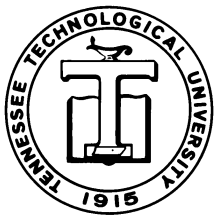
DEPARTMENT OF MATHEMATICS
TECHNICAL REPORT

COMPUTATIONS WITH CLIFFORD AND GRASSMANN ALGEBRAS

RAFAL ABLAMOWICZ

MAY 2009

No. 2009-4



TENNESSEE TECHNOLOGICAL UNIVERSITY
Cookeville, TN 38505

Computations with Clifford and Grassmann Algebras

Rafał Abłamowicz

Abstract. Various computations in Grassmann and Clifford algebras can be performed with a Maple package `CLIFFORD`. It can solve algebraic equations when searching for general elements satisfying certain conditions, solve an eigenvalue problem for a Clifford number, and find its minimal polynomial. It can compute with quaternions, octonions, and matrices with entries in $C\ell(B)$ - the Clifford algebra of a vector space V endowed with an arbitrary bilinear form B. It uses standard (undotted) Grassmann basis in $C\ell(Q)$ but when the antisymmetric part of B is non zero, it can also compute in a dotted Grassmann basis. Some examples of computations are discussed.

Mathematics Subject Classification (2000). 15A66, 68W30.

Keywords. Quantum Clifford algebra, conformal group, contraction, dotted wedge product, grade involution, Grassmann algebra, Hopf algebra, multivector, octonions, quaternions, reversion, singular value decomposition, spinors, Vahlen matrix, wedge product.

Contents

1. Introduction

Some twenty years ago, late Professor Pertti Lounesto together with his colleagues at Helsinki University of Technology developed `CLICAL`, a first semi-symbolic "Clifford algebra calculator". [32] Along with it, Pertti brought to the world of Clifford algebraists a concept of *experimental mathematics*, *algorithmic understanding*, and *counter examples*. [33] One could say that he was a pioneer in bringing together theoretical aspects and the computational part. His works, see for example [30], abound in concrete examples, and counter examples.

This author also believes that learning how mathematical concepts can be handled by a computer provides a combination of theory and practice that, as a result, gives a better understanding of the theory of Clifford algebras and shows how the theory can be applied. This computational approach also provides a fast way to enter into the abstract field. We exemplify this approach by using a Maple package `CLIFFORD`, a system for computations with Grassmann polynomials, that was developed, like `CLICAL`, to support mathematical research in Grassmann and Clifford algebras but using a full computer algebra system. [1, 8, 9]. More recently, this system was vastly extended with `BIGEBRA` for computations with tensors in a general setting of Hopf algebras and co-algebras. [9]

This approach of course is common to many other systems and fields that became possible with the onset of fast computers capable of non-trivial symbolic computations which had been intractable before by hand. For example, in the area of commutative algebra one such system is provided by `Singular` [26] and another by *CoCoA* [18]; in the area of algebraic geometry by *Macaulay*2 [36]; and in the area of general mathematics, a category theory based `AXIOM` which is explicitly "dedicated to research and development of mathematical algorithms". [16] Finally, a review of existing software for Clifford algebras often developed with a specific computation in mind, can be found in [13].

A vector space V endowed with a quadratic form Q is common to a vast host of mathematical, physical and engineering problems. This leads naturally to an algebra structure of the Clifford algebra $C\ell(V, Q)$ and its generalization - a *quantum Clifford algebra* $C\ell(V, B)$ for any bilinear form B. [6] This formalism, as

compared to a standard vector calculus, can now be applied to solving completely new problems. CLIFFORD was developed as a basic tool for all investigations and applications which can be carried in finite dimensional vector spaces equipped with a quadratic form or, equivalently, with a symmetric bilinear form commonly referred to as an *inner* or a *scalar* product. The intrinsic abilities of Maple even allow one to use CLIFFORD in projective and affine geometries while visualizing complicated incidence relations that is helpful, e.g., for image processing, visual perception and robotics.

The authors of CLIFFORD and BIGEBRA have been interested in fundamental questions about q-deformed symmetries and quantum field theory. Just asking questions like such as "What is the most general element fulfilling ...?" has led to unexpected results and new insights. [4, 6, 7, 11] Checking the consistency of the software by testing theorems and known results has led them in the foot steps of Pertti Lounesto to counter examples that have made a rethinking and a more careful restatement of those theorems necessary. However, the most striking ability of CLIFFORD is that it is unique in being able to handle Clifford algebras of an *arbitrary bilinear form* not restricted by symmetry and not directly related to any quadratic form. Since it is now well known that such structures are related to Hopf algebraic twists, later versions of CLIFFORD make an extensive use of a process called *Rota-Stein cliffordization* described in [20, 22–25]. This, in turn, has necessitated introduction of a new algorithm based on this process for a more efficient computation of the Clifford product in $C\ell(V, B)$. As much as Buchberger's celebrated algorithm is indispensable for computing Gröbner bases, this new algorithm described in [9, 10] is indispensable for the Clifford product. Having developed this faster algorithm one was able to find the q-Young Clifford idempotents [6] possessing a desired symmetry; explicitly describe the structure of the **Spin**(3) group; discover continuous families of idempotents in $C\ell(Q)$ [11]; or gain a better understanding of the properties of the dotted wedge product. [9]

The present paper brings to the reader a few examples of computations and results derived with CLIFFORD. It is assumed that the reader is already familiar with Maple [44], a general purpose CAS; if not please consult e.g., [45]. The article is intended as a quick computational introduction to the abstract field of the Clifford algebras, and, especially, to the field of quantum Clifford algebras. For computations with tensor and Hopf algebras we refer to [10] that describes the supplementary package BIGEBRA intended for such computations.

2. Notation and Basic Computations

CLIFFORD uses as default a standard Grassmann basis (Grassmann multivectors) in $\bigwedge V$ where $V = \text{span}\{\mathbf{e}_i \,|\, 1 \leq i \leq n\}$ for $1 \leq n \leq 9$. Then

$$\bigwedge V = \text{span}\{\mathbf{e}_i \wedge \mathbf{e}_j \wedge \ldots \wedge \mathbf{e}_k \,|\, 0 \leq i < j < \ldots < k \leq n\}.$$

In CLIFFORD these basis monomials are written as strings {*Id, e1, ..., e9, e1we2, e1we3, ..., e1we2we3, ...* } although they can be aliased to {*Id, e1, ..., e9, e12, e13, ..., e123, ...* } to shorten input. Here *e1we2* is a string that denotes $\mathbf{e}_1 \wedge \mathbf{e}_2$ and *Id* denotes the identity $\mathbf{1}$ in $\bigwedge V$. However, CLIFFORD can also use one-character long symbolic indices as in *eiwej* which stands for $\mathbf{e}_i \wedge \mathbf{e}_j$. Thus, in principle, it can compute with Clifford algebras in dimensions higher than 9. For example, when $n = 3$, Grassmann basis monomials are:

```
> W=cbasis(3);
```

$$\mathrm{W} = [Id,\ e1,\ e2,\ e3,\ e1we2,\ e1we3,\ e2we3,\ e1we2we3]$$

but aliases can also be used to shorten input/output:

```
> eval(makealiases(3));
```

$$e12,\ e21,\ e13,\ e31,\ e23,\ e32,\ e123,\ e132,\ e213,\ e231,\ e312,\ e321$$

In the above, *eijk* = *eiwejwek* is the wedge product of three 1-vectors: $\mathbf{e}_i, \mathbf{e}_j, \mathbf{e}_k$. Thus, the most general element in the Grassmann algebra $\bigwedge V$ is a Grassmann polynomial which is just a linear combination of Grassmann basis monomials with real coefficients. Notice that symbolic indices are allowed:

```
> p1:=Id+4.5*ei-alpha*e1we2we3;
```

$$p1 := Id + 4.5\, ei - \alpha\, e123$$

The wedge product $\wedge$ is computed with a procedure `wedge` or its ampersand counterpart *&w*:

```
> wedge(e1,e2),e1 &w e2; wedge(ea,eb,ec),ea &w eb &w ec; p1 &w p2;
```

$$e12,\ e12$$

$$eawebwec,\ eawebwec$$

$$e123 - x0\ Id - 4.500000000\ x0\ e1 + \alpha\ x0\ e123 - x12\ e12$$

Following Chevalley's recursive definition, a Clifford product can be introduced in $\bigwedge V$ by means of a left $\lrcorner_B$ (or right $\llcorner_B$) contraction dependent on an arbitrary bilinear form $B : V \times V \to \mathbb{R}$. This leads to elements of the Clifford algebra $C\ell(B)$ expanded into multivectors and makes the Clifford multiplication implicitly dependent on B. The associative Clifford product is given by a procedure `cmul` or its infix form *&c*.

```
> cmul(e1,e2),&c(e1,e2); cmul(ea,eb,ec);
```

$$e12 + B_{1,2}\ Id,\ e12 + B_{1,2}\ Id$$

$$eawebwec + B_{b,c}\ ea - B_{a,c}\ eb + B_{a,b}\ ec$$

Computations in $C\ell(K)$ and $C\ell(B)$ can be performed in the same worksheet since the name of a bilinear form can be passed to `cmul` as a parameter. For example,

```
> cmul[K](e1,e2),&c[K](e1,e2); cmul[K](ei,ej,ek);
```

$$e12 + K_{1,2}\ Id, e12 + K_{1,2}\ Id$$

$$eiwejwek + K_{j,k}\, ei - K_{i,k}\, ej + K_{i,j}\, ek$$

The form B can be numeric or symbolic. For example, when

```
> B:=matrix(2,2,[1,a,a,1]);
```

$$B := \begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix}$$

then the Grassmann basis for $C\ell(B)$ or $\bigwedge V$ will be:

```
> cbas:=cbasis(2);
```

$$cbas := [Id,\ e1,\ e2,\ e12]$$

while the Clifford multiplication table of the basis Grassmann monomials will look as follows:

```
> MultTable:=matrix(4,4,(i,j)->cmul(cbas[i],cbas[j]));
```

$$MultTable := \begin{bmatrix} Id & e1 & e2 & e12 \\ e1 & Id & e12 + a\, Id & e2 - a\, e1 \\ e2 & -e12 + a\, Id & Id & a\, e2 - e1 \\ e12 & a\, e1 - e2 & e1 - a\, e2 & (-1 + a^2)\, Id \end{bmatrix}$$

Irrespective of the bilinear form chosen, the Grassmann multiplication table will always remain as:

```
> wedgetable:=matrix(4,4,(i,j)->wedge(cbas[i],cbas[j]));
```

$$wedgetable := \begin{bmatrix} Id & e1 & e2 & e12 \\ e1 & 0 & e12 & 0 \\ e2 & -e12 & 0 & 0 \\ e12 & 0 & 0 & 0 \end{bmatrix}$$

Let $B = g + F$ where g and F are, respectively, the symmetric and the antisymmetric part of B:

```
> g,F:=matrix(2,2,[g11,g12,g12,g22]),matrix(2,2,[0,F12,-F12,0]);
> B:=evalm(g+F);
```

$$g,\ F := \begin{bmatrix} g11 & g12 \\ g12 & g22 \end{bmatrix}, \begin{bmatrix} 0 & F12 \\ -F12 & 0 \end{bmatrix}$$

$$B := \begin{bmatrix} g11 & g12 + F12 \\ g12 - F12 & g22 \end{bmatrix}$$

Then, the Clifford multiplication table of the basis monomials in $C\ell(B)$ will be as follows:

```
> MultTable:=matrix(4,4,(i,j)->cmul(cbas[i],cbas[j]));
```

$$\begin{aligned}MultTable := &[Id\,,\ e1\,,\ e2\,,\ e12]\\ &[e1\,,\ g11\ Id\,,\ e12+(g12+F12)\,Id\,,\ g11\,e2-(g12+F12)\,e1]\\ &[e2\,,\ (g12-F12)\,Id-e12\,,\ g22\,Id\,,\ (g12-F12)\,e2-g22\,e1]\\ &[e12\,,\ (g12-F12)\,e1-g11\,e2\,,\ g22\,e1-(g12+F12)\,e2\,,\\ &(g12^2-F12^2-g22\,g11)\,Id-2\,e12\,F12]\end{aligned}$$

Observe, that the "standard" anticommutation relations

$$\mathbf{e}_i\mathbf{e}_j+\mathbf{e}_j\mathbf{e}_i=(B_{i,j}+B_{j,i})\mathbf{1}=2g_{i,j}\mathbf{1} \tag{1}$$

are satisfied by the generators $\mathbf{e}_i$, $i=1,2,\ldots,n$, irrespective of the presence of the antisymmetric part F in B. For example,

```
> cmul[g](e1,e2)+cmul[g](e2,e1);
> cmul[B](e1,e2)+cmul[B](e2,e1);
```

$$2\,Id\,g12$$

$$(g12+F12)\,Id+(g12-F12)\,Id\ =\ 2\,g12\,Id$$

It is well known [29, 35] that real Clifford algebras $C\ell(V,Q)=C\ell_{p,q}$ are classified in terms of the signature (p,q) of Q and the dimension $\dim V=n=p+q$. Information about all Clifford algebras $C\ell_{p,q}$, $1\le n\le 9$, for any signature (p,q) has been pre-computed and stored in `CLIFFORD`, and it can be retrieved with a procedure `clidata`. For example, for the Clifford algebra $C\ell_{2,0}$ (also denoted as $C\ell_2$) of the Euclidean plane $\mathbb{R}^2$ we find:

```
> clidata([2,0]); #Clifford algebra of the Euclidean plane
```

$$[real,\ 2,\ simple,\ \frac{1}{2}\,Id+\frac{1}{2}\,e1,\ [Id,\ e2],\ [Id],\ [Id,\ e2]]$$

The meaning of the first three entries in the above output list is that $C\ell_2$ is a simple algebra isomorphic to $\mathrm{Mat}(2,\mathbb{R})$. The 4th entry in the list gives a primitive idempotent f that has been used to generate a minimal left spinor ideal $S=C\ell_2 f$ and, subsequently, the left spinor (lowest dimensional and faithful) representation of $C\ell_2$ in S. In general it is known that, depending on (p,q) and $n=\dim V$, the spinor ideal $S=C\ell_{p,q}f$ is a right K-module where K is either $\mathbb{R}$, $\mathbb{C}$, or $\mathbb{H}$ for simple Clifford algebras when $(p-q)\neq 1 \mod 4$, or $\mathbb{R}\oplus\mathbb{R}$ and $\mathbb{H}\oplus\mathbb{H}$ for semisimple algebras when $(p-q)=1 \mod 4$. [27,30] Elements in the 5th entry (here $[Id, e2]$) generate a real basis in S with respect to f, that is, $S=\mathrm{span}\,\{Id\,\&c\,f, e2\,\&c\,f\}=\mathrm{span}\,\{f,\ e2\,\&c\,f\}$. Elements in the 6th entry span a subalgebra F of $C\ell(Q)$ that is isomorphic to K. In the case of $C\ell_2$ we find that $F=\mathrm{span}\,\{Id\}\cong\mathbb{R}$. The last entry in the output gives 2^k generators of S (with respect to f) viewed as a right module over K where $k=q-r_{q-p}$ and r is the Radon-Hurwitz number.[1] Number k is the number of factors $\frac{1}{2}(\mathbf{1}+T_i)$, where $\{T_i\}$, $i=1,\ldots,k$, is a set of commuting basis Grassmann monomials squaring in $C\ell(Q)$ to $\mathbf{1}$, whose product gives a primitive idempotent f in $C\ell(Q)$. Spinor representation for all Clifford

[1] Type `?RHnumber` in a Maple session when `CLIFFORD` is installed for more help.

algebras $C\ell(Q)$, $1 \le n = p+q \le 9$, and for any signature (p,q) has been precomputed [3] and can be retrieved from `CLIFFORD` with a procedure `matKrepr`. For example, 1-vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ in $C\ell_2$ have the following spinor representation in the basis $\{f, e2\,\&c\,f\}$ of $S = C\ell_2 f$:[2]

```
> matKrepr([2,0]);
```

$$[e1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},\ e2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}]$$

In another example, Clifford algebra $C\ell_3$ of $\mathbb{R}^3$ is isomorphic with $\mathrm{Mat}(2,\mathbb{C})$:

```
> B:=linalg[diag](1,1,1):clidata([3,0]);
```

$$[complex,\ 2,\ simple,\ \frac{1}{2}\,Id + \frac{1}{2}\,e1,\ [Id,\ e2,\ e3,\ e23],\ [Id,\ e23],\ [Id,\ e2]]$$

and its spinor representation is given in terms of Pauli matrices:

```
> matKrepr([3,0]);
```

$$[e1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},\ e2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ e3 = \begin{bmatrix} 0 & -e23 \\ e23 & 0 \end{bmatrix}]$$

Notice that $F = \mathrm{span}\,\{Id, e23\}$ ($e23 = e2we3$) is a subalgebra of $C\ell_3$ isomorphic to $\mathbb{C}$. Since Pauli matrices belong to $\mathrm{Mat}(2,F)$, it is necessary for `CLIFFORD` to compute with Clifford matrices, that is, matrices of a type `'type/climatrix'` with entries in a Clifford algebra.

```
> M1,M2,M3:=rhs(%[1]),rhs(%[2]),rhs(%[3]);
```

$$M1,\ M2,\ M3 := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & -e23 \\ e23 & 0 \end{bmatrix}.$$

Of course Pauli matrices satisfy the same defining relations as the basis vectors $\mathbf{e}_1, \mathbf{e}_2$, and $\mathbf{e}_3$:[3] For example:

```
> 'M1 &cm M2 + M2 &cm M1' = evalm(M1 &cm M2 + M2 &cm M1);
> 'e1 &c e2 + e2 &c e1'=e1 &c e2 + e2 &c e1;
```

$$M1\ \&cm\ M2\ +\ M2\ \&cm\ M1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$e1\ \&c\ e2\ +\ e2\ \&c\ e1 = 0$$

[2] We use the sloppy notation $1 \equiv \mathbf{1}$ in Clifford algebra valued matrices which produces a simpler display.

