Clifford algebra implementations in Maxima

Dimiter Prodanov

IMEC

Alterman Conference on Geometric Algebra, Brasov, Romania, 4 – 6 Aug 2016

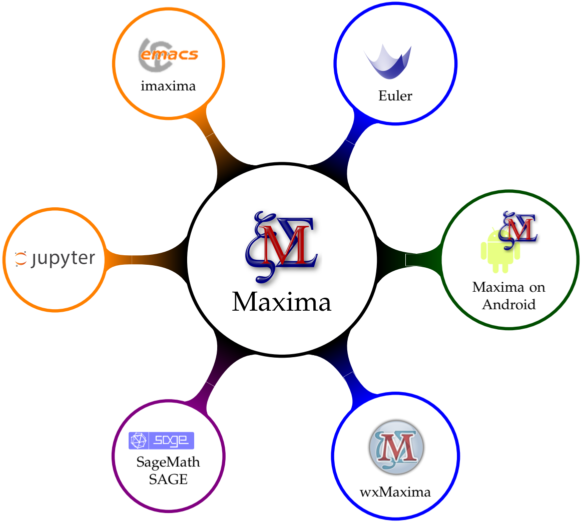
ASPIRE
INVENT
ACHIEVE

Overview

Maxima

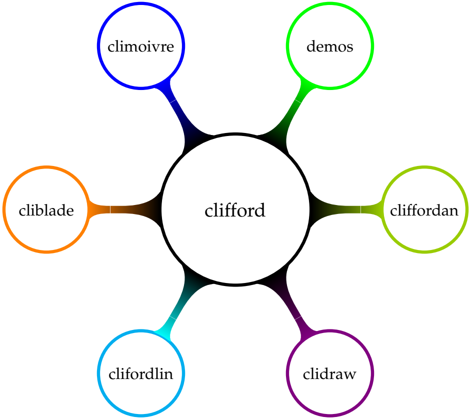
WHY ANOTHER PACKAGE?

The Maxima ecosystem



- Maxima is widely used
- open source allows for shorted development cycles
- errors are corrected quickly

THE `Clifford` PACKAGE



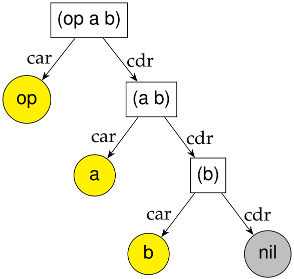
- minimalistic design
- unit tests
- demos
- experimental
- GPL license

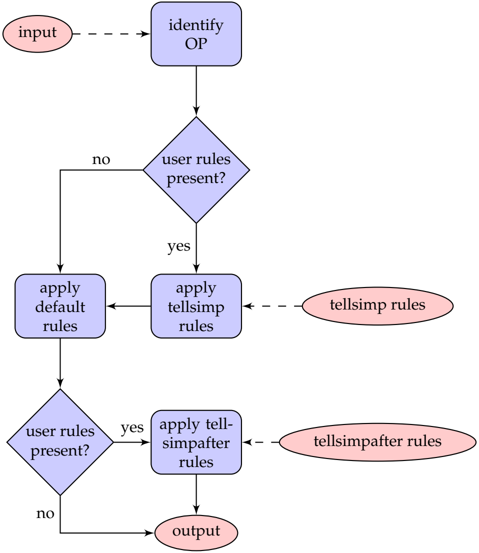
Available from GitHub

`http://dprodanov.github.io/clifford/`

Expression simplification in Maxima

Parse tree Lisp representation





- car - first element
- cdr - rest (a new list)

Clifford algebras

ELEMENTARY CONSTRUCTION OF CLIFFORD ALGEBRAS

- Define 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** $E := \{e_1 \ldots e_n\} \subset \mathbb{G}^n$.
- Define 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
    $$\forall \lambda \in \mathbb{K}, \forall e_i \in E, \quad \lambda e_1 \ldots e_k \in \mathbb{G}^n \tag{C}$$
  - 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
    $$\forall \lambda \in \mathbb{K}, \forall e_i \in E, \quad e_i \lambda = \lambda e_i \tag{S-C}$$

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

$$a \cdot b \mapsto a * b$$

Product simplification

Definition

A multivector of the Clifford algebra is a linear combination of elements over the 2^n-dimensional vector space spanned by the power-set

$$P(E) := \{1, e_1, \ldots, e_n, e_1e_2, e_1e_3, \ldots, e_1e_2...e_n\}.$$

Lemma (Permutation equivalence)

Let $B = e_{k_1} \ldots e_{k_i}$ be an arbitrary Clifford product, where the i basis vectors are not necessarily different. Then

$$B = s\, P_{\prec} \left\{ e_{k_1} \ldots e_{k_i} \right\}$$

where $s = \pm 1$ is the sign of permutation of B and $P_{\prec} \left\{ e_{k_1} \ldots e_{k_i} \right\}$ is the product permutation according to the ordering $\prec$.

