An Algorithm for Construction of Non-degenerate Clifford Algebra Matrix Representations in Computer Algebra Systems

Dimiter Prodanov[1,2]

[1]IMEC, [2]BAS

ICCA12, 3–7 Aug 2020, Hefei

Motivation

Applications

- General Clifford multivector inverse
- Automatic computer code generation for the lower-dimensional algebras
- Numerical algorithms for Matlab, Octave, C++, Java – "Cliffordization"

Motivation
The Maxima Computer Alebra System
Clifford algebras in Maxima
Examples: multiplication tables
Demonstrations

Applications

- General Clifford multivector inverse
- Automatic computer code generation for the lower-dimensional algebras
- Numerical algorithms for Matlab, Octave, C++, Java – "Cliffordization"

Motivation
The Maxima Computer Alebra System
Clifford algebras in Maxima
Examples: multiplication tables
Demonstrations

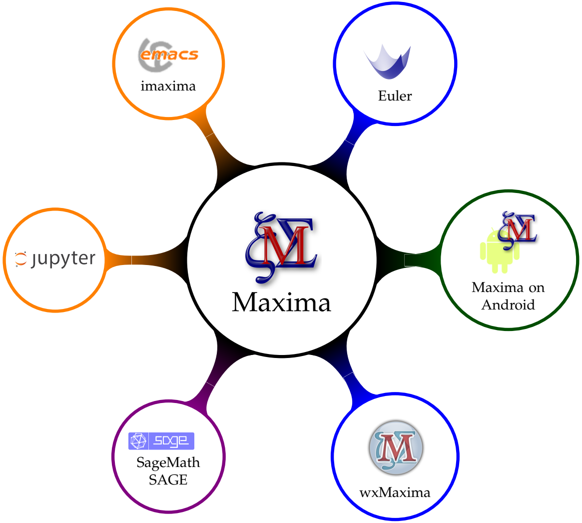
Applications

- General Clifford multivector inverse
- Automatic computer code generation for the lower-dimensional algebras
- Numerical algorithms for Matlab, Octave, C++, Java – "Cliffordization"

Motivation
The Maxima Computer Alebra System
Clifford algebras in Maxima
Examples: multiplication tables
Demonstrations

The Maxima Computer Alebra System

Why Maxima?

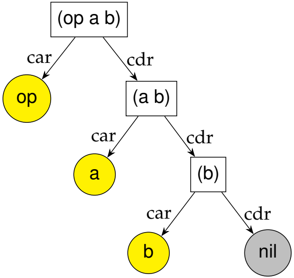


Maxima is the open source descendant of the first ever computer algebra system MACSYMA.

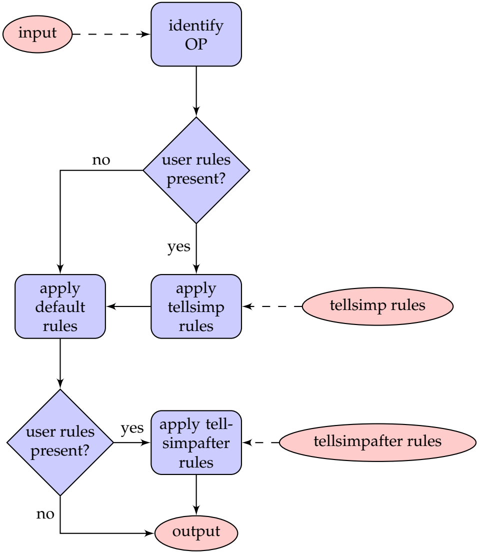
- Maxima is widely used
- open source allows for fast development cycles
- bugs are corrected quickly.

Expression simplification in Maxima

Parse tree Lisp representation



- car – first element (map atom)
- cdr – rest (a new list)



Clifford algebras in Maxima

Elementary construction of Clifford algebras

We assume everywhere a base field of characteristic 0!