[3] Here $\&cm$ is a matrix product where Clifford multiplication is applied to the matrix entries. See `?&cm` for more information.

```
> `M1 &cm M1` = evalm(M1 &cm M1),`M2 &cm M2` = evalm(M2 &cm M2),
> `M3 &cm M3` = evalm(M3 &cm M3);
> `e1 &c e1` = e1 &c e1,`e2 &c e2` = e2 &c e2,`e3 &c e3` = e3 &c e3;
```

$$M1\ \&cm\ M1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, M2\ \&cm\ M2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, M3\ \&cm\ M3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$e1\ \&c\ e1 = Id,\ e2\ \&c\ e2 = Id,\ e3\ \&c\ e3 = Id$$

The procedure `matKrepr` gives the linear isomorphism $C\ell(Q) \simeq \mathrm{Mat}(2, \mathbb{R})$, and, in general, $C\ell(Q) \simeq \mathrm{Mat}(2^k, K)$, where $K = \mathbb{R}, \mathbb{C}, \mathbb{H}$, for simple algebras and $C\ell(Q) \simeq \mathrm{Mat}(2^k, K) \oplus \mathrm{Mat}(2^k, K)$, where $K = \mathbb{R}, \mathbb{H}$, for semisimple algebras. In this latter case, it is customary to represent an element in $C\ell(Q)$ in terms of a single matrix over a double field $\mathbb{R} \oplus \mathbb{R}$ or $\mathbb{H} \oplus \mathbb{H}$ rather than as pair of matrices.[4]

One can easily list signatures of the quadratic form Q for which $C\ell(Q)$ is simple or semisimple. For more information, type `?all_sigs`. For example, $C\ell_{1,3}$ has a spinor representation given in terms of 2 by 2 quaternionic matrices whose entries belong to a subalgebra F of $C\ell_{1,3}$ spanned by $\{Id, e2, e3, e2we3\}$:

```
> B:=linalg[diag](1,-1,-1,-1):clidata([1,3]);
```

$[quaternionic, 2, simple, \frac{1}{2}\,Id + \frac{1}{2}\,e1we4, [Id, e1, e2, e3, e12, e13, e23, e123],$
$[Id, e2, e3, e23], [Id, e1]]$

```
> matKrepr([1,3]); #quaternionic matrices
```

$$[e1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, e2 = \begin{bmatrix} e2 & 0 \\ 0 & -e2 \end{bmatrix}, e3 = \begin{bmatrix} e3 & 0 \\ 0 & -e3 \end{bmatrix}, e4 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}]$$

`CLIFFORD` includes several special-purpose procedures to deal with quaternions and octonions (type `?quaternion` and `?octonion` for help), and with quaternionic and octonionic matrices. In particular, following [32], octonions are treated as paravectors in $C\ell_{0,7}$ while their non-associative multiplication, defined via Fano triples, is related to the Fano projective plane $\mathbb{F}_2$ (see `?omultable`, or `?Fano_triples` for more information). User can select different Fano triples and redefine the octonionic multiplication table. Since the bilinear form B can be degenerate[5], one can use `CLIFFORD` to perform computations in Clifford algebras $C\ell_{p,q,d}$ of a degenerate quadratic form Q of signature (p, q) and the dimension d of its radical. For example, the Clifford algebra $C\ell_{0,3,1}$ of the quadratic form $Q(\mathbf{x}) = -x_1^2 - x_2^2 - x_3^2$ where $\mathbf{x} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3 + x_4\mathbf{e}_4 \in \mathbb{R}^4$ is used in robotics to represent rigid motions in $\mathbb{R}^3$ and screw motions in terms of dual quaternions. [4, 41]

Thus, `CLIFFORD` is a repository of mathematical knowledge about Clifford algebras of a quadratic form in dimensions 1 through 9. Together with a supplementary package `BIGEBRA` [8] it can be extended to graded tensor products of

[4]Procedures `adfmatrix` and `mdfmatrix` add and multiply matrices of type `dfmatrix` over such double fields. For more information see `?matKrepr`.

[5]When $B \equiv 0$ then $C\ell(V, B) = C\ell_{0,0,n} = \bigwedge V$ and computations in the Grassmann algebra $\bigwedge V$ can then be done with `CLIFFORD`.

Clifford algebras in higher dimensions. The `BIGEBRA` package is described in [10]. For more information about any `CLIFFORD` or `BIGEBRA` procedure, type `?Clifford` or `?Bigebra` to see its top level help page in the Maple browser. For a computation of spinor representations with `CLIFFORD` we refer to [3].

3. Clifford Product in $C\ell(B)$

The Clifford product in a Clifford algebra $C\ell(B)$ of an arbitrary bilinear form B is introduced via the Chevalley deformation and the Clifford map. [35] The Clifford map $\gamma_{\mathbf{x}}$ is defined on $u \in \bigwedge V$ as

(i) $\gamma_{\mathbf{x}}(u) = \mathtt{LC}(x, u, B) + \mathtt{wedge}(x, u) = \mathbf{x} \lrcorner_B u + \mathbf{x} \wedge u$
(ii) $\gamma_{\mathbf{x}}\gamma_{\mathbf{y}} = \gamma_{\mathbf{x}\wedge\mathbf{y}} + B(\mathbf{x}, \mathbf{y})\gamma_{\mathbf{1}}$
(iii) $\gamma_{a\mathbf{x}+b\mathbf{y}} = a\gamma_{\mathbf{x}} + b\gamma_{\mathbf{y}}$

where $\mathbf{x}, \mathbf{y} \in V$ (see, for example, [35]). One knows how to compute with the wedge $\mathbf{x}\wedge u$ and the left contraction $\mathbf{x} \lrcorner_B u$ of u by $\mathbf{x}$ with respect to the bilinear form B.[6] Following Chevalley, the left contraction has the following properties:

(i) $\mathbf{x} \lrcorner_B \mathbf{y} = B(\mathbf{x}, \mathbf{y})$
(ii) $\mathbf{x} \lrcorner_B (u \wedge v) = (\mathbf{x} \lrcorner_B u) \wedge v + \hat{u} \wedge (\mathbf{x} \lrcorner_B v)$
(iii) $(u \wedge v) \lrcorner_B w = u \lrcorner_B (v \lrcorner_B w)$

where $\mathbf{x} \in V$, $u, v \in \bigwedge V$ and $\hat{u}$ is the Grassmann grade involution. Hence we can use the Clifford map $\gamma_{\mathbf{x}}$ (Chevalley deformation of the Grassmann algebra) to define a Clifford product of a one-vector $\mathbf{x}$ and a multivector u as

$$\mathbf{x}u = \mathbf{x} \lrcorner_B u + \mathbf{x} \wedge u. \tag{2}$$

Analogous formula can also be given for a right Clifford map using the right contraction $\llcorner_B$ implemented as the procedure `RC`.

The Clifford product `cmul` or its ampersand form `&c` of two Grassmann basis monomials can now be defined as follows: A single element from the first factor of the product is split off recursively and then the Chevalley's Clifford map is applied. Namely,

$$\begin{aligned}(\mathbf{e}_a \wedge \ldots \wedge \mathbf{e}_b \wedge \mathbf{e}_c)\&\mathtt{c}(\mathbf{e}_f \wedge \ldots \wedge \mathbf{e}_g) = \\ (\mathbf{e}_a \wedge \ldots \wedge \mathbf{e}_b)\&\mathtt{c}(\mathbf{e}_c \lrcorner_B (\mathbf{e}_f \wedge \ldots \wedge \mathbf{e}_g) + \mathbf{e}_c \wedge \mathbf{e}_f \wedge \ldots \wedge \mathbf{e}_g) \\ - ((\mathbf{e}_a \wedge \ldots \wedge \mathbf{e}_b) \llcorner_B \mathbf{e}_c)\&\mathtt{c}(\mathbf{e}_f \wedge \ldots \wedge \mathbf{e}_g).\end{aligned} \tag{3}$$

Specifically, for $(\mathbf{e}_1 \wedge \mathbf{e}_2)\&\mathtt{c}(\mathbf{e}_3 \wedge \mathbf{e}_4)$ we have

$$\begin{aligned}(\mathbf{e}_1 \wedge \mathbf{e}_2)\&\mathtt{c}(\mathbf{e}_3 \wedge \mathbf{e}_4) &= (\mathbf{e}_1\&\mathtt{c}\mathbf{e}_2)\&\mathtt{c}(\mathbf{e}_3 \wedge \mathbf{e}_4) - B(\mathbf{e}_1, \mathbf{e}_2)\mathbf{1}\&\mathtt{c}(\mathbf{e}_3 \wedge \mathbf{e}_4) \\ &= \mathbf{e}_1\&\mathtt{c}(B(\mathbf{e}_2, \mathbf{e}_3)\mathbf{e}_4 - B(\mathbf{e}_2, \mathbf{e}_4)\mathbf{e}_3 + \mathbf{e}_2 \wedge \mathbf{e}_3 \wedge \mathbf{e}_4) \\ &\quad - B(\mathbf{e}_1, \mathbf{e}_2)\mathbf{1}\&\mathtt{c}(\mathbf{e}_3 \wedge \mathbf{e}_4)\end{aligned}$$

[6] In `CLIFFORD`, the left contraction $\lrcorner_B$ is given by the procedure $\mathtt{LC}(\mathbf{x}, u, B)$, or, simply, by $\mathtt{LC}(\mathbf{x}, u)$ where B is assumed as default. The right contraction $u \llcorner_B \mathbf{x}$ of u by $\mathbf{x}$ is encoded as a procedure $\mathtt{RC}(u, \mathbf{x}, B)$, or simply $\mathtt{RC}(u, \mathbf{x})$.

and the second recursion of the process gives now

$$\begin{aligned} = \quad & B(\mathbf{e}_2,\mathbf{e}_3)B(\mathbf{e}_1,\mathbf{e}_4) - B(\mathbf{e}_2,\mathbf{e}_4)B(\mathbf{e}_1,\mathbf{e}_3) + B(\mathbf{e}_2,\mathbf{e}_3)(\mathbf{e}_1\wedge\mathbf{e}_4) \\ & -B(\mathbf{e}_2,\mathbf{e}_4)(\mathbf{e}_1\wedge\mathbf{e}_3) + \mathbf{B}(\mathbf{e_1},\mathbf{e_2})(\mathbf{e_3}\wedge\mathbf{e_4}) - B(\mathbf{e}_1,\mathbf{e}_3)(\mathbf{e}_2\wedge\mathbf{e}_4) \\ & +B(\mathbf{e}_1,\mathbf{e}_4)(\mathbf{e}_2\wedge\mathbf{e}_3) + \mathbf{e}_1\wedge\mathbf{e}_2\wedge\mathbf{e}_3\wedge\mathbf{e}_4 - \mathbf{B}(\mathbf{e_1},\mathbf{e_2})(\mathbf{e_3}\wedge\mathbf{e_4}) \end{aligned}$$

with the bold terms canceling out. Note that the last term in the r.h.s. was superfluously generated in the first step of the recursion.

The Clifford product can then be derived from the above recursion by linearity and associativity. The induction starts with a left factor of grade one or grade zero which is trivial, i.e., $\mathbf{1}\,\&\mathtt{c}\,\mathbf{e}_a\wedge\ldots\wedge\mathbf{e}_b = \mathbf{e}_a\wedge\ldots\wedge\mathbf{e}_b$. In the case when the left factor is of grade one, we use the Clifford product expressed by the Clifford map of Chevalley, i.e.,

$$\mathbf{e}_a\,\&\mathtt{c}\,\mathbf{e}_b\wedge\ldots\wedge\mathbf{e}_c = \mathbf{e}_a\lrcorner_B\,(\mathbf{e}_b\wedge\ldots\wedge\mathbf{e}_c) + \mathbf{e}_a\wedge\mathbf{e}_b\wedge\ldots\wedge\mathbf{e}_c.$$

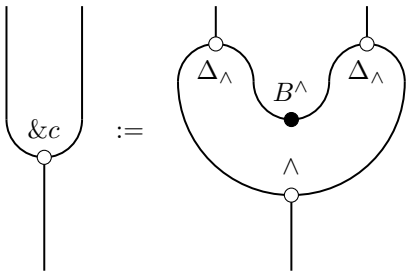
We make a complete induction in the following way: If the left factor is of higher grade, say n, one application of the recursion yields Clifford products where the new left factor is of grade either $n-1$ or $n-2$, hence the recursion stops after at most $n-1$ steps.

The above shows clearly that both the Clifford product and the contraction are explicitly dependent on B. Furthermore, the Clifford product algorithm based on the Chevalley's approach is recursive. It has been encoded in a procedure `cmulNUM` (see Appendix A).

Since the Clifford product provides the main functionality of the `CLIFFORD` package, care has been exercised to select the most appropriate and sound mathematics. Two internal user-selectable functions `cmulNUM` and `cmulRS`, an algorithm based on a combinatorial approach due to Rota and Stein, have been used to encode the Clifford product but the user normally does not use either one. Instead, the user uses a wrapper function $\&c[K](arg1, arg2, \ldots)$ or $\mathtt{cmul}[K](arg1, arg2, \ldots)$ that passes the name of a bilinear form K to either `cmulRS` or `cmulNUM`, whichever one has been selected to act on the multivector basis monomials. This approach allows the user to compute in the same worksheet simultaneously with different Clifford algebras of different bilinear forms. The wrapper function can also act on any number of arguments of type `'type/clipolynom'` as the Clifford product is associative and on a much wider class of types including Clifford matrices of type `'type/climatrix'`. It can also accept Clifford polynomials in other bases such as the Clifford basis $\{\mathbf{1}, \mathbf{e}_i, \mathbf{e}_i\,\&\mathtt{C}\,\mathbf{e}_j, \&\mathtt{C}(\mathbf{e}_i,\mathbf{e}_j,\mathbf{e}_k), \ldots\}$ where $\&\mathtt{C}$ denotes the unevaluated Clifford product. Clifford basis differs from the Grassmann exterior basis when B is not a diagonal matrix.[7]

[7]Procedures converting between Grassmann and Clifford bases belong to a supplementary package `Cliplus` [8] while Clifford polynomials expressed in the Clifford basis are of type `'type/cliprod'`. Type `?cliprod` for more information.

The procedure `cmulRS` is encoded a non-recursive Rota-Stein cliffordization. See [10, 20, 22, 24, 40] and `BIGEBRA` help pages for additional references.[8] The cliffordization process is based on the Hopf algebra theory. The Clifford product is obtained from the Grassmann wedge product and its Grassmann co-product as shown by the following tangle:



(4)

Here $\wedge$ is the Grassmann exterior wedge product and $\Delta_\wedge$ is the Grassmann exterior co-product which is obtained from the wedge product by a categorial duality: To every algebra over a linear space A with a product we find a co-algebra with a co-product over the same space by reversing all arrows in all axiomatic commutative diagrams. Note that the co-product splits each input 'factor' x into a sum of tensor products of ordered pairs $x_{(1)i}, x_{(2)i}$. The main requirement is that every such pair multiplies back to the input x when the dual operation of multiplication is applied, i.e., $x_{(1)i} \wedge x_{(2)i} = x$ for each i-th pair. The 'cup' like part of the tangle decorated with $B^\wedge$ is the bilinear form B on the generating space V extended to the whole Grassmann algebra: It is a map $B^\wedge : \bigwedge V \times \bigwedge V \to k$ with $B : V \times V \to k$ evaluating to $B(\mathbf{x}, \mathbf{y})$ on vectors in V. Hence, `cmulRS` computes the Clifford product on Grassmann basis monomials x and y for the given B, which is later extended to Clifford polynomials by bilinearity, as follows:

$$\texttt{cmulRS}(x, y, B) = \sum_{i=1}^{n} \sum_{j=1}^{m} (\pm) x_{(1)i} \wedge y_{(2)j} B(x_{(2)i}, y_{(1)j}) \tag{5}$$

where n and m give the cardinalities of the required splits and the sign is due to the parity of a permutation needed to arrange the factors.

After reviewing the code of `cmulRS` given in Appendix B it becomes clear that in this algorithm only those terms are created which might be non-zero and are likely to remain in the final result: If all $B_{i,j}$ are non-zero and different so that no cancellation takes place, only exactly those terms will be returned. Yet, no superfluous terms are created like in the recursive procedure `cmulNUM` that later are canceled in subsequent recursive steps. The combinatorial power of the Hopf algebraic approach is clearly demonstrated with this algorithm and its superior behavior shows up in benchmarks. [9, 10]

[8] `BIGEBRA` is an extension of `CLIFFORD` and was specifically developed to work with Hopf algebras. [8]

The two internal Clifford multiplication procedures along with their advantages and disadvantages are discussed in [9]. It should suffice to say here that `cmulNUM` is fast on sparse numeric matrices and on numeric matrices in general when $\dim V \geq 5$, while the procedure `cmulRS` was designed for high efficiency with symbolic calculations.

4. Dotted and Undotted Grassmann Bases

4.1. The dotted wedge

The dotted wedge product was introduced by Lounesto. [35] Clifford algebras with the dotted and the undotted wedge products are isomorphic. It turns out that in various situations, i.e., in quantum field theory or in a representation theory, one is interested in studying isomorphic but not identical algebras in which mathematical objects can be expressed in different bases such as the dotted and the undotted wedge bases. [9]

It was shown above that `CLIFFORD` uses the Grassmann algebra $\bigwedge V$ as the underlying vector space of the Clifford algebra $C\ell(V,B)$. Thus, the Grassmann wedge basis of monomials is the standard basis used in `CLIFFORD`. A general element u in $C\ell(V,B)$ can be therefore viewed as a Grassmann polynomial.

