Geometric Entity Dualization and Dual Quaternion Geometric Algebra in PGA G(3,0,1) with Double PGA G(6,0,2) for General Quadrics

Robert Benjamin Easter

Independent Researcher in Bangkok, Thailand

Email: reaster2015@gmail.com

Scopus SC: 57190277660

ORCID iD: 0000-0002-8725-1835

Daranee Pimchangthong

Assoc. Prof. Dr. at UTK ISIC in Bangkok, Thailand

Email: daranee.p@mail.rmutk.ac.th
Web: isic.rmutk.ac.th

Scopus SC: 35273219500

Abstract

In Geometric Algebra, G(3,0,1) is a degenerate-metric algebra known as PGA, originally called Projective Geometric Algebra in prior literature. It includes within it a point-based algebra, plane-based algebra, and a dual quaternion geometric algebra (DQGA). In the point-based algebra of PGA, there are outer product null space (OPNS) geometric entities based on a 1-blade point entity, and the join (outer product) of two or three points forms a 2-blade line or 3-blade plane. In the plane-based algebra of PGA, there are commutator product null space (CPNS) geometric entities based on a 1-blade plane entity, and the meet (outer product) of two or three planes forms a 2-blade line or 3-blade point. The point-based OPNS entities are dual to the plane-based CPNS entities through a new geometric entity dualization operation J_e that is defined by careful observation of the entity duals in same orientation and collected in a table of basis-blade duals. The paper contributes the new operation J_e and its implementations using three different nondegenerate algebras {G(4),G(3,1),G(1,3)} as forms of Hodge star dualizations, which in geometric algebra are various products of entities with nondegenerate unit pseudoscalars, taking a grade k entity to its dual grade $4-k$ entity copied back into G(3,0,1). The paper contributes a detailed development of DQGA. DQGA represents and emulates the dual quaternion algebra (DQA) as a geometric algebra that is entirely within the even-grades subalgebra of PGA G(3,0,1). DQGA has a close relation to the plane-based CPNS PGA entities through identities, which allows to derive dual quaternion representations of points, lines, planes, and many operations on them (reflection, rotation, translation, intersection, projection), all within the dual quaternion algebra. In DQGA, all dual quaternion operations are implemented by using the larger PGA algebra. The DQGA standard operations include complex conjugate, quaternion conjugate, dual conjugate, and part operators (scalar, vector, tensor, unit, real, imaginary), and some new operations are defined for taking more parts (point, plane, line) and taking the real component of the imaginary part by using the new operation J_e. All DQGA entities and operations are derived in detail. It is possible to easily convert any point-based OPNS PGA entity to and from its dual plane-based CPNS PGA entity, and then also convert any CPNS PGA entity to and from its DQGA entity form, all without changing orientation of the entities.

Thus, each of the three algebras within PGA can be taken advantage of for what it does best, made possible by the operation J_e and identities relating CPNS PGA to DQGA. PGA G(3,0,1) is then doubled into a Double PGA (DPGA) G(6,0,2) including a Double DQGA (DDQGA), which feature two closely related forms of a general quadric entity that can be rotated, translated, and intersected with planes and lines. The paper then concludes with final remarks.

1 Introduction

This paper[1] is about the degenerate metric Geometric Algebra $\mathcal{G}_{3,0,1}$ (known as PGA), with some comparisons to using the non-degenerate metric $\mathcal{G}_{4,1}$ (known as CGA), with the goal of being a useful exposition on this sparsely published subject that has only recently gained more attention. In addition to being a contribution to the literature as another exposition on the subject, we also have some new results to contribute in this paper.

This paper contributes the following: This paper contributes discussion of $\mathcal{G}_{3,0,1}$ that covers the details of the point-based, plane-based, and dual quaternion-based algebras that coexist within $\mathcal{G}_{3,0,1}$, with some comparisons to similar methods in $\mathcal{G}_{4,1}$. This paper contributes a simple method for implementing the entity dualization operation J_e that is similar to, but not the same as, the operation J or $\star$ as found in some other prior literature. This paper also contributes a detailed discussion of the Dual Quaternion Geometric Algebra of the even-grades subalgebra $\mathcal{G}^+_{3,0,1}$ that includes deriving dual quaternion representations of points, lines, and planes and methods for their rotations, translations, reflections, and intersections. Some of the dual quaternion entities and operations may be new, or the method by which these entities and operations in dual quaternions are derived may be new.

We assume the reader is familiar with Geometric Algebra (GA) [19], Conformal Geometric Algebra (CGA) [5][22], dual numbers, and quaternions, though we will also review some aspects of these subjects as we discuss the algebras and introduce our notations.

The Geometric Algebra $\mathcal{G}_{3,0,1}$ is already known in the published and unpublished literature and has many names, including *Clifford algebra of points, lines and planes* (in the similar $\mathcal{G}_{0,3,1}$) [23], *Projective Geometric Algebra* [13][14][15][16], *Plane-based Geometric Algebra* [20], and *Point-based Geometric Algebra* [20] (dual to plane-based). As suggested in [20] and [17], the name PGA abbreviates all of these names in $\mathcal{G}_{3,0,1}$, and we use the name PGA throughout this paper.

In this paper, we use $\mathcal{G}_{3,0,1}$ with basis vectors $\{\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, unit pseudoscalar $\mathbf{I}_4 = \mathbf{e}_0\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$, and metric $g = [\mathbf{e}_i \cdot \mathbf{e}_j] = [g_{ij}] = \text{diag}(0, 1, 1, 1)$. For the algebra of the Euclidean 3D subspace $\mathcal{G}_3$, we define the unit pseudoscalar $\mathbf{I}_3 = \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$, so that $\mathbf{I}_4 = \mathbf{e}_0\mathbf{I}_3$. The metric of $\mathcal{G}_{p,q,r}$ with $r \neq 0$ is called a degenerate metric. In Geometric Algebra, most of the published literature has concentrated on the non-degenerate algebras $\mathcal{G}_{p,q}$, so less is known about how to use the degenerate algebras such as $\mathcal{G}_{3,0,1}$. In a degenerate algebra, the inner product produces 0 for many inner products, so the inner product cannot be used in all the usual ways. In particular, dualization by inner product with the unit pseudoscalar does not work in the usual way. Dualization in $\mathcal{G}_{3,0,1}$ will be provided by a special operation J_e that we will develop in Section 4. In PGA $\mathcal{G}_{3,0,1}$, we cannot use the inner product to generate geometric null spaces as is done in CGA, so we must use other products as null spaces.

1. Version v2, 20 Dec 2023, correcting trivial errata found in v1. First version was v1, 16 Dec 2023, uploaded to viXra.org preprint repository. This research paper may later be split into 2 or 3 published papers.

In [23], the algebra is $\mathcal{G}_{0,3,1}$ with basis vectors $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}\}$, unit pseudoscalar $\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3\mathbf{e}$, and metric $\text{diag}(-1,-1,-1,0)$. These metrical differences, as compared to PGA $\mathcal{G}_{3,0,1}$, mainly cause changes in the signs in some expressions but do not fundamentally change the geometric content of the algebra.

The book [22] discusses the concept of Geometric Product Null Space, which we make extensive use of in this paper. Usually, only certain parts of the full geometric product are considered to be null spaces. The inner product part is called the Geometric Inner Product Null Space (GIPNS). Similarly, for the outer product part there is the Geometric Outer Product Null Space (GOPNS). These names are often shortened to just IPNS and OPNS, and have been used in CGA, where OPNS entities dualize to IPNS entities. In this paper, we introduce the terminology of Commutator Product Null Space (CPNS) entities as dual to the OPNS entities in $\mathcal{G}_{3,0,1}$.

In CGA $\mathcal{G}_{4,1}$, we dualize the OPNS CGA surface entity $\mathbf{A}$ to its corresponding IPNS CGA surface entity $\mathbf{A}^* = \mathbf{A}/\mathbf{I}_5$, while the CGA point entity $\mathbf{P}$ is usually not dualized since the metric is non-degenerate and $(\mathbf{P} \wedge \mathbf{A})/\mathbf{I}_5 = \mathbf{P} \cdot (\mathbf{A}/\mathbf{I}_5)$. If $\mathbf{P} \wedge \mathbf{A}$ is a grade k, then $\mathbf{P} \cdot (\mathbf{A}/\mathbf{I}_5)$ is grade $5-k$.

However, in PGA we have to avoid the degenerate inner product. Both surface $\mathbf{A}$ and point $\mathbf{P}$ are dualized from OPNS PGA to CPNS PGA as $J_e(\mathbf{A})$ and $J_e(\mathbf{P})$, and then $\mathbf{P} \wedge \mathbf{A}$ (of grade k) is taken again from the geometric product of these duals as $\langle J_e(\mathbf{P}) J_e(\mathbf{A}) \rangle_k = \mathbf{P} \wedge \mathbf{A}$. This grade k part is not given by the inner product, but by the commutator product $\times$. Thus, the same null space entity $\mathbf{P} \wedge \mathbf{A}$ is obtained in OPNS PGA and in CPNS PGA, and degenerate inner products are avoided.

In $\mathcal{G}_{3,0,1}$, when we dualize the OPNS PGA 1-blade point, we obtain the dual CPNS PGA 3-blade point, and we take its geometric product with another dual grade 3 point, or dual grade 2 line, or dual grade 1 plane, but in each case the part of the geometric product that does not include the inner product is the commutator product, which gives the correct grade k part for the null space. The grade k product resulting from using the commutator product of the dual CPNS PGA entities is the same as the outer product of the corresponding OPNS PGA entities, and the geometric significance is the same. For this reason, we call the dual entities, dual to the OPNS PGA entities, the Commutator Product Null Space (CPNS) PGA entities.

The CPNS PGA entities have a correspondence with IPNS CGA entities, except that no actual correspondence exists for the CGA point at infinity $\mathbf{e}_\infty$. The CGA $\mathbf{e}_\infty$ is replaced in CPNS PGA by what may be called its pseudo-correspondence, $\mathbf{e}_\infty \leftrightarrow J_e(-\mathbf{I}_3) = \mathbf{e}_0$. We cannot dualize by multiplication or division by the PGA unit pseudoscalar $\mathbf{I}_4$ since it is a null pseudoscalar. The PGA entity dualization operation J_e has to be provided as a special operation. As mentioned before, the complete details of the PGA entity dualization operation J_e are given in Section 4.

It should be pointed out that, some may argue that using the degenerate algebra $\mathcal{G}_{3,0,1}$ is unnecessarily complicated, and whenever one wants to use a null vector, then use a non-degenerate null vector formed by adding an algebra of the Minkowski plane $\mathcal{G}_{1,1}$ with unit vectors $\{\mathbf{e}_+, \mathbf{e}_-\}$, where $\mathbf{e}_+ + \mathbf{e}_-$ and $-\mathbf{e}_+ + \mathbf{e}_-$ are non-degenerate null vectors. The PGA null vector $\mathbf{e}_0$ is degenerate, and makes the algebra degenerate, since its inner product with any other basis blade is 0, including with the PGA unit pseudoscalar $\mathbf{I}_4$. Then, the usual dualization of an element A as $A^* = A\mathbf{I}_4^{-1}$ cannot work in PGA since $\mathbf{I}_4^{-1}$ does not exist. However, for any element $A \in \mathcal{G}_3$, in the subalgebra $\mathcal{G}_3$, we have the special dualization $J_e(A) = -\mathbf{e}_0 A^* = -\mathbf{e}_0(A/\mathbf{I}_3)$ in PGA. Otherwise, we require the entity dualization operation J_e to dualize any element in PGA. The non-degenerate null vector

$\mathbf{e}_+ + \mathbf{e}_-$ has non-zero inner products with other vectors in the Minkowski (hyperbolic) plane, but its square, or inner product with any multiple of itself, is 0. So, if we have $\mathcal{G}_3$ and also want to have a null vector like $\mathbf{e}_0$, then we could use $\mathcal{G}_{3+1,1} = \mathcal{G}_{4,1}$, perhaps with $\mathbf{e}_0 = \frac{1}{2}(-\mathbf{e}_+ + \mathbf{e}_-) = \mathbf{e}_o$, and this is the algebra of CGA, so we might as well also use $\mathbf{e}_\infty = \mathbf{e}_+ + \mathbf{e}_-$. CGA is a larger algebra requiring more computations than PGA. Though if only points, lines, and planes are utilized in CGA, then CGA acts almost like PGA, yet it still has more computational complexity since it still retains more basis blades to compute.

Even if CGA is used only for points, lines, and planes, there are at least three differences compared to PGA: (1) While CGA has a single point-at-infinity entity $\mathbf{e}_\infty$, PGA has directed points at infinity represented by unit 3D vectors $\hat{\mathbf{n}}$ and has no corresponding element to the CGA $\mathbf{e}_\infty$. (2) The OPNS CGA surface entities are one grade larger than the OPNS PGA entities, increasing the computations required. However, the OPNS PGA line and plane entities dualize to the same grades and forms as their corresponding entities in IPNS CGA. (3) In CGA, we do not dualize the OPNS CGA point entity (it remains a vector), while in PGA we dualize the OPNS PGA vector point entity into a CPNS PGA 3-blade point entity, which is multiplied with the dual surface entities by commutator product. In the dual CPNS PGA forms, using a 3-blade point and the commutator product, the computations may not be much more efficient than the inner products of the corresponding IPNS CGA entities. It is beyond the scope of this paper to analyze and compare PGA to CGA computational efficiency, which would also depend on the software implementation. PGA and CGA are similar for some entities and very different in other ways. We will not make too many more comparisons.

This paper is organized as follows. In Section 2, we discuss the Point-based algebra of OPNS entities in PGA, which we call OPNS PGA. The point-based algebra allows points to join (span) by outer product into lines and planes. In Section 3, we discuss the Plane-based algebra of CPNS entities in PGA, which we call CPNS PGA. The plane-based algebra allows planes to meet (intersect) by outer product into lines and points. In Section 4, we develop the new PGA entity dualization operation J_e that dualizes the OPNS PGA entities into CPNS entities. The dualization operation seems to have been a difficult problem in the prior literature, and the new operation J_e appears to solve the dualization problem for PGA $\mathcal{G}_{3,0,1}$. In Section 5, we explore the details of the Dual Quaternion Geometric Algebra (DQGA) within the even-grades subalgebra $\mathcal{G}^+_{3,0,1}$ of PGA. In DQGA, we rediscover many results that may be known in older published literature, while there may be some new results on representing lines and planes and various operations on them that are derived through identities to the CPNS PGA entities and operations. In Section 6, we discuss Double PGA $\mathcal{G}_{6,0,2}$ (DPGA) in which the main result is the ability to form a general quadric entity as the bivector $\boldsymbol{\omega}$. Within DPGA, the DQGA is also doubled into Double DQGA (DDQGA), in which again the main result is a general quadric entity ω based closely on $\boldsymbol{\omega}$, as $\omega = \mathbf{I}_3\boldsymbol{\omega}$. In Section 7, the paper concludes with final remarks.

2 OPNS Point-based Geometric Algebra

In this section, we discuss the Outer Product Null Space Point-based Geometric Algebra $\mathcal{G}_{3,0,1}$, OPNS PGA. In this geometrical interpretation of $\mathcal{G}_{3,0,1}$, a point entity $\mathbf{P}$ and a surface entity $\boldsymbol{\Pi}$ are multiplied by the outer product $\wedge$ as $\mathbf{P} \wedge \boldsymbol{\Pi}$, with the geometrical interpretation that point $\mathbf{P}$ is a point of the surface $\boldsymbol{\Pi}$ if and only if $\mathbf{P} \wedge \boldsymbol{\Pi} = 0$.

A geometric *entity* is an element of the algebra that embeds or represents a geometric object (e.g., point, line, plane). The most useful forms of geometric entities allow for the products of entities to yield an element of the algebra that expresses the geometrical relationships between the entities, and also allow for one or more versors of the algebra to act on the entities as transformation operators (e.g., rotation, translation).

The algebra of this section is similar to the "Algebra in Projective Space" (Section 7.4 in [4]), that uses $\mathcal{G}_{1,3,0}$. Any metric can be used for an algebra in projective 4D space, but in this paper we use $\mathcal{G}_{3,0,1}$ which makes the inner product degenerate, but still allows for a rotation operation. Since it is degenerate, we have to avoid using Inner Product Null Space (IPNS) as explained in Section 1. This algebra, using the wedge product of homogenized vector (1-blade) points, is also sometimes called Projective Geometric Algebra.

2.1 OPNS PGA Introduction

The Outer Product Null Space Point-based Geometric Algebra $\mathcal{G}_{3,0,1}$ (OPNS PGA) is the Projective Geometric Algebra $\mathcal{G}_{3,0,1}$ of homogeneous (or homogenized) points of the form $\mathbf{P_p} = \mathbf{e}_0 + \mathbf{p}$, where $\mathbf{e}_0$ is a degenerate null unit basis vector (a basis 1-blade *defined* so that $\mathbf{e}_0 \cdot \mathbf{e}_0 = 0$), and $\mathbf{p} = p_x\mathbf{e}_1 + p_y\mathbf{e}_2 + p_z\mathbf{e}_3$ is a Euclidean vector in $\mathcal{G}_3$ with metric $\mathbf{e}_i \cdot \mathbf{e}_j = \text{diag}(1, 1, 1)$ for $i, j \in \{1, 2, 3\}$. We can scale $\mathbf{P}$ by any real scalar $t \neq 0$ as $t\mathbf{P}$ without changing the point represented, which is the essence of what "homogeneous" means in this context. All of the geometric entities in PGA are homogeneous geometric entities. The point $\mathbf{P}$ is also called a projective point, representing a ray (or line) $t\mathbf{P}$, $t \neq 0$, through the origin and passing through the hyperplane $\mathbf{e}_0$ at $\mathbf{p}$. Points in 3D are where these rays intersect, or project in the sense of light rays, onto the hyperplane $\mathbf{e}_0$. A point $t\mathbf{P}$ is "projected" by dividing by t to normalize the homogeneous component $\mathbf{e}_0$ and then taking the 3D vector part $\mathbf{p}$. The coordinates of $\mathbf{P}$, $(1, x, y, z)$, are called *homogeneous coordinates*.

In Linear Algebra, homogeneous coordinates, often in the form of a column vector $\mathbf{P} = [x, y, z, 1]^{\mathrm{T}}$, are well-known for their use in computer *graphics pipeline* calculations, where they are transformed by a 4×4 matrix $\mathbf{M}$ as $\mathbf{P}' = \mathbf{MP}$. A matrix $\mathbf{M}$ may be the composition of various transformations, including change-of-basis, rotation, translation, scaling, shearing, and orthogonal or perspective projections into a view volume.

In OPNS PGA, we have not only a homogeneous point representation $\mathbf{P} = \mathbf{e}_0 + \mathbf{p}$ but also a line $\mathbf{L} = \mathbf{P}_1 \wedge \mathbf{P}_2$ and a plane $\mathbf{\Pi} = \mathbf{P}_1 \wedge \mathbf{P}_2 \wedge \mathbf{P}_3$ represented by the outer products of two or three homogeneous points, respectively, which is arguably the most useful feature of OPNS PGA and why it is called the *point-based* algebra of PGA. However, transformation operations are limited in OPNS PGA to only rotation using a rotation versor $R = \exp\left(\frac{1}{2}\theta\hat{\mathbf{n}}/\mathbf{I}_3\right)$.

In Section 2.2, we derive and discuss the OPNS PGA geometric entities for 1-blade point $\mathbf{P}$, 2-blade line $\mathbf{L}$, and 3-blade plane $\mathbf{\Pi}$. In Section 2.3, we discuss some of the OPNS PGA operations that act on the OPNS PGA entities, including the rotation operation using the rotation operator R, and also translation, projection, and intersection operations using the PGA entity dualization operation J_e that is introduced in Section 4.

2.2 OPNS PGA Geometric Entities

In this section, we derive and discuss the three OPNS PGA homogeneous geometric entities: OPNS PGA 1-blade point $\mathbf{P}$, OPNS PGA 2-blade line $\mathbf{L}$, and OPNS PGA 3-blade plane $\mathbf{\Pi}$.

We will use many geometric algebra identities, including:

$$\mathbf{I}_4 = -\mathbf{I}_3\mathbf{e}_0, \tag{1}$$

$$\mathbf{b}^* = \mathbf{b}/\mathbf{I}_3 = -\mathbf{b}\mathbf{I}_3, \tag{2}$$

for the 2-blade *dual* $\mathbf{b}^*$ of 1-blade $\mathbf{b}$ in $\mathcal{G}_3$,

$$\mathbf{aB} = \mathbf{a}\cdot\mathbf{B} + \mathbf{a}\wedge\mathbf{B} = \frac{1}{2}(\mathbf{aB} - (-1)^k\mathbf{Ba}) + \frac{1}{2}(\mathbf{aB} + (-1)^k\mathbf{Ba}), \tag{3}$$

for the product $\mathbf{aB}$ of 1-vector $\mathbf{a}$ and k-vector $\mathbf{B}$,

$$(\mathbf{a}\cdot\mathbf{b})\mathbf{I}_3 = \frac{1}{2}(\mathbf{ab} + \mathbf{ba})\mathbf{I}_3 = -\frac{1}{2}(\mathbf{ab}^* + \mathbf{b}^*\mathbf{a}) = -\mathbf{a}\wedge\mathbf{b}^*, \tag{4}$$

$$(\mathbf{a}\cdot\mathbf{b})\mathbf{I}_4 = \mathbf{a}\wedge\mathbf{b}^*\wedge\mathbf{e}_0, \tag{5}$$

and

$$(\mathbf{a}\cdot\mathbf{b}^*)\mathbf{I}_4 = -\frac{1}{2}(\mathbf{ab}^* - \mathbf{b}^*\mathbf{a})\mathbf{I}_3\mathbf{e}_0 = \mathbf{a}\wedge\mathbf{e}_0\wedge\mathbf{b}. \tag{6}$$

2.2.1 OPNS PGA 1-blade Point Entity

The OPNS PGA 1-blade point entity $\mathbf{P_t}$ is defined as

$$\mathbf{P_t} = \mathbf{e}_0 + \mathbf{t}, \tag{7}$$

where $\mathbf{t} = x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3$. The OPNS point $\mathbf{P_t}$ represents homogeneous coordinates $(1, x, y, z)$ and is called a homogenized 3D vector. We will also use the notation $\mathbf{P_p}$ for a point that embeds the vector $\mathbf{p}$.

We consider $\mathbf{P_t}$ to be the symbolic test point, where x, y, z are symbolic variables or symbols. An actual numerical point is $\mathbf{P_p} = \mathbf{e}_0 + \mathbf{p} = \mathbf{e}_0 + p_x\mathbf{e}_1 + p_y\mathbf{e}_2 + p_z\mathbf{e}_3$, where p_x, p_y, p_z are real numbers. This distinction between the symbolic test point $\mathbf{P_t}$ and other numerical points, such as $\mathbf{P_p}$, is used in symbolic calculations that help to analyze and understand the algebra and its products.

Just like for homogeneous coordinates, we may scale a point $\mathbf{P_p}$ with any real number $t \neq 0$ as $t\mathbf{P}$, but we must remember that our actual 3D point $\mathbf{p}$ is in the hyperplane $\mathbf{e}_0$.

To obtain $\mathbf{p}$ from $t\mathbf{P_p}$, we must project $t\mathbf{P_p}$ onto the hyperplane $\mathbf{e}_0$ by normalizing the $\mathbf{e}_0$ component of $\mathbf{P_p}$ (dividing $t\mathbf{P_p}/t = \mathbf{P_p}$) and then taking $\mathbf{p}$. To extract the scalar t from $t\mathbf{e}_0$ requires that we have a special operation for this scalar extraction, or a dualization operation that dualizes $\mathbf{e}_0$ to a non-degenerate element of the algebra that can be contracted by its inverse to leave t. The entity dualization operation J_e is discussed in Section 4.

The homogeneous component $\mathbf{e}_0$ represents the point at the origin $(x, y, z) = (0, 0, 0)$. This is similar to the CGA point at the origin $\mathbf{e}_o$. Recall that a CGA point has the form $\mathcal{C}(\mathbf{t}) = \mathbf{t} + \frac{1}{2}\mathbf{t}^2\mathbf{e}_\infty + \mathbf{e}_o$. In PGA, the CGA $\mathbf{e}_\infty$, representing the point at infinity, has no corresponding counterpart or element. Instead of having "the point" at infinity, we have the unit sphere of directional 3D vectors $\hat{\mathbf{n}}$, representing directed points at infinity: $\lim_{\|\mathbf{n}\|\to\infty} \frac{(\mathbf{e}_0 + \mathbf{n})}{\|\mathbf{n}\|} = \frac{\mathbf{n}}{\|\mathbf{n}\|} = \hat{\mathbf{n}}$. Although we cannot scale a point $\mathbf{P_n} = \mathbf{e}_0 + \mathbf{n}$ by $t = 0$, we can scale $\mathbf{P_n}$ by $\frac{1}{\|\mathbf{n}\|}$ in the limit as $\|\mathbf{n}\| \to \infty$ and then $\mathbf{P_n} \neq 0$, which is what is required. Further non-zero scaling of the directional unit vector $\hat{\mathbf{n}}$ representing a directed point at infinity is still permissible. Therefore, more generally, any 3D vector $\mathbf{n}$ represents, by its direction regardless of magnitude, a directed point at infinity. In CGA, we have this limit: $\lim_{\mathbf{t}^2\to\infty} 2\mathcal{C}(\mathbf{t})/\mathbf{t}^2 = \mathbf{e}_\infty$, which is a single point at infinity that has no counterpart in PGA.

2.2.2 OPNS PGA 2-blade Line Entity

The PGA OPNS 2-blade line $\mathbf{L}$ is constructed from basic geometric principles: Given two points on the line as 3D vectors, $\mathbf{p}_1$ and $\mathbf{p}_2$, we form the displacement $\mathbf{d}$ from $\mathbf{p}_1$ to $\mathbf{p}_2$ as $\mathbf{d} = \mathbf{p}_2 - \mathbf{p}_1$. Given any third point $\mathbf{p}_3 = \mathbf{t}$, we can test if it is on the line of $\mathbf{p}_1$ and $\mathbf{p}_2$ as follows. We take the displacement $\mathbf{t} - \mathbf{p}_1$, and note that it should be parallel to $\mathbf{d}$ if $\mathbf{t}$ is on the line. To test this parallel condition, we can dualize the direction $\mathbf{d}$ as $\mathbf{d}^* = \mathbf{d}/\mathbf{I}_3$ and note that the projection of $\mathbf{d}$, or any vector parallel to $\mathbf{d}$, onto the plane through the origin represented by $\mathbf{d}^*$ is 0. If the displacement $\mathbf{t} - \mathbf{p}_1$ projects onto $\mathbf{d}^*$ as 0, then $\mathbf{t}$ is on the line. The projection is $((\mathbf{t} - \mathbf{p}_1) \cdot \mathbf{d}^*)\mathbf{d}^{*-1} = -((\mathbf{t} - \mathbf{p}_1) \cdot \mathbf{d}^*)\hat{\mathbf{d}}^* \|\mathbf{d}\|^{-1}$. The RHS $\mathbf{d}^{*-1}$, which performs a counterclockwise rotation in the plane by 90° around $\hat{\mathbf{d}}$ and scales by $\|\mathbf{d}\|^{-1}$, is not important for the test. Therefore, we choose to abridge the vector projection by $\hat{\mathbf{d}}^* \|\mathbf{d}\|^{-1}$ to $-(\mathbf{t} - \mathbf{p}_1) \cdot \mathbf{d}^* = (\mathbf{p}_1 - \mathbf{t}) \cdot \mathbf{d}^*$ as a vector-valued null space test condition. Notice that, we do not abridge the minus sign, so that we maintain the orientation of the projection. As geometric entities, the two points that define the line are $\mathbf{P}_1 = \mathbf{e}_0 + \mathbf{p}_1$ and $\mathbf{P}_2 = \mathbf{e}_0 + \mathbf{p}_2$, the test point is $\mathbf{P}_\mathbf{t} = \mathbf{e}_0 + \mathbf{t}$, and we seek to derive the line entity $\mathbf{L}$. Now, we use the identity $(\mathbf{a} \cdot \mathbf{b}^*)\mathbf{I}_4 = \mathbf{a} \wedge \mathbf{e}_0 \wedge \mathbf{b}$ and dualize the vector-valued null space $(\mathbf{p}_1 - \mathbf{t}) \cdot \mathbf{d}^*$ by multiplication with the PGA unit pseudoscalar $\mathbf{I}_4$ to produce the geometric null space entity $\mathbf{P}_\mathbf{t} \wedge \mathbf{L} = ((\mathbf{p}_1 - \mathbf{t}) \cdot \mathbf{d}^*)\mathbf{I}_4 = (\mathbf{p}_1 - \mathbf{t}) \wedge (\mathbf{e}_0 \wedge \mathbf{d})$. Therefore, $(\mathbf{e}_0 + \mathbf{t}) \wedge \mathbf{L} = -\mathbf{e}_0 \wedge \mathbf{p}_1 \wedge \mathbf{d} - \mathbf{t} \wedge \mathbf{e}_0 \wedge \mathbf{d}$. Let $\mathbf{L} = -\mathbf{e}_0 \wedge \mathbf{d} - \mathbf{p}_1 \wedge \mathbf{d} = -(\mathbf{e}_0 + \mathbf{p}_1) \wedge \mathbf{d} = \mathbf{d} \wedge \mathbf{P}_1$ and note that $\mathbf{t} \wedge \mathbf{p}_1 \wedge \mathbf{d} = 0$ if $\mathbf{t}$ is on the line. We also have $\mathbf{L} = \mathbf{d} \wedge (\mathbf{e}_0 + \mathbf{p}_1) = (\mathbf{P}_2 - \mathbf{P}_1) \wedge \mathbf{P}_1 = \mathbf{P}_2 \wedge \mathbf{P}_1$. If we like, we can normalize $\mathbf{d}$ as $\hat{\mathbf{d}} = \mathbf{d}/\sqrt{\mathbf{d}^2}$ and think of it as a unit direction vector or as a unit directed point at infinity. We can now define the PGA OPNS 2-blade line entity $\mathbf{L}$ containing points $\mathbf{P}_1$ and $\mathbf{P}_2$ as

$$\mathbf{L} = \mathbf{P}_2 \wedge \mathbf{P}_1, \tag{8}$$

or as the line through points $\mathbf{P}$ and $\hat{\mathbf{d}}$ (a directed point at infinity, or direction through $\mathbf{P}$) as

$$\mathbf{L} = \hat{\mathbf{d}} \wedge \mathbf{P}. \tag{9}$$

It may seem odd that we write the points in reverse, but this maintains the orientation of the line such that the line represents an axis of rotation around $\mathbf{d}$ when we dualize to the CPNS PGA 2-blade line $\boldsymbol{l} = J_e(\mathbf{L})$, where $\exp(\theta\hat{\boldsymbol{l}}/2)$ is a rotor for counterclockwise rotation around $\boldsymbol{l}$ by angle θ with $\hat{\mathbf{d}}$ as the axis of rotation through $\boldsymbol{l}$ in the sense of the right-hand rule. We will also use the notation $\mathbf{L}_{\mathbf{p},\hat{\mathbf{d}}}$ for the line through point $\mathbf{P}_\mathbf{p}$ in direction $\hat{\mathbf{d}}$.

Note that, the condition $(\mathbf{t} - \mathbf{p}_1) \cdot \mathbf{d}^* = 0$ is the essence of Plücker coordinates $(d_1, d_2, d_3 : m_1, m_2, m_3) = (\mathbf{d}^* : \mathbf{p}_1 \cdot \mathbf{d}^*)$ for the line, where any point $\mathbf{t}$ is on the line if $\mathbf{t} \cdot \mathbf{d}^* - \mathbf{p}_1 \cdot \mathbf{d}^* = 0$. The correspondence ($\hat{=}$) between vector calculus and geometric algebra is $\mathbf{a} \times \mathbf{b} \hat{=} ((\mathbf{a} \wedge \mathbf{b})/\mathbf{I}_3)^* = (\mathbf{a} \cdot (\mathbf{b}/\mathbf{I}_3))^* = (\mathbf{a} \cdot \mathbf{b}^*)^*$ and $\mathbf{a} \hat{=} \mathbf{a}^* = \mathbf{a}/\mathbf{I}_3$, so in vector calculus $(d_1, d_2, d_3 : m_1, m_2, m_3) = (\mathbf{d} : \mathbf{p}_1 \times \mathbf{d}) = (\mathbf{d} : \mathbf{p}_1 \times \mathbf{p}_2) = (\mathbf{d} : \mathbf{m})$ with point $\mathbf{t}$ on the line if $\mathbf{t} \times \mathbf{d} - \mathbf{p}_1 \times \mathbf{d} = \mathbf{t} \times \mathbf{d} - \mathbf{m} = 0$, or if $\mathbf{t} \times \mathbf{d} = \mathbf{m}$. We distinguish the vector calculus cross product symbol $\times$ (bold cross) from the geometric algebra commutator product symbol $\times$ (cross), $A \times B = \frac{1}{2}(AB - BA)$.

In CGA $\mathcal{G}_{4,1}$, we dualize the line condition $(\mathbf{t}-\mathbf{p}_1)\cdot\mathbf{d}^*=0$ as $((\mathbf{t}-\mathbf{p}_1)\cdot\mathbf{d}^*)\mathbf{I}_5=0$, which yields a 4-blade test condition and a 3-blade line entity of the form $\mathbf{P}_1\wedge\mathbf{P}_2\wedge\mathbf{e}_\infty=\mathbf{P}_1\wedge\mathbf{d}\wedge\mathbf{e}_\infty$. CGA has a single point at infinity $\mathbf{e}_\infty$ which comes from the spherical inversion, or reflection, of the center point $\mathbf{C}$ of any sphere $\mathbf{S}$ in the same sphere as $-2\mathbf{S}\mathbf{C}\mathbf{S}^{-1}=\mathbf{e}_\infty$. In CGA, every line includes the point $\mathbf{e}_\infty$. In PGA, our OPNS line $\mathbf{L}$ only includes the two directed points at infinity $\hat{\mathbf{d}}$ and $-\hat{\mathbf{d}}$. Note that, assuming the CGA points $\mathbf{P}_1$ and $\mathbf{P}_2$ are unit scale in $\mathbf{P}_1\wedge\mathbf{P}_2\wedge\mathbf{e}_\infty$, we can make the substitution $\mathbf{P}_1\wedge(\mathbf{P}_2-\mathbf{P}_1)\wedge\mathbf{e}_\infty=\mathbf{P}_1\wedge(\mathbf{p}_2-\mathbf{p}_1)\wedge\mathbf{e}_\infty=\mathbf{P}_1\wedge\mathbf{d}\wedge\mathbf{e}_\infty$ since $\frac{1}{2}(\mathbf{p}_2^2-\mathbf{p}_1^2)\mathbf{e}_\infty\wedge\mathbf{e}_\infty=0$.

2.2.3 OPNS PGA 3-blade Plane Entity

Given any point $\mathbf{p}$ on the plane $\mathbf{\Pi}$ and the unit direction vector $\hat{\mathbf{n}}$ normal to the plane (so that $\mathbf{p}$ and $\hat{\mathbf{n}}$ define the plane), then the scalar null space test condition that a point $\mathbf{t}$ is on the plane is $\mathbf{t}\cdot\hat{\mathbf{n}}-\mathbf{p}\cdot\hat{\mathbf{n}}=0$, where $\mathbf{p}\cdot\hat{\mathbf{n}}=d$ is the distance of the plane from the origin and $\mathbf{t}\cdot\hat{\mathbf{n}}-\mathbf{p}\cdot\hat{\mathbf{n}}$ is the distance of $\mathbf{t}$ from the plane. As geometric entities, the plane point $\mathbf{p}$ is $\mathbf{P}_\mathbf{p}=\mathbf{e}_0+\mathbf{p}$, the test point $\mathbf{t}$ is $\mathbf{P}_\mathbf{t}=\mathbf{e}_0+\mathbf{t}$, and we seek to derive the plane entity $\mathbf{\Pi}$. To put the scalar null space test condition on a geometric basis as a geometric null space entity, we use the identity $(\mathbf{a}\cdot\mathbf{b})\mathbf{I}_4=\mathbf{a}\wedge\mathbf{b}^*\wedge\mathbf{e}_0$ and multiply $\mathbf{t}\cdot\hat{\mathbf{n}}-\mathbf{p}\cdot\hat{\mathbf{n}}$ by the PGA unit pseudoscalar $\mathbf{I}_4$ as $\mathbf{P}_\mathbf{t}\wedge\mathbf{\Pi}=(\mathbf{t}\cdot\hat{\mathbf{n}}-\mathbf{p}\cdot\hat{\mathbf{n}})\mathbf{I}_4=\mathbf{t}\wedge\hat{\mathbf{n}}^*\wedge\mathbf{e}_0-\mathbf{p}\wedge\hat{\mathbf{n}}^*\wedge\mathbf{e}_0$. Then, $\mathbf{P}_\mathbf{t}\wedge\mathbf{\Pi}=\mathbf{t}\wedge\mathbf{e}_0\wedge\hat{\mathbf{n}}^*+\mathbf{e}_0\wedge\mathbf{p}\wedge\hat{\mathbf{n}}^*$. Let $\mathbf{\Pi}=\mathbf{e}_0\wedge\hat{\mathbf{n}}^*+\mathbf{p}\wedge\hat{\mathbf{n}}^*=\mathbf{P}_\mathbf{p}\wedge\hat{\mathbf{n}}^*$. Therefore, the PGA OPNS 3-blade plane entity $\mathbf{\Pi}$ representing the plane through point $\mathbf{P}$ and normal to the unit direction vector $\hat{\mathbf{n}}$ is defined as

$$\mathbf{\Pi}=\mathbf{P}\wedge\hat{\mathbf{n}}^*. \tag{10}$$

Normalizing the plane normal vector $\mathbf{n}$ is not required, but is the usual practice. Now, we can make a useful substitution of the plane bivector $\mathbf{n}^*$ as follows: Given three unit scale points on the plane $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ (assuming these points are arranged counterclockwise on the plane to fix the orientation), it is easy to see that the bivector $\mathbf{n}^*$ can be formed as $\mathbf{n}^*=(\mathbf{P}_3-\mathbf{P}_1)\wedge(\mathbf{P}_2-\mathbf{P}_1)$. Then, we have $\mathbf{\Pi}=\mathbf{P}_1\wedge(\mathbf{P}_3-\mathbf{P}_1)\wedge(\mathbf{P}_2-\mathbf{P}_1)=\mathbf{P}_1\wedge\mathbf{P}_3\wedge\mathbf{P}_2$, with the points ordered clockwise in this 3-blade product while the points on the plane are still geometrically arranged counterclockwise. The points do not actually have to be unit scale. Therefore, the PGA OPNS 3-blade plane entity $\mathbf{\Pi}$ for the plane of three points $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$, where these points are arranged *clockwise* on the plane as viewed from over the plane's normal direction $\hat{\mathbf{n}}$, is defined as

$$\mathbf{\Pi} \;=\; \mathbf{P}_1\wedge\mathbf{P}_2\wedge\mathbf{P}_3. \tag{11}$$

Alternatively, if the points are arranged counterclockwise on the plane, then the plane is

$$\mathbf{\Pi} \;=\; \mathbf{P}_3\wedge\mathbf{P}_2\wedge\mathbf{P}_1. \tag{12}$$

We will also use the notation $\mathbf{\Pi}_{\mathbf{p},\hat{\mathbf{n}}}=\mathbf{P}_\mathbf{p}\wedge\hat{\mathbf{n}}$ for the plane through point $\mathbf{P}_\mathbf{p}$ with normal $\hat{\mathbf{n}}$. Using the identity $\mathbf{a}\wedge\mathbf{b}^*=-(\mathbf{a}\cdot\mathbf{b})\mathbf{I}_3$ we can write

$$\mathbf{\Pi}_{d,\hat{\mathbf{n}}}=\mathbf{e}_0\wedge\hat{\mathbf{n}}^*+\mathbf{p}\wedge\hat{\mathbf{n}}^*=\mathbf{e}_0\wedge\hat{\mathbf{n}}^*-(\mathbf{p}\cdot\hat{\mathbf{n}})\mathbf{I}_3 \tag{13}$$

for the plane through point $\mathbf{p}$ at distance $d = \mathbf{p} \cdot \hat{\mathbf{n}}$ from the origin.

2.3 OPNS PGA Operations

The OPNS PGA operations include rotation and, via entity dualization using J_e to CPNS PGA plane-based entities, translation. The OPNS PGA point-based entities support the join (of points) operation as the wedge product of 2 or 3 points to form a line or plane. The meet (of planes) operation is also possible via dualization to plane-based entities, where the wedge product of 2 or 3 planes forms a line or plane.

2.3.1 OPNS PGA 2-versor Rotation Operator

A versor V in geometric algebra is a geometric product one or more vectors $V = \mathbf{a}_3\mathbf{a}_2\mathbf{a}_1\ldots$ having inverses $\mathbf{a}^{-1}$. We call a single vector $V = \mathbf{a}$ with an inverse (any non-null vector) a 1-versor V, the product of two vectors is a 2-versor $V = \mathbf{a}_2\mathbf{a}_1$ and so on. Versors are, in general, operators for transforming an element A as $A' = VAV^{-1} = \ldots\mathbf{a}_3\mathbf{a}_2\mathbf{a}_1A\mathbf{a}_1^{-1}\mathbf{a}_2^{-1}\mathbf{a}_3^{-1}\ldots$, called a versor "sandwich" operation or product, and represents successive reflections in vectors (composing $A^{i'} = -\mathbf{a}_iA\mathbf{a}_i^{-1}$ is successive reflections in hyperplanes).

