**Documentation**

This package was written using Mathematica version 11. The best way to get started is by becoming familiar with the palette. After this, additional examples can be found by running the file "Examples". These instructions assume you have already read and implemented the steps in Quick Start.

Open a new notebook\*. If the palette is not open, select it from the Palette menu. Select your initialization options and then click on the "Needs" statement in the palette to invoke the GA package.

In the 2nd section of the palette, Subscripts & Operators, click on e1, then e2. Hover over the 6 operators (geometric, wedge, scalar, and dot products and right and left contractions) until you find the small circle that represents geometric product, and click it. Then click on e1 again and press [EnterKey]. Try other examples and use the wedge and dot operators to examine the results. Observe that the tooltips explain how to enter both the subscripted e's and the operators from the keyboard. If you find the operator symbols too small to see, click on a magnification level at the bottom of the palette. Zoom back when you wish to restore a smaller palette.

Caution. When in doubt, use parentheses in your operations. For example, e1 e2 °e2 is not the same as (e1 e2 ) °e2. Also, wedge product (^) is associative so you are safe when you enter

x °y °z, but not when you enter x • y • z, x 🞊 y 🞊 z, etc. For convenience, all the operators have been given default definitions mirroring the following example:

x • y • z = (x • y) • z

x • y  • z • w = ( (x • y ) • z ) • w

Just remember that in general (x • y) • z ≠ x • (y • z).

And, you should use parentheses when you mix operators like x • y ^ z .

I wish to point out two things at the outset. First, unlike other GA packages, in this package you use standard notation for subscripts. It was a LOT of work to program this way, and I was discouraged from doing so by several of the IT experts, but I believe it is up to the programmer to make a package friendly for the user rather than making the work easy for the programmer.

The second thing is the matter of dimension. Other than computation time, there is no constraint on dimension. The odd thing that I wish to point out up-front is that there is no need for the user to specify dimension except in rare cases such as the Hodge Dual. It might seem that the dimension n would be required for, say, the wedge product of two multivectors (or clifs, as I prefer to call them) of grades p & q because if the sum of the grades exceeds n then the wedge product is zero. However it is not necessary to enter n in this case because some of the basis elements would have to repeat and thus cancel each other out, resulting in a term of grade less than p + q and a wedge product of 0. So the message is to not worry about the space dimension. Just enter your clifs and the operations take care of dimension for you.

I apologize for not using the standard symbols for left and right contractions but those symbols are not set up in Mathematica for use as binary operators. It is possible to utilize the correct symbols but is a lot of work. Finally, the geometric product is usually represented by juxtaposition but the ° operator is used in this package since Mathematica already uses juxtaposition for regular multiplication.

In the next section of the palette, Multivector Generators, you may need to click the triangle to expand the section. (Click it again to hide the section.) In the 1st column enter a command by selecting it. In the workbook, press [EnterKey]. The output should be the same as the corresponding entry in the right-hand column of the palette. Now modify some parameters of the formula you just inputted, say, change the letter a to the letter b, or dimension 3 to dimension 4, and press [EnterKey] again. Next, select something from column 2. This generates the same output, but now it is provided in the input area for you to use, saving you from having to copy and paste the output. Finally, hoover over the commands in both columns to get more information. These functions have been provided to hopefully simplify typing by providing inputs that you modify rather than having to type all inputs from scratch.

Rotors are used to perform rotations in any number of dimensions. They are used in Clifford algebras and there are examples in the Examples file.

The last 2 items in this section are Complex numbers and Quaternions. The complex numbers are the Clifford subalgebra {a + b i} with i = e1 e2 and a and b are reals. Quaternions are the Clifford subalgebra of elements {a + b i + c j + d k} where a, b, c, & d are real numbers and i, j, and k are as explained in the palette (remember to hover your mouse to reveal tooltips). GA operations that are appropriate (such as InverseG and NormG) for complex numbers or quaternions will also work when restricted to these subalgebras. To use this package to perform complex multiplication, simply type e1e2 where you would normally type i. Similarly for Quaternions, type e2e3 for i, - e1e3 for j, and e1e2 for k.

At this time (2017) there seems not to be a standard vocabulary for GA objects like multivectors, n-vectors, and blades. For example, is a 2-vector a vector in 2-space or is it a bivector, a grade 2 multivector? The vocabulary used in the palette is illustrated in the spreadsheet named Multivector Terminology.

