Abstract:

Computations in Grassmann algebra and Clifford algebra Cl(B) of an arbitrary bilinear form B can be performed with a Maple package CLIFFORD and its extensions such as BIGEBRA, Cliplus, Octonion, etc.. 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 Cl(B). It uses a standard (undotted) Grassmann basis in Cl(Q) but when the antisymmetric part of B is non zero, it can also compute in Lounesto's dotted Grassmann basis. The Clifford product can be computed either by using Chevalley's recursive formula or Rota-Stein's combinatorial formula. The aim of this tutorial is to provide a brief introduction into mathematical foundations of CLIFFORD as well as into its module structure, interconnection with the other packages, built-in mathematical information about Clifford algebras Cl(V,Q) of a quadratic, non-degenerate form Q and their spinor representations for 1 <= dim V <= 9, and built-in 278 help worksheets taking up 1,130 pages of documentation. It is expected that after this tutorial any listener will be able to start computing with CLIFFORD.

Keywords: Quantum Clifford algebra, contraction, dotted wedge product,
grade involution, Grassmann algebra, help pages, Clifford product, Chevalley recursive formula, Rota-Stein combinatorial formula, Hopf algebra, multivector, octonion, quaternion, reversion, spinor representation, wedge product, CLIFFORDENV, allsigs, clidata, cmulNUM, cmulRS, prolevel, SINGULARPLURALlink.

Note:

- **CLIFFORD** is a Maple package developed and maintained with Bertfried Fauser, Universitat Konstanz, Fachbereich Physik, Fach M678, 78457 Konstanz, Germany, Bertfried.Fauser@uni-konstanz.de, http://kaluza.physik.uni-konstanz.de/~fauser/

- **BIGEBRA** is a Maple package for computing with tensors and Hopf algebras. It has been developed jointly with Bertfried Fauser.

- In order to run this Maple worksheet under Maple, you will need the current CLIFFORD library posted at http://math.tntech.edu/rafal/cliff11/ that works with Maple 11. This library should also work with Maple 12. If you use an earlier version of Maple such as 8, 9, 9.5, or 10, please download an earlier version of CLIFFORD, as appropriate, from http://www.math.tntech.edu/rafal/. This worksheet will not work under older versions of Maple than version 8.

References:


1. Introduction: Basic types, enviromental variables, etc.

```maple
> restart:
```
```maple
`Number of procedures in CLIFFORD` = nops(%);
```
```maple
with(Bigebra):
```
```maple
`Number of procedures in Bigebra` = nops(%);
```
[ &m, Bs-phone, CLIFFORD_ENV, Kfield, LC, LCQ, RC, RCQ, RH-number, adf-matrix,
  all_sigs, beta-minus, beta-plus, buildm, by-grade, c-conjug, chasis, cdf-matrix, cexp, cexpQ,
  cinv, c-lib-linear, c-locate, c-locate, c-linear, c-limin-poly, c-locate, c-remove, c-solve, c-sort,
  cliterms, cmul, cmulNUM, cmulQ, cmulRS, cmul-gen, cocycle, commuting-elements,
  conjugation, ddf-matrix, diagonalize, displayid, extract, fact-ori-depotent, find1-str, findbasis,
  grade-inv, init, is-Vahlen-matrix, is-product, make-aliases, make-c-lib-as-mon, mat-Krepr, max-grade,
  max-index, mdf-matrix, minimal-ideal, ord, permsign, pseudodet, q-conjug, q-display, q-inv, q-mul,
  q-norm, rd-c-plib-as-mon, rd_c-limon, rd_c-lipolynom, reorder, reversion, rmulm, rot3d, scalar-part
  sexp, specify-constants, spinorK-basis, spinor Krepr, square-mod, subs_c-lipolynom,
  use-product, vector-part, version, wedge, wexp ]

Number of procedures in CLIFFORD = 1
Increase verbosity by infolevel[`function`]=val -- use online help > ?Bigebra[help]

Number of procedures in Bigebra = 33

> version();

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

CLIFFORD - A Maple 13 Package for Clifford Algebras with "Bigebra"

(Version 13.3 with environmental variables given by CLIFFORD_ENV())

Last revised: June 17, 2012 (Source file: clifford_M13_3.mws)

Copyright 1995-2012 by Rafal Ablamowicz (*) and Bertfried Fauser ($) 

(*) Department of Mathematics, Box 5054
Tennessee Technological University, Cookeville, TN 38505
tel: USA (931) 372-3569, fax: USA (931) 372-6353
rablamowicz@tntech.edu

http://math.tntech.edu/rafal/

($) Universit"at Konstanz, Fachbereich Physik, Fach M678
78457 Konstanz, Germany
Bertfried.Fauser@uni-konstanz.de
If you are a Clifford algebra pro, assign 'true' to '_prolevel' and see how much faster your computations will be! But watch your syntax!

Use 'useproduct' to change value of _default_Clifford_product in Cl(B) from cmulRS when B is symbolic to cmulNUM when B is numeric. Type ?cmul for help.