When the bilinear form B has an antisymmetric part $F = -F^T$, it is convenient to split it as $B = g+F$, where g is the symmetric part of B, and to introduce the so called "dotted Grassmann basis" [12] and the dotted wedge product $\dot{\wedge}$. The original Grassmann basis will be referred to here as the "undotted Grassmann basis". In `CLIFFORD`, the wedge product is given by the procedure `wedge` and $\&w$ while the dotted wedge product is given by `dwedge` and $\&dw$.

According to Chevalley's definition of the Clifford product $\&c$, we have

$$\mathbf{x}\,\&\mathtt{c}\,u = \mathbf{x}\lrcorner_B\, u + \mathbf{x}\,\&\mathtt{w}\,u = \mathtt{LC}(\mathbf{x},u,B) + \mathtt{wedge}(\mathbf{x},u) \tag{6}$$

for a 1-vector $\mathbf{x}$ and an arbitrary element u of $C\ell(B)$. As before, $\mathtt{LC}(\mathbf{x},u,B)$ denotes the left contraction of u by $\mathbf{x}$ with respect to the bilinear form B. However, when $B = g + F$ then the left contraction splits too:

$$\mathbf{x}\lrcorner_B\, u = \mathtt{LC}(\mathbf{x},u,B) = \mathbf{x}\lrcorner_g\, u + \mathbf{x}\lrcorner_F\, u = \mathtt{LC}(\mathbf{x},u,g) + \mathtt{LC}(\mathbf{x},u,F) \tag{7}$$

and

$$\begin{aligned} \mathbf{x}\,\&\mathtt{c}\,u &= \mathtt{LC}(\mathbf{x},u,B) + \mathbf{x}\,\&\mathtt{w}\,u && (8)\\ &= \mathtt{LC}(\mathbf{x},u,g) + \mathtt{LC}(\mathbf{x},u,F) + \mathbf{x}\,\&\mathtt{w}\,u && (9)\\ &= \mathtt{LC}(\mathbf{x},u,g) + \mathtt{dwedge}[F](\mathbf{x},u) = \mathtt{LC}(\mathbf{x},u,g) + \mathbf{x}\,\&\mathtt{dw}\,u && (10) \end{aligned}$$

where $\mathbf{x}\,\&\mathtt{dw}\,u = \mathbf{x}\,\&\mathtt{w}\,u + \mathtt{LC}(\mathbf{x},u,F)$. That is, the wedge and the dotted wedge "differ" by the contraction term(s) with respect to the antisymmetric part F of B. This dotted wedge $\&dw$ can be extended to elements of higher grades. Its properties are discussed next.

Procedure `dwedge` (and its infix form $\&dw$) requires an index which can be a symbol or an antisymmetric matrix. That is, `dwedge` computes the dotted wedge

product of two Grassmann polynomials and expresses its answer in the undotted basis. Special procedures exist which convert polynomials between the undotted and dotted bases. When no index is used, the default is F:

```
> dwedge[K](e1+2*e2we3,e4+3*e1we2);&dw(ei+2*ejwek,ei+2*ejwek);
```

$$-(-K_{1,4}+6\,K_{2,3}\,K_{1,2})\,Id-6\,K_{1,2}\,e2we3-6\,K_{2,3}\,e1we2$$
$$-2\,K_{2,4}\,e3+2\,K_{3,4}\,e2-3\,K_{1,2}\,e1+e1we4+2\,e2we3we4$$
$$4\,eiwejwek-4\,F_{i,k}\,ej+4\,F_{i,j}\,ek-8\,F_{j,k}\,ejwek-4\,{F_{j,k}}^{2}\,Id$$

Observe that conversion from the undotted wedge basis to the dotted wedge basis using antisymmetric form F and `dwedge[F]` are related through the following `convert` function:

$$\texttt{dwedge}[F](e1,e2,...,en)=\texttt{convert}(e1we2w...wen,\texttt{wedge\_to\_dwedge},F)$$

which can be shown as follows:

```
> F:=array(1..9,1..9,antisymmetric):
> dwedge[F](e1,e2)=convert(wedge(e1,e2),wedge_to_dwedge,F);
```

$$e1we2+F_{1,2}\,Id=e1we2+F_{1,2}\,Id$$

```
> dwedge[F](e1,e2,e3)=convert(wedge(e1,e2,e3),wedge_to_dwedge,F);
```

$$e1we2we3+F_{2,3}\,e1-F_{1,3}\,e2+F_{1,2}\,e3=e1we2we3+F_{2,3}\,e1-F_{1,3}\,e2+F_{1,2}\,e3$$

$$C\ell(B)_{\wedge} \underset{\texttt{dwedge\_to\_wedge}}{\overset{\texttt{wedge\_to\_dwedge}}{\rightleftarrows}} C\ell(B)_{\hat{\wedge}}$$

Diagram 1. Isomorphisms between $C\ell(B)_{\wedge}$ and $C\ell(B)_{\hat{\wedge}}$.

For a more complete treatment see [9].

4.2. Dotted and undotted wedge bases

Symbolic capabilities of the computer algebra system allow for an investigation of properties of the Clifford product, contraction, and the reversion in the dotted and the undotted bases. In this way, the CAS allows for a better understanding of these fundamental to any Clifford algebra $C\ell(B)$ operations. Here we show only a few facts and refer to [9,10]. For example, we expand the basis of the original wedge into the dotted wedge, and back using the two conversion functions mentioned above. For this purpose we choose $\dim V=3$ and consider $C\ell(B)$ with the antisymmetric part F. The undotted wedge basis for $\bigwedge V$ is then:

```
> w_bas:=cbasis(dim_V); #the wedge basis
```

$$w\_bas:=[Id,\ e1,\ e2,\ e3,\ e1we2,\ e1we3,\ e2we3,\ e1we2we3]$$

Now we map the convert function onto this basis to get the dotted wedge basis:

```
> d_bas:=map(convert,w_bas,wedge_to_dwedge,F);
> test_wbas:=map(convert,d_bas,dwedge_to_wedge,-F);
```

$$d\_bas := [Id,\ e1,\ e2,\ e3,\ e1we2 + F_{1,2}\,Id,\ e1we3 + F_{1,3}\,Id,\ e2we3 + F_{2,3}\,Id,\ e1we2we3 + F_{2,3}\,e1 - F_{1,3}\,e2 + F_{1,2}\,e3]$$

$$test\_wbas := [Id,\ e1,\ e2,\ e3,\ e1we2,\ e1we3,\ e2we3,\ e1we2we3]$$

Notice that only the unity $\mathbf{1}$ and the one vector basis elements $\mathbf{e}_i$ remain unaltered and that the other basis elements of higher grades pick up additional terms of lower grades (which preserves the filtration). It is possible to define aliases in `CLIFFORD` for the dotted wedge basis "monomials" similar to the Grassmann basis monomials. For example, we could denote the element $e1we2 + F[1,2] * Id$ by $e1We2$ ($= \mathbf{e}_1 \dot{\wedge} \mathbf{e}_2$) and similarly for other elements:

```
> alias(e1We2=e1we2 + F[1,2]*Id,e1We3=e1we3 + F[1,3]*Id,
> e2We3=e2we3 + F[2,3]*Id,
> e1We2We3=e1we2we3+F[2,3]*e1-F[1,3]*e2+F[1,2]*e3);
```

$$I,\ e1We2,\ e1We3,\ e2We3,\ e1We2We3$$

and then Maple will automatically display dotted basis in terms of the aliases:

```
> d_bas;
```

$$[Id,\ e1,\ e2,\ e3,\ e1We2,\ e1We3,\ e2We3,\ e1We2We3]$$

That is, as linear spaces we find the isomorphism:

$$\begin{aligned} C\ell(B) &\cong \langle \mathbf{1}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_1 \wedge \mathbf{e}_2, \mathbf{e}_1 \wedge \mathbf{e}_3, \mathbf{e}_2 \wedge \mathbf{e}_3, \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 \rangle \\ &\cong \langle \mathbf{1}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_1 \dot{\wedge} \mathbf{e}_2, \mathbf{e}_1 \dot{\wedge} \mathbf{e}_3, \mathbf{e}_2 \dot{\wedge} \mathbf{e}_3, \mathbf{e}_1 \dot{\wedge} \mathbf{e}_2 \dot{\wedge} \mathbf{e}_3 \rangle \end{aligned}$$

where $\mathbf{e}_1 \dot{\wedge} \mathbf{e}_2 = e1We2$, etc.

4.3. Contraction and Clifford product in dotted and undotted bases

For details we refer to [9]. The contraction $\lrcorner_B$ w.r.t. any bilinear form B works on both undotted and dotted bases in a consistent and essentially the same manner as can be see from the next diagram which utilizes the conversion functions between the two bases. Let F be the antisymmetric part of B. To read more about the left contraction `LC` in $C\ell(B)$ check the help page for `LC` or see [12]. We have the following identity for any two elements u and v in $C\ell(B)$ expressed in the undotted Grassmann basis:

$$v \lrcorner_B u = (v \lrcorner_B u_F)_{-F} \tag{11}$$

As before, u_F is the element u expressed in the dotted basis while $(\ldots)_{-F}$ accomplishes conversion back to the undotted basis. To illustrate this fact, we first contract from the left an arbitrary element u in $C\ell(B)$ by $\mathbf{1}, \mathbf{e}_i, \mathbf{e}_i \wedge \mathbf{e}_j, \mathbf{e}_i \wedge \mathbf{e}_j \wedge \mathbf{e}_k$, $1 \le i, j, k \le 3$ (here we limit our example to $\dim V = 3$) and then we extend it to a left contraction by an arbitrary element v in $C\ell(B)$.

```
> u:=add(x.i*w_bas[i+1],i=0..7):uF:=convert(uw,wedge_to_dwedge,F):
> v:=add(y.i*w_bas[i+1],i=0..7):
```

Contraction with respect to $\mathbf{1}$:

```
> evalb(LC(Id,u,B)=convert(LC(Id,uF,B),dwedge_to_wedge,-F));
```

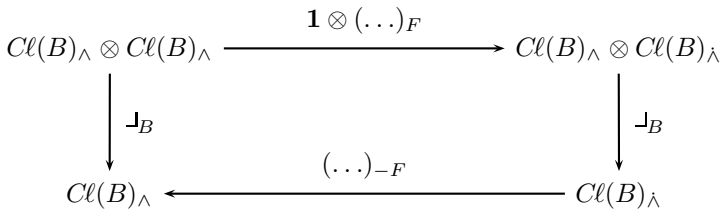


Diagram 2. Contraction w.r.t. wedge and dotted wedge.

true

Contraction with respect to $\mathbf{e}_i$:

```
> evalb(LC(ei,u,B)=convert(LC(ei,uF,B),dwedge_to_wedge,-F));
```

true

Contraction with respect to $\mathbf{e}_i \wedge \mathbf{e}_j$:

```
> evalb(LC(eiwej,u,B)=convert(LC(eiwej,uF,B),dwedge_to_wedge,-F));
```

true

Contraction with respect to $\mathbf{e}_i \wedge \mathbf{e}_j \wedge \mathbf{e}_k$:

```
> evalb(LC(eiwejwek,u,B)=convert(LC(eiwejwek,uF,B),dwedge_to_wedge,-F));
```

true

Finally, contraction with respect to an arbitrary element v:

```
> evalb(LC(v,u,B)=convert(LC(v,uF,B),dwedge_to_wedge,-F));
```

true

Once we have the dotted and the undotted Grassmann bases, we can build a Clifford algebra $C\ell(B)$ over each basis set but with different bilinear forms: $B = g$ or $B = g + F$ respectively (following notation from [12]). Let us compute various Clifford products with respect to the symmetric form g and with respect to the full form B using procedure `cmul` that takes a bilinear form as its index. As an example, we will use two most general elements u and v in $\bigwedge V$ when $\dim V = 3$. Most output will be eliminated.

```
> u:=add(x.k*w_bas[k+1],k=0..7):v:=add(y.k*w_bas[k+1],k=0..7):
```

We can then define in $\bigwedge V$ a Clifford product `cmul[g]` with respect to the symmetric part g and another Clifford product `cmul[B]` with respect to the entire form B:

```
> cmulg:=proc() return cmul[g](args) end proc:
> cmulB:=proc() return cmul[B](args) end proc:
```

We will illustrate relation between the two Clifford products by chasing the following commutative diagram, however most output will be eliminated to save space.

First, we compute the Clifford product `cmul[g]`(u, v) in $C\ell(g)$ in undotted Grassmann basis.

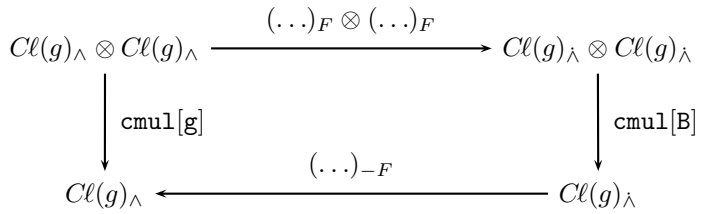


Diagram 3. Clifford multiplications `cmul[g]` and `cmul[B]` w.r.t. dotted and undotted basis.

```
> uv:=cmulg(u,v): #Clifford product w.r.t. g in Cl(g) in wedge basis
```
Now, we convert u and v to u_F and v_F, respectively, expressed in the dotted wedge basis:
```
> uF:=convert(u,wedge_to_dwedge,F):vF:=convert(v,wedge_to_dwedge,F):
```
We now compute the Clifford product of u_F and v_F in $C\ell(B)$ in the dotted wedge basis,
```
> uFvF:=cmulB(uF,vF): #Clifford product in Cl(B) in dwedge basis
```
convert back the above result back to the undotted wedge basis:
```
> uv2:=convert(uFvF,dwedge_to_wedge,-F): #convert result dwedge->wedge
```
and verify that the results are the same:
```
> simplify(uv-uv2); #shows equality!
```

$$0$$

Thus, we have shown that the following identity involving `cmul[g]` and `cmul[B]` is true (at least when $\dim V = 3$).[9] For a general result see, e.g., [14, 28].

$$(u\,v)_g = u\,\&\mathtt{c}_g\,v = (u_F\,\&\mathtt{c}_B\,v_F)_{-F} = ((u_F\,v_F)_B)_{-F} \tag{12}$$

This shows that the Clifford algebra $C\ell(g)$ of the symmetric part g of B using the undotted exterior basis is isomorphic, as an associative algebra, to the Clifford algebra $C\ell(B)$ of the entire bilinear form $B = g + F$ spanned by the dotted wedge basis if the antisymmetric part F of B is exactly the same as F used to connect the two bases.

$$(\ldots)_F \in \mathrm{Hom}_{\mathrm{Alg}}(C\ell(g), C\ell(B)), \qquad\qquad B = g + F$$

5. More on the Associativity of the Dotted Wedge

It was shown above that `CLIFFORD` uses Grassmann algebra $\bigwedge V$ as the underlying vector space of the Clifford algebra $C\ell(V, B)$. Thus, the Grassmann wedge basis of monomials is the standard basis used in the package. A general element $u \in C\ell(V, B)$ can be therefore viewed as a Grassmann polynomial.

Operation `dwedge` introduced in Sect. 4.1 is associative with the unity $\mathbf{1} = Id$ as its unit:

[9] Here, $(u\,v)_g$ is the Clifford product with respect to g while $u_F\,\&\mathtt{c}_B\,v_F$ and $(u_F v_F)_B$ are the Clifford products with respect to B, that is, in $C\ell(g)$ and $C\ell(B)$, respectively.

```
> evalb(dwedge[F](dwedge[F](e1,e2),e3)=dwedge[F](e1,dwedge[F](e2,e3)));
```

true

The associativity of the dotted wedge implies that the diagram 4 commutes. It was checked with CLIFFORD up to dimension 5.

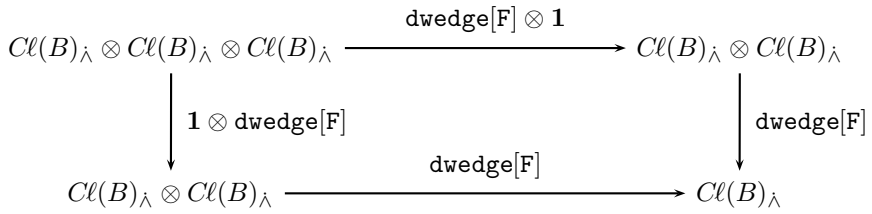


Diagram 4. Associativity of `dwedge[F]` in $C\ell(B)_{\dot\wedge}$.

For some arbitrary random Clifford polynomials[10] u, v, z expressed in Grassmann undotted basis we can show associativity as follows:

```
> u:=2*Id+e1-3*e2we3:v:=3*Id-4*e1we3+e7:z:=4*Id-2*e3+e1we2we3:
> evalb(dwedge[F](Id,u)=u),evalb(dwedge[F](u,Id)=u);
```

true, *true*

```
> evalb(dwedge[F](dwedge[F](u,v),z)=dwedge[F](u,dwedge[F](v,z)));
```

true

We have, therefore, the following identity that expresses an isomorphism between two Clifford algebras: dotted and undotted. For any two elements u and v in $C\ell(B)$, $B = g + F$, that are, by default, expressed in terms of the undotted Grassmann basis, we find:

$$u \wedge v = (u_F \,\dot\wedge\, v_F)_{-F}. \tag{13}$$

Here u_F and v_F are the elements u and v expressed in the dotted basis with respect to the form F while $(\ldots)_{-F}$ denotes conversion back from the dotted basis to the undotted basis w.r.t. $-F = F^T$. $C\ell(B)_\wedge$ and $C\ell(B)_{\dot\wedge}$ denote the modules w.r.t. the two filtrations in use. This can be illustrated in CLIFFORD as follows:

```
> uu:=convert(u,wedge_to_dwedge,F); vv:=convert(v,wedge_to_dwedge,F);
```

$$uu := e1 - 3\,e2we3 - 3\,F_{2,3}\,Id + 2\,Id$$

$$vv := 3\,Id - 4\,e1we3 - 4\,F_{1,3}\,Id + e7$$

[10] In CLIFFORD ver. 6 and higher there are three procedures useful for testing that return a random Grassmann basis monomial, a random monomial and a random polynomial, respectively. See `?rd_clibasmon, ?rd_climon, ?rd_clipolynom`.