The versor (version operator, or product of vectors having inverses) for the transformation of vector $\mathbf{a}$ into the vector $\mathbf{b}$ is the ratio $\mathbf{b}/\mathbf{a} = \mathbf{b}\mathbf{a}^{-1}$, which is a 2-versor. Assume these are unit vectors, then $(\mathbf{b}/\mathbf{a})\mathbf{a} = \mathbf{b}$ is only rotation in the $\mathbf{ab}$-plane, and we can write $\mathbf{b}/\mathbf{a} = \mathbf{b}\mathbf{a} = \mathbf{b}\cdot\mathbf{a} + \mathbf{b}\wedge\mathbf{a} = \cos(\theta) + \sin(\theta)\hat{\mathbf{n}}^* = \exp(\theta\hat{\mathbf{n}}^*) = R^2$, where $\hat{\mathbf{n}}^* = (\mathbf{b}\wedge\mathbf{a})/\|\mathbf{b}\wedge\mathbf{a}\|$ and $\hat{\mathbf{n}}^*\mathbf{I}_3 = \hat{\mathbf{n}}$ is the axis of rotation. The rotor R^2 can be applied to any vector $\mathbf{c}$ in the $\mathbf{ab}$-plane as just $\mathbf{c}' = R^2\mathbf{c}$ to rotate $\mathbf{c}$ by angle θ (the angle from $\mathbf{a}$ to $\mathbf{b}$) in the $\mathbf{ab}$-plane. $R = \exp(\theta\hat{\mathbf{n}}^*/2)$ rotates by $\theta/2$. To rotate any vector $\mathbf{v}$ around the axis $\mathbf{n}$, we must leave the component of $\mathbf{v}$ parallel to $\hat{\mathbf{n}}$, $\mathbf{v}^{\|}$, unchanged and rotate only the component of $\mathbf{v}$ perpendicular to $\hat{\mathbf{n}}$, $\mathbf{v}^{\perp}$. Therefore, we can see that $R\mathbf{v}R^{-1} = R(\mathbf{v}^{\|} + \mathbf{v}^{\perp})R^{-1} = RR^{-1}\mathbf{v}^{\|} + R^2\mathbf{v}^{\perp}$ is $\mathbf{v}$ rotated by angle θ around axis $\hat{\mathbf{n}}$. The rotation can also be seen as successive reflections in the vector $\mathbf{a}$ then in $\mathbf{b}$ as $\mathbf{v}'' = \mathbf{bava}^{-1}\mathbf{b}^{-1}$, which rotates $\mathbf{v}$ by 2θ. One interpretation is reflection in the line (through the origin) of $\mathbf{a}$, $\mathbf{v}' = \mathbf{ava}^{-1}$, and then in the line of $\mathbf{b}$, $\mathbf{v}'' = \mathbf{bv}'\mathbf{b}^{-1}$. Another interpretation is reflection in the plane (through the origin) perpendicular to $\mathbf{a}$, $\mathbf{v}' = -\mathbf{ava}^{-1}$, and then in the plane perpendicular to $\mathbf{b}$, $\mathbf{v}'' = -\mathbf{bv}'\mathbf{b}^{-1}$ (these two planes intersect in the line of axis $\hat{\mathbf{n}}$). For any product of elements $AB\ldots$, we can apply the versor operator R as $RAB\ldots R^{-1} = RAR^{-1}RB\ldots R^{-1}$, inserting $R^{-1}R = 1$ anywhere we choose, which is called versor outermorphism. The outermorphism applies to all parts of the geometric product, including the outer product. The OPNS PGA entities are point-based, formed as outer products of the point entities. By outermorphism, the rotor R can be applied to any of these entities, point $\mathbf{P}$, line $\mathbf{L}$, or plane $\mathbf{\Pi}$, and R rotates each point, $R\mathbf{P}R^{-1} = R\mathbf{p}R^{-1} + \mathbf{e}_0$, in the entity and therefore rotates the entire entity. In summary:

The rotation operator (rotor) R, for rotation relative to the origin around axis $\hat{\mathbf{n}}$ by angle θ, is

$$R = \exp\left(\frac{\theta}{2}\hat{\mathbf{n}}^*\right) = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\hat{\mathbf{n}}^*. \tag{14}$$

The rotor R is applied to any element A of the algebra as the versor "sandwich" operation

$$A' \;=\; RAR^{-1}, \tag{15}$$

which applies R^2 to vector components in the plane of rotation, and therefore the full angle θ of rotation.

Note that, $R^{-1} = R^{\sim} = \exp(-\theta\hat{\mathbf{n}}^*/2)$, the reverse of R. The reverse of $R^2 = \mathbf{ba}$ is $(R^2)^{\sim} = \mathbf{ab}$, reversing ($^{\sim}$) the order of all vector products (the reflections), or reversing the sign of the bivector $\hat{\mathbf{n}}^*$, which works like complex number conjugation to rotate in the reverse direction. If $\mathbf{a}$ and $\mathbf{b}$ are both unit vectors in $R^2 = \mathbf{ba}$, then $(R^2)^{\sim} = (R^2)^{-1}$, but otherwise the scale of the vectors remain in the reverse $(R^2)^{\sim}$.

It is not possible in OPNS PGA to reflect in the OPNS PGA 3-blade plane entity $\mathbf{\Pi}$, which otherwise would be able to generate translations and rotations around lines. We overcome this limitation in the CPNS PGA, where we are able to reflect in general planes and form a translation operator T acting as a versor outermorphism operator. With a translation operator T, we can then also rotate relative to points other than the origin by using translated rotors TRT^{-1}.

2.3.2 Using Entity Dualization for Other Operations

In Section 3, we discuss the CPNS PGA entities and operations. In Section 4, we discuss the geometric entity dualization operation J_e that dualizes any OPNS PGA entity into its dual CPNS PGA entity.

In CPNS PGA, we have a translation operation T as a versor operation on CPNS PGA entities. Therefore, we can translate any OPNS PGA entity in dual form as a CPNS PGA entity and then dualize the translated entity back into an OPNS PGA entity. The translation operation on an OPNS PGA entity $\mathbf{A}$ is $\mathbf{A}' = -J_e(TJ_e(\mathbf{A})T^{-1})$, where $J_e(\mathbf{A}) = \mathbf{A}^\star$ is the dual CPNS PGA entity of $\mathbf{A}$. The inverse dual ("undual") operation is $-J_e = D_e$, dualizing a CPNS PGA entity $\mathbf{A}^\star$ to its dual OPNS PGA entity $\mathbf{A} = -J_e(\mathbf{A}^\star) = \mathbf{A}^{-\star\star}$.

The intersection of two OPNS PGA 3-blade planes $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$ cannot be done directly within OPNS PGA, but we can dualize the planes to CPNS PGA 1-blade planes as $\mathbf{\Pi}_1^\star = J_e(\mathbf{\Pi}_1) = \boldsymbol{\pi}_1$ and $\mathbf{\Pi}_2^\star = J_e(\mathbf{\Pi}_1) = \boldsymbol{\pi}_2$, and then form their intersection line entity in CPNS PGA as $\mathbf{L}^\star = \boldsymbol{l} = \mathbf{\Pi}_1^\star \wedge \mathbf{\Pi}_2^\star$. The resulting line $\mathbf{L}^\star$ is then undualized to the OPNS PGA line $\mathbf{L} = -J_e(\mathbf{L}^\star) = \mathbf{L}^{-\star\star}$. The intersection can be written more compactly as

$$\mathbf{L} = (\mathbf{\Pi}_1^\star \wedge \mathbf{\Pi}_2^\star)^{-\star}, \tag{16}$$

which is also known as the *meet operation* (on planes) $\mathbf{L} = \mathbf{\Pi}_1 \vee \mathbf{\Pi}_2$. Likewise, we can form a point as the meet product of three planes as

$$\mathbf{P} = (\mathbf{\Pi}_1^\star \wedge \mathbf{\Pi}_2^\star \wedge \mathbf{\Pi}_3^\star)^{-\star}, \tag{17}$$

which may also be written as $\mathbf{P} = \mathbf{\Pi}_1 \vee \mathbf{\Pi}_2 \vee \mathbf{\Pi}_3$.

In OPNS PGA, we instead have the *join operation* (on points), such as $\mathbf{L} = \mathbf{P}_1 \wedge \mathbf{P}_2$. In CPNS PGA, we do not have the join operation on points, so it will be useful to dualize CPNS PGA points to OPNS PGA points to use the OPNS PGA join operation on points.

Any OPNS PGA entity $\mathbf{A}$ that has been dualized to its dual CPNS PGA entity $\mathbf{A}^\star = \boldsymbol{a}$ can also be transformed into its dual quaternion form a. As a dual quaternion entity a, the entity a can be operated on by all of the available dual quaternion operations as a'. The resulting entity a' can then be transformed back to CPNS PGA entity $\boldsymbol{a}'$ and then dualized to OPNS PGA entity $\mathbf{A}'$. We can use all three of the algebraic forms of any entity to take advantage of each form and its operations.

2.4 OPNS PGA Conclusion

The OPNS PGA is also called the point-based algebra of PGA. The OPNS PGA 1-blade point $\mathbf{P_t} = \mathbf{e}_0 + \mathbf{t}$ was defined as a homogeneous point. We derived the OPNS PGA 2-blade line $\mathbf{L}$ and 3-blade plane $\mathbf{\Pi}$ entities in detail, starting from the basic geometry and geometric null space conditions (Plücker coordinates or implicit surface). We were careful to consider the orientation of the entities.

We can join two or three OPNS PGA 1-blade points $\mathbf{P}$ by wedge product to form a 2-blade line $\mathbf{L} = \mathbf{P}_1 \wedge \mathbf{P}_2$ or 3-blade plane $\mathbf{\Pi} = \mathbf{P}_1 \wedge \mathbf{P}_2 \wedge \mathbf{P}_3$ which is one of its most useful features. The form of the OPNS PGA 1-blade point does not support any known translation operation, but all of the OPNS PGA entities can be rotated using the PGA rotation operator R, which is a standard form of rotation operator.

The plane-based algebra of PGA, which is discussed in the next section, is usually considered to be more interesting than the point-based algebra since it also supports a translation operator T. We can still use the point-based algebra very easily by dualizing the plane-based entities to their dual point-based entities. Therefore, we can take full advantage of the point-based *join* operations (point spanning). We can also dualize the point-based entities to their dual plane-based entities to form the *meet* of planes (plane intersection), which cannot be done directly in the point-based algebra.

3 CPNS Plane-based Geometric Algebra

This section is about the Commutator Product Null Space Plane-based Geometric Algebra $\mathcal{G}_{3,0,1}$ (CPNS PGA). We call it CPNS PGA since the plane $\boldsymbol{\pi}$, line $\boldsymbol{l}$, and point $\boldsymbol{p}$ entities each represent a plane, line, or point geometric null space $\{\boldsymbol{p_t} \times \boldsymbol{\pi}, \boldsymbol{p_t} \times \boldsymbol{l}, \boldsymbol{p_t} \times \boldsymbol{p}\}$, respectively, that is produced by the commutator product $\times$. For example, the plane $\boldsymbol{\pi}$ represents the set of points $\{\boldsymbol{p_t} \colon \boldsymbol{p_t} \times \boldsymbol{\pi} = 0\}$. We also call it Plane-based PGA since the planes can be intersected by wedge product, also called the meet product (of planes), to form the line and point entities.

3.1 CPNS PGA Introduction

The CPNS PGA is also called the plane-based algebra of PGA. For those who are familiar with Conformal Geometric Algebra (CGA) $\mathcal{G}_{4,1}$, the entities of the Plane-based CPNS PGA will appear familiar and look like the IPNS CGA entities. However, we cannot use the inner product as in CGA. Instead, we use the commutator product $\times$, or sometimes the outer product $\wedge$.

In the plane-based algebra of PGA, the meet of planes (their intersection) is by the outer product of two or three planes, forming a line or point. The plane is the primitive or primary entity in the plane-based algebra. Again, this is much like the IPNS CGA. If we want to join points by spanning, then we have to dualize the CPNS PGA entities to their dual OPNS PGA entities in the point-based algebra of PGA. Using the new entity dualization operation J_e makes this easy enough, so we can take full advantage of both the plane-based and point-based entities and algebras of PGA.

In the following sections, we explain in detail the CPNS PGA 1-blade plane $\boldsymbol{\pi}$, 2-blade line $\boldsymbol{l}$, and 3-blade point $\boldsymbol{p}_\mathbf{t}$ entities. Then, we talk about the plane-based translation operation T, and then the rotation operation R. Finally, there is a conclusion with final remarks on CPNS PGA.

3.2 CPNS PGA Geometric Entities

In this section, we derive and discuss the CPNS PGA plane-based geometric entities, which include the CPNS PGA 1-blade plane $\boldsymbol{\pi}$, 2-blade line $\boldsymbol{l}$, and 3-blade point $\boldsymbol{p}$.

3.2.1 CPNS PGA 1-blade Plane Entity

We form the CPNS PGA 1-blade plane entity $\boldsymbol{\pi}$ starting from basic geometric principles. In the 3D space, a hyperplane is just a plane with the linear implicit surface equation $xn_x+yn_y+zn_z-d=0$, where $\hat{\mathbf{n}}=n_x\mathbf{e}_1+n_y\mathbf{e}_2+n_z\mathbf{e}_3$ is the unit normal vector of the plane, point $\mathbf{t}=x\mathbf{e}_1+y\mathbf{e}_2+z\mathbf{e}_3$ is being tested for intersection with the plane, and $d=\mathbf{p}\cdot\hat{\mathbf{n}}$ is the distance of the plane from the origin, given that $\mathbf{p}$ is a point on the plane. We rewrite the equation as $\mathbf{t}\cdot\hat{\mathbf{n}}-\mathbf{p}\cdot\hat{\mathbf{n}}=0$, where $\mathbf{t}\cdot\hat{\mathbf{n}}-\mathbf{p}\cdot\hat{\mathbf{n}}=d_\mathbf{t}$ is the distance of $\mathbf{t}$ from the plane. We cannot actually work with scalar inner products in PGA since the metric is degenerate. In the OPNS PGA, we derived an entity for the plane as a 3-blade $\boldsymbol{\Pi}$. Here, in CPNS PGA, we will derive the plane entity $\boldsymbol{\pi}$ as dual-grade, as a 1-blade (vector). Similarly, the OPNS PGA point $\mathbf{P}$ is a vector, while in CPNS PGA we will develop the point $\boldsymbol{p}_\mathbf{t}$ as dual-grade, as a 3-blade. In OPNS, the pseudoscalar-valued geometric null space entity is $\mathbf{P}_\mathbf{t}\wedge\boldsymbol{\Pi}$, the outer product of a 1-blade point $\mathbf{P}_\mathbf{t}$ and 3-blade plane $\boldsymbol{\Pi}$. In CPNS PGA, we also obtain the very same geometric null space entity $\mathbf{P}_\mathbf{t}\wedge\boldsymbol{\Pi}=\boldsymbol{p}_\mathbf{t}\times\boldsymbol{\pi}=\boldsymbol{p}_\mathbf{t}\wedge\boldsymbol{\pi}$, where $\boldsymbol{p}_\mathbf{t}=J_e(\mathbf{P}_\mathbf{t})$ is the dual of $\mathbf{P}_\mathbf{t}$ and $\boldsymbol{\pi}=J_e(\boldsymbol{\Pi})$ is the dual of $\boldsymbol{\Pi}$. This avoids using the degenerate IPNS scalar metric, and instead uses the non-degenerate OPNS pseudoscalar metric in dualized form. The outer product behaves the same regardless of the scalar metric of the algebra. Now, to proceed, we know that we want a pseudoscalar geometric null space entity as the product $\boldsymbol{p}_\mathbf{t}\times\boldsymbol{\pi}=\mathbf{P}_\mathbf{t}\wedge\boldsymbol{\Pi}$, and this equality is important to maintain orientation through dualizations. We again, using the identity $(\mathbf{a}\cdot\mathbf{b})\mathbf{I}_4=\mathbf{a}\wedge\mathbf{b}^*\wedge\mathbf{e}_0$, dualize the scalar null space test condition as $(\mathbf{t}\cdot\hat{\mathbf{n}}-\mathbf{p}\cdot\hat{\mathbf{n}})\mathbf{I}_4$. This is exactly how we started when deriving the OPNS PGA plane $\boldsymbol{\Pi}$. But now, since $\mathbf{a}\cdot\mathbf{b}=\mathbf{b}\cdot\mathbf{a}$, we use the identity as $(\mathbf{a}\cdot\mathbf{b})\mathbf{I}_4=\mathbf{b}\wedge\mathbf{a}^*\wedge\mathbf{e}_0$, and obtain $(\mathbf{t}\cdot\hat{\mathbf{n}}-\mathbf{p}\cdot\hat{\mathbf{n}})\mathbf{I}_4=-\mathbf{t}^*\wedge\mathbf{e}_0\wedge\hat{\mathbf{n}}+\mathbf{I}_3(\mathbf{p}\cdot\hat{\mathbf{n}})\mathbf{e}_0=\boldsymbol{p}_\mathbf{t}\wedge\boldsymbol{\pi}$. Let $\boldsymbol{\pi}=\hat{\mathbf{n}}+(\mathbf{p}\cdot\hat{\mathbf{n}})\mathbf{e}_0$, which contains everything that defines the plane as a vector. Let $\boldsymbol{p}_\mathbf{t}=-\mathbf{t}^*\wedge\mathbf{e}_0+\mathbf{I}_3=(1+\mathbf{t}^*\mathbf{I}_4)\mathbf{I}_3$, which we further discuss in Section 3.2.3. Finally, we can summarize as follows.

The CPNS PGA 1-bade plane entity $\boldsymbol{\pi}=\boldsymbol{\pi}_{\mathbf{p},\hat{\mathbf{n}}}=\boldsymbol{\pi}_{d,\hat{\mathbf{n}}}$ for the plane with unit normal vector $\hat{\mathbf{n}}$ through point $\mathbf{p}$, or at distance $d=\mathbf{p}\cdot\hat{\mathbf{n}}$ from the origin, is defined as

$$\boldsymbol{\pi}=\boldsymbol{\pi}_{\mathbf{p},\hat{\mathbf{n}}}=\boldsymbol{\pi}_{d,\hat{\mathbf{n}}}=\hat{\mathbf{n}}+(\mathbf{p}\cdot\hat{\mathbf{n}})\mathbf{e}_0=\hat{\mathbf{n}}+d\mathbf{e}_0. \tag{18}$$

To test if a vector point $\mathbf{t}$, represented by the CPNS PGA 3-blade point entity $\boldsymbol{p}_\mathbf{t}$, is on the 1-blade $\boldsymbol{\pi}$, we produce the geometric null space entity as $\boldsymbol{p}_\mathbf{t} \times \boldsymbol{\pi} = \boldsymbol{p}_\mathbf{t} \wedge \boldsymbol{\pi}$ that produces the same pseudoscalar as the corresponding test $\mathbf{P}_\mathbf{t} \wedge \boldsymbol{\Pi}$ in the dual OPNS PGA. The point $\boldsymbol{p}_\mathbf{t}$ is on the plane $\boldsymbol{\pi}$ if and only if $\boldsymbol{p}_\mathbf{t} \wedge \boldsymbol{\pi} = 0$.

If we look at the geometric product $\boldsymbol{p}_\mathbf{t}\boldsymbol{\pi} = \boldsymbol{p}_\mathbf{t} \cdot \boldsymbol{\pi} + \boldsymbol{p}_\mathbf{t} \wedge \boldsymbol{\pi}$, we notice that the possible graded parts of this product are of grades 4, 2, or 0. The product symmetry test for the inner product of an r-blade $\mathbf{A}_r$ and s-blade $\mathbf{B}_s$, $r \leq s$, is that $(-1)^{r(s-1)} = 1$ indicates product symmetry $\mathbf{A}_r \cdot \mathbf{B}_s = \mathbf{B}_s \cdot \mathbf{A}_r$ and this inner product is part of the symmetric anti-commutator product $\mathbf{A}_r \mathbin{\overline{\times}} \mathbf{B}_s = \frac{1}{2}(\mathbf{A}_r\mathbf{B}_s + \mathbf{B}_s\mathbf{A}_r)$, otherwise $(-1)^{r(s-1)} = -1$ indicates product antisymmetry and is part of the antisymmetric commutator product $\mathbf{A}_r \times \mathbf{B}_s = \frac{1}{2}(\mathbf{A}_r\mathbf{B}_s - \mathbf{B}_s\mathbf{A}_r)$. By the product symmetry test for inner product of blades, we see that $\boldsymbol{p}_\mathbf{t} \cdot \boldsymbol{\pi}$ is symmetric and part of the symmetric anti-commutator product. The product symmetry test for the outer product of an r-blade $\mathbf{A}_r$ and s-blade $\mathbf{B}_s$ is that $(-1)^{rs} = 1$ indicates $\mathbf{A}_r \wedge \mathbf{B}_s = \mathbf{B}_s \wedge \mathbf{A}_r$ and this outer product is part of the symmetric anti-commutator product $\mathbf{A}_r \mathbin{\overline{\times}} \mathbf{B}_s = \frac{1}{2}(\mathbf{A}_r\mathbf{B}_s + \mathbf{B}_s\mathbf{A}_r)$, otherwise it is part of the antisymmetric commutator product $\mathbf{A}_r \times \mathbf{B}_s = \frac{1}{2}(\mathbf{A}_r\mathbf{B}_s - \mathbf{B}_s\mathbf{A}_r)$. By the product symmetry test for outer product of blades, we see that $\boldsymbol{p}_\mathbf{t} \wedge \boldsymbol{\pi}$ is antisymmetric and is part of the commutator product. Therefore, if we use the commutator product, then we get $\boldsymbol{p}_\mathbf{t} \times \boldsymbol{\pi} = \boldsymbol{p}_\mathbf{t} \wedge \boldsymbol{\pi}$, the pseudoscalar part that we are wanting. By this result, we call the entities commutator product null space (CPNS) entities. We will continue to find that the commutator product gives the part of the geometric product we want as the geometric null space entity.

In IPNS CGA, the plane entity is $\hat{\mathbf{n}} + d\mathbf{e}_\infty$, which is very similar to the CPNS PGA plane $\boldsymbol{\pi} = \hat{\mathbf{n}} + d\mathbf{e}_0$. The correspondence between IPNS CGA and CPNS PGA is not exact since no geometrical counterpart to $\mathbf{e}_\infty$ exists in PGA. In OPNS PGA, $\mathbf{e}_0$ and $-\mathbf{I}_3$ fulfill the roles of the CGA $\mathbf{e}_o$ and $\mathbf{e}_\infty$, respectively. In the CPNS PGA, the duals $\mathbf{I}_3 = J_e(\mathbf{e}_0)$ and $\mathbf{e}_0 = J_e(-\mathbf{I}_3)$ fulfill the roles of the CGA $\mathbf{e}_o$ and $\mathbf{e}_\infty$, respectively. Recall that, in CGA, $\mathbf{e}_o \cdot \mathbf{e}_\infty = -1$, similar to in CPNS PGA, $\mathbf{I}_3\mathbf{e}_0 = J_e(\mathbf{e}_0)J_e(-\mathbf{I}_3) = -\mathbf{I}_4$ since we use pseudoscalars instead of scalars to work in the degenerate metric of the inner product. We have in CGA, $\mathbf{e}_o \wedge (\mathbf{e}_\infty\mathbf{I}_5) = -\mathbf{I}_5$, similar to OPNS PGA, $\mathbf{e}_0(-J_e(\mathbf{e}_0)) = -\mathbf{e}_0\mathbf{I}_3 = -\mathbf{I}_4$, where we undual $-J_e(\mathbf{e}_0) = -\mathbf{I}_3$ the $\mathbf{e}_\infty$-like counterpart $\mathbf{e}_0$ back to OPNS PGA as its dual element $-\mathbf{I}_3$ fulfilling the role of $\mathbf{e}_\infty$ again. We also have in CGA, $(\mathbf{e}_0\mathbf{I}_5) \wedge \mathbf{e}_\infty = -\mathbf{I}_5$, similar to CPNS PGA, $\mathbf{I}_3(J_e(-\mathbf{I}_3)) = \mathbf{I}_3\mathbf{e}_0 = -\mathbf{I}_4$, where we dualize $J_e(-\mathbf{I}_3) = \mathbf{e}_0$ to CPNS PGA as its dual element $\mathbf{e}_0$ fulfilling the role of $\mathbf{e}_\infty$ again. So, $\mathbf{e}_0$ and $-\mathbf{I}_3$ and their duals and unduals keep swapping roles as either $\mathbf{e}_o$ or $\mathbf{e}_\infty$ through the dualization J_e from OPNS PGA to CPNS PGA, or undual $-J_e$ the other way. As we will see, the dual operator J_e, as we define it in this paper, handles all of this correctly.

3.2.2 CPNS PGA 2-blade Line Entity

The CPNS PGA is the plane-based GA. By this, we mean that, we can form the CPNS PGA 2-blade line entity $\boldsymbol{l}$ as the wedge of two plane entities,

$$\boldsymbol{l} = \boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2, \tag{19}$$

representing their intersection line. To test a point $\boldsymbol{p}_\mathbf{t}$ against the line $\boldsymbol{l}$, we use the commutator product, $\boldsymbol{p}_\mathbf{t} \times \boldsymbol{l}$. The point $\boldsymbol{p}_\mathbf{t}$ is on the line if and only if $\boldsymbol{p}_\mathbf{t} \times \boldsymbol{l} = 0$.

To prove that this test $\boldsymbol{p_t} \times \boldsymbol{l} = 0$ works, we need to use the identity $A \times (BC) = (A \times B)C + B(A \times C)$ [(1.57) p.14 in [19]]. Then we have, $\boldsymbol{p_t} \times \boldsymbol{l} = \mathbf{p_t} \times (\boldsymbol{\pi}_1 \boldsymbol{\pi}_2) = \boldsymbol{p_t} \times (\boldsymbol{\pi}_1 \cdot \boldsymbol{\pi}_2 + \boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2) = (\boldsymbol{p_t} \times \boldsymbol{\pi}_1)\boldsymbol{\pi}_2 + \boldsymbol{\pi}_1(\boldsymbol{p_t} \times \boldsymbol{\pi}_2)$, ignoring the scalar $\boldsymbol{\pi}_1 \cdot \boldsymbol{\pi}_2$ part since commutator product with scalars is 0. Now, $(\boldsymbol{p_t} \times \boldsymbol{\pi}_1)\boldsymbol{\pi}_2 + \boldsymbol{\pi}_1(\boldsymbol{p_t} \times \boldsymbol{\pi}_2)$ equals 0 if and only if both $\boldsymbol{p_t} \times \boldsymbol{\pi}_1$ and $\boldsymbol{p_t} \times \boldsymbol{\pi}_2$ are 0, and they are the pseudoscalar-valued point-plane tests. A product like $(\boldsymbol{p_t} \times \boldsymbol{\pi})\boldsymbol{\pi}_2 = d_{\mathbf{t}} \mathbf{I}_4 \boldsymbol{\pi}_2 = d_{\mathbf{t}} \mathbf{I}_4 \hat{\mathbf{n}}_2 = -d_{\mathbf{t}} \hat{\mathbf{n}}_2^* \mathbf{e}_0$, which is a 3-blade, cannot be 0 unless the scalar $d_{\mathbf{t}} = x n_x + y n_y + z n_z - d$ of the point-plane condition, the distance of $\boldsymbol{p_t}$ from the plane $\boldsymbol{\pi}$, is 0. In IPNS CGA, the inner product of a point and line $\mathbf{P} \cdot \mathbf{L} = \mathbf{P} \cdot (\boldsymbol{\Pi}_1 \wedge \boldsymbol{\Pi}_2) = (\mathbf{P} \cdot \boldsymbol{\Pi}_1)\boldsymbol{\Pi}_2 - \boldsymbol{\Pi}_1(\mathbf{P} \cdot \boldsymbol{\Pi}_2)$ can be compared to what the commutator product gives in CPNS PGA. We get two linearly independent terms when the planes are not parallel and we have a real line entity.

We can also derive the line $\boldsymbol{l}$ from geometric principles as we did for the OPNS PGA line entity $\mathbf{L}$. We derived the vector-valued null space line condition $(\mathbf{p}_1 - \mathbf{t}) \cdot \mathbf{d}^* = 0$, where $\mathbf{d} = \mathbf{p}_2 - \mathbf{p}_1$, and $\mathbf{p}_1$ and $\mathbf{p}_2$ are any two points on the line. The vector $\mathbf{t}$ is a third point being tested against the line, and $\mathbf{t}$ is on the line if and only if $(\mathbf{p}_1 - \mathbf{t}) \cdot \mathbf{d}^* = 0$. For the line entity $\boldsymbol{l}$, we require that $\boldsymbol{l} = J_e(\mathbf{L})$ such that $\boldsymbol{p_t} \times \boldsymbol{l} = J_e(\mathbf{P_t}) \times J_e(\mathbf{L}) = \mathbf{P_t} \wedge \mathbf{L}$ is the very same 3-blade geometric null space entity.

By dualization, we know that the line $\boldsymbol{l}$ is again a 2-blade, the test point $\boldsymbol{p_t} = \mathbf{I}_3 - \mathbf{e}_0 \mathbf{t}^*$ is a 3-blade, and that the test is the 3-blade geometric null space entity $\boldsymbol{p_t} \times \boldsymbol{l} = ((\mathbf{p}_1 - \mathbf{t}) \cdot \mathbf{d}^*)\mathbf{I}_4$, where we are solving for $\boldsymbol{l}$. We cannot straightforwardly use the identity $(\mathbf{a} \cdot \mathbf{b}^*)\mathbf{I}_4 = \mathbf{a} \wedge \mathbf{e}_0 \wedge \mathbf{b}$ as we did to solve for $\mathbf{L}$. So, we take a different approach to solving for $\boldsymbol{l}$. Take geometric products and grade 3 as $\boldsymbol{p_t} \times \boldsymbol{l} = \langle((\mathbf{p}_1 - \mathbf{t})\mathbf{d}^*)\mathbf{I}_4\rangle_3 = \langle(\mathbf{I}_3 - \mathbf{t}^*\mathbf{e}_0)\boldsymbol{l}\rangle_3$. Now, we expand geometric products and we have $((\mathbf{p}_1 - \mathbf{t})\mathbf{d}^*)\mathbf{I}_4 = \mathbf{p}_1\mathbf{d}^*\mathbf{I}_4 - \mathbf{t}\mathbf{d}^*\mathbf{I}_4 = \mathbf{I}_3\mathbf{e}_0\mathbf{p}_1\mathbf{d}^* - \mathbf{e}_0\mathbf{t}^*\mathbf{d}^*$. Let $\boldsymbol{l} = \langle\mathbf{e}_0\mathbf{p}_1\mathbf{d}^*\rangle_2 + \mathbf{d}^* = \mathbf{e}_0(\mathbf{p}_1 \cdot \mathbf{d}^*) + \mathbf{d}^*$. Take grade 3 with respect to $\boldsymbol{l}$ as $\langle\mathbf{I}_3\mathbf{e}_0\mathbf{p}_1\mathbf{d}^* - \mathbf{e}_0\mathbf{t}^*\mathbf{d}^*\rangle_3 = \mathbf{I}_3 \times (\mathbf{e}_0(\mathbf{p}_1 \cdot \mathbf{d}^*)) - (\mathbf{e}_0\mathbf{t}^*) \times \mathbf{d}^*$. Now check the product $\boldsymbol{p_t} \times \boldsymbol{l} = (\mathbf{I}_3 - \mathbf{e}_0\mathbf{t}^*) \times (\mathbf{d}^* + \mathbf{e}_0(\mathbf{p}_1 \cdot \mathbf{d}^*)) = \mathbf{I}_3 \times (\mathbf{e}_0(\mathbf{p}_1 \cdot \mathbf{d}^*)) - (\mathbf{e}_0\mathbf{t}^*) \times \mathbf{d}^*$. Therefore, the CPNS PGA 2-blade line entity $\boldsymbol{l}$ through point $\mathbf{p}$ in the direction $\hat{\mathbf{d}}$ is

$$\boldsymbol{l} = \hat{\mathbf{d}}^* - (\mathbf{p} \cdot \hat{\mathbf{d}}^*)\mathbf{e}_0. \tag{20}$$

We choose to normalize the line to unit magnitude using unit direction $\hat{\mathbf{d}}$. This is the same exact form and orientation as the IPNS CGA line entity $\hat{\mathbf{d}} - (\mathbf{p} \cdot \hat{\mathbf{d}}^*)\mathbf{e}_\infty$, except that $\mathbf{e}_0$ is replaced in CGA with $\mathbf{e}_\infty$. As in CGA, $\boldsymbol{l}$ is able to act as an axis of rotation for a rotor $\exp(\theta\hat{\boldsymbol{l}}/2)$. We have that the CPNS PGA $\mathbf{e}_0$ fulfills the role of the CGA $\mathbf{e}_\infty$, while the OPNS PGA $\mathbf{e}_0$ filfills the role of the CGA $\mathbf{e}_o$. We have that the CPNS PGA $\mathbf{I}_3$ fulfills the role of the CGA $\mathbf{e}_o$, while the OPNS PGA $-\mathbf{I}_3$ fulfills the role of the CGA $\mathbf{e}_\infty$. The dual operator J_e handles the sign changes correctly so that orientation is preserved through the dualizations. The dual J_e is discussed in Section 4.

3.2.3 CPNS PGA 3-blade Point Entity

As part of our construction of the CPNS PGA 1-blade plane entity $\boldsymbol{\pi}$, we also derived the required form of the CPNS PGA 3-blade point entity $\boldsymbol{p_t}$ as

$$\boldsymbol{p_t} = (1 + \mathbf{e}_0\mathbf{t})\mathbf{I}_3 = \mathbf{I}_3 - \mathbf{e}_0\mathbf{t}^*. \tag{21}$$

Note that, $1 + \mathbf{e}_0\mathbf{t} = 1 + \mathbf{t}^*\mathbf{I}_4 = p_{\mathbf{t}}$ represents a homogeneous point $p_{\mathbf{t}}$ in dual quaternions, which is discussed further in Section 5. Other forms are $\boldsymbol{p_t} = (1 + \mathbf{t}^*\mathbf{I}_4)\mathbf{I}_3 = \mathbf{I}_3 + \mathbf{I}_4\mathbf{t}$.

Notice that, the product of two points is $\boldsymbol{p}_\mathbf{a}\boldsymbol{p}_\mathbf{b} = -p_\mathbf{a}\bar{p}_\mathbf{b} = -(1+\mathbf{e}_0(\mathbf{a}-\mathbf{b}))$, representing their difference as a point in dual quaternions. The bivector part $-\mathbf{e}_0(\mathbf{a}-\mathbf{b})$ is given by $\boldsymbol{p}_\mathbf{a} \times \boldsymbol{p}_\mathbf{b}$ as a test for point equality, where the product is 0 if and only if the two points are representing the same point. Any scaling will not affect the point equality test.

Since a point is the intersection of three non-parallel planes, we should be able to form a point using three planes. This is done very simply as the wedge of three non-parallel plane entities,

$$\boldsymbol{p} = \boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_3 = \boldsymbol{\pi} \wedge \boldsymbol{l}, \tag{22}$$

which can also be seen as the wedge of a plane and a line. We test a point $\boldsymbol{p}_\mathbf{t}$ to check if it is the intersection point (although this may seem unnecessary) by the commutator product as $\boldsymbol{p}_\mathbf{t} \times \boldsymbol{p}$. The point $\boldsymbol{p}_\mathbf{t}$ is the point $\boldsymbol{p}$ (although possibly of different scale) if and only if $\boldsymbol{p}_\mathbf{t} \times \boldsymbol{p} = 0$. It can be shown through repeated use of the identity $A \times (BC) = (A \times B)C + B(A \times C)$ that a point $\boldsymbol{p}_\mathbf{t}$ satisfies $\boldsymbol{p}_\mathbf{t} \times (\boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_3) = 0$ if and only if $\boldsymbol{p}_\mathbf{t} \times \boldsymbol{\pi}_i = 0,\ \forall i \in \{1,2,3\}$, meaning that $\boldsymbol{p}_\mathbf{t}$ is coincident on all three planes at their intersection point.

When $\mathbf{t} = 0$, then $\boldsymbol{p}_0 = \mathbf{I}_3$, so that $\mathbf{I}_3$ represents the point at the origin. In the limit, $\lim_{\|\mathbf{t}\|\to\infty} (\boldsymbol{p}_\mathbf{t} / \|\mathbf{t}\|) = \mathbf{e}_0\hat{\mathbf{t}}\mathbf{I}_3 = \mathbf{I}_4\hat{\mathbf{t}}$, we see that $\boldsymbol{p}_{\infty\hat{\mathbf{t}}} = \mathbf{I}_4\hat{\mathbf{t}}$ represents a directed point at infinity. The dual is $D_e(\mathbf{I}_4\hat{\mathbf{t}}) = -J_e(\mathbf{I}_4\hat{\mathbf{t}}) = \hat{\mathbf{t}}$, which also represents a directed point at infinity in OPNS PGA. Any scalar multiple $\|\mathbf{t}\|$ still represents the same entity, so $\mathbf{I}_4\mathbf{t}$ is still a directed point at infinity. This may lend some insight into the form $\boldsymbol{p}_\mathbf{t} = \mathbf{I}_3 + \mathbf{I}_4\mathbf{t} = \boldsymbol{p}_0 + \boldsymbol{p}_{\infty\mathbf{t}}$, that it represents the sum of the point at the origin and a directed point at infinity, which dualizes to $D_e(\boldsymbol{p}_\mathbf{t}) = \mathbf{e}_0 + \mathbf{t} = \mathbf{P}_\mathbf{t}$, representing the same point in OPNS PGA.

3.3 CPNS PGA Operations

The CPNS PGA operations include rotation and translation. The CPNS PGA plane-based entities support the meet (of planes) operation as the wedge product of 2 or 3 planes to form a line or plane. The join (of points) operation is also possible via dualization (using the geometric entity dualization operation J_e) to point-based entities, where the join operation is the wedge product of 2 or 3 dual points to form a line or plane.

3.3.1 CPNS PGA 2-versor Translation Operator

We noted that the point entity $\boldsymbol{p}_\mathbf{t}$ has the form $(1+\mathbf{e}_0\mathbf{t})\mathbf{I}_3$, where $p_\mathbf{t} = (1+\mathbf{e}_0\mathbf{t}) = \exp(\mathbf{e}_0\mathbf{t})$ has the form of a dual quaternion homogeneous point. We will elaborate on dual quaternions in Section 5, but for now just note further that the product of two of these dual quaternion points acts as translation, $\exp(\mathbf{e}_0\mathbf{t})\exp(\mathbf{e}_0\mathbf{p}) = \exp(\mathbf{e}_0(\mathbf{t}+\mathbf{p}))$. This is because their multiplication is commutative, similar to complex or dual numbers where magnitudes multiply and "angles" add. Since $\mathbf{e}_0^2 = 0$, the commutativity of the multiplication holds true no matter what "angles" are on $\mathbf{e}_0$. Therefore, a translation operator has the form $T = \exp(\mathbf{e}_0\mathbf{d})$ for translation by a vector displacement $\mathbf{d}$ in dual quaternions. We can apply this translation operator *as a versor* on a CPNS PGA point as $T\boldsymbol{p}T^{-1} = T\exp(\mathbf{e}_0\mathbf{p})T\mathbf{I}_3 = (1+\mathbf{e}_0(\mathbf{p}+2\mathbf{d}))\mathbf{I}_3$. To translate by just $\mathbf{d}$, then we use the form $T = \exp(\mathbf{e}_0\mathbf{d}/2)$. Since T acts correctly by versor outermorphism on a point entity $\boldsymbol{p}$, then we look at the form $\boldsymbol{p} = \boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_3$, and $\boldsymbol{p}' = T(\boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_3)T^{-1}$. We must conclude that $\boldsymbol{p}' = \langle T\boldsymbol{\pi}_1T^{-1}T\boldsymbol{\pi}_2T^{-1}T\boldsymbol{\pi}_3T^{-1}\rangle_3$ is also correct and that each plane must be correctly translated as $\boldsymbol{\pi}' = T\boldsymbol{\pi}T^{-1}$. Then, by versor outermorphism again, we must also have the correct results from $\boldsymbol{l}' = T\boldsymbol{l}T^{-1} = T(\boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2)T^{-1} = \langle T\boldsymbol{\pi}_1T^{-1}T\boldsymbol{\pi}_2T^{-1}\rangle_2$.

A plane through the origin is represented by $\boldsymbol{\pi}_1 = \hat{\mathbf{n}}_1 + 0\mathbf{e}_0 = \hat{\mathbf{n}}_1$. We reflect a general plane $\boldsymbol{\pi}_2 = \hat{\mathbf{n}}_2 + d_2\mathbf{e}_0$ in $\boldsymbol{\pi}_1$ as $\boldsymbol{\pi}_2' = -\boldsymbol{\pi}_1\boldsymbol{\pi}_2\boldsymbol{\pi}_1 = -\hat{\mathbf{n}}_1\hat{\mathbf{n}}_2\hat{\mathbf{n}}_1 + d_2\mathbf{e}_0$, which is correct. Using the translation operator, we can take a general plane and translate it to the origin, and translate other planes by the same translation so that they translate together as a rigid body. Then, we can reflect planes in the plane that is at the origin. After the reflection, we translate all the planes back to where they were by the reverse translation operation. This allows reflection in general planes. For example, using $\boldsymbol{\pi}_1 = \hat{\mathbf{n}}_1 + d_1\mathbf{e}_0$ and $\boldsymbol{\pi}_2 = \hat{\mathbf{n}}_2 + d_2\mathbf{e}_0$ and $T = \exp(d_1\mathbf{e}_0\hat{\mathbf{n}}_1/2)$, we reflect $\boldsymbol{\pi}_2$ in $\boldsymbol{\pi}_1$ as $\boldsymbol{\pi}_2' = -TT^{-1}\boldsymbol{\pi}_1\boldsymbol{\pi}_2\boldsymbol{\pi}_1TT^{-1} = -\boldsymbol{\pi}_1\boldsymbol{\pi}_2\boldsymbol{\pi}_1$. The result is that we can simply reflect general planes in general planes without using translation. Again, by versor outermorphism, we can also reflect a line $\boldsymbol{l}$ or point $\boldsymbol{p}$ in a plane $\boldsymbol{\pi}$. Now, we can successively reflect in two non-parallel or two parallel general planes and generate rotations around general lines or translations.

Note that, the CPNS PGA translator has depended on the form of the CPNS PGA 3-blade point entity $\boldsymbol{p}$, that $\boldsymbol{p}$ is essentially a dual quaternion homogeneous point $p = 1 + \mathbf{I}_4\mathbf{p}^*$ that is "dualized" as $\boldsymbol{p} = p\mathbf{I}_3$ to transform it into a grade 3 element of CPNS PGA. The OPNS PGA point $\mathbf{P}$ does not have a form that allows a translation versor. We will overcome this by dualizing OPNS to CPNS, and also possible transformation to dual quaternion representations. In summary, we have the following forms of the translator T:

First form: The CPNS PGA translation operator (or versor), called the translator T, for translation by displacement vector $\mathbf{d}$, is defined as

$$T = \exp(\mathbf{e}_0\mathbf{d}/2), \tag{23}$$

which can be interpreted as representing a dual quaternion homogeneous point representation or embedding $T = p_{\mathbf{d}/2}$.

