia> @basis tangent(ℝ^3,2,3); ⋆d(v1+v2+v3)
0 -∂₂ 1v1 +∂₃ 1v1 +∂₁ 1v2 -∂₃ 1v2 -∂₁ 1v3 +∂₂ 1v3

julia> ∂(Λ(tangent(ℝ^4,2,4)).v1234)
0.0 -∂₄ 1v123 +∂₃ 1v124 -∂₂ 1v134 +∂₁ 1v234
```

As a result of Grassmann's exterior & interior products, the Hodge-DeRahm chain complex from cohomology theory is

$$0 \underset{\partial}{\overset{d}{\rightleftarrows}} \Omega^0(M) \underset{\partial}{\overset{d}{\rightleftarrows}} \Omega^1(M) \underset{\partial}{\overset{d}{\leftrightarrows}} \cdots \underset{\partial}{\overset{d}{\rightleftarrows}} \Omega^n(M) \underset{\partial}{\overset{d}{\rightleftarrows}} 0,$$

having dimensional equivalence brought by the Grassmann-Poincare-Hodge complement duality,

$$\mathcal{H}^{n-p} M \cong \frac{\ker(d\Omega^{n-p} M)}{\mathrm{im}(d\Omega^{n-p+1} M)}, \qquad \dim \mathcal{H}^p M = \dim \frac{\ker(\partial\Omega^p M)}{\mathrm{im}(\partial\Omega^{p+1} M)}$$

The rank of the grade p boundary incidence operator is

$$\mathrm{rank} \left\langle \partial \langle M \rangle_{p+1} \right\rangle_p = \min \left\{ \dim \left\langle \partial \langle M \rangle_{p+1} \right\rangle_p, \dim \langle M \rangle_{p+1} \right\}$$

Invariant topological information can be computed using ranks of homology groups, where $b_p(M) = \dim \mathcal{H}^p M$

$$b_p(M) = \dim \langle M \rangle_{p+1} - \mathrm{rank} \left\langle \partial \langle M \rangle_{p+1} \right\rangle_p - \mathrm{rank} \left\langle \partial \langle M \rangle_{p+2} \right\rangle_{p+1}$$

are Betti numbers with Euler characteristic $\chi(M) = \sum_p (-1)^p b_p$.

Theorem 5 (Integration by parts & Stokes). *Let $\nabla \in \Omega_1 V$ be a Leibnizian vector field operator, then $d, -\partial$ are Hilbert adjoint Hodge-DeRahm operators*

$$\int_M d\omega \wedge \star\eta + \int_M \omega \wedge \star\partial\eta = 0, \qquad \langle d\omega \vee \star\eta\rangle = \langle \omega \vee \star - \partial\eta\rangle .$$

Proof. $\partial\omega = \omega\cdot\nabla = |^{-1}(|\omega\wedge\star|\nabla) = (-1)^n(-1)^{nk}\star d\star\omega$. Then substitute this into $\int_M \omega\wedge(-1)^{mk+m+1}\star\star d\star\eta = (-1)^{km+m+1}(-1)^{(m-k+1)(k-1)}\int_M \omega\wedge d\star\eta$, apply identity $(-1)^{km+m+1}(-1)^{(m-k+1)(k-1)} = (-1)^k$ and $(-1)^k\int_M \omega\wedge d\star\eta = \int_M d(\omega\wedge\star\eta) - (-1)^{k-1}\omega\wedge d\star\eta = \int_M d\omega\wedge\star\eta$. Stokes identity can be proved by relying on a variant of the *common factor theorem* by Browne [6]. □

Theorem 6 (Clifford-Dirac-Laplacian). *The Dirac operator* [11] *or Laplacian square root is* $(\nabla^2)^{\frac{1}{2}}\omega = \pm\nabla\omega = \pm\nabla\wedge\omega \pm \nabla\cdot\omega = \pm d\omega \pm \partial\omega$.

$$\nabla^2\omega = \nabla\wedge(\omega\cdot\nabla) + (\nabla\wedge\omega)\cdot\nabla) = \mp(\mp\omega\nabla)\nabla).$$

Elements ω are harmonic if $\nabla\omega = 0$ and both closed $d\omega = 0$ and coclosed $\delta\omega = 0$, such that $\mathcal{H}^pM = \{\nabla\omega = 0 : \omega \in \Omega^pM\}$. Note: $\nabla\omega \neq \omega\nabla$ yet $\nabla^2\omega = \omega\nabla^2 = d\partial\omega + \partial d\omega$

Example 10. S"$\infty\emptyset$+++"(∇^2) $\mapsto$ $-2\partial_{\infty\emptyset} + \partial_1^2 + \partial_2^2 + \partial_3^2$

Theorem 7 (Hodge-DeRahm). $\Omega^pM = \mathcal{H}^pM \oplus im(d\Omega^{p-1}M) \oplus im(\partial\Omega^{p+1}M)$.

Definition 27 (Faraday bivector dA with $ddA = 0$).

$$Ev_t + \star(Bv_t) = (\nabla V - \partial_t A)v_t + \star((\star dA)v_t) = dA,$$

where E is electric field, B magnetic field, A is vector potential.

Example 11 (Maxwell's equations rewritten).