Proof.

The proof follows directly from the anti-commutativity of multiplication for any two basis vectors, observing that the sign of a permutation of S can be defined from its decomposition into the product of transpositions as $sgn(B) = (-1)^m$, m – number of transpositions in the decomposition. $\square$

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
    )
);
```

Blade decomposition

Definition (Simplified form)

A product of k basis elements reordered canonically.

Definition (Blade)

A blade of grade k is a product of k basis elements in simplified form.

Grade projection operator : $\langle\ \rangle_k$

Grade decomposition : $M = \sum_{k=0}^{n} \langle M \rangle_k$

MAIN FUNCTIONS IN `Clifford`

- PRODUCT SIMPLIFICATION

`cliffsimpall` (expr)	full simplification of expressions
`dotsimp1` (ab)	canonic reordering of dot products
`dotsimpc` (ab)	simplification of dot products
`dotinvsimp` (ab)	simplification of inverses
`powsimp` (ab)	simplification of exponents

- INVOLUTIONS

`dotreverse` (ab)	Clifford reverse of product
`cinvolve` (expr)	Clifford involution of expression
`dotconjugate` (expr)	Clifford conjugate of expression

- GRADE FUNCTIONS

`grade` (expr)	grade decomposition of expression
`scalarpart` (expr)	$< expr >_0$
`vectorpart` (expr)	$< expr >_1$
`grpart` (expr,k)	$< expr >_k$
`mvectorpart` (expr)	$< expr >_{2+}$
`bdecompose` (expr)	blade decomposition of expression

OUTER PRODUCT IN `Clifford`

$$A_k \wedge B_l = \begin{cases} \langle A_k B_l \rangle_{k+l} & k+l \leq n \\ 0 & k+l > n \end{cases}$$

```
infix("&", 135, 134);
declare("&", additive);
"&"(a, b):= block ([l,r, ret:0],
    if ndim=0 then return ( buildq([a,b],"&"(a, b))),
    if not mapatom(a) then a:expand(a),
    if not mapatom(b) then b:expand(b),
    if inop(a)="+" then return (map( lambda ([u],  u & b), a)),
    if inop(b)="+" then return (map( lambda ([u],  a & u), b)),
    if wedgesimp then (
        if not freeof(".", a) then
            a:cliffsimp1(a),
        if not freeof(".", b) then
            b:cliffsimp1(b)
    ),
    l:maxgrade(a),
    r:maxgrade(b),
    ret:cliffsimp1(a.b),
    grpart(ret, l+r)
);
```

Reference implementation based on the definition

Demonstrations, part I

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)

Outer product in $\mathbb{G}^3$

```
clifford(e, 3);
e[1] &e[2] &e[3];
(1+e[1])&(1+e[1]);
(1+e[1])&(1-e[1]);
mtable2o();
```

Outer product multiplication table

$$\begin{pmatrix}
1 & e_1 & e_2 & e_3 & e_1\cdot e_2 & e_1\cdot e_3 & e_2\cdot e_3 & e_1\cdot e_2\cdot e_3\\
e_1 & 0 & e_1\cdot e_2 & e_1\cdot e_3 & 0 & 0 & e_1\cdot e_2\cdot e_3 & 0\\
e_2 & -e_1\cdot e_2 & 0 & e_2\cdot e_3 & 0 & -e_1\cdot e_2\cdot e_3 & 0 & 0\\
e_3 & -e_1\cdot e_3 & -e_2\cdot e_3 & 0 & e_1\cdot e_2\cdot e_3 & 0 & 0 & 0\\
e_1\cdot e_2 & 0 & 0 & e_1\cdot e_2\cdot e_3 & 0 & 0 & 0 & 0\\
e_1\cdot e_3 & 0 & -e_1\cdot e_2\cdot e_3 & 0 & 0 & 0 & 0 & 0\\
e_2\cdot e_3 & e_1\cdot e_2\cdot e_3 & 0 & 0 & 0 & 0 & 0 & 0\\
e_1\cdot e_2\cdot e_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Inner product(s) in $\mathbb{G}^3$

Inner product types: `lc` – left contraction, `rc` – right contraction , `sym` – for symmetric