Second form: The CPNS PGA translator T for translation by displacement vector $\mathbf{d}$, can be defined by successive reflections in two parallel planes, in $\boldsymbol{\pi}_1$ and then in $\boldsymbol{\pi}_2$, that are separated by $\mathbf{d}/2$ so that $\boldsymbol{\pi}_1 = \hat{\mathbf{d}} + d_1\mathbf{e}_0$ and $\boldsymbol{\pi}_2 = \hat{\mathbf{d}} + (d_1 + \|\mathbf{d}\|/2)\mathbf{e}_0$. The translator T is then

$$T = \boldsymbol{\pi}_2\boldsymbol{\pi}_1 = \boldsymbol{\pi}_2 \cdot \boldsymbol{\pi}_1 + \boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_1 = 1 + \mathbf{e}_0\mathbf{d}/2 = \exp(\mathbf{e}_0\mathbf{d}/2). \tag{24}$$

The contribution of d_1 cancels out and does not matter, so it could just as well be $d_1 = 0$.

3.3.2 CPNS PGA 2-versor Rotation Operator

We use the same rotor in CPNS PGA as in the OPNS PGA, $R = \exp(\theta\hat{\mathbf{a}}^*/2)$. In the plane-based PGA, to prove the correctness of R as a rotor for the CPNS PGA entities, all we need to check is that R correctly rotates the plane entity $\boldsymbol{\pi}$. The other entities, the 2-blade line $\boldsymbol{l}$ and 3-blade point $\boldsymbol{p}$, are wedge products of 2 or 3 of the plane entities. By outermorphism, each plane is rotated and therefore the entire entity. We check as follows: $R\boldsymbol{\pi}R^{-1} = R(\hat{\mathbf{n}} + d\mathbf{e}_0)R^{-1} = R\hat{\mathbf{n}}R^{-1} + RR^{-1}d\mathbf{e}_0$. The scalar and bivector parts of R^{-1} commute without sign changes with $d\mathbf{e}_0$, leaving it unchanged. We already know that vectors are correctly rotated by R as $\hat{\mathbf{n}}' = R\hat{\mathbf{n}}R^{-1}$. The rotor R rotates the plane $\boldsymbol{\pi}$ as a rigid body relative to the origin, around axis $\hat{\mathbf{a}}$ by angle θ, leaving the distance to the origin d unchanged, and the plane normal vector $\hat{\mathbf{n}}$ along the line of the distance d is rotated around axis $\hat{\mathbf{a}}$ by angle θ, all as expected. Again, by versor outermorphism, we can rest assured that rotation of a line $\boldsymbol{l}' = R\boldsymbol{l}R^{-1}$ or point $\boldsymbol{p}' = R\boldsymbol{p}R^{-1}$ also works as expected.

We also found, in the previous section on the translator T, that we can reflect in general planes. This means that we can successively reflect in two non-parallel planes to generate a rotation around the intersecting line of the two planes by twice the angle

$\alpha = \theta/2$ between the planes, which is a known result in geometry. Given two non-parallel unit planes $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$, the rotor R that reflects in these planes is

$$R = \boldsymbol{\pi}_2\boldsymbol{\pi}_1 = \boldsymbol{\pi}_2 \cdot \boldsymbol{\pi}_1 + \boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_1 = \cos(\alpha) + \sin(\alpha)\hat{\boldsymbol{l}} = \exp(\theta\hat{\boldsymbol{l}}/2). \tag{25}$$

The result is a rotor R for rotation by angle θ in the direction of $\boldsymbol{\pi}_1$ toward $\boldsymbol{\pi}_2$ around their intersection unit line $\hat{\boldsymbol{l}} = \boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_1 / \|\boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_1\|$. Note that, the unit planes, $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$, have the form $\boldsymbol{\pi} = \hat{\mathbf{n}} + d\mathbf{e}_0$, but it is not required that they actually be unit planes since any scale β is removed by the versor sandwiching product $A' = RAR^{-1}$, but we get $\beta^2 A' = RAR^{\sim}$ when using reverse. If we make direct use of a line $\boldsymbol{l}$, we must normalize it to a unit line $\hat{\boldsymbol{l}}$ and know the orientation of its direction $\hat{\mathbf{d}}$ as the axis of rotation, or else the angle could be wrong.

If we like, we can now rotate by angle θ around direction $\hat{\mathbf{n}}$ centered on a point $\mathbf{c}$ by using the translated rotor

$$R = TRT^{-1}, \tag{26}$$

where $R = \exp(\theta\hat{\mathbf{n}}^*/2)$ and $T = \exp(\mathbf{e}_0\mathbf{c}/2)$. This translates $\mathbf{c}$ to the origin, performs the usual rotation centered on the origin, then translates back to $\mathbf{c}$. This is another form of $R = \exp(\theta\hat{\boldsymbol{l}}/2)$, but we set the axis $\hat{\mathbf{n}}$ without any ambiguity about its orientation.

3.3.3 Using Entity Dualization for the Join Operation

Using the geometric entity dualization operation J_e that is discussed in Section 4, we can (un)dualize any CPNS PGA plane-based entity $\boldsymbol{a} = \mathbf{A}^\star = J_e(\mathbf{A})$ to its (un)dual OPNS PGA point-based entity $\mathbf{A} = -J_e(\mathbf{A}^\star) = D_e(\mathbf{A}^\star) = D_e(\boldsymbol{a}) = \boldsymbol{a}^{-\star}$.

The line $\boldsymbol{l}$ of two points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is the join (of points) operation

$$\boldsymbol{l} = (\boldsymbol{p}_2^{-\star} \wedge \boldsymbol{p}_1^{-*})^\star. \tag{27}$$

The plane of three points $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and $\boldsymbol{p}_3$ (assuming they are arranged counterclockwise from above the plane) is the join (of points) operation

$$\boldsymbol{\pi} = (\boldsymbol{p}_3^{-\star} \wedge \boldsymbol{p}_2^{-\star} \wedge \boldsymbol{p}_1^{-*})^\star. \tag{28}$$

3.4 CPNS PGA Conclusion

We covered in detail the plane-based CPNS PGA, including its 1-bade plane $\boldsymbol{\pi}$, 2-blade $\boldsymbol{l}$, and 3-blade point $\boldsymbol{p}_\mathbf{t}$ entities. We derived the entities starting from the basic geometry, forming a null space, and putting the null space into the form of a geometric entity.

Then, we talked about the plane-based translation operation T, which can be used on only the plane-based entities, not on the point-based entities. The translation operator T is also found to be a dual quaternion point $T = p_{\mathbf{d}/2}$, so it can also be used in dual quaternions. The rotation operation R was also discussed, including how it can be translated for rotations around lines, or for rotations centered on a point $\mathbf{c}$ other than the origin.

The plane-based entities have the 1-blade plane $\boldsymbol{\pi}$ as the primary entity, and the line $\boldsymbol{l} = \boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2$ and point $\boldsymbol{p} = \boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_3$ can be formed as the meet (intersection by wedge product) of two or three planes.

We cannot join points (span points by wedge product) in the plane-based algebra of PGA, but we can easily dualize plane-based entities to their point-based duals in the point-based algebra using the new entity dualization operation J_e. Therefore, we can use both plane-based and point-based algebras as we like through the geometric entity dualization operation J_e, which we develop and discuss in detail in the next section.

4 Geometric Entity Dualization in PGA

This section introduces a new geometric entity dualization operation J_e for PGA $\mathcal{G}_{3,0,1}$.

4.1 Introduction to Geometric Entity Dualization in PGA

In prior literature on PGA $\mathcal{G}_{3,0,1}$, such as in [17] and [21], the PGA dualization operation seems to have been difficult to clearly and correctly define and implement. The two papers [17] and [21] appear to have been the current understanding on PGA as of December 2023 at the time of writing this paper. In [17], the PGA dualization operation is denoted $J(\mathbf{e})$, and in [21] the PGA dualization is denoted $\star\mathbf{A}$, which is a Hodge star $\star$ notation. As will be explained, the PGA dualization operations as given in [17] and [21] do not appear to be correct. The dualization operations given in [23] for the similar algebra $\mathcal{G}_{0,3,1}$ also seem to be incorrect for the same reasons as will be explained. To remedy this situation, we study the PGA dualization and define and implement a new geometric entity dualization operation for PGA, which we have denoted J_e as distinct from $J(\mathbf{e})$ and $\star\mathbf{A}$. However, as will be shown, we have closely borrowed similar notation, including some usage of the Hodge star $\star$ notation.

In the following sections, we first derive the correct dualizations of each basis blade of $\mathcal{G}_{3,0,1}$, dualizing from the OPNS PGA geometric elements to the corresponding CPNS PGA geometric elements representing the same geometry. The requirement of the dualization operation J, as named in the prior published literature, is that an OPNS PGA geometric entity must dualize to its corresponding CPNS PGA geometric entity as a linear combination of basis blades, each dualized to their geometrically corresponding dual basis blade. At first, we do not assume this is Hodge star $\star$ dualization or any other known type of dualization. The correct dualization operation, denoted J_e in this paper, will be empirically determined from the OPNS PGA entities and their corresponding dual CPNS PGA entities by direct observation. We then tabulate the observed duals into an empirical dualization table for J_e of each basis blade to its dual basis blade, going from OPNS PGA to CPNS PGA. Finally, we formulate algebraic dualization procedures for J_e to match the empirical dualization table for J_e, rendering the table thereafter unnecessary. We call J_e, which is the dualization operation that we will determine empirically, the geometric *empirical dualization* operation. We will then implement J_e, calling it the geometric *entity dualization* operation for PGA $\mathcal{G}_{3,0,1}$.

4.2 Empirical Determination of Entity Dualization Operation

In the following sections, we will begin with the four basis 1-blades, then the six basis 2-blades, and then the four basis 3-blades. For each grade k of basis blades, we compare grade k entities in OPNS PGA to corresponding dual grade $4-k$ entities in CPNS PGA and directly observe the corresponding dual basis blades. How the 0-blade 1 and 4-blade $\mathbf{I}_4$ dualize is shown to be a consequence of how the other basis blades dualize and the resulting properties of the algebraic dualization operation for implementing J_e. We then discuss the Hodge star $\star$ dualization and its relation to geometric algebra and J_e before discussing the implementation of J_e.

4.2.1 Empirical Dualization of the Four Basis 1-blades

We compare the OPNS PGA 1-blade point $\mathbf{P_t} = \mathbf{e}_0 + \mathbf{t}$ with the CPNS PGA 3-blade point $\boldsymbol{p}_\mathbf{t} = \mathbf{I}_3 - \mathbf{e}_0\mathbf{t}^* = \mathbf{I}_3 + \mathbf{e}_0\mathbf{t}\mathbf{I}_3$. The points are considered to be in standard unit point form and orientation. We compare the three points $\mathbf{x} = \mathbf{e}_1$, $\mathbf{y} = \mathbf{e}_2$, and $\mathbf{z} = \mathbf{e}_3$. The basis 1-blades should dualize to basis 3-blades as follows:

For $\mathbf{x}$, we have $\mathbf{P_x} = \mathbf{e}_0 + \mathbf{e}_1$ and $\boldsymbol{p}_\mathbf{x} = \mathbf{I}_3 - \mathbf{e}_0\mathbf{e}_1^* = \mathbf{I}_3 - \mathbf{e}_0\mathbf{e}_3\mathbf{e}_2$. We observe the duals $J_e(\mathbf{e}_0) = \mathbf{I}_3$ and $J_e(\mathbf{e}_1) = -\mathbf{e}_0\mathbf{e}_3\mathbf{e}_2 = \mathbf{e}_0\mathbf{e}_2\mathbf{e}_3$.

For $\mathbf{y}$, we have $\mathbf{P_y} = \mathbf{e}_0 + \mathbf{e}_2$ and $\boldsymbol{p}_\mathbf{y} = \mathbf{I}_3 - \mathbf{e}_0\mathbf{e}_2^* = \mathbf{I}_3 - \mathbf{e}_0\mathbf{e}_1\mathbf{e}_3$. We observe the duals $J_e(\mathbf{e}_0) = \mathbf{I}_3$ and $J_e(\mathbf{e}_2) = -\mathbf{e}_0\mathbf{e}_1\mathbf{e}_3$.

For $\mathbf{z}$, we have $\mathbf{P_z} = \mathbf{e}_0 + \mathbf{e}_3$ and $\boldsymbol{p}_\mathbf{z} = \mathbf{I}_3 - \mathbf{e}_0\mathbf{e}_3^* = \mathbf{I}_3 - \mathbf{e}_0\mathbf{e}_2\mathbf{e}_1$. We observe the duals $J_e(\mathbf{e}_0) = \mathbf{I}_3$ and $J_e(\mathbf{e}_3) = -\mathbf{e}_0\mathbf{e}_2\mathbf{e}_1 = \mathbf{e}_0\mathbf{e}_1\mathbf{e}_2$.

4.2.2 Empirical Dualization of the Six Basis 2-blades

For observing the correct basis 2-blade duals, we will look at entities for three lines along different directions, and for each line we compare the OPNS PGA 2-blade line entity $\mathbf{L}_{\mathbf{p},\hat{\mathbf{d}}} = \hat{\mathbf{d}} \wedge \mathbf{P_p}$ with its corresponding dual CPNS PGA 2-blade line entity $\boldsymbol{l}_{\mathbf{p},\hat{\mathbf{d}}} = \hat{\mathbf{d}}^* - (\mathbf{p} \cdot \hat{\mathbf{d}}^*)\mathbf{e}_0$. First, we always check that the two dual entities have the same geometric null space entity and orientation. Also, for the OPNS PGA line entity, we can ignore the third pseudoscalar term of the null space entity that represents $\mathbf{t} \wedge \mathbf{p} \wedge \hat{\mathbf{d}}$, which is 0 for any point $\mathbf{t}$ on the line through $\mathbf{p}$ in direction $\hat{\mathbf{d}}$.

For $\mathbf{p} = \mathbf{e}_1$, $\hat{\mathbf{d}} = \mathbf{e}_2$, we have the 3-blade null space entities with same orientation:

$$\begin{aligned}
\mathbf{P_t} \wedge \mathbf{L}_{\mathbf{p},\hat{\mathbf{d}}} &= (x-1)\mathbf{e}_0\mathbf{e}_1\mathbf{e}_2 - z\mathbf{e}_0\mathbf{e}_2\mathbf{e}_3 - z\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3 && (29)\\
\boldsymbol{p}_\mathbf{t} \times \boldsymbol{l}_{\mathbf{p},\hat{\mathbf{d}}} &= (x-1)\mathbf{e}_0\mathbf{e}_1\mathbf{e}_2 - z\mathbf{e}_0\mathbf{e}_2\mathbf{e}_3 && (30)\\
\mathbf{L}_{\mathbf{p},\hat{\mathbf{d}}} &= -\mathbf{e}_0\mathbf{e}_2 - \mathbf{e}_1\mathbf{e}_2 && (31)\\
\boldsymbol{l}_{\mathbf{p},\hat{\mathbf{d}}} &= \mathbf{e}_0\mathbf{e}_3 + \mathbf{e}_1\mathbf{e}_3. && (32)
\end{aligned}$$

We observe the basis 2-blade duals: $J_e(\mathbf{e}_0\mathbf{e}_2) = -\mathbf{e}_1\mathbf{e}_3$ and $J_e(\mathbf{e}_1\mathbf{e}_2) = -\mathbf{e}_0\mathbf{e}_3$.

For $\mathbf{p} = \mathbf{e}_2$, $\hat{\mathbf{d}} = \mathbf{e}_3$, we have the 3-blade null space entities with same orientation:

$$\begin{aligned}
\mathbf{P_t} \wedge \mathbf{L}_{\mathbf{p},\hat{\mathbf{d}}} &= x\mathbf{e}_0\mathbf{e}_1\mathbf{e}_3 + (y-1)\mathbf{e}_0\mathbf{e}_2\mathbf{e}_3 - x\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3 && (33)\\
\boldsymbol{p}_\mathbf{t} \times \boldsymbol{l}_{\mathbf{p},\hat{\mathbf{d}}} &= x\mathbf{e}_0\mathbf{e}_1\mathbf{e}_3 + (y-1)\mathbf{e}_0\mathbf{e}_2\mathbf{e}_3 && (34)\\
\mathbf{L}_{\mathbf{p},\hat{\mathbf{d}}} &= -\mathbf{e}_0\mathbf{e}_3 - \mathbf{e}_2\mathbf{e}_3 && (35)\\
\boldsymbol{l}_{\mathbf{p},\hat{\mathbf{d}}} &= \mathbf{e}_0\mathbf{e}_1 - \mathbf{e}_1\mathbf{e}_2. && (36)
\end{aligned}$$

We observe the basis 2-blade duals: $J_e(\mathbf{e}_0\mathbf{e}_3) = \mathbf{e}_1\mathbf{e}_2$ and $J_e(\mathbf{e}_2\mathbf{e}_3) = -\mathbf{e}_0\mathbf{e}_1$.

For $\mathbf{p} = \mathbf{e}_3$, $\hat{\mathbf{d}} = \mathbf{e}_1$, we have the 3-blade null space entities with same orientation:

$$\begin{aligned}
\mathbf{P_t} \wedge \mathbf{L}_{\mathbf{p},\hat{\mathbf{d}}} &= -y\mathbf{e}_0\mathbf{e}_1\mathbf{e}_2 + (1-z)\mathbf{e}_0\mathbf{e}_1\mathbf{e}_3 - y\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3 && (37)\\
\boldsymbol{p}_\mathbf{t} \times \boldsymbol{l}_{\mathbf{p},\hat{\mathbf{d}}} &= -y\mathbf{e}_0\mathbf{e}_1\mathbf{e}_2 + (1-z)\mathbf{e}_0\mathbf{e}_1\mathbf{e}_3 && (38)\\
\mathbf{L}_{\mathbf{p},\hat{\mathbf{d}}} &= -\mathbf{e}_0\mathbf{e}_1 + \mathbf{e}_1\mathbf{e}_3 && (39)\\
\boldsymbol{l}_{\mathbf{p},\hat{\mathbf{d}}} &= \mathbf{e}_0\mathbf{e}_2 - \mathbf{e}_2\mathbf{e}_3. && (40)
\end{aligned}$$

We observe the basis 2-blade duals: $J_e(\mathbf{e}_0\mathbf{e}_1) = \mathbf{e}_2\mathbf{e}_3$ and $J_e(\mathbf{e}_1\mathbf{e}_3) = \mathbf{e}_0\mathbf{e}_2$.

We can further observe that these six basis 2-blade empirical duals appear to support an anti-involution, $J_e(J_e(\mathbf{A})) = -\mathbf{A}$. We assume that J_e is a linear operator so that $J_e(a\mathbf{A} + b\mathbf{B}) = a\,J_e(\mathbf{A}) + b\,J_e(\mathbf{B})$.

4.2.3 Empirical Dualization of the Four Basis 3-blades

We compare the OPNS PGA 3-blade plane $\mathbf{\Pi}_{\mathbf{p},\hat{\mathbf{n}}} = \mathbf{P}_{\mathbf{p}} \wedge \hat{\mathbf{n}}^*$ with its dual CPNS PGA 1-blade plane $\boldsymbol{\pi}_{\mathbf{p},\hat{\mathbf{n}}} = \hat{\mathbf{n}} + (\mathbf{p} \cdot \hat{\mathbf{n}})\mathbf{e}_0$. We compare the three planes, $x = 1$, $y = 1$, and $z = 1$. First, we compare the 4-blade geometric null space entities to make sure they have the same scale and orientation, then we observe the duals.

For $x = 1$ ($\mathbf{p} = \mathbf{e}_1$, $\hat{\mathbf{n}} = \mathbf{e}_1$), we have the 4-blade null space entities with same orientation:

$$\begin{aligned} \mathbf{P}_{\mathbf{t}} \wedge \mathbf{\Pi}_{\mathbf{p},\hat{\mathbf{n}}} &= (x-1)\mathbf{I}_4 && (41)\\ \boldsymbol{p}_{\mathbf{t}} \times \boldsymbol{\pi}_{\mathbf{p},\hat{\mathbf{n}}} &= (x-1)\mathbf{I}_4 && (42)\\ \mathbf{\Pi}_{\mathbf{p},\hat{\mathbf{n}}} &= -\mathbf{e}_0\mathbf{e}_2\mathbf{e}_3 - \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3 && (43)\\ \boldsymbol{\pi}_{\mathbf{p},\hat{\mathbf{n}}} &= \mathbf{e}_0 + \mathbf{e}_1. && (44) \end{aligned}$$

We observe the duals of the basis 3-blades: $J_e(\mathbf{e}_0\mathbf{e}_2\mathbf{e}_3) = -\mathbf{e}_1$ and $J_e(\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3) = -\mathbf{e}_0$.

For $y = 1$ ($\mathbf{p} = \mathbf{e}_2$, $\hat{\mathbf{n}} = \mathbf{e}_2$), we have the 4-blade null space entities with same orientation:

$$\begin{aligned} \mathbf{P}_{\mathbf{t}} \wedge \mathbf{\Pi}_{\mathbf{p},\hat{\mathbf{n}}} &= (y-1)\mathbf{I}_4 && (45)\\ \boldsymbol{p}_{\mathbf{t}} \times \boldsymbol{\pi}_{\mathbf{p},\hat{\mathbf{n}}} &= (y-1)\mathbf{I}_4 && (46)\\ \mathbf{\Pi}_{\mathbf{p},\hat{\mathbf{n}}} &= \mathbf{e}_0\mathbf{e}_1\mathbf{e}_3 - \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3 && (47)\\ \boldsymbol{\pi}_{\mathbf{p},\hat{\mathbf{n}}} &= \mathbf{e}_0 + \mathbf{e}_2. && (48) \end{aligned}$$

We observe the duals of the basis 3-blades: $J_e(\mathbf{e}_0\mathbf{e}_1\mathbf{e}_3) = \mathbf{e}_2$ and $J_e(\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3) = -\mathbf{e}_0$.

For $z = 1$ ($\mathbf{p} = \mathbf{e}_3$, $\hat{\mathbf{n}} = \mathbf{e}_3$), we have the 4-blade null space entities with same orientation:

$$\begin{aligned} \mathbf{P}_{\mathbf{t}} \wedge \mathbf{\Pi}_{\mathbf{p},\hat{\mathbf{n}}} &= (z-1)\mathbf{I}_4 && (49)\\ \boldsymbol{p}_{\mathbf{t}} \times \boldsymbol{\pi}_{\mathbf{p},\hat{\mathbf{n}}} &= (z-1)\mathbf{I}_4 && (50)\\ \mathbf{\Pi}_{\mathbf{p},\hat{\mathbf{n}}} &= -\mathbf{e}_0\mathbf{e}_1\mathbf{e}_2 - \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3 && (51)\\ \boldsymbol{\pi}_{\mathbf{p},\hat{\mathbf{n}}} &= \mathbf{e}_0 + \mathbf{e}_3. && (52) \end{aligned}$$

We observe the duals of the basis 3-blades: $J_e(\mathbf{e}_0\mathbf{e}_1\mathbf{e}_2) = -\mathbf{e}_3$ and $J_e(\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3) = -\mathbf{e}_0$.

We further observe that, between the duals of 3-blades and 1-blades, we also have an anti-involution for J_e.

4.2.4 The Empirical Dualization Operation

For the empirical dualization operation J_e, we now accept by observation that J_e is an anti-involution from OPNS PGA entities to CPNS PGA entities. Therefore, the inverse (or "undual") is $J_e^{-1} = -J_e$ from CPNS PGA entities back to OPNS PGA entities. The anti-involution dualization operation J_e is characteristic of dualization by multiplication (or division) by a non-degenerate unit pseudoscalar $\mathbf{I}_4$, where $\mathbf{I}_4^2 = -1$. If it were an involution, then it would be characteristic of multiplication by a non-degenerate pseudoscalar $\mathbf{I}_4$, where $\mathbf{I}_4^2 = 1$.

If we define dualization as multiplication by $\mathbf{I}_4$, then 1 dualizes to $\mathbf{I}_4$, and $\mathbf{I}_4$ dualizes to -1. If we define dualization as division by $\mathbf{I}_4$, then 1 dualizes to $-\mathbf{I}_4$ and $\mathbf{I}_4$ dualizes to 1. The choice between multiplication or division by the unit pseudoscalar is not arbitrary, and we will find that multiplication by the unit pseudoscalar is the correct choice, or orientation, for J_e. Therefore, as we will see, the dualization of a geometric entity $\mathbf{A}$ (point, line, or plane) will have the form $\mathbf{A}^* = \mathbf{A}\mathbf{I}_4$ in $\mathcal{G}_{1,3}$, and $\mathbf{A}^* = \mathbf{I}_4\mathbf{A}$ in $\mathcal{G}_{3,1}$, and $\mathbf{A}^* = \mathbf{I}_3\mathbf{I}_4\mathbf{A}\mathbf{I}_3 = \mathbf{e}_0\mathbf{A}\mathbf{I}_3$ in $\mathcal{G}_4$. In $\mathcal{G}_3$, with unit pseudoscalar $\mathbf{I}_3$, we will continue to use division by $\mathbf{I}_3$ for dualizing vectors and bivectors, as is the common practice.

$\mathbf{A}$	1	$\mathbf{e}_0$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_0\mathbf{e}_1$	$\mathbf{e}_0\mathbf{e}_2$	$\mathbf{e}_0\mathbf{e}_3$	$\mathbf{e}_1\mathbf{e}_2$	$\mathbf{e}_1\mathbf{e}_3$	$\mathbf{e}_2\mathbf{e}_3$	$\mathbf{e}_0\mathbf{e}_1\mathbf{e}_2$	$\mathbf{e}_0\mathbf{e}_1\mathbf{e}_3$	$\mathbf{e}_0\mathbf{e}_2\mathbf{e}_3$	$\mathbf{I}_3$	$\mathbf{I}_4$
$J_e(\mathbf{A})$	$\mathbf{I}_4$	$\mathbf{I}_3$	$\mathbf{e}_0\mathbf{e}_2\mathbf{e}_3$	$-\mathbf{e}_0\mathbf{e}_1\mathbf{e}_3$	$\mathbf{e}_0\mathbf{e}_1\mathbf{e}_2$	$\mathbf{e}_2\mathbf{e}_3$	$-\mathbf{e}_1\mathbf{e}_3$	$\mathbf{e}_1\mathbf{e}_2$	$-\mathbf{e}_0\mathbf{e}_3$	$\mathbf{e}_0\mathbf{e}_2$	$-\mathbf{e}_0\mathbf{e}_1$	$-\mathbf{e}_3$	$\mathbf{e}_2$	$-\mathbf{e}_1$	$-\mathbf{e}_0$	-1

Table 1. Entity Dualization of OPNS PGA basis blade $\mathbf{A}$ to its dual CPNS PGA basis blade $J_e(\mathbf{A})$.

Table 1 shows the empirical dual $J_e(\mathbf{A})$ for each basis blade $\mathbf{A}$ in $\mathcal{G}_{3,0,1}$, going from OPNS PGA to CPNS PGA, summarizing our observations of what the dual $J_e(\mathbf{A})$ should be for each basis blade $\mathbf{A}$ so that entities dualize to corresponding entities representing the same geometric null space entity with the same orientation. To dualize from CPNS PGA to OPNS PGA, the inverse dual $J_e^{-1} = -J_e$ should be used instead. If the orientation of $-J_e$ is preferred, an alias could be used such as $D_e = -J_e$ to stand for the entity dualization from CPNS PGA to OPNS PGA, again with undual $-D_e = D_e^{-1}$.

The Hodge star $\star$ dualization is discussed in prior literature, so we also discuss it in the next section. Then, we discuss several ways to implement J_e as unit pseudoscalar dualization (or as a modified sandwiching product), which we show to be the same as Hodge star $\star$ dualization.

4.2.5 Hodge Star Dualization in Geometric Algebra

The Hodge star $\star$ operation on a k-vector (or basis k-blade) $\mathbf{A}$ produces its Hodge dual $(n-k)$-vector $\star\mathbf{A}$ in an $(n=p+q)$-dimensional pseudo-Euclidean geometric algebra $\mathcal{G}_{p,q,0}$ with unit pseudoscalar $\mathbf{I}_n$. The Hodge star $\star$ operation, defined only for pseudo-Euclidean metric $\mathcal{G}_{p,q,0}$, has the defining property:

$$\mathbf{A} \wedge (\star\mathbf{B}) = \star(\mathbf{A} \cdot \mathbf{B}), \tag{53}$$

for k-vectors $\mathbf{A}$ and $\mathbf{B}$ [11]. Using the identity

$$\mathbf{A} \cdot \mathbf{B} = (-1)^{r(s-1)}\mathbf{B} \cdot \mathbf{A} \tag{54}$$

for an r-vector $\mathbf{A}$ and s-vector $\mathbf{B}$, $r \leq s$, we see that $\mathbf{A} \cdot \mathbf{B} = \mathbf{B} \cdot \mathbf{A}$ for any two k-vectors $\mathbf{A}$ and $\mathbf{B}$. Since $\mathbf{A} \cdot \mathbf{B} = \mathbf{B} \cdot \mathbf{A}$, we can also write

$$\mathbf{A} \wedge (\star\mathbf{B}) = \mathbf{B} \wedge (\star\mathbf{A}) = \star(\mathbf{A} \cdot \mathbf{B}), \tag{55}$$

where we omit the orientation change $(-1)^{(n-t)/2}$ used in [11]:16.

In [11]:16, the orientation change $(-1)^{(n-t)/2}$ appears to be an error for two reasons. It seems that, the orientation change $(-1)^{(n-t)/2}$ in [11]:16 tries to enforce the orientation $\star\mathbf{I}_n = 1$, but $\star\mathbf{I}_n = -1$ is also a possible and correct orientation for some algebras. The entity dualization operation J_e is an anti-involution and we have $J_e(\mathbf{I}_4) = -1$ for our algebra. It seems that, the sign expression $(-1)^{(n-t)/2}$ is supposed to represent $\mathbf{I}_n^2 = \pm 1$, so that if $\mathbf{I}_n^2 = -1$, then the sign is changed. The sign expression for $\mathbf{I}_n^2$ is actually a little different than $(-1)^{(n-t)/2}$ for the general case, and we should have $\mathbf{I}_n^2 = (-1)^{n(n-1)/2}\mathbf{I}_n\tilde{\mathbf{I}}_n = (-1)^{n(n-1)/2}(-1)^q = (-1)^{(n^2-n+2q)/2} = (-1)^{(n^2-p+q)/2} = (-1)^{(n^2-t)/2}$. For example, in $\mathcal{G}_{5,2}$, the "signature" is $t = p - q = 5 - 2 = 3$, and $(-1)^{(7-t)/2} = (-1)^2 = 1$, but the correct sign is $(-1)^{(7^2-t)/2} = (-1)^{23} = -1$. The error of the sign (orientation) change in [11]:16 will

become more clear as we express the Hodge star dualization in terms of geometric algebra in the following. For now we note that, a dualization that is an involution has no orientation about which way the dualization is to be taken, and a dualization that is an anti-involution (such as J_e) has an orientation about which way the dualization and undualization is to be taken. Dualization using a unit pseudoscalar where $\mathbf{I}_n^2 = 1$ creates an involution, and dualization using a unit pseudoscalar where $\mathbf{I}_n^2 = -1$ creates an anti-involution. We will have the later case for J_e. We cannot arbitrarily change signs or orientations on any elements if we want to maintain correct orientations.

As an example, in Section 2.2 we used the identity $\mathbf{a} \wedge \mathbf{b}^* = -(\mathbf{a} \cdot \mathbf{b})\mathbf{I}_3$, which we may already guess is an example of the Hodge dual defining relation (53). It almost looks like it requires the sign change $(-1)^{(n^2-t)/2}$ [$(-1)^{(n-t)/2}$ fails], but remember that $\mathbf{b}^* = -\mathbf{b}\mathbf{I}_3$, so the dualization is defined using the unit pseudoscalar $-\mathbf{I}_3$, which has the sign on it. Therefore, the defining relation (53) holds without the sign (orientation) change $(-1)^{(n^2-t)/2}$.

It seems that, the Hodge star operation is not properly defined for a degenerate metric as we have in $\mathcal{G}_{3,0,1}$. Therefore, it is assumed that, to implement any form of Hodge dual operation on multivectors in $\mathcal{G}_{3,0,1}$, we must use a non-degenerate pseudo-Euclidean geometric algebra $\mathcal{G}_{p,q,0}$ acting as a copy of $\mathcal{G}_{3,0,1}$ but on a non-degenerate metric. A multivector M in $\mathcal{G}_{3,0,1}$ is copied to a corresponding multivector M in $\mathcal{G}_{p,q,0}$. The Hodge star dualization operation is then performed on the copied multivector M in $\mathcal{G}_{p,q,0}$ as $\star M$. The resulting Hodge dual multivector $\star M$ in $\mathcal{G}_{p,q,0}$ is copied back to a corresponding multivector $\star$M in $\mathcal{G}_{3,0,1}$ as the Hodge dual of M.

In geometric algebra $\mathcal{G}_{p,q,0}$ $(n = p + q)$, we often define the dual of multivector M as $M^* = M\mathbf{I}_n = M \cdot \mathbf{I}_n$ or $M^* = M/\mathbf{I}_n = M\mathbf{I}_n^{-1} = M \cdot \mathbf{I}_n^{-1}$, where $\mathbf{I}_n = \bigwedge^n$ is the unit pseudoscalar. The inverse pseudoscalar is $\mathbf{I}_n^{-1} = \mathbf{I}_n$ when $\mathbf{I}_n^2 = 1$, and $\mathbf{I}_n^{-1} = -\mathbf{I}_n$ when $\mathbf{I}_n^2 = -1$. It is also possible to define the dual as $M^* = \mathbf{I}_n M = \mathbf{I}_n \cdot M$ or $M^* = \mathbf{I}_n^{-1} M = \mathbf{I}_n^{-1} \cdot M$. These can each give a different dual. Multiplication with the unit pseudoscalar is always an inner product. Furthermore, we may take the unit pseudoscalar $\mathbf{I}_m$ of any subalgebra of $\mathcal{G}_{p,q,0}$ and sandwich M^* as $\mathbf{I}_m M^* \mathbf{I}_m$, $\mathbf{I}_m M^* \mathbf{I}_m^{-1}$, or $\mathbf{I}_m^{-1} M^* \mathbf{I}_m$ to obtain a modified dual $M^{*\prime}$, but without changing the grade relationship between duals, that a k-vector dualizes to a $(n-k)$-vector. We could compose many sandwich products to modify the dual further by pseudoscalar reflections (m odd) or rotations (m even). The sandwiching can alter signs (or orientations) and can change the dualization operation to be an involution $M^{**} = M$ or an anti-involution $M^{**} = -M$. The dualization $M^* = M\mathbf{I}_n$ is an involution $M^{**} = M$ for $\mathbf{I}_n^2 = 1$ and an anti-involution $M^{**} = -M$ for $\mathbf{I}_n^2 = -1$. For example, in $\mathcal{G}_{4,0,0}$ with 1-blade basis $\{\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ and pseudoscalars $\mathbf{I}_3 = \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$ and $\mathbf{I}_4 = \mathbf{e}_0\mathbf{I}_3$, the dualization $M^* = \mathbf{I}_4 M$ is an involution, but if we modify as $M^{*\prime} = \mathbf{I}_3 M^* \mathbf{I}_3 = \mathbf{e}_0 M \mathbf{I}_3$, then it is an anti-involution (we will make use of this example). The form $M^{*\prime}$ is dualization by a sandwiching product, not by only multiplication by the unit pseudoscalar, but in another algebra it is possible to find a dualization like M^* that acts like $M^{*\prime}$ on corresponding basis blades. Dualizations of the forms like M^* or $M^{*\prime}$ are dualization operations that each satisfy certain orientations and requirement to be an involution or anti-involution. We have determined the required entity dualization operation J_e for PGA $\mathcal{G}_{3,0,1}$, and now we seek to find a dualization of the form M^* or $M^{*\prime}$ in a non-degenerate pseudo-Euclidean algebra $\mathcal{G}_{p,q,0}$ that produces the dualization J_e on corresponding basis blades to those in PGA $\mathcal{G}_{3,0,1}$ as the implementation of J_e.

In geometric algebra $\mathcal{G}_{p,q,0}$ $(n=p+q)$, the Hodge star $\star$ dualization can be easily compared to the form of geometric algebra dualization M^*, which is a RHS or LHS multiplication with the unit pseudoscalar $\mathbf{I}=\mathbf{I}_n$ or its inverse $\mathbf{I}^{-1}$. It is more difficult to compare the Hodge star to the modified form $M^{*\prime}$ (a sandwiching product as dualization), yet the modified form $M^{*\prime}$ in one algebra may take the same duals as the form M^* in some other algebra when comparing corresponding basis blades. If we define the Hodge star operation as $\star\mathbf{A}=\mathbf{A}\mathbf{I}$, then we can check the Hodge star defining property $\mathbf{A}\wedge(\star\mathbf{B})=\star(\mathbf{A}\cdot\mathbf{B})$. We get

$$\mathbf{A}\wedge(\mathbf{B}\mathbf{I})=(\mathbf{A}\cdot\mathbf{B})\mathbf{I}. \tag{56}$$

Next, we dualize both sides to obtain $(\mathbf{A}\wedge(\mathbf{B}\mathbf{I}))\mathbf{I}=(\mathbf{A}\cdot\mathbf{B})\mathbf{I}^2$. The LHS can be rewritten as $(\mathbf{A}\wedge(\mathbf{B}\mathbf{I}))\cdot\mathbf{I}=\mathbf{A}\cdot((\mathbf{B}\mathbf{I})\cdot\mathbf{I})=(\mathbf{A}\cdot\mathbf{B})\mathbf{I}^2$. So, the Hodge star matches this form of dualization. We can also state the Hodge star defining relation as $(\star\mathbf{A})\wedge\mathbf{B}=\star(\mathbf{A}\cdot\mathbf{B})$, where we then define $\star\mathbf{A}=\mathbf{I}\mathbf{A}$ and

$$(\mathbf{I}\mathbf{A})\wedge\mathbf{B}=\mathbf{I}(\mathbf{A}\cdot\mathbf{B}). \tag{57}$$

Then, we check the Hodge star defining relation as $\mathbf{I}((\mathbf{I}\mathbf{A})\wedge\mathbf{B})=\mathbf{I}^2(\mathbf{A}\cdot\mathbf{B})$. The LHS expands to $\mathbf{I}^2(\mathbf{A}\cdot\mathbf{B})$, so it works. Using the identities

$$\mathbf{I}\mathbf{A}=(-1)^{k(n-1)}\mathbf{A}\mathbf{I} \tag{58}$$

and

$$(\mathbf{A}\mathbf{I})\wedge\mathbf{B}=(-1)^{k(n-k)}\mathbf{B}\wedge(\mathbf{A}\mathbf{I}), \tag{59}$$

it can be shown that $\mathbf{A}\wedge(\star\mathbf{B})=\mathbf{B}\wedge(\star\mathbf{A})=\star(\mathbf{A}\cdot\mathbf{B})$ corresponds to

$$\mathbf{A}\wedge(\mathbf{B}\mathbf{I})=\mathbf{B}\wedge(\mathbf{A}\mathbf{I})=(\mathbf{A}\cdot\mathbf{B})\mathbf{I}, \tag{60}$$

since $(-1)^{k(n-1)}(-1)^{\pm k(n-k)}=1$. Therefore, we can say that the form M^* is the same as a Hodge star dualization. We will also say that the modified form $M^{*\prime}$ is also a Hodge star dualization, but only after another algebra is found in which the form M^* gives the same duals as $M^{*\prime}$ on corresponding basis blades. Notice that, the Hodge star dualization is either an involution for $\mathbf{I}^2=1$ or an anti-involution for $\mathbf{I}^2=-1$, and it cannot be a mixture of them where it is an involution for dualizing some basis blades and an anti-involution for dualizing other basis blades. We consider the Hodge star $\star$ dualization as an abstraction and generalization of the geometric algebra dualization M^* (and also $M^{*\prime}$ when it matches the form M^* in another algebra) that extends to the implementation of a dualization for a degenerate geometric algebra using a corresponding non-degenerate geometric algebra.

In geometric algebra $\mathcal{G}_{p,q,0}$ $(n=p+q)$, the Hodge star $\star$ dual of a k-vector (or k-blade) $\mathbf{A}$ is the $(n-k)$-vector $\star\mathbf{A}=\mathbf{A}^*$, where $\mathbf{A}^*$ (or $\mathbf{A}^{*\prime}$) is a form of dualization like M^* (or $M^{*\prime}$) in a non-degenerate pseudo-Euclidean algebra $\mathcal{G}_{p,q,0}$, which may be made to correspond, basis-blade to basis-blade, with a degenerate algebra $\mathcal{G}_{p,q,r}$, as just discussed. We now change notation and suggest the notation $\star\mathbf{A}=\mathbf{A}^\star$ instead of $\star\mathbf{A}=\mathbf{A}^*$. In PGA $\mathcal{G}_{3,0,1}$, we reserve the notation $\mathbf{A}^*$ for the case where $\mathbf{A}\in\mathcal{G}_3$, and we define

$$\mathbf{A}^*=\mathbf{A}/\mathbf{I}_3. \tag{61}$$

Recall that, we cannot actually perform the dual $\mathbf{A}^\star$ in the degenerate metric of PGA $\mathcal{G}_{3,0,1}$, so we suggest that one could define

$$\mathbf{A}^\star=J_e(\mathbf{A})=\boldsymbol{a} \tag{62}$$

for dualization orientation from OPNS PGA to CPNS PGA, and

$$\boldsymbol{a}^{-\star} = -J_e(\boldsymbol{a}) = D_e(\boldsymbol{a}) = \mathbf{A} \tag{63}$$

for dualization orientation from CPNS PGA to OPNS PGA. We also have the notations

$$\mathbf{A}^{-\star\star} = \mathbf{A} \text{ and } \mathbf{A}^{\star\star} = -\mathbf{A}. \tag{64}$$

If one prefers, the opposite dualization orientation could be defined as $\boldsymbol{a}^{\star} = -J_e(\boldsymbol{a}) = D_e(\boldsymbol{a}) = \mathbf{A}$ for dualization from CPNS PGA to OPNS PGA. For notation, we will continue to mostly just use J_e and D_e to avoid confusion, but we will use the suggested notation some. In the suggested notation $\mathbf{A}^{\star} = J_e(\mathbf{A})$, for the case where $\mathbf{A} \in \mathcal{G}_3$, one could write expressions such as $\mathbf{A}^{*} = \mathbf{A}\,/\,\mathbf{I}_3$ and $\mathbf{A}^{\star} = -\mathbf{e}_0\mathbf{A}^{*}$. In the suggested notation, we also have (for example) $\mathbf{P}_{\mathbf{t}}^{\star} = (\mathbf{e}_0 + \mathbf{t})^{\star} = \mathbf{I}_3 - \mathbf{e}_0\mathbf{t}^{*} = \boldsymbol{p}_{\mathbf{t}}$, which is the entity dualization of the OPNS PGA point $\mathbf{P}_{\mathbf{t}}$ to the CPNS PGA point $\boldsymbol{p}_{\mathbf{t}}$.