$$ddA = 0 \iff \begin{cases} \partial B = 0 & \text{Gauss's law} \\ \star dE = -\partial_t B & \text{Faraday's law} \end{cases}$$

$$\star d\star dA = J \iff \begin{cases} \partial E = \rho & \text{Gauss's law} \\ \star dB = J + \partial_t E & \text{Ampere's law} \end{cases}$$

Maxwell's equations simplify to a single spacetime wave equation

$$\nabla(Ev_t + \star(Bv_t)) = \nabla dA = \star d\star dA = \nabla^2 A = J$$

Theorem 8 (First grade sandwich product). *Reflection by hyperplane $\star\nabla$ has isometry $\omega \oslash \nabla = -\nabla\backslash\omega\nabla$.*

Proof. Let $\nabla \in \Lambda^1 V$, then $\omega = (\nabla\backslash\nabla)\omega = \nabla\backslash(d\omega + \partial\omega)$ where $\nabla \parallel \partial\omega$ and $\nabla \perp d\omega$. Let's reflect across the hyperplane $\star\nabla$,

$$\begin{aligned} \nabla\backslash(d\omega - \partial\omega) &= \nabla\backslash(d\omega - \partial\omega)(\nabla\backslash\nabla) \\ &= -\nabla^2\backslash(d\omega + \partial\omega)\nabla = -\nabla\backslash\omega\nabla. \end{aligned}$$

Hence, reflection by hyperplane $\star\nabla$ has isometry $\omega \oslash \nabla$ which for $\nabla = v_j$ is the map $\mathbb{R}^n \to \mathbb{R}_1 \times \cdots \times \overline{\mathbb{R}}_j \cdots \times \mathbb{R}_n$. □

Theorem 9 (Cartan-Dieudonne). *Every isometry of $V \to V$ is the composite of at most k reflections across non-singular hyperplanes* [3] [26]. *Hence there exist vectors ∇_j such that*

$$(((\omega \oslash \nabla_1) \oslash \nabla_2) \oslash \cdots) \oslash \nabla_k = \omega \oslash (\nabla_1 \nabla_2 \cdots \nabla_k)$$

for any isometry element of the orthogonal group $O(p, q)$.

Note that elements under transformations of this group preserve inner product relations. The even grade operators make up the rotational group, where each bivector isometry is a composition of two reflections [3] [7].

Exponential map and Lie group parameter special cases: consider the differential equation $\partial_i \epsilon_j = \epsilon_j \oslash \omega$, solution: $\epsilon_j(x) = \epsilon_j(0) \oslash e^{x_i \omega}$ where $\theta = 2x_i$ is Lie group parameter. Then for a normalized ω,

$$e^{\theta\omega} = \sum_k \frac{(\theta\omega)^k}{k!} = \begin{cases} \cosh\theta + \omega \sinh\theta, & \text{if } \omega^2 = 1, \\ \cos\theta + \omega \sin\theta, & \text{if } \omega^2 = -1, \\ 1 + \theta\omega, & \text{if } \omega^2 = 0. \end{cases}$$

Note that $\nabla \oslash e^{\theta\omega/2} = \nabla e^{\theta\omega}$ is a double covering when using the complex numbers in the Euclidean plane.

Remark. The sandwich must be with reversion on the left side, otherwise the rotation is clockwise and opposite of the phase parameter convention used by Euler's formula. For example, observe the resultant direction of rotation

$$e^{\frac{\pi}{4} v_{12}} v_1 \widetilde{e^{\frac{\pi}{4} v_{12}}} = -v_2$$

which means it is rotating in the wrong direction opposite of Euler, while

$$\widetilde{e^{\frac{\pi}{4} v_{12}}} v_1 e^{\frac{\pi}{4} v_{12}} = v_2$$

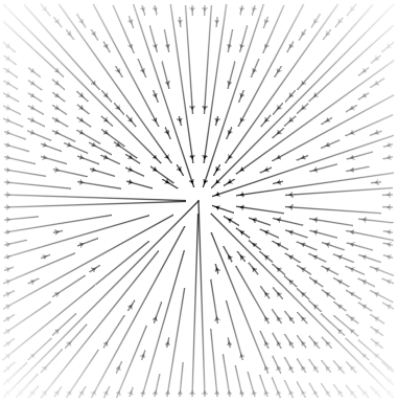
is compatible with Euler's convention. So, the sandwich must be applied with its reversion on the left side–if the standard Euler rotation direction is desired. However, many authors follow the opposite convention of clockwise instead.

Theorem 10 (Leibniz-Taylor series). *Let $\partial_X = \prod_k \partial_k^{\mu_k}$ be defined so that size of its index multi-set is constrained to $|X| = \sum_k \mu_k$, then $e^{\partial\epsilon}\omega(x)$ is*

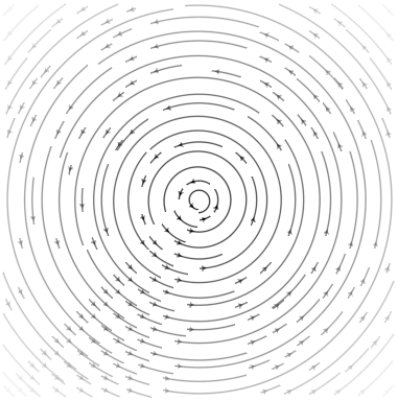
$$e^{\partial\epsilon}\omega(x) = \sum_{j=0}^{\mu} \frac{(\partial\epsilon)^j}{j!}\omega(x) = \sum_{j=0}^{\mu} \sum_{|X|=j} \prod_k \frac{(\partial_k \epsilon_k(x))^{\mu_k}}{\mu_k!}\omega(x).$$

The multivariate *product rule* is encoded into the geometric algebraic product when using mixed-symmetry.

