# Simple Algorithms for implementing the products of Geometric Algebra

Spencer T. Parkin

July 7, 2012

## Introduction

One of the hurtles in learning geometric algebra is coming to terms with all of its products in every situation, especially the most general case, which is that of taking one of the products, (the inner, outer or geometric product), between two multivectors. Starting with a few definitions in geometric algebra, however, we show in this paper that there are simple algorithms for implementing this general case in terms of algebraic manipulations. A computer program based on such manipulations is not likely to perform as well as its counter-parts that use more sufficisticated methods, (such as those that embed geometric algebra in a matrix algebra), but what we'll see here is that, with a sufficient knowledge of geometric algebra, anyone can implement the following aglorithms.

## Definitions

All elements of a geometric algebra are generated by defining the outer product on vectors taken from a vector space. Not having closure, this expands the set of elements from merely a set of vectors to a set of multivectors. These are simply sums of blades. Representing such elements in a computer is easy to do in terms of any set of basis vectors from the generating vector space. We use these, because we can pre-define the inner-product relationships between them. (The set of choices we make for these inner-products is what we'll refer to as the signature of the geometric algebra.) In deriving our algorithms for the products of geometric algebra, however, we need not assume that we're dealing with basis vectors, and consequently, need not assume that we're dealing with basis blades either. All of our descriptions of products may be done with general vectors and blades of the algebra. Therefore, we will make no further reference to basis elements in this paper.

Furthermore, we might be tempted to let our designs of product-taking algorithms be influenced by the signature of the geometric algebra. A Euclidean geometric algebra has the nice property that it is easy to convert blades and versors to and from one another. This is not the case, however, in non-Euclidean geometric algebras, such as those used by the conformal model. Therefore, we will also make no assumptions here about the signature of our geometric algebra when designing our algorithms.

Letting $\mathbb{G}$ denote the set of all elements in our geometric algebra, and having chosen the signature of our algebra, we begin by defining the inner product between any vector $v \in \mathbb{G}$ and any $n$-blade $B \in \mathbb{G}$ as

$$v \cdot B = -\sum_{i=1}^{n}(-1)^i (v \cdot b_i) B^{(i)}, \tag{1}$$

where $B$ may be written as the outer product of vectors $\bigwedge_{i=1}^{n} b_i$, and we define the notation $B^{(i)}$ as the blade $\bigwedge_{j=1,j\neq i}^{n} b_j$. We can clearly compute this result in any computer algebra system, because we know how to compute inner products between vectors as dictated by the signature of our geometric algebra. A similar definition is given for $B \cdot v$, and we may write it terms of what we already have as

$$v \cdot B = -(-1)^n B \cdot v.$$

We will now define the geometric product between any vector $v \in \mathbb{G}$ and any blade $B \in \mathbb{G}$ as

$$vB = v \cdot B + v \wedge B. \tag{2}$$

Similarly, we define $Bv = B \cdot v + B \wedge v$.

Lastly, having now defined the geometric product, we will define the inner product between an $m$-blade $A \in \mathbb{G}$ and an $n$-blade $B \in \mathbb{G}$ as

$$A \cdot B = \langle AB \rangle_{|m-n|}, \tag{3}$$

where, given any multivector $M \in \mathbb{G}$, $\langle M \rangle_i$ denotes the grade $i$ part of $M$. It may not be obvious now, but even though we may not yet realize how to take the geometric product between two blades generally, it can be done using definition (2), as will be shown later.

We now have enough defined to come up with algorithms for all products between any pair of elements in any geometric algebra. In the rest of this paper we will treat the inner and geometric products, but not the outer product, as this latter product is trivial and forms the basis for the entire algebra.

## The Geometric Product

There is perhaps more in common between the outer and geometric products than we might think. What we'll see here is that the problem of finding the geometric product between two multivectors can be solved in almost the same way as finding the outer product between two sums of pseudo-versors. The term pseudo-versor is meant to reference any geometric product of vectors. Unlike a versor, not all vectors in the geometric product of a pseudo-versor need be invertible. (i.e. a pseudo-versor may contain a null-vector in its product.)

We begin with the observation that it is easy compute geometric products between sums of psuedo-versors, and it is easy to compute outer products of sums of blades. The basis for our algorithm to compute a geometric product is therefore an ability to rewrite a blade as a sum of psuedo-versors, and a pseudo-versor as a sum of blades. The geometric product between any two multivectors can then be performed by first converting all blades to sums of pseudo-versors, carrying out the geometric product, and then finally converting all pseudo-versors back to sums of blades. Once the conversion processes are figured out, one needs only know how to apply the distributive properties of the outer and geometric products over addition, and how to combine and cancel like terms. We will not cover these here as they are trivial, and prerequisite to an understanding of the conversion processes anyway.

### Converting a Blade to a Sum of Pseudo-Versors

Let $B$ be an $n$-blade taken from $\mathbb{G}$ with $n > 1$. We wish to rewrite $B$ as a sum of pseudo-versors, (a sum of geometric products of vectors). Let $\{b_i\}_{i=1}^{n}$ be any set of $n$ vectors taken from $\mathbb{G}$ such that $B = \bigwedge_{i=1}^{n} b_i$. Then, using definitions (1) and (2), we see that

$$\begin{aligned} B &= b_1 B^{(1)} - b_1 \cdot B^{(1)} \\ &= b_1 B^{(1)} - \sum_{i=2}^{n} (-1)^i (b_1 \cdot b_i) B^{(1)(i)}. \end{aligned} \tag{4}$$

It is now easy to see how this result may be used as the basis for a recursive algorithm. Notice that $B^{(1)}$ and each of $B^{(1)(i)}$ are of grades lower than $B$. What we've shown is that we can write the sum-of-pseudo-versors expansion of $B$ in terms of the sum-of-pseudo-versors expansion of blades of grades lower than $B$, so that the recursive algorithm terminates when we reach a blade of grade one, which is already its own sum of psuedo-versors.