In [17], the dualization operation denoted $J(\mathbf{e})$ for PGA $\mathcal{G}_{3,0,1}$ appears to be a mixture of involution for some basis blades and anti-involution for other basis blades. Therefore, it seems that $J(\mathbf{e})$ in [17] cannot be a Hodge dual as we have defined it. Reciprocals do not exist in $\mathcal{G}_{3,0,1}$ since $\mathbf{I}_4^{-1}$ does not exist, so we find it difficult to accept the notation of reciprocal basis blades in [17]. The dualization for geometric entities in PGA $\mathcal{G}_{3,0,1}$ that we observe as J_e (and have called the empirical dualization or entity dualization) is clearly an anti-involution, so we find it difficult to accept $J(\mathbf{e})$ or any of the dualization operations given in [17] as being the correct dualization for PGA $\mathcal{G}_{3,0,1}$. We accept our empirically determined dualization operation J_e as the correct dualization operation for the geometric entities in PGA $\mathcal{G}_{3,0,1}$.

In [21], a Hodge star dualization notation ($\star 1 = \mathbf{e}_0\mathbf{e}_2\mathbf{e}_3\mathbf{e}_4$ etc.) is used for their dualization operation in PGA $\mathcal{G}_{3,0,1}$, but some basis blades are dualized by an involution and other basis blades are dualized by an anti-involution. It seems that, this cannot actually be a Hodge star dualization operation as we have defined it. Furthermore, the table of duals given in [21] matches the table of duals given in [17], which we find difficult to accept as the correct duals.

In the older paper [23], which uses the similar algebra $\mathcal{G}_{0,3,1}$, a Hodge star $\star$ notation is also used for the dualization operation. The table of duals given in [23] also shows some basis blades dualized by an involution and other basis blades dualized by an anti-involution. Again, we find it difficult to accept a mixture of involution and anti-involution as the correct dualization. Although [23] uses $\mathcal{G}_{0,3,1}$, it is likely that other later papers were based closely on [23] while using $\mathcal{G}_{3,0,1}$. The mistakes seem to have been carried forward into later papers.

We begin calling $J_e(\mathbf{A}) = \mathbf{A}^{\star}$ the *entity dualization* operation, as we have now completed our empirical observations and determination of J_e (in Sections 4.2.1, 4.2.2, and 4.2.3), tabulation of J_e (in Section 4.2.4), and conclusions on J_e as an anti-involution that should be implementable as a Hodge star $\star$ dualization (unit pseudoscalar dualization of form M^{*} or sandwiching of form $M^{*\prime}$) in a non-degenerate geometric algebra $\mathcal{G}_{p,q,0}$ that can be made to correspond to PGA $\mathcal{G}_{3,0,1}$. In the next section, we find these dualizations of form M^{*} or $M^{*\prime}$ in three different geometric algebras $\{\mathcal{G}_{4,0,0}, \mathcal{G}_{3,1,0}, \mathcal{G}_{1,3,0}\}$ made to correspond to PGA $\mathcal{G}_{3,0,1}$ and use each of them as a possible implementation for J_e in PGA $\mathcal{G}_{3,0,1}$.

4.3 Methods for Implementing the Entity Dualization

In this section we give three methods for implementing the entity dualization J_e as in Table 1. It is not possible to implement J_e directly within $\mathcal{G}_{3,0,1}$ by any algebraic method due to its degenerate metric (i.e., $\mathbf{e}_0^2 = 0$). We first look at $\mathcal{G}_{4,0,0}$ and find that J_e can be implemented within it even though its unit pseudoscalar has the involution characteristic $\mathbf{I}_4^2 = 1$. Next, we look at $\mathcal{G}_{3,1}$, sometimes called Time-Space Algebra, which has the correct unit pseudoscalar characteristic $\mathbf{I}_4^2 = -1$ and we find an implementation for J_e. Finally, we look at $\mathcal{G}_{1,3}$, called Space-Time Algebra [18], which also has the correct unit pseudoscalar characteristic $\mathbf{I}_4^2 = -1$ and we find an implementation for J_e.

4.3.1 Entity Dualization in G(4,0,0)

In $\mathcal{G}_{4,0,0}$ with metric diag$(1,1,1,1)$, using the 1-blade basis $\{\boldsymbol{e}_0, \boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$, we have the unit pseudoscalars $\boldsymbol{I}_3 = \boldsymbol{e}_1\boldsymbol{e}_2\boldsymbol{e}_3$ and $\boldsymbol{I}_4 = \boldsymbol{e}_0\boldsymbol{I}_3$, where $\boldsymbol{I}_3^2 = -1$ and $\boldsymbol{I}_4^2 = 1$. The corresponding notation for PGA $\mathcal{G}_{3,0,1}$ is $\{\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, $\mathbf{I}_3$, and $\mathbf{I}_4$. Please notice that upright bold letters (e.g., $\mathbf{I}$) are for PGA $\mathcal{G}_{3,0,1}$, and italic bold letters (e.g., $\boldsymbol{I}$) are for $\mathcal{G}_{4,0,0}$. The idea is that, we can think of the italic bold symbols for the basis blades of $\mathcal{G}_{4,0,0}$ as indicating a *pretend basis*, pretending to be PGA $\mathcal{G}_{3,0,1}$ but with a different non-degenerate metric. We pretend for long enough to run entity dualization operations in $\mathcal{G}_{4,0,0}$, then we stop pretending and switch back to PGA $\mathcal{G}_{3,0,1}$.

Clearly, there is a direct correspondence between the basis blades of PGA $\mathcal{G}_{3,0,1}$ and $\mathcal{G}_{4,0,0}$, allowing for a direct transfer of basis blade coordinates between the two algebras. This transfer of coordinates should be easily implemented in most software implementations. For example, in *GAlgebra* for *SymPy* [3], we can take an entity $\mathbf{A}$ in PGA $\mathcal{G}_{3,0,1}$ and transfer it to a corresponding entity $\boldsymbol{A}$ in $\mathcal{G}_{4,0,0}$ by corresponding coordinates transfer as $\boldsymbol{A} = \mathcal{G}_{4,0,0}(\mathbf{A})$, although this is written in the Python code slightly differently. We also transfer $\mathbf{I}_3$ and $\mathbf{I}_4$ as $\boldsymbol{I}_3 = \mathcal{G}_{4,0,0}(\mathbf{I}_3)$ and $\boldsymbol{I}_4 = \mathcal{G}_{4,0,0}(\mathbf{I}_4)$.

Now, using the corresponding elements $\boldsymbol{A}$, $\boldsymbol{I}_3$, and $\boldsymbol{I}_4$ in $\mathcal{G}_{4,0,0}$, we form the "pretend" *entity dualization* of $\boldsymbol{A}$ in $\mathcal{G}_{4,0,0}$, denoted $\mathcal{J}_e(\boldsymbol{A})$, as

$$\mathcal{J}_e(\boldsymbol{A}) = \boldsymbol{I}_3\boldsymbol{I}_4\boldsymbol{A}\boldsymbol{I}_3 = \boldsymbol{e}_0\boldsymbol{A}\boldsymbol{I}_3. \tag{65}$$

Next, we "stop pretending" and switch (or transfer) the result back to PGA $\mathcal{G}_{3,0,1}$ as $\mathbf{A}^\star = \mathcal{G}_{3,0,1}(\mathcal{J}_e(\boldsymbol{A}))$. So, the complete entity dualization operation is

$$\mathbf{A}^\star = J_e(\mathbf{A}) = \mathcal{G}_{3,0,1}(\mathcal{J}_e(\mathcal{G}_{4,0,0}(\mathbf{A}))). \tag{66}$$

This is a dualization by a sandwiching product as discussed in Section 4.2.5 and it is not directly in the form of a Hodge star $\star$ dualization, but it is equivalent to the dualizations in Sections 4.3.2 and 4.3.3 that are Hodge star dualizations, so we consider this dualization to also be a Hodge star dualization in the abstract and general sense.

Inside of the implementation of $\mathcal{J}_e$, as a defined function, it may need to first perform the transfers $\boldsymbol{I}_3 = \mathcal{G}_{4,0,0}(\mathbf{I}_3)$ and $\boldsymbol{I}_4 = \mathcal{G}_{4,0,0}(\mathbf{I}_4)$, which are needed for the dualization product. Depending on the geometric algebra software implementation, it may or may not be possible (or allowed) to simply alter the underlying metric of an algebra, run a single computation such as the dualization, and then change the metric back (assuming no other computations run in parallel on the wrong metric). In Python code[2], an example of running the entity dualization is as follows:

```
g301=Ga('e*0|1|2|3',g=[0,1,1,1]); g400=Ga('e*0|1|2|3',g=[1,1,1,1])
```

2. See Appendix A for more Python code that was used in the research.

```
(e0,e1,e2,e3)=g301.mv(); I3=e1*e2*e3; I4=e0*I3

def Je(A):
    EA=g400.mv(A); EI3=g400.mv(I3); EI4=g400.mv(I4)
    return g301.mv(EI3*EI4*EA*EI3)

Je(e0)
e1^e2^e3
```

It can be verified that this implementation of the entity dualization J_e (as Je) matches exactly Table 1. This dualization uses the form $M^{*\prime}$ (a sandwiching product) as discussed in Section 4.2.5. We have two more ways to implement J_e (using the form M^*), given briefly in the next two sections. The procedure is similar, so we do not repeat all of the details and just give formulas.

4.3.2 Entity Dualization in G(3,1,0)

In $\mathcal{G}_{3,1,0}$ with metric diag$(-1,1,1,1)$, using the 1-blade basis $\{\boldsymbol{e}_0,\boldsymbol{e}_1,\boldsymbol{e}_2,\boldsymbol{e}_3\}$, we have the unit pseudoscalars $\boldsymbol{I}_3=\boldsymbol{e}_1\boldsymbol{e}_2\boldsymbol{e}_3$ and $\boldsymbol{I}_4=\boldsymbol{e}_0\boldsymbol{I}_3$, where $\boldsymbol{I}_3^2=-1$ and $\boldsymbol{I}_4^2=-1$. The corresponding notation for PGA $\mathcal{G}_{3,0,1}$ is $\{\mathbf{e}_0,\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3\}$, $\mathbf{I}_3$, and $\mathbf{I}_4$. Please notice that upright bold letters (e.g., $\mathbf{I}$) are for PGA $\mathcal{G}_{3,0,1}$, and italic bold letters (e.g., $\boldsymbol{I}$) are for $\mathcal{G}_{3,1,0}$. The idea is that, we can think of the italic bold symbols for the basis blades of $\mathcal{G}_{3,1,0}$ as indicating a *pretend basis*, pretending to be PGA $\mathcal{G}_{3,0,1}$ but with a different non-degenerate metric. We pretend for long enough to run entity dualization operations in $\mathcal{G}_{3,1,0}$, then we stop pretending and switch back to PGA $\mathcal{G}_{3,0,1}$. The complete dualization operation for $\mathbf{A}\in\mathcal{G}_{3,0,1}$ is

$$\mathbf{A}^\star=J_e(\mathbf{A})=\mathcal{G}_{3,0,1}(\mathcal{J}_e(\mathcal{G}_{3,1,0}(\mathbf{A}))), \tag{67}$$

where

$$\mathcal{J}_e(\boldsymbol{A})=\boldsymbol{I}_4\boldsymbol{A} \tag{68}$$

and $\boldsymbol{A}=\mathcal{G}_{3,1,0}(\mathbf{A})$. This is a Hodge star $\star$ dualization in the form M^* as discussed in Section 4.2.5. For $\mathbf{A}\in\mathcal{G}_3$, it is easy to see that $\mathbf{A}^\star=\mathbf{e}_0\mathbf{I}_3\mathbf{A}=-\mathbf{e}_0(\mathbf{A}\,/\,\mathbf{I}_3)=-\mathbf{e}_0\mathbf{A}^*$. It can be verified that this implementation of the entity dualization J_e matches exactly Table 1.

4.3.3 Entity Dualization in G(1,3,0)

In $\mathcal{G}_{1,3,0}$ with metric diag$(1,-1,-1,-1)$, using the 1-blade basis $\{\boldsymbol{e}_0,\boldsymbol{e}_1,\boldsymbol{e}_2,\boldsymbol{e}_3\}$, we have the unit pseudoscalars $\boldsymbol{I}_3=\boldsymbol{e}_1\boldsymbol{e}_2\boldsymbol{e}_3$ and $\boldsymbol{I}_4=\boldsymbol{e}_0\boldsymbol{I}_3$, where $\boldsymbol{I}_3^2=-1$ and $\boldsymbol{I}_4^2=-1$. The corresponding notation for PGA $\mathcal{G}_{3,0,1}$ is $\{\mathbf{e}_0,\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3\}$, $\mathbf{I}_3$, and $\mathbf{I}_4$. Please notice that upright bold letters (e.g., $\mathbf{I}$) are for PGA $\mathcal{G}_{3,0,1}$, and italic bold letters (e.g., $\boldsymbol{I}$) are for $\mathcal{G}_{1,3,0}$. The idea is that, we can think of the italic bold symbols for the basis blades of $\mathcal{G}_{1,3,0}$ as indicating a *pretend basis*, pretending to be PGA $\mathcal{G}_{3,0,1}$ but with a different non-degenerate metric. We pretend for long enough to run entity dualization operations in $\mathcal{G}_{1,3,0}$, then we stop pretending and switch back to PGA $\mathcal{G}_{3,0,1}$. The complete dualization operation for $\mathbf{A}\in\mathcal{G}_{3,0,1}$ is

$$\mathbf{A}^\star=J_e(\mathbf{A})=\mathcal{G}_{3,0,1}(\mathcal{J}_e(\mathcal{G}_{1,3,0}(\mathbf{A}))), \tag{69}$$

where

$$\mathcal{J}_e(\boldsymbol{A}) = \boldsymbol{A}\boldsymbol{I}_4 \tag{70}$$

and $\boldsymbol{A} = \mathcal{G}_{1,3,0}(\mathbf{A})$. This is a Hodge star $\star$ dualization in the form M^* as discussed in Section 4.2.5. It can be verified that this implementation of the entity dualization J_e matches exactly Table 1.

4.4 Conclusion on Geometric Entity Dualization in PGA

Table 1 defines the geometric entity dualization operation J_e based on direct observation, comparing dual geometric entities of same orientation and noting the duals of basis blades within the dual geometric entities. We found that J_e is an anti-involution with inverse

$$J_e^{-1} = -J_e. \tag{71}$$

For any geometric entity (or basis blade) $\mathbf{A}$ of grade k in OPNS PGA, $J_e(\mathbf{A}) = \mathbf{A}^\star = \boldsymbol{a}$ of grade $4-k$ is its dual geometric entity in CPNS PGA representing the same geometry and with the same orientation. The geometric entity dualization operation

$$D_e = -J_e \tag{72}$$

is for the opposite orientation of the dualization, dualizing any geometric entity in CPNS PGA to its dual geometric entity in OPNS PGA, again with undual (or inverse) $D_e^{-1} = -D_e$.

We found three methods for $\mathcal{J}_e$, that implements J_e, in three different non-degenerate geometric algebras that are able to correspond to PGA $\mathcal{G}_{3,0,1}$. We have to transfer to a non-degenerate geometric algebra to implement J_e as $\mathcal{J}_e$ for the dualization since PGA $\mathcal{G}_{3,0,1}$ with degenerate metric cannot directly implement its own dualization J_e by multiplying into its degenerate unit pseudoscalar $\mathbf{I}_4^2 = 0$.

$\mathcal{G}_{p,q,0}$	$\mathcal{G}_{4,0,0}$	$\mathcal{G}_{3,1,0}$	$\mathcal{G}_{1,3,0}$
$\mathcal{J}_e(\boldsymbol{A})$	$\boldsymbol{I}_3\boldsymbol{I}_4\boldsymbol{A}\boldsymbol{I}_3 = \boldsymbol{e}_0\boldsymbol{A}\boldsymbol{I}_3$	$\boldsymbol{I}_4\boldsymbol{A}$	$\boldsymbol{A}\boldsymbol{I}_4$

Table 2. Entity dualization $\mathcal{J}_e(\boldsymbol{A})$ in non-degenerate geometric algebras $\mathcal{G}_{p,q,0}$.

Table 2 summarizes the entity dualization implementations $\mathcal{J}_e$ as performed in three different corresponding non-degenerate geometric algebras. $\mathcal{J}_e$ implements J_e, but while using a non-degenerate geometric algebra corresponding to PGA $\mathcal{G}_{3,0,1}$. It can be seen that each form of $\mathcal{J}_e$ is a linear operation, able to act on an entire entity $\boldsymbol{A} \in \mathcal{G}_{p,q,0}$ (in one of the three algebras of Table 2) that corresponds to an entity $\mathbf{A} \in \mathcal{G}_{3,0,1}$ by direct transfer of coordinates from $\mathbf{A}$ into $\boldsymbol{A}$ on corresponding basis blades. Then, $\mathcal{J}_e(\boldsymbol{A})$ represents the correct dual, which is then transferred back onto corresponding basis blades in $\mathcal{G}_{3,0,1}$ as the dual entity

$$\mathbf{A}^\star = J_e(\mathbf{A}) = \mathcal{G}_{3,0,1}(\mathcal{J}_e(\mathcal{G}_{p,q,0}(\mathbf{A}))) = \mathcal{G}_{3,0,1}(\mathcal{J}_e(\boldsymbol{A})) = \mathcal{G}_{3,0,1}(\boldsymbol{A}^\star). \tag{73}$$

The form of $\mathcal{J}_e$ in $\mathcal{G}_{1,3,0}$, as $\mathcal{J}_e(\boldsymbol{A}) = \boldsymbol{A}\boldsymbol{I}_4$, is most like the usual geometric algebra (or Hodge star $\star$) dualization. The $\mathcal{J}_e$ are each anti-involutions, where usually $\boldsymbol{I}_4^2 = -1$. In $\mathcal{G}_{4,0,0}$, where $\boldsymbol{I}_4^2 = 1$, the form of $\mathcal{J}_e$ is the most unusual, but it works completely the same as the others as the dualization.

Having the PGA entity dualization operation J_e easily implemented in its correct and reliable form overcomes many of the possible difficulties with using PGA $\mathcal{G}_{3,0,1}$. One of those difficulties can be as simple as extracting the scalar y from $y\mathbf{e}_0$, which we can now easily do as $y = J_e(y\mathbf{e}_0)/\mathbf{I}_3$. In the next section, we discuss dual quaternions in geometric algebra, where we make significant usage of J_e. The geometric entity dualization operation J_e allows to dualize entities from OPNS PGA to CPNS PGA, and the opposite direction using $D_e = -J_e$. We will also be able to convert CPNS PGA entities to forms in the dual quaternion geometric algebra of $\mathcal{G}^+_{3,0,1}$, which has some nice algebraic properties and operations. Therefore, we can easily use the whole PGA algebra, now that we have the PGA entity dualization operation J_e.

5 Dual Quaternion Geometric Algebra in PGA

This section is about the Dual Quaternion Geometric Algebra $\mathcal{G}^+_{3,0,1}$ in PGA $\mathcal{G}_{3,0,1}$, DQGA/PGA. In the prior sections, we developed PGA enough that we can now use it to examine the even-grades subalgebra $\mathcal{G}^+_{3,0,1}$ that represents, or emulates faithfully, the dual quaternion algebra, DQA. All of the results derived in DQGA that involve only corresponding elements of DQA can be used outside of geometric algebra in a purely DQA implementation. Using DQGA, we emulate DQA in the even-grades subalgebra $\mathcal{G}^+_{3,0,1}$ and implement certain special operations, such as conjugates, taking advantage of PGA and the PGA entity dualization operation J_e.

5.1 Introduction to Dual Quaternion Geometric Algebra

In DQA, dual quaternions are linear combinations of basis elements $\{1, \varepsilon, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$. The element ε is the nilpotent scalar ($\varepsilon^2 = 0$) that has commutative multiplication (as does the scalar 1) with all other elements. The elements $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are pure quaternion unit vectors with the quaternion product rule $\mathbf{ijk} = \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$ and the usual vector calculus dot (bold dot $\cdot$) and cross (bold cross $\times$) products.

In DQA, there is a homogeneous point representation $p_\mathbf{p} = 1 + \varepsilon\mathbf{p}$, embedding a vector point $\mathbf{p}$, with rotation $R = \exp(\theta\hat{\mathbf{n}}/2)$ and translation $T = \exp(\mathbf{d}\varepsilon/2)$ operations on the homogeneous point as $p'_\mathbf{p} = Rp_\mathbf{p}R^{-1}$ and $p'_\mathbf{p} = Tp_\mathbf{p}T = p_{\mathbf{p}+\mathbf{d}}$, respectively. The rotor R acts as a versor on a point p, just as it does on quaternions. The translator T is not acting as a versor, but it acts as just another homogeneous point $T = \exp(\mathbf{d}\varepsilon/2) = 1 + \mathbf{d}\varepsilon/2 = p_{\mathbf{d}/2}$. The points are in the form of unit magnitude dual numbers (but these are dual quaternions) that multiply commutatively with magnitudes multiplying and "angles" adding. The magnitude is 1, so there is no scaling. The "angles" are pure quaternion vectors $\mathbf{p}$ and $\mathbf{d}$. We could just as well write $TTp_\mathbf{p} = p_{\mathbf{d}/2}p_{\mathbf{d}/2}p_\mathbf{p} = p_{\mathbf{p}+\mathbf{d}}$, and the order of multiplication does not matter. The translator T and rotor R do not commute unless $\mathbf{d} = \alpha\hat{\mathbf{n}}$, and then $RT = TR$. In any case, RT cannot act as a versor since $(RT)^{-1} = T^{-1}R^{-1}$ and then the translation is by 0 displacement. We can usefully compose translation and rotation to rotate about a center $\mathbf{c}$ as $p'_\mathbf{t} = p_\mathbf{c}Rp_\mathbf{t}p_{-\mathbf{c}}R^{-1}$. The dual quaternion algebra, having a versor rotation operator R and a non-versor translation operator $T = p_\mathbf{d}$ as just another homogeneous point embedding, does not have the elegance of being able to compose rotation and translation more generally as versors.

To use dual quaternions, we soon become involved with cumbersome formulas involving the use of complex and quaternion conjugates, and other special operations, rather than simple versor sandwich products, outer products, or commutator products. Nevertheless, more is possible in dual quaternions than is commonly known. In the sections that follow, we will show that, not only is there the dual quaternion point p embedding, but there are also a line l and a plane π embedding. Furthermore, we can rotate and translate all of these embedded entities. Although the formulas are not so elegant, in theory it is possible to do most of what PGA can do within just the dual quaternion algebra, which is a smaller algebra than the full PGA. The dual quaternion algebra is just the even subalgebra of $\mathcal{G}_{3,0,1}$. The formulas may be of interest to those who want to use dual quaternions for points, lines, and planes.

In the following sections, we review dual numbers, quaternions, and dual quaternions and show how they are represented in the DQGA of the even-grades subalgebra $\mathcal{G}^+_{3,0,1}$ of PGA, using only the eight even-grade basis blades of $\mathcal{G}_{3,0,1}$, which are $\{1, \mathbf{I}_4, \mathbf{e}^*_1 = \mathbf{e}_3\mathbf{e}_2, \mathbf{e}^*_2 = \mathbf{e}_1\mathbf{e}_3, \mathbf{e}^*_3 = \mathbf{e}_2\mathbf{e}_1, \mathbf{e}^*_1\mathbf{I}_4 = \mathbf{e}_0\mathbf{e}_1, \mathbf{e}^*_2\mathbf{I}_4 = \mathbf{e}_0\mathbf{e}_2, \mathbf{e}^*_3\mathbf{I}_4 = \mathbf{e}_0\mathbf{e}_3\}$. Then, we derive the representations of the dual quaternion point p, line l, and plane π and their rotation, translation, and other operations.

5.2 Dual Numbers in PGA

In this section, we discuss dual number algebra (DNA), and then its representation in geometric algebra $\mathcal{G}^+_{3,0,1}$ as what we will call dual number geometric algebra (DNGA). Note that, it is also possible to represent dual numbers in the subalgebra $\mathcal{G}_{0,0,1}$ of $\mathcal{G}_{3,0,1}$, but that representation does not fulfill our algebraic requirements for dual quaternion geometric algebra.

5.2.1 Dual Number Algebra

Dual numbers, also sometimes called parabolic numbers, are a type of complex numbers with exponential form. Recall that, there are three kinds of complex numbers using either $i^2 = -1$, $j^2 = 1$, or $\varepsilon^2 = 0$, and forms of these numbers are used in geometric algebras for rotation, dilation (or boost / hyperbolic rotation), and translation operations, respectively. We will be concerned with the rotation and translation operations in PGA and DQA.

In dual number algebra, a dual number z is written as the sum of a real number part x and an imaginary part $y\varepsilon$ as

$$z = x + y\varepsilon, \tag{74}$$

where y is also a real number. The imaginary scalar unit ε is defined to square to zero, $\varepsilon^2 = 0$, and is also called a nilpotent scalar. This is very much like the usual complex numbers $x + yi$ with $i^2 = -1$. The algebra is very similar, and the points (x, y) can be graphed on an xy-plane. A dual number z has a complex conjugate

$$\bar{z} = x - y\varepsilon, \tag{75}$$

magnitude

$$|z| = \sqrt{z\bar{z}} = |x|, \tag{76}$$

and exponential form

$$z = x\exp(y\varepsilon/x) = x(1 + y\varepsilon/x). \tag{77}$$

We will use only dual numbers with unit magnitude with $x=1$, $z=1+y\varepsilon=\exp(y\varepsilon)$, and the product of two dual numbers in this unit form is

$$(1+y_1\varepsilon)(1+y_2\varepsilon)=\exp(y_1\varepsilon)\exp(y_2\varepsilon)=1+(y_1+y_2)\varepsilon=\exp((y_1+y_2)\varepsilon). \tag{78}$$

This unit form allows the important trick, that addition is performed as multiplication, which is used extensively in many geometric algebras for translation operations. Though not used in this paper, dual numbers also allow a trick to take derivatives of polynomial functions (e.g., $f\in\mathbb{R}[x],\ f(x+\varepsilon)=f(x)+f'(x)\varepsilon$).

If we have any dual number $z=r+l\varepsilon$ and multiply it by $R=\exp(\theta\varepsilon)$, we get $Rz=\exp(\theta\varepsilon)r\exp(l\varepsilon/r)=r\exp(\theta\varepsilon)\exp(\theta_z\varepsilon)=r\exp((\theta+l/r)\varepsilon)=r+(l+r\theta)\varepsilon$. Here, r is the radius and $l=r\theta_z$ is the arc length of z. R is a parabolic rotor for rotation by angle θ. In the product Rz, the radius r is unchanged by the rotation, and the added arc length of the rotation along the "parabola" is $r\theta$, or we can say that the angles add as $\theta_z+\theta$. This is similar to the other unimodular exponential operators for circular (elliptic) rotation using a complex (elliptic) number $\exp(\theta i)$, and hyperbolic rotation using a split-complex (hyperbolic) number $\exp(\theta j)$ where $j^2=1$. In each case, the angle θ is the arc length added per radius ($\Delta l=r\theta$) along a circle, hyperbola, or degenerate parabola (a line, but more like a degenerate hyperbola that has opened so large that it is a line). The parabolic rotation is along the line $x=r$, rotating (r,l) into $(r,l+r\theta=r(\theta_z+\theta))$. We can think of having parabolic cosine $\mathrm{cosp}(\theta)$, parabolic sine $\mathrm{sinp}(\theta)$, and parabolic tangent $\mathrm{tanp}(\theta)$ functions, similar to $\sin(\theta)$ etc. and $\sinh(\theta)$ etc. for elliptic and hyperbolic numbers. Then, we have $\exp(l\varepsilon/r)=\exp(\theta_z\varepsilon)=1+l\varepsilon/r=\mathrm{cosp}(\theta_z)+\mathrm{sinp}(\theta_z)\varepsilon$, where $\mathrm{cosp}(\theta_z)=1$, $\mathrm{sinp}(\theta_z)=\theta_z=l/r=\mathrm{tanp}(\theta_z)$. The very small angle θ approximation, $\sin(\theta)=\theta$, for moving a very small and nearly straight line of arc length upward the unit circle, is similar to $\mathrm{sinp}(\theta_z)=\theta_z$, for moving absolutely up the straight line of a degenerate parabola (or hyperbola).

In CGA $\mathcal{G}_{4,1}$, we encounter the parabolic nature of dual numbers in a different form when we use the CGA translator $\exp(\mathbf{e}_\infty\mathbf{d}/2)$ to translate the CGA point entity $\mathbf{x}+\frac{1}{2}\mathbf{x}^2\mathbf{e}_\infty+\mathbf{e}_o$ to $\mathbf{x}'+\frac{1}{2}\mathbf{x}'^2\mathbf{e}_\infty+\mathbf{e}_o$, which clearly has a parabolic form in $\mathbf{x}$, known to cut a null parabola from the null cone on the $\mathbf{e}_o$ hyperplane.

5.2.2 Dual Number Geometric Algebra

In $\mathcal{G}^+_{3,0,1}$, we have the basis 0-blade 1 and basis 4-blade unit pseudoscalar $\mathbf{I}_4$, where $\{1,\mathbf{I}_4\}\hat{=}\{1,\varepsilon\}$, and we can represent a dual number $z=r+l\varepsilon$ as

$$z=r+l\mathbf{I}_4=x+y\mathbf{I}_4. \tag{79}$$

The geometric products of dual numbers in this form obey all the usual rules of dual numbers since $\mathbf{I}_4^2=0$ and $\mathbf{I}_4$ commutes with all other elements in the even-grades subalgebra $\mathcal{G}^+_{3,0,1}$, acting similar to the nilpotent scalar ε. We can implement the complex conjugate as

$$\bar{z}=\mathbf{I}_3 z\mathbf{I}_3^{-1}. \tag{80}$$

Following the usual formulas for complex numbers, we can take the real part of z as

$$\Re(z)=(z+\bar{z})/2, \tag{81}$$

and the imaginary part as

$$\Im(z)=(z-\bar{z})/2. \tag{82}$$

To extract the real number l from $\Im(z) = l\mathbf{I}_4$, we can take advantage of the dualization operation J_e and implement this as

$$y = l = \mathrm{Y}(z) = -J_e(\Im(z)). \tag{83}$$

If we like, we can say $x = r = \mathrm{X}(z) = \Re(z)$.

We can refer to dual numbers on the basis $\{1, \varepsilon\}$ as dual number algebra (DNA), and dual numbers on the basis $\{1, \mathbf{I}_4\}$ as dual number geometric algebra (DNGA).

Next, we discuss quaternions and how they are also represented in the even-grades subalgebra $\mathcal{G}^+_{3,0,1}$ of PGA along with the dual numbers representation just discussed.

5.3 Quaternions in PGA

In this section, we review the quaternion algebra (GA) in its original form as introduced by William Rowan Hamilton in 1843, and then we discuss its representation in the even-grades subalgebra $\mathcal{G}^+_3$, which we will call quaternion geometric algebra (QGA).

5.3.1 Quaternion Algebra

In quaternion algebra (QA), quaternions are defined as linear combinations of the basis elements $\{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$ with product rule

$$\mathbf{ijk} = \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1. \tag{84}$$

A general quaternion q is written

$$q = q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k} = q_w + \mathbf{q}. \tag{85}$$

The vectors $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ are used as a vector basis for $\mathbb{R}^3$, 3D space, with points $(q_x, q_y, q_z) = (x, y, z)$. A consequence of the product rule is that $\mathbf{i}^{-1} = -\mathbf{i}$, and similarly for $\mathbf{j}$ and $\mathbf{k}$, and we have $\mathbf{k} = \mathbf{j}/\mathbf{i} = \mathbf{j}\mathbf{i}^{-1} = -\mathbf{j}\mathbf{i}$, $\mathbf{j} = \mathbf{i}/\mathbf{k}$, and $\mathbf{i} = \mathbf{k}/\mathbf{j}$. These ratios are more often written as the cross products $\mathbf{k} = -\mathbf{j}\times\mathbf{i} = \mathbf{i}\times\mathbf{j}$, $\mathbf{j} = \mathbf{k}\times\mathbf{i}$, and $\mathbf{i} = \mathbf{j}\times\mathbf{k}$. These cross products define the so-called *right-hand rule* for the products of the quaternion vectors, which are the same as the vectors used in Vector Calculus. The general product of two quaternions, p and q, can be shown to be

$$pq = p_w q_w - \mathbf{p}\cdot\mathbf{q} + p_w\mathbf{q} + q_w\mathbf{p} + \mathbf{p}\times\mathbf{q}, \tag{86}$$

where

$$\mathbf{p}\cdot\mathbf{q} = -(\mathbf{pq} + \mathbf{qp})/2 \tag{87}$$

and

$$\mathbf{p}\times\mathbf{q} = (\mathbf{pq} - \mathbf{qp})/2. \tag{88}$$

From this general product, the product of two vectors is

$$\mathbf{pq} = -\mathbf{p}\cdot\mathbf{q} + \mathbf{p}\times\mathbf{q}. \tag{89}$$

The conjugate of q is

$$\mathrm{K}(q) = q^\dagger = q_w - \mathbf{q}, \tag{90}$$

where $\mathrm{K}(q_1 q_2 \ldots) = \ldots q_2^\dagger q_1^\dagger$ in the reverse order. We take the scalar part of q as

$$\mathrm{S}(q) = q_w = (q + q^\dagger)/2 \tag{91}$$

and the vector part as

$$\mathrm{V}(q) = \mathbf{q} = (q - q^\dagger)/2, \tag{92}$$

The tensor (or magnitude) of q is

$$\mathrm{T}(q) = |q| = \sqrt{qq^\dagger} = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2}. \tag{93}$$

A unit quaterion $\hat{q}$, called a versor, is

$$\hat{q} = \mathrm{U}(q) = q/\mathrm{T}(q) = \cos(\theta/2) + \sin(\theta/2)\hat{\mathbf{q}} = \exp(\theta\hat{\mathbf{q}}/2) = R \tag{94}$$

and represents a rotation operator for rotation by an angle θ around axis $\hat{\mathbf{q}}$. Any quaternion q can be written as the product of its unit $\mathrm{U}(q)$ and tensor $\mathrm{T}(q)$ as

$$q = \mathrm{T}(q)\mathrm{U}(q). \tag{95}$$

The inverse of q is

$$q^{-1} = q^\dagger / qq^\dagger = q^\dagger / \mathrm{T}(q)^2 = \mathrm{U}(q)^\dagger \mathrm{T}(q)^{-1}, \tag{96}$$

where $\mathrm{U}(q)^\dagger = \mathrm{U}(q)^{-1}$.

Quaternion rotation of a vector $\mathbf{p}$ is

$$\mathbf{p}' = R\mathbf{p}R^\dagger, \tag{97}$$

rotating $\mathbf{p}$ around the axis $\hat{\mathbf{q}}$ by angle θ centered on the origin. The versor product $\mathbf{p}' = R\mathbf{p}R^\dagger$ only rotates the component of $\mathbf{p}$ perpendicular to the axis $\hat{\mathbf{q}}$ in a certain plane, leaving the part of $\mathbf{p}$ parallel to $\mathbf{q}$ unchanged. Note that, the exponential form allows to write any unit vector $\hat{\mathbf{v}}$ as $\hat{\mathbf{v}} = \exp(\pi\hat{\mathbf{v}}/2)$, showing that it represents rotation by π around $\hat{\mathbf{v}}$ as a versor, or a rotation by $\pi/2$ when multiplied with any vector in the plane perpendicular to $\hat{\mathbf{v}}$. There is an exponential form for every quaternion, just as there is for any complex number. The square of any unit vector is -1, $\hat{\mathbf{v}}^2 = -1$. This is all very similar to complex numbers, with quaternions extending the basic algebraic forms of complex numbers into 3D space. Quaternions of the form $a + b\hat{\mathbf{v}}$ are basically like a scalar plane of complex numbers with the usual complex number products that multiply magnitudes and add angles, but $a + b\hat{\mathbf{v}}$ also rotates vectors in a geometrical plane perpendicular to $\hat{\mathbf{v}}$.

A simpler way to look at quaternions is to say that a quaternion q is nothing more than the ratio (or product) of two pure quaternion vectors $\mathbf{b} = b_x\mathbf{i} + b_y\mathbf{j} + b_z\mathbf{k}$ and $\mathbf{a}$, writing $q = \mathbf{b}/\mathbf{a} = \mathbf{b}\mathbf{a}^{-1} = \mathbf{b}\mathbf{a}/\mathbf{a}^2 = -\mathbf{b}\mathbf{a}/\|\mathbf{a}\|^2$, recalling that $\mathbf{a}^2 = \|\mathbf{a}\|^2\hat{\mathbf{a}}^2 = -\|\mathbf{a}\|^2$. A quaternion q is a transition operator, transforming vector $\mathbf{a}$ into vector $\mathbf{b}$ as $\mathbf{b} = q\mathbf{a}$. We can write q as $q = -(\|\mathbf{b}\|/\|\mathbf{a}\|)\hat{\mathbf{b}}\hat{\mathbf{a}} = -|q|(-\hat{\mathbf{b}}\cdot\hat{\mathbf{a}} + \hat{\mathbf{b}}\times\hat{\mathbf{a}})$, then $q = |q|(\cos(\theta) + \sin(\theta)\hat{\mathbf{n}})$, where $\hat{\mathbf{n}} = \hat{\mathbf{a}}\times\hat{\mathbf{b}}/\|\hat{\mathbf{a}}\times\hat{\mathbf{b}}\|$ is the rotation axis with rotation orientation of the angle θ to rotate from $\mathbf{a}$ toward $\mathbf{b}$ in the $\mathbf{ab}$-plane, by the right-hand rule of the cross product. All we really care about most of the time is using quaternions as rotation operators, or versors. Then we have $\hat{q} = \cos(\theta) + \sin(\theta)\hat{\mathbf{n}}$, and if we use this in a versor sandwich product, then it rotates by 2θ around axis $\hat{\mathbf{n}}$. We simply use the form and replace the angle with half of it as $\cos(\theta/2) + \sin(\theta/2)\hat{\mathbf{n}} = \exp(\theta\hat{\mathbf{n}}/2)$.

Vector Calculus, using the pure quaternion vectors, is well known at any engineering college and in all of the standard Calculus textbooks. But still, it is more confusing than it needs to be. It is actually less confusing to use Geometric Algebra to do all of the same things as quaternions and Vector Calculus. For now, we are only concerned with quaternions, so let's now discuss how they are represented in geometric algebra.

5.3.2 Quaternion Geometric Algebra

In $\mathcal{G}_{3,0,1}$, the correspondence from quaternion algebra (QA) to quaternion geometric algebra (QGA) is $\{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\} \hat{=} \{1, \mathbf{e}_1^*, \mathbf{e}_2^*, \mathbf{e}_3^*\}$. QGA is the algebra of the even-grades subalgebra $\mathcal{G}_3^+$. That is almost all you need to know to use QGA. Any vector $\mathbf{v} \in \mathcal{G}_3^1$ is transformed into its quaternion vector form by the dualization

$$\mathbf{v}^* = \mathbf{v}/\mathbf{I}_3 \in \mathcal{G}_3^2 \tag{98}$$

into QGA $\mathcal{G}_3^+$. For example, in QA we have that $\mathbf{k}$ fundamentally represents $\mathbf{j}/\mathbf{i}$, and in QGA we have the corresponding ratio $\mathbf{e}_2^*/\mathbf{e}_1^* = \mathbf{e}_2/\mathbf{e}_1 = \mathbf{e}_2\mathbf{e}_1$. Then, this is simpler to express as the dual $\mathbf{k} \hat{=} \mathbf{e}_3^* = \mathbf{e}_3/\mathbf{I}_3 = \mathbf{e}_3\mathbf{e}_3\mathbf{e}_2\mathbf{e}_1 = \mathbf{e}_2\mathbf{e}_1$. It is similar for the other corresponding elements. The QA basis vectors correspond to the basis 2-blades in QGA $\mathcal{G}_3^+$. The QA product rule is a trick that hides the true algebra, that quaternions are an algebra of bivectors in QGA. We could just use the Euclidean vectors $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ and use the quaternion "vectors" $\{\mathbf{e}_1^*, \mathbf{e}_2^*, \mathbf{e}_3^*\}$ as the bivectors generating rotations around the axes $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. But, we want to emulate the quaternions in just the even-grades subalgebra $\mathcal{G}_{3,0,1}^+$, so we cannot use any odd grade elements such as $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ in QGA $\mathcal{G}_3^+$. So, we now proceed to implement all of the quaternion operations and products in QGA $\mathcal{G}_3^+$.