```
inprotype:lc;
mtable2i();
```

Left contraction multiplication table

$$\begin{pmatrix} 1 & e_1 & e_2 & e_3 & e_1 \cdot e_2 & e_1 \cdot e_3 & e_2 \cdot e_3 & e_1 \cdot e_2 \cdot e_3 \\ e_1 & 1 & 0 & 0 & e_2 & e_3 & 0 & e_2 \cdot e_3 \\ e_2 & 0 & 1 & 0 & -e_1 & 0 & e_3 & -e_1 \cdot e_3 \\ e_3 & 0 & 0 & 1 & 0 & -e_1 & -e_2 & e_1 \cdot e_2 \\ e_1 \cdot e_2 & 0 & 0 & 0 & -1 & 0 & 0 & -e_3 \\ e_1 \cdot e_3 & 0 & 0 & 0 & 0 & -1 & 0 & e_2 \\ e_2 \cdot e_3 & 0 & 0 & 0 & 0 & 0 & -1 & -e_1 \\ e_1 \cdot e_2 \cdot e_3 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Geometric calculus

Generalized derivatives in Clifford algebras

Definition

Consider the vector $r \in Span\left\{e_1, \ldots, e_n\right\}$ and reciprocal frames $e^k = e_k^{-1}$.
Vector derivative

$$\nabla_r F(x) := \sum_{i=1}^{n} e^i \lim_{\epsilon \to 0} \frac{F(x + \epsilon\, e_i) - F(x)}{\epsilon} = e^i\, \partial_i\, F$$

for $\epsilon > 0$

$$\nabla_r F = \nabla_r \cdot F + \nabla_r \wedge F$$

A possible generalization

$$\nabla_{\pm}^{\beta} F(x) = \sum_{i=1}^{n} \pm\, e^i \lim_{\epsilon \to 0} \frac{F(x \pm \epsilon\, e_i) - F(x)}{\epsilon^{\beta}} = e^i \partial_{\pm i}^{\beta} F(x)$$

for the exponent $0 < \beta \leq 1$.

MAIN FUNCTIONS IN `Clifordan`

`ctotdiff`(f, x)	total derivative w.r.t. multivector x
`ctotintdiff`(f, x)	inner total derivative w.r.t. x
`ctotextdiff`(f, x)	outer total derivative w.r.t. x
`vectdiff`(f, ee, k)	vector derivative of order k w.r.t. basis vector list *ee*
`mvectdiff`(f, x, k)	multivector derivative of order k w.r.t. multivector x
`parmvectdiff`(f, x, k)	partial multivector derivative of order k w.r.t. multivector x
`convderiv`(f, t, xx, [vs])	convective derivative w.r.t. multivector x
`coordsubst`(x, eqs)	substitutes coordinates in multivector x w.r.t. new variables in the list *eqs*
`clivolel`(x, eqs)	computes volume element of $Span\,\{x\}$ w.r.t. new variables in the list *eqs*

```
/*
Clifford-valued
total differentiation; */
ctotdiff(f,x):=block( [ret:0, lv],
    if mapatom(x) then
    ret: diff( f, x )
    else (
        lv: sublist( listofvars(x), lambda ([z], freeof(asymbol,z))),
        for u in lv do
            ret: ret + cinv(diff(x, u)). subst(".", "*", diff( f, u ))
    ),
    ret
);
```

Demonstrations, part II

POTENTIAL PROBLEMS IN $\mathbb{G}^3$

Potential equation for the Green's function

$$\nabla_r G = \delta(r)$$
$$G(x,y,z) = \frac{e_1\,x + e_2\,y + e_3\,z}{\sqrt{(x^2+y^2+z^2)^3}}$$

CARTESIAN COORDINATES

```
/* Initialization */
load('clifford);
load ('cliffordan);

/* G(3) construction */
clifford(e,3);
r:cvect([x,y,z]);
G:r/sqrt(-cnorm(r))^3;
mvectdiff(G,r);

mvectdiff(-1/sqrt(-cnorm(r)),r);
mvectdiff(-1/sqrt(-cnorm(r)),r,2);
```

Potential problems in $\mathbb{G}^3$

Cylindrical coordinates