```
using Grassmann, Makie
basis"2" # Euclidean
streamplot(vectorfield(exp(π*v12/2)),-1.5..1.5,-1.5..1.5)
streamplot(vectorfield(exp((π/2)*v12/2)),-1.5..1.5,-1.5..1.5)
streamplot(vectorfield(exp((π/4)*v12/2)),-1.5..1.5,-1.5..1.5)
streamplot(vectorfield(v1*exp((π/4)*v12/2)),-1.5..1.5,-1.5..1.5)
@basis S"+-" # Hyperbolic
streamplot(vectorfield(exp((π/8)*v12/2)),-1.5..1.5,-1.5..1.5)
streamplot(vectorfield(v1*exp((π/4)*v12/2)),-1.5..1.5,-1.5..1.5)
```

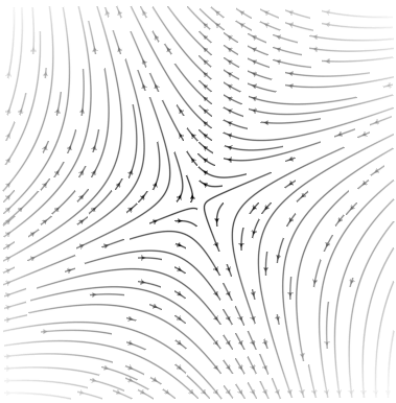


(A) $x \oslash e^{\pi v_{12}/2}$, $\mathbb{R}^2$

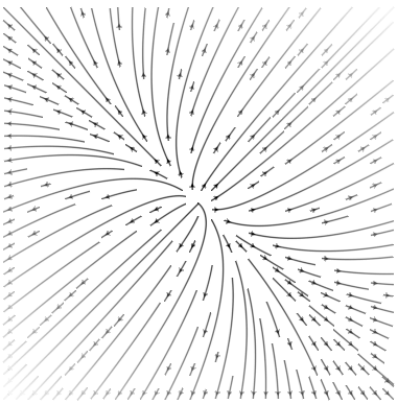


(B) $x \oslash e^{\frac{\pi}{2} v_{12}/2}$, $\mathbb{R}^2$

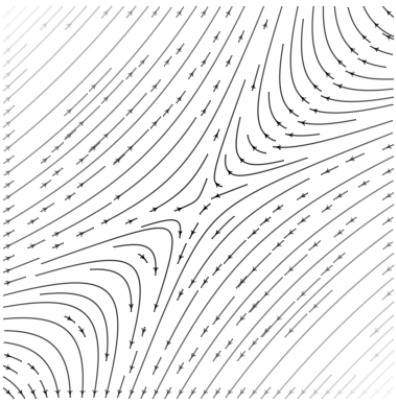
(C) $x \oslash e^{\frac{\pi}{4} v_{12}/2}$, $\mathbb{R}^2$



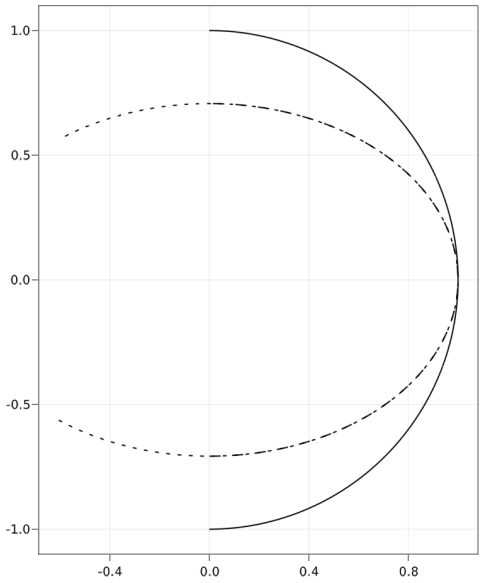
(D) $x \oslash (v_1 \ominus e^{\frac{\pi}{4} v_{12}/2})$, $\overline{\mathbb{R}} \oplus \mathbb{R}$



(E) $x \oslash e^{\frac{\pi}{8} v_{12}/2}$, $\mathbb{R} \oplus \mathbb{R}'$



(F) $x \oslash (v_1 \ominus e^{\frac{\pi}{4} v_{12}/2})$, $\overline{\mathbb{R}} \oplus \mathbb{R}'$



$v_1 \oslash \exp\left(\frac{\theta}{\sqrt{2}} v_{12}\right)$, $\theta \in [-\frac{\pi}{4}, \frac{\pi}{4}]$

Example 12 (Eccentric geometric product). Consider the elliptic algebra

$$v_1, v_2 \in \Lambda^1 \texttt{D"1,2"`} : \qquad v_{12} v_{12} = -2, \quad \frac{v_{12}}{\sqrt{2}} \frac{v_{12}}{\sqrt{2}} = -1.$$

The numbers start growing faster on one axis, so due to the spatial dilation the exponential θ parameter is also dilated by an eccentricity factor.

$$v_1 \oslash \exp\left(\frac{\theta}{\sqrt{2}} v_{12}\right), \qquad |v_{12}| = \sqrt{2}, \qquad \frac{v_{12}}{|v_{12}|} = \frac{v_{12}}{\sqrt{2}}.$$

This elliptic clock is $\sqrt{2}$ faster than a standard circular clock, since bivector v_{12} has a greater norm than 1 by that factor, it is fixed by normalization.

Example 13 (Approximation to Earth's geoid metric geometric algebra). The radius of the Earth along the equator is $r_E = 6378137$ m, but the Earth is not a perfect sphere; it is better approximated by an ellipse with flattening parameter $f =$ `Geophysics.flattening(Earth)` and this eccentricity factor

$$1.0067395 \approx \frac{1}{(1-f)^2}, \qquad f \approx \frac{1}{298.257223563}$$

using the metric `D"1,1.0067395"`, the radius along the North can be computed an error of about $1.6179573236741 \times 10^{-7}$ with normalization $|v_{12}| = \frac{1}{1-f}$

$$(r_E v_1) \oslash \exp\left(\frac{\pi}{4}(1-f) v_{12}\right) \approx 6356752.304e6.$$

This result demonstrates that it is feasible to do nuanced calculations with Earth's elliptical geoid approximation by adjusting the metric tensor for tuned geometric algebra bivector exponentiation.