In QGA, using the QGA basis $\{1, \mathbf{e}_1^*, \mathbf{e}_2^*, \mathbf{e}_3^*\}$ in geometric algebra, we write a quaternion q as

$$q = q_w + q_x\mathbf{e}_1^* + q_y\mathbf{e}_2^* + q_z\mathbf{e}_3^* = q_w + \mathbf{q}^*. \tag{99}$$

A vector $\mathbf{v} = v_x\mathbf{e}_1 + v_y\mathbf{e}_2 + v_z\mathbf{e}_3$ is mapped to a quaternion vector as the bivector $\mathbf{v}^* = \mathbf{v}/\mathbf{I}_3$ of the orthogonal plane, so we use the notation $\mathbf{v}^*$ as a shorthand even though the vectors $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ are not elements of QGA. We implement the quaternion conjugate as

$$\mathrm{K}(q) = q^\dagger = q^\sim, \tag{100}$$

where $q^\sim$ is the geometric algebra *reverse* operator. The notation $q^\dagger$ is also a geometric algebra notation for reverse [19], so there is no notational conflict. The vector calculus dot product $\cdot$ (bold dot) is implemented by an inner product $\cdot$ (dot), or symmetric product of bivectors, as

$$\mathbf{p}^* \cdot \mathbf{q}^* = -\mathbf{p}^* \cdot \mathbf{q}^* = -(\mathbf{p}^*\mathbf{q}^* + \mathbf{q}^*\mathbf{p}^*)/2, \tag{101}$$

where the negative sign is necessary since unit bivectors square to -1 just as do all unit quaternion vectors. The wedge (outer) product of two bivectors is part of the symmetric product $(\mathbf{pq} + \mathbf{qp})/2$ but is 0 in this case. The vector calculus cross product $\times$ (bold cross) is implemented as

$$\mathbf{p}^* \times \mathbf{q}^* = \mathbf{p}^* \times \mathbf{q}^* = (\mathbf{p}^*\mathbf{q}^* - \mathbf{q}^*\mathbf{p}^*)/2 \tag{102}$$

(the commutator product $\times$ (cross)), so this is the same as in actual QA except we are not following the quaternion product rule, we are just letting the geometric algebra compute the result. The scalar part

$$\mathrm{S}(q) = (q + q^\dagger)/2, \tag{103}$$

vector part

$$\mathrm{V}(q) = (q - q^\dagger)/2, \tag{104}$$

and tensor or magnitude

$$\mathrm{T}(q) = |q| = \sqrt{qq^\dagger} \tag{105}$$

are implemented the same as in QA by using the conjugate. The unit $\hat{q}$ of q is

$$\hat{q} = \mathrm{U}(q) = q/\mathrm{T}(q). \tag{106}$$

We can check that the QGA quaternion vector units still obey the quaternion product rule:

$$\mathbf{e}_1^*\mathbf{e}_2^*\mathbf{e}_3^* = \mathbf{e}_1^{*2} = \mathbf{e}_2^{*2} = \mathbf{e}_3^{*2} = -1. \tag{107}$$

It can be shown that this rule holds. That is about it, since everything that can be done in QA can be done in QGA by using the basis and operations as they have been implemented here.

At this point, we should not be confused in understanding that in QA (quaternion algebra in original form and notation) we have vector $\hat{\mathbf{n}}$ on the basis $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$, in PGA we have $\hat{\mathbf{n}}$ on the basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, and in QGA we have the PGA vector $\hat{\mathbf{n}}$ corresponding to (or converted to, or dualized to, or sometimes called "quaternionized" to) bivector $\hat{\mathbf{n}}^* = \hat{\mathbf{n}}/\mathbf{I}_3$ on the basis $\{\mathbf{e}_1^*, \mathbf{e}_2^*, \mathbf{e}_3^*\}$ that faithfully emulates QA, but that each represents the same vector. We will refer to a dualized vector such as $\mathbf{v}/\mathbf{I}_3 = \mathbf{v}^*$ as just being the quaternion vector $\mathbf{v}^*$.

5.4 Dual Quaternions in PGA

In the prior sections, we reviewed dual number algebra (DNA) and quaternion algebra (QA) in their original forms, and then how they are represented or emulated in the geometric algebra $\mathcal{G}^+_{3,0,1}$ as subalgebras that we have called dual number geometric algebra (DNGA) and quaternion geometric algebra (QGA). In this section, we discuss dual quaternion algebra (DQA) and its representation in $\mathcal{G}^+_{3,0,1}$ as dual quaternion geometric algebra (DQGA).

5.4.1 Dual Quaternion Algebra

In dual quaternion algebra (DQA), dual quaternions are very similar to quaternions, but instead of using only the real numbers, we extend the real numbers to dual numbers. The dual quaternion basis elements are $\{1, \varepsilon, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$. A dual quaternion d has the general form $d = q_1 + q_2\varepsilon$, where $q_1 = q_{1w} + \mathbf{q}_1$ and $q_2 = q_{2w} + \mathbf{q}_2$ are quaternions and ε is the nilpotent scalar, $\varepsilon^2 = 0$.

5.4.2 Dual Quaternion Geometric Algebra

In dual quaternion geometric algebra (DQGA), the correspondence of elements from DQA to DQGA in the even-grades subalgebra $\mathcal{G}^+_{3,0,1}$ of PGA is $\{1, \varepsilon, \mathbf{i}, \mathbf{j}, \mathbf{k}\} \mathrel{\hat{=}} \{1, \mathbf{I}_4, \mathbf{e}_1^*, \mathbf{e}_2^*, \mathbf{e}_3^*\}$. In DQGA, we write $q_1 = q_{1w} + \mathbf{q}_1^*$, $q_2 = q_{2w} + \mathbf{q}_2^*$, and the dual quaternion is

$$d = q_1 + q_2\mathbf{I}_4 = q_{1w} + \mathbf{q}_1^* + q_{2w}\mathbf{I}_4 + \mathbf{q}_2^*\mathbf{I}_4. \tag{108}$$

The DQGA dual quaternion *complex conjugate* $\bar{d} = q_1 - q_2\mathbf{I}_4$ is implemented in PGA as

$$\bar{d} = \mathbf{I}_3 d \mathbf{I}_3^{-1} \tag{109}$$

(the same as for DNGA), where $(d_1 d_2 \ldots)^- = \bar{d}_1 \bar{d}_2 \ldots$ in the same order.

The DQGA dual quaternion *quaternion conjugate* $\mathrm{K}(d) = d^\dagger = q_1^\dagger + q_2^\dagger \mathbf{I}_4 = (q_{1w} + q_{2w}\mathbf{I}_4) - (\mathbf{q}_1^* + \mathbf{q}_2^*\mathbf{I}_4)$ is implemented by using the geometric algebra *reverse* operation as

$$\mathrm{K}(d) = d^\dagger = d^\sim \tag{110}$$

(the same as for QGA), where $(d_1 d_2 \ldots)^\dagger = \cdots d_2^\dagger d_1^\dagger$ in the reverse order.

We can compose $\bar{d}$ and $d^\dagger$ as the DQGA dual quaternion "*dual conjugate*"

$$\bar{d}^\dagger = q_1^\dagger - q_2^\dagger \mathbf{I}_4 = q_{1w} - \mathbf{q}_1^* - q_{2w}\mathbf{I}_4 + \mathbf{q}_2^*\mathbf{I}_4 \tag{111}$$

Using the conjugates, we can now take the *real part*

$$\Re(d) = (d + \bar{d})/2 \tag{112}$$

and the *imaginary part*

$$\Im(d) = (d - \bar{d})/2, \tag{113}$$

which works the same as for dual numbers. The *scalar part* is

$$\mathrm{S}(d) = (d + d^\dagger)/2, \tag{114}$$

which is a dual number, since the dual numbers are the scalars. The *vector part* is

$$\mathrm{V}(d) = (d - d^\dagger)/2, \tag{115}$$

which includes the real vector and imaginary vector parts. We can compose $\Re$ or $\Im$ with S or V to get any one of the four parts from $d = q_{1w} + \mathbf{q}_1^* + q_{2w}\mathbf{I}_4 + \mathbf{q}_2^*\mathbf{I}_4$.

The tensor or magnitude part $\mathrm{T}(d)$ is more complicated for dual quaternions. For the tensor, we could try $\sqrt{d\bar{d}^\dagger}$, but $d\bar{d}^\dagger$ produces a dual quaternion of the form $s + \mathbf{v}^*\mathbf{I}_4$, which is not a real scalar or dual number scalar in the general case, so this is not the tensor. We could try $d\bar{d}$, which again produces a dual quaternion of the form $s + \mathbf{v}^*\mathbf{I}_4$, and so is not the tensor. Finally we try $dd^\dagger$, which has the general form $a + b\mathbf{I}_4$ of a dual number scalar, but if $d = z$ (just a dual number), then $dd^\dagger = zz$ is not the (maybe) expected real squared magnitude $|z|^2$ of z. However, we accept $dd^\dagger = |q_1|^2 + 2(q_{1w}q_{2w} + \mathbf{q}_1^* \cdot \mathbf{q}_2^*)\mathbf{I}_4$ and take the *dual number-valued* tensor as

$$\begin{aligned} \mathrm{T}(d) &= \sqrt{d d^\dagger} = \sqrt{|q_1|^2(1 + 2((q_{1w}q_{2w} + \mathbf{q}_1^* \cdot \mathbf{q}_2^*)/|q_1|^2)\mathbf{I}_4)} && (116) \\ &= |q_1|(1 + ((q_{1w}q_{2w} + \mathbf{q}_1^* \cdot \mathbf{q}_2^*)/|q_1|^2)\mathbf{I}_4), && (117) \end{aligned}$$

for $|q_1| \neq 0$. The real tensor is $\Re(\mathrm{T}(d)) = |q_1| = \mathrm{T}(q_1)$. If $d = z$ (just a dual number), then $\mathrm{T}(d) = z \neq |z|$, so we cannot generally use the notation $\mathrm{T}(d) = |d|$ in dual quaternions as we can in quaternions, but we can use a modified notation such as $\mathrm{T}(d) = |d|_{\mathcal{D}}$ to indicate dual number-valued. The inverse tensor is

$$\mathrm{T}(d)^{-1} = |q_1|^{-1}(1 - ((q_{1w}q_{2w} + \mathbf{q}_1^* \cdot \mathbf{q}_2^*)/|q_1|^2)\mathbf{I}_4). \tag{118}$$

Using $\mathrm{T}(d)^{-1}$, a unit dual quaternion is

$$\hat{d} = \mathrm{U}(d) = d\mathrm{T}(d)^{-1}, \tag{119}$$

which we can often think of as *normalizing* the dual quaternion entity d. For example, we will see that the unnormalized DQGA plane entity is $\pi = -(\mathbf{n}^* + (\mathbf{p}^* \cdot \mathbf{n}^*)\mathbf{I}_4)$, which is normalized as the unit plane $\hat{\pi} = \pi|\pi|_{\mathcal{D}}^{-1} = -(\hat{\mathbf{n}}^* + (\mathbf{p}^* \cdot \hat{\mathbf{n}}^*)\mathbf{I}_4)$ in our standard unit plane form (in this case, $|\pi|_{\mathcal{D}}^{-1} = 1/\Re(\mathrm{T}(\pi))$).

After learning about the DQGA forms of a point $p_{\mathbf{t}}$ (point embedding of vector $\mathbf{t}$) and a plane $\pi_{\mathbf{p},\hat{\mathbf{n}}}$ (plane normal to $\hat{\mathbf{n}}^*$ at distance $d = \mathbf{p}^* \cdot \hat{\mathbf{n}}^*$ from origin), we will see that by using the dual conjugate $\bar{d}^\dagger$, the *point part* $\mathrm{P}(d)$ of a dual quaternion d is

$$\mathrm{P}(d) = (d + \bar{d}^\dagger)/2 = q_{w1} p_{\mathbf{q}_2/q_{w1}} = q_{1w} + \mathbf{q}_2^* \mathbf{I}_4 \tag{120}$$

and the *plane part* $\Pi(d)$ is

$$\Pi(d) = (d - \bar{d}^\dagger)/2 = -\|\mathbf{q}_1^*\| \pi_{d,\hat{\mathbf{n}}} = -\|\mathbf{q}_1^*\| \pi_{q_{2w}/\|\mathbf{q}_1\|,\hat{\mathbf{q}}_1} = \mathbf{q}_1^* + q_{2w} \mathbf{I}_4. \tag{121}$$

Therefore, a dual quaternion q can represent a point and a plane together as a single entity. When $\mathbf{q}_1^* \cdot \mathbf{q}_2^* = 0$ so that $\mathbf{q}_1^*$ and $\mathbf{q}_2^*$ are orthogonal, we will see that the vector part $\mathrm{V}(q) = \mathbf{q}_1^* + \mathbf{q}_2^* \mathbf{I}_4$ represents the *line part* $l_{(\mathbf{q}_1 : -\mathbf{q}_2)}$ with Plücker coordinates $(\mathbf{d}^* : \mathbf{m}^*) = (\mathbf{q}_1^* : -\mathbf{q}_2^*)$. So, we define the *line part* $\mathrm{L}(d)$ as

$$\mathrm{L}(d) = \mathrm{V}(q) = \mathbf{q}_1^* + \mathbf{q}_2^* \mathbf{I}_4, \tag{122}$$

an alias for the vector part. We will use the point, plane, and line part operators to extract these entities from the products for intersections.

In dual number geometric algebra, we defined the operator $\mathrm{Y}(z) = -J_e(\Im(z))$, which depends on the PGA entity dualization operation J_e, to take the real scalar y from the imaginary part of $z = x + y\varepsilon$ as $y = \mathrm{Y}(z)$. In dual quaternions, we can use Y to take the real quaternion q_2 from the imaginary part of d as $q_2 = \mathrm{Y}(d)$, so it works the same if the dual quaternion is just a dual number $d = z$ or any imaginary part.

In the following sections, we show that a dual quaternion can represent a point p, line l, or plane π, and that there are operations for rotation, reflection, translation, intersection, and projection. Of course, dual quaternions, which includes the quaternions and the pure quaternion versors (the vectors $\{\mathbf{e}_1^*, \mathbf{e}_2^*, \mathbf{e}_3^*\}$), can also be used for vector calculus.

5.5 DQGA Geometric Entities

In this section, we derive the DQGA point entity $p = p_{\mathbf{t}}$ (embedding quaternion vector $\mathbf{t}^*$), DQGA plane entity $\pi = \pi_{\mathbf{p},\hat{\mathbf{n}}} = \pi_{d=\mathbf{p}\cdot\hat{\mathbf{n}},\hat{\mathbf{n}}}$ (through quaternion vector point $\mathbf{p}^*$ with normal $\hat{\mathbf{n}}^*$), and the DQGA line entity $l = l_{\mathbf{p},\hat{\mathbf{d}}}$ (through quaternion vector point $\mathbf{p}^*$ in direction $\hat{\mathbf{d}}^*$). Each entity $A \in \{p, \pi, l\}$ represents a *null space set* of points, $\mathcal{N}_A = \{p_{\mathbf{t}} : \mathrm{N}_A(p_{\mathbf{t}} A) = 0\}$. The product $p_{\mathbf{t}} A$, of the symbolic DQGA point $p_{\mathbf{t}}$ (embedding symbolic vector $\mathbf{t} = x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3$) and an entity A, is the test of point $p_{\mathbf{t}}$ for coincidence with the surface represented by A. Points in the null space set $\mathcal{N}_A$ are on the surface represented by A. The *null space entity part* $\mathrm{N}_A(p_{\mathbf{t}} A)$ for the product $p_{\mathbf{t}} A$ is the part that represents the null space of entity A. For a point $A = p_{\mathbf{a}}$, $\mathrm{N}_p(p_{\mathbf{t}} p_{\mathbf{a}}) = \mathrm{V}(\Im(p_{\mathbf{t}} p_{\mathbf{a}}))$ and $\mathcal{N}_{p_{\mathbf{a}}} = \{p_{-\mathbf{a}}\}$. For a plane $A = \pi$, $\mathrm{N}_\pi(p_{\mathbf{t}} \pi) = \mathrm{S}(\Im(p_{\mathbf{t}} \pi))$. For a line $A = l$, $\mathrm{N}_l(p_{\mathbf{t}} l) = \mathrm{V}(\Im(p_{\mathbf{t}} l))$. In CPNS PGA, the commutator product always gives the null space entity part, but in DQGA we do not have a single product that always gives the null space entity. Therefore, in DQGA, we have to take the dual quaternion product, and then take part N_A and check it for 0. For each kind of entity A, there are also entity-specific formulas for their reflections in planes and translations. Rotation is performed the same on all entities using the usual rotor R in a versor sandwich product of the entity.

5.5.1 DQGA Point Entity

In DQA, a quaternion vector $\mathbf{t}$ is embedded as a homogeneous point $p_{\mathbf{t}} = 1 + \mathbf{t}\varepsilon$.

In DQGA, a quaternion vector $\mathbf{t}^* = (x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3)/\mathbf{I}_3$ is embedded as the homogeneous DQGA point

$$p_{\mathbf{t}} = 1 + \mathbf{t}^*\mathbf{I}_4 = \exp(\mathbf{t}^*\mathbf{I}_4) = 1 + \mathbf{e}_0\mathbf{t} = 1 + x\mathbf{e}_0\mathbf{e}_1 + y\mathbf{e}_0\mathbf{e}_2 + z\mathbf{e}_0\mathbf{e}_3. \tag{123}$$

In $p_{\mathbf{t}}$, we usually do not make direct use of the basis 2-blades, or vectors $\mathbf{e}_0$ and $\mathbf{t}$, since the vectors $\{\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ are not elements of DQGA in $\mathcal{G}^+_{3,0,1}$, but these 2-blade products are what results from the DQGA product $\mathbf{t}^*\mathbf{I}_4$, representing an imaginary vector part.

When we test one point $p_{\mathbf{a}}$ with another $p_{\mathbf{b}}$ for equality, we could just do an equality test by testing scalar components for equality. We *cannot* multiply $p_{\mathbf{a}} \times p_{\mathbf{b}}$ as a null space equality test, as we do for points in CPNS PGA, since $p_{\mathbf{a}} \times p_{\mathbf{b}} = 0$ for both $p_{\mathbf{a}} = p_{\mathbf{b}}$ and $p_{\mathbf{a}} = p_{-\mathbf{b}}$. As a surface entity, a point $p_{\mathbf{b}}$ represents the null space, $\{p_{\mathbf{t}} : \mathrm{N}_p(p_{\mathbf{t}}p_{\mathbf{b}}) = \mathrm{V}(\Im(p_{\mathbf{t}}p_{\mathbf{b}})) = 0\} = \{\bar{p}_{\mathbf{b}} = p_{-\mathbf{b}}\}$, containing just the conjugate point $p_{\mathbf{t}} = \bar{p}_{\mathbf{b}} = p_{-\mathbf{b}}$. We call $\mathrm{N}_p(p_{\mathbf{t}}p_{\mathbf{b}}) = \mathrm{V}(\Im(p_{\mathbf{t}}p_{\mathbf{b}}))$ the null space entity part of the test. We can extract the real vector of the null space entity as $\mathrm{Y}(\mathrm{V}(\Im(p_{\mathbf{t}}p_{\mathbf{b}})))$.

We have already discussed the DQGA point embedding in Section 3.2.3 on the CPNS PGA 3-blade point $\boldsymbol{p}_{\mathbf{t}}$, where we found the identity $\boldsymbol{p}_{\mathbf{t}} = p_{\mathbf{t}}\mathbf{I}_3$, or

$$p_{\mathbf{t}} = \boldsymbol{p}_{\mathbf{t}}\mathbf{I}_3^{-1} = -\boldsymbol{p}_{\mathbf{t}}\mathbf{I}_3. \tag{124}$$

We also talked about the DQGA point $p_{\mathbf{t}}$ in Section 3.3.1 on the CPNS PGA 2-versor translator T, where we found that T basically is a DQGA point,

$$T = p_{\mathbf{d}/2}. \tag{125}$$

The connection between the CPNS PGA and DQGA allows for identities to convert many expressions in CPNS PGA into DQGA, which corresponds to regular DQA. Therefore, we can derive DQA representations of points, lines, and planes with many operations on them by using PGA.

In CPNS PGA, we can test if point $\boldsymbol{p}_{\mathbf{t}}$ is $\boldsymbol{p}_{\mathbf{p}}$ as $\boldsymbol{p}_{\mathbf{t}} \times \boldsymbol{p}_{\mathbf{p}} = (\boldsymbol{p}_{\mathbf{t}}\boldsymbol{p}_{\mathbf{p}} - \boldsymbol{p}_{\mathbf{p}}\boldsymbol{p}_{\mathbf{t}})/2$. We can convert this test into DQGA by using the identity $\boldsymbol{p} = p\mathbf{I}_3$ and obtain $(p_{\mathbf{t}}\mathbf{I}_3 p_{\mathbf{p}}\mathbf{I}_3 - p_{\mathbf{p}}\mathbf{I}_3 p_{\mathbf{t}}\mathbf{I}_3)/2$. So, the exact null space entity part is $(-p_{\mathbf{t}}\bar{p}_{\mathbf{p}} + p_{\mathbf{p}}\bar{p}_{\mathbf{t}})/2$. Then, $p_{\mathbf{t}}$ is the point $p_{\mathbf{p}}$ if and only if

$$(-p_{\mathbf{t}}\bar{p}_{\mathbf{p}} + p_{\mathbf{p}}\bar{p}_{\mathbf{t}})/2 = 0. \tag{126}$$

We can save computation by using $\mathrm{N}_p(p_{\mathbf{t}}\bar{p}_{\mathbf{p}}) = 0$.

We can also switch $\boldsymbol{p}_{\mathbf{t}} \times \boldsymbol{p}_{\mathbf{p}}$ to geometric product and write $p_{\mathbf{t}}\mathbf{I}_3 p_{\mathbf{p}}\mathbf{I}_3 = -p_{\mathbf{t}}\bar{p}_{\mathbf{p}}$. Then, it is clear that $\mathrm{N}_p(-p_{\mathbf{t}}\bar{p}_{\mathbf{p}}) = \mathrm{V}(\Im(-p_{\mathbf{t}}\bar{p}_{\mathbf{p}})) = 0$ if and only if $p_{\mathbf{t}}$ is $p_{\mathbf{p}}$. This saves computation. It does not matter if $p_{\mathbf{t}}$ or $p_{\mathbf{p}}$ have been scaled, even by a dual number scalar, since they are homogeneous points. We could write the product and null space set condition as $\mathrm{V}(\Im(-\mathrm{U}(p_{\mathbf{t}})\mathrm{T}(p_{\mathbf{t}})\mathrm{U}(\bar{p}_{\mathbf{p}})\mathrm{T}(\bar{p}_{\mathbf{p}}))) = 0$, or $\mathrm{V}(\Im(-\mathrm{U}(p_{\mathbf{t}})\mathrm{U}(\bar{p}_{\mathbf{p}})\mathrm{T}(p_{\mathbf{t}})\mathrm{T}(\bar{p}_{\mathbf{p}}))) = 0$. Only the units are significant to the test, and this is similar for all of the homogeneous DQGA entities.

5.5.2 DQGA Plane Entity

In the CPNS PGA, we have the CPNS PGA 1-blade plane $\boldsymbol{\pi} = \hat{\mathbf{n}} + d\mathbf{e}_0$ and the CPNS PGA 3-blade point $\boldsymbol{p}_\mathbf{t} = (1 + \mathbf{t}^*\mathbf{I}_4)\mathbf{I}_3 = \mathbf{I}_3 - \mathbf{e}_0\mathbf{t}^* = p_\mathbf{t}\mathbf{I}_3$. We test a point $\boldsymbol{p}_\mathbf{t}$ for intersection with a plane $\boldsymbol{\pi}$ as $\boldsymbol{p}_\mathbf{t} \times \boldsymbol{\pi} = \boldsymbol{p}_\mathbf{t} \wedge \boldsymbol{\pi}$. Now, we switch to geometric product as $\boldsymbol{p}_\mathbf{t}\boldsymbol{\pi}$ and use the identity $\boldsymbol{p}_\mathbf{t} = p_\mathbf{t}\mathbf{I}_3$ to obtain $\boldsymbol{p}_\mathbf{t}\boldsymbol{\pi} = p_\mathbf{t}\mathbf{I}_3\boldsymbol{\pi}$. We can take $\mathbf{I}_3\boldsymbol{\pi}$ as the DQGA plane entity. Therefore, the DQGA plane π is

$$\pi = \pi_{d,\hat{\mathbf{n}}} = \pi_{\mathbf{p},\hat{\mathbf{n}}} = \mathbf{I}_3\boldsymbol{\pi} = \mathbf{I}_3(\hat{\mathbf{n}} + d\mathbf{e}_0) = -(\hat{\mathbf{n}}^* + d\mathbf{I}_4) = -(\hat{\mathbf{n}}^* + (\mathbf{p}^* \cdot \hat{\mathbf{n}}^*)\mathbf{I}_4), \tag{127}$$

for a plane through point $\mathbf{p}^*$ and normal to $\hat{\mathbf{n}}^*$, where $d = \mathbf{p}^* \cdot \hat{\mathbf{n}}^* = -\mathbf{p}^* \cdot \hat{\mathbf{n}}^*$ is the distance of the plane from the origin. We have introduced a subscript notation to indicate the plane normal $\hat{\mathbf{n}}$ and distance d from origin $(d, \hat{\mathbf{n}})$ or plane normal and surface point $(\mathbf{p},\hat{\mathbf{n}})$. The minus sign is important for maintaining the scale or orientation.

Like the CPNS PGA plane $\boldsymbol{\pi}$, the DQGA plane π is a plane in general position and it can be rotated by the usual rotor R. The translation of π is slightly more involved. The dual quaternion π is the sum of a quaternion vector and imaginary scalar, so it is fully within DQA.

The test product $\boldsymbol{p}_\mathbf{t} \times \boldsymbol{\pi}$ is grade 4, a pseudoscalar, which corresponds to the DQGA element $\mathbf{I}_4 \mathrel{\hat{=}} \varepsilon$. Therefore, the *null space entity part* $\mathrm{N}_\pi(p_\mathbf{t}\pi)$ of the point-plane test $p_\mathbf{t}\mathbf{I}_3\boldsymbol{\pi} = p_\mathbf{t}\pi$ is $\mathrm{N}_\pi(p_\mathbf{t}\pi) = \mathrm{S}(\Im(p_\mathbf{t}\pi))$. The point $p_\mathbf{t}$ is a point of the plane π ($p_\mathbf{t} \in \pi$) if and only if $\mathrm{N}_\pi(p_\mathbf{t}\pi) = 0$. In this case, we can extract the scalar

$$\mathrm{Y}(\mathrm{N}_\pi(p_\mathbf{t}\pi)) = x\hat{n}_x + y\hat{n}_y + z\hat{n}_z - d \tag{128}$$

representing the implicit plane equation.

If we do not want to use the null space entity part $\mathrm{N}_\pi(p_\mathbf{t}\pi)$ for the null space set condition $\mathrm{N}_\pi(p_\mathbf{t}\pi) = 0$, then we can reformulate the test product $\boldsymbol{p}_t \times \boldsymbol{\pi}$ into full geometric products as $(\boldsymbol{p}_\mathbf{t}\boldsymbol{\pi} - \boldsymbol{\pi}\boldsymbol{p}_\mathbf{t})/2$. Then, we use identities to make substitutions and obtain $(p_\mathbf{t}\mathbf{I}_3\mathbf{I}_3^{-1}\pi - \mathbf{I}_3^{-1}\pi p_\mathbf{t}\mathbf{I}_3)/2$. So, the test product is now exactly the null space entity part $(p_\mathbf{t}\pi - \overline{\pi}\overline{p}_\mathbf{t})/2$. Then, $p_\mathbf{t}$ is a point on the plane π if and only if

$$(p_\mathbf{t}\pi - \overline{\pi}\overline{p}_\mathbf{t})/2 = 0. \tag{129}$$

We may save computation by using $\mathrm{N}_\pi(p_\mathbf{t}\pi) = 0$.

5.5.3 DQGA Line Entity

In CPNS PGA, we have the CPNS PGA 2-blade line $\boldsymbol{l} = \hat{\mathbf{d}}^* - (\mathbf{p} \cdot \hat{\mathbf{d}}^*)\mathbf{e}_0$ and the CPNS PGA 3-blade point $\boldsymbol{p}_\mathbf{t} = (1 + \mathbf{t}^*\mathbf{I}_4)\mathbf{I}_3 = \mathbf{I}_3 - \mathbf{e}_0\mathbf{t}^* = p_\mathbf{t}\mathbf{I}_3$. We test a point $\boldsymbol{p}_\mathbf{t}$ for intersection with a line $\boldsymbol{l}$ as $\boldsymbol{p}_\mathbf{t} \times \boldsymbol{l}$, which is a grade 3 null space entity. We cannot have any grade 3 elements in the DQGA, so we have to do something a little special to get a line entity in the even grades. Now, we switch to geometric product as $\boldsymbol{p}_\mathbf{t}\boldsymbol{l}$ and use the identity $\boldsymbol{p}_\mathbf{t} = p_\mathbf{t}\mathbf{I}_3$ to obtain $\boldsymbol{p}_\mathbf{t}\boldsymbol{l} = p_\mathbf{t}\mathbf{I}_3\boldsymbol{l}$. We cannot just take $\mathbf{I}_3\boldsymbol{l}$ as the DQGA line entity because it is in the odd grades outside of DQGA. We can use the identity $\mathbf{I}_3^{-1}\mathbf{I}_3 = 1$ to write $\mathbf{I}_3\boldsymbol{l}\mathbf{I}_3^{-1}\mathbf{I}_3$, which does not change it yet. Now, we see the test as $p_\mathbf{t}\mathbf{I}_3\boldsymbol{l}\mathbf{I}_3^{-1}\mathbf{I}_3$ and the important part of this test is the part in parentheses $(p_\mathbf{t}\mathbf{I}_3\boldsymbol{l}\mathbf{I}_3^{-1})\mathbf{I}_3$. The RHS $\mathbf{I}_3$ is merely a constant that just alters grades and that we can abridge from the test. By abridgment, we take the DQGA line l to be $l = \mathbf{I}_3\boldsymbol{l}\mathbf{I}_3^{-1} = \bar{\boldsymbol{l}} = \hat{\mathbf{d}}^* + (\mathbf{p} \cdot \hat{\mathbf{d}}^*)\mathbf{e}_0$. We still cannot have $\mathbf{p}$, so we use the identity $\mathbf{a}^* \times \mathbf{b}^* = ((\mathbf{a} \wedge \mathbf{b})/\mathbf{I}_3)/\mathbf{I}_3 = (\mathbf{a} \cdot \mathbf{b}^*)/\mathbf{I}_3$, and $(\mathbf{a}^* \times \mathbf{b}^*)\mathbf{I}_3 = \mathbf{a} \cdot \mathbf{b}^*$. Then, $l = \hat{\mathbf{d}}^* + (\mathbf{p}^* \times \hat{\mathbf{d}}^*)\mathbf{I}_3\mathbf{e}_0$. Therefore, we have the following:

The DQGA line l in the direction $\hat{\mathbf{d}}^*$ and through the point $\mathbf{p}^*$ is defined as

$$l = \hat{\mathbf{d}}^* - (\mathbf{p}^* \times \hat{\mathbf{d}}^*)\mathbf{I}_4, \tag{130}$$

which is a 2-blade in the even-grades subalgebra of DQGA.

The test in DQGA now looks like this: $p_\mathbf{t} l = (1 + \mathbf{t}^*\mathbf{I}_4)(\hat{\mathbf{d}}^* - (\mathbf{p}^* \times \hat{\mathbf{d}}^*)\mathbf{I}_4) = \hat{\mathbf{d}}^* - (\mathbf{p}^* \times \hat{\mathbf{d}}^*)\mathbf{I}_4 + \mathbf{t}^*\mathbf{I}_4\hat{\mathbf{d}}^*$. The term $\mathbf{t}^*\mathbf{I}_4\hat{\mathbf{d}}^*$ can be rewritten as $\mathbf{I}_4\mathbf{t}^*\hat{\mathbf{d}}^* = \mathbf{I}_4(-\mathbf{t}^* \cdot \hat{\mathbf{d}}^* + \mathbf{t}^* \times \hat{\mathbf{d}}^*) = -(\mathbf{t}^* \cdot \hat{\mathbf{d}}^*)\mathbf{I}_4 + (\mathbf{t}^* \times \hat{\mathbf{d}}^*)\mathbf{I}_4$, which is an imaginary quaternion part . The null space entity is the imaginary vector part of the test,

$$\mathrm{Y}(\mathrm{N}_l(p_\mathbf{t} l)) = \mathrm{Y}(\mathrm{V}(\Im(p_\mathbf{t} l))) = (\mathbf{t}^* \times \hat{\mathbf{d}}^*) - (\mathbf{p}^* \times \hat{\mathbf{d}}^*), \tag{131}$$

and it looks like the Plücker coordinates $(\hat{\mathbf{d}}^* : \mathbf{m}^*)$ condition for the line with $\mathbf{m}^* = \mathbf{p}^* \times \hat{\mathbf{d}}^*$. When a point $\mathbf{t}^*$ satisfies the null space condition $(\mathbf{t}^* \times \hat{\mathbf{d}}^*) - (\mathbf{p}^* \times \hat{\mathbf{d}}^*) = 0$, or the Plücker coordinates condition $\mathbf{t}^* \times \hat{\mathbf{d}}^* = \mathbf{m}^*$, then $\mathbf{t}^*$ is a point on the line. Plücker coordinates were derived in Section 2.2.2 about the OPNS PGA 2-blade line $\mathbf{L}$.

A useful identity is

$$l = \bar{\boldsymbol{l}} = \mathbf{I}_3 \boldsymbol{l} \mathbf{I}_3^{-1}, \tag{132}$$

or

$$\boldsymbol{l} = \bar{l} = \mathbf{I}_3 l \mathbf{I}_3^{-1}, \tag{133}$$

that we make use of in deriving other expressions from CPNS PGA to DQGA.

If we do not want to use the null space entity part $\mathrm{N}_l(p_\mathbf{t} l)$ for the null space set condition $\mathrm{N}_l(p_\mathbf{t} l) = 0$, then we can reformulate the test product $\boldsymbol{p}_\mathbf{t} \times \boldsymbol{l}$ into full geometric products as $(\boldsymbol{p}_\mathbf{t}\boldsymbol{l} - \boldsymbol{l}\boldsymbol{p}_\mathbf{t})/2$. Then, we use identities to make substitutions and obtain $(p_\mathbf{t}\mathbf{I}_3\bar{l} - \bar{l}p_\mathbf{t}\mathbf{I}_3)/2 = (p_\mathbf{t} l - \bar{l}p_\mathbf{t})\mathbf{I}_3/2$. So, we abridge the RHS $\mathbf{I}_3$ to obtain exactly the null space entity part $(p_\mathbf{t} l - \bar{l}p_\mathbf{t})/2$. Then, $p_\mathbf{t}$ is a point of the line l if and only if

$$(p_\mathbf{t} l - \bar{l}p_\mathbf{t})/2 = 0. \tag{134}$$

We may save computation by using $\mathrm{N}_l(p_\mathbf{t} l) = 0$.

5.6 DQGA Operations

5.6.1 DQGA Tensor Magnitude Operation

The tensor (or magnitude) of any dual quaternion, or DQGA entity, d is

$$\mathrm{T}(d) = |d|_\mathcal{D} = \sqrt{d d^\dagger} = |q_1|(1 + ((q_{1w}q_{2w} + \mathbf{q}_1^* \cdot \mathbf{q}_2^*)/|q_1|^2)\mathbf{I}_4). \tag{135}$$

The tensor is a *dual number-valued scalar* in the general case. We can use the dual quaternion part operations $\{\Re, \Im, \mathrm{S}, \mathrm{V}\}$, vector calculus dot product $\mathbf{a}^* \cdot \mathbf{b}^* = -\mathbf{a}^* \cdot \mathbf{b}^*$, quaternion tensor $|q_1| = \sqrt{q_1 q_1^\dagger}$, and the PGA dualization operation Y to obtain all of the values in the expression for the dual quaternion tensor $\mathrm{T}(d)$.

Often, we want the inverse of the tensor,

$$T(d)^{-1} = |q_1|^{-1}(1 - ((q_{1w}q_{2w} + \mathbf{q}_1^* \cdot \mathbf{q}_2^*)/|q_1|^2)\mathbf{I}_4), \tag{136}$$

which could be defined as another operation since taking dual number inverses may be troublesome for software implementations.

5.6.2 DQGA Normalization Operation

The normalization $\hat{d} = \mathrm{U}(d)$ ("taking the unit $\hat{d}$ of d") to unit scale (or magnitude) of any homogeneous DQGA dual quaternion entity $d \in \{p, l, \pi\}$, or of $d = d_1 d_2 \ldots$ as any dual quaternion product of entities, is the unit dual quaternion

$$\begin{aligned} \hat{d} = \mathrm{U}(d) &= d(|q_1|^{-1}(1 - ((q_{1w}q_{2w} + \mathbf{q}_1^* \cdot \mathbf{q}_2^*)/|q_1|^2)\mathbf{I}_4)) && (137) \\ &= d|d|_{\mathcal{D}}^{-1} = d\mathrm{T}(d)^{-1} = d\big(\sqrt{d d^\dagger}\big)^{-1}. && (138) \end{aligned}$$

The DQGA entities for point $p_{\mathbf{t}} = 1 + \mathbf{t}^*\mathbf{I}_4$, line $l_{\mathbf{p},\hat{\mathbf{d}}} = \hat{\mathbf{d}}^* - (\mathbf{p}^* \times \hat{\mathbf{d}}^*)\mathbf{I}_4$, and plane $\pi_{\mathbf{p},\hat{\mathbf{n}}} = -(\hat{\mathbf{n}}^* + (\mathbf{p}^* \cdot \hat{\mathbf{n}}^*)\mathbf{I}_4)$ have been defined in standard form as already unit dual quaternions. The form of each entity $d \in \{p, l, \pi\}$ is such that $\hat{d} = d/\mathrm{T}(\Re(d))$. The general case normalization $\hat{d} = d\mathrm{T}(d)^{-1}$ is implemented using $|q_1| = \mathrm{T}(\Re(d))$, $q_{1w}q_{2w} = \mathrm{S}(\Re(d))\mathrm{Y}(\mathrm{S}(\Im(d)))$, and $\mathbf{q}_1^* \cdot \mathbf{q}_2^* = -\mathrm{V}(\Re(d)) \cdot \mathrm{Y}(\mathrm{V}(\Im(d)))$.

The dual quaternion inverse is

$$d^{-1} = \mathrm{U}(d)^\dagger \mathrm{T}(d)^{-1} = d^\dagger \mathrm{T}(d)^{-1}\mathrm{T}(d)^{-1}, \tag{139}$$

for $|q_1| \neq 0$, which could be defined as another operation.

If we want, we can normalize the CPNS PGA entities $\{\boldsymbol{p}, \boldsymbol{l}, \boldsymbol{\pi}\}$ by converting them to their corresponding DQGA entities $\{p = \boldsymbol{p}\mathbf{I}_3^{-1}, l = \bar{\boldsymbol{l}}, \pi = \mathbf{I}_3\boldsymbol{\pi}\}$, normalizing them, and then converting them back to CPNS PGA entities. To normalize the CPNS PGA 3-blade point $\boldsymbol{p}$, this is $\hat{\boldsymbol{p}} = \boldsymbol{p}\mathbf{I}_3^{-1}\mathrm{T}(\boldsymbol{p}\mathbf{I}_3^{-1})^{-1}\mathbf{I}_3 = \boldsymbol{p}(\mathrm{T}(\boldsymbol{p}\mathbf{I}_3^{-1})^{-1})^-$. To normalize the CPNS 2-blade line $\boldsymbol{l}$, this is $\hat{\boldsymbol{l}} = (\bar{\boldsymbol{l}}\mathrm{T}(\bar{\boldsymbol{l}})^{-1})^-$. To normalize the CPNS PGA 1-blade plane $\boldsymbol{\pi}$, this is $\hat{\boldsymbol{\pi}} = \mathbf{I}_3^{-1}\mathbf{I}_3\boldsymbol{\pi}\mathrm{T}(\mathbf{I}_3\boldsymbol{\pi})^{-1} = \boldsymbol{\pi}\mathrm{T}(\mathbf{I}_3\boldsymbol{\pi})^{-1}$.

Any homogeneous DQGA entity $d \in \{p, l, \pi\}$ may not always be unit scale, but d still represents the same entity. In general, a unit dual quaternion $\hat{d}$ may have been multiplied by a dual number z scalar as $d = z\hat{d}$ and still homogeneously represent the same entity. We can we normalize d using the tensor magnitude normalization operation to obtain $\hat{d} = \mathrm{U}(d)$. However, $\hat{d}$ may still have been multiplied by $+1$ or -1 also, and $\hat{d}$ only ensures $T(\hat{d}) = |d|_{\mathcal{D}} = 1$. The orientation of $\hat{d}$ may still be -1 of its intended *orientation*. The DQGA entities $\{p, l, \pi\}$ each have a standard form, but only the point has a standard orientation, and the line and plane each have a subjective application-dependent orientation.

The standard form for the DQGA point, $p_{\mathbf{t}} = 1 + \mathbf{t}^*\mathbf{I}_4$, has orientation $\Re(p_{\mathbf{t}}) = 1$. To ensure that a point $p_{\mathbf{p}}$ is in standard form and standard orientation, we could normalize it first as $\hat{p}_{\mathbf{p}} = p_{\mathbf{p}}T(p_{\mathbf{p}})^{-1} = \mathrm{U}(p_{\mathbf{p}})$ and then check orientation $\Re(\hat{p}_{\mathbf{p}})$, or we simply normalize and set standard form orientation as $\hat{p}_{\mathbf{p}} = p_{\mathbf{p}}\Re(p_{\mathbf{p}})^{-1}$. This should be done before extracting the vector as $\mathbf{p}^* = \mathrm{Y}(\hat{p}_{\mathbf{p}}) = -J_e(\Im(\hat{p}_{\mathbf{p}}))$.

The standard form for the DQGA line, $l_{\mathbf{p},\hat{\mathbf{d}}} = \hat{\mathbf{d}}^* - (\mathbf{p}^* \times \hat{\mathbf{d}}^*)\mathbf{I}_4$, assumes that $\hat{\mathbf{d}}^*$ is the direction of the line acting as axis of counterclockwise rotation around the line in the sense of the right-hand rule. Before using l as an axis of rotation, normalize l as $\hat{l} = l\mathrm{T}(l)^{-1} = \mathrm{U}(l)$ then check $\Re(\hat{l}) = \hat{\mathbf{d}}^*$ to see if $\hat{\mathbf{d}}^*$ has the correct orientation for a positive angle θ of rotation, and if not then use -$\hat{l}$. Rotations and translations preserve the orientation, so if the line is created in the intended correct orientation and then only rotated and translated properly, then $\Re(\hat{l}) = \hat{\mathbf{d}}^*$ will still be the correct intended orientation of the line as it was created. If the line has been reflected in a plane, then we should expect that the reflected axis of the reflected line has the correctly reflected (reversed) orientation and still take $\Re(\hat{l}) = \hat{\mathbf{d}}^*$.

The standard form for the DQGA plane, $\pi_{\mathbf{p},\hat{\mathbf{n}}} = -(\hat{\mathbf{n}}^* + (\mathbf{p}^* \cdot \hat{\mathbf{n}}^*)\mathbf{I}_4)$, usually assumes that $d = \mathbf{p}^* \cdot \hat{\mathbf{n}}^* \geq 0$, but this is not always the case. The orientation, $\hat{\mathbf{n}}^*$ or $-\hat{\mathbf{n}}^*$, is subjective and application-dependent. When using π as a reflection operator, π or $-\pi$ will reflect in the plane the same way and will not matter. Be sure to take the unit of π before extracting its normal vector as $\hat{\mathbf{n}}^* = -\Re(\mathrm{U}(\pi)) = -\Re(\hat{\pi})$. When extracting $\hat{\mathbf{n}}^* = -\Re(\hat{\pi})$, it can be subjective to take either $\{\hat{\mathbf{n}}^*, \hat{\pi}\}$ or $\{-\hat{\mathbf{n}}^*, -\hat{\pi}\}$ when we are not sure which was intended when π was first created. However, most operations such as rotation and translation will preserve the orientation and we just take $\{\hat{\mathbf{n}}^*, \hat{\pi}\}$. If the plane has been reflected in another plane, then we should expect that the reflected normal vector of the reflected plane has the correctly reflected (reversed) orientation and still take $\hat{\mathbf{n}}^* = -\Re(\hat{\pi})$.