The 4th section of the palette contains the main Geometric Algebra (GA) operations. The Geometric Product can be entered using the first command in this section, or by using the symbol in section 2, or by entering either of these manually. Some texts restrict the Geometric Product to homogenous multivectors; that is, multivectors whose terms are all of the same grade. These texts then discuss "extensions", or define other terms, to discuss the natural extensions of geometric products to non-homogeneous multivectors. This Mathematica package simply uses the term Geometric Product to cover products of any two (or more) multivectors, whether or not homogeneous.

The same thing goes for Wedge, Dot, and Scalar products and Contractions. The user can enter homogeneous or non-homogeneous multivectors. The Wedge Product of 2 multivectors is computed by taking the terms of the Geometric Product whose grade is the sum of the grades of its factors. That is,

A ^ B = [ A ° B ] p+q where p = Grade [A} and q = Grade [B].

Similarly, (see correct symbols in Notebook version of this file, Documentation)

A • B = [ A ° B ] |p – q| Dot Product

A ⎦ B = [ A ° B ] q – p Left Contraction

A ⎣ B = [ A ° B ] p – q Right Contraction

A • B = [ A 🞊 B ] 0 Scalar Product

There are several different definitions given in current literature for the Hodge Dual so I have provided a couple of Hodge functions to allow the user freedom of choice. The equations defining the two I have chosen are shown in the tooltips (hover the mouse over the palette). The first definition is consistent with the implicit definition that the Hodge Dual is the unique operator that satisfies clif2 ^ Hodge[ clif ] = ( clif · clif2 ) ○ i for all multivectors clif2 and where i is the pseudoscalar. By using this package I was able to experimentally verify that the simple explicit definition I use in function HodgeDualG works. The HodgeDual2G definition does NOT satisfy the implicit equations but does have nice geometric properties and seems to be pretty commonly used. Hover your mouse over the palette to view these definitions in the tooltips.

Gorm is basically the square of the norm. The Examples file illustrates both of these. Inverse computes the inverse of a multivector but you have to use caution.

InverseG computes the inverse of a multivector, but be aware that InverseG always provides a result, even if the multivector doesn't actually have an inverse. You can always geometrically multiply InverseG[ x ] with x to see if you get unity.

The reverse of e1 e3 e4 is e4 e3 e1. You cannot enter e4 e3 e1 in Mathematica because Mathematica always sorts the order back to e1 e3 e4. Thus, the ReverseG function simply returns the original multivector preceded by the appropriate ±1. For example, GormG[e1 e3 e4 ] =

- e1e3e4 .

Definitions of the terms can either be found in the tooltips and by examining the source code. The source code organization mirrors the organization in the toolbar.

The operators in the next section, Multivector Support, were developed to simplify the definitions of the primary operators in the prior section and thus may or may not be useful to the user. ExpandG and CollectG are similar to Mathematica's Expand and Collect commands except they are blade-aware, able to expand and collect while handling ei terms.

InitializeG is used internally to reduce squares of basis elements. In the event that the user creates his own products, not using the GA operators, InitializeG can be used to reduce the squares according to the GA initialization specified by the user in step 1. MaxDimG will find the highest basis subscript among the terms of a multivector.

ConstantG will pick out the constant term or terms, if any, in a multivector. Similarly, FreeTermG will pick out the non-constant term or terms, if any.

GradePTerm will pick out the grade p term(s), if any, of a multivector. EijTermG is even more selective. It will pick out only the terms, if any, that have eij as a factor, where eij is a blade such as 1, e1, e4 e5, e2 e3 e4, etc.

It is not necessary to revert to lists in order to perform most operations. This package allows straight-forward operation on the multivectors themselves using standard mathematical notation. However, it can be convenient to use lists for complex operations where one needs to keep the terms of a multivector in a particular order in order to operate on them. (Mathematica, as we have mentioned, puts terms in its own inimitable and mostly-uncontrollable order.) Thus, the last section of the palette is list-related. The top two operators quickly switch between a multivector and its corresponding list of terms.