- Choose a generator symbol **e** and adjoin an index $k \leq n$ to the symbol $e \mapsto e_k$ producing a set of n **basis vectors** $E := \{e_1 \dots e_n\} \subset \mathbb{G}^n$.
- Assign a canonical lexicographic order $\prec$ over E, such that $i < j \implies e_i \prec e_j$.
- Define the associative and distributive Clifford product with properties:
  - Closure

$$\forall \lambda \in \mathbb{K}, \forall e_i \in E, \quad \lambda e_1 \dots e_k \in \mathbb{G}^n \tag{C}$$

  - Reducibility

$$\forall e_k \in E, \quad e_k e_k = \sigma_k \tag{R}$$

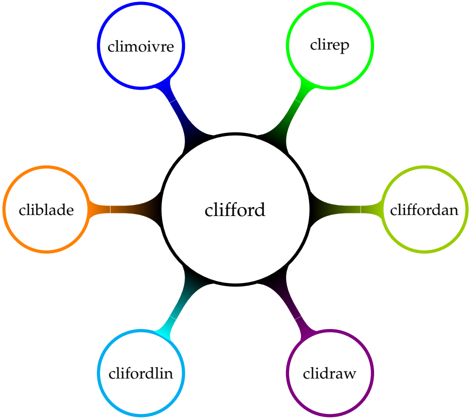
  $\sigma \in \{1, -1, 0\}$ – scalars of the field $\mathbb{K}$.
  - Anti-Commutativity

$$e_j e_i = -e_i e_j, \; e_i \prec e_j \tag{A-C}$$

  - Scalar Commutativity

$$\forall \lambda \in \mathbb{K}, \forall e_i \in E, \quad e_i \lambda = \lambda e_i \tag{S-C}$$

The `Clifford` package(s)



Developed since 2015

- minimalistic design
- unit tests
- demos + presentations
- mature code

Available in GitHub: `http://dprodanov.github.io/clifford/`
GPL license

Clifford algebra construction in `Clifford`

```
/*
Abstract Cliford algebra construction
*/
matchdeclare([aa, ee], lambda([u], not freeof(asymbol,u) and freeof ("+", u) and
    not scalarp(u) ), [bb,cc], true,
[kk, mm, nn], lambda([z],  integerp(z) and z>0) );

if get('clifford,'version)=false then (
    tellsimp(aa[kk].aa[kk], signature[kk] ),
    tellsimpafter(aa[kk].aa[mm], dotsimp2(aa[kk].aa[mm])),
    tellsimpafter(bb.ee.cc, dotsimpc(bb.ee.cc)),
    tellsimp(bb^nn, bb^^nn)
);
```

Clifford product is represented by the non-commutative operator "."

For scalars a, b

$$a \cdot b = a * b$$

Product simplification

Definition (Canonical real algebra)

Define the canonical ordering as the nested lexicographical order ϱ, such that $i < j \Longrightarrow e_i \prec e_j$ and extend it over $P(E)$ as:

$$e_1 \prec e_2 \prec \underbrace{e_1 e_2}_{e_{12}} \prec \ldots \prec \underbrace{e_1 \ldots e_n}_{e_N}$$

In addition assume that the first p elements square to 1, the next q elements square to -1 and the last r elements square to 0. Then the algebra $C\ell_{p,q,r} \equiv \{E, \varrho, \mathbb{R}\}$ is the canonical Clifford algebra.

Lemma (Permutation equivalence)

Let $M = e_{k_1} \ldots e_{k_i}$ be a Clifford multinomial, where the indices are not necessarily different. Then

$$M = s\, P_\rho \left\{ e_{k_1} \ldots e_{k_i} \right\}$$

where $s = \pm 1$ is the sign of permutation of M and $P_\rho \left\{ e_{k_1} \ldots e_{k_i} \right\} \mapsto e_{k_\alpha} \ldots e_{k_\omega}$ is the product permutation according to the canonical ordering.