We have been careful to make operations that preserve orientation when used properly or carefully, but it is easy to cause a change of orientation (multiplication by -1) when using identities plugged into formulas that intend the opposite orientation. The orientation of any entity is reversed by multiplying the entity by -1, which is unaffected by the unit operation $\mathrm{U}(d)$. Orientation ($\pm$) and magnitude (tensor) $\mathrm{U}(d) = |d|_{\mathcal{D}}$ are separate parts of any entity. The orientation part of a point $p_{\mathbf{p}}$ can be taken as $\Re(\mathrm{U}(p_{\mathbf{p}}))$, as compared to its defined standard form. We cannot take any definite orientation part ($\pm$) of a line or plane, as compared to their derived standard forms.

5.6.3 DQGA Rotation Operation

In DQA, we can rotate all dual quaternions and dual quaternion geometric entities using the quaternion algebra rotor $R = \exp(\theta\hat{\mathbf{n}}/2)$.

In DQGA, the DQA rotor $R = \exp(\theta\hat{\mathbf{n}}/2)$ corresponds to the DQGA rotor

$$R = \exp(\theta\hat{\mathbf{n}}^*/2), \tag{140}$$

where $\hat{\mathbf{n}}^* = \hat{\mathbf{n}}/\mathbf{I}_3$, that we have also used to rotate entities in OPNS PGA and CPNS PGA. By outermorphism, R rotates any quaternion vector $\mathbf{v}^*$ within DQGA expressions, thereby rotating them as a whole rigid body. The rotation is centered on the origin, around the axis $\hat{\mathbf{n}}^*$, by angle θ using R as a versor sandwich product on any DQGA element A as $A' = RAR^{-1}$.

This versor operation is valid on all DQGA entities for rotation centered on the origin, but most of the other DQGA operations, for translation and rotation around lines, are not versor sandwich products, but are instead entity-specific special sandwich products that are derived for each entity. Admittedly, the algebra of operations is much nicer in CPNS PGA since it works very much like CGA with versor operators, but our results on DQGA may still be of interest to those wanting to use just DQA or to those finding some special use of DQGA which might otherwise be neglected.

5.6.4 DQGA Point Rotation Operation Around Line

In CPNS PGA, a point $\boldsymbol{p}$ is rotated around the unit line $\boldsymbol{l} = \hat{\mathbf{d}}^* - (\mathbf{p} \cdot \hat{\mathbf{d}}^*)\mathbf{e}_0$ using the rotor $R_{\boldsymbol{l}} = \exp(\theta\boldsymbol{l}/2) = \cos(\theta/2) + \sin(\theta/2)\boldsymbol{l}$ as $\boldsymbol{p}' = R_{\boldsymbol{l}}\boldsymbol{p}R_{\boldsymbol{l}}^{-1}$. This rotor can also be formed as a composition of rotation and translation as a translated rotor $R_{\boldsymbol{l}} = TRT^{-1}$ or as reflection in two non-parallel planes $R_{\boldsymbol{l}} = \boldsymbol{\pi}_2\boldsymbol{\pi}_1$. We will just use $\exp(\theta\boldsymbol{l}/2)$, which is the easier and more intuitive form.

We use identities $\boldsymbol{p} = p\mathbf{I}_3$, $\boldsymbol{l} = \bar{l}$, $\mathbf{I}_3\bar{l} = l\mathbf{I}_3$. Then, $\boldsymbol{p}' = p'\mathbf{I}_3 = \exp(\theta\bar{l}/2)p\mathbf{I}_3\exp(-\theta\bar{l}/2)$. Therefore, the rotation of point p around line l by angle θ is

$$p' = \exp(\theta\bar{l}/2)p\exp(-\theta l/2). \tag{141}$$

It is important that $l = \hat{l} = \mathrm{U}(l)$ be a unit line, or else the angle θ will be scaled incorrectly by any magnitude $\mathrm{T}(l)$ on l. The sense of rotation is by right-hand rule around the line through axis of rotation direction $\hat{\mathbf{d}}^*$. This is not a versor operation; it is a special dual quaternion sandwich product that is entity-specific, for rotating a DQGA point around a DQGA line. Each DQGA entity has a different formula for this operation in DQGA.

5.6.5 DQGA Plane Rotation Operation Around Line

Similar to Section 5.6.4 for points, in CPNS PGA, a plane $\boldsymbol{\pi}$ is rotated around the unit line $\boldsymbol{l}$ using the rotor $R_{\boldsymbol{l}} = \exp(\theta\boldsymbol{l}/2)$ as $\boldsymbol{\pi}' = R_{\boldsymbol{l}}\boldsymbol{\pi}R_{\boldsymbol{l}}^{-1}$. We use identities $\boldsymbol{\pi} = \mathbf{I}_3^{-1}\pi$, $\boldsymbol{l} = \bar{l}$, $\bar{l}\mathbf{I}_3^{-1} = \mathbf{I}_3^{-1}l$. Then, $\boldsymbol{\pi}' = \mathbf{I}_3^{-1}\pi' = \exp(\theta\bar{l}/2)\mathbf{I}_3^{-1}\pi\exp(-\theta\bar{l}/2)$. Therefore, the rotation of plane π around line l by angle θ is

$$\pi' = \exp(\theta l/2)\pi\exp(-\theta\bar{l}/2). \tag{142}$$

5.6.6 DQGA Line Rotation Operation Around Line

Similar to (5.6.4) for points, in CPNS PGA, a line $\boldsymbol{l}_1$ is rotated around the unit line $\boldsymbol{l}_2$ using the rotor $R_{\boldsymbol{l}_2} = \exp(\theta\boldsymbol{l}_2/2)$ as $\boldsymbol{l}_1' = R_{\boldsymbol{l}_2}\boldsymbol{l}_1R_{\boldsymbol{l}_2}^{-1}$. We use identities $\boldsymbol{l} = \bar{l}$, $\bar{l}\mathbf{I}_3^{-1} = \mathbf{I}_3^{-1}l$. Then, $\boldsymbol{l}_1' = \bar{l}_1' = \exp(\theta\bar{l}_2/2)\bar{l}_1\exp(-\theta\bar{l}_2/2)$. Therefore, the rotation of line l_1 around line l_2 by angle θ is

$$l_1' = \exp(\theta l_2/2)l_1\exp(-\theta l_2/2). \tag{143}$$

This is a versor operation.

Only for the DQGA line entity is this operation, rotation around a line, a versor operation. It turns out that, for the DQGA line entity, all rotation and translation operations are versor operations, just as they are in CPNS PGA. So, for DQGA lines, the versor operations can be further composed as versors and applied to a DQGA line entity. For the DQGA point and plane, we have to be careful to use their entity-specific rotation and translation operations, and compose them as special sandwich products that are not versor sandwich products.

5.6.7 DQGA Plane Reflection in Plane Operation

In CPNS PGA, the plane $\boldsymbol{\pi}_1$ is reflected in unit plane $\boldsymbol{\pi}_2$ as $\boldsymbol{\pi}_1' = -\boldsymbol{\pi}_2\boldsymbol{\pi}_1\boldsymbol{\pi}_2$. We use the identities $\boldsymbol{\pi} = \mathbf{I}_3^{-1}\pi$, $\bar{d} = \mathbf{I}_3 d\mathbf{I}_3^{-1}$. Then, the reflected plane is $\boldsymbol{\pi}_1' = \mathbf{I}_3^{-1}\pi_1' = -\mathbf{I}_3^{-1}\pi_2\mathbf{I}_3^{-1}\pi_1\mathbf{I}_3^{-1}\pi_2$. Therefore, plane π_1 reflected in plane π_2 is

$$\pi_1' = \pi_2\bar{\pi}_1\pi_2. \tag{144}$$

We can compose two reflections to generate rotation or translation. Recall that, we must use $-\boldsymbol{\pi}_2\boldsymbol{\pi}_1\boldsymbol{\pi}_2$, not $\boldsymbol{\pi}_2\boldsymbol{\pi}_1\boldsymbol{\pi}_2$. The later is reflection in a normal vector, and the former is reflection in the plane of the normal vector.

5.6.8 DQGA Line Reflection in Plane Operation

In CPNS PGA, the line $\boldsymbol{l}$ is reflected in unit plane $\boldsymbol{\pi}$ as $\boldsymbol{l}' = \boldsymbol{\pi l \pi}$. We use the identities $\boldsymbol{l} = \bar{l}$, $\boldsymbol{\pi} = \mathbf{I}_3^{-1}\pi$, $\bar{d} = \mathbf{I}_3 d \mathbf{I}_3^{-1}$. Then, the reflected line is $\boldsymbol{l}' = \bar{l}' = \mathbf{I}_3^{-1}\pi\bar{l}\mathbf{I}_3^{-1}\pi$. Therefore, line l reflected in plane π is

$$l' = -(\bar{\pi} l \pi)^- = -\pi \bar{l} \bar{\pi}. \tag{145}$$

We can compose two reflections to generate rotation or translation.

If we view the CPNS PGA line as $\boldsymbol{l} = \boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2$, then it is correctly argued that we must reflect each plane as $\boldsymbol{\pi}_i' = -\boldsymbol{\pi}\boldsymbol{\pi}_i\boldsymbol{\pi}$, and therefore $\boldsymbol{l}' = (-1)^2\boldsymbol{\pi l \pi} = \boldsymbol{\pi l \pi}$, which is what we have done already. The handedness of the axis or direction of l' is opposite of l, so that it generates the opposite handedness of rotations. The effect can be understood by looking at a ceiling fan and holding your right hand according to the right-hand-rule, so that your right hand is "holding" the fan axis with your fingers curling around it in the direction of the fan rotation and your thumb points into the axis direction. Then, look at the fan in a mirror and hold it with your right hand again. Your thumb will have to reverse direction to curl with the spinning as seen in the mirror. In the mirror, your left hand will hold the axis and still point the same direction. The handedness is reversed. The image of the line itself is a mirror image as expected. The direction or axis of l' is the negative of what you might expect, but it is correct for a reflection.

5.6.9 DQGA Point Reflection in Plane Operation

In CPNA PGA, the point $\boldsymbol{p}$ is reflected in the unit plane $\boldsymbol{\pi}$ as $\boldsymbol{p}' = \boldsymbol{\pi p \pi}$. We use the identities $\boldsymbol{\pi} = \mathbf{I}_3^{-1}\pi = -\bar{\pi}\mathbf{I}_3$, $\boldsymbol{p} = p\mathbf{I}_3$, $\bar{d} = \mathbf{I}_3 d \mathbf{I}_3^{-1}$. Then, the reflected point is $\boldsymbol{p}' = p'\mathbf{I}_3 = -\mathbf{I}_3^{-1}\pi p \mathbf{I}_3 \bar{\pi}\mathbf{I}_3$. Therefore, point p reflected in plane π is

$$p' = -\bar{\pi}\bar{p}\bar{\pi} = -(\pi p \pi)^-. \tag{146}$$

We can compose two reflections to generate rotation or translation. Caution: This form of reflection maintains p' in standard form orientation as a non-oriented point. In the following paragraphs, we explain non-oriented point and oriented point reflection.

If we view the CPNS PGA point as $\boldsymbol{p} = \boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_3$, then some may argue that we must reflect each plane as $\boldsymbol{\pi}_i' = -\boldsymbol{\pi}\boldsymbol{\pi}_i\boldsymbol{\pi}$, and therefore $\boldsymbol{p}' = (-1)^3\boldsymbol{\pi p \pi} = -\boldsymbol{\pi p \pi}$. However, a point does not actually have an orientation the way that a plane or line does, and it turns out that $-\boldsymbol{\pi p \pi}$ scales the point into a non-standard form with negative orientation $-\boldsymbol{p}' = -\mathbf{I}_3 + \mathbf{e}_0\mathbf{p}^{*\prime}$. We believe that points should usually be maintained in standard form orientation, while the orientation of planes and lines must be allowed to reflect.

Then again, if we join three points $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3\}$ arranged counterclockwise as the plane $\boldsymbol{\pi}_{123} = (\boldsymbol{p}_3^{-\star} \wedge \boldsymbol{p}_2^{-\star} \wedge \boldsymbol{p}_1^{-\star})^\star$, then the three reflected points $\{\boldsymbol{p}_1', \boldsymbol{p}_2', \boldsymbol{p}_3'\}$ appear to be arranged clockwise in the mirror view. If we allow point orientation $\boldsymbol{p}_i' = -\boldsymbol{\pi}\boldsymbol{p}_i\boldsymbol{\pi}$, then the reflected plane is $\boldsymbol{\pi}_{123}' = -(\boldsymbol{p}_3'^{-\star} \wedge \boldsymbol{p}_2'^{-\star} \wedge \boldsymbol{p}_1'^{-\star})^\star = (\boldsymbol{p}_1'^{-\star} \wedge \boldsymbol{p}_2'^{-\star} \wedge \boldsymbol{p}_3'^{-\star})^\star$. The plane $\boldsymbol{\pi}_{123}$ and its reflected image $\boldsymbol{\pi}_{123}'$ are either both facing the reflection plane $\boldsymbol{\pi}$ or backing the reflection plane $\boldsymbol{\pi}$. This gives the correct reflection.

Whether to reflect non-oriented standard form points $\boldsymbol{p}' = \boldsymbol{\pi p \pi}$ or to reflect oriented points $\boldsymbol{p}' = -\boldsymbol{\pi p \pi}$ depends on how the points are being used. So, it is application-specific how to choose this. Two classes of points could be defined: non-oriented points and oriented points. For the class of non-oriented points, the reflection in a plane is $\boldsymbol{p}' = \boldsymbol{\pi p \pi}$. For the class of oriented points, the reflection in a plane is $\boldsymbol{p}' = -\boldsymbol{\pi p \pi}$.

All of these orientation considerations carry over to the DQGA reflection when we make identities substitutions as we have already shown.

5.6.10 DQGA Point Reflection in Line Operation

In CPNS PGA, the point $\boldsymbol{p}$ is reflected in unit line $\boldsymbol{l}$ as $\boldsymbol{p}' = \boldsymbol{l}\boldsymbol{p}\boldsymbol{l}^{-1} = -\boldsymbol{l}\boldsymbol{p}\boldsymbol{l}$. We use identities ($\boldsymbol{p} = p\mathbf{I}_3$, $\boldsymbol{l} = \bar{l}$, $\mathbf{I}_3\bar{l} = l\mathbf{I}_3$) and switch to geometric products. Then, the reflected point is $\boldsymbol{p}' = p'\mathbf{I}_3 = -\bar{l}p\mathbf{I}_3\bar{l}$. Therefore, point p "reflected" in line l (actually rotation by 180° around line l) is

$$p' = -\bar{l}pl. \tag{147}$$

The "reflection" in a line $\boldsymbol{l}$ is not actually a reflection, it is an orientation-preserving rotation around the line by 180° since $\boldsymbol{l} = \exp(\pi\boldsymbol{l}/2)$, which is a line rotor or translated rotor.

5.6.11 DQGA Line Refection in Line Operation

In CPNS PGA, the unit line $\boldsymbol{l}_1$ is reflected in unit line $\boldsymbol{l}_2$ as $\boldsymbol{l}_1' = \boldsymbol{l}_2\boldsymbol{l}_1\boldsymbol{l}_2^{-1} = -\boldsymbol{l}_2\boldsymbol{l}_1\boldsymbol{l}_2$. We use identities ($\boldsymbol{l} = \bar{l}$, $ll = -1$) and switch to geometric products. Then, the reflected line is $\boldsymbol{l}_1' = \bar{l}_1' = -\bar{l}_2\bar{l}_1\bar{l}_2$. Therefore, line l_1 "reflected" in line l_2 (actually rotation by 180° around line l_2) is

$$l_1' = -l_2l_1l_2. \tag{148}$$

Intuitively, if $l_1 = l_2$, then $l_1' = l_2$ as expected. The "reflection" in a line $\boldsymbol{l}$ is not actually a reflection, it is an orientation-preserving rotation around the line by 180° since $\boldsymbol{l} = \exp(\pi\boldsymbol{l}/2)$, which is a line rotor or translated rotor.

5.6.12 DQGA Point Translation Operation

In CPNS PGA, a point $\boldsymbol{p}_\mathbf{p}$ is translated as $\boldsymbol{p}_{\mathbf{p}'} = T\boldsymbol{p}_\mathbf{p}T^{-1}$. We use the identities $T = p_{\mathbf{d}/2} = 1 + (\mathbf{d}^*/2)\mathbf{I}_4$, $T^{-1} = p_{-\mathbf{d}/2}$, $\mathbf{I}_3T^{-1} = T\mathbf{I}_3$, $\boldsymbol{p}_\mathbf{p} = p_\mathbf{p}\mathbf{I}_3$, $p_\mathbf{p} = 1 + \mathbf{p}^*\mathbf{I}_4$. Then, the translated point is $\boldsymbol{p}_{\mathbf{p}'} = p_{\mathbf{p}'}\mathbf{I}_3 = Tp_\mathbf{p}T\mathbf{I}_3$. Therefore, point $p_\mathbf{p}$ translated by $\mathbf{d}^*$ is

$$p_{\mathbf{p}'} = Tp_\mathbf{p}T. \tag{149}$$

Points have commutative multiplication, so $p_{\mathbf{p}'} = T^2p_\mathbf{p} = p_\mathbf{d}p_\mathbf{p} = p_{\mathbf{p}+\mathbf{d}}$. This confirms the expected form of this translation operation, as multiplication of one point with another displacement point as addition.

5.6.13 DQGA Plane Translation Operation

In CPNS PGA, a plane $\boldsymbol{\pi}$ is translated as $\boldsymbol{\pi}' = T\boldsymbol{\pi}T^{-1}$. We use the identities $T = p_{\mathbf{d}/2} = 1 + (\mathbf{d}^*/2)\mathbf{I}_4$, $T^{-1} = p_{-\mathbf{d}/2}$, $\boldsymbol{\pi} = \mathbf{I}_3^{-1}\pi$, $T\mathbf{I}_3^{-1} = \mathbf{I}_3^{-1}T^{-1}$. Then, the translated plane is $\boldsymbol{\pi}' = \mathbf{I}_3^{-1}\pi' = \mathbf{I}_3^{-1}p_{-\mathbf{d}/2}\pi p_{-\mathbf{d}/2}$. Therefore, plane π translated by $\mathbf{d}^*$ is

$$\pi' = p_{-\mathbf{d}/2}\pi p_{-\mathbf{d}/2} = T^{-1}\pi T^{-1} = \bar{p}_{\mathbf{d}/2}\pi\bar{p}_{\mathbf{d}/2} = \bar{T}p_\mathbf{d}\bar{T}. \tag{150}$$

5.6.14 DQGA Line Translation Operation

In CPNS PGA, a line $\boldsymbol{l}$ is translated as $\boldsymbol{l}' = T\boldsymbol{l}T^{-1}$. We use the identities $T = p_{\mathbf{d}/2} = 1 + (\mathbf{d}^*/2)\mathbf{I}_4$, $T^{-1} = p_{-\mathbf{d}/2}$, $\boldsymbol{l} = \bar{l}$, $\bar{d} = \mathbf{I}_3d\mathbf{I}_3^{-1}$, $\bar{T} = T^{-1}$. Then, the translated line is $\boldsymbol{l}' = \bar{l}' = T\bar{l}T^{-1}$. Therefore, line l translated by $\mathbf{d}^*$ is

$$l' = T^{-1}lT = \bar{T}lT. \tag{151}$$

This is like a translation versor operation for translating by the negative displacement $-\mathbf{d}^*$, but this is an actual versor operation that achieves the positive displacement $\mathbf{d}^*$. Just to be clear, we set $T^{-1} = p_{-\mathbf{d}/2}$ and $T = p_{\mathbf{d}/2}$, and translate l by displacement $\mathbf{d}^*$ as $l' = T^{-1}lT$. As a check, $\bar{l}' = \boldsymbol{l}' = (T^{-1}lT)^- = T\boldsymbol{l}T^{-1}$. For the DQGA line entity l, we can compose rotations and translations as versors, but we cannot do that for the other DQGA entities.

5.7 DQGA Intersections and Point Tests

5.7.1 DQGA Point Intersection Tests

We have defined the DQGA surface entities $A \in \{\bar{p}_{\mathbf{p}}, \pi_{\mathbf{p},\hat{\mathbf{n}}}, l_{\mathbf{p},\hat{\mathbf{d}}}\}$ with respect to testing a DQGA point $p_{\mathbf{t}}$ against them, to determine if $p_{\mathbf{t}}$ is in the *null space set* $\{p_{\mathbf{t}} : \mathrm{N}_A(p_{\mathbf{t}}A) = 0\}$ of the *null space entity part* $\mathrm{N}_A(p_{\mathbf{t}}A)$ of the test product $p_{\mathbf{t}}A$. The null space entity part $\mathrm{N}_A(p_{\mathbf{t}}A)$ depends on type of surface A.

For a point surface $p = \bar{p}_{\mathbf{p}}$, $A = p$ and

$$\mathrm{N}_p(p_{\mathbf{t}}\bar{p}_{\mathbf{b}}) = \mathrm{V}(\Im(p_{\mathbf{t}}\bar{p}_{\mathbf{b}})) = 0 \tag{152}$$

for the null space set of a single point $\{p_{\mathbf{t}} : \mathrm{N}_p(p_{\mathbf{t}}\bar{p}_{\mathbf{b}}) = (\mathbf{t}^* - \mathbf{b}^*)\mathbf{I}_4 = 0\} = \{p_{\mathbf{t}} = p_{\mathbf{b}}\}$. The point of a point surface $\bar{p}_{\mathbf{b}}$ is $\bar{\bar{p}}_{\mathbf{b}} = p_{\mathbf{b}}$.

For a plane surface $\pi = \pi_{\mathbf{p},\hat{\mathbf{n}}}$, $A = \pi$ and

$$\mathrm{N}_\pi(p_{\mathbf{t}}\pi_{\mathbf{p},\hat{\mathbf{n}}}) = \mathrm{S}(\Im(p_{\mathbf{t}}\pi_{\mathbf{p},\hat{\mathbf{n}}})) = 0 \tag{153}$$

for the null space set of the entire plain of points $\{p_{\mathbf{t}} : \mathrm{N}_\pi(p_{\mathbf{t}}\pi_{\mathbf{p},\hat{\mathbf{n}}}) = (\mathbf{t}^*\cdot\hat{\mathbf{n}}^* - \mathbf{p}^*\cdot\hat{\mathbf{n}}^*)\mathbf{I}_4 = 0\}$.

For a line surface $l = l_{\mathbf{p},\hat{\mathbf{d}}}$, $A = l$ and

$$\mathrm{N}_l(p_{\mathbf{t}}l_{\mathbf{p},\hat{\mathbf{d}}}) = \mathrm{V}(\Im(p_{\mathbf{t}}l_{\mathbf{p},\hat{\mathbf{d}}})) = 0 \tag{154}$$

for the null space set of the entire line of points $\{p_{\mathbf{t}} : \mathrm{N}_l(p_{\mathbf{t}}l_{\mathbf{p},\hat{\mathbf{d}}}) = (\mathbf{t}^*\times\hat{\mathbf{d}}^* - \mathbf{p}^*\times\hat{\mathbf{d}}^*)\mathbf{I}_4 = 0\}$, where $\mathbf{p}^*\times\hat{\mathbf{d}}^* = \mathbf{m}^*$ is called the *moment* in Plücker coordinates $(\hat{\mathbf{d}}^* : \mathbf{m}^*)$ for the line.

5.7.2 DQGA Plane and Plane Intersection as Line

In CPNS PGA, plane $\boldsymbol{\pi}_1$ and plane $\boldsymbol{\pi}_2$ intersect as the line $\boldsymbol{l} = \boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2$. We use the identities $\boldsymbol{l} = \bar{l}$, $\boldsymbol{\pi} = \mathbf{I}_3^{-1}\pi$, and $\bar{d} = \mathbf{I}_3 d \mathbf{I}_3^{-1}$ and switch to geometric products. Then, the line is $l = \bar{\boldsymbol{l}} = \mathrm{L}(\mathbf{I}_3\mathbf{I}_3^{-1}\pi_1\mathbf{I}_3^{-1}\pi_2\mathbf{I}_3^{-1})$. Therefore, the intersection of plane π_1 and plane π_2 is the line

$$l = \mathrm{L}(-\pi_1\bar{\pi}_2). \tag{155}$$

Notice that, we have to take the *line part* using the $\mathrm{L}(d) = \mathrm{V}(d)$ operator, which replaces the wedge product (a geometric product part operator) used in the corresponding CPNS PGA.

If we do not want to use $\mathrm{L}(d)$, then we can fully reformulate into geometric products as $\boldsymbol{l} = \boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2 = (\boldsymbol{\pi}_1\boldsymbol{\pi}_2 - \boldsymbol{\pi}_2\boldsymbol{\pi}_1)/2$ for two vectors. Then, $l = \bar{\boldsymbol{l}} = \mathbf{I}_3(\mathbf{I}_3^{-1}\pi_1\mathbf{I}_3^{-1}\pi_2 - \mathbf{I}_3^{-1}\pi_2\mathbf{I}_3^{-1}\pi_1)\mathbf{I}_3^{-1}/2$. Therefore, the intersection of plane π_1 and plane π_2 is the line

$$l = -(\pi_1\bar{\pi}_2 - \pi_2\bar{\pi}_1)/2. \tag{156}$$

Using $\mathrm{L}(d)$ may save computation.

When forming a line l as intersection of two planes, we cannot be sure of its scale, so it may need to be normalized as $\hat{l} = \mathrm{U}(l)$, but it will have the same scale and orientation as the corresponding CPNS PGA line $\boldsymbol{l} = \bar{l}$.

5.7.3 DQGA Line and Plane Intersection as Point

In CPNS PGA, non-parallel line $\boldsymbol{l}$ and plane $\boldsymbol{\pi}$ intersect as the point $\boldsymbol{p} = \boldsymbol{l} \wedge \boldsymbol{\pi}$. We use the identities $\boldsymbol{l} = \bar{l}$, $\boldsymbol{\pi} = \mathbf{I}_3^{-1}\pi$, $\bar{d} = \mathbf{I}_3 d \mathbf{I}_3^{-1}$, and $\boldsymbol{p} = p\mathbf{I}_3$ and switch to geometric products. Then, the point is $p = \boldsymbol{p}\mathbf{I}_3^{-1} = \mathrm{P}(\bar{l}\mathbf{I}_3^{-1}\pi\mathbf{I}_3^{-1})$. Therefore, the intersection of line l and plane π is the point

$$p = \mathrm{P}(-\bar{l}\bar{\pi}). \tag{157}$$

Notice that, we have to take the *point part* using the $\mathrm{P}(d) = (d + \bar{d}^{\dagger})/2$ operator, which replaces the wedge product (a geometric product part operator) used in the corresponding CPNS PGA.

If we do not want to use $\mathrm{P}(d)$, then we can fully reformulate into geometric products as $\boldsymbol{p} = \boldsymbol{l} \wedge \boldsymbol{\pi} = (\boldsymbol{l}\boldsymbol{\pi} + \boldsymbol{\pi}\boldsymbol{l})/2$ for a bivector $\boldsymbol{l}$ and vector $\boldsymbol{\pi}$. The symmetry here is $(-1)^{rs}$ with $r = 2$ and $s = 1$, so $\boldsymbol{l} \wedge \boldsymbol{\pi}$ is part of the symmetric product. Then, $p = \boldsymbol{p}\mathbf{I}_3^{-1} = (\bar{l}\mathbf{I}_3^{-1}\pi + \mathbf{I}_3^{-1}\pi\bar{l})\mathbf{I}_3^{-1}/2$. Therefore, the intersection of line l and plane π is the point

$$p = -(\bar{l}\bar{\pi} + \bar{\pi}l)/2. \tag{158}$$

Using $\mathrm{P}(d)$ may save computation.

When forming a point p as intersection of line and plane, we cannot be sure of its scale, so it may need to be normalized as $\hat{p} = \mathrm{U}(p)$, but it will have the same scale and orientation as the corresponding CPNS PGA point $\boldsymbol{p} = p\mathbf{I}_3$. If a point surface is desired, then it is $\bar{p} = \mathrm{P}(-l\pi)$ so that the null space set is p.

5.7.4 DQGA Plane and Plane and Plane Intersection as Point

In CPNS PGA, three non-parallel planes, plane $\boldsymbol{\pi}_1$ and plane $\boldsymbol{\pi}_2$ and plane $\boldsymbol{\pi}_3$, intersect as the point $\boldsymbol{p} = \boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_3$. We use the identities $\boldsymbol{\pi} = \mathbf{I}_3^{-1}\pi$, $\boldsymbol{p} = p\mathbf{I}_3$, and $\bar{d} = \mathbf{I}_3 d \mathbf{I}_3^{-1}$ and switch to geometric products. Then, the point is $p = \boldsymbol{p}\mathbf{I}_3^{-1} = \mathrm{P}(\mathbf{I}_3^{-1}\pi_1\mathbf{I}_3^{-1}\pi_2\mathbf{I}_3^{-1}\pi_3\mathbf{I}_3^{-1})$. Therefore, the intersection of three planes, π_1, π_2, and π_3, is the point

$$p = \mathrm{P}(\bar{\pi}_1\pi_2\bar{\pi}_3) = \mathrm{P}(-\bar{l}_{12}\bar{\pi}_3) = \mathrm{P}(-\bar{\pi}_1 l_{23}). \tag{159}$$

Notice that, we have to take the *point part* using the $\mathrm{P}(d)$ operator, which replaces the wedge products (a geometric product part operator) used in the corresponding CPNS PGA. The line products, $l_{12} = -\pi_1\bar{\pi}_2$ and $l_{23} = -\pi_2\bar{\pi}_3$, have been deferred from taking their line parts using operator $\mathrm{L}(d)$ on them so that they are further used to generate another entity. When taking products of dual quaternion entities that correspond to geometric or wedge products in CPNS PGA, we can defer taking the correct entity part (point $\mathrm{P}(d)$, plane $\Pi(d)$, or line $\mathrm{L}(d)$) until the end when we have our final dual quaternion product of the entities that produces another entity of an expected type, which we then take from the product by using the correct part operator.

If we do not want to use $\mathrm{P}(d)$, then we can fully reformulate into geometric products as $p = \boldsymbol{p}\mathbf{I}_3^{-1} = (\boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_3)\mathbf{I}_3^{-1} = ((\boldsymbol{\pi}_1\boldsymbol{\pi}_2 - \boldsymbol{\pi}_2\boldsymbol{\pi}_1)\boldsymbol{\pi}_3 + \boldsymbol{\pi}_3(\boldsymbol{\pi}_1\boldsymbol{\pi}_2 - \boldsymbol{\pi}_2\boldsymbol{\pi}_1))\mathbf{I}_3^{-1}/4$. Then, we make the substitutions and get $p = ((-\bar{\pi}_1\pi_2 + \bar{\pi}_2\pi_1)(-\bar{\pi}_3) + (-\bar{\pi}_3)(-\pi_1\bar{\pi}_2 + \pi_2\bar{\pi}_1))/4$. Therefore, the intersection of three planes, π_1, π_2, and π_3, is the point

$$p = (\bar{\pi}_1\pi_2\bar{\pi}_3 - \bar{\pi}_2\pi_1\bar{\pi}_3 + \bar{\pi}_3\pi_1\bar{\pi}_2 - \bar{\pi}_3\pi_2\bar{\pi}_1)/4. \tag{160}$$

Using $\mathrm{P}(d)$ may save computation, but both ways give the same point entity.

When forming a point p as intersection of three planes, we cannot be sure of its scale, so it may need to be normalized as $\hat{p} = \mathrm{U}(p)$, but it will have the same scale and orientation as the corresponding CPNS PGA point $\boldsymbol{p} = p\mathbf{I}_3$. If a point surface is desired, then it is $\bar{p} = \mathrm{P}(\pi_1\bar{\pi}_2\pi_3) = \mathrm{P}(-l_{12}\pi_3) = \mathrm{P}(-\pi_1\bar{l}_{23})$ so that the null space set is p.

5.8 DQGA Projection Operations

In CPNS PGA, we only make projections of a smaller-dimensional geometric entity $a \in \{\boldsymbol{p}, \boldsymbol{l}, \boldsymbol{\pi}\}$ onto a subspace of a larger-dimensional geometric entity $A \in \{\boldsymbol{p}, \boldsymbol{l}, \boldsymbol{\pi}\}$. A point is 0-dimensional, a line is 1-dimensional, and a plane is 2-dimensional in terms of geometric degrees of freedom. Though it can be done, the projection of same-dimensional geometric entities just results in bijectively mapping entity a one-to-one and onto the entire surface of entity A, resulting in A again. The general projection operation in CPNS PGA is $a' = (a \cdot A)A^{-1}$ (same as in CGA) and results in orthographic projection of a onto A. Therefore, we have three projections: $(\boldsymbol{p} \cdot \boldsymbol{\pi})\boldsymbol{\pi}^{-1}$, $(\boldsymbol{l} \cdot \boldsymbol{\pi})\boldsymbol{\pi}^{-1}$, $(\boldsymbol{p} \cdot \boldsymbol{l})\boldsymbol{l}^{-1}$. In this section, we use identities to convert these projections into DQGA forms.

5.8.1 DQGA Point Projection onto Plane

In CPGA PGA, the projection $\boldsymbol{p}'$ of point $\boldsymbol{p}$ onto unit plane $\boldsymbol{\pi}$ is $\boldsymbol{p}' = (\boldsymbol{p} \cdot \boldsymbol{\pi})\boldsymbol{\pi}$. We use identities ($\boldsymbol{p} = p\mathbf{I}_3$, $\boldsymbol{\pi} = \mathbf{I}_3^{-1}\pi$, $\bar{d} = \mathbf{I}_3 d\mathbf{I}_3^{-1}$, $\pi\bar{\pi} = -1$, $\boldsymbol{p} \cdot \boldsymbol{\pi} = (\boldsymbol{p}\boldsymbol{\pi} + \boldsymbol{\pi}\boldsymbol{p})/2$ [n.b., symmetry $(-1)^{r(s-1)} = (-1)^{1(3-1)} = +1$]) and switch to geometric products. Then, $p' = \boldsymbol{p}'\mathbf{I}_3^{-1} = ((p\mathbf{I}_3\mathbf{I}_3^{-1}\pi + \mathbf{I}_3^{-1}\pi p\mathbf{I}_3)\mathbf{I}_3^{-1}\pi/2)\mathbf{I}_3^{-1}$. Therefore, the projection of point p onto plane π is the point

$$p' = -(p\pi\bar{\pi} + \bar{\pi}\bar{p}\bar{\pi})/2. \tag{161}$$

The projected point p' is the point on the plane π that is closest to the point p, so that displacement $\mathbf{d} = p - p'$ is normal to plane π and directed toward p.

For unit plane π, we have $\pi\bar{\pi} = -1$ and $p' = (p - \bar{\pi}\bar{p}\bar{\pi})/2$. We notice that, $-\bar{\pi}\bar{p}\bar{\pi}$ is p reflected in the plane π as a non-oriented point, and that $(p - \bar{\pi}\bar{p}\bar{\pi})/2$ is the average point between p and $-\bar{\pi}\bar{p}\bar{\pi}$, which is the point p' on the plane between them.

Notice that, we do not have to take the point part using $\mathrm{P}(d)$ since we reformulated as $(\boldsymbol{p} \cdot \boldsymbol{\pi})\boldsymbol{\pi} = (\boldsymbol{p}\boldsymbol{\pi} + \boldsymbol{\pi}\boldsymbol{p})\boldsymbol{\pi}/2$, which is all geometric products that take no special part. We can often do this, but it gets messy for some products. It was necessary in this case to reformulate into all geometric products since $\boldsymbol{p} \cdot \boldsymbol{\pi}$ must be done first according to geometric algebra operator precedence rules (i.e., $\boldsymbol{p} \cdot \boldsymbol{\pi}\boldsymbol{\pi} = (\boldsymbol{p} \cdot \boldsymbol{\pi})\boldsymbol{\pi} \neq \boldsymbol{p} \cdot (\boldsymbol{\pi}\boldsymbol{\pi})$). For a wedge product, we have $\boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_3 = (\boldsymbol{\pi}_1 \wedge \boldsymbol{\pi}_2) \wedge \boldsymbol{\pi}_3 = \boldsymbol{\pi}_1 \wedge (\boldsymbol{\pi}_2 \wedge \boldsymbol{\pi}_3)$, so we are not constrained to enforce any precedence of the operations and we can defer taking parts until later.

5.8.2 DQGA Line Projection onto Plane

In CPNS PGA, the projection $\boldsymbol{l}'$ of line $\boldsymbol{l}$ onto unit plane $\boldsymbol{\pi}$ is $\boldsymbol{l}' = (\boldsymbol{l} \cdot \boldsymbol{\pi})\boldsymbol{\pi}$. We use identities ($\boldsymbol{l} = \bar{l}$, $\boldsymbol{\pi} = \mathbf{I}_3^{-1}\pi$, $\bar{d} = \mathbf{I}_3 d\mathbf{I}_3^{-1}$, $\pi\bar{\pi} = -1$, $\boldsymbol{l} \cdot \boldsymbol{\pi} = (\boldsymbol{l}\boldsymbol{\pi} - \boldsymbol{\pi}\boldsymbol{l})/2$ [n.b., symmetry $(-1)^{r(s-1)} = (-1)^{1(2-1)} = -1$]) and switch to geometric products. Then, $l' = \bar{\boldsymbol{l}}' = \mathbf{I}_3((\bar{l}\mathbf{I}_3^{-1}\pi - \mathbf{I}_3^{-1}\pi\bar{l})\mathbf{I}_3^{-1}\pi/2)\mathbf{I}_3^{-1}$. Therefore, the projection of line l onto plane π is the line

$$l' = (-l\pi\bar{\pi} + \pi\bar{l}\bar{\pi})/2. \tag{162}$$

The projected line l' is the line of points on the plane π that each are closest to corresponding points on l, so it is an orthographic projection, projecting points of the line l straight onto the plane along paths parallel to the plane normal vector.

For unit plane π, we have $\pi\bar{\pi} = -1$ and $l' = (l + \pi\bar{l}\bar{\pi})/2$. We notice that, $-\pi\bar{l}\bar{\pi}$ is l reflected in the plane π. Reflection in a plane causes an orientation reversal in the reflected entity (except for the non-oriented point reflection). Then, $\pi\bar{l}\bar{\pi}$ restores orientation to that of l, and then they are averaged as l'. The average line l' is on the plane between them.

Notice, we do not have to take the line part using $\mathrm{L}(d)$ since we reformulated as $\boldsymbol{l} \cdot \boldsymbol{\pi} = (\boldsymbol{l}\boldsymbol{\pi} - \boldsymbol{\pi}\boldsymbol{l})\boldsymbol{\pi}/2$, which is all geometric products that take no special part. It was necessary in this case to reformulate into all geometric products since $\boldsymbol{l} \cdot \boldsymbol{\pi}$ must be done first according to geometric algebra operator precedence rules, similar to the case of the point projection onto plane.

5.8.3 DQGA Point Projection onto Line

In CPNS PGA, the projection $\boldsymbol{p}'$ of point $\boldsymbol{p}$ onto the unit line $\boldsymbol{l}$ is $\boldsymbol{p}' = (\boldsymbol{p} \cdot \boldsymbol{l})\boldsymbol{l}^{-1} = -(\boldsymbol{p} \cdot \boldsymbol{l})\boldsymbol{l}$. We use identities ($\boldsymbol{p} = p\mathbf{I}_3$, $\boldsymbol{l} = \bar{l}$, $\bar{d} = \mathbf{I}_3 d \mathbf{I}_3^{-1}$, $\mathbf{I}_3\bar{l} = l\mathbf{I}_3$, $\boldsymbol{p} \cdot \boldsymbol{l} = (\boldsymbol{p}\boldsymbol{l} + \boldsymbol{l}\boldsymbol{p})/2$) and switch to geometric products. Then, $\boldsymbol{p}' = p'\mathbf{I}_3 = -(p\mathbf{I}_3\bar{l} + \bar{l}p\mathbf{I}_3)\bar{l}/2$. Therefore, the projection of point p onto line l is the point

$$p' = -(pl + \bar{l}p)l/2. \tag{163}$$

For unit line l, we have $ll = l^2 = -1$ and $p' = (p - \bar{l}pl)/2$. We notice that, $-\bar{l}pl$ is p "reflected" in l, which is actually an orientation-preserving rotation around l by 180°. Then, p' is the average point between p and $-\bar{l}pl$, which is the projected point on the line.