```
cyl_eq:[x=rho*cos(phi), y=rho*sin(phi)];
   declare( [rho, phi], scalar);
   GG_c:coordsubst(G, cyl_eq),factor;
   rc:coordsubst(r, cyl_eq);
   mvectdiff(GG_c,rc);
   V:coordsubst(-1/sqrt(-cnorm(r)),cyl_eq);
   mvectdiff(V,rc);
```

$$GG_c = \frac{e_1 \rho \cos\phi + e_2 \rho \sin\phi + e_3 z}{\left(\rho^2 + z^2\right)^{\frac{3}{2}}}$$

$$V = \frac{e_1 \rho \cos\phi + e_2 \rho \sin\phi + e_3 z}{\left(\rho^2 + z^2\right)^{\frac{3}{2}}}$$

imec

EULER-LAGRANGE PROBLEMS IN $\mathbb{G}^3$

Euler-Lagrange field equations according to Lasenby (1993)

$$\begin{aligned}\pi &= \nabla_r\, A \\ \nabla_A \mathcal{L} &= \nabla_r\, (\nabla_\pi \mathcal{L})\end{aligned}$$

Paravector potential and derivative object

$$\begin{aligned}A =& A_t + e_1\, A_x + e_2\, A_y + e_3\, A_z \\ F =& \nabla_{t-r} A = \pi^\bullet\end{aligned}$$

$$\begin{aligned}F =& \underbrace{A_{tt} - A_{xx} - A_{y_y} - A_{zz}}_{S} \\ &+ \underbrace{e_1\ (A_{xt} - A_{tx}) + e_2\ (A_{y_t} - A_{ty}) + e_3\ (A_{zt} - A_{tz})}_{V} \\ &+ \underbrace{e_1 \cdot e_2\ (A_{xy} - A_{y_x}) + e_1 \cdot e_3\ (A_{xz} - A_{zx}) + e_2 \cdot e_3\ (A_{y_z} - A_{zy})}_{Q}\end{aligned}$$

Euler-Lagrange problems in $\mathbb{G}^3$

Euler-Lagrange field equations according to Lasenby (1993)

$$\pi = \nabla_r A$$
$$\nabla_A \mathcal{L} = \nabla_r (\nabla_\pi \mathcal{L})$$

quadratic Lagrangian density

$$\mathcal{L}_a = \frac{1}{2}\left\langle F^2 \right\rangle_0 = \frac{1}{2}\left(\langle F\rangle_0^2 + \langle F\rangle_1^2 + \langle F\rangle_2^2\right)$$

$$\begin{aligned}
\mathcal{L}_a = &((A_{tt})^2 + (A_{tx})^2 + (A_{ty})^2 + (A_{tz})^2 - 2\,(A_{tx})\,(A_{xt}) + (A_{xt})^2 - 2\,(A_{tt})\,(A_{xx}) \\
&+ (A_{xx})^2 - (A_{xy})^2 - (A_{xz})^2 - 2\,(A_{ty})\,(A_{y_t}) + (A_{y_t})^2 \\
&+ 2\,(A_{xy})\,(A_{y_x}) - (A_{y_x})^2 - 2\,(A_{tt})\left(A_{y_y}\right) \\
&+ 2\,(A_{xx})\left(A_{y_y}\right) + \left(A_{y_y}\right)^2 - (A_{y_z})^2 - 2\,(A_{tz})\,(A_{zt}) + (A_{zt})^2 \\
&+ 2\,(A_{xz})\,(A_{zx}) - (A_{zx})^2 + 2\,(A_{y_z})\,(A_{zy}) - (A_{zy})^2 - 2\,(A_{tt})\,(A_{zz}) \\
&+ 2\,(A_{xx})\,(A_{zz}) + 2\left(A_{y_y}\right)(A_{zz}) + (A_{zz})^2)/\,2
\end{aligned}$$

EULER-LAGRANGE PROBLEMS IN $\mathbb{G}^3$

```
AA: celem(A,[t,x,y,z]);
dependsv(A,[t,x,y,z]);
r:cvect([x,y,z]);
F:mvectdiff(AA,t-r);
F:factorby(F, %elements);
L:lambda([x],1/2*scalarpart(cliffsimpall(x.x)))(F);
S:scalarpart(F);
V:vectorpart(F);
Q:grpart(F,2);
L-1/2*(S.S+V.V+Q.Q),cliffsimpall;
dA:mvectdiff(AA,r);
EuLagEq2(L, t+r,[AA,dA]);
bdecompose(%);
```

D'Alembert's equation

$$\nabla_r \nabla_r \bullet A = 0$$

$$\left(-A_{ttt} + A_{txx} + A_{tyy} + A_{tzz}\right) + e_1 \left(-A_{xtt} + A_{xxx} + A_{xyy} + A_{xzz}\right) + e_2 \left(-A_{y_{tt}} + A_{y_{xx}} + A_{y_{yy}} + A_{y_{zz}}\right) + e_3 \left(-A_{ztt} + A_{zxx} + A_{zyy} + A_{zzz}\right)$$

References

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

A. Macdonald.
A survey of geometric algebra and geometric calculus.
Advances in Applied Clifford Algebras, pages 1–39, 2016.

D. Prodanov and V. T. Toth.
Sparse representations of clifford and tensor algebras in Maxima.
Advances in Applied Clifford Algebras, pages 1–23, 2016.
`http://arxiv.org/pdf/1604.06967.pdf`

THANK YOU FOR THE ATTENTION!

Imec campus, Leuven, Brussels area

The work has been supported in part by a grant from Research Fund - Flanders (FWO) contract numbers G.0C75.13N, VS.097.16N.

imec