Parity of permutation algorithm

```
permsign(arr):=block([k:0, len, ret:0 ],
    if not listp(arr) then return (false),
        len:length(arr),
        for i:1 thru len do (
            if not mapatom(arr[i]) then ret:nil,
            for j:i+1 thru len do
                if ordergreatp(arr[i], arr[j]) then k:k+1
            ),
    if ret#nil then
        if evenp(k)  then 1 else -1
    else 0
);
```

`ordergreatp` computes the predicate $\Pi\left(e_i \prec e_j\right)$

Product simplification algorithm in `Clifford`

```
dotsimpc(ab):=block([c:1, v, w:1, q, r, l, sop],
    sop:inop(ab),
    if mapatom(ab) or freeof(".", ab) or sop='nil or sop="^"  or sop="^^" then
         return(ab),
    if sop="+" then map(dotsimpc, ab)
    else if sop="*" then (
        [r,l]: oppart(ab, lambda([u], freeof(".", u))),
        r:subst(nil=1, r),
        l:subst(".","*",l),
        r*dotsimpc(l)
    ) else (
        v:inargs(copy(ab)),
        w:sublist(v, lambda([z], not freeof(asymbol,z) and mapatom(z))),
        w:permsign(w),
        if w#0 then (
            v:sort(v),
            for q in v do c:c.q,
            w*c
        ) else ab
    )
);
```

Multiplication tables

Definition (Full Matrix multiplication table)

Consider the extended basis **E**. Define the multiplication table matrix as the mapping

$$\begin{aligned}&\mu : \Xi(\mathbf{B}\times\mathbf{B}) \mapsto \mathbf{Mat}(2^n \times 2^n),\ n = p+q+r\\ &\mu(\mathbf{B}) = \mathbf{M}_{C\ell_{p,q,r}}\end{aligned}$$

with matrix consisting of the ordered product entries using the multi-index notation

$$\mathbf{M} := \{m_{\mu\nu}\, e_M e_N \,|M \prec N\},\ \ m_{\mu\nu} = \{-1, 0, 1\}$$

$$C\ell_{2,0,0} : \quad \mathbf{Mat} = \left(\begin{array}{c|ccc} 1 & e_1 & e_2 & e_1e_2 \\ \hline e_1 & 1 & e_1e_2 & e_2 \\ e_2 & -e_1e_2 & 1 & -e_1 \\ e_1e_2 & -e_2 & e_1 & -1 \end{array}\right)$$

imec

Examples: multiplication tables

Quaternions

```
load('clifford);
clifford(e,0,2);
mtable1([1, e[1],e[2], e[1] . e[2]]);
```

Quaternion multiplication table

$$\begin{pmatrix} 1 & e_1 & e_2 & e_1.e_2 \\ e_1 & -1 & e_1.e_2 & -e_2 \\ e_2 & -e_1.e_2 & -1 & e_1 \\ e_1.e_2 & e_2 & -e_1 & -1 \end{pmatrix}$$

(minimal manual formatting)

Scalar product

Definition (Scalar product table)

scalar product of the blades A and B

$$A * B := \langle A\, B \rangle_0$$

Define the scalar product table

$$\mathbf{G} := \{ g_{\mu\nu}\, e_M * e_N \, | M \prec N \}$$

Quaternion scalar product table, command: mtable2s();

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

Coefficient map

Definition (Element map)

Define the linear map acting element-wise $C_a : C\ell \mapsto \mathbb{R}$ by the action

$$C_a : \begin{cases} ax & \mapsto x, \quad x \in \mathbb{R}, a \in \mathbf{B} \\ b & \mapsto 0, \quad b \in \mathbf{B} \end{cases}$$

Define the coefficient map indexed by the multi-index S as

$$C_S : \mathbf{M} \mapsto \mathbf{A}_S$$

$$C_1 : \begin{pmatrix} 1 & e_1 & e_2 & e_1.e_2 \\ e_1 & -1 & e_1.e_2 & -e_2 \\ e_2 & -e_1.e_2 & -1 & e_1 \\ e_1.e_2 & e_2 & -e_1 & -1 \end{pmatrix} \mapsto \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix} = \mathbf{A}_1$$

imec

Coefficient map

Definition (Canonical matrix map)