5.9 Conclusion on Dual Quaternion Geometric Algebra

We reviewed dual number algebra (DNA), quaternion algebra (QA), and dual quaternion algebra (DQA) in their original notations, and then provided the details on how each algebra is represented in the even-grades subalgebra $\mathcal{G}^+_{3,0,1}$ of PGA. The representations in geometric algebra are called the dual number geometric algebra (DNGA), quaternion geometric algebra (QGA), and dual quaternion geometric algebra (DQGA) to distinguish the different notations and implementations of certain special operations by taking advantage of the larger $\mathcal{G}_{3,0,1}$ and PGA. We have discussed the Dual Quaternion Geometric Algebra $\mathcal{G}^+_{3,0,1}$ in PGA $\mathcal{G}_{3,0,1}$ (DQGA/PGA) in much detail that seemed to be missing in the prior literature that we are aware of, though we acknowledge that dual quaternions are an old subject and many results are probably found somewhere in the published literature. We arrive at our view of dual quaternions by a method that may be new in the literature, by using PGA. We derived identities that convert the CPNS PGA entities $\{\boldsymbol{p}, \boldsymbol{l}, \boldsymbol{\pi}\}$ between their corresponding DQGA entities $\{p, l, \pi\}$ without change of orientation. By using the identities, we also converted CPNS PGA operations on entities into their corresponding operations on the DQGA entities, including rotations, translations, intersections, and projections. Nearly anything that can be done in CPNS PGA can also be done in DQGA, and DQGA is a smaller subalgebra of the full PGA algebra. All of the DQGA entities and operations could be implemented in a pure DQA implementation that could be more efficient than the full PGA algebra. For each of the DQGA entities $\{p, l, \pi\}$, we can rotate, translate, reflect in planes, make intersections between the entities, and make projections onto planes and lines, all in dual quaternions. It may also be possible make rejections from lines and planes, and even more operations from CPNS PGA. The dual

quaternion algebra of DQGA has nice algebraic properties and a dual number-valued tensor (magnitude) operation T that allows any dual quaternion d to be scaled into a unit dual quaternion $\hat{d}$ by a fairly simple and reliable method. We borrow from PGA the entity dualization operation J_e, pseudoscalar $\mathbf{I}_3$, and certain geometric algebra operators (reverse † or ˜, inner product $\cdot$, commutator product $\times$) to implement a complete set of special dual quaternion operations including conjugates (dual number conjugate $\bar{z}$, quaternion conjugate $\mathrm{K}(q) = q^\dagger$, dual conjugate $\bar{d}^\dagger$), dual quaternion part extraction operators (real $\Re$, imaginary $\Im$, scalar S, vector V, point P, plane Π, and line L parts), the dual number-valued tensor T (magnitude) of a dual quaternion, a normalization operation $\hat{d} = \mathrm{U}(d)$ of a dual quaternion d to a unit dual quaternion $\hat{d}$, the vector calculus dot product $\mathbf{a}^* \cdot \mathbf{b}^* = -\mathbf{a}^* \cdot \mathbf{b}^*$ and cross product $\mathbf{a}^* \times \mathbf{b}^* = \mathbf{a}^* \times \mathbf{b}^*$, and an operation Y (using J_e and $\Im$) to extract the quaternion $q_2 = \mathrm{Y}(d)$ from $d = q_1 + q_2\mathbf{I}_4$ ($\varepsilon \hat{=} \mathbf{I}_4$). Using the point P, plane Π, and line L parts operations, we improve the computational efficiency of intersection operations. DQGA is itself a very complete implementation of DQA, but it is also extended in PGA to other algebraic forms of the entities and operations in CPNS PGA and OPNS PGA.

It is possible to convert between the DQGA entities $\{p, l, \pi\}$ and the CPNS PGA entities $\{\boldsymbol{p}, \boldsymbol{l}, \boldsymbol{\pi}\}$ by using simple identities between them. Using the PGA entity dualization operation J_e (which we implemented by what may be a new method), we can convert between CPNS PGA entities $\{\boldsymbol{p}, \boldsymbol{l}, \boldsymbol{\pi}\}$ and OPNS PGA entities $\{\mathbf{P}, \mathbf{L}, \mathbf{\Pi}\}$. Therefore, we can freely convert a point, line, or plane entity into three different forms within PGA $\mathcal{G}_{3,0,1}$ without orientation change, and take advantage of each form of entity and operations in the three different algebras in PGA.

6 Double PGA G(6,0,2) for General Quadrics

This section discusses using two orthogonal copies of PGA $\mathcal{G}_{3,0,1}$, multiplied together as Double PGA (DPGA) $\mathcal{G}_{6,0,2}$ in which general quadrics are represented by methods similar to Double CGA (DCGA) [7]. The doubling of CPNS PGA, called CPNS Double PGA (CPNS DPGA), can represent general quadrics, which can be operated on by all of the operations of CPNS PGA in doubled CPNS DPGA forms. Within DPGA is the subalgebra of Double Dual Quaternion Geometric Algebra (DDQGA) that uses only 64 even-grade basis blades while representing general quadrics with all of the operations available in CPNS PGA doubled and converted through identities as operations in DDQGA. Operations on general quadrics in DDQGA include rotation, translation, reflection in planes, and intersecting with lines and planes.

6.1 Introduction to Double PGA G(6,0,2)

In Conformal Geometric Algebra (CGA) $\mathcal{G}_{4,1}$, it has been shown in a series of papers [6][7][8][9][10] that CGA can be doubled into a Double CGA (DCGA) $\mathcal{G}_{8,2}$ that can represent Darboux cyclides, which includes general quadric surfaces, parabolic cyclides, and Dupin cyclides. There is much similarity between the Plane-based PGA and CGA. Therefore, it is straightforward to apply all of the same doubling methods to the Plane-based PGA. We use two copies of PGA, called PGA1 and PGA2 in $\mathcal{G}_{6,0,2}$, which we call Double PGA (DPGA).

In PGA, we have the Point-based OPNS PGA entities $\mathbf{X}$ and the Plane-based CPNS PGA entities $\boldsymbol{x}$ that are dual to each other through the entity dualization operation $J_e(\mathbf{X}) = \boldsymbol{x}$. In DPGA we have their doubled forms as the Point-based DPGA entities $\mathbf{X}_{:\mathrm{D}}$ and the Plane-based DPGA entities $\boldsymbol{x}_{:\mathrm{D}}$ that are dual to each other through a doubled entity dualization operation $J_{e:\mathrm{D}}(\mathbf{X}_{:\mathrm{D}}) = \boldsymbol{x}_{:\mathrm{D}}$.

All of the entities and operations X of CPNS PGA, including the entity dualization J_e, can be put into a doubled form $X_{:\mathrm{D}}$ in DPGA. While $J_{e:1}$ in PGA1 and $J_{e:2}$ in PGA2 are each anti-involutions, the composed double $J_{e:\mathrm{D}}$ in DPGA becomes an involution. Similar to DCGA, we can form the 2-vector extraction operators (or elements) T_s to extract the 8-blade pseudoscalar $s\mathbf{I}_8$ from the Plane-based DPGA 6-blade point $\boldsymbol{p}_{:\mathrm{D}}$ as $s\mathbf{I}_8 = \boldsymbol{p}_{:\mathrm{D}} \wedge T_s$ by outer product. The possible values $s\mathbf{I}_8$ extracted from DPGA point $\boldsymbol{p}_{:\mathrm{D}}$ are $s \in \{1, x, y, z, x^2, y^2, z^2, xy, yz, zx\}$, which are the quadratic scalar components in the doubled homogeneous point $\boldsymbol{p}_{:\mathrm{D}}$. A linear combination of the extraction elements T_s as $\boldsymbol{\omega} = \sum_s a_s T_s$ defines the Plane-based DPGA 2-vector general quadric entity $\boldsymbol{\omega}$. The PGA1 2-versor rotor $R_{:1}$ is doubled with its copy PGA2 rotor $R_{:2}$ into the DPGA 4-versor rotor $R_{:\mathrm{D}} = R_{:1}R_{:2}$. Similarly, the Plane-based DPGA translator is $T_{:\mathrm{D}} = T_{:1}T_{:2}$. The DPGA 4-versor rotor $R_{:\mathrm{D}}$ and plane-based translator $T_{:\mathrm{D}}$ operate as expected, as versor sandwich operations, on all Plane-based DPGA entities, including the Plane-based DPGA 2-vector general quadric entity $\boldsymbol{\omega}$. In DCGA, there are differential operators $\{D_x, D_y, D_z\}$, which are also available in DPGA in the same forms as in DCGA. The Plane-based DPGA general quadric entity $\boldsymbol{\omega}$ can be reflected in a doubled plane $\boldsymbol{\pi}_{:\mathrm{D}}$, rotated using $R_{:\mathrm{D}}$, translated using $T_{:\mathrm{D}}$, and we can represent its intersection with a doubled plane $\boldsymbol{\pi}_{:\mathrm{D}}$ or line $\boldsymbol{l}_{:\mathrm{D}}$.

DPGA can be compared to many other geometric algebras that can represent general quadrics and DPGA is one of the smaller of these algebras. DPGA $\mathcal{G}_{6,0,2}$ has $n = p + q + r = 6 + 0 + 2 = 8$ vector dimensions and $2^8 = 256$ total basis blades. The geometric algebra $\mathcal{G}_{4,4}$ [12] also has $2^8 = 256$ total basis blades and can represent quadrics using more complicated sandwiching products but also has many operations used in computer graphics. The geometric algebra $\mathcal{G}_{6,3}$ [24] has $2^9 = 512$ total basis blades and can represent point-based general quadrics. DCGA $\mathcal{G}_{8,2}$ [7] has $2^{10} = 1024$ total basis blades and is the most comparable to DPGA. The geometric algebra $\mathcal{G}_{9,6}$ [1] has $2^{15} = 32768$ total basis blades and can represent point-based general quadrics and their intersections. Most of these algebras use only a subset of the complete set of basis blades, such as using only the even-grades subalgebra or some other subset.

6.2 Double PGA G(6,0,2) Basis and Metric

The 1-blade basis for DPGA $\mathcal{G}_{6,0,2}$ is $\{\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6, \mathbf{e}_7\}$ with metric diag(0, 1, 1, 1, 0, 1, 1, 1).

We define PGA on the basis $\{\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ with metric diag(0, 1, 1, 1), which we shall call PGA1. PGA1 is exactly the same as PGA as we have discussed in prior sections of this paper. We introduce the notation $:1$ to mark elements in PGA1. The PGA1 unit pseudoscalar for the subalgebra $\mathcal{G}_{3,0,0:1}$ is $\mathbf{I}_{3:1} = \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$, and the PGA1 4-blade unit pseudoscalar is $\mathbf{I}_{4:1} = \mathbf{e}_0\mathbf{I}_{3:1}$.

We define a copy of PGA on the basis $\{\mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6, \mathbf{e}_7\}$ with metric diag(0, 1, 1, 1), which we shall call PGA2. Every element, entity, and operation in PGA1 is copied to a corresponding element, entity, and operation in PGA2. The PGA2 unit pseudoscalar for the subalgebra $\mathcal{G}_{3,0,0:2}$ is $\mathbf{I}_{3:2} = \mathbf{e}_5\mathbf{e}_6\mathbf{e}_7$, and the PGA2 4-blade unit pseudoscalar is $\mathbf{I}_{4:2} = \mathbf{e}_4\mathbf{I}_{3:2}$.

The geometric product of PGA1 and PGA2 is DPGA. Since PGA1 and PGA2 are completely orthogonal, the geometric product is the outer (wedge) product. The outer product of any PGA1 element $\mathbf{A}_{:1}$ with its corresponding (copy) element $\mathbf{A}_{:2}$ in PGA2 is the DPGA element $\mathbf{A}_{:\mathrm{D}} = \mathbf{A}_{:1}\mathbf{A}_{:2} = \mathbf{A}_{:1} \wedge \mathbf{A}_{:2}$ in the even-grades subalgebra $\mathcal{G}^{+}_{6,0,2}$. The DPGA unit pseudoscalar for $\mathcal{G}_{3,0,0:\mathrm{D}}$ is $\mathbf{I}_{3:\mathrm{D}} = \mathbf{I}_{3:1}\mathbf{I}_{3:2}$. The DPGA unit pseudoscalar is $\mathbf{I}_{4:\mathrm{D}} = \mathbf{e}_0\mathbf{I}_{3:1}\mathbf{e}_4\mathbf{I}_{3:2} = \mathbf{I}_{4:1}\mathbf{I}_{4:2} = \mathbf{I}_8$.

6.3 DPGA Geometric Entities

6.3.1 Plane-based DPGA CPNS 6-blade Point Entity

The Plane-based DPGA CPNS 6-blade point is $\boldsymbol{p}_{\mathbf{t}:\mathrm{D}} = \boldsymbol{p}_{\mathbf{t}:1}\boldsymbol{p}_{\mathbf{t}:2} = (\mathbf{I}_{3:1} - \mathbf{e}_0\mathbf{t}^{*}_{:1})(\mathbf{I}_{3:2} - \mathbf{e}_4\mathbf{t}^{*}_{:2})$. Both points, $\boldsymbol{p}_{\mathbf{t}:1}$ and $\boldsymbol{p}_{\mathbf{t}:2}$, embed the same vector point $\mathbf{t}_{:1} = x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3$, with $\boldsymbol{p}_{\mathbf{t}:2}$ using the corresponding copy vector point $\mathbf{t}_{:2} = x\mathbf{e}_5 + y\mathbf{e}_6 + z\mathbf{e}_7$. The point $\boldsymbol{p}_{\mathbf{t}:\mathrm{D}}$ is the same point as $\boldsymbol{p}_{\mathbf{p}:\mathrm{D}}$ if and only if the 2-vector CPNS test entity is $\boldsymbol{p}_{\mathbf{t}:\mathrm{D}} \times \boldsymbol{p}_{\mathbf{p}:\mathrm{D}} = 0$.

6.3.2 Plane-based DPGA OPNS 2-vector Quadric Elements

In PGA1, we can extract 4-blade pseudoscalars $\{1, x, y, z\}\mathbf{I}_{4:1}$ from $\boldsymbol{p}_{\mathbf{t}:1}$ as $\boldsymbol{p}_{\mathbf{t}:1} \wedge \{-\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. In PGA2, we can extract 4-blade pseudoscalars $\{1, x, y, z\}\mathbf{I}_{4:2}$ from $\boldsymbol{p}_{\mathbf{t}:2}$ as $\boldsymbol{p}_{\mathbf{t}:2} \wedge \{-\mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6, \mathbf{e}_7\}$. In DPGA, we can extract 8-blade pseudoscalars $s\mathbf{I}_{4:\mathrm{D}} \in \{1, x, y, z, x^2, y^2, z^2, xy, yz, zx\}\mathbf{I}_{4:\mathrm{D}}$ from $\boldsymbol{p}_{\mathbf{t}:\mathrm{D}}$ as $s\mathbf{I}_8 = \boldsymbol{p}_{\mathbf{t}:\mathrm{D}} \wedge T_s$ by using the DPGA elements T_s shown in Table 3.

Notice that, the product is not $\times$, but is now $\wedge$ for using T_s. In CPNS PGA, the commutator product $\times$ gives the correct part of the geometric product, and $\boldsymbol{p}_{\mathbf{t}} \times \boldsymbol{\pi} = \boldsymbol{p}_{\mathbf{t}} \wedge \boldsymbol{\pi}$. In DPGA, we have $\boldsymbol{p}_{:\mathrm{D}} \times T_s \neq \boldsymbol{p}_{:\mathrm{D}} \wedge T_s = s\mathbf{I}_8$, and $\boldsymbol{p}_{:\mathrm{D}} \wedge T_s = s\mathbf{I}_8$ is the correct product for extracting $s\mathbf{I}_8$. The T_s are plane-based, and their duals $J_{e:\mathrm{D}}(T_s)$ are point-based. The T_s are OPNS entities in both Plane-based and Point-based DPGA.

$T_1 = \mathbf{e}_4\mathbf{e}_0$	$T_x = \frac{1}{2}(\mathbf{e}_1\mathbf{e}_4 + \mathbf{e}_0\mathbf{e}_5)$	$T_y = \frac{1}{2}(\mathbf{e}_2\mathbf{e}_4 + \mathbf{e}_0\mathbf{e}_6)$	$T_z = \frac{1}{2}(\mathbf{e}_3\mathbf{e}_4 + \mathbf{e}_0\mathbf{e}_7)$	$T_{xy} = \frac{1}{2}(\mathbf{e}_6\mathbf{e}_1 + \mathbf{e}_5\mathbf{e}_2)$
$T_{yz} = \frac{1}{2}(\mathbf{e}_7\mathbf{e}_2 + \mathbf{e}_6\mathbf{e}_3)$	$T_{x^2} = \mathbf{e}_5\mathbf{e}_1$	$T_{y^2} = \mathbf{e}_6\mathbf{e}_2$	$T_{z^2} = \mathbf{e}_7\mathbf{e}_3$	$T_{zx} = \frac{1}{2}(\mathbf{e}_5\mathbf{e}_3 + \mathbf{e}_7\mathbf{e}_1)$

Table 3. Plane-based DPGA OPNS 2-vector quadric elements T_s for extracting $s\mathbf{I}_8 = \boldsymbol{p}_{t:\mathrm{D}} \wedge T_s$.

6.3.3 Plane-based DPGA OPNS 2-vector Quadric Entity

The Plane-based DPGA OPNS 2-vector quadric entity $\boldsymbol{\omega}$ is defined as a linear combination of the Plane-based DPGA OPNS 2-vector quadric elements T_s in Table 3. That is, $\boldsymbol{\omega} = \sum_s a_s T_s$, where the a_s are scalars. The Plane-based DPGA 6-blade point $\boldsymbol{p}_{\mathbf{t}:\mathrm{D}}$ is on the quadric represented by $\boldsymbol{\omega}$ if and only if the 8-blade (pseudoscalar) OPNS entity is $\boldsymbol{p}_{\mathbf{t}:\mathrm{D}} \wedge \boldsymbol{\omega} = 0$. Using the DPGA entity dualization operation $\boldsymbol{\Omega} = J_{e:\mathrm{D}}(\boldsymbol{\omega})$, which is an involution, the Point-based DPGA 2-blade point $\mathbf{P}_{\mathbf{t}:\mathrm{D}} = \mathbf{P}_{\mathbf{t}:1}\mathbf{P}_{\mathbf{t}:2} = J_{e:\mathrm{D}}(\boldsymbol{p}_{\mathbf{t}:\mathrm{D}})$ is on the Point-based DPGA 6-vector quadric $\boldsymbol{\Omega}$ if and only if the 8-blade (pseudoscalar) OPNS entity is $\mathbf{P}_{\mathbf{t}:\mathrm{D}} \wedge \boldsymbol{\Omega} = 0$.

For example, we can form an ellipsoid entity $\boldsymbol{\omega}$ representing the implicit quadric equation

$$\frac{(x - p_x)^2}{r_x^2} + \frac{(y - p_y)^2}{r_y^2} + \frac{(z - p_z)^2}{r_z^2} - 1 = 0 \tag{164}$$

as

$$\begin{aligned}\boldsymbol{\omega} \;=\; & \frac{-2p_x}{r_x^2}T_x + \frac{-2p_y}{r_y^2}T_y + \frac{-2p_z}{r_z^2}T_z + \\ & \frac{1}{r_x^2}T_{x^2} + \frac{1}{r_y^2}T_{y^2} + \frac{1}{r_z^2}T_{z^2} + \\ & \left(\frac{p_x^2}{r_x^2} + \frac{p_y^2}{r_y^2} + \frac{p_z^2}{r_z^2} - 1\right)T_1\end{aligned} \tag{165}$$

All other general quadrics can be formed similarly [6]. The plane-based DPGA quadric entity $\boldsymbol{\omega}$ can be rotated and translated using the DPGA rotor $R_{:\mathrm{D}}$ and plane-based DPGA translator $T_{:\mathrm{D}}$. The point-based quadric entity $\boldsymbol{\Omega} = J_{e:\mathrm{D}}(\boldsymbol{\omega})$ *cannot* be translated using the plane-based translator $T_{:\mathrm{D}}$, but it can be rotated using the DPGA rotor $R_{:\mathrm{D}}$.

The quadric entity $\boldsymbol{\omega}$ is based on the 2-vector elements of Table 3 using the outer product with the Plane-based DPGA 6-blade point $\boldsymbol{p}_{\mathbf{t}:\mathrm{D}}$, therefore $\boldsymbol{\omega}$ is an OPNS entity. Its dual $J_{e:\mathrm{D}}(\boldsymbol{\omega}) = \boldsymbol{\Omega}$ is called the Point-based DPGA OPNS 6-vector quadric entity $\boldsymbol{\Omega}$, which is also an OPNS quadric entity. The OPNS entities are identical, $\boldsymbol{p}_{\mathbf{t}:\mathrm{D}} \wedge \boldsymbol{\omega} = \mathbf{P}_{\mathbf{t}:\mathrm{D}} \wedge \boldsymbol{\Omega}$, including orientation.

Quadric-Plane Intersection: The intersection of the 2-vector quadric $\boldsymbol{\omega}$ and 2-blade plane $\boldsymbol{\pi}_{:\mathrm{D}}$ is the 4-vector $\boldsymbol{c} = \boldsymbol{\pi}_{:\mathrm{D}} \wedge \boldsymbol{\omega}$, which is an entity representing a conic section or quadratic curve. The point $\boldsymbol{p}_{:\mathrm{D}}$ is on $\boldsymbol{c}$ if and only if the grade 6 null space entity is $\langle \boldsymbol{p}_{:\mathrm{D}} \boldsymbol{c} \rangle_6 = 0$. We have to use the grade k part operator $\langle\rangle_k$ since none of the usual part operators $(\cdot, \wedge, \times, \overline{\times})$ give the grade 6 part of the geometric product $\boldsymbol{p}_{:\mathrm{D}}\boldsymbol{c}$, which has a grade 6 null space entity.

Quadric-Line Intersection: The intersection of the 2-vector quadric $\boldsymbol{\omega}$ and 4-blade line $\boldsymbol{l}_{:\mathrm{D}}$ is the 6-vector $\boldsymbol{d} = \boldsymbol{l}_{:\mathrm{D}} \wedge \boldsymbol{\omega}$, which is an entity representing a pair of real or imaginary points, depending on if the intersection exists. The point $\boldsymbol{p}_{:\mathrm{D}}$ is one of the pair of points in $\boldsymbol{d}$ if and only if the grade 4 null space entity is $\langle \boldsymbol{p}_{:\mathrm{D}} \boldsymbol{d} \rangle_4 = 0$. We have to use the grade k part operator $\langle\rangle_k$ since none of the usual part operators $(\cdot, \wedge, \times, \overline{\times})$ give the grade 4 part of the geometric product $\boldsymbol{p}_{:\mathrm{D}}\boldsymbol{d}$, which has a grade 4 null space entity.

6.3.4 Plane-based DPGA OPNS/CPNS Plane and CPNS Line Entities

The CPNS PGA 1-blade $\boldsymbol{\pi}$ and 2-blade line $\boldsymbol{l}$ are doubled into their Plane-based DPGA forms as follows.

Plane: The OPNS/CPNS PGA 1-blade plane entity $\boldsymbol{\pi}$ is doubled in DPGA as the Plane-based DPGA OPNS/CPNS 2-blade plane $\boldsymbol{\pi}_{:\mathrm{D}} = \boldsymbol{\pi}_{:1}\boldsymbol{\pi}_{:2}$. The Plane-based DPGA 6-blade point $\boldsymbol{p}_{:\mathrm{D}}$ is on the Plane-based DPGA OPNS/CPNS 2-blade plane $\boldsymbol{\pi}_{:\mathrm{D}}$ if and only if the 8-blade (pseudoscalar) OPNS entity is $\boldsymbol{p}_{:\mathrm{D}} \wedge \boldsymbol{\pi}_{:\mathrm{D}} = 0$, or if and only if the 6-vector CPNS entity is $\boldsymbol{p}_{:\mathrm{D}} \times \boldsymbol{\pi}_{:\mathrm{D}} = 0$.

For the plane $\boldsymbol{\pi}_{:\mathrm{D}}$, we have two possible null space entities, an 8-blade OPNS entity $\boldsymbol{p}_{:\mathrm{D}} \wedge \boldsymbol{\pi}_{:\mathrm{D}}$ and a 6-vector CPNS entity $\boldsymbol{p}_{:\mathrm{D}} \times \boldsymbol{\pi}_{:\mathrm{D}}$. For the OPNS, we have $\boldsymbol{p}_{:\mathrm{D}} \wedge \boldsymbol{\pi}_{:\mathrm{D}} = \boldsymbol{p}_{:1} \wedge \boldsymbol{p}_{:2} \wedge \boldsymbol{\pi}_{:1} \wedge \boldsymbol{\pi}_{:2} = -(\boldsymbol{p}_{:1} \wedge \boldsymbol{\pi}_{:1}) \wedge (\boldsymbol{p}_{:2} \wedge \boldsymbol{\pi}_{:2})$, where either the 4-blades $\boldsymbol{p}_{:1} \wedge \boldsymbol{\pi}_{:1}$ and $\boldsymbol{p}_{:2} \wedge \boldsymbol{\pi}_{:2}$ are both 0 or both not 0 since they are corresponding copies of the plane. For the CPNS, we use the identity $A \times (BC) = (A \times B)C + B(A \times C)$ and we have (after some algebra) $\boldsymbol{p}_{:\mathrm{D}} \times \boldsymbol{\pi}_{:\mathrm{D}} = (\boldsymbol{p}_{:2} \cdot \boldsymbol{\pi}_{:2})(\boldsymbol{\pi}_{:1} \times \boldsymbol{p}_{:1}) + (\boldsymbol{p}_{:1} \cdot \boldsymbol{\pi}_{:1})(\boldsymbol{\pi}_{:2} \times \boldsymbol{p}_{:2})$, which is a 6-vector, where either the 4-blades $\boldsymbol{\pi}_{:1} \times \boldsymbol{p}_{:1}$ and $\boldsymbol{\pi}_{:2} \times \boldsymbol{p}_{:2}$ are both 0 or both not 0, and $\boldsymbol{p}_{:1} \cdot \boldsymbol{\pi}_{:1}$ and $\boldsymbol{p}_{:2} \cdot \boldsymbol{\pi}_{:2}$ are linearly independent 2-blades.

Line: The CPNS PGA 2-blade line entity $\boldsymbol{l}$ is doubled in DPGA as the Plane-based DPGA CPNS 4-blade line $\boldsymbol{l}_{:\mathrm{D}} = \boldsymbol{l}_{:1}\boldsymbol{l}_{:2}$. The Plane-based DPGA 6-blade point $\boldsymbol{p}_{:\mathrm{D}}$ is on the Plane-based DPGA CPNS 4-blade line $\boldsymbol{l}_{:\mathrm{D}}$ if and only if the 4-vector CPNS entity is $\boldsymbol{p}_{:\mathrm{D}} \times \boldsymbol{\pi}_{:\mathrm{D}} = 0$.

If we expand $\boldsymbol{p}_{:\mathrm{D}} \times \boldsymbol{l}_{:\mathrm{D}}$, we find $\boldsymbol{p}_{:\mathrm{D}} \times \boldsymbol{l}_{:\mathrm{D}} = (\boldsymbol{p}_{:2} \cdot \boldsymbol{l}_{:2})(\boldsymbol{l}_{:1} \times \boldsymbol{p}_{:1}) + (\boldsymbol{p}_{:1} \cdot \boldsymbol{l}_{:1})(\boldsymbol{p}_{:2} \times \boldsymbol{l}_{:2})$, where the 3-vectors $(\boldsymbol{l}_{:1} \times \boldsymbol{p}_{:1})$ and $(\boldsymbol{p}_{:2} \times \boldsymbol{l}_{:2})$ are either both 0 or both not 0, and $(\boldsymbol{l}_{:1} \cdot \boldsymbol{p}_{:1})$ and $(\boldsymbol{l}_{:2} \cdot \boldsymbol{p}_{:2})$ are linearly independent 1-blades.

6.3.5 Point-based DPGA OPNS Geometric Entities

The OPNS PGA 1-blade point $\mathbf{P_t}$, 2-blade $\mathbf{L}$, and 3-blade $\mathbf{\Pi}$ are doubled into their Point-based DPGA forms as follows.

Point: The OPNS PGA 1-blade point $\mathbf{P_t}$ is doubled in DPGA as the Point-based DPGA OPNS 2-blade point $\mathbf{P}_{\mathbf{t}:\mathrm{D}} = \mathbf{P}_{\mathbf{t}:1}\mathbf{P}_{\mathbf{t}:2} = J_{e:\mathrm{D}}(\boldsymbol{p}_{\mathbf{t}:\mathrm{D}})$. The Point-based DPGA 2-blade point $\mathbf{P}_{\mathbf{t}:\mathrm{D}}$ is the same point as $\mathbf{P}_{\mathbf{p}:\mathrm{D}}$ if and only if the 4-vector OPNS entity is $\mathbf{P}_{\mathbf{t}:\mathrm{D}} \wedge \mathbf{P}_{\mathbf{p}:\mathrm{D}} = 0$.

Line: The OPNS PGA 2-blade line $\mathbf{L}$ is doubled in DPGA as the Point-based DPGA OPNS 4-blade line $\mathbf{L}_{:\mathrm{D}} = \mathbf{L}_{:1}\mathbf{L}_{:2} = J_{e:\mathrm{D}}(\boldsymbol{l}_{:\mathrm{D}})$. The Point-based DPGA 2-blade point $\mathbf{P}_{\mathbf{t}:\mathrm{D}}$ is on the Point-based DPGA OPNS 4-blade line $\mathbf{L}_{:\mathrm{D}}$ if and only if the 6-vector OPNS entity is $\mathbf{P}_{\mathbf{t}:\mathrm{D}} \wedge \mathbf{L}_{:\mathrm{D}} = 0$.

Plane: The OPNS PGA 3-blade plane $\mathbf{\Pi}$ is doubled in DPGA as the Point-based DPGA OPNS 6-blade plane $\mathbf{\Pi}_{:\mathrm{D}} = \mathbf{\Pi}_{:1}\mathbf{\Pi}_{:2} = J_{e:\mathrm{D}}(\boldsymbol{\pi}_{:\mathrm{D}})$. The Point-based DPGA 2-blade point $\mathbf{P}_{\mathbf{t}:\mathrm{D}}$ is on the Point-based DPGA OPNS 6-blade plane $\mathbf{\Pi}_{:\mathrm{D}}$ if and only if the 8-vector (pseudoscalar) OPNS entity is $\mathbf{P}_{\mathbf{t}:\mathrm{D}} \wedge \mathbf{\Pi}_{:\mathrm{D}} = 0$.

Quadric: The Plane-based DPGA OPNS 2-vector general quadric entity $\boldsymbol{\omega}$ is dualized as the Point-based DPGA OPNS 6-vector general quadric entity $\mathbf{\Omega} = J_{e:\mathrm{D}}(\boldsymbol{\omega})$. The Point-based DPGA 2-blade point $\mathbf{P}_{\mathbf{t}:\mathrm{D}}$ is on the Point-based DPGA OPNS 6-blade quadric $\mathbf{\Omega}$ if and only if the 8-vector (pseudoscalar) OPNS entity is $\mathbf{P}_{\mathbf{t}:\mathrm{D}} \wedge \mathbf{\Omega} = 0$.

6.4 DPGA Operations

6.4.1 DPGA Entity Dualization Operation

The DPGA entity dualization operation $J_{:\mathrm{D}}$ is a composition of the PGA1 and PGA2 entity dualization operations $J_{e:1}$ and $J_{e:2}$. Since $J_{e:1}$ and $J_{e:2}$ are anti-involutions, their composition as $J_{e:\mathrm{D}}$ is an involution. As an involution, $J_{e:\mathrm{D}}$ dualizes Plane-based and Point-based DPGA entities without any distinction as to signs or orientation of the direction of the dualization operation. The DPGA entity dualization operation $J_{e:\mathrm{D}}$ is its own inverse and we do not require the alias $D_{e:\mathrm{D}} = J_{e:\mathrm{D}}$.

For PGA, we implement J_e in any one of the non-degenerate algebras $\mathcal{G}_{p,q,0} \in \{\mathcal{G}_{4,0,0}, \mathcal{G}_{3,1,0}, \mathcal{G}_{1,3,0}\}$ as we like. For DPGA, we implement $J_{e:\mathrm{D}}$ in any one of the doubled non-degenerate algebras $\mathcal{G}_{p,q,0:\mathrm{D}} \in \{\mathcal{G}_{4,0,0:\mathrm{D}}, \mathcal{G}_{3,1,0:\mathrm{D}}, \mathcal{G}_{1,3,0:\mathrm{D}}\} = \{\mathcal{G}_{8,0,0}, \mathcal{G}_{6,2,0}, \mathcal{G}_{2,6,0}\}$.

In $\mathcal{G}_{p,q,0:\mathrm{D}}$, we implement $J_{e:\mathrm{D}}$ as $\mathbf{A}^\star = J_{e:\mathrm{D}}(\mathbf{A}) = \mathcal{G}_{6,0,2}(\mathcal{J}_{e:\mathrm{D}}(\mathcal{G}_{p,q,0:\mathrm{D}}(\mathbf{A}))) = \mathcal{G}_{6,0,2}(\mathcal{J}_{e:\mathrm{D}}(\boldsymbol{A}))$, where $\mathbf{A}$ is any DPGA entity (Plane-based or Point-based) that is transferred into the non-degenerate doubled algebra $\mathcal{G}_{p,q,0:\mathrm{D}}$ as $\boldsymbol{A} = \mathcal{G}_{p,q,0:\mathrm{D}}(\mathbf{A})$ so that $\mathbf{A} \in \mathcal{G}_{6,0,2}$ and $\boldsymbol{A} \in \mathcal{G}_{p,q,0:\mathrm{D}}$ have the same coordinates on corresponding basis blades. The dualization is performed in $\mathcal{G}_{p,q,0:\mathrm{D}}$ on the corresponding entity $\boldsymbol{A}$ by the entity dualization implementation operation $\mathcal{J}_{e:\mathrm{D}}(\boldsymbol{A})$. The dualized corresponding entity $\mathcal{J}_{e:\mathrm{D}}(\boldsymbol{A})$ is transferred back into the dual entity $\mathbf{A}^\star$ in the degenerate algebra $\mathcal{G}_{6,0,2}$ as $\mathbf{A}^\star = \mathcal{G}_{6,0,2}(\mathcal{J}_{e:\mathrm{D}}(\boldsymbol{A})) = J_{e:\mathrm{D}}(\mathbf{A})$.

The operation $\mathcal{J}_{e:\mathrm{D}}$ in $\mathcal{G}_{p,q,0:\mathrm{D}}$ is a kind of Hodge star $\star$ dualization that is formed as a product of $\boldsymbol{A}$ with other corresponding non-degenerate unit pseudoscalars or elements $\mathcal{G}_{p,q,0}(\{\mathbf{I}_{3:1}, \mathbf{I}_{3:2}, \mathbf{I}_{4:1}, \mathbf{I}_{4:2}, \mathbf{e}_0, \mathbf{e}_4\}) = \{\boldsymbol{I}_{3:1}, \boldsymbol{I}_{3:2}, \boldsymbol{I}_{4:1}, \boldsymbol{I}_{4:2}, \boldsymbol{e}_0, \boldsymbol{e}_4\} \in \mathcal{G}_{p,q,0:\mathrm{D}}$. The DPGA entity dualization implementation operation $\mathcal{J}_{e:\mathrm{D}}(\boldsymbol{A})$ for each algebra $\mathcal{G}_{p,q,0:\mathrm{D}} \in \{\mathcal{G}_{8,0,0}, \mathcal{G}_{6,2,0}, \mathcal{G}_{2,6,0}\}$ is shown in Table 4.

$\mathcal{G}_{p,q,0:\mathrm{D}}$	$\mathcal{G}_{8,0,0}$	$\mathcal{G}_{6,2,0}$	$\mathcal{G}_{2,6,0}$
$\mathcal{J}_{e:\mathrm{D}}$	$\boldsymbol{I}_{3:2}\boldsymbol{I}_{4:2}\boldsymbol{I}_{3:1}\boldsymbol{I}_{4:1}\boldsymbol{A}\boldsymbol{I}_{3:1}\boldsymbol{I}_{3:2} = \boldsymbol{e}_4\boldsymbol{e}_0\boldsymbol{A}\boldsymbol{I}_{3:\mathrm{D}}$	$\boldsymbol{I}_{4:\mathrm{D}}\boldsymbol{A}$	$\boldsymbol{A}\boldsymbol{I}_{4:\mathrm{D}}$

Table 4. DPGA entity dualization $J_{e:\mathrm{D}}$ implementation operation $\mathcal{J}_{e:\mathrm{D}}$ in $\mathcal{G}_{p,q,0:\mathrm{D}}$.

6.4.2 DPGA 4-versor Rotation Operator

The DPGA 4-versor rotation operator (rotor) is defined as $R_{:\mathrm{D}} = R_{:1}R_{:2}$. The rotor $R_{:\mathrm{D}}$ can be applied to any DPGA entity, both the Plane-based and Point-based, by the usual versor sandwiching operation $\boldsymbol{a}' = R_{:\mathrm{D}}\boldsymbol{a}R_{:\mathrm{D}}^{-1}$ for any DPGA geometric entity or element $\boldsymbol{a}$.

6.4.3 Plane-based DPGA 4-versor Translation Operator

The Plane-based DPGA 4-versor translation operator (translator) is defined as $T_{:\mathrm{D}} = T_{:1}T_{:2}$. The translator $T_{:\mathrm{D}}$ is plane-based and can only be applied to the Plane-based DPGA entities $E_{\Pi:\mathrm{D}} = \{\boldsymbol{p}_{:\mathrm{D}}, \boldsymbol{l}_{:\mathrm{D}}, \boldsymbol{\pi}_{:\mathrm{D}}, \boldsymbol{\omega}, R_{:\mathrm{D}}, T_{:\mathrm{D}}\}$ by the usual versor sandwich operation, $\boldsymbol{a}' = T\boldsymbol{a}T^{-1}$ for $\boldsymbol{a} \in E_{\Pi:\mathrm{D}}$. The Point-based DPGA OPNS entities can be translated via dualization by $J_{e:\mathrm{D}}$ to Plane-based entities.

6.4.4 DPGA Differential Operators

The DPGA differential operators are defined as: $D_x = -2T_xT_{x^2}$, $D_y = -2T_yT_{y^2}$, $D_z = -2T_zT_{z^2}$. By commutator product $\times$, the differential operators $\{D_x, D_y, D_z\}$ act on the elements T_s of Table 3 as differential operators as shown in Table 5.

$D \times T_s$	T_1	T_x	T_y	T_z	T_{x^2}	T_{y^2}	T_{z^2}	T_{xy}	T_{yz}	T_{zx}
D_x	0	T_1	0	0	$2T_x$	0	0	T_y	0	T_z
D_y	0	0	T_1	0	0	$2T_y$	0	T_x	T_z	0
D_z	0	0	0	T_1	0	0	$2T_z$	0	T_y	T_x

Table 5. DPGA Differential Operators product table.

General quadrics, represented by the quadric entity $\boldsymbol{\omega}$, can be differentiated as $D \times \boldsymbol{\omega}$ using $D \in \{D_x, D_y, D_z\}$. So, we can take derivatives with respect to x, y, and z.

6.5 Double Dual Quaternion Geometric Algebra

In DPGA, the Dual Quaternion Geometric Algebra (DQGA) is also doubled into Double DQGA (DDQGA) as the product of DQGA1 and DQGA2. In DDQGA, we can form a general quadric entity ω by using identities that relate DDQGA to Plane-based DPGA. The quadric entity ω is the focus of our attention in DDQGA and we see little use for much else in DDQGA.

We will not go into great detail about DDQGA since Plane-based DPGA is likely to be preferred, and the doubling and conversion through the established identities may seem somewhat routine and obvious by now, yielding arguably less intuitive DDQGA expressions of the same geometry as in Plane-based DPGA. We could say much the same in comparing Plane-based PGA to DQGA, but DQGA seems more interesting since it has shown what is possible and less-known in dual quaternions for representing points, lines, and planes.

6.5.1 DDQGA Entities

We can double the DQGA point p, plane π, and line l as the DDQGA entities. We can also convert the Plane-based DPGA quadric $\boldsymbol{\omega}$ to DDQGA form ω.

Point: The DDQGA point is $p_{:\mathrm{D}} = p_{:1}p_{:2} = \boldsymbol{p}_{:1}\mathbf{I}_{3:1}\boldsymbol{p}_{:2}\mathbf{I}_{3:2} = -\boldsymbol{p}_{:\mathrm{D}}\mathbf{I}_{3:\mathrm{D}}$.

Plane: The DDQGA plane is $\pi_{:\mathrm{D}} = \pi_{:1}\pi_{:2} = \mathbf{I}_{3:1}\boldsymbol{\pi}_{:1}\mathbf{I}_{3:2}\boldsymbol{\pi}_{:2} = -\mathbf{I}_{3:\mathrm{D}}\boldsymbol{\pi}_{:\mathrm{D}}$.

Line: The DDQGA line is $l_{:\mathrm{D}} = l_{:1}l_{:2} = \bar{\boldsymbol{l}}_{:1}\bar{\boldsymbol{l}}_{:2} = \mathbf{I}_{3:\mathrm{D}}\boldsymbol{l}_{:\mathrm{D}}\mathbf{I}_{3:\mathrm{D}}^{-1} = \bar{\boldsymbol{l}}_{:\mathrm{D}}$.

Quadric Element: The DDQGA quadric element is $t_s = \mathbf{I}_{3:\mathrm{D}}T_s$, using T_s in Table 3.

Quadric: The DDQGA quadric is $\omega = \sum_s a_s t_s = \mathbf{I}_{3:\mathrm{D}}\boldsymbol{\omega}$, which is a linear combination of the t_s where the a_s are real scalars. The DDQGA point $p_{:\mathrm{D}}$ is on ω if and only if $\mathrm{Y}_{:\mathrm{D}}(p_{:\mathrm{D}} \wedge \omega) = 0$. The null space entity is the 8-blade pseudoscalar part $\Im_{:\mathrm{D}}(p_{:\mathrm{D}} \wedge \omega)$, which is taken and dualized using $\mathrm{Y}_{:\mathrm{D}}$ to obtain a real scalar representing the implicit quadric equation. The DDQGA quadric ω is rotated as $\omega' = R_{:\mathrm{D}}\omega R_{:\mathrm{D}}^{-1} = R_{:\mathrm{D}}\omega R_{:\mathrm{D}}^{\sim}$, where $R_{:\mathrm{D}} = R_{:1}R_{:2}$. The DDQGA quadric w is translated as $\omega' = \bar{p}_{\mathbf{d}:\mathrm{D}}\omega$, where $\bar{p}_{\mathbf{d}:\mathrm{D}} = \mathbf{I}_{3:\mathrm{D}}p_{\mathbf{d}:\mathrm{D}}\mathbf{I}_{3:\mathrm{D}}^{-1} = (1 - \mathbf{d}_{:1}^{*}\mathbf{I}_{4:1})(1 - \mathbf{d}_{:2}^{*}\mathbf{I}_{4:2})$ for translation by $\mathbf{d}^{*}$. Since $\bar{p}_{\mathbf{d}:\mathrm{D}}$ and ω are commutative, the translation can also be $\omega' = \omega\bar{p}_{\mathbf{d}:\mathrm{D}}$. Clearly, $\bar{p}_{\mathbf{d}:\mathrm{D}}$ acts as an offset against any test point so that $p_{\mathbf{t}:\mathrm{D}}\bar{p}_{\mathbf{d}:\mathrm{D}}\omega = p_{\mathbf{t}-\mathbf{d}:\mathrm{D}}\omega$ and the null space is shifted by $\mathbf{d}$ (then we take the grade 8 part). If we want, we can write $\bar{p}_{\mathbf{d}:\mathrm{D}} = \bar{T}_{\mathbf{d}:\mathrm{D}}^{2}$, where $T_{\mathbf{d}:\mathrm{D}}$ is usually the double of $T = 1 + \mathbf{d}^{*}\mathbf{I}_4/2$.