Take the real part and imaginary part to transform into eigenvalues:

$$\omega = \Re\omega + \Im\omega \mapsto \Re\omega \pm \sqrt{(\Im\omega)^2} = \Re\omega \pm \sqrt{-|\Im\omega|^2} = \lambda$$

Let Ω be a linear operator, λ_i are non-repeating eigenvalues of Ω and v_i are the corresponding eigenvectors such that $\Omega v_i = \lambda_i v_i$ with action of Ω on v_i.

Theorem 11 (Normalized eigenblade theorem). *Let Ω, λ_i, v_i be as above,*

$$\star(\Omega - \lambda_i) \star v_i = v_i \prod_{j \neq i} (\lambda_i - \lambda_j).$$

Proof. Apply $(\Omega - \lambda_i)$ as an outermorphism operator to the v_j to see that

$$\begin{aligned}
\star(\Omega - \lambda_i) \star v_i &= \star(\Omega - \lambda_i) \bigwedge_{j \neq i} v_j \\
&= \star \bigwedge_{j \neq i} (\Omega - \lambda_i) v_j = \star \bigwedge_{j \neq i} (\lambda_j - \lambda_i) v_j \\
&= \star \bigwedge_{j \neq i} v_j \prod_{j \neq i} (\lambda_i - \lambda_j) = v_i \prod_{j \neq i} (\lambda_i - \lambda_j)
\end{aligned}$$

Hence, a normalized eigenblade $\star v_i$ is naturally proportional to the outermorphism operating on the eigenblade by an eigenvalue product factor. □

Theorem 12 (Eigenblade normalization conjecture). *Let Ω, λ_i, v_i be as above and also let e_j be an element of an orthogonal basis, then*

$$Proj\star\,(\Omega - \lambda_i) \star v_i = Proj\star\,(\Omega - \lambda_i) \star e_j.$$

Unfortunately, the margins are currently too narrow to contain this proof.

In spirit of the recent investigations [22][4] into eigenblades, the following are adaptations of the Lagrange-Sylverster theorem to Grassmann algebra.

Theorem 13 (Analogue of Frobenius covariant). *Let Ω, λ_i, v_i be as above,*

$$Proj\star\,(\Omega - \lambda_i) \star v_i = Proj\star \prod_{j \neq i} \frac{(\Omega - \lambda_i) v_j}{\lambda_i - \lambda_j} = \prod_{j \neq i} \frac{\Omega - \lambda_i I}{\lambda_i - \lambda_j}$$

Proof. By Theorem 11 observe $\star(\Omega - \lambda_i) \star v_i$ is proportional to eigenvectors v_i by a product factor based on eigenvalues. Given the established Langrange-Sylvester theorem in the matrix algebra literature [28], the result follows. □

Theorem 14 (Analogue of Lagrange-Sylvester). *Let $\Omega, \lambda_i, v_i, e_j$ be as above,*

$$\Omega = \sum_{i=1}^{n} Proj\star\,(\Omega - \lambda_i) \star e_i = \sum_{i=1}^{n} Proj\star \bigwedge_{\substack{j=1 \\ j \neq i}}^{n} (\Omega - \lambda_i) e_j = \sum_{i=1}^{n} \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{\Omega - \lambda_i I}{\lambda_i - \lambda_j}$$

Proof. Take the expression from Theorem 13 and apply Theorem 12. □

Example 14 (Exercise: what are the solutions to this equation?).

$$\frac{\star(\Omega - \lambda_i) \star v_i}{\prod_{j \neq i} (\lambda_i - \lambda_j)} = \frac{\star(\Omega - \lambda_i) \star e_j}{\sqrt{|\lambda_i|^{n-1}} \prod_{j \neq i} \sqrt{\lambda_i - \lambda_j}}$$

Dyadic tensors (matrices) are represented in Grassmann's algebra by nested `Chain{V,1,Chain{V,1}}` elements, existing in a 2^{2n}-dimensional mother algebra from direct sum of the n-dimensional vector space and its dual vector space. The dyadic product of the vector basis and covector basis elements form the n^2-dimensional bivector subspace of the full $\frac{(2n)!}{2(2n-2)!}$-dimensional bivector sub-algebra of that space. Note that `Λ(ℝ3)` gives the vector basis, and `Λ(ℝ3)'` gives the covector basis:

```
julia> Λ(ℝ3)
DirectSum.Basis{()×××,8}(v, v1, v2, v3, v12, v13, v23, v123)

julia> Λ(ℝ3)'
DirectSum.Basis(){---',8}(w, w1, w2, w3, w12, w13, w23, w123)
```

The `@mixedbasis` command yields a local vector and covector basis. The sum `w1+2w2` is interpreted as a covector element of the dual vector space, which can be evaluated as a linear functional when a vector argument is input.

```
julia> L = (v1+2v2)∧(3w1+4w2)
0v12 + 3v1w1 + 4v1w2 + 6v2w1 + 8v2w2 + 0w12

julia> L(v1+v2)
7v1 + 14v2 + 0w1 + 0w2
```

The element L is a linear form which can be evaluated as a computation equivalent to a matrix multiplication. Thus it is possible to express the Frobenius covariant projectors within the mother algebra formalism. However, since this requires a higher dimensional geometric algebra implementation, the natural efficient alternative is a nested `Chain{V,1,Chain{V,1}}` algebra.