For the multi-index S define the map

$$\pi : e_S \mapsto \mathbf{E}_s = \mathbf{GA}_s$$

where s is the ordinal of e_S in the multivector basis $\mathbf{B}$. Further, denote the set of all maps as $\pi = \{\pi_s\}$ and let $\pi_s \equiv \pi(e_s)$.

$$\mathbf{E_1} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Semigroup property

Theorem (Semigroup property)

Let e_s and e_t are basis elements. Then the map π acts on $C\ell_{p,q}$ according to the following diagram

$$\begin{tikzcd}
e_s \arrow[r, "\pi"] \arrow[d, "e_t"] & \mathbf{E}_s \arrow[d, "\mathbf{E}_t"] \\
e_s e_t \equiv e_{st} \arrow[r, "\pi"] & \mathbf{E}_{st} \equiv \mathbf{E}_s \mathbf{E}_t
\end{tikzcd}$$

The map π distributes over the Clifford product:

$$\pi(e_s e_t) = \pi(e_s)\pi(e_t)$$

Canonical Matrix Representation

Theorem (Canonical Matrix Representation)

Define the map $g : \mathbf{A} \mapsto \mathbf{GA}$. *Then*

$$\pi_s = C_s \circ g = g \circ C_s$$

so that the diagram

$$\begin{tikzcd}
\mathbf{M} \arrow[r, "C_s"] \arrow[d, "g"] \arrow[dr, "\pi_s"] & \mathbf{A}_s \arrow[d, "g"] \\
\mathbf{GM} \arrow[r, "C_s"] & \mathbf{E}_s = \mathbf{GA}_s
\end{tikzcd}$$

commutes.

π *is an isomorphism inducing a Clifford algebra representation in the real matrix algebra:*

$$C\ell_{p,q}(\mathbb{R}) \underset{\pi^{-1}}{\overset{\pi}{\longleftrightarrow}} C\ell_{p,q}\left[\mathbf{Mat}_{\mathbb{R}}(2^n \times 2^n)\right]$$

π *distributes over the Clifford product (homomorphism):*

$$\pi_{st} = \pi_s \pi_t$$

Proof sketch

Proof.

The π-map is a linear isomorphism. The set $\{\mathbf{E}_s\}$ forms a semigroup, which is a subset of the matrix algebra $\mathbf{Mat}_{\mathbb{R}}(2^n \times 2^n)$. Let

$$\pi(e_s) = \mathbf{E}_s, \quad \pi(e_t) = \mathbf{E}_t$$

It is claimed that

1. $\mathbf{E}_s \mathbf{E}_s = \sigma_s \mathbf{I}$.
2. $\mathbf{E}_s \mathbf{E}_t \neq \mathbf{0}$.
3. $\mathbf{E}_s \mathbf{E}_t = -\mathbf{E}_t \mathbf{E}_s$.

Therefore, $\{\mathbf{E}_s\}$ is an image of $C\ell_{p,q}$. $\square$

Details of proofs and supporting lemmas given in

D. Prodanov, A Symbolic Algorithm for Computation of Non-degenerate Clifford Algebra Matrix Representations, ArXiv:1904.00084, 2019.

Clifford code

π map implementation for blades

```
/*  computes blade representation */
climatrep1(vv):=block([n, AA, lst, EE, opsubst:false,
    gs:gensym(string(asymbol)) ],

    local(AA, EE),
    if emptyp(%elements) then lst:elements(all)
    elseif %elements[1] #1 then lst: cons(1, %elements)
    else lst: %elements,

    n:length(lst),
    /* multiplication table of the algebra */
    AA:genmatrix( lambda([i,j], dotsimpc(lst[i] . lst[j] ) ), n),
    AA:subst( asymbol[gs]=asymbol[1], subst(1=gs, AA)),
    [l, r]: oppart(vv, lambda([u], freeof(asymbol, u))),

    if l='nil then l:1, if r='nil then r:gs,

    EE:matrixmap(lambda([q], coeff (q, r)), AA),
    /*  twiddle to get the signs right*/
    l*genmatrix( lambda([i,j],  dotsimpc( lst[i] . lst[i] )*EE[i,j] ), n)
);
```