6.5.2 DDQGA Operations

We can compose some of the operations of DQGA1 and DQGA2 as DDQGA operations.

The DDQGA complex conjugate is composed as $\bar{d}_{:\mathrm{D}} = \mathbf{I}_{3:2}\mathbf{I}_{3:1}d_{:\mathrm{D}}\mathbf{I}_{3:1}^{-1}\mathbf{I}_{3:2}^{-1} = \mathbf{I}_{3:\mathrm{D}}d_{:\mathrm{D}}\mathbf{I}_{3:\mathrm{D}}^{-1}$. The operators based on the complex conjugate also compose, so we have the real part $\Re_{:\mathrm{D}}(d_{:\mathrm{D}}) = \Re_{:1}(\Re_{:2}(d_{:\mathrm{D}}))$, the imaginary part $\Im_{:\mathrm{D}}(d_{:\mathrm{D}}) = \Im_{:2}(\Im_{:1}(d_{:\mathrm{D}}))$, and $\mathrm{Y}_{:\mathrm{D}}(d_{:\mathrm{D}}) = \mathrm{Y}_{:1}(\mathrm{Y}_{:2}(d_{:\mathrm{D}})) = J_{e:\mathrm{D}}(\Im_{:\mathrm{D}}(d_{:\mathrm{D}}))$ to take the real component from the imaginary part.

The DQGA quaternion conjugate is implemented using the reverse $d^{\dagger}$, so we just use the usual scalar part $\mathrm{S}(d_{:\mathrm{D}})$ and vector part $\mathrm{V}(d_{:\mathrm{D}})$ operators that are based on the quaternion conjugate. We do not need special doubled forms for S and V.

7 Conclusion

In Section 1, we introduced the subject of this paper, which is about the geometric algebra PGA $\mathcal{G}_{3,0,1}$ for points, lines and planes, and its double DPGA $\mathcal{G}_{6,0,2}$ for general quadrics. We discussed the contributions of this paper, which are mainly the entity dualization operation J_e, the detailed development of the Dual Quaternion Geometric Algebra (DQGA), and the doubling to DPGA $\mathcal{G}_{6,0,2}$ including Double DQGA (DDQGA), which provide general quadric entities. The overall detailed exposition of PGA that has been provided in this paper could also be seen as a contribution into the literature on this subject.

In Section 2, we discussed the Point-based algebra of OPNS entities in PGA, which we call OPNS PGA. The point-based algebra allows points to join (span) by outer product into lines and planes.

In Section 3, we discussed the Plane-based algebra of CPNS entities in PGA, which we call CPNS PGA. The plane-based algebra allows planes to meet (intersect) by outer product into lines and points.

In Section 4, we developed the new PGA entity dualization operation J_e that dualizes the OPNS PGA entities into CPNS entities. The dualization operation seems to have been a difficult problem in the prior literature, and the new operation J_e appears to contribute a solution to the dualization problem for PGA $\mathcal{G}_{3,0,1}$.

In Section 5, we explored the details of the Dual Quaternion Geometric Algebra (DQGA) within the even-grades subalgebra $\mathcal{G}^+_{3,0,1}$ of PGA. In DQGA, we rediscovered many results that may be known in older published literature, while we may have contributed some new results on representing lines and planes and various operations on them that are derived through identities to the CPNS PGA entities and operations.

In Section 6, we discussed Double PGA $\mathcal{G}_{6,0,2}$ (DPGA) in which the main contribution is the ability to form a general quadric entity as the bivector $\boldsymbol{\omega}$. Within DPGA, the DQGA is also doubled into Double DQGA (DDQGA), in which again the main contribution is a general quadric entity ω based closely on $\boldsymbol{\omega}$, as $\omega = \mathbf{I}_{3:\mathrm{D}}\boldsymbol{\omega}$.

The Projective Geometric Algebra (PGA) $\mathcal{G}_{3,0,1}$, also more recently called the Point-based and Plane-based Geometric Algebras (PGA), is a degenerate-metric geometric algebra, which makes it more difficult to understand and use than a non-degenerate algebra such as Conformal Geometric Algebra (CGA) $\mathcal{G}_{4,1}$. Although based on a small 4D space, it is not a simple algebra to fully comprehend.

The entity dualization operation J_e required very careful formation of the entities and observation of their duals in same orientations to see how each basis blade should dualize to maintain geometric content and orientation. At first, the duals could only be collected in table form, for lookup. The fact that it is an anti-involution gave a big hint that the operation J_e should be implementable as a dualization in a non-degenerate algebra with unit pseudoscalar $\mathbf{I}_4$ where $\mathbf{I}_4^2 = -1$ (as a kind of Hodge dualization). What seems like errors were found in some of the referenced literature on the Hodge star dualization, and these errors had to be understood and corrected to finally make use of the Hodge star dualization theory. As discussed, three algebras were found in which J_e could be implemented, which seemed to further validate the table of empirical dualizations that initially defined J_e, since we searched for the non-degenerate dualization algebras that would match the table for J_e. We did not initially assume J_e could be matched by any kind of known dualization or algebra, but it seems to have turned out simple enough and followed a basic idea that the first author had several years ago about how the dualization for $\mathcal{G}_{3,0,1}$ probably could be done.

The details of the Dual Quaternion Geometric Algebra may contribute to the literature on Dual Quaternions, showing that much more can be done with dual quaternions than seems to be commonly known. Some old literature could turn up with many of the results, but we are not aware of it. The dual quaternions are probably, for the most part, a curiosity that is superseded by the Plane-based algebra of PGA that has a nicer and simpler form. There is also the point-based algebra of PGA through the dualization J_e that offers the ability to join points, which we did not find in dual quaternions. Still, for those who are wanting to try to get the most out of an efficient dual quaternion implementation, the DQGA entities and operations for points, lines, and planes may be of interest.

In the doubling of PGA $\mathcal{G}_{3,0,1}$ as DPGA $\mathcal{G}_{6,0,2}$, we have a new quadric entity $\boldsymbol{\omega}$, and also its other form in the Double DQGA (DDQGA) $\omega = \mathbf{I}_{3:\mathrm{D}}\boldsymbol{\omega}$. We are unsure how useful this quadric entity will be in applications, but it may be faster to compute than the similar quadric entity in Double CGA (DCGA) since there are fewer basis blades to compute in DPGA. In any case, it contributes another representation of quadrics into the literature that admits rotation and translation versors, and intersections with lines and planes.

The difficulty of fully comprehending all aspects of PGA $\mathcal{G}_{3,0,1}$ and DPGA $\mathcal{G}_{6,0,2}$ probably leaves much more open for further research, but we hope this paper provides a guide to begin using these algebras more practically.

References

[1] Stéphane Breuils, Vincent Nozick, Akihiro Sugimoto, and Eckhard Hitzer. Quadric Conformal Geometric Algebra of G(9,6). *Advances in Applied Clifford Algebras*, 28(2):35, Mar 2018.

[2] Alan Bromborsky. *Geometric Algebra Module for Sympy*. 2016.

[3] Alan Bromborsky, Utensil Song, Eric Wieser, Hugo Hadfield, and The Pygae Team. Pygae/galgebra: v0.5.0. June 2020.

[4] Eduardo Bayro Corrochano and Joan Lasenby. The geometry algebra of computer vision. In Eduardo Bayro Corrochano and Garret Sobczyk, editors, *Geometric Algebra with Applications in Science and Engineering*, pages 123–146. Birkhäuser Boston, Boston, MA, 2001.

[5] L. Dorst, D. Fontijne, and S. Mann. *Geometric Algebra for Computer Science (Revised Edition): An Object-Oriented Approach to Geometry*. The Morgan Kaufmann Series in Computer Graphics. Elsevier Science, 2009.

[6] Robert Benjamin Easter. G8,2 Geometric Algebra, DCGA. https://vixra.org/abs/1508.0086, 2015. Accessed: 2015-10-01.

[7] Robert Benjamin Easter and Eckhard Hitzer. Double Conformal Geometric Algebra. *Advances in Applied Clifford Algebras*, 27(3):2175–2199, 2017. DOI: 10.1007/s00006-017-0784-0. Preprint: vixra.org/abs/1705.0019.

[8] Robert Benjamin Easter and Eckhard Hitzer. Double Conformal Space-Time Algebra. *AIP Conference Proceedings*, 1798(1):20066, 2017. DOI: 10.1063/1.4972658.

[9] Robert Benjamin Easter and Eckhard Hitzer. Triple conformal geometric algebra for cubic plane curves. *Mathematical Methods in the Applied Sciences*, pages 1–15, 2017. DOI: 10.1002/mma.4597.

[10] Robert Benjamin Easter and Eckhard Hitzer. Conic and cyclidic sections in double conformal geometric algebra g8,2 with computing and visualization using gaalop. *Mathematical Methods in the Applied Sciences*, 43(1):334–357, 2020.

[11] Harley Flanders. *Differential Forms with Applications to the Physical Sciences*. Dover Books on Advanced Mathematics. Academic Press, 1963.

[12] Ron Goldman and Stephen Mann. R(4,4) As a Computational Framework for 3-Dimensional Computer Graphics. *Advances in Applied Clifford Algebras*, 25(1):113–149, 2015.

[13] Charles Gunn. *Geometry, Kinematics, and Rigid Body Mechanics in Cayley-Klein Geometries*. PhD thesis, Technical University Berlin, 2011.

[14] Charles Gunn. On the homogeneous model of euclidean geometry. In Leo Dorst and Joan Lasenby, editors, *Guide to Geometric Algebra in Practice*, chapter 15, pages 297–327. Springer, 2011.

[15] Charles Gunn. Doing euclidean plane geometry using projective geometric algebra. *Advances in Applied Clifford Algebras*, 27(2):1203–1232, 2017.

[16] Charles Gunn. Geometric algebras for euclidean geometry. *Advances in Applied Clifford Algebras*, 27(1):185–208, 2017.

[17] Charles G. Gunn. A bit better: Variants of duality in geometric algebras with degenerate metrics (v4, Oct 19, 2022). https://arxiv.org/abs/2206.02459. Accessed: 2023-12-04.

[18] David Hestenes. *Space-Time Algebra*. Springer, Second edition, 2015.

[19] David Hestenes and Garret Sobczyk. *Clifford Algebra to Geometric Calculus, A Unified Language for Mathematics and Physics*, volume 5 of *Fundamental Theories of Physics*. Dordrecht-Boston-Lancaster: D. Reidel Publishing Company, a Member of the Kluwer Academic Publishers Group, 1984.

[20] Leo Dorst. PGA4CS: A Guided Tour to the Plane-Based Geometric Algebra PGA, Version 1.15 – July 6, 2020. https://bivector.net/PGA4CS.html. Accessed: 2020-10-02.

[21] Leo Dorst and Steven De Keninck. PGA4CS: A Guided Tour to the Plane-Based Geometric Algebra PGA, Version 2.0 – March 14, 2022. https://bivector.net/PGA4CS.html. Accessed: 2023-11-16.

[22] Christian Perwass. *Geometric Algebra with Applications in Engineering*, volume 4 of *Geometry and Computing*. Springer, 2009. Habilitation thesis, Christian-Albrechts-Universität zu Kiel.

[23] Jonathan Selig. Clifford algebra of points, lines and planes. *Robotica*, 18:545–556, 09 2000.

[24] Julio Zamora-Esquivel. G6,3 Geometric Algebra; Description and Implementation. *Advances in Applied Clifford Algebras*, 24(2):493–514, 2014.

Appendix A GAlgebra for SymPy Python Code

In this appendix, we provide example code for using $\mathcal{G}$Algebra [2] for *SymPy* to run computations in PGA $\mathcal{G}_{3,0,1}$ and its double, DPGA $\mathcal{G}_{6,0,2}$ for quadrics. This code was run using the *Anaconda* python software distribution and its `jupyter-notebook` application, which allows the code blocks to be easily edited and executed in cells interactively.

The entity dualization operation is included. Many entities and operations are defined for running the Dual Quaternion Geometric Algebra for points, lines, and planes. Using formulas from the paper, more computations can entered and run. Much of this code was used during the research to test various computations. The symbolic point entities can be multiplied into the line, plane, and quadric entities to obtain symbolic products showing null space entities and their null space conditions or implicit surface equations.

A.1 Packages

```
from sympy import *
from sympy.printing import *
from galgebra.ga import *
from galgebra.mv import *
from galgebra.lt import *
from galgebra.metric import *
from galgebra.printer import *
```

A.2 Doubled Algebras

```
# DPGA G(6,0,2) = G(3,0,1) x G(3,0,1)
g602 = Ga('e*0|1|2|3|4|5|6|7', g=[ 0, 1, 1, 1, 0, 1, 1, 1])

# G(6,2,0) for Entity Dualization
g620 = Ga('e*0|1|2|3|4|5|6|7', g=[-1, 1, 1, 1,-1, 1, 1, 1])

# G(2,6,0) for Entity Dualization
g260 = Ga('e*0|1|2|3|4|5|6|7', g=[ 1,-1,-1,-1, 1,-1,-1,-1])

# G(8,0,0) for Entity Dualization
g800 = Ga('e*0|1|2|3|4|5|6|7', g=[ 1, 1, 1, 1, 1, 1, 1, 1])
```

A.3 Symbols

```
(e0,e1,e2,e3,e4,e5,e6,e7) = g602.mv()
(V,v,x,y,z,t,d,nx,ny,nz) = symbols('V v x y z t d n_x n_y n_z')
(px,py,pz,nx,ny,nz) = symbols('p_x p_y p_z n_x n_y n_z')

(I31,I32,I41,I42,I8) = symbols('I31 I41 I32 I42 I8')
(I3D,I4D) = symbols('I3D I4D')

(px1,py1,pz1) = symbols('p_x1 p_y1 p_z1')
(px2,py2,pz2) = symbols('p_x2 p_y2 p_z2')
(px3,py3,pz3) = symbols('p_x3 p_y3 p_z3')
```

A.4 Unit Pseudoscalars

```
I31 = e1^e2^e3; I41 = e0^I31;
I32 = e5^e6^e7; I42 = e4^I32;
I8  = I41^I42; I3D = I31^I32; I4D = I8
```

A.5 Symbolic Vectors

```
v = x*e1 + y*e2 + z*e3; v1 = v; v2 = x*e5 + y*e6 + z*e7
p = px*e1 + py*e2 + pz*e3; n = nx*e1 + ny*e2 + nz*e3
```

A.6 Useful Functions

```
def Normalize(v):
    """
    Normalize vector v.
    """
    return v*Pow( sqrt(scalar(v|v)), -1)

def d2r(d):
    """
    Convert degrees to radians.
    """
    return (pi/180)*d

def v1v2(v1):
    """
    Convert vector v1 in PGA1 to v2 in PGA2.
    """
    return (v1|e1)*e5 + (v1|e2)*e6 + (v1|e3)*e7
```

A.7 Rotation Operators

```
def Rotor1(n,d):
```

```
    """
    PGA1 Rotor.
    n is axis of rotation
    d is angle in degrees
    """
    hr = d2r(d)/2
    N = -Normalize(n)*I31
    return cos(hr) + sin(hr)*N

def Rotor2(n,d):
    """
    PGA2 Rotor.
    n is axis of rotation (in PGA1, then we convert)
    d is angle in degrees
    """
    n2 = v1v2(n)
    hr = d2r(d)/2
    N = -Normalize(n2)*I32
    return cos(hr) + sin(hr)*N

def RotorD(n,d):
    """
    DPGA Rotor.
    n is axis of rotation (in PGA1)
    d is angle in degrees.
    """
    return Rotor1(n,d)*Rotor2(n,d)
```

A.8 Plane-based PGA Translation Operators

```
def Translator1(t):
    """
    PGA1 Translator.
    It is just a dual quaternion point for t/2.
    """
    return ( 1 + I41*(t/I31)/2 )

def Translator2(t):
    """
    PGA2 Translator. Give t as 3D vector in PGA1.
    """
    t2 = v1v2(t)
    return ( 1 + I42*(t2/I32)/2 )

def TranslatorD(t):
    """
    DPGA translator. Give t as 3D vector in PGA1.
    """
```

```
    return Translator1(t)^Translator2(t)
```

A.9 Point-based PGA Point Entities

```
def OPNS_Point1(p):
    """
    PGA1 point. Give p as a PGA1 3D vector point.
    """
    return e0 + p

def OPNS_Point2(p):
    """
    PGA2 point. Give p as a PGA1 3D vector point.
    """
    return e4 + v1v2(p)

def OPNS_PointD(p):
    """
    DPGA point. Give p as a PGA1 3D vector point.
    """
    return OPNS_Point1(p)^OPNS_Point2(p)
```

A.10 Point-based PGA Symbolic Points

```
OV1 = OPNS_Point1(v)
OV2 = OPNS_Point2(v)
OVD = OPNS_PointD(v)
```

A.11 Point-based PGA Line Entity

```
def OPNS_Line1(p,d):
    """
    OPNS PGA1 Line formed using point p and direction d,
    both given as PGA1 3D vectors.
    """
    return Normalize(d)^(e0+p)

def OPNS_Line2(p,d):
    """
    OPNS PGA2 Line formed using point p and direction d,
    both given as PGA1 3D vectors.
    """
    return Normalize(v1v2(d))^(e4+v1v2(p))

def OPNS_LineD(p,d):
    """
    OPNS DPGA Line formed using point p and direction d,
    both given as PGA1 3D vectors.
```

```
    """
    return OPNS_Line1(p,d)^OPNS_Line2(p,d)
```

A.12 Point-based PGA Plane Entity

```
def OPNS_Plane1(p,n):
    """
    OPNS PGA1 Plane formed using point p and normal n,
    both given as PGA1 3D vectors.
    """
    return e0*(Normalize(n)/I31) - (p|Normalize(n))*I31

def OPNS_Plane2(p,n):
    """
    OPNS PGA2 Plane formed using point p and normal n,
    both given as PGA1 3D vectors.
    """
    p2 = v1v2(p)
    n2 = v1v2(n)
    return e4*(Normalize(n2)/I32) - (p2|Normalize(n2))*I32

def OPNS_PlaneD(p,n):
    """
    OPNS DPGA Plane formed using point p and normal n,
    both given as PGA1 3D vectors.
    """
    return OPNS_Plane1(p,n)^OPNS_Plane2(p,n)
```

A.13 PGA Entity Dualization Operation

```
def J1(A):
    """
    PGA1 Entity Dualization J1(A).
    This assumes that A is in PGA1.
    Dualizes A from point-based to plane-based PGA1.
    This dualization is an anti-involution.
    Use -J1(A) to dualize A from plane-based to point-based.
    We can use g620, g260, or g800 as explained in the paper.
    We will use g260.
    """
    EA = g260.mv(A)
    EI41 = g260.mv(I41)
    return g602.mv(EA*EI41)

def J2(A):
    """
    PGA2 Entity Dualization J2(A).
    This assumes that A is in PGA2.
```

```
    Dualizes A from point-based to plane-based PGA2.
    This dualization is an anti-involution.
    Use -J2(A) to dualize A from plane-based to point-based.
    We can use g620, g260, or g800 as explained in the paper.
    We will use g260.
    """
    EA = g260.mv(A)
    EI42 = g260.mv(I42)
    return g602.mv(EA*EI42)

def JD(A):
    """
    DPGA Entity Dualization JD(A).
    This assumes that A is in DPGA as a doubled entity or quadric.
    Dualizes A between point-based and plane-based DPGA.
    This dualization is an involution.
    We can use g620, g260, or g800 as explained in the paper.
    We will use g260.
    """
    EA = g260.mv(A)
    EI4D = g260.mv(I4D)
    return g602.mv(EA*EI4D)

def D1(A):
    """
    J1 has orientation to dualize OPNS PGA to CPNS PGA,
    with -J1 acting as the undual operation.
    To reduce confusion, define D1 = -J1 so that D1 has the
    orientation to dualize CPNS PGA to OPNS PGA,
    with -D1 acting as the undual operation.
    """
    return -J1(A)

def D2(A):
    return -J2(A)
```

A.14 Plane-based PGA Point Entity

```
def CPNS_Point1(p):
    """
    PGA1 CPNS plane-based point.
    p is a PGA1 3D vector to be embedded.
    """
    return I31 + I41*p

def CPNS_Point2(p):
    """
    PGA2 CPNS plane-based point.
    p is a PGA1 vector (it is converted to PGA2)
```

```
    """
    return I32 + I42*v1v2(p)

def CPNS_PointD(p):
    """
    DPGA CPNS/OPNS plane-based point.
    p is a PGA1 3D vector to be embedded.
    """
    return CPNS_Point1(p)^CPNS_Point2(p)
```

A.15 Plane-based PGA Symbolic Points

```
CV1 = CPNS_Point1(v)
CV2 = CPNS_Point2(v)
CVD = CPNS_PointD(v)
```

A.16 Plane-based PGA Plane Entity

```
def CPNS_Plane1(p,n):
    """
    PGA1 Plane-based CPNS plane entity, where
    n is normal vector to plane, and
    p is any point on the plane, and
    both given as PGA1 3D vectors.
    """
    n1 = Normalize(n)
    d = p|n1
    return n1 + d*e0

def CPNS_Plane2(p,n):
    """
    PGA2 Plane-based CPNS plane entity, where
    n is normal vector to plane, and
    p is any point on the plane, and
    both given as PGA1 3D vectors.
    """
    p2 = v1v2(p)
    n2 = Normalize(v1v2(n))
    d = p2|n2
    return n2 + d*e4

def CPNS_PlaneD(p,n):
    """
    DPGA Plane-based CPNS plane entity, where
    n is normal vector to plane, and
    p is any point on the plane, and
    both given as PGA1 3D vectors.
    """
```

```
    return CPNS_Plane1(p,n)^CPNS_Plane2(p,n)
```

A.17 Plane-based PGA Line Entity

```
def CPNS_Line1(p,d):
    """
    PGA1 Plane-based CPNS Line Entity, where
    p is any point on the line, and
    d is the direction of the line, and
    both are given as PGA1 3D vectors.
    It has the same form as the CGA IPNS line.
    """
    D = Normalize(d)/I31
    return D - (p|D)*e0

def CPNS_Line2(p,d):
    """
    PGA2 Plane-based CPNS Line Entity, where
    p is any point on the line, and
    d is the direction of the line, and
    both are given as PGA1 3D vectors.
    """
    p2 = v1v2(p)
    d2 = v1v2(d)
    D2 = Normalize(d2)/I32
    return D2 - (p2|D2)*e4

def CPNS_LineD(p,d):
    """
    DPGA Plane-based CPNS Line Entity, where
    p is any point on the line, and
    d is the direction of the line, and
    both are given as PGA1 3D vectors.
    """
    return CPNS_Line1(p,d)^CPNS_Line2(p,d)
```

A.18 Dual Quaternion Point Entity

```
def DQ_Point1(p):
    """
    PGA1 Dual Quaternion point: 1 + I41*(p/I31) = 1 + e0^p.
    p is the PGA1 3D vector point to be embedded.
    """
    return 1 + (e0^p)

def DQ_Point2(p):
    """
    PGA1 Dual Quaternion point: 1 + I41*(p/I31) = 1 + e0^p.
```

```
    p is the PGA1 3D vector point to be embedded.
    """
    return 1 + (e4^v1v2(p))

def DQ_PointD(p):
    """
    PGA1 Dual Quaternion point: 1 + I41*(p/I31) = 1 + e0^p.
    p is the PGA1 3D vector point to be embedded.
    """
    return DQ_Point1(p)^DQ_Point2(p)
```

A.19 Dual Quaternion Symbolic Point Entity

```
DQV1 = DQ_Point1(v)
DQV2 = DQ_Point2(v)
DQVD = DQ_PointD(v)
```

A.20 Dual Quaternion Symbolic Dual Quaternions

```
# DQGA1/PGA1 symbolic dual quaternion Q1
(sr1,xr1,yr1,zr1,si1,xi1,yi1,zi1,Q1) =
symbols('s_r1 x_r1 y_r1 z_r1 s_i1 x_i1 y_i1 z_i1 Q1')
Q1 = (sr1 + (xr1*e1+yr1*e2+zr1*e3)/I31) + I41*(si1 +
(xi1*e1+yi1*e2+zi1*e3)/I31)

# DQGA2/PGA2 symbolic dual quaternion Q2
(sr2,xr2,yr2,zr2,si2,xi2,yi2,zi2,Q2) =
symbols('s_r2 x_r2 y_r2 z_r2 s_i2 x_i2 y_i2 z_i2 Q2')
Q2 = (sr2 + (xr2*e5+yr2*e6+zr2*e7)/I32) + I42*(si2 +
(xi2*e5+yi2*e6+zi2*e7)/I32)
```

A.21 Dual Quaternion Operators

```
def DQ_CC1(q):
    """
    DQGA1/PGA1 Dual Number complex conjugate of dual quaternion
    q = q1 + I41*q2, where q1 and q2 are quaternions
    of form s + v/I31, a sum of scalar s and quaternion
    vector (bivector) v/I31. I41 is the null pseudoscalar
    that acts as the nilpotent scalar varepsilon e.

    q         = (s1 + v1/I31) + I41*(s2 + v2/I31)
    DQ_CC1(q) = (s1 + v1/I31) - I41*(s2 + v2/I31)
    """
    return I31*q*-I31

def DQ_CC2(q):
    """
```

```
        DQGA2/PGA2 Dual Number complex conjugate.
        """
        return I32*q*-I32

    def DQ_CCD(q):
        """
        DDQGA/DPGA Dual Number complex conjugate.
        """
        return DQ_CC1(DQ_CC2(q))

    def DQ_QC1(q):
        """
        DQGA1/PGA1 quaternion conjugate of dual quaternion
        q          = (s1 + v1/I31) + I41*(s2 + v2/I31)
        DQ_QC1(q) = (s1 - v1/I31) + I41*(s2 - v2/I31)
        """
        return q.rev()

    def DQ_QC2(q):
        """
        DQGA2/PGA2 quaternion conjugate of dual quaternion.
        """
        return q.rev()

    def DQ_QCD(q):
        """
        DDQGA/DPGA quaternion conjugate of dual quaternion.
        """
        return q.rev()

    def DQ_DC1(q):
        """
        DQGA1/PGA1 dual conjugate of q,
        takes both complex and quaternion conjugates,
        DQ_CC1(DQ_QC1(q)) or DQ_QC1(DQ_CC1(q)).
        """
        return DQ_QC1(DQ_CC1(q))

    def DQ_DC2(q):
        """
        DQGA2/PGA2 dual conjugate of q,
        takes both complex and quaternion conjugates.
        """
        return DQ_QC2(DQ_CC2(q))

    def DQ_DCD(q):
        """
        DDQGA/DPGA dual conjugate of q,
        takes both complex and quaternion conjugates.
        """
```

```
        return DQ_DC1(DQ_DC2(q))

    def DQ_RE1(q):
        """
        DQGA1/PGA1 (RE)al quaternion part.
        """
        return (q + DQ_CC1(q))/2

    def DQ_RE2(q):
        """
        DQGA2/PGA2 (RE)al quaternion part.
        """
        return (q + DQ_CC2(q))/2

    def DQ_RED(q):
        """
        DDQGA/DPGA (RE)al quaternion part.
        """
        return DQ_RE1(DQ_RE2(q))

    def DQ_IM1(q):
        """
        DQGA1/PGA1 (IM)aginary quaternion part.
        This keeps the imaginary unit on the part, not just
        the real quaternion of the part.
        """
        return (q - DQ_CC1(q))/2

    def DQ_IM2(q):
        """
        DQGA2/PGA2 (IM)aginary quaternion part.
        """
        return (q - DQ_CC2(q))/2

    def DQ_IMD(q):
        """
        DDQGA/DPGA (IM)aginary quaternion part.
        """
        return DQ_IM1(DQ_IM2(q))

    def DQ_Y1(q):
        """
        DQGA1/PGA1 real component of imaginary.
        Return y from dual number z = x + y*I41.
        Return q2 from dual quaternion d = q1 + q2*I41.
        In any case, it returns the value off of I41.
        """
        return -J1(DQ_IM1(q))

    def DQ_Y2(q):
```

```
        """
        DQGA2/PGA2 real component of imaginary.
        Return y from dual number z = x + y*I42.
        Return q2 from dual quaternion d = q1 + q2*I42.
        In any case, it returns the value off of I42.
        """
        return -J2(DQ_IM2(q))

    def DQ_YD(q):
        """
        DDQGA/DPGA2 real component of imaginary.
        """
        return DQ_Y1(DQ_Y2(q))

    def DQ_S1(q):
        """
        DQGA1/PGA1 scalar part of dual quaternion q.
        The scalars are dual numbers, not real numbers.
        This has to be followed by DQ_RE1 or DQ_IM1
        for the real or imaginary dual number part.
        """
        return (q + DQ_QC1(q))/2

    def DQ_S2(q):
        """
        DQGA2/PGA2 scalar part of dual quaternion q.
        The scalars are dual numbers, not real numbers.
        This has to be followed by DQ_RE2 or DQ_IM2
        for the real or imaginary dual number part.
        """
        return (q + DQ_QC2(q))/2

    def DQ_SD(q):
        """
        DDQGA/DPGA scalar part of double dual quaternion q.
        The scalars are doubled dual numbers, not real numbers.
        This has to be followed by DQ_RED or DQ_IMD
        for the real or imaginary doubled dual number part.
        Just use DQ_S1 or DQ_S2. No need to compose.
        They just use the reverse.
        """
        return DQ_S1(q)

    def DQ_V1(q):
        """
        DQGA1/PGA1 vector part of dual quaternion q.
        This has to be followed by DQ_RE1 or DQ_IM1
        for the real or imaginary vector part.
        """
        return (q - DQ_QC1(q))/2
```

```
def DQ_V2(q):
    """
    DQGA2/PGA2 vector part of dual quaternion q.
    This has to be followed by DQ_RE2 or DQ_IM2
    for the real or imaginary vector part.
    """
    return (q - DQ_QC2(q))/2

def DQ_VD(q):
    """
    DDQGA/DPGA vector part of dual quaternion q.
    This has to be followed by DQ_RED or DQ_IMD
    for the real or imaginary vector part.
    Just use DQ_V1 or DQ_V2. No need to compose.
    They just use the reverse.
    """
    return DQ_V1(q)

def DQ_T1(d):
    """
    DQGA1/PGA1 tensor (dual number-valued) of dual quaternion d.
    """
    mQ1 = sqrt(scalar(DQ_RE1(d)*DQ_QC1(DQ_RE1(d))))
    mQ1_2 = Pow(mQ1,-2)
    dotq1q2 = -DQ_V1(DQ_RE1(d))|DQ_Y1(DQ_V1(DQ_IM1(d)))
    q1wq2w = DQ_S1(DQ_RE1(d))*DQ_Y1(DQ_S1(DQ_IM1(d)))
    return mQ1*(1+((q1wq2w+dotq1q2)*mQ1_2)*I41)

def DQ_T2(d):
    """
    DQGA2/PGA2 tensor (dual number-valued) of dual quaternion d.
    """
    mQ1 = sqrt(scalar(DQ_RE2(d)*DQ_QC2(DQ_RE2(d))))
    mQ1_2 = Pow(mQ1,-2)
    dotq1q2 = -DQ_V2(DQ_RE2(d))|DQ_Y2(DQ_V2(DQ_IM2(d)))
    q1wq2w = DQ_S2(DQ_RE2(d))*DQ_Y2(DQ_S2(DQ_IM2(d)))
    return mQ1*(1+((q1wq2w+dotq1q2)*mQ1_2)*I42)

def DQ_TINV1(d):
    """DQGA1/PGA1
    Inverse of Tensor (dual number-valued) of d.
    Troublesome for software to invert T, so we have
    to formulate it special.
    """
    mQ1 = sqrt(scalar(DQ_RE1(d)*DQ_QC1(DQ_RE1(d))))
    mQ1_2 = Pow(mQ1,-2)
    dotq1q2 = -DQ_V1(DQ_RE1(d))|DQ_Y1(DQ_V1(DQ_IM1(d)))
    q1wq2w = DQ_S1(DQ_RE1(d))*DQ_Y1(DQ_S1(DQ_IM1(d)))
    return Pow(mQ1,-1)*(1-((q1wq2w+dotq1q2)*mQ1_2)*I41)
```

```
    def DQ_TINV2(d):
        """DQGA2/PGA2
        PGA2 Inverse of Tensor (dual number-valued) of d.
        Troublesome for software to invert T, so we have
        to formulate it special.
        """
        mQ1 = sqrt(scalar(DQ_RE2(d)*DQ_QC2(DQ_RE2(d))))
        mQ1_2 = Pow(mQ1,-2)
        dotq1q2 = -DQ_V2(DQ_RE2(d))|DQ_Y2(DQ_V2(DQ_IM2(d)))
        q1wq2w = DQ_S2(DQ_RE2(d))*DQ_Y2(DQ_S2(DQ_IM2(d)))
        return Pow(mQ1,-1)*(1-((q1wq2w+dotq1q2)*mQ1_2)*I42)

    def DQ_U1(d):
        """DQGA1/PGA1
        Taking the unit of d. Basically, this is normalizing.
        Normalize dual quaternion d using the inverse (reciprocal)
        tensor. N(d)=d*T(d)^(-1)
        """
        mQ1 = sqrt(scalar(DQ_RE1(d)*DQ_QC1(DQ_RE1(d))))
        mQ1_1 = Pow(mQ1,-1)
        mQ1_2 = Pow(mQ1,-2)
        dotq1q2 = -DQ_V1(DQ_RE1(d))|DQ_Y1(DQ_V1(DQ_IM1(d)))
        q1wq2w = DQ_S1(DQ_RE1(d))*DQ_Y1(DQ_S1(DQ_IM1(d)))
        return d*(mQ1_1*(1-((q1wq2w+dotq1q2)*mQ1_2)*I41))

    def DQ_U2(d):
        """DQGA2/PGA2
        PGA2 Taking the unit of d. Basically, this is normalizing.
        Normalize dual quaternion d using the inverse (reciprocal)
        tensor. N(d)=d*T(d)^(-1)
        """
        mQ1 = sqrt(scalar(DQ_RE2(d)*DQ_QC2(DQ_RE2(d))))
        mQ1_1 = Pow(mQ1,-1)
        mQ1_2 = Pow(mQ1,-2)
        dotq1q2 = -DQ_V2(DQ_RE2(d))|DQ_Y2(DQ_V2(DQ_IM2(d)))
        q1wq2w = DQ_S2(DQ_RE2(d))*DQ_Y2(DQ_S2(DQ_IM2(d)))
        return d*(mQ1_1*(1-((q1wq2w+dotq1q2)*mQ1_2)*I42))

    def DQ_INV1(d):
        """DQGA1/PGA1
        Inverse d^(-1) of dual quaternion d, for |q1| not 0.
        Troublesome for software to invert, so we
        have to formulate this special.
        """
        return DQ_QC1(DQ_U1(d))*DQ_TINV1(d)

    def DQ_INV2(d):
        """DQGA2/PGA2
        Inverse d^(-1) of dual quaternion d, for |q1| not 0.
```

```
        Troublesome for software to invert, so we
        have to formulate this special.
        """
        return DQ_QC2(DQ_U2(d))*DQ_TINV2(d)

    def DQ_PT1(d):
        """DQGA1/PGA1
        Take the point PT part of d.
        """
        return (d + DQ_DC1(d))/2

    def DQ_PT2(d):
        """DQGA2/PGA2
        PGA2 Take the point PT part of d.
        """
        return (d + DQ_DC2(d))/2

    def DQ_PTD(d):
        """DDQGA/DPGA␣point␣part"""
        return DQ_PT1(DQ_PT2(d))

    def DQ_PL1(d):
        """DQGA1/PGA1
        Take the plane part of d.
        """
        return (d - DQ_DC1(d))/2

    def DQ_PL2(d):
        """DQGA2/PGA2
        Take the plane part of d.
        """
        return (d - DQ_DC2(d))/2

    def DQ_PLD(d):
        """DDQGA/DPGA␣plane␣part"""
        return DQ_PL1(DQ_PL2(d))

    def DQ_LN1(d):
        """DQGA1/PGA1
        Take the line part of d, the vector part.
        """
        return DQ_V1(d)

    def DQ_LN2(d):
        """DQGA2/PGA2
        PGA2 Take the line part of d, the vector part.
        """
        return DQ_V2(d)

    def DQ_LND(d):
```

```
    """DDQGA/DPGA line part"""
    return DQ_LN1(DQ_LN2(d))

def DQ_DOT(d1,d2):
    """Dot product of quaternion vectors d1 and d2"""
    return -(d1*d2 + d2*d1)/2

def DQ_CROSS(d1,d2):
    """Cross product of quaternion vectors d1 and d2"""
    return (d1*d2 - d2*d1)/2
```

A.22 Dual Quaternion Plane and Line Entities

```
def DQ_Plane1(p,n):
    """
    DQGA1/PGA1 DQ Plane, where
    p is any point on the plane, and
    n is the normal vector to the plane, and
    both are given as PGA1 3D vectors.
    """
    return I31*CPNS_Plane1(p,n)

def DQ_Plane2(p,n):
    """
    DQGA2/PGA2 DQ Plane, where
    p is any point on the plane, and
    n is the normal vector to the plane, and
    both are given as PGA1 3D vectors.
    """
    return I32*CPNS_Plane2(p,n)

def DQ_PlaneD(p,n):
    """
    DDQGA/DPGA DQ Plane, where
    p is any point on the plane, and
    n is the normal vector to the plane, and
    both are given as PGA1 3D vectors.
    """
    return DQ_Plane1(p,n)^DQ_Plane2(p,n)

def DQ_Line1(p,d):
    """DQGA1/PGA1
    Dual Quaternion 2-blade Line entity, based on the CPNS_Line1.
    """
    return I31*CPNS_Line1(p,d)*I31.rev()

def DQ_Line2(p,d):
    """DQGA2/PGA2
    Dual Quaternion 2-blade Line entity, based on the CPNS_Line2.
```

```
    """
    return I32*CPNS_Line2(p,d)*I32.rev()

def DQ_LineD(p,d):
    """DDQGA/DPGA
    DDQGA 4-blade Line entity.
    """
    return DQ_Line1(p,d)^DQ_Line2(p,d)
```

A.23 Dual Quaternion Reflection Operations

```
def DQ_Reflect_pPL(p,P):
    """DQGA1/PGA1
    Reflect DQ point p in DQ plane P.
    """
    return -DQ_CC1(P*p*P)

def DQ_Reflect_lPL(l,P):
    """DQGA1/PGA1
    Reflect DQ line l in DQ plane P.
    """
    return -P*DQ_CC1(l*P)

def DQ_Reflect_PLPL(a,b):
    """DQGA1/PGA1
    Reflect DQ plane a in DQ plane b.
    """
    return -b*DQ_CC1(a)*b
```

A.24 Dual Quaternion Intersection Operations

```
def DQ_Intersect_PP(p1,p2):
    """
    Intersect DQ Planes p1 and p2 as a line
    which requires using the full geometric
    product, taking no special parts after.
    """
    return -(p1*DQ_CC1(p2)-p2*DQ_CC1(p1))/2

def DQ_Intersect_LP(l,p):
    """
    Intersect DQ Line and DQ Plane using full geometric products.
    """
    return -(DQ_CC1(l)*DQ_CC1(p)+DQ_CC1(p)*l)/2

def DQ_Intersect_PPP(p1,p2,p3):
    """
    Intersect three DQ Planes as DQ Point using
```

```
    full geometric products to intersect planes.
    There are 4 terms to add.
    """
    def C(d):
        return DQ_CC1(d)
    TM1 =  C(p1)*p2*C(p3)
    TM2 = -C(p2)*p1*C(p3)
    TM3 =  C(p3)*p1*C(p2)
    TM4 = -C(p3)*p2*C(p1)
    return (TM1+TM2+TM3+TM4)/4
```

A.25 Dual Quaternion Projection Operations

```
def DQ_Project_PP(p,P):
    """
    Projection of DQGA point p onto DQGA plane P.
    Returns the DQGA point of the projection as
    nearest point on plane to point, orthographic
    projection.
    """
    cp = DQ_CC1(p)
    cP = DQ_CC1(P)
    return (p - cP*cp*cP)/2

def DQ_Project_LP(l,P):
    """
    Projection of DQGA line l onto DQGA plane P.
    Returns DQGA line of the projection as
    nearest line on plane to line, orthographic
    projection.
    """
    cl = DQ_CC1(l)
    cP = DQ_CC1(P)
    return (l + P*cl*cP)/2

def DQ_Project_PL(p,l):
    """
    Project DQGA point onto DQGA line l.
    This is the point on the line closest to p.
    """
    cp = DQ_CC1(p)
    cl = DQ_CC1(l)
    return (p-cl*p*l)/2
```

A.26 Double PGA Quadric Elements Ts

```
(T1,Tx,Ty,Tz,Txx,Tyy,Tzz,Txy,Tyz,Tzx) =
symbols('T_1,T_x,T_y,T_z,T_xx,T_yy,T_zz,T_xy,T_yz,T_zx')
```

```
T1 = (e4^e0)
Tx = ((e1^e4)+(e0^e5))/2
Ty = ((e2^e4)+(e0^e6))/2
Tz = ((e3^e4)+(e0^e7))/2

Txx = (e5^e1)
Tyy = (e6^e2)
Tzz = (e7^e3)

Txy = ((e6^e1)+(e5^e2))/2
Tyz = ((e7^e2)+(e6^e3))/2
Tzx = ((e5^e3)+(e7^e1))/2
```

A.27 Double PGA Differential Operators

```
(Dx,Dy,Dz) = symbols('D_x,D_y,D_z')

Dx = -2*Tx*Txx
Dy = -2*Ty*Tyy
Dz = -2*Tz*Tzz
```

A.28 Double PGA Example Quadric: Ellipsoid

```
def Ellipsoid(c,r1,r2,r3):
    """
    c = center 3D point
    r1,2,3 radius x,y,z
    """
    px = (c|e1)
    py = (c|e2)
    pz = (c|e3)
    r1si = Pow(r1*r1,-1)
    r2si = Pow(r2*r2,-1)
    r3si = Pow(r3*r3,-1)
    TM1 = (-2*px*Tx + Txx)*r1si
    TM2 = (-2*py*Ty + Tyy)*r2si
    TM3 = (-2*pz*Tz + Tzz)*r3si
    TM4 = (px*px*r1si + py*py*r2si + pz*pz*r3si - 1)*T1
    return (TM1 + TM2 + TM3 + TM4)
```