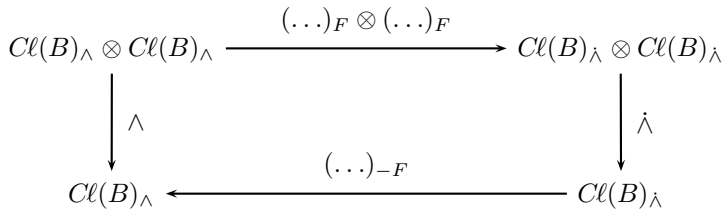


Diagram 5. Relation between $\wedge$ and $\dot\wedge$ products.

```
> out1:=dwedge[F](uu,vv): #dwedge computed w.r.t. F
> out2:=convert(out1,dwedge_to_wedge,-F); #back to undotted basis
```

$$out2 := 3\,e1 - 9\,e2we3 + 6\,Id - 8\,e1we3 + e1we7 - 3\,e2we3we7 + 2\,e7$$

```
> out3:=wedge(u,v); #direct computation of wedge product
```

$$out3 := 3\,e1 - 9\,e2we3 + 6\,Id - 8\,e1we3 + e1we7 - 3\,e2we3we7 + 2\,e7$$

and it can be seen that `out2` = `out3` establishing the relation (13).

The dotted and the undotted wedge bases are treated fully in [9]. One can also find there a discussion of a dependence of contraction, the Clifford product, and the reversion on the antisymmetric part F of B in the Clifford algebra $C\ell(B)$. In the following section we illustrate properties of the reversion in dotted and undotted bases.

6. Reversion in Dotted and Undotted Bases

Following [9] we proceed to show that the expansion of the Clifford basis elements into the dotted or undotted exterior products has also implications for other well known operations such as the Clifford reversion anti-automorphism

$$\tilde{} : C\ell(B) \to C\ell(B), \quad uv \mapsto \tilde{v}\tilde{u},$$

which preserves the grades in $\dot\bigwedge V$ [but not in $\bigwedge V$ unless B is symmetric.] Only when the bilinear form is symmetric, we find that the reversion is grade preserving, otherwise it reflects only the filtration: That is, reversed elements are in general sums of terms of the same and lower degrees.

```
> reversion(e1we2,B); #reversion with respect to B
> reversion(e1we2,g); #reversion with respect to g (classical result)
```

$$-e1we2 - 2\,F_{1,2}\,Id$$

$$-e1we2$$

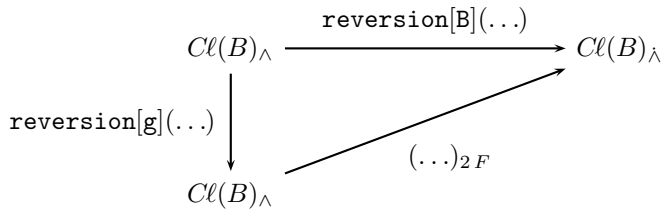


Diagram 6. Relation between `reversion[g]` and `reversion[B]` and the basis transformation $(\ldots)_{2F}$.

We illustrate how the various reversions are related in the following commutative diagram:

The reader should note that the map, depicted by the diagonal arrow in Diagram 6, involves a change of basis induced by the antisymmetric bilinear form $2\,F$ and not F. The factor 2 is crucial and appears due to an asymmetry between the undotted and dotted bases. This suggests to introduce a symmetrically related *triple* of bases w.r.t. $-\frac{1}{2}F$, $F \equiv 0$ and $\frac{1}{2}F$. In such a setting, F (resp. $-F$) connects the two dotted bases induced by $\pm\frac{1}{2}F$.

Observe in the pre-last display above that only when $B_{1,2} = B_{2,1}$, the result $-\mathbf{e}_1 \wedge \mathbf{e}_2$ known from the theory of classical Clifford algebras is obtained. Likewise,

```
> cbas:=cbasis(3);
```

$$cbas := [Id,\ e1,\ e2,\ e3,\ e1we2,\ e1we3,\ e2we3,\ e1we2we3]$$

```
> map(reversion,cbas,B);
```

$$[Id,\ e1,\ e2,\ e3,\ -e1we2 - 2\,F_{1,2}\,Id,\ -e1we3 - 2\,F_{1,3}\,Id,\ -e2we3 - 2\,F_{2,3}\,Id,$$
$$-2\,F_{2,3}\,e1 + 2\,F_{1,3}\,e2 - 2\,F_{1,2}\,e3 - e1we2we3]$$

If instead of B we use a symmetric matrix $g = g^T$ (or the symmetric part of B), then

```
> map(reversion,cbas,g);
```

$$[Id,\ e1,\ e2,\ e3,\ -e1we2,\ -e1we3,\ -e2we3,\ -e1we2we3]$$

Convert now $\mathbf{e}_1 \wedge \mathbf{e}_2$ to the dotted basis to get $\mathbf{e}_1 \dot{\wedge} \mathbf{e}_2 = e1We2$:

```
> convert(e1we2,wedge_to_dwedge,F);
```

$$e1We2$$

Applying reversion to $e1We2$ with respect to F one gets the reversed element in the dotted basis:

```
> reversed_e1We2:=reversion(e1We2,F);
```

$$reversed\_e1We2 := -e1we2 - F_{1,2}\,Id$$

Observe, that the above element is equal to the negative of *e1We2* just like reversing *e1we2* with respect to the symmetric part g of B:

```
> reversed_e1We2+e1We2;
```

$$0$$

Finally, convert reversed *e1We2* to the undotted standard Grassmann basis to get $-e1we2$:

```
> convert(reversed_e1We2,dwedge_to_wedge,-F);
```

$$-e1we2$$

The above, of course, can be obtained by applying reversion to *e1we2* with respect to the symmetric part g of B:

```
> reversion(e1we2,g); #reversion w.r.t. the symmetric part g
```

$$-e1we2$$

This shows that the dotted wedge basis is the particular basis which is stable under the Clifford reversion computed with respect to F, the antisymmetric part of the bilinear form B. This requirement allows one to distinguish Clifford algebras $C\ell(g)$ which have a symmetric bilinear form g from those which do not have such symmetric bilinear form but a more general form B instead. We call the former **classical Clifford algebras** while we use the term **quantum Clifford algebras** for the general not necessarily symmetric case. [6]

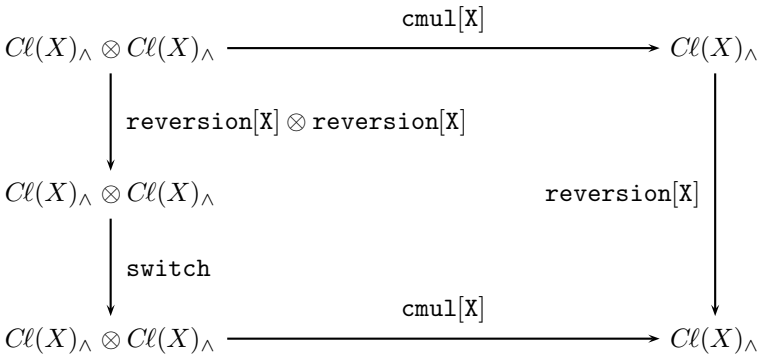


Diagram 7. Relation between the `reversion[X]` of type `X`$\in$ {`g,F,B`} with the corresponding Clifford multiplication `cmul[X]`. The map called `switch` is the ungraded switch of tensor factors, that is, `switch`$(A \otimes B) = B \otimes A$.

7. Spinor Representation of $C\ell(Q)$ in Minimal Left Ideals

See [3] for a complete treatment of symbolic computation of spinor representations of simple and semisimple Clifford algebras. Here we provide some basic facts and a few examples. We will use a procedure `spinorKrepr` from `CLIFFORD`.

Procedure `spinorKrepr` finds a matrix spinor representation of any Clifford polynomial in a minimal left ideal $S = C\ell(Q)f$ or a minimal right ideal $S = fC\ell(Q)$ over the field $K \simeq fC\ell(q)f$. Depending on the signature of Q, the field K is isomorphic to $\mathbb{R}$ when $(p-q) \mod 8 = 0, 1, 2$; $\mathbb{C}$ when $(p-q) \mod 8 = 3, 7$; or $\mathbb{H}$ when $(p-q) \mod 8 = 4, 5, 6$. In order to compute the spinor representation, one needs (i) a Clifford polynomial p whose matrix of the field K needs to be found; (ii) a list of basis elements of the type `'type/clipolynom'` which give a K-basis for S over the field K. Among those elements there is the primitive idempotent f used to generate S; (iii) a list of elements of the type `'type/clibasmon'` which generate the field K, and (iv) a string `'left'` or `'right'` depending whether S is a left or right minimal ideal.

Since the steps needed to compute spinor representations are rather involved, the user may just want to use already pre-computed and stored matrices over K representing 1-vectors. Procedure `matKrepr` uses stored data and can compute matrices representing any Clifford polynomial. A few simple examples are shown below.

Example 1. *Clifford algebra $C\ell_{2,0}$ of the Euclidean plane $\mathbb{R}^2$ is known to be isomorphic to $\mathbb{R}(2)$, the ring of 2×2 real matrices.*

```
> dim:=2:B:=linalg[diag](1,1): #define the bilinear form B for Cl(2,0)
> clibasis:=cbasis(dim): #compute a Clifford basis for Cl(2,0)
> data:=clidata(B); #retrieve and display data about Cl(2,0)
```

$$data := [real,\ 2,\ simple,\ \frac{Id}{2} + \frac{e1}{2},\ [Id,\ e2],\ [Id],\ [Id,\ e2]]$$

```
> f:=data[4]: #assign pre-stored idempotent to f or use your own here
> sbasis:=minimalideal(clibasis,f,'left');#compute a real basis in Cl(2,0)f
```

$$sbasis := [[\frac{Id}{2} + \frac{e1}{2},\ \frac{e2}{2} - \frac{e12}{2}],\ [Id,\ e2],\ left]$$

```
> Kbasis:=Kfield(sbasis,f): #compute a basis for the field K
> SBgens:=sbasis[2]: #generators for a real basis in S
> FBgens:=Kbasis[2]; #generator for K is only one since K=R
```

$$FBgens := [Id]$$

```
> K_basis:=spinorKbasis(SBgens,f,FBgens,'left'); #K-basis for S
```

$$K\_basis := [[\frac{Id}{2} + \frac{e1}{2},\ \frac{e2}{2} - \frac{e12}{2}],\ [Id,\ e2],\ left]$$

Here are matrices representing basis monomials of $C\ell_{2,0}$:

```
> M0,M1,M2,M3:=op(map(spinorKrepr,clibasis,K_basis[1],FBgens,'left'));
```

$$M0,\ M1,\ M2,\ M3 := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Since the spinor representation of $C\ell_{2,0}$ is an algebra isomorphism from $C\ell_{2,0}$ to $\mathbb{R}(2)$, matrix M_{12} that represents $\mathbf{e}_1\mathbf{e}_2 = \mathbf{e}_1 \wedge \mathbf{e}_2$ is a product of matrices M_1 and M_2 with Clifford multiplication applied to their entries. Procedure which handles multiplication of such matrices is called `rmulm` and it can also be entered in its infix form `&cm`:

```
> M12:=M1 &cm M2;
```

$$M12 := \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Notice that M_1 and M_2 have the same algebraic properties as the basis elements they represent: $\mathbf{e}_1\mathbf{e}_2 + \mathbf{e}_2\mathbf{e}_1 = 0$:

```
> e1 &c e2 + e2 &c e1, evalm(M1 &cm M2 + M2 &cm M1);
```

$$0, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Let's find a matrix representing an arbitrary Clifford polynomial p in $C\ell_{2,0}$:

```
> p:=a0+a1*e1+a2*e2+a12*e12;
```

$$p := a0 + a1\ e1 + a2\ e2 + a12\ e12$$

```
> spinorKrepr(p,K_basis[1],FBgens,'left');#matrix of p in S
```

$$\begin{bmatrix} a0 + a1 & a2 + a12 \\ a2 - a12 & a0 - a1 \end{bmatrix}$$

The simplest way to compute that matrix is to use procedure `matKrepr` that uses pre-computed spinor representations of all Clifford algebras $C\ell_{p,q}, p+q \leq 9$, that are stored in `CLIFFORD`:

```
> matKrepr(p);
```

$$\begin{bmatrix} a0 + a1 & a2 + a12 \\ a2 - a12 & a0 - a1 \end{bmatrix}$$

Example 2. *In this example we consider the Clifford algebra $C\ell_{3,0} \simeq \mathbb{C}(2)$. In the following we will see how matrices with entries in $C\ell_{3,0}$, and more precisely, in $K = fC\ell_{3,0}f \simeq \mathbb{C}$ are handled.*

```
> dim:=3:B:=linalg[diag](1,1,1):#define the bilinear form B for Cl(3,0)
> clibasis:=cbasis(dim): #compute Clifford basis for Cl(3,0)
> data:=clidata(B); #retrieve and display data about Cl(3,0)
```

$$data := [complex,\ 2,\ simple,\ \frac{Id}{2} + \frac{e1}{2},\ [Id,\ e2,\ e3,\ e23],\ [Id,\ e23],\ [Id,\ e2]]$$

```
> f:=data[4]: #assign pre-stored idempotent to f or use your own here
> sbasis:=minimalideal(clibasis,f,'left'):#compute a real basis in Cl(3,0)f
> Kbasis:=Kfield(sbasis,f); #compute a basis for the field K
```

$$Kbasis := [[\frac{Id}{2} + \frac{e1}{2},\ \frac{e23}{2} + \frac{e123}{2}],\ [Id,\ e23]]$$

```
> SBgens:=sbasis[2]: #generators for a real basis in S
> FBgens:=Kbasis[2]; #generators for K are two since K=C
```

$$FBgens := [Id,\ e23]$$

```
> K_basis:=spinorKbasis(SBgens,f,FBgens,'left');
```

$$K\_basis := [[\frac{Id}{2} + \frac{e1}{2},\ \frac{e2}{2} - \frac{e12}{2}],\ [Id,\ e2],\ left]$$

Here are the matrices representing 1-vector basis monomials of $C\ell_{3,0}$. Matrices `sigma[1]`, `sigma[2]` and `sigma[3]` are the well-known *Pauli matrices* with entries in the field K:

```
> sigma[1],sigma[2],sigma[3]:=
> op(map(spinorKrepr,[e1,e2,e3],K_basis[1],FBgens,'left'));
```

$$\sigma_1,\ \sigma_2,\ \sigma_3 := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ \begin{bmatrix} 0 & -e23 \\ e23 & 0 \end{bmatrix}$$

Let's find matrices representing the two basis elements in the spinor ideal $S = C\ell_{3,0}f$. As expected, these matrices over K have the following form:

```
> f1,f2:=K_basis[1][1],K_basis[1][2];
```

$$f1,\ f2 := \frac{Id}{2} + \frac{e1}{2},\ \frac{e2}{2} - \frac{e12}{2}$$

```
> F1,F2:=op(map(spinorKrepr,[f1,f2],K_basis[1],FBgens,'left'));
```

$$F1,\ F2 := \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},\ \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

Thus, a spinor s is a complex vector written in terms of the basis $\{f_1, f_2\}$ and its one-column complex matrix with entries in $K = \{1, \mathbf{e}_2 \wedge \mathbf{e}_3\}$ is:

```
> psi[1],psi[2]:=a*Id+b*e23,c*Id+d*e23;
```

$$\psi_1,\ \psi_2 := a\ Id + b\ e23,\ c\ Id + d\ e23$$

```
> s:=f1 &c psi[1] + f2 &c psi[2];#remember that S is a right K-vector space
```

$$s := \frac{a\,Id}{2} + \frac{b\,e23}{2} + \frac{a\,e1}{2} + \frac{b\,e123}{2} - \frac{c\,e12}{2} - \frac{d\,e13}{2} + \frac{c\,e2}{2} + \frac{d\,e3}{2}$$

```
> sm:=matKrepr(s); #matrix of s
```

$$sm := \begin{bmatrix} a + b\,e23 & 0 \\ c + d\,e23 & 0 \end{bmatrix}$$

Since CLIFFORD can handle computations with matrices in any Clifford algebra[11], it can also handle spinor representations in quaternionic spinor spaces and in spinor spaces over dual numbers in the case of semisimple Clifford algebras. [3]

8. Two Scalar Products in Spinor Ideals

Scalar products β_+ and β_- in spinor ideals $S = C\ell_{p,q}f$ are discussed and classified in [30, 39]. In CLIFFORD there are corresponding procedures `beta_plus` and `beta_minus` that compute scalar products $\beta_+(\psi,\phi)$ and $\beta_-(\psi,\phi)$, respectively, for any two spinors $\psi, \phi \in S$. Recall that β_+ denotes the reversion in $C\ell_{p,q}$ while β_- denotes the conjugation, that is, a composition of the grade involution $\hat{}$ and the reversion $\tilde{}$ in $C\ell_{p,q}$. Let β be either of the two anti involutions β_+ or β_- of $C\ell_{p,q}$. Following [39] Lounesto argues that when $C\ell_{p,q}$ is simple, for any spinor ideal $V = C\ell_{p,q}f$ generated by a primitive idempotent f there exists an invertible element u in $C\ell_{p,q}$ such that $fu = u\beta(f)$. Let $F = fC\ell_{p,q}f$. Then V becomes a right F-module. Define a map $\lambda \to \lambda^\sigma = u\beta(\lambda)u^{-1}$ which is an automorphism of the ring F. Then the map $\psi \to \psi^\xi = u\beta(\psi)$ right-to-left F-semilinear on V and, therefore, the map $V \times V \to F$ defined by

$$(\psi, \phi) \mapsto u\psi^\xi\beta(\psi)\phi \tag{14}$$

is a scalar product on V.[12] It turns out that the element u can be chosen to be an element e_A of the Grassmann basis for $C\ell_{p,q}$.