Clifford code

Extension by linearity

```
/*  computes expression representation */
elem2mat1(expr, [ulst] ):=block(
    if emptyp(ulst) then ulst:false
    else ulst:true,
    if mapatom(expr) then return(climatrep1(expr)),
    if ulst=true then
        maplist(climatrep1, expr)
    else
        map(climatrep1, expr)
);
```

Demonstrations

n=2

Geometric algebra $C\ell_{2,0} = \mathbb{R} \oplus \mathbb{R}$

$$a_1 + e_1\, a_2 + e_2\, a_3 + a_4\ (e_1 e_2) \mapsto \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ a_2 & a_1 & a_4 & a_3 \\ a_3 & -a_4 & a_1 & -a_2 \\ -a_4 & a_3 & -a_2 & a_1 \end{pmatrix}$$

n=2

Split-quaternions $C\ell_{1,1}$

$$a_1 + e_1\, a_2 + e_2\, a_3 + a_4\, (e_1 e_2) \mapsto \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ a_2 & a_1 & a_4 & a_3 \\ -a_3 & a_4 & a_1 & -a_2 \\ a_4 & -a_3 & -a_2 & a_1 \end{pmatrix}$$

n=2

Quaternions $C\ell_{0,2}$

$$a_1 + e_1\, a_2 + e_2\, a_3 + a_4\ (e_1 e_2) \mapsto \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ -a_2 & a_1 & -a_4 & a_3 \\ -a_3 & a_4 & a_1 & -a_2 \\ -a_4 & -a_3 & a_2 & a_1 \end{pmatrix}$$

n=3

$$a_1 + e_1\, a_2 + e_2\, a_3 + e_3\, a_4 + a_5\ (e_1e_2) + a_6\ (e_1e_3) + a_7\ (e_2e_3) + a_8\ (e_1e_2e_3)$$

Geometric algebra $C\ell_{3,0}$

$$\begin{pmatrix}
a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \\
a_2 & a_1 & a_5 & a_6 & a_3 & a_4 & a_8 & a_7 \\
a_3 & -a_5 & a_1 & a_7 & -a_2 & -a_8 & a_4 & -a_6 \\
a_4 & -a_6 & -a_7 & a_1 & a_8 & -a_2 & -a_3 & a_5 \\
-a_5 & a_3 & -a_2 & -a_8 & a_1 & a_7 & -a_6 & a_4 \\
-a_6 & a_4 & a_8 & -a_2 & -a_7 & a_1 & a_5 & -a_3 \\
-a_7 & -a_8 & a_4 & -a_3 & a_6 & -a_5 & a_1 & a_2 \\
-a_8 & -a_7 & a_6 & -a_5 & a_4 & -a_3 & a_2 & a_1
\end{pmatrix}$$

n=3

$$a_1 + e_1\, a_2 + e_2\, a_3 + e_3\, a_4 + a_5\ (e_1 e_2) + a_6\ (e_1 e_3) + a_7\ (e_2 e_3) + a_8\ (e_1 e_2 e_3)$$

$C\ell_{2,1}$

$$\begin{pmatrix}
a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \\
a_2 & a_1 & a_5 & a_6 & a_3 & a_4 & a_8 & a_7 \\
a_3 & -a_5 & a_1 & a_7 & -a_2 & -a_8 & a_4 & -a_6 \\
-a_4 & a_6 & a_7 & a_1 & -a_8 & -a_2 & -a_3 & a_5 \\
-a_5 & a_3 & -a_2 & -a_8 & a_1 & a_7 & -a_6 & a_4 \\
a_6 & -a_4 & -a_8 & -a_2 & a_7 & a_1 & a_5 & -a_3 \\
a_7 & a_8 & -a_4 & -a_3 & -a_6 & -a_5 & a_1 & a_2 \\
a_8 & a_7 & -a_6 & -a_5 & -a_4 & -a_3 & a_2 & a_1
\end{pmatrix}$$

n=3

$$a_1 + e_1\, a_2 + e_2\, a_3 + e_3\, a_4 + a_5\ (e_1 e_2) + a_6\ (e_1 e_3) + a_7\ (e_2 e_3) + a_8\ (e_1 e_2 e_3)$$