Type CLIFFORD_ENV() to see current values of environmental variables.

cmulNUM, cmulRS, cmulWalsh3, and reorder are thread-safe in this version

++++++++++++This is CLIFFORD version 13.3++++++++++++

To get help in CLIFFORD, type

> #?Clifford

The following are the default values of a some most important environmental variables:

> 'dim_V'=dim_V;
'prolevel'=prolevel;
'_default_Clifford_product'=_default_Clifford_product;
'_warnings_flag'=warnings_flag;
'_scalartypes'=_scalartypes;

\[
\text{dim}_V = 9
\]

\[
\text{prolevel} = \text{false}
\]

\[
\text{_default_Clifford_product} = \text{cmulRS}
\]

\[
\text{_warnings_flag} = \text{false}
\]

\[
\text{_scalartypes} = \{\text{^, RootOf, complex, indexed, numeric, constant, function, mathfunc, rational}\}
\]

> cmul(e1,e2);

Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

\[
e1we2 + B_{1,2} I\text{d}
\]

> cmul(e1,e2we3);
All global environmental variables are defined at the loading time, i.e., when the command with(Clifford) is executed. To see all environment variables and their default values, check CLIFFORD_ENV.

```plaintext
> CLIFFORD_ENV();

`>>> Global variables describing a default system defined in Clifford:-setup are now available and have these values: <<<`
`************* Start *************`
_processor_type = "Intel (R) Core (TM) 2 Duo CPU 2.19 GHz, 2.9 GB RAM"
_operating_system = "Win XP Professional (SP3)"
`************* End *************`

`>>> Global variables defined in Clifford:-setup are now available and have these values: <<<`
`************* Start *************`
dim_V = 9
_default_Clifford_product = Clifford:-cmulRS
_prolevel = false
Shortcut_in_minimalideal = true
Shortcut_in_Kfield = true
Shortcut_in_spinorKbasis = true
Shortcut_in_spinorKrepr = true
_warnings_flag = false
_scalartypes = {`^`, RootOf, complex, indexed, numeric, constant, function, mathfunc, rational}
_quatbasis = [[[Id, e3we2, e1we3, e2we1], {`Maple has assigned qi:=-e2we3, qj:=e1we3, qk:=-e1we2`}]}
`************* End *************`

`>>> Global variables defined in Cliplus:-setup are now available and have these values: <<<`
`************* Start *************`
macro(Clifford:-cmul = climul)
macro(Clifford:-cmulQ = climul)
macro(`&c` = climul)
macro(`&cQ` = climul)
macro(Clifford:-reversion = clirev)
macro(Clifford:-LC = LCbig)
macro(Clifford:-RC = RCbig)
`************* End *************`

`************* Start *************`
`>>> There are no new global variables or macros in GTP yet. <<<`
`************* End *************`

-B_{1.3} e2 + B_{1.2} e3 + e1we2we3
```
Global variables defined in Octonion:-setup are now available and have these values:

**Start**

```plaintext
_octbasis = [Id, e1, e2, e3, e4, e5, e6, e7]
_pureoctbasis = [e1, e2, e3, e4, e5, e6, e7]
_default_Fano_triples = [[1, 3, 7], [1, 2, 4], [1, 5, 6], [2, 3, 5], [2, 6, 7], [3, 4, 6], [4, 5, 7]]
_default_squares = [-Id, -Id, -Id, -Id, -Id, -Id, -Id]
_default_Clifford_product = Clifford:-cmulNUM
**End**
```

To see the meaning of all variables, ask for help

> #?CLIFFORD_ENV

In CLIFFORD, there are the following Maple types defined:

**Types in 'CLIFFORD':**

- `type/antisymmatrix` - an antisymmetric matrix
- `type/clibasmon` - a basis monomial in Cl(B)
- `type/climatrix` - matrix of with entries in Cl(B)
- `type/climonom` - a monomial in Cl(B)
- `type/clipolynom` - a polynomial in Cl(B)
- `type/cliprod` - basis monomial in a Clifford basis, used in the 'Cliplus' package described below
- `type/cliscalar` - a scalar in Cl(B)
- `type/dfmatrix` - matrix over a double field
- `type/diagmatrix` - a diagonal matrix
- `type/evenelement` - even element in Cl(B)
- `type/fieldelement` - a basis element in Cl(Q) or fCl(Q) which is in some field K
- `type/gencomplex` - a generalized complex number
- `type/gengquatbasis` - a generalized quaternion basis
- `type/genuquaternion` - a generalized quaternion
- `type/idempotent` - an idempotent element in Cl(B)
- `type/nilpotent` - a nilpotent element in Cl(B)
- `type/oddelement` - an odd element in Cl(B)
- `type/primitiveidemp` - a primitive idempotent in Cl(B)
- `type/purequatbasis` - a basis for pure quaternions
- `type/quaternion` - quaternion type
- `type/symmatrix` - a symmetric matrix
- `type/tensorprod` - place holders of tensor products
Additional functionalities are provided by supplementary packages such as Cliplus, Bigebra, Octonion, code_support, SchurFkt, SINGULARPLURALlink, SP, TNB, RJgrobner, asvd. Cliplus is loaded automatically by CLIFFORD when it is needed. It allows user to convert Grassmann basis (default) to Clifford basis (option) with a procedure cliexpand, and use dotted wedge basis versus the un-dotted (standard) wedge (Grassmann) basis. Procedure clieval converts back unevaluated Clifford product basis back to Grassmann basis:

```plaintext
with(Cliplus);
`Number of procedures in Cliplus` = nops(%);  

[LCbig, RCbig, clibasis, clieval, cliexpand, climul, clirev, dottedcbasis, dwedge, makeclialiases]

Number of procedures in Cliplus = 10

> wbas:=cbasis(3);  #default Grassmann basis in Cl(V), dim V = 3
    wbas := [Id, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3]

> map(cliexpand,wbas);  #unevaluated Clifford products in Cl(B)

[&C(Id), &C(e1), &C(e2), &C(e3), -B_{1,2} &C(Id) + (e1 &C e2),
 -B_{1,3} &C(Id) + (e1 &C e3), -B_{2,3} &C(Id) + (e2 &C e3),
 B_{1,3} &C(e2) - B_{2,3} &C(e1) - B_{1,2} &C(e3) + &C(e1, e2, e3)]

> map(clieval,%);  #convert back to Grassmann monomial basis

[Id, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3]

> map(cliexpand,wbas,K);  #unevaluated Clifford products in Cl(K)

[&C_k(Id), &C_k(e1), &C_k(e2), &C_k(e3), -K_{1,2} &C_k(Id) + &C_k(e1, e2),
 -K_{1,3} &C_k(Id) + &C_k(e1, e3), -K_{2,3} &C_k(Id) + &C_k(e2, e3),
 K_{1,3} &C_k(e2) - K_{2,3} &C_k(e1) - K_{1,2} &C_k(e3) + &C_k(e1, e2, e3)]

> p:=2-3*e1we2+e4;  #arbitrary Grassmann polynomial p in undotted basis

p := 2 - 3 e1we2 + e4

> convert(p,wedge_to_dwedge,F);  #p in dotted Grassmann basis
> #?Clilplus, setup

**Bigebra** package (with Bertfried Fauser) allows for computations with Hopf Grassmann and Clifford algebras.

> #?Bigebra

**Octonion** package allows for computations with octonions which are treated as paravectors in Cl(0,7). Octonionic multiplication table can be defined by selecting an appropriate Fano triple:

> B:=diag(-1$7):with(Octonion);`Number of procedures in Octonion`
   = nops(%);

\[ \Phi, \text{associator, commutator, def_omultable, o_conjug, oinv, omul, omultable, onorm, oversion, purevectorpart, realpart} \]

Number of procedures in Octonion = 12

> p:=1-e2+3*e7; #assigning an octonion
  type(p, octonion);
  oinv(p); #octonionic inverse
  oinv(p) &o p; #verification of the inverse
  onorm(p); #octonionic norm
  o_conjug(p); #octonionic conjugation

\( p := 1 - e2 + 3 e7 \)

\( \text{true} \)

\( \frac{1}{11} + \frac{e2}{11} - \frac{3 e7}{11} \)

\( \text{Id} \)

\( \sqrt{11} \)

\( 1 + e2 - 3 e7 \)

> #?Octonion

Additional Maple packages available at http://www.math.tntech.edu/rafal/:

- **code_support** (with Bertfried Fauser) - a small package for automatic copying and editing Maple help worksheets, and for their inserting into a
2. Notation and basic computation

Standard bases: $(V$ linear space, $W = \wedge V$ Grassmann multivectors) $V =$ span \{e.i \mid i \in \{1,\ldots,n\}\} with n in \{1,\ldots,9\} \quad \wedge V = W =$ span \{ e.i \wedge e.j \wedge \ldots \wedge e.k \mid i,j,k \in \{1..n\}\} with n in \{1..9\}

In Clifford these basis Grassmann monomials are normally written as \{Id,
e1,...,e9, e1we2,e1we3,..., e1we2we3, etc.}

> restart:with(Clifford):with(linalg):

\[
V=\text{cbasis}(2,1), W=\text{cbasis}(2); \text{cbasis}(3); \text{cbasis}(3, '\text{even}')
\]

\[
V = [e1, e2], W = [Id, e1, e2, e1we2]
\]

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

\[
[ Id, e1we2, e1we3, e2we3]
\]

Observe that expressions Id, e1we2, e1we3, e2we3, etc. are of Maple type symbol and can be manipulated as strings.

> map(type, %, symbol);

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

Command extract extracts indices of a Grassmann monomial whereas procedures makeclibasmon makes a Grassmann monomial with the given indices. Note that indices may only be 1 character long! This is because, by design, dim V <= 9 in Cl(V,B).

> 'cbasis(3)'=cbasis(3);

\[
\text{cbasis}(3) = [Id, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3]
\]

To extend functionality of CLIFFORD to vector spaces of dimension 10 and higher, use Bigebra package or the GTP package (a precursor to Bigebra).

Note, that the identity element of Cl(B) is denoted as Id. User can, however, enter just 1 for it. CLIFFORD will return Id in the final output:

> cmul(1+3*e1,e3-3*e4);
$$3 \left(-3 B_{1,4} + B_{1,3}\right) \text{Id} + e3 - 3 e4 + 3 e1we3 - 9 e1we4$$

Also, procedure `cliorder` always returns Grassmann monomials with indices in standard order $1 < 2 < \ldots < 8 < 9$. When used with symbolic 1-character indices, such indices are ordered lexicographically.

```plaintext
> wedge(e3,e1);

  -e1we3

> wedge(ek,ej,ea,eb);

  -eawebwejwek
```

Aliases for the basis Grassmann monomials can be used to shorten input/output:

```plaintext
> makealiases(3); #one can alias wedge monomials as it is shown
  eval(%); #this is needed to define them

alias(e12 = e1we2, e21 = e2we1, e13 = e1we3, e31 = e3we1, e23 = e2we3, e32 = e3we2,
      e123 = e1we2we3, e132 = e1we3we2, e213 = e2we1we3, e231 = e2we3we1, e312 = e3we1we2,
      e321 = e3we2we1)

```

Then, the wedge product of two Grassmann polynomials can be computed using the function `wedge` or the ampersand form `&w`. For more help on random clibasmon, climon, and clipolynom generators, see `rd_clipolynom`. Function `wedge` is of course multilinear.