The 2nd row generates two lists from a multivector. SubscriptListG provides a list of the subscripts of the blades of the multivector terms. For example, SubscriptListG[ 2 + e1 e4 ] = { {0}, {1,4} }. eSubscriptListG generates a similar list but with the subscripts represented as e-subscripts. For example, eSubscriptListG[ 2 + e1 e4 ] = { 0, e1, 4 }. Both lists can come in handy.

GradeListG generates a list of integers that represent the grades of the terms in a multivector. The order of the grades matches the order of the terms in ClifToListG [clif].

The last item is a Signature operator that extends Mathematica's Signature function. Mathematica's operator will find the signature of a list of, say, integers as long as the integers are all distinct. But, in GA we often as not deal with duplicate integers such as in e2 e3 ^ e1 e3. SignatureG will find the signature for { 2, 3, 1, 3 } whereas Signature will not. I believe, actually, that Signature G will find the signature for any class of items for which Mathematica's Signature function works but I have not extensively tested this.

Signature is a measure of the number of pairwise transpositions of adjacent terms required to put the list in natural order. An odd number of transpositions reverses sign and an even number preserves the sign. It is used to determine the correct sign during antisymmetrization operations. For example, e1 e2 ^ e1 = - e2 in 2-space in a + + Clifford algebra.

# Cautions and Work-arounds

1. If you receive a warning about context shadowing or if you notice ei2 terms in your output, you should quit the kernel using the Evaluation menu, close and reopen the palette, and reset your GA initialization settings in the palette.
2. Until such time as the author might implement operator precedence, it is necessary to put parentheses around your multivectors when using the binary operator symbols (see 1st section of palette). For example, you must enter (2 e1) • (3 + e1 e2). However, if A = 2 e1 and B = 3 + e1 e2, then A • B is OK. So is DotPrdtG [ 2 e1, 3 + e1 e2 ].

This package contains only a very basic set of GA operations, but other GA operations can easily be built upon them. The geometric product is somewhat complicated to program because it must handle antisymmetrization. Most GA operations are easy to implement once the geometric product is defined. Thus, the benefits of this package are:

1. It uses natural mathematical notation for subscripts and standard wedge and dot symbols, not requiring you to learn new notation. (Both geometric product and Mathematica's product use a space. To distinguish them, we use a small circle for geometrical product.)

2. Operations are performed naturally, not by converting back and forth to lists (though lists are supported)

3. The function definitions are generally very short, self-documented, and easy to follow, thus easily modified or extended

4. The palette greatly simplifies typing input, entering sample multivectors that can quickly be edited

5. Because symbolic equations are handled (as well as numeric), it is easy to generate generic formulas and to test hypotheses such as whether an identity works in dimensions higher than, say, 3 or whether an identity can be expanded from vectors to blades or beyond.

6. The numeric capabilities allow you to instantly carry out computations that would require an extensive amount of time and tedium to do manually

7. One can quickly configure, or switch between, spatial dimensions, space and space-time, and Clifford and Grassmann.

8. Except for printouts (which can be pages long in higher dimensions) most operations compute instantly even in higher dimensions. The definitions are written entirely without inefficient "for loops" and other such constructs.

Finally, the author is retired and built this package working in a vacuum as a way of teaching himself both GA and Mathematica. If you are using this package, consider yourself a beta-tester. Let me know of any bugs you find. I might find time to correct them. I have thoroughly tested - + + + and somewhat tested + - - - Clifford algebras, but haven't much tested Grassmann algebras as of yet. Contact me at [budsimrin@sbcglobal.net](mailto:budsimrin@sbcglobal.net).

\* You can use the notebook as is or else implement a private notebook context. Private cell contexts have not been tested but are likely compatible. The issue is that the author has had to take great care to manage the context of the symbol e used in the basis e1, e2, ... . . This is because only the multivector symbols, like A, are passed to the package yet the package manipulates the basis elements contained in A. Were the package "e" to have a different context than the notebook "e", those manipulations would fail.

**Warning:** If you open a second notebook during a session, you must execute the “Needs” statement before performing any package operations. Otherwise the context of e will not be preserved and those computations will fail dramatically. If you forget and try to execute a function in the package, then you need to quit the kernel and then execute Needs ["GeomAlg2017Feb`"] as soon as you see the failure. And, as mentioned earlier, you must also close and reopen the palette and use section 1 to re-initialize your GA.