algebra $C\ell_{1,2}$

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \\ a_2 & a_1 & a_5 & a_6 & a_3 & a_4 & a_8 & a_7 \\ -a_3 & a_5 & a_1 & -a_7 & -a_2 & a_8 & a_4 & -a_6 \\ -a_4 & a_6 & a_7 & a_1 & -a_8 & -a_2 & -a_3 & a_5 \\ a_5 & -a_3 & -a_2 & a_8 & a_1 & -a_7 & -a_6 & a_4 \\ a_6 & -a_4 & -a_8 & -a_2 & a_7 & a_1 & a_5 & -a_3 \\ -a_7 & -a_8 & -a_4 & a_3 & -a_6 & a_5 & a_1 & a_2 \\ -a_8 & -a_7 & -a_6 & a_5 & -a_4 & a_3 & a_2 & a_1 \end{pmatrix}$$

n=3

$$a_1 + e_1\, a_2 + e_2\, a_3 + e_3\, a_4 + a_5\ (e_1 e_2) + a_6\ (e_1 e_3) + a_7\ (e_2 e_3) + a_8\ (e_1 e_2 e_3)$$

algebra $C\ell_{0,3}$

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \\ -a_2 & a_1 & -a_5 & -a_6 & a_3 & a_4 & -a_8 & a_7 \\ -a_3 & a_5 & a_1 & -a_7 & -a_2 & a_8 & a_4 & -a_6 \\ -a_4 & a_6 & a_7 & a_1 & -a_8 & -a_2 & -a_3 & a_5 \\ -a_5 & -a_3 & a_2 & -a_8 & a_1 & -a_7 & a_6 & a_4 \\ -a_6 & -a_4 & a_8 & a_2 & a_7 & a_1 & -a_5 & -a_3 \\ -a_7 & -a_8 & -a_4 & a_3 & -a_6 & a_5 & a_1 & a_2 \\ a_8 & -a_7 & a_6 & -a_5 & -a_4 & a_3 & -a_2 & a_1 \end{pmatrix}$$

n=4

For a general element of the form

$$a_1 + e_1\,a_2 + e_2\,a_3 + e_3\,a_4 + e_4\,a_5 + a_6\,(e_1e_2) + a_7\,(e_1e_3) + a_8\,(e_1e_4) + a_9\,(e_2e_3) + a_{10}\,(e_2e_4) + a_{11}\,(e_3e_4) + a_{12}\,(e_1e_2e_3) + a_{13}\,(e_1e_2e_4) + a_{14}\,(e_1e_3e_4) + a_{15}\,(e_2e_3e_4) + a_{16}\,(e_1e_2e_3e_4)$$