```plaintext
> p1:=rd_clipolynom(3); #generating a random Clifford/Grassmann polynomial in \(V\), dim \(V\) = 3
p2:=rd_clipolynom(3); #generating a random Clifford/Grassmann polynomial in \(V\), dim \(V\) = 3
p3:=rd_clipolynom(3); #generating a random Clifford/Grassmann polynomial in \(V\), dim \(V\) = 3

  p1 := 9 e3 - 5 Id

  p2 := 2 e12 - 2 Id - 2 e3

  p3 := -3 e12 - Id - 2 e2 + 3 e23
```

```plaintext
> wedge(p1,p2),p1 &w p2;
```

12
wedge(p1,p2,p3), `&w` (p1,p2,p3);

\[18 \, e123 - 10 \, e12 - 8 \, e3 + 10 \, Id, \quad 18 \, e123 - 10 \, e12 - 8 \, e3 + 10 \, Id\]

\[6 \, e123 - 20 \, e12 + 8 \, e3 - 10 \, Id + 14 \, e23 - 20 \, e2, \quad 6 \, e123 - 20 \, e12 + 8 \, e3 - 10 \, Id + 14 \, e23 - 20 \, e2\]

The Clifford product in \(\text{Cl}(B)\) is expanded into Grassmann multivectors. It is given by the procedure \(\text{cmul}\) with infix form `&c`. Function \(\text{cmul}\) is of course multilinear.

NOTE: By default, computations are performed w.r.t. a bilinear form \(B\) even if it has not been defined yet. Symbolic coefficients are allowed.

\[\text{cmul}(e1,e2), &c(e1,e2); \text{cmul}(e_i,e_j,e_k);\]

\[B_{1,2} \, Id + e12, B_{1,2} \, Id + e12\]

\[B_{j,k} \, e_i - B_{i,k} \, e_j + B_{i,j} \, e_k + eiwejwek\]

\[\text{cmul}(p1,p2,p3);\]

\[-2 \, (5 - 18 \, B_{3,2} \, B_{1,2} + 15 \, B_{2,2} \, B_{1,3} - 15 \, B_{2,3} \, B_{1,2} + 18 \, B_{3,1} \, B_{2,2} - 8 \, B_{3,2} - 15 \, B_{2,1} \, B_{1,2} + 15 \, B_{2,2} \, B_{1,1} - 9 \, B_{3,3}) \, Id\]

\[-2 \, (-27 \, B_{3,2} \, B_{2,1} + 3 \, B_{3,2} - 10 \, B_{2,2} + 27 \, B_{3,3} \, B_{2,2} - 27 \, B_{3,2} \, B_{2,3} + 27 \, B_{3,2} \, B_{1,2}) \, e1 + 2(27 \, B_{3,3} \, B_{1,2} - 27 \, B_{3,1} \, B_{2,1} + 3 \, B_{3,1} + 27 \, B_{3,1} \, B_{1,2} - 10 \, 10 \, B_{1,2} - 27 \, B_{3,1} \, B_{2,3} + 30 \, B_{3,3}) \, e2\]

\[+ 2 \, (-27 \, B_{2,1} \, B_{1,2} - 12 \, B_{3,2} + 27 \, B_{2,2} \, B_{1,1} + 4 + 27 \, B_{3,1} \, B_{2,2} + 27 \, B_{2,2} \, B_{1,3} - 27 \, B_{2,3} \, B_{1,2} - 27 \, B_{3,2} \, B_{1,2}) \, e3 - 2 \, (10 - 27 \, B_{3,3} + 15 \, B_{1,2} - 15 \, B_{2,1} - 15 \, B_{2,3}) \, e12 + 6 \, B_{2,2} \, e13\]

\[-2 \, (-7 + 27 \, B_{3,3} + 3 \, B_{1,2}) \, e23 + 6 \, (-9 \, B_{2,3} - 9 \, B_{2,1} + 9 \, B_{1,2} + 1) \, e123\]

Simultaneous computation in \(\text{Cl}(K)\) and \(\text{Cl}(B)\) can be performed since \(\text{cmul}\) can accept a name of a bilinear form as a parameter. Also, symbolic indices can be used. For example,

\[\text{cmul}[K](e1,e2); &c[K](e1,e2); \text{cmul}[K](e_i,e_j,e_k);\]

\[K_{1,2} \, Id + e12\]
\[ K_{1,2} \text{Id} + e12 \]
\[ K_{j,k} e_i - K_{i,k} e_j + K_{i,j} e_k + \text{eiwejwek} \]

Of course, form B or K can be numeric/symbolic. For example,

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

then Grassmann basis for Cl(B) or \( \wedge V \) will be:

\[
cbas := [\text{Id}, e1, e2, e12]
\]

and the Grassmann multiplication table of the basis Grassmann monomials will be as follows:

\[
\text{wedgeable} := \begin{bmatrix} \text{Id} & e1 & e2 & e12 \\ e1 & 0 & e12 & 0 \\ e2 & -e12 & 0 & 0 \\ e12 & 0 & 0 & 0 \end{bmatrix}
\]

Of course, the Clifford multiplication table of the basis Grassmann monomials will be as follows:

\[
\text{Cliffordtable} := \begin{bmatrix} \text{Id} & e1 & e2 & e12 \\ e1 & \text{Id} & a \text{Id} + e12 & -a e1 + e2 \\ e2 & a \text{Id} - e12 & \text{Id} & -e1 + a e2 \\ e12 & a e1 - e2 & e1 - a e2 & (-1 + a^2) \text{Id} \end{bmatrix}
\]

In general, when B is not assigned, we get

\[
B := 'B' ;
\]

\[
\text{Cliffordtable} := \begin{bmatrix} [\text{Id}, e1, e2, e12] \\ [e1, B_{1,1} \text{Id}, B_{1,2} \text{Id} + e12, -B_{1,2} e1 + B_{1,1} e2] \end{bmatrix}
\]
In particular, let $B = g + F$ where $g$ is the symmetric part of $B$ whereas $F$ is the antisymmetric part of $B$:

$$g,F := \text{matrix}(2, 2, [g_{11}, g_{12}, g_{12}, g_{22}], \text{matrix}(2, 2, [0, F_{12}, -F_{12}, 0]));$$

$$B := \text{evalm}(g + F);$$

$$g, F := \begin{bmatrix} g_{11} & g_{12} \\ g_{12} & g_{22} \end{bmatrix}, \begin{bmatrix} 0 & F_{12} \\ -F_{12} & 0 \end{bmatrix}$$

$$B := \begin{bmatrix} g_{11} & g_{12} + F_{12} \\ g_{12} - F_{12} & g_{22} \end{bmatrix}$$

$$cmul[g](e1, e2) + cmul[g](e2, e1); \ # \text{standard generator relation in } Cl(g)$$

$$2g_{12}Id$$

$$cmul[B](e1, e2) + cmul[B](e2, e1); \ # \text{standard generator relation in } Cl(g+F)$$

$$(g_{12} + F_{12})Id - (-g_{12} + F_{12})Id$$

$$\text{clisort(simplify(%)};$$

$$2g_{12}Id$$

3. Built-in database on Clifford algebras $Cl(Q)$

Information about all Clifford algebras $Cl(p,q)$, $1 <= p+q <= 9$, for any signatures $(p,q)$ of a quadratic form $Q$ has been precomputed and is stored in CLIFFORD. It can be retrieved with a procedure `clidata`. In particular, we get a list that includes a dimension of isomorphic matrix rings over a division ring $D$; information whether algebra is simple or semi-simple; a primitive idempotent $f$ that has been used to precompute and store spinor representation of $Cl(p,q)$ in a left spinor ideal $S = Cl(p,q)f$; generators for $S$ as a real vector space; generators for $D$ inside $Cl(p,q)$; and generators for $S$ as a right module over $D$. Additional way to list various Clifford algebras according to their spinor representations is a procedure `all_sigs`. 
Data about Clifford algebras $\text{Cl}(Q)$ is stored in Maple data files matrealL.m, matrealR.m, matcompL.m, matcompR.m, matquatL.m, and matquatR.m which are part of the CLIFFORD archive. They can be accessed directly.