```
julia> L = DyadicChain((1v1+2v2)⊗(3v1+4v2))
(3v1 + 6v2)v1 + (4v1 + 8v2)v2

julia> L(v1+v2)
7v1 + 14v2
```

This leads naturally to a design choice where a matrix is represented as a nested data structure as a vector of column vectors of the matrix. For purposes of discrete differential geometry, there are many cases where this nested data structure is advantageous when applying exterior products to various combinations of nested elements. Although the initial goal with geometric algebra was to eliminate the traditional matrix representations in favor of a multilinear foundation, matrix formalisms can be found to exist naturally within nested `Chain` algebra or in a bivector subspace of the mother algebra.

Due to convenience, it is found that this nested approach is the most computationally practical in terms of programming design; thus, the mother algebra alternative remains as more of a theoretical interest.

Definition 28. $[p_{1\ldots n}]$ is a nested vector $(p_1)v_1 + \cdots + (p_n)v_n$ with $p_i \in \Lambda^1 V$.

To help with the unification with matrix formalisms, additional methods such as the double dot : were implemented, to help compute the norm.

Example 15 (Triangle in projective coordinates). Let v_1, v_2, v_3 be given point vertices of a triangle in the v_{23} plane of $\mathbb{R}^3$, so $\begin{cases} 0 : p_0 & = 1v_1 + 0v_2 + 0v_3, \\ x : p_1 & = 1v_1 + 1v_2 + 0v_3, \\ y : p_2 & = 1v_1 + 0v_2 + 1v_3 \end{cases}$

Then $p_{012} = p_0 \wedge p_1 \wedge p_2 = v_{123} = I$ is a trivector volume, while

$$[p_{012}] = p_0 v_1 + p_1 v_2 + p_2 v_3 = (v_1)v_1 + (v_1 + v_2)v_2 + (v_1 + v_3)v_3$$

is a dyadic tensor. Subtract base point $[p_{000}]$ to get vector frame $[\vec{p}_{12-0}]$

$$\vec{p}_{12-0} = (p_1 - p_0) \wedge (p_2 - p_0) = (p_0 - p_2) \wedge (p_1 - p_2) = \vec{p}_{01-2}$$

The volume (area) equivalence class of any triangle is the bivector

$$|\star \vec{p}_{12-0}| = |\star \vec{p}_{02-1}| = |\star \vec{p}_{01-2}|.$$

Theorem 15 (Linear system with Grassmann products). *Let* $p_0, \ldots, p_n \in \Lambda^1 V$,

$$[p_{1\ldots n}] \vee \star \sum_{i=1}^{n} \frac{p_{1\ldots(i-1)} \wedge p_0 \wedge p_{(i+1)\ldots n}}{p_{1\ldots n}} v_i = p_0.$$

Proof. Given a dyadic tensor product $[p_{1\ldots n}] \cdot x = p_0$, let's solve for x explicitly

$$\begin{aligned} p_0 \wedge p_{1\ldots(i-1)} \wedge p_{(i+1)\ldots n} &= ([p_{1\ldots n}] \cdot x) \wedge p_{1\ldots(i-1)} \wedge p_{(i+1)\ldots n} \\ &= (x_i p_i) \wedge p_{1\ldots(i-1)} \wedge p_{(i+1)\ldots n} \end{aligned}$$

Hence $x = \sum_{i=1}^{n} \frac{p_{1\ldots(i-1)} \wedge p_0 \wedge p_{(i+1)\ldots n}}{p_{1\ldots n}} v_i$ and the result follows. □

This means that using only exterior products enables efficiently solving the linear system by allocating $\{p_{1\ldots i} \wedge p_{i+1}\}_{i=0}^{n-1}$ and $\left\{p_{n-i} \wedge p_{(n-i+1)\ldots n}\right\}_{i=0}^{n-1}$ and then taking exterior products with p_0 also.

$$p_0 \in p_{1\ldots n} \iff \forall i : p_{1..n} = p_{1\ldots(i-1)} \wedge p_0 \wedge p_{(i+1)\ldots n}$$

Since exterior products are oriented, it is sufficient to check the orientation of the hyperplanes with respect to the reference point for determining whether p_0 is a point contained in the simplex $p_{1\ldots n}$. Thus, it is sufficient to check the orientation of all the same exterior products as for the linear system, while calculating a linear `inv` is only a partial application of this principle and requires also allocating a transposed dyadic result:

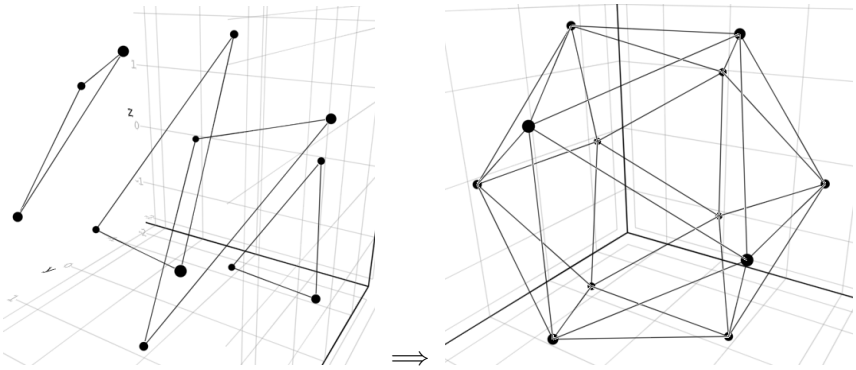
$$[p_{1\ldots n}]^{-1} = \left(\sum_{i=1}^{n} \star \frac{p_{1\ldots(i-1)} \wedge p_{(i+1)\ldots n}}{((-1)^i)^{n-1} p_{1\ldots n}} v_i \right)^T$$

Programming the `A\b` method is straight forward with some Julia language metaprogramming in the Grassmann.jl package. Benchmarks with `Grassmann` in Julia indicated a 3× faster performance than `StaticArrays.SMatrix` for applying `A\b` to bundles of dyadic elements at the time of testing.