The first two arguments `beta_plus` and `beta_minus` are spinors ψ and ϕ which are Clifford polynomials of `'type/clipolynom'`. The third argument is a primitive idempotent f (for simple Clifford algebras or $f+\hat{f}$ for semisimple Clifford algebras). The fourth optional argument `'s'` will be a placeholder for the invertible element u described above.

Example 3. *Let's compute the two bilinear forms* `beta_plus` *and* `beta_minus` *on* $S = C\ell_{3,0}f$*, a spinor space of the Clifford algebra of the Euclidean space* $\mathbb{R}^3$*. To shorten output, procedure* `makealiases` *is used.*

[11] When computing matrix products, one can apply the Clifford product, the wedge product, or any product to the matrix entries. See help in CLIFFORD.

[12] Similar discussion is extended to semi-simple Clifford algebras $C\ell_{p,q}$. In that case one considers $W = C\ell_{p,q}e$ where $e = f + \hat{f}$.

```
> B:=diag(1,1,1); #define B for Cl(3,0)
```

$$B := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> dim:=coldim(B):eval(makealiases(dim)):
> data:=clidata(B); #retrieve and display data about Cl(B)
```

$$data := [complex,\ 2,\ simple,\ \frac{Id}{2} + \frac{e1}{2},\ [Id,\ e2,\ e3,\ e23],\ [Id,\ e23],\ [Id,\ e2]]$$

```
> f:=data[4]: #assign pre-stored idempotent to f or use your own here
> for i from 1 to nops(data[7]) do f||i:=data[7][i] &c f od;
```

$$f1 := \frac{Id}{2} + \frac{e1}{2}, \quad f2 := \frac{e2}{2} - \frac{e12}{2}$$

```
> Kbasis:=data[6]; #here K = C
```

$$Kbasis := [Id,\ e23]$$

Let's define arbitrary (complex) spinor coefficients `psi1`, `psi2`, `phi1` and `phi2` for two spinors ψ and ϕ in $S = C\ell_{3,0} f \simeq \mathbb{C}^2$. Notice, that these coefficients belong to a subalgebra K of $C\ell_{3,0}$ spanned by $\{1, \mathbf{e}_{23}\}$ that is isomorphic to $\mathbb{C}$ since $\mathbf{e}_{23}^2 = -1$. Recall also that the left minimal ideal $S = C\ell(Q)f$ is a *right* K-module. That's why the 'complex' coefficients must be written on the right of the spinor basis elements `f1` and `f2` in S:

```
> psi1:=psi11 * Id + psi12 * e23;psi2:=psi21 * Id + psi22 * e23;
```

$$\psi 1 := \psi 11\ Id + \psi 12\ e23, \quad \psi 2 := \psi 21\ Id + \psi 22\ e23$$

```
> phi1:=phi11 * Id + phi12 * e23;phi2:=phi21 * Id + phi22 * e23;
```

$$\phi 1 := \phi 11\ Id + \phi 12\ e23, \quad \phi 2 := \phi 21\ Id + \phi 22\ e23$$

Thus, $\psi = f_1\psi_1 + f_2\psi_2$ and $\phi = f_1\phi_1 + f_2\phi_2$ which is shown in Maple with a help of an unevaluated Clifford product `climul` as follows:

```
> psi:='f1 &c psi1' + 'f2 &c psi2';phi:='f1 &c phi1' + 'f2 &c phi2';
```

$$\psi := \mathrm{climul}(f1,\ \psi 1) + \mathrm{climul}(f2,\ \psi 2), \quad \phi := \mathrm{climul}(f1,\ \phi 1) + \mathrm{climul}(f2,\ \phi 2)$$

Now, we compute $\beta_+(\psi, \phi)$ while we store the purespinor u under the name `purespinor1`. Notice, that β_+ is invariant under the unitary group $U(2)$.

```
> beta_plus(psi,phi,f,'purespinor1');purespinor1;
```

$$(\psi 22\,\phi 22 + \psi 21\,\phi 21 + \psi 11\,\phi 11 + \psi 12\,\phi 12)\ Id + (\psi 21\,\phi 22 - \psi 12\,\phi 11 + \psi 11\,\phi 12 - \psi 22\,\phi 21)\ e23$$

$$Id$$

Observe that $\beta_+(\psi,\phi) = \psi_1^*\phi_1 + \psi_2^*\phi_2$ where * denotes complex conjugation, whereas a pure spinor in this case is the identity element.[13]

Finally, we compute $\beta_-(\psi,\phi)$ while we store the purespinor u under the name `purespinor2`. Notice, that β_- is invariant under the complex symplectic group $Sp(2,\mathbb{C})$.

```
> beta_minus(psi,phi,f,'purespinor2');purespinor2;
```

$$(-\psi 12\,\phi 22 + \psi 22\,\phi 12 + \psi 11\,\phi 21 - \psi 21\,\phi 11)\,Id + (-\psi 22\,\phi 11 + \psi 12\,\phi 21 + \psi 11\,\phi 22 - \psi 21\,\phi 12)\,e23\,e2$$

Observe that $\beta_-(\psi,\phi) = \psi_1\phi_2 - \psi_2\phi_1$ whereas a pure spinor in this case is $\mathbf{e}_2$. We can easily check now that pure spinors `purespinor1` and `purespinor2` have the desired commuting properties with the idempotent f:

```
> u:=purespinor1: f &c u - u &c reversion(f);
```

$$0$$

```
> u:=purespinor2: f &c u - u &c conjugation(f);
```

$$0$$

For more information see [3] and [30].

9. Continuous Families of Idempotents: Low Dimensional Examples

In this section we will show how one can discover with `CLIFFORD` existence of continuous families of idempotents. For a complete treatment of this topic we refer to [11].

It is well known [30] that any primitive idempotent f in $C\ell_{p,q}$ is expressible as a product

$$f = \frac{1}{2}(1 \pm e_{T_1})\frac{1}{2}(1 \pm e_{T_2}) \ldots \frac{1}{2}(1 \pm e_{T_k}) \tag{15}$$

where $e_{T_i}, i = 1,\ldots,k$, are commuting basis monomials with square 1, and $k = q - r_{q-p}$, where r_i is the Radon-Hurwitz number.[14] Furthermore, $C\ell_{p,q}$ has a complete set of 2^k primitive *mutually annihilating idempotents*[15] each with k factors as shown in (15). In `CLIFFORD` procedure `clidata` displays one chosen primitive idempotent to generate precomputed spinor representations of Clifford algebras in dimensions up to 9.

---

[13] One should not confuse this complex conjugation with Maple's symbol for multiplication as in `phi1:= phi11*Id+phi12*e23;` above.

[14] The Radon-Hurwitz number is defined by recursion as $r_{i+8} = r_i + 4$ and these initial values: $r_0 = 0$, $r_1 = 1$, $r_2 = r_3 = 2$, $r_4 = r_5 = r_6 = r_7 = 3$. In `CLIFFORD` it is given by the procedure `RHnumber`.

[15] There are 2^k possible sign choices for the k factors in (15). Any two primitive idempotents f and g obtained by selecting different signs in (15) are *mutually annihilating*, that is $fg = gf = 0$.

We will show how to find continuous families of idempotents in a Clifford algebra $C\ell(Q)$ by finding a general solution to the equation $f^2 = f$ with a procedure `clisolve`. As low dimensional examples, we will use $C\ell_{2,0}$, $C\ell_{1,1}$ and $C\ell_{3,0}$.

Example 4. *Families of idempotents in $C\ell_{2,0}$ (see also [43]) can be discovered as follows.*

```
> dim_V:=2:B:=diag(1,1):bas:=cbasis(dim_V):clidata();
```

$$[real,\ 2,\ simple,\ \frac{Id}{2} + \frac{e1}{2},\ [Id,\ e2],\ [Id],\ [Id,\ e2]]$$

As shown above, a standard primitive idempotent in $C\ell_{2,0}$ is $f_1 = \frac{1}{2} + \frac{1}{2}\mathbf{e}_1$. We will look, however, for the most general element f in $C\ell_{2,0}$ that satisfies $f^2 = f$.

```
> f:=add(x[i]*bas[i],i=1..2^dim_V);
```

$$f := x_1\, Id + x_2\, e1 + x_3\, e2 + x_4\, e12$$

There are four real solutions:

```
> sol:=map(allvalues,clisolve(cmul(f,f)-f,f)):sol_real:=remove(has,sol,I);
```

$$\begin{aligned} sol\_real := [0,\ Id,\ & \frac{Id}{2} + \frac{1}{2}\sqrt{1 + 4\,{x_4}^2}\, e2 + x_4\, e12,\ \frac{Id}{2} - \frac{1}{2}\sqrt{1 + 4\,{x_4}^2}\, e2 + x_4\, e12, \\ & \frac{Id}{2} + \frac{1}{2}\sqrt{1 - 4\,{x_3}^2 + 4\,{x_4}^2}\, e1 + x_3\, e2 + x_4\, e12, \\ & \frac{Id}{2} - \frac{1}{2}\sqrt{1 - 4\,{x_3}^2 + 4\,{x_4}^2}\, e1 + x_3\, e2 + x_4\, e12] \end{aligned}$$

We verify that each solution is an idempotent (of course, 0 and Id are the trivial ones):

```
> map(x -> is(simplify(cmul(x,x)=x)),sol_real);
```

$$[true,\ true,\ true,\ true,\ true,\ true]$$

Observe that all nontrivial idempotents found above are ungraded, i.e., they are neither odd nor even. If we set $x_4 = x_3 = 0$ in the above two idempotents that contain $\sqrt{1 - 4\,{x_3}^2 + 4\,{x_4}^2}$, we recover the default mutually annihilating primitive pair. However,

$$\frac{1}{2} \pm \frac{1}{2}\sqrt{1 + 4\,{x_4}^2 - 4\,{x_3}^2}\,\mathbf{e}_1 + x_3\,\mathbf{e}_2 + x_4\,\mathbf{e}_1 \wedge \mathbf{e}_2 \tag{16}$$

gives a two-parameter family of idempotents in $C\ell_{2,0}$ as long as $1 + 4\,{x_4}^2 - 4\,{x_3}^2 \geq 0$. The classical (discrete) idempotents occupy the center $(0, 0)$ of that parameterized region in the real x_3x_4-plane. It can be easily checked that the above two idempotents, in general, do not add up to 1 and do not annihilate each other unless both parameters are zero.

Example 5. *Let's now change the signature from $(2, 0)$ to $(1, 1)$ and repeat the above computations in the Clifford algebra $C\ell_{1,1}$ of neutral signature.*

```
> dim_V:=2:B:=diag(1,-1):bas:=cbasis(dim_V):clidata();
```

$$[real,\ 2,\ simple,\ \frac{Id}{2} + \frac{e12}{2},\ [Id,\ e1],\ [Id],\ [Id,\ e1]]$$

```
> f:=add(x[i]*bas[i],i=1..2^dim_V);
```

$$f := x_1\, Id + x_2\, e1 + x_3\, e2 + x_4\, e12$$

```
> sol:=map(allvalues,clisolve(cmul(f,f)-f,f)):sol_real:=remove(has,sol,I);
```

$$sol\_real := [0,\ Id,\ \frac{Id}{2} + \frac{1}{2}\sqrt{1-4\,x_4{}^2}\, e2 + x_4\, e12,\ \frac{Id}{2} - \frac{1}{2}\sqrt{1-4\,x_4{}^2}\, e2 + x_4\, e12,$$
$$\frac{Id}{2} + \frac{1}{2}\sqrt{1+4\,x_3{}^2-4\,x_4{}^2}\, e1 + x_3\, e2 + x_4\, e12,$$
$$\frac{Id}{2} - \frac{1}{2}\sqrt{1+4\,x_3{}^2-4\,x_4{}^2}\, e1 + x_3\, e2 + x_4\, e12]$$

```
> map(x -> is(simplify(cmul(x,x)=x)),sol_real);
```

$$[true,\ true,\ true,\ true,\ true,\ true]$$

Thus, like in the Euclidean case, we find that

$$\frac{1}{2} \pm \frac{1}{2}\sqrt{1+4\,x_3{}^2-4\,x_4{}^2}\,\mathbf{e}_1 + x_3\,\mathbf{e}_2 + x_4\,\mathbf{e}_1\wedge\mathbf{e}_2 \tag{17}$$

gives a two parameter family of idempotents provided $1+4\,x_3{}^2-4\,x_4{}^2 \geq 0$. Like in the Euclidean case we find that the idempotents in the pair (17) do not add up to 1 and do not mutually annihilate unless $x_3 = x_4 = 0$. In that case we find graded idempotents $\frac{1}{2} \pm \frac{1}{2}\,\mathbf{e}_1\wedge\mathbf{e}_2$.

In the anti-Euclidean signature $(0,2)$ we only find, as expected, trivial idempotents in $C\ell_{0,2} \simeq \mathbb{H}$. In higher dimensions, for example in $C\ell_{3,0}$, one also finds families parameterized by more than two parameters.

10. Vahlen Matrices

For the background material on Vahlen matrices and conformal transformations, see [15,31,33,34,38]. Procedure `isVahlenmatrix` determines if a given 2×2 Clifford matrix $V \in \mathrm{Mat}(2, C\ell(Q))$ is a Vahlen matrix and it returns `true` or `false` accordingly. Any matrix with entries in a Clifford algebra is of '`type/climatrix`'.

A *Vahlen matrix* is a 2×2 matrix $V = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ with entries in a Clifford algebra $C\ell_{p,q}$ such that the following conditions are met:

1. a, b, c, d are products of 1-vectors,
2. The pseudo-determinant[16] of V computed as $a\tilde{d} - b\tilde{c}$ equals $+1$ or -1,
3. $a\tilde{b}$, $\tilde{b}d$, $d\tilde{c}$, and $\tilde{c}a$ are all 1-vectors.[17]

Condition (i) above implies that a, b, c, and d are elements of the *Lipschitz group* $L_{p,q}$ of $C\ell_{p,q}$. Recall [35] that this group is defined as follows:

$$L_{p,q} = \{s \in C\ell_{p,q} \mid x\mathbf{x}\hat{s}^{-1} \in \mathbb{R}^{p,q}, \mathbf{x} \in \mathbb{R}^{p,q}\}.$$

[16] In `CLIFFORD` it is computed with a procedure `pseudodet`.
[17] Here $\tilde{}$ denotes the reversion anti-automorphism in $C\ell_{p,q}$. In `CLIFFORD` it is the `reversion` operation.

Procedure `isproduct` is used to determine whether this condition is met. Recall that in dimensions $n \geq 3$ sense preserving conformal mappings are restrictions of the *Möbius transformations* and are compositions of rotations, translations, dilations and transversions (called also special conformal transformations). A Möbius transformation in $\mathbb{R}^{p,q}$ can be written in the form

$$\mathbf{x} \rightarrow \frac{a\mathbf{x}+b}{c\mathbf{x}+d} \tag{18}$$

where $\mathbf{x}$ is a 1-vector that belongs to $\mathbb{R}^{p,q}$, a, b, c, d belong to $C\ell_{p,q}$, and the products and the inverse are taken in $C\ell_{p,q}$. This transformation may be represented by the Vahlen matrix V defined above. Rotations, translations, dilations, and transversions will then be represented as follows:

- Rotations: $\mathbf{x} \rightarrow a\mathbf{x}a^{-1}$ where a belongs to $\mathbf{Spin}^{+}(p,q)$, the identity component of $\mathbf{Spin}(p,q)$, and $V = \begin{pmatrix} a & 0 \\ 0 & a \end{pmatrix}$,
- Translations: $\mathbf{x} \rightarrow \mathbf{x}+\mathbf{b}$ where $\mathbf{b} \in \mathbb{R}^{p,q}$ and $V = \begin{pmatrix} 1 & \mathbf{b} \\ 0 & 1 \end{pmatrix}$,
- Dilations: $\mathbf{x} \rightarrow s\mathbf{x}$ where $s > 0$ and $V = \begin{pmatrix} \sqrt{s} & 0 \\ 0 & \frac{1}{\sqrt{s}} \end{pmatrix}$,
- Transversions: $\mathbf{x} \rightarrow \dfrac{(\mathbf{x}+\mathbf{x}^2\mathbf{c})}{(1+2\mathbf{x}\cdot\mathbf{c}+\mathbf{x}^2\mathbf{c}^2)}$ where $\mathbf{c} \in \mathbb{R}^{p,q}$, $\mathbf{x}\cdot\mathbf{c}$ is the dot product in $\mathbb{R}^{p,q}$ and $V = \begin{pmatrix} 1 & 0 \\ \mathbf{c} & 1 \end{pmatrix}$.

Let's consider a few simple examples in the signature $(3,1)$. Our goal is to see how `CLIFFORD` manipulates with Clifford matrices. At the same time we will verify some results from [38]. We begin with a Vahlen matrix R that gives a rotation:

```
> B:=linalg[diag](1,1,1,-1); #bilinear form for the Minkowski space
```

$$B := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

```
> a:=e1we2; #an element of grade 2 in Spin+(3,1)
> R:=linalg[matrix](2,2,[a,0,0,a]); #Vahlen matrix that gives a rotation
> 'isVahlenmatrix(R)'=isVahlenmatrix(R);
```

$$a := e12, \quad R := \begin{bmatrix} e12 & 0 \\ 0 & e12 \end{bmatrix}$$

$$'isVahlenmatrix(R)' = true$$

Next, we consider a Vahlen matrix T that gives a translation:

```
> b:=e1+2*e3; #vector in R^(3,1)
> T:=linalg[matrix](2,2,[1,b,0,1]);
> 'isVahlenmatrix(T)'=isVahlenmatrix(T);
```

$$b := e1 + 2\,e3, \quad T := \begin{bmatrix} 1 & e1 + 2\,e3 \\ 0 & 1 \end{bmatrix}$$

$$'isVahlenmatrix(T)' = true$$

A Vahlen matrix Dil that gives a dilation transformation:

```
> delta:=1/4: #a positive parameter
> Dil:=linalg[matrix](2,2,[sqrt(delta),0,0,1/sqrt(delta)]);
> 'isVahlenmatrix(Dil)'=isVahlenmatrix(Dil);
```

$$Dil := \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & 2 \end{bmatrix}$$

$$'isVahlenmatrix(Dil)' = true$$

Finally, a Vahlen matrix Tv that gives a transversion transformation:

```
> c:=2*e1-e3; #a vector in R^(3,1)
> Tv:=linalg[matrix](2,2,[1,0,c,1]);
> 'isVahlenmatrix(Tv)'=isVahlenmatrix(Tv);
```

$$c := 2\,e1 - e3, \quad Tv := \begin{bmatrix} 1 & 0 \\ 2\,e1 - e3 & 1 \end{bmatrix}$$

$$'isVahlenmatrix(Tv)' = true$$

If we now take a product of these four matrices above,[18] we will obtain an element $conf$ of the conformal group in $\mathbb{R}^{3,1}$:

```
> conf:=R &cm T &cm Dil &cm Tv;
```

$$conf := \begin{bmatrix} \frac{e12}{2} + 10\,e23 & 4\,e123 - 2\,e2 \\ -2\,e123 - 4\,e2 & 2\,e12 \end{bmatrix}$$

Since in the product above each matrix appeared exactly once, the diagonal entries of $conf$ must be invertible. We find the inverses of each element with `cinv`:

```
> cinv(conf[1,1]); #inverse of conf[1,1]
```

$$-\frac{2\,e12}{401} - \frac{40\,e23}{401}$$

[18] `&cm` denotes a matrix multiplication in `CLIFFORD` with the Clifford product applied to the matrix entries.