```maple
> restart: with(Clifford): with(linalg):
clidata([2,0]); #Clifford algebra of the Euclidean plane $\mathbb{R}^2$

\[
\begin{bmatrix}
\text{real, 2, simple, } \frac{\text{Id}}{2} + \frac{e_1}{2}, [\text{Id}, e_2], [\text{Id}], [\text{Id}, e_2]
\end{bmatrix}
\]

> matKrepr([2,0]); #matrices of the basis vectors $e_1$ and $e_2$ in spinor representation

Cl1plus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

\[
\begin{bmatrix}
e_1 &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \\
e_2 &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}
\end{bmatrix}
\]

> B:=diag(1,1,1): clidata([3,0]); #Clifford algebra of $\mathbb{R}^3$

\[
\begin{bmatrix}
\text{complex, 2, simple, } \frac{\text{Id}}{2} + \frac{e_1}{2}, [\text{Id}, e_2, e_3, e_2w_3], [\text{Id}, e_2w_3], [\text{Id}, e_2]
\end{bmatrix}
\]

> matKrepr([3,0]); #matrices of the basis vectors $e_1$, $e_2$, $e_3$ in spinor representation

\[
\begin{bmatrix}
e_1 &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \\
e_2 &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \\
e_3 &= \begin{bmatrix} 0 & -e_2w_3 \\ e_2w_3 & 0 \end{bmatrix}
\end{bmatrix}
\]

Notice that the above three matrices representing three basis vectors $e_1$, $e_2$, $e_3$ have entries in $\{1, e_{23}\} = \mathbb{C}$:

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

\[
\begin{bmatrix}
M1, M2, M3 := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & -e_2w_3 \\ e_2w_3 & 0 \end{bmatrix}
\end{bmatrix}
\]

and, of course, they satisfy the same defining relations as the basis vectors themselves. For example,

> `M1 &cm M2 + M2 &cm M1` = evalm(M1 &cm M2 + M2 &cm M1); #computations with Clifford matrices