When the dyadic chain triangle (or simplex) example is extended to a bundle of simplices, in `Grassmann` the nested usage of pure `ChainBundle` parametric types is the preferred design choice for larger re-usable global cell geometries, from which local dyadics can be selected. It is typical to represent a discrete manifold with a set of finite element hyperedges over a bundle of points. The set of points $P = \{p_i\}_{i=1}^{n_p} \subset \Lambda^1 V$ is a `ChainBundle` and the set of triangles (or tetrahedra or $(m-1)$-simplices) is $T = \{t_i\}_{i=1}^{n_t} \subset \Lambda^m P$. Hence, the natural data structure for a `ChainBundle` is `Vector{Chain}`. For points it is `Vector{Chain{V,1}}` and for hyperedges it is `Vector{Chain{P}}`, so the underlying structure can be nested (especially for finite element hyperedges).

Example 16 (Grassmann element bundles). Icosahedron $P \subset \Lambda^1\mathbb{R}^{12}, T \subset \Lambda^3 P$

$$
\begin{array}{l}
p_1 : 1v_1 + 0v_2 + 1v_3 + \varphi v_4 \\
p_2 : 1v_1 + \varphi v_2 + 0v_3 + 1v_4 \\
p_3 : 1v_1 + 1v_2 + \varphi v_3 + 0v_4 \\
p_4 : 1v_1 + 0v_2 + 1v_3 - \varphi v_4 \\
p_5 : 1v_1 - \varphi v_2 + 0v_3 + 1v_4 \\
p_6 : 1v_1 + 1v_2 - \varphi v_3 + 0v_4 \\
p_7 : 1v_1 + 0v_2 - 1v_3 + \varphi v_4 \\
p_8 : 1v_1 + \varphi v_2 + 0v_3 - 1v_4 \\
p_9 : 1v_1 - 1v_2 + \varphi v_3 + 0v_4 \\
p_{10} : 1v_1 + 0v_2 - 1v_3 - \varphi v_4 \\
p_{11} : 1v_1 - \varphi v_2 + 0v_3 - 1v_4 \\
p_{12} : 1v_1 - 1v_2 - \varphi v_3 + 0v_4
\end{array}
\quad \overset{\partial(\text{hull})}{\mapsto} \quad
\begin{array}{ll}
t_1 : -p_{\{1,5,7\}} & t_{11} : +p_{\{10,11,12\}} \\
t_2 : +p_{\{5,7,12\}} & t_{12} : +p_{\{2,3,8\}} \\
t_3 : +p_{\{6,8,10\}} & t_{13} : -p_{\{3,4,8\}} \\
t_4 : +p_{\{6,10,12\}} & t_{14} : +p_{\{1,2,3\}} \\
t_5 : +p_{\{2,6,7\}} & t_{15} : +p_{\{1,5,9\}} \\
t_6 : +p_{\{6,7,12\}} & t_{16} : +p_{\{5,9,11\}} \\
t_7 : -p_{\{1,2,7\}} & t_{17} : +p_{\{5,11,12\}} \\
t_8 : +p_{\{4,8,10\}} & t_{18} : +p_{\{4,9,11\}} \\
t_9 : -p_{\{2,6,8\}} & t_{19} : -p_{\{1,3,9\}} \\
t_{10} : -p_{\{4,10,11\}} & t_{20} : +p_{\{3,4,9\}}
\end{array}
$$



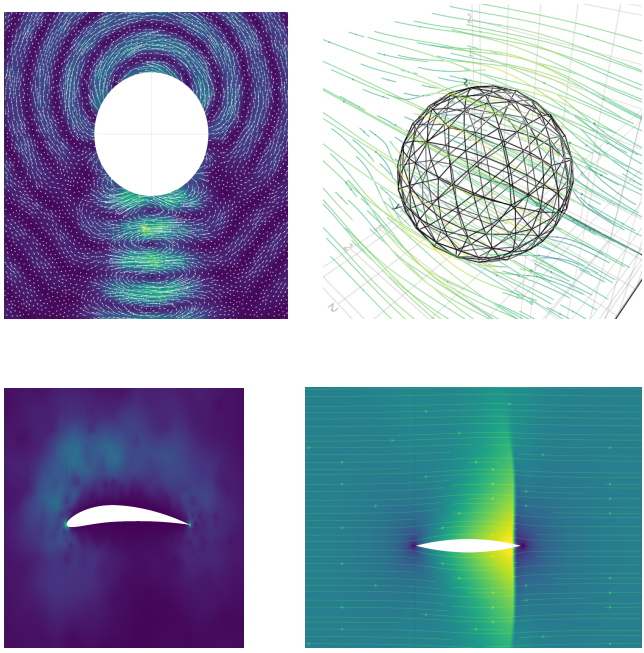
Cyclic permute points $\{(0, \pm 1, \pm\varphi)\} \mapsto \partial(\text{icosahedron hull})$

One of the goals of `Grassmann` is to provide a fundamental toolkit from which to build finite element models, so syntax such as `P[T[i]]` can be used to assemble the local dyadic tensor with the indices of simplex t_i from point cloud P. Similarly, `P[T]` will assemble an entire bundle of dyadics. This nested `ChainBundle` data structure is ideal for efficiently interfacing finite element `Chain` hyperedges over the dyadic paradigm with Grassmann exterior algebra.