```
> cinv(conf[2,2]); #inverse of conf[2,2]
```

$$-\frac{e12}{2}$$

However, there are elements in the conformal group of $\mathbb{R}^{3,1}$ whose Vahlen matrices do not have invertible elements at all. The following example of such matrix is due to Johannes Maks. [38] Matrix W defined below represents an element in the identity component of the conformal group of $\mathbb{R}^{3,1}$:

```
> W:=evalm((1/2)*linalg[matrix](2,2,[1-e14,-e1+e4,e1+e4,1+e14]));
```

$$\mathrm{W} := \begin{bmatrix} \frac{1}{2} - \frac{e14}{2} & -\frac{e1}{2} + \frac{e4}{2} \\ \frac{e1}{2} + \frac{e4}{2} & \frac{1}{2} + \frac{e14}{2} \end{bmatrix}$$

Notice that the diagonal elements of W are non-trivial idempotents in $C\ell_{3,1}$ hence as such they are not invertible:

```
> type(W[1,1],idempotent); #element (1,1) of W is an idempotent
```

true

```
> type(W[2,2],idempotent); #element (2,2) of W is an idempotent
```

true

Notice also that the off-diagonal elements of W are isotropic vectors in $\mathbb{R}^{3,1}$, hence they are also non-invertible. In $C\ell_{3,1}$ such vectors have zero squares:

```
> cmul(W[1,2],W[1,2]),cmul(W[2,1],W[2,1]);
```

0, 0

Let's now verify that matrix W defined above is a Vahlen matrix:

```
> 'isVahlenmatrix(W)'=isVahlenmatrix(W);
```

true

However, matrix W represents an element of the identity component of the conformal group in $\mathbb{R}^{3,1}$ since its pseudo-determinant is 1, and since it can be written as a product of a transversion, a translation, and a transversion. Thus, in another words, W is <u>not</u> a product of <u>just one</u> rotation, <u>one</u> translation, <u>one</u> dilation, and/or <u>one</u> transversion:

```
> Tv:=linalg[matrix](2,2,[1,0,(e1+e4)/2,1]);
```

$$Tv := \begin{bmatrix} 1 & 0 \\ \frac{e4}{2} + \frac{e1}{2} & 1 \end{bmatrix}$$

```
> T:=linalg[matrix](2,2,[1,(-e1+e4)/2,0,1]);
```

$$T := \begin{bmatrix} 1 & \frac{e4}{2} - \frac{e1}{2} \\ 0 & 1 \end{bmatrix}$$

```
> Tv &cm T &cm Tv = evalm(W); # W = Tv &cm T &cm Tv
```

$$\begin{bmatrix} \frac{1}{2}-\frac{e14}{2} & \frac{e4}{2}-\frac{e1}{2} \\ \frac{e4}{2}+\frac{e1}{2} & \frac{1}{2}+\frac{e14}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}-\frac{e14}{2} & \frac{e4}{2}-\frac{e1}{2} \\ \frac{e4}{2}+\frac{e1}{2} & \frac{1}{2}+\frac{e14}{2} \end{bmatrix}$$

```
> pseudodet(W); #computing pseudo-determinant of W
```

$$Id$$

Thus, the above computation confirms that $W = Tv \,\&cm\, T \,\&cm\, Tv$ and that the pseudo-determinant of W is 1.

There is another variation of Johannes Maks' example of a Vahlen matrix W without any invertible entries. Matrix W represents an element in the identity component of the conformal group of $\mathbb{R}^{3,1}$.

```
> W:=evalm((1/2)*linalg[matrix](2,2,[1-e24,-e2+e4,e2+e4,1+e24]));
```

$$\mathrm{W} := \begin{bmatrix} \frac{1}{2}-\frac{e24}{2} & -\frac{e2}{2}+\frac{e4}{2} \\ \frac{e2}{2}+\frac{e4}{2} & \frac{1}{2}+\frac{e24}{2} \end{bmatrix}$$

Notice that the diagonal elements of W are non-trivial idempotents in $C\ell_{3,1}$, hence they are not invertible in $C\ell_{3,1}$:

```
> type(W[1,1],idempotent); #element (1,1) of W is an idempotent
> type(W[2,2],idempotent); #element (2,2) of W is an idempotent
```

$$true, true$$

Notice also that the off-diagonal elements of W are isotropic vectors in $\mathbb{R}^{3,1}$, hence they are also non-invertible:

```
> cmul(W[1,2],W[1,2]),cmul(W[2,1],W[2,1]);
```

$$0,\ 0$$

Finally, we verify that W is a Vahlen matrix:

```
> 'isVahlenmatrix(W)'=isVahlenmatrix(W);
```

$$'isVahlenmatrix(W)' = true$$

However, W is an element of the identity component of the conformal group in $\mathbb{R}^{3,1}$ since its pseudo-determinant is 1, and since it can be written as a product of a transversion, a translation, and a transversion. As before, W is *not* a product of just one rotation, one translation, one dilation, and/or one transversion:

```
> Tv:=linalg[matrix](2,2,[1,0,(e2+e4)/2,1]);
```

$$Tv := \begin{bmatrix} 1 & 0 \\ \frac{e4}{2}+\frac{e2}{2} & 1 \end{bmatrix}$$

```
> T:=linalg[matrix](2,2,[1,(-e2+e4)/2,0,1]);
```

$$T := \begin{bmatrix} 1 & \frac{e4}{2} - \frac{e2}{2} \\ 0 & 1 \end{bmatrix}$$

```
> Tv &cm T &cm Tv = evalm(W); #W = Tv &cm T &cm Tv
```

$$\begin{bmatrix} \frac{1}{2} - \frac{e24}{2} & \frac{e4}{2} - \frac{e2}{2} \\ \frac{e4}{2} + \frac{e2}{2} & \frac{1}{2} + \frac{e24}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} - \frac{e24}{2} & \frac{e4}{2} - \frac{e2}{2} \\ \frac{e4}{2} + \frac{e2}{2} & \frac{1}{2} + \frac{e24}{2} \end{bmatrix}$$

```
> pseudodet(W); #computing pseudo-determinant of W
```

$$Id$$

Thus, the above computation again confirms that $W = Tv \,\&cm\, T \,\&cm\, Tv$ and that the pseudo-determinant of W is 1.

11. Singular Value Decomposition and Clifford Algebra

In this section we will show how the Singular Value Decomposition (SVD) of a matrix can be translated into the Clifford algebra language. For the background information on SVD we refer to [42]. There are many uses of SVD such as in image processing, description of the so called *principal gains* in a multivariable system [37], or in an automated data indexing known as *Latent Semantic Indexing* (or LSI). LSI presents a very interesting and useful technique in information retrieval models and it is based on the SVD. [19] While in these practical cases computations are done numerically, it may be of interest to ask whether SVD of a matrix can be performed in the framework of Clifford algebras. Likewise, whether SVD of a Clifford number can be found without using matrices. That is, if any new insights, theoretical or otherwise, into such decomposition could be gained when stated in the Clifford algebra language.

We will explore a well-known fact that when $p - q \neq 1 \bmod 4$, the Clifford algebra $C\ell_{p,q}$ is a simple algebra of dimension 2^n, $n = p + q$, isomorphic to a full matrix algebra $\mathrm{Mat}(2^k, \mathbb{K})$ of $2^k \times 2^k$ matrices[19] with entries in the division ring $\mathbb{K}$. The ring $\mathbb{K}$ is a subalgebra of $C\ell_{p,q}$ isomorphic to $\mathbb{R}$, $\mathbb{C}$, or $\mathbb{H}$ depending on the signature (p, q) and the dimension n (see [3]). Thus, any operation performed on a matrix $A \in \mathrm{Mat}(2^k, \mathbb{K})$ can be expressed as an operation on the uniquely corresponding to it element $\mathtt{p}$ in $C\ell_{p,q}$. The choice of the signature (p, q) depends on the size of A and the division ring $\mathbb{K}$. Of course, for computational reasons one should find the smallest Clifford algebra $C\ell_{p,q}$ such that the given matrix A can be

[19] The value k is determined by the formula $k = q - r_{q-p}$, where r_i is the Radon-Hurwitz number. The Radon-Hurwitz number is defined by a recursion as $r_{i+8} = r_i + 4$ and these initial values: $r_0 = 0$, $r_1 = 1$, $r_2 = r_3 = 2$, $r_4 = r_5 = r_6 = r_7 = 3$.

embedded into $\mathrm{Mat}(2^k,\mathbb{K}) \overset{\varphi}{\simeq} C\ell_{p,q}$. In the following we will use the same approach as in [2] where a technique for matrix exponentiation based on the isomorphism φ was presented. In particular, we will use a faithful spinor representation of $C\ell_{p,q}$ in a minimal left ideal $S = C\ell_{p,q}f$ generated by a primitive idempotent f. Symbolic computations of such representations with `CLIFFORD` were shown in [3].

Following [42], let A be an $m \times n$ real matrix of rank r. Then the SVD of A is defined a factorization of A into a product of three matrices U, Σ, V^{-1} where U and V are orthogonal matrices $m \times m$ and $n \times n$ respectively, and Σ is a $m \times n$ matrix containing *singular values* of A on its "diagonal".

$$A = U\Sigma V^{-1}, \quad U^TU = I, \quad V^TV = I. \tag{19}$$

The matrices $V = [v_1|v_2|\ldots|v_n]$ and $U = [u_1|u_2|\ldots|u_m]$ contain orthonormal bases for all four fundamental spaces of A. Namely, the first r columns $v_1, v_2, \ldots, v_r$ of V provide a basis for the row space $\mathcal{R}(A^T)$ while the remaining $n-r$ columns of V provide a basis for the null space $\mathcal{N}(A)$. Likewise, the first r columns $u_1, u_2, \ldots, u_r$ of U provide a basis for the column space $\mathcal{C}(A)$ while the remaining $m-r$ columns of U provide a basis for the left-null space $\mathcal{N}(A^T)$. Vectors v_i are the normalized eigenvectors of A^TA while vectors u_i are the normalized eigenvectors of AA^T. For $i = 1, \ldots, r$, these vectors can be chosen to be related via the positive singular values σ_i of A which are just the square roots of the eigenvalues of A^TA (or of AA^T.) Namely,

$$Av_i = \sigma_i u_i, \quad i = 1, \ldots, r. \tag{20}$$

It is a little tricky to make sure that the above relation is satisfied: this is because the choice of vectors u_i is independent of the choice of vectors v_i. However, it is always possible to do so as we will see below (see also [42]). In order to complete the picture, the orthonormal set $\{v_1, \ldots, v_r\}$ needs to be completed to a full orthonormal basis for $\mathbb{R}^n$ while $\{u_1, \ldots, u_r\}$ needs to be completed to a full orthonormal basis for $\mathbb{R}^m$. Since the additional vectors are being annihilated by A and A^T respectively, that is, they are eigenvectors of A and A^T (or of A^TA and AA^T) that correspond to the eigenvalue 0, care has to be exercised when finding them. For example, while the eigenvectors of the symmetric matrix AA^T are automatically orthogonal provided they correspond to different eigenvalues, eigenvectors of AA^T that correspond to the 0 eigenvalue don't need to be orthogonal: in this case the Gram-Schmidt orthogonalization process is used to complete the two sets.

11.1. SVD of a 2×2 matrix of rank 2

In this section we present a simple example of SVD of a 2×2 real matrix of rank 2. The purpose of this example is just to show step by step how finding the SVD of a matrix can be done in the Clifford algebra language. Reader is encouraged to perform these computations with `CLIFFORD` and an additional package `asvd`.[20]

```
> A:=matrix(2,2,[2,3,1,2]); #defining A
```

[20] Package `asvd` contains the following procedures used in the text: `phi` that provides an isomorphism between a matrix algebra and a Clifford algebra; `radsimplify` that simplifies radical expressions in matrices and vectors; `assignL` that writes an output from a Maple procedure `eigenvects` in a suitable form: it sorts eigenvectors according to the corresponding eigenvalues

$$A := \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}$$

Since $A \in \mathrm{Mat}(2,\mathbb{R})$, we need to find (p,q) such that $C\ell_{p,q} \overset{\varphi}{\simeq} \mathrm{Mat}(2,\mathbb{R})$. Procedure `all_sigs` built into `CLIFFORD` displays two possible choices for the signature (p,q) such that $p+q=2$, $\mathbb{K} \simeq \mathbb{R}$ and $C\ell_{p,q}$ is a simple algebra:

```
> all_sigs(2..2,real,simple);
```

$$[[1,\ 1],\ [2,\ 0]]$$

Thus, we can pick either $C\ell_{1,1}$ or $C\ell_{2,0}$. Our choice is $C\ell_{2,0}$. We define B as the 2×2 identity matrix and use `CLIFFORD`'s procedure `clidata` to display information about $C\ell_{2,0}$.

```
> dim:=2:B:=diag(1,1):eval(makealiases(dim)):data:=clidata();
```

$$data := [real,\ 2,\ simple,\ \frac{1}{2}\,Id + \frac{1}{2}\,e1,\ [Id,\ e2],\ [Id],\ [Id,\ e2]]$$

The above output means that $C\ell_{2,0}$ is a simple algebra isomorphic to $\mathrm{Mat}(2,\mathbb{R})$; that the element $\frac{1}{2}+\frac{1}{2}\mathbf{e}_1$ displayed by Maple as $\frac{1}{2}Id+\frac{1}{2}e1$ is a primitive idempotent; that the list $[Id, e2]$ shown as the fifth entry displays generators of a minimal left-ideal $C\ell_{2,0}f$ considered as vector space over $\mathbb{R}$; that the division ring $\mathbb{K} = fC\ell_{2,0}f = \langle Id\rangle_{\mathbb{R}} \simeq \mathbb{R}$; and that the last list $[Id, e2]$ gives generators of $C\ell_{2,0}f$ over $\mathbb{K}$, and, since $\mathbb{K} \simeq \mathbb{R}$, it is the same as the fifth entry.[21] In the following, we define a Grassmann basis in $C\ell_{2,0}$, assign the primitive idempotent to f, and generate a spinor basis in $C\ell_{2,0}f$.