`e1 &c e2 + e2 &c e1` = e1 &c e2 + e2 &c e1;
```
\[ M1 \& cm M2 + M2 \& cm M1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \]

\[ e1 \& c e2 + e2 \& c e1 = 0 \]

\[ 'M1 \& cm M1' = \text{evalm}(M1 \& cm M1); \]
\[ 'e1 \& c e1' = e1 \& c e1; \]

\[ M1 \& cm M1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \]

\[ e1 \& c e1 = \text{Id} \]

The data files can be accessed directly. For example,

\[ \texttt{indices(matrealL)}; \]  
#display indices of 'matrealL'

\[[[5, 4]], [[3, 1]], [[1, 8]], [[2, 2]], [[0, 8]], [[2, 0]], [[1, 7]], [[0, 6]], [[1, 1]],
[[8, 0]], [[3, 3]], [[0, 7]], [[4, 2]], [[2, 1]], [[3, 2]], [[9, 0]], [[4, 4]], [[5, 3]],
[[4, 3]]\]

Show Dirac gamma matrices representing basis 1-vectors in the signature (2,2):

\[ \texttt{matrealL[[2, 2]]}; \]

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

Show matrices representing basis 1-vectors in the signature (2,0):

\[ \texttt{matrealL[[2, 0]]}; \]

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

Let's see the indices of matcompL and signatures for which complex matrices have been computed in S=Cl(Q)f.

\[ \texttt{indices(matcompL)}; \]  
#display indices of 'matcompL'

\[[[3, 4]], [[5, 2]], [[6, 3]], [[7, 0]], [[0, 9]], [[8, 1]], [[0, 5]], [[4, 1]], [[4, 5]],
[[2, 3]], [[2, 7]], [[1, 2]], [[3, 0]], [[1, 6]]\]

Show Pauli matrices representing basis 1-vectors in the signature (3,0) over
the field $K$ spanned by $[\text{Id}, e_{2w^3}]$:

$\begin{bmatrix}
ed1 &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},
ed2 &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},
ed3 &= \begin{bmatrix} 0 & -e_{2w^3} \\ e_{2w^3} & 0 \end{bmatrix}
\end{bmatrix}$

One can also quickly list signatures of a quadratic form $Q$ of signature $(p,q)$ that give simple/semisimple Clifford algebras $\text{Cl}(Q)$.

$\text{all_sigs(1..9,'real');}$ # signatures of all $\text{Cl}(p,q)$ represented by real matrices

$[[0, 6], [0, 7], [0, 8], [1, 0], [1, 1], [1, 7], [1, 8], [2, 0], [2, 1], [2, 2], [3, 1], [3, 2], [3, 3], [4, 2], [4, 3], [4, 4], [5, 3], [5, 4], [8, 0], [9, 0]]$

$\text{all_sigs(1..9,'real','simple');}$ # signatures of simple $\text{Cl}(p,q)$

$[[0, 6], [0, 8], [1, 1], [1, 7], [2, 0], [2, 2], [3, 1], [3, 3], [4, 2], [4, 4], [5, 3], [8, 0]]$

$\text{all_sigs(1..9,'real','semisimple');}$ # signatures of semisimple $\text{Cl}(p,q)$

$[[0, 7], [1, 0], [1, 8], [2, 1], [3, 2], [4, 3], [5, 4], [9, 0]]$

For example, $\text{Cl}(2,1)$ is a semisimple Clifford algebra isomorphic with $\text{Mat}(2,\mathbb{R}) + \text{Mat}(2,\mathbb{R}) = \text{Mat}(2,\mathbb{R}+\mathbb{R})$, that is, a ring of 2 by 2 matrices with entries in the double field $\mathbb{R}+\mathbb{R}$:

$\text{B:=diag(1,1,-1);clidata([2,1]);}$

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

$\text{matKrepr([2,1]);}$

$\begin{bmatrix}
ed1 &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},
ed2 &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},
ed3 &= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}
\end{bmatrix}$

$\text{M1,M2,M3:=rhs([1]),rhs([2]),rhs([3]);}$

$\begin{bmatrix}
M1, M2, M3 &:=& \begin{bmatrix} [1, -1] & [0, 0] \\ [0, 0] & [-1, 1] \end{bmatrix}, \begin{bmatrix} [0, 0] & [1, 1] \\ [1, 1] & [0, 0] \end{bmatrix}, \begin{bmatrix} [0, 0] & [-1, -1] \\ [1, 1] & [0, 0] \end{bmatrix}
\end{bmatrix}$
\( M_3; \)

\[
M_1 \, \text{cm} \, M_1, \, M_2 \, \text{cm} \, M_2, \, M_3 \, \text{cm} \, M_3 = \begin{bmatrix}
[1, 1] & [0, 0] \\
[0, 0] & [1, 1]
\end{bmatrix}
\begin{bmatrix}
[1, 1] & [0, 0] \\
[0, 0] & [1, 1]
\end{bmatrix}
\begin{bmatrix}
[-1, -1] & [0, 0] \\
[0, 0] & [-1, -1]
\end{bmatrix}
\]

> `M1 \, \text{cm} \, M2 + M2 \, \text{cm} \, M1` = \text{adfmatrix}(M1 \, \text{cm} \, M2,M2 \, \text{cm} \, M1);

\[
M_1 \, \text{cm} \, M2 + M2 \, \text{cm} \, M1 = \begin{bmatrix}
[0, 0] & [0, 0] \\
[0, 0] & [0, 0]
\end{bmatrix}
\]

> \text{all_sigs}(1..9, 'quat');

\[
([0, 2], [0, 3], [0, 4], [1, 3], [1, 4], [1, 5], [2, 4], [2, 5], [2, 6], [3, 5], [3, 6], [4, 0], [5, 0], [5, 1], [6, 0], [6, 1], [6, 2], [7, 1], [7, 2])
\]

> \text{B:=diag}(1,-1,-1,-1);clidata([1,3]);

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

Cl(1,3) has a spinor representation given in terms of 2 by 2 quaternionic matrices whose entries belong to a subring of Cl(1,3) generated by \{Id,e2,e3,e2we3\} = \text{H}:

> \text{matKrepr([1,3]); # quaternionic matrices representing e1,e2,e3,e4}

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

> \text{gamma(1-2*e2+e2we3-3*e2we4)=matKrepr(1-2*e2+e2we3-3*e2we4)};

\[
\chi(1 - 2 \, e2 + e2we3 - 3 \, e2we4) = \begin{bmatrix}
1 - 2 \, e2 + e2we3 & 3 \, e2 \\
3 \, e2 & 1 + 2 \, e2 + e2we3
\end{bmatrix}
\]
4. Linearity, bilinearity, and multilinearity

Linearity with respect to \_scalartypes is handled by the procedure clilinear, whereas bilinearity is handled by clibilinear. Multilinearity is then implemented by a recursive application of the bilinearity procedure.

> restart: with(Clifford);

[ &m, Bsigs, CLIFFORD_ENV, Kfield, LC, LCQ, RC, RCQ, RHnumber, adfmatrix,
  all_sigs, beta_minus, beta_plus, buildm, bygrade, c_conjug, cbasis, cdfmatrix, cexp, cexpQ,
  cinv, clibilinear, clicollect, clidata, clilinear, climinpoly, cliparse, cliremove, clisolve, clisort,
  cliterms, cmul, cmulNUM, cmulQ, cmulRS, cmulgen, cocycle, commutingelements,
  conjugation, ddfmatrix, diagonalize, displayid, extract, factoridempotent, find1str, findbasis,
  gradeinv, init, isVahlenmatrix, isproduct, makealiases, makeclibasmon, matKrepr, maxgrade,
  maxindex, mdfmatrix, minimalideal, ord, permsign, pseudodet, q_conjug, qdisplay, qinv, qmul,
  qnorm, rd_clibasmon, rd_climon, rd_clipolynom, reorder, reversion, rmulm, rot3d, scalarpart
  sexp, specifyconstants, spinorKbasis, spinorKrepr, squaremodf, subs_clipolynom,
  useproduct, vectorpart, version, wedge, wexp ]

User can modify _scalartypes as follows:

> _scalartypes; #default scalar types loaded by CLIFFORD_ENV

    { ^, RootOf, complex, indexed, numeric, constant, function, mathfunc, rational }

> _scalartypes := {op(_scalartypes), polynom}; #modified
  _scalartypes: polynomials of type 'polynom' are now 'scalars'

> _scalartypes :=
    { ^, RootOf, complex, indexed, numeric, polynom, constant, function, mathfunc, rational }

Now, we have multilinearity with respect to the modified scalars:

> c1, c2 := 2*x^6*y^4-z, -3*x*y-z^4;
  p2 := c1*e1we3-c2*(1-3*e1+e2);
  p1 := rd_clipolynom(3);

  c1, c2 := 2x^6y^4-z, -3xy-z^4
  p2 := (2x^6y^4-z)e1we3 + (3xy-z^4)(1-3e1+e2)
  p1 := 9e3-5Id

> cmul(c1*p1, p2);
This is how it can be made into a bilinear procedure:

\[ 2 x^6 y^4 K(Id) - z K(Id) + 2 K(e2) - 3 x y K(elwe2) - z^4 K(elwe2) - K(e3) \]

There are many important procedures that are multilinear, the most important being the wedge product and the Clifford product cmul.

\[ -5 (z^4 + 3 x y) p3 Id - (10 x^6 y^4 - 5 z - 27 z^4 - 81 x y) p3 elwe3 - 9 (z^4 + 3 x y) p3 e2we3 \]
\[ + 15 (z^4 + 3 x y) p3 e2 - 5 (z^4 + 3 x y) p3 e2 + 9 (z^4 + 3 x y) p3 e3 \]
Procedure **clicollect** collects monomial terms in the output. It is typically built into CLIFFORD procedures but sometimes user needs to call it. For example, the above two outputs from wedge and cmul have been clicollected.