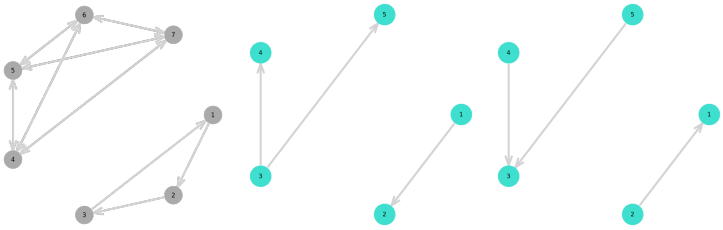
To build on the `ChainBundle` functionality of `Grassmann`, the numerical analysis package `Adapode` is being developed to provide utilities for finite element method assemblies. Syntax for Poisson $(-\nabla\cdot(a\nabla u) = f)$ or transport $(-\epsilon\nabla^2 u + b\cdot\nabla u = f)$ equations with finite element methods can be expressed in terms of methods like `volume(T)` using exterior products or `gradienthat` by applying the dyadic inverse exterior linear methods above. This kind of `Grassmann` element assembly is ideal for applying geometric algebra locally on per element basis and combining it with a global manifold topology.

```
function solvepoisson(T,E,c,f,κ,gD=0,gN=0)
    m = volumes(T)
    b = assemblefunction(T,f,m)
    A = assemblestiffness(T,c,m)
    R,r = assemblerobin(E,κ,gD,gN)
    return (A+R)\(b+r)
end
function solvetransport(T,E,c,ϵ=0.1)
    m = volumes(T)
    g = gradienthat(T,m)
    A = assemblestiffness(T,ϵ,m,g)
    b = assembleload(T,m)
    C = assembleconvection(T,c,m,g)
    return solvedirichlet(A+C,b,e)
end
```

Example 17 (Nedelec edges, scalar potential, full potential). More complicated models in given mesh geometries of any dimension can also be simulated; this foundation makes for a hands on computational interface.



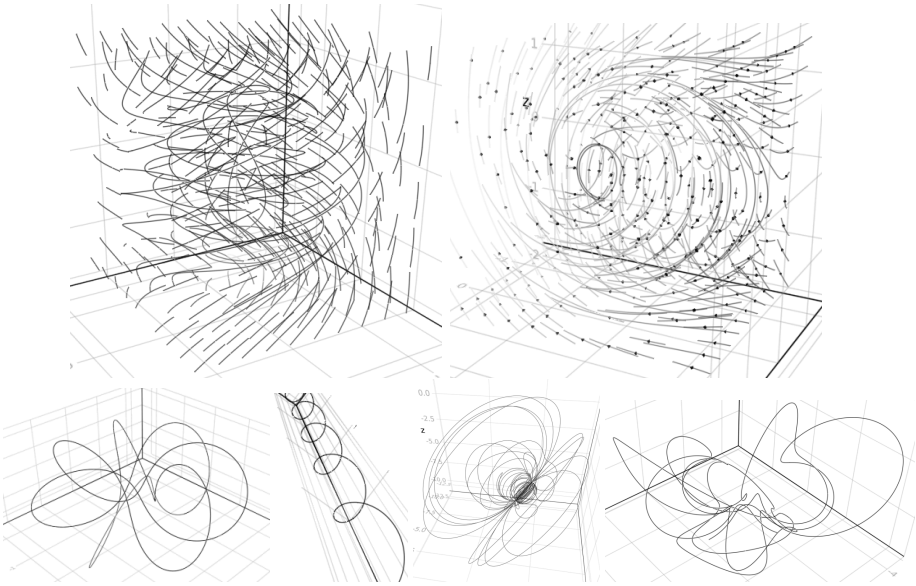
In terms of directed graphs, reversal of bivectors can be understood as the reversal of the directed edges of the graph.



$v_{123} + \star v_{123}$ in $\mathbb{R}^7$ $\quad v_{12} + v_{34} + v_{35}$ $\quad v_{21} + v_{43} + v_{53}$

With a discrete hypergraph M, in a finite element global calculation it is necessary to consider both boundary elements $\Lambda^{n-1}M$ and volume elements $\Lambda^n M$ of varying dimension, yet the global manifold is of constant dimension. However, in the Wolfram Physics Project there is a measured variation of dimension in their topology which takes away the manifold assumption. If the hyperedges themselves become quantities of fractional grade, then this may yet still be unexplored territory. Yet as long as the space is being represented by a finite element set of hyperedges, some of the formulas and principles of differential geometric algebra should still apply in a similar way as before.

In the figures below, different possible discrete bivector topologies are examined in a projective Riemann sphere [2] based on two rotation bivectors. First the space is projected up $\uparrow : \omega \mapsto (2\omega + v_\infty(\omega^2 - 1))/(\omega^2 + 1)$ and then the rotation is applied before rejecting back down $\downarrow : \omega \mapsto ((\omega \wedge b)v_\infty)/(1 - v_\infty \cdot \omega)$.



3. Conclusion

Geometric algebra has a long and somewhat obscure history [10] [3] yet it was practiced and developed further into a full foundation [21] [17] [16] [20] [5]. One of the more active areas in geometric algebra research is in computer graphics [9] [1] with the Javascript library *ganja.js* [19] [14] being especially notable as it is compatible to use with *Grassmann.jl* (and the python clifford project) for making hybrid visualization computattions. Another interesting development is the usage of hypergraph rewriting with the ZX-Calculus in the Wolfram Physics Project [12] as this is a type of quantum logic formalism isomorphic to what the multilinear geometric algebra can express.