```
> clibas:=cbasis(dim); #ordered basis in Cl(2,0)
```

$$clibas := [Id,\ e1,\ e2,\ e12]$$

```
> f:=data[4]:#a primitive idempotent in Cl(2,0)
> SBgens:=data[5]:#generators for a real basis in S
> FBgens:=data[6]:#generators for the division ring K
```

`SBgens` contains generators for a $\mathbb{K}$-basis for $S = C\ell_{2,0}f = \langle f, \mathbf{e}_2 f\rangle$. Since in the signature $(2,0)$ we have $\mathbb{K} \simeq \mathbb{R}$, $S \simeq \mathbb{R}^2$, and $C\ell_{2,0} \simeq \mathrm{Mat}(2,\mathbb{R})$, the output from the procedure `spinorKbasis` shown below has two basis elements and their generators modulo f:

```
> Kbasis:=spinorKbasis(SBgens,f,FBgens,'left');
```

$$Kbasis := [[\frac{1}{2}\,Id + \frac{1}{2}\,e1,\ \frac{1}{2}\,e2 - \frac{1}{2}\,e12],\ [Id,\ e2],\ left]$$

Thus, the real spinor basis in S consists of the following two polynomials:

```
> for i to nops(Kbasis[1]) do f.i:=Kbasis[1][i] od;
```

$$f1 := \frac{1}{2}\,Id + \frac{1}{2}\,e1, \quad f2 := \frac{1}{2}\,e2 - \frac{1}{2}\,e12.$$

---

and it uses the Gram-Schmidt orthogonalization process, if necessary, to return a complete list of orthogonal eigenvectors; `makediag` makes a "diagonal" Σ matrix consisting of singular values.

[21] For more information see [3] and help pages in `CLIFFORD`.

We are in position now to compute matrices M_1, M_2, M_3, M_4 representing each of the four basis elements $\{\mathbf{1}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_{12}\}$ of $C\ell_{2,0}$ in the basis $\{f_1, f_2\}$.[22]

```
> for i to nops(clibas) do M[i]:=subs(Id=1,matKrepr(clibas[i])) end do:
```

We will use a new procedure `phi` which realizes the isomorphism φ from $\mathrm{Mat}(2,\mathbb{R})$ to $C\ell_{2,0}$. This way we can find the image $\mathtt{p} = \varphi(A)$ in $C\ell_{2,0}$ of any real 2×2 matrix A. Knowing image $\varphi(M_i)$ of each matrix M_i, $i = 1, \ldots, 4$, in terms of some Clifford polynomial in $C\ell_{2,0}$ we can easily find $\mathtt{p} = \varphi(A)$ as follows:[23]

```
> p:=phi(A,M); #finding image of A in Cl(2,0)
```

$$p := 2\,Id + 2\,e2 + e12$$

```
> pT:=phi(t(A),M); #finding image of t(A) in Cl(2,0)
```

$$pT := 2\,Id + 2\,e2 - e12$$

Next, we compute a symmetric matrix $A^T A$ (denoted in Maple as `ATA`), its characteristic polynomial, eigenvalues, and its orthonormal eigenvectors v_1, v_2. Vectors v_1 and v_2 will become columns of an orthogonal matrix V needed for the SVD of A:

```
> ATA:=evalm(t(A) &* A); #finding matrix ATA
```

$$ATA := \begin{bmatrix} 5 & 8 \\ 8 & 13 \end{bmatrix}$$

```
> pTp:=phi(ATA,M); #finding image of ATA in Cl(2,0)
```

$$pTp := 9\,Id - 4\,e1 + 8\,e2$$

which is the same as

```
> 'pTp'=cmul(pT,p);
```

$$pTp = 9\,Id - 4\,e1 + 8\,e2$$

The minimal polynomial of $A^T A$, whose image in $C\ell_{2,0}$ is called `pTp`, is computed with `climinpoly`:

```
> climinpoly(pTp);
```

$$x^2 - 18\,x + 1$$

and it is the same as the characteristic polynomial of $A^T A$:

```
> pol:=charpoly(ATA,x);#characteristic polynomial of ATA
```

$$pol := x^2 - 18\,x + 1$$

---

[22] Since a similar computation was done in [2, 3], we won't display the matrices.

[23] From now on, we use Maple's `alias(t = transpose)`, that is, `t(A)` denotes matrix transposition in Maple. The second argument to the procedure `phi` is a table M containing matrices M_1, M_2, M_3, M_4 as its entries.

In order to find eigenvalues and eigenvectors of A^TA, we will use Maple's procedure `eigenvects` modified by our own sorting via a new procedure `assignL`. The latter displays a list containing two lists: one has the eigenvalues while the second has the eigenvectors.[24] In the following, we assign the eigenvalues of A^TA to λ_1, λ_2 and the not-yet-normalized but orthogonal eigenvectors of A^TA we assign to `v1`, `v2`.

```
> P:=assignL(sort([eigenvects(ATA)],byeigenvals));N:=P[1]:
```

$$P := [2,\ [9+4\sqrt{5},\ 9-4\sqrt{5}],\ [\left[1,\ \frac{1}{2}+\frac{1}{2}\sqrt{5}\right],\ \left[1,\ \frac{1}{2}-\frac{1}{2}\sqrt{5}\right]]]$$

```
> for i to N do lambda.i:=P[2][i]; v.i:=map(simplify,normalize(P[3][i]))
> end do:
```

We can now verify that vectors v_1, v_2 are eigenvectors of A^TA with the eigenvalues λ_1, λ_2.

```
> for i to N do map(simplify,evalm(ATA &* v.i - lambda.i*v.i)) od;
```

$$[0,\ 0],\ [0,\ 0]$$

Similar verification can be done in $C\ell_{2,0}$ since one can view the 1-column eigenvectors v_1, v_2 as one-column spinors in $S = C\ell_{2,0}f$. We simply convert the two vectors v_1, v_2 to spinors $\mathtt{sv_1} = \varphi(v_1)$, $\mathtt{sv_2} = \varphi(v_2)$ which we express in the previously computed spinor basis f_1, f_2.

```
> spinorbasis:=[''f1'',''f2'']:
> for i to N do sv.i:=convert(v.i,spinor,spinorbasis) od;
```

$$sv1 := 2\,\frac{f1}{\sqrt{10+2\sqrt{5}}} + \frac{(1+\sqrt{5})\,f2}{\sqrt{10+2\sqrt{5}}},\ sv2 := 2\,\frac{f1}{\sqrt{10-2\sqrt{5}}} - \frac{(-1+\sqrt{5})\,f2}{\sqrt{10-2\sqrt{5}}}.$$

Since v_1, v_2 are the eigenvectors of A^TA, spinors $\mathtt{sv_1}$, $\mathtt{sv_2}$ must be the eigenspinors of $\mathtt{pTp} = \varphi(A^TA)$. This fact can be verified as follows:

```
> for i to N do simplify((pTp - lambda.i) &c sv.i) od;
```

$$0,\ 0$$

We define now the orthogonal matrix $V = [v_1|v_2]$ whose entries we will simply with the procedure `radsimplify`.

```
> V:=radsimplify(augment(v.(1..N))); #defining matrix V
```

$$V := \begin{bmatrix} 2\,\dfrac{1}{\sqrt{10+2\sqrt{5}}} & 2\,\dfrac{1}{\sqrt{10-2\sqrt{5}}} \\ \dfrac{1+\sqrt{5}}{\sqrt{10+2\sqrt{5}}} & \dfrac{-\sqrt{5}+1}{\sqrt{10-2\sqrt{5}}} \end{bmatrix}$$

[24] The first entry 2 in the output is just the number of eigenvectors.

Since later we will need images of V and V^T under φ in $C\ell_{2,0}$, we compute them now and store them under the variables `pV` and `pVt` respectively.

```
> pV:=phi(V,M); #finding image of V in Cl(2,0)
> pVt:=phi(t(V),M): #finding image of t(V) in Cl(2,0)
```

$$pV := (-\frac{1}{20}\,\%1\sqrt{5} - \frac{1}{40}\,\%2\sqrt{5} + \frac{1}{8}\,\%2)\,Id + (\frac{1}{20}\,\%1\sqrt{5} - \frac{1}{40}\,\%2\sqrt{5} + \frac{1}{8}\,\%2)\,e1 + (\frac{1}{20}\,\%2\sqrt{5} + \frac{1}{40}\,\%1\sqrt{5} + \frac{1}{8}\,\%1)\,e2 + (-\frac{1}{20}\,\%2\sqrt{5} + \frac{1}{40}\,\%1\sqrt{5} + \frac{1}{8}\,\%1)\,e12$$

$$\%1 := \sqrt{10 - 2\sqrt{5}}$$

$$\%2 := \sqrt{10 + 2\sqrt{5}}$$

The fact that V is orthogonal can be easily verified in the matrix language; in $C\ell_{2,0}$ it can be done as follows:

```
> simplify(cmul(pVt,pV));
```

$$Id$$

We repeat the above steps and apply them to AA^T. In the process, we will find its eigenvectors u_1, u_2. We must make sure that $Av_i = \sigma_i u_i$ where $\sigma_i = \sqrt{\lambda_i}$, $i = 1, 2$. This will require extra checking and possibly redefining of the u's.

```
> AAT:=evalm(A &* transpose(A)); #computing AAT
```

$$AAT := \begin{bmatrix} 13 & 8 \\ 8 & 5 \end{bmatrix}$$

The image of AA^T under φ in $C\ell_{2,0}$ we denote as `ppT`.

```
> ppT:=phi(AAT,M); #finding image of AAT in Cl(2,0)
```

$$ppT := 9\,Id + 4\,e1 + 8\,e2$$

In this case, the minimal polynomial of `ppT` and the characteristic polynomial of AA^T are, of course, the same.

```
> pol2:=charpoly(AAT,lambda); #characteristic polynomial of AAT
```

$$pol2 := \lambda^2 - 18\,\lambda + 1$$

```
> 'ppT'=climinpoly(ppT);
```

$$ppT = x^2 - 18\,x + 1$$

Since matrices A^TA and AA^T have the same characteristic polynomials, their eigenvalues λ_1, λ_2 will be the same. We define therefore the *singular values* σ_1 and σ_2 of A as the square roots of λ_1 and λ_2.

```
> for i to N do sigma.i:=sqrt(lambda.i) od;
```

$$\sigma 1 := \sqrt{5} + 2, \quad \sigma 2 := \sqrt{5} - 2$$

When we compute the eigenvectors u_1, u_2 of AA^T, we will not necessarily have $Av_i = \sigma_i u_i$, $i = 1, 2$. This is because the choice of u_1, u_2 is not consistent with the choice of v_1, v_2.

```
> P:=assignL(sort([eigenvects(AAT)],byeigenvals)):
> for i to N do lambda.i:=P[2][i];
> u.i:=map(simplify,normalize(P[3][i])) end do:
```

Notice that $Av_1 = \sigma_1 u_1$ while $Av_2 = -\sigma_2 u_2$:

```
> radsimplify(evalm(A &* v1-sigma1*u1)); #this checks out
```

$$[0,\ 0]$$

```
> radsimplify(evalm(A &* v2-sigma2*u2)); #this does not
> radsimplify(evalm(A &* v2+sigma2*u2)); #this checks out
```

$$\left[\frac{14-6\sqrt{5}}{\sqrt{10-2\sqrt{5}}},\ \frac{8-4\sqrt{5}}{\sqrt{10-2\sqrt{5}}}\right]$$

$$[0,\ 0]$$

Let's re-define u_2 as $-u_2$ and call it u_{22}, and let's also rename u_1 as u_{11}:

```
> u11:=evalm(u1):u22:=evalm(-u2):
```

Then we will have $Av_{ii} = \sigma u_{ii}$, $i = 1, 2$.

In the Clifford algebra $C\ell_{2,0}$ we need to perform similar computations with v_1, v_2. The elements $\mathtt{su_1} = \varphi(u_{11})$, $\mathtt{su_2} = \varphi(u_{22})$ contained in the spinor ideal $S = C\ell_{2,0}f$ need to be found first.

```
> for i to N do su.i:=convert(u.i.i,spinor,spinorbasis) od;
```

$$su1 := \frac{(1+\sqrt{5})\,f1}{\sqrt{10+2\sqrt{5}}} + 2\,\frac{f2}{\sqrt{10+2\sqrt{5}}},\ su2 := \frac{(-1+\sqrt{5})\,f1}{\sqrt{10-2\sqrt{5}}} - 2\,\frac{f2}{\sqrt{10-2\sqrt{5}}}.$$

The verification of the condition (20) in $C\ell_{2,0}$ looks as follows:

```
> for i to N do simplify(p &c sv.i-sigma.i*su.i) od;
```

$$0,\ 0$$

Now we may define the orthogonal matrix $U = [u_{11}|u_{22}]$ and its image $\varphi(U)$ in $C\ell_{2,0}$ which we assign to $\mathtt{pU}$:[25]

```
> U:=radsimplify(augment(u11,u22)); #defining matrix U
```

$$U := \begin{bmatrix} \dfrac{1+\sqrt{5}}{\sqrt{10+2\sqrt{5}}} & \dfrac{-1+\sqrt{5}}{\sqrt{10-2\sqrt{5}}} \\ 2\,\dfrac{1}{\sqrt{10+2\sqrt{5}}} & -2\,\dfrac{1}{\sqrt{10-2\sqrt{5}}} \end{bmatrix}$$

[25] Expressions %1 and %2 showing up in the Maple output for pU are just place holders for $\sqrt{10+2\sqrt{5}}$ and $\sqrt{10-2\sqrt{5}}$ respectively and are as shown at the end of the display.

```
> pU:=phi(U,M);#finding image of U in Cl(2,0)
> pUt:=phi(t(U),M):#finding image of t(U) in Cl(2,0)
```

$$pU := (\frac{1}{20}\,\%1\sqrt{5} - \frac{1}{8}\,\%2 - \frac{1}{40}\,\%2\sqrt{5})\,Id + (\frac{1}{20}\,\%1\sqrt{5} + \frac{1}{8}\,\%2 + \frac{1}{40}\,\%2\sqrt{5})\,e1 + (\frac{1}{20}\,\%2\sqrt{5} + \frac{1}{8}\,\%1 - \frac{1}{40}\,\%1\sqrt{5})\,e2 + (\frac{1}{20}\,\%2\sqrt{5} - \frac{1}{8}\,\%1 + \frac{1}{40}\,\%1\sqrt{5})\,e12$$

$$\%1 := \sqrt{10 + 2\sqrt{5}}$$

$$\%2 := \sqrt{10 - 2\sqrt{5}}$$

The fact that U is an orthogonal matrix can be easily now checked both in the matrix language and in the Clifford language:

```
> radsimplify(evalm(t(U) &* U));#U is an orthogonal matrix
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

```
> simplify(pUt &c pU);
```

$$Id$$

Finally, we define matrix Σ using a procedure `makediag`. Recall from [42] that Σ has the same dimensions as the original matrix A and that $\Sigma^T\Sigma, \Sigma\Sigma^T$ are the diagonal forms of A^TA and AA^T respectively. In this example matrices $\Sigma^T\Sigma$ and $\Sigma\Sigma^T$ are the same since Σ is a square diagonal matrix. Normally these matrices are different although their nonzero "diagonal" entries are the same. Therefore we have

$$\begin{gathered} A^TA = V\Sigma^T\Sigma V^T, \quad AA^T = U\Sigma\Sigma^T U^T, \quad \Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}, \\ \Sigma^T\Sigma = \Sigma\Sigma^T = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}. \end{gathered} \tag{21}$$

Matrices Σ, $\Sigma^T\Sigma$ and $\Sigma\Sigma^T$ we assign to Maple variables `Sigma`, `STS` and `SST` respectively:

```
> Sigma:=makediag(m,n,[seq(sigma.i,i=1..N)]);
> STS,SST:=evalm(t(Sigma) &* Sigma),evalm(Sigma &* t(Sigma));
```

$$\Sigma := \begin{bmatrix} \sqrt{5}+2 & 0 \\ 0 & \sqrt{5}-2 \end{bmatrix}$$

$$STS,\ SST := \begin{bmatrix} (\sqrt{5}+2)^2 & 0 \\ 0 & (\sqrt{5}-2)^2 \end{bmatrix}, \begin{bmatrix} (\sqrt{5}+2)^2 & 0 \\ 0 & (\sqrt{5}-2)^2 \end{bmatrix}$$

The corresponding images $\varphi(\Sigma)$, $\varphi(\Sigma^T\Sigma)$ and $\varphi(\Sigma\Sigma^T)$ and $C\ell_{2,0}$ will be assigned to the Maple variables `pSigma`, `pSTS` and `pSST` respectively:

```
> pSigma,pSTS,pSST:=phi(Sigma,M),phi(STS,M,FBgens),phi(SST,M);
```

$$pSigma,\ pSTS,\ pSST := \sqrt{5}\,Id + 2\,e1,\ 9\,Id + 4\sqrt{5}\,e1,\ 9\,Id + 4\sqrt{5}\,e1$$

We should be able to verify in $C\ell_{2,0}$ the following two factorizations of AA^T and A^TA:

$$A^TA = V\Sigma^T\Sigma V^T \quad (22)$$
$$AA^T = U\Sigma\Sigma^T U^T \quad (23)$$

like this:

```
> evalb(pTp=simplify(pV &c pSTS &c pVt)),
> evalb(ppT=simplify(pU &c pSST &c pUt));
```

$$true,\ true$$

Finally, we check the SVD of A, which is $A = U\Sigma V^T$,[26] in the Clifford algebra language:

```
> evalb(p=simplify(pU &c pSigma &c pVt));
```

$$true$$

11.2. Additional comments

In the previous section we have shown that it is possible to translate the matrix algebra picture of the Singular Value Decomposition of a matrix A into the Clifford algebra language. Although we have not abandoned entirely the linear algebra formalism in our example, e.g., we have computed the eigenvalues and the eigenvectors of A^TA and AA^T, and only then we have found images of their eigenvectors in the spinor space $C\ell_{2,0}f$, these computations including solving the eigenvalue problem can be done entirely in the Clifford algebra language. In the following we show how this can be accomplished.

The first comment is that it is also possible to compute the eigenvalues λ_1, λ_2 and the eigenspinors $\mathtt{sv_1} = \varphi(v_1)$, $\mathtt{sv_2} = \varphi(v_2)$ of the Clifford number $\mathtt{pTp} = \varphi(A^TA)$ directly in $C\ell_{2,0}$ using the procedure `clisolve` capable of solving algebraic equations.

```
> eigenspinor:=x1*f1+x2*f2;
> eigeneq:=clicollect(expand(cmul(pTp,eigenspinor)-lambda*eigenspinor));
```

$$eigenspinor := x1\,(\frac{Id}{2} + \frac{e1}{2}) + x2\,(\frac{e2}{2} - \frac{e12}{2})$$

$$eigeneq := -\frac{(-5\,x1 - 8\,x2 + \lambda\,x1)\,e1}{2} - \frac{(-5\,x1 - 8\,x2 + \lambda\,x1)\,Id}{2} + \frac{(\lambda\,x2 - 13\,x2 - 8\,x1)\,e12}{2} - \frac{(\lambda\,x2 - 13\,x2 - 8\,x1)\,e2}{2}$$

[26] The SVD of A is not unique: For example, $A = (-U)\Sigma(-V^T)$ is another such factorization.

```
> sol:=remove(has,map(allvalues,clisolve(eigeneq,[lambda,x1,x2])),
> lambda=lambda);
```

$$sol := [\{\lambda = 9 + 4\sqrt{5},\ x2 = x2,\ x1 = \frac{(9+4\sqrt{5})\,x2}{8} - \frac{13\,x2}{8}\},$$
$$\{\lambda = 9 - 4\sqrt{5},\ x2 = x2,\ x1 = \frac{(9-4\sqrt{5})\,x2}{8} - \frac{13\,x2}{8}\}]$$

```
> for i from 1 to nops(sol) do;
> lambda||i:=subs(sol[1],lambda);
> eigenspinor||i:=clicollect(subs({x1=1,x2=1},subs(sol[1],eigenspinor)));
> end do;
```

$$\lambda 1 := 9 + 4\sqrt{5}$$

$$eigenspinor1 := \frac{(-1+\sqrt{5})\,e1}{4} + \frac{(-1+\sqrt{5})\,Id}{4} - \frac{e12}{2} + \frac{e2}{2}$$

$$\lambda 2 := 9 + 4\sqrt{5}$$

$$eigenspinor2 := \frac{(-1+\sqrt{5})\,e1}{4} + \frac{(-1+\sqrt{5})\,Id}{4} - \frac{e12}{2} + \frac{e2}{2}$$

The second comment is that it is possible to realize an anti-automorphism $tp : C\ell(Q) \to C\ell(Q)$ that acts on homogeneous basis multivectors as follows:

$$\mathbf{e}_{i_1} \wedge \mathbf{e}_{i_2} \wedge \cdots \wedge \mathbf{e}_{i_n} \mapsto Q_{i_1 i_1} Q_{i_2 i_2} \cdots Q_{i_n i_n} (\mathbf{e}_{i_1} \wedge \mathbf{e}_{i_2} \wedge \cdots \wedge \mathbf{e}_{i_n})\tilde{} \qquad (24)$$

where $(\mathbf{e}_{i_1}\wedge\mathbf{e}_{i_2}\wedge\cdots\wedge\mathbf{e}_{i_n})\tilde{} = (-1)^{\frac{n(n-1)}{2}}\,\mathbf{e}_{i_1}\wedge\mathbf{e}_{i_2}\wedge\cdots\wedge\mathbf{e}_{i_n}$ is the reversed multivector while $Q_{i_1 i_1}, Q_{i_2 i_2}, \ldots, Q_{i_n i_n}$ are the diagonal entries of the quadratic form Q (to be precise, $Q_{ii} = \mathbf{e}_i^2$, $i = 1, \ldots, n$), extended by linearity to the whole algebra. Then, tp gives the transposition of the corresponding spinor representation.[27] For a thorough treatment of the singular value decomposition for Grassmann and Clifford algebras in the Hopf algebraic language see [23].