## Converting a Pseudo-Versor to a Sum of Blades

This conversion process is similar to the last, and uses the same definitions. With pseudo-versors, however, we have to take care as to whether we're dealing with an even or odd pseudo-versor. Even pseudo-versors potentially have a scalar component, which behaves differently than its non-zero grade counter-parts under certain products.

Let $V$ be a pseudo-versor taken from $\mathbb{G}$ such that it may be written as the geometric product of $n > 1$ vectors as $\prod_{i=1}^{n} v_i$. We wish to rewrite it as a multivector, (a sum of blades). Again, using definitions (1) and (2), we have

$$\begin{aligned} V &= v_1 \sum_{i=1}^{n} \langle V^{(1)} \rangle_i \\ &= \langle V^{(1)} \rangle_0 v_1 + \sum_{i=2}^{n} \left( v_1 \wedge \langle V^{(1)} \rangle_i + v_1 \cdot \langle V^{(1)} \rangle_i \right) \\ &= \langle V^{(1)} \rangle_0 v_1 + \sum_{i=2}^{n} \left( v_1 \wedge \langle V^{(1)} \rangle_i - \sum_{j=1}^{i} (-1)^j (v_1 \cdot b_{i,j}) \langle V^{(1)} \rangle_i^{(j)} \right) . \end{aligned} \tag{5}$$

Here we have re-used the notation $V^{(j)}$ to mean the psuedo-versor $\prod_{i=1, i \neq j}^{n} v_i$, and we are writing each $i$-blade $\langle V^{(1)} \rangle_i$ as the outer product of vectors $\bigwedge_{j=1}^{i} b_{i,j}$. (Strictly speaking, for any multivector $M \in \mathbb{G}$, $\langle M \rangle_i$ is an $i$-vector, which is not necessarily an $i$-blade. Never-the-less, there is no loss in generality or mistake here when we abuse the notation $\langle M \rangle_i$ to enumerate the blades in the sum of blades forming $M$. With this understanding, $\langle V^{(1)} \rangle_i^{(j)}$ becomes meaningful above.)

So again, here we have shown how to write the sum-of-blades expansion of $V$ in terms of the sum-of-blades expansions of the psuedo-versor $V^{(1)}$, a lower order or lower grade psuedo-versor, if you will. A recursive algorithm based on this formula terminates when we reach a psuedo-versor of one vector, which is also a blade.

# The Inner Product

At this point we may be tempted to calculate the inner product between an $m$-blade $A \in \mathbb{G}$ and an $n$-blade $B \in \mathbb{G}$ directly from (3) using the algorithm we have just developed for the geometric product. It would, however, be quite a waste of time to compute $AB$ only to throw away all blades in the result that are not of grade $|m - n|$. Fortunately, a little work will show that we can come up with an equation for $A \cdot B$ that will, again, lend itself well to a recursive procedure.

Let us first consider the case $m \leq n$. It then follows that

$$\begin{aligned} A \cdot B &= \langle AB \rangle_{n-m} \\ &= \left\langle A^{(m)} a_m B \right\rangle_{n-m} - \left\langle (A^{(m)}) \cdot a_m B \right\rangle_{n-m} , \end{aligned}$$

where here, to avoid too many parenthesis, we give the inner product precedence over the geometric product, and we let $A$ be written as the outer product of vectors $\bigwedge_{i=1}^{m} a_i$. What we must realize now is that the right-hand term is an outer product between an $(m-2)$-blade and an $n$-blade, which will result in a blade of grade $n - (m - 2) = n - m + 2$, which is also the blade of smallest grade in the expansion of the geometric product. It then follows that this term completely cancels out as we're searching for blades of grade $n-m$. We now have

$$\begin{aligned} A \cdot B &= \left\langle A^{(m)} a_m B \right\rangle_{n-m} \\ &= \left\langle A^{(m)} (a_m \cdot B) \right\rangle_{n-m} + \left\langle A^{(m)} (a_m \wedge B) \right\rangle_{n-m} . \end{aligned}$$

Similarly, notice that the right-hand term is the inner product between an $(m-1)$-blade and an $(n+1)$-blade, which results in a blade of grade $n+1-(m-1) = n-m+2$. This term cancels and we turn our attention to the remaining term on the right-hand side of the equation, and realize that it is the inner product between an $(m-1)$-blade and an $(n-1)$-blade, giving us a blade of grade $n-1-(m-1) = n-m$. We now have the result

$$A \cdot B = A^{(m)} \cdot (a_m \cdot B), \tag{6}$$

whenever $m \leq n$. The inner product expansion of $A \cdot B$ has now been written in terms of the inner product expansions of blades of smaller grade, showing us a way to implement this as a recursive procedure.

Proven similarly, in the case that $n \leq m$, we also have the result

$$A \cdot B = (A \cdot b_1) \cdot B^{(1)},$$

where $B = \bigwedge_{i=1}^{n} b_i$, which may be used to recursively compute $A \cdot B$.

Before we run off and write a recursive algorithm, however, we should simply attempt to apply the formula recursively to itself. Doing so with (6) gives us the result

$$A \cdot B = a_1 \cdot a_2 \cdot \dots \cdot a_m \cdot B,$$

where here we must add that we intend in this equation for the inner product to be right-to-left associative. Similarly, when $n \leq m$, we have

$$A \cdot B = A \cdot b_1 \cdot b_2 \cdot \dots \cdot b_n,$$

where here we intend the inner product to be left-to-right associative. We can now compute inner products as successive applications of definition (1).