Grassmann.jl and its accompanying support packages provide extensible platforms for fully generalized computing with geometric algebra at high dimensions. All of the types and operations in this paper are implemented using only a few thousand lines of code with Julia's type polymorphism code generation, which has been refined is being optimized for research over time.

Thus, computations involving geometric algebra, anti-symmetric tensor products, rotational algebras, and Lie bivector groups are possible with a full trigonometric suite. Conformal geometric algebra is possible with the Minkowski plane $v_{\infty\emptyset}$, based on the null-basis. Multivalued quantum logic is enabled by the $\wedge, \vee, \star$ Grassmann lattice. Mixed-symmetry algebra based on Leibniz differential algebra and Grassmann exterior calculus, having the geometric algebraic product chain rule, yields automatic differentiation and Hodge-DeRahm co/homology as unveiled by the algebra. Most importantly, the Dirac-Clifford product yields generalized Hodge-Laplacian and the Betti numbers with Euler characteristic χ. Geometric algebra unifies this all.

Grassmann algebra is a unifying mathematical foundation. Improving efficiency of multi-disciplinary research using differential geometric algebra by relying on universal mathematical principles is possible. Transforming superficial knowledge into deeper understanding is then achieved with the unified foundations widely applicable to the many different sub-disciplines related to geometry and mathematical physics. Software tooling will help overcome present challenges, such as geometric algebra not yet being standardized or taught in schools & universities or used at scale for industrial engineering science unification. Thus, the *Grassmann.jl* software enables ascending future steps to teach geometric algebra in schools/universities, use it in research and development practices, and deploy it in industrial applications. Differential geometric algebra is the future of unifying math & engineering pipelines.

Acknowledgements: Many thanks for discussions with Utensil Song, Steven De Keninck, Alexander Arsenovic, Hugo Hadfield, David Hestenes, Chris Doran, Charles Gunn, Eric Wieser; project supporters Alan Reed, Micah Fitch, Zack Li, Karl Wessel, Oliver Evans, Hongying Li, Petr Krysl, Saketh Rama, Michael Ewert; the helpful Julia community (e.g. Jameson Nash, Kristoffer Carlsson, Nathan Smith, etc); and thanks for my family.

References

[1] Joan Lasenby Anthony Lasenby and Richard Wareham. A covariant approach to geometry using geometric algebra.

[2] Alex Arsenovic. Applications of conformal geometric algebra to transmission line theory. *IEEE*, 2017.

[3] Emil Artin. *Geometric Algebra*. Interscience, 1957.

[4] Mauricio Belon. Robust quaternion estimation with geometric algebra. 2019.

[5] Alan Bromborsky. *An Introduction to Geometric Algebra and Calculus*. 2016.

[6] John Browne. *Grassmann Algebra, Volume 1: Foundations*. Barnard, 2011.

[7] F. Sommen Chris Doran, David Hestenes and N. van Acker. Lie groups as spin groups. *J. Math. Phys.*, 1993.

[8] Leo Dorst. The inner products of geometric algebra. *Applications of Geometric Algebra in Computer Science and Engineering*, 2002.

[9] Leo Dorst. A guided tour to the plane-based geometric algebra pga. 2020.

[10] D. Fearnly-Sander. Hermann grassmann & the creation of linear algebra. 1979.

[11] Garling. *Clifford Algebras: An Introduction*. 2011.

[12] Jon. Gorard. Zx-calculus and extended hypergraph rewriting systems. 2020.

[13] Hermann Grassmann. *Extension Theory (Ausdehnungslehre 1862)*. AMS, 2000.

[14] Charles G. Gunn. Geometric algebra for computer graphics (siggraph). 2019.

[15] D. Hestenes. Tutorial on geometric calculus. *Adv. Appl. Clifford Algebras*, 2013.

[16] David Hestenes. New tools for computational geometry and screw theory.

[17] Eckhard Hitzer. Introduction to clifford's geometric algebra. *Control, Measurement, and System Integration*, 2011.

[18] Lachlan Gunn Derek Abbott James Chappell, Ashar Iqbal. Functions of multivector variables. 2011.

[19] Stephen De Keninck. Non-parametric realtime rendering of subspace objects in arbitrary geometric algebras. 2019.

[20] Chris Doran & Anthony Lasenby. *Geometric Algebra for Physicists*. 2003.

[21] P. Lounesto. Marcel riesz's work on clifford algebras.

[22] Terence Tao Xining Zhang Peter B. Denton, Stephen J. Parke. Eigenvectors from eigenvalues: A survey of a basic identity in linear algebra. *arXiv*, 2019.

[23] D. Miralles R. A. Mosna and J. Vaz Jr. Z2-gradings of clifford algebras and multivector structures. *arXiv*, 2003.

[24] Siavash Shahshahani. *An Intro. Course on Differentiable Manifolds*. 2016.

[25] Dmitry S. Shirokov. On determinant, other characteristic polynomial coefficients, and inverses in clifford algebras of arbitrary dimension. *arXiv*, 2020.

[26] Ernst Snapper and Robert J. Troyer. *Affine Metric Geometry*. Academic, 1971.

[27] Garret Sobczyk. *New Foundations in Mathematics: The Geometric Concept of Number*. Springer, 2013.

[28] Michael E. Starzak. *Math. Methods in Chemistry and Physics*. Plenum, 1989.