Space-time algebra $C\ell_{1,3}$

$$\begin{pmatrix}
a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\
a_2 & a_1 & a_6 & a_7 & a_8 & a_3 & a_4 & a_5 & a_{12} & a_{13} & a_{14} & a_9 & a_{10} & a_{11} & a_{16} & a_{15} \\
-a_3 & a_6 & a_1 & -a_9 & -a_{10} & -a_2 & a_{12} & a_{13} & a_4 & a_5 & -a_{15} & -a_7 & -a_8 & a_{16} & a_{11} & -a_{14} \\
-a_4 & a_7 & a_9 & a_1 & -a_{11} & -a_{12} & -a_2 & a_{14} & -a_3 & a_{15} & a_5 & a_6 & -a_{16} & -a_8 & -a_{10} & a_{13} \\
-a_5 & a_8 & a_{10} & a_{11} & a_1 & -a_{13} & -a_{14} & -a_2 & -a_{15} & -a_3 & -a_4 & a_{16} & a_6 & a_7 & a_9 & -a_{12} \\
a_6 & -a_3 & -a_2 & a_{12} & a_{13} & a_1 & -a_9 & -a_{10} & -a_7 & -a_8 & a_{16} & a_4 & a_5 & -a_{15} & -a_{14} & a_{11} \\
a_7 & -a_4 & -a_{12} & -a_2 & a_{14} & a_9 & a_1 & -a_{11} & a_6 & -a_{16} & -a_8 & -a_3 & a_{15} & a_5 & a_{13} & -a_{10} \\
a_8 & -a_5 & -a_{13} & -a_{14} & -a_2 & a_{10} & a_{11} & a_1 & a_{16} & a_6 & a_7 & -a_{15} & -a_3 & -a_4 & -a_{12} & a_9 \\
-a_9 & -a_{12} & -a_4 & a_3 & -a_{15} & -a_7 & a_6 & -a_{16} & a_1 & -a_{11} & a_{10} & a_2 & -a_{14} & a_{13} & a_5 & a_8 \\
-a_{10} & -a_{13} & -a_5 & a_{15} & a_3 & -a_8 & a_{16} & a_6 & a_{11} & a_1 & -a_9 & a_{14} & a_2 & -a_{12} & -a_4 & -a_7 \\
-a_{11} & -a_{14} & -a_{15} & -a_5 & a_4 & -a_{16} & -a_8 & a_7 & -a_{10} & a_9 & a_1 & -a_{13} & a_{12} & a_2 & a_3 & a_6 \\
-a_{12} & -a_9 & -a_7 & a_6 & -a_{16} & -a_4 & a_3 & -a_{15} & a_2 & -a_{14} & a_{13} & a_1 & -a_{11} & a_{10} & a_8 & a_5 \\
-a_{13} & -a_{10} & -a_8 & a_{16} & a_6 & -a_5 & a_{15} & a_3 & a_{14} & a_2 & -a_{12} & a_{11} & a_1 & -a_9 & -a_7 & -a_4 \\
-a_{14} & -a_{11} & -a_{16} & -a_8 & a_7 & -a_{15} & -a_5 & a_4 & -a_{13} & a_{12} & a_2 & -a_{10} & a_9 & a_1 & a_6 & a_3 \\
a_{15} & -a_{16} & -a_{11} & a_{10} & -a_9 & a_{14} & -a_{13} & a_{12} & -a_5 & a_4 & -a_3 & a_8 & -a_7 & a_6 & a_1 & -a_2 \\
-a_{16} & a_{15} & a_{14} & -a_{13} & a_{12} & -a_{11} & a_{10} & -a_9 & a_8 & -a_7 & a_6 & -a_5 & a_4 & -a_3 & -a_2 & a_1
\end{pmatrix}$$

Conclusion

- The algorithm is limited by the system resources (not a problem for clusters of cloud computing).
- Construction can be implemented in any general-purpose linear algebra software.
- While this is not the most economical way of representation it offers a transparent mechanism for translation between a Clifford algebra and its faithful real matrix representation.

Conclusion

- The algorithm is limited by the system resources (not a problem for clusters of cloud computing).
- Construction can be implemented in any general-purpose linear algebra software.
- While this is not the most economical way of representation it offers a transparent mechanism for translation between a Clifford algebra and its faithful real matrix representation.

Conclusion

- The algorithm is limited by the system resources (not a problem for clusters of cloud computing).
- Construction can be implemented in any general-purpose linear algebra software.
- While this is not the most economical way of representation it offers a transparent mechanism for translation between a Clifford algebra and its faithful real matrix representation.

References

A. Macdonald.
An elementary construction of the geometric algebra.
Advances in Applied Clifford Algebras, 12(1):1 – 6, 2002.

D. Prodanov and V. T. Toth.
Sparse representations of clifford and tensor algebras in Maxima.
Advances in Applied Clifford Algebras, 27(1), 661–683, 2017.
http://arxiv.org/pdf/1604.06967.pdf

D. Prodanov
Clifford Algebra Implementations in Maxima
Journal of Geometry and Symmetry in Physics, 43, 73–105, 2016.

D. Prodanov
A Symbolic Algorithm for Computation of Non-degenerate Clifford Algebra Matrix Representations
ArXiv:1904.00084, 2019.

THANK YOU FOR YOUR ATTENTION!

Imec campus, Leuven, Brussels area