```maple
> cmul(p2,p1)+wedge(p1,p2);
```

```
-27 x y B_{2,3} + 5 z^4 + 15 x y + 81 x y B_{1,3} + 27 z^4 B_{1,3} - 9 z^4 B_{2,3}) Id
  + 3 (15 x y - 3 z B_{3,3} + 6 x^6 y^4 B_{3,3} + 5 z^4) e1 - 10 (z^4 + 3 x y) e2
  - 9 (-3 x y - z B_{1,3} + 2 x^6 y^4 B_{1,3} - z^4) e3 - (-5 z + 81 x y + 27 z^4 + 10 x^6 y^4) e1we3
  + 9 (z^4 + 3 x y) e3 - 5 (z^4 + 3 x y) Id - 5 (2 x^6 y^4 - z) e1we3 + 27 (z^4 + 3 x y) e1we3
  + 15 (z^4 + 3 x y) e1
```

```maple
> clicollect(%);
```

```
-(30 x y - 9 z^4 B_{2,3} + 81 x y B_{1,3} - 27 x y B_{2,3} + 10 z^4 + 27 z^4 B_{1,3}) Id
  - 10 (2 x^6 y^4 - z) e1we3 + 3 (-3 z B_{3,3} + 6 x^6 y^4 B_{3,3} + 30 x y + 10 z^4) e1 - 10 (z^4 + 3 x y) e2
  - 9 (-z B_{1,3} + 2 x^6 y^4 B_{1,3} - 6 x y - 2 z^4) e3
```

Users often use various symbols in their symbolic computations with CLIFFORD. It is impossible of course to predict what symbolic coefficients they will use in Clifford polynomials, for example. It may happen that **cliparse** procedure that checks input for correct syntax gets 'confused' and returns false or an error message. It is especially sensitive when user's input looks similar to how basis Grassmann monomials are defined. This procedure parses user's input by the default set through an environmental variable **_prolevel** which is set to false when CLIFFORD is loaded.