12. Conclusions

The aim of this note was to present a few examples of computations with Clifford algebras where symbolic capabilities of Maple and the `CLIFFORD`, `BIGEBRA` packages were utilized. In some, a better understanding of the Clifford algebra structure and its properties, especially of the Clifford algebras $C\ell(B)$ of an arbitrary bilinear form B, was achieved (see also [6]). Gaining a computational proficiency with Clifford algebras of a quadratic form notwithstanding, such as their spinor representations or computation of bilinear forms on spinor spaces, properties of the contraction, the reversion, the wedge, the dotted wedge, and the Clifford products, all curiously dependent on the antisymmetric form of B have been successfully formulated. In others, a discovery of specific elements was accomplished like finding,

[27] This has been verified by brute force with `CLIFFORD` in dimensions up to 9 for simple $C\ell_{p,q}$ and such that $(p-q) \equiv 0, 1, 2 \pmod 8$. That is, for the Clifford algebras whose spinor representations are real. A Maple code for a procedure `tp` is shown in Appendix C.

unexpectedly continuous, families of idempotents or a computation of eigenvalues and eigenvectors of a Clifford number, or its singular values. In the process, a connection between a transposition and the reversion in $C\ell(Q)$ was discovered.

Some other computational and theoretical results facilitated with the packages and already reported include

- Finding generators for Hecke algebras realized as even elements in a Clifford algebra of a suitable, non symmetric, bilinear form. [7]
- Finding q-Young idempotents in some Hecke algebras of mixed symmetry that generate a representation space for these algebras. [7]
- Finding necessary and sufficient condition that a Clifford biconvolution, that is, a Clifford Hopf gebra for a two-dimensional real space posses an antipode. [25]
- Investigation of Wick normal ordering [21]
- Verification of Helmstetter formula [28] that expresses an isomorphism between two Clifford algebras $C\ell(B)$ and $C\ell(B')$ where the bilinear forms share the symmetric part. [4]
- Explicit description of all elements in **Pin**(3) and **Spin**(3) in [4]
- Description of space singularities of a robotic platform [5, 17]

For a complete discussion of the mathematical capabilities of `CLIFFORD` and `BIGEBRA` packages we refer to [9] and [10].

Appendix A. Appendix: Code of `cmulNUM`

Here is a pseudocode of the recursive procedure `cmulNUM`.

```
cmulNUM(a1,a2,B) [a1, a2 - two Grassmann monomials, B - name of bilinear form]
begin
  if nargs <>3 then error "exactly three arguments are needed" end if
  if has(0,map(simplify,[a1,a2])) then return 0 end if
  if a2='Id' then return a1 end if
  if a1='Id' then return a2 end if
L <- indices from a1
N <- length of L
coB,nameB <- coefficient of B, B [to handle -B]
  if N=0 then return coeff(a1,Id)*a2 elif N=1 then
L2 <- list of indices from a2
  return reorder(simplify(makeclibasmon([L[1],op(L2)])
       +add((-1)^(i-1)*coB*nameB[L[1],L2[i]]*makeclibasmon(subs(L2[i]=NULL,L2)),
            i=1..nops(L2))))
    elif N=2 then
      x1 <- substring(a1,1..2)
      x2 <- substring(a1,4..5)
      p2 <- procname(x2,a2,B)
      S <- clibilinear(x1,p2,procname,B)
      return simplify(S-coB*nameB[op(L)]*a2)
  end if;
```

```
x <- cat(e,L[-1])
p1 <- substring(a1,1..(3*N-4))
p2 <- procname(x,a2,B)
S <- clibilinear(p1,p2,procname,B)
     -add((-1)^(i)*coB*nameB[L[-i],L[-1]]*
          procname(makeclibasmon(subs(L[-i]=NULL,L[1..-2])),a2,B),i=2..N)
return reorder(simplify(S))
end cmulNUM
```

Appendix B. Appendix: Code of `cmulRS`

Here is a pseudocode of the procedure `cmulRS` based on the combinatorial process of Rota-Stein:

```
cmulRS(x,y,B) [x, y two Grassmann monomials, B - bilinear form]
begin
  lstx <- list of indices from x
  lsty <- list of indices from y
  NX <- length of lstx
  NY <- length of lsty
  funx <- function maps integers 1..NX onto elements of lstx keeping their order
  funy <- function maps integers 1..NY onto elements of lsty keeping their order
    (this is to calculate with arbitrary indices and to compute necessary signs)
  psetx <- power set of 1..NX (actually a list in a certain order)
    (the i-th and (2^NX+1-i)-th element are disjoint adding up to the set {1..NX})
  psety <- power set of 1..NY (actually a list in a certain order)
    (the i-th and (2^NY+1-i)-th element are disjoint adding up to the set {1..NY})
    (for faster computation we sort this power sets by grade)
    (we compute the sign for any term in the power set)
  psetx <- sort psetx by grade
  psety <- sort psety by grade
  pSgnx <- sum_(i in psetx) (-1)^sum_(j in psetx[i]) (psetx[i][j]-j)
  pSgny <- sum_(i in psety) (-1)^sum_(j in psety[i]) (psety[i][j]-j)
  (we need a subroutine for cup tangle computing the bilinear form cup(x,y,B))
    begin cup
      if |x| <> |y| then return 0 end if
      if |x| = 0 then return 1 end if
      if |x| = 1 then return B[x[1],y[1]] end if
      return sum_(j in 1..|x|)(-1)^(j-1)*B(x[1],y[j])*cup(x[2..-1],y/y[j],B)
    end cup
  (now we compute the double sum, to gain efficiency we do this grade wise)
  (note that there are r over NX r-vectors in psetx, analogously for psety)
  max_grade - |lstx <- convert_to_set union lsty <- convert_to_set|
  res <- 0, pos1 <- 0
  for j from 0 to NX (iterate over all j-vectors of psetx)
    begin
      F1 <- N1!/((N1-j)!*j!) (number of terms (N1 over j))
```

```
    pos2 <- 0
   for i from 0 to min(N2,max_grade-j)
   (iterate over all i-vectors of psety not exceeding max_grade while others are zero)
   begin
    F2 <- N2!/((N2-i)!*i!) (number of terms (N2 over i))
   for n from 1 to F1 (for all j-vectors)
    begin
     for m from 1 to F2 (for all i-vectors)
      begin
      res <- res + pSgnx[pos1+n]*pSgny[pos2+m]*
           cup(fun1(psetx[PN1-pos1-n]),fun2(psety[pos2+m]),lname)*
           makeclibasmon -> ([fun1 -> psetx[pos1+n],fun2 -> psety[PN2-pos2-m]])
     end
   end
    pos2 <- pos2F2
  end
   pos1 <- pos1F1
 end
 reorder -> res (reorder basis elements in res into standard order)
 end cmulRS
```

Appendix C. Appendix: Code of the Transposition Procedure `tp`

Here is a code of the procedure `tp` that accomplishes the 'transposition' anti-automorphism in $C\ell_{p,q}$.

```
tp:=proc(xx) local x,L,p,co,u:
  x:=displayid(xx):
  if type(x,clibasmon) then
    if x=Id then Id else
      p:=op(cliterms(x));
      L:=extract(p,'integers'): #list L of indices
      L:=[seq(L[nops(L)-i+1],i=1..nops(L))]; #reversed list L
      u:=cmul(seq(B[L[i],L[i]]*cat(e,L[i]),i=1..nops(L)));
      reorder(u)
    end if:
  elif type(x,climon) then
    p:=op(cliterms(x)):
    co,p:=coeff(x,p),p;
    co*procname(p);
  elif type(x,clipolynom) then
    u:=clilinear(x,procname);
  end if:
end tp:
```

References

[1] R. Abłamowicz, *Clifford algebra computations with Maple*. In Clifford (Geometric) Algebras with Applications in Physics, Mathematics, and Engineering, W. E. Baylis, ed. (Birkhäuser, Boston, 1996) 463–502

[2] R. Abłamowicz, *Matrix exponential via Clifford algebras*. J. of Nonlinear Math. Phys., **5**, 3 (1998) 294–313

[3] R. Abłamowicz, *Spinor Representations of Clifford Algebras: A Symbolic Approach*. CPC Thematic Issue - Computer Algebra in Physics Research, Phys. Comm. **115** (1998) 510–535

[4] R. Abłamowicz, *Helmstetter formula and rigid motions with* `CLIFFORD`. In Advances in Geometric Algebra with Applications in Science and Engineering – Automatic Theorem proving, Computer Vision, Quantum and Neural Computing, and Robotics, E. Bayro-Corrochano and G. Sobczyk, eds., (Birkhäuser, Boston, 2001) 512–534

[5] R. Abłamowicz, J. Anderson and M. Baswell, *Clifford algebra space singularities of inline planar platforms*. In Applications of Geometric Algebra in Computer Science and Engineering, L. Dorst, ed., Chapter 36 (Birkhäuser, Boston, 2002) 463–502

[6] R. Abłamowicz and B. Fauser, *On the Decomposition of Clifford Algebras of Arbitrary Bilinear Form*. In Clifford Algebras and their Applications in Mathematical Physics, R. Abłamowicz and B. Fauser, eds. (Birkhäuser, Boston, 2000) 341–366

[7] R. Abłamowicz and B. Fauser, *Hecke algebra representations in ideals generated by q-Young Clifford idempotents*. In Clifford Algebras and their Applications in Mathematical Physics, R. Abłamowicz and B. Fauser, eds., Vol. 1: Algebra and Physics, (Birkhäuser, Boston, 2000) 245–268

[8] R. Abłamowicz and B. Fauser, `CLIFFORD` and `BIGEBRA` for Maple, `http://math.tntech.edu/rafal/` (2009)

[9] R. Abłamowicz and B. Fauser, *Mathematics of* `CLIFFORD` *- A Maple Package for Clifford and Grassmann Algebras*. Advances in Applied Clifford Algebras, Vol. **15**, No. 2 (2005), 157–181

[10] R. Abłamowicz and B. Fauser, *Clifford and Grassmann Hopf algebras via the* `BIGEBRA` *package for Maple*. Computer Physics Communications **170** (2005) 115–130

[11] R. Abłamowicz, B. Fauser, K. Podlaski and J. Rembieliński, *Idempotents of Clifford algebras*. Czech. J. Phys. **53**, 11 (2003) 949–955

[12] R. Abłamowicz and P. Lounesto, *On Clifford algebras of a bilinear form with an antisymmetric part*. In Clifford Algebras with Numeric and Symbolic Computations, R. Abłamowicz, P. Lounesto, and J. Parra, eds. (Birkhäuser, Boston, 1996) 167–188

[13] R. Abłamowicz and G. Sobczyk, *Software for Clifford (geometric) algebras*. Appendix in Lectures on Clifford (Geometric) Algebras and Applications, R. Abłamowicz and G. Sobczyk, eds. (Birkhäuser, Boston, 2004) 189–209

[14] J.L. Anderson, *Green's functions in quantum electrodynamics*, Phys. Rev. **171**, 94 (1954) 703–11

[15] P. Angles, *Construction de revêtements du groupe conforme d'un espace vectoriel muni d'uni métrique de type* (p, q). Ann. Inst. Henri Poincareé **XXXIII**, 1 (10980) 33–51

[16] *Axiom Computer Algebra System*, `http://en.wikipedia.org/wiki/AXIOM` (2009)

[17] M. Baswell, *Clifford Algebra Space Singularities of a Redundant Variable Geometry Truss Manipulator*. M.S. Thesis, Tennessee Technological University, 2000

[18] *CoCoA System for computations in commutative algebra*, `http://cocoa.dima.unige.it/` (2009)

[19] M. Berry and J. Dongarra, *Atlanta Organizers Put Mathematics to Work for the Math Sciences Community*, SIAM News, Vol. **32**, 6 (1999), and references therein.

[20] Ch. Brouder, B. Fauser, A. Frabetti and R. Oeckl, *Quantum field theory and Hopf algebra cohomology* [formerly 'Let's twist again']. J. Phys. A: Math. Gen: **37**, 22 (2004) 5895–5927

[21] B. Fauser, *Clifford geometric parameterization of Wick normal-ordering*. J. Phys. A: Math. Gen. **34** (2001) 105–115

[22] B. Fauser, *A Treatise on Quantum Clifford Algebras* (Habilitationsschrift, Universität Konstanz, Konstanz, 2002)

[23] B. Fauser, *Products, coproducts and singular value decomposition*. Int. J. Theor. Phys. Vol. **45**, No. 9 (2006) 1731–1755

[24] B. Fauser and P.D. Jarvis, *A Hopf laboratory for symmetric functions*. J. Phys. A: Math. Gen. **37** (2004) 1633–1663

[25] B. Fauser, and Z. Oziewicz, *Clifford Hopf gebra for two-dimensional space*. Miscellanea Algebraica **2**, 1 (2001) 31–42

[26] G.-M. Greuel and G. Pfister, *A* `Singular` *Introduction to Commutative Algebra*. (Springer, New York, 2000)

[27] J. Helmstetter, *Algèbres de Clifford et algèbres de Weyl*. Cahiers Math **25** (1982)

[28] J. Helmstetter, *Monoïdes de Clifford et déformations d'algèbres de Clifford*. J. of Alg. *111*, 1 (1987) 14–48

[29] T.Y. Lam, *The Algebraic Theory of Quadratic Forms*. (The Benjamin Cummings Publishing Company, Reading, 1973)

[30] P. Lounesto, *Scalar products of spinors and an extension of Brauer-Wall groups*. Found. Phys. **11** (1981) 721–740

[31] P. Lounesto and E. Latvamaa, *Conformal transformations and Clifford algebras*. Proceedings of the American Mathematical Society, **79**, 4 (1980) 533–538

[32] P. Lounesto, R. Mikkola and V. Vierros, `CLICAL` *User Manual*. Helsinki University of Technology, Institute of Mathematics, Research Reports **A248** (1987)

[33] P. Lounesto, `CLICAL` *and counter-examples*. In Clifford Algebras with Symbolic and Numeric Computations, R. Abłamowicz, P. Lounesto and J. Parra (Birkhäuser, Boston, 1996) 3–30

[34] P. Lounesto and A. Springer, *Möbius transformations and Clifford algebras of Euclidean and anti-Euclidean spaces*. In Deformations of Mathematical Physics, J. Ławrynowicz (Kluwer Academic Publishers, 1989) 79–90

[35] P. Lounesto, *Clifford Algebras and Spinors*. 2nd ed. (Cambridge University Press, Cambridge, 2001)

[36] `Macaulay 2`*: A software system for research in algebraic geometry and commutative algebra*, `http://www.math.uiuc.edu/Macaulay2/` (2009)

[37] J.M. Maciejowski; *Multivariable Feedback Design* (Addison-Wesley, Wokingham, England, 1989)

[38] J. Maks, *Modulo (1,1) periodicity of Clifford algebras and the generalized (anti-) Möbius transformations*. Thesis, Technische Universiteit Delft, 1989

[39] I. Porteous, *Topological Geometry* (Van Nostrand-Reinhold, London, 1969)

[40] G.-C. Rota and J.A. Stein, *Plethystic Hopf algebras*. Proc. Natl. Acad. Sci. USA **91** (1994) 13057–13061

[41] J.M. Selig, *Geometrical Methods in Robotics* (Springer Verlag, New York, 1996)

[42] G. Strang, *Introduction to Linear Algebra* (Wellesley-Cambridge Press, Wellesley, 1998)

[43] S. Tunney, *On the Continuous Families of Idempotents in the Clifford Algebra of the Euclidean Plane*. M.S. Thesis, Tennessee Technological University, 2003

[44] Waterloo Maple Incorporated, *Maple, a general purpose computer algebra system*. Waterloo, `http://www.maplesoft.com` (2009)

[45] F. Wright, *Computing with Maple* (Chapman & Hall/CRC, Boca Raton, 2002)

Rafał Abłamowicz
Department of Mathematics, Box 5054
Tennessee Technological University
Cookeville, TN 38505, USA
e-mail: `rablamowicz@tntech.edu`