```maple
> restart:with(Clifford):
cliparse(e1we2);
```

```
true
```
The purpose of setting _prolevel to true is to speed up computations as then
user's input in many procedures is then not parsed, and types are not
checked. However, errors may of course result (see next) when one enters
wrong input. For example, quantity e1e2 does not raise a flag now that
_prolevel = true, but we get this false, most likely, result:

> cmul(e1e2,Id);

That is, in case like that behavior is rather unpredictable. It is therefore safer
when one starts using CLIFFORD to keep _prolevel set to false.

Thus, there is another mechanism to tell CLIFFORD that certain symbols
should be treated as constants or, as scalars. This is accomplished with a
procedure specify_constants. For example, a user wants 'e1e2' to be a
symbolic scalar coefficient in the following Clifford polynomial f1:

> _prolevel:=false:
    f1:= e1e2*e1-alpha2*e1we2;

> cliparse(f1);###<-- intended error message

To avoid this problem define e1e2 as a constant:
5. Manipulating CLIFFORD input and output

There are several procedures that help user manipulate CLIFFORD input and output. Most of these procedures are used internally by other procedures before they return results. We have seen already cliparse and clicollect. Some other procedures are:

- **cbasis** - gives a list of basis Grassmann monomials, also of specific grade

\[
basis(3); \text{#all basis monomials that span Grassmann algebra when dim V = 3;}
basis(3,2); \text{#all basis monomials that span a subspace of 2-vectors in the Grassmann algebra when dim V = 3;}
\]

\[
\{1d, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3\}
\]

\[
\{e1we2, e1we3, e2we3\}
\]

- **cliterms** - displays Grassmann monomials in the given Clifford polynomial

\[
p := -3*e4we5 + 7*e1we2we3 - 3*e2we3 - 3*e3we5 - 3*e1we2we5;
S := cliterms(p);
\]

\[
p := -3 e4we5 + 7 e1we2we3 - 3 e2we3 - 3 e3we5 - 3 e1we2we5
\]

Clplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

\[
S := \{e2we3, e3we5, e4we5, e1we2we3, e1we2we5\}
\]

- **bygrade** - permits sorting a list of Grassmann monomials by grade and within each grade, by the lex order e1 > e2 > ... > e9.

\[
L := convert(S, list);
\]
sort(L, bygrade);

\[ L := \{ e2we3, e3we5, e4we5, e1we2we3, e1we2we5 \} \]

\[ \{ e2we3, e3we5, e4we5, e1we2we3, e1we2we5 \} \]

- **clisort** - sorts Grassmann monomials in a Clifford polynomial so that scalar coefficients appear before the basis monomial terms:

```maple
> p := e1we3*4 - Id*a;
clisort(p);

\[ p := 4 e1we3 - Id a \]

\[ \neg Id + 4 e1we3 \]

- **displayid** - displays scalar \( k \) as \( k \cdot Id \) in a Clifford polynomial

```maple
> f := 3 - 3*e1we3;
displayid(f);

\[ f := 3 - 3 e1we3 \]

\[ 3 Id - 3 e1we3 \]

- **maxgrade** - displays maximum grade in a Clifford polynomial

- **maxindex** - displays maximum index in a Clifford polynomial

```maple
> p := -5*Id - 5*e2we4we5 - 5*e1we2we8 + 3*e3we7 - 5*e3we7we8;
maxgrade(p);
maxindex(p);

\[ p := -5 Id - 5 e2we4we5 - 5 e1we2we8 + 3 e3we7 - 5 e3we7we8 \]

\[ 3 \]

\[ 8 \]

- **reorder** - this procedure reorders Grassmann monomials so that their indices are written in the standard ascending order. This, of course, results often in extra sign changes. See ?permsign procedure for more.

```maple
> p := 2*e4we2we3 - 6 + e7we3we1we2;
reorder(p);
```
\[ p := 2\ e4we2we3 − 6 + e7we3we1we2 \]
\[ 2\ e2we3we4 − 6 − e1we2we3we7 \]

Finally, procedures \texttt{scalarpart} and \texttt{vectorpart} identify parts of various grades in a Clifford polynomial. Note the difference between \texttt{scalarpart}(p) and \texttt{vectorpart}(p,0) below:

```plaintext
> p:=-4*Id-4*e2we3we8+2*e2we3we4we5we6we8we9-4*e1we2we5we6we8we9+2*e1we2we3we4we6we9+2*e1we4we7we9;
  scalarpart(p);
  vectorpart(p,0);
  maxgrade(p);
  for i from 1 to maxgrade(p) do
      'vectorpart'(p,i)=vectorpart(p,i);
  end do;

p := −4 \text{Id} − 4 \text{e}2\text{we}3\text{we}8 + 2 \text{e}2\text{we}3\text{we}4\text{we}5\text{we}6\text{we}8\text{we}9 − 4 \text{e}1\text{we}2\text{we}5\text{we}6\text{we}8\text{we}9
    + 2 \text{e}1\text{we}2\text{we}3\text{we}4\text{we}6\text{we}9 + 2 \text{e}1\text{we}4\text{we}7\text{we}9

-4

-4 \text{Id}

7

\text{vectorpart}(-4 \text{Id} − 4 \text{e}2\text{we}3\text{we}8 + 2 \text{e}2\text{we}3\text{we}4\text{we}5\text{we}6\text{we}8\text{we}9 − 4 \text{e}1\text{we}2\text{we}5\text{we}6\text{we}8\text{we}9
    + 2 \text{e}1\text{we}2\text{we}3\text{we}4\text{we}6\text{we}9 + 2 \text{e}1\text{we}4\text{we}7\text{we}9, 1) = 0

\text{vectorpart}(-4 \text{Id} − 4 \text{e}2\text{we}3\text{we}8 + 2 \text{e}2\text{we}3\text{we}4\text{we}5\text{we}6\text{we}8\text{we}9 − 4 \text{e}1\text{we}2\text{we}5\text{we}6\text{we}8\text{we}9
    + 2 \text{e}1\text{we}2\text{we}3\text{we}4\text{we}6\text{we}9 + 2 \text{e}1\text{we}4\text{we}7\text{we}9, 2) = 0

\text{vectorpart}(-4 \text{Id} − 4 \text{e}2\text{we}3\text{we}8 + 2 \text{e}2\text{we}3\text{we}4\text{we}5\text{we}6\text{we}8\text{we}9 − 4 \text{e}1\text{we}2\text{we}5\text{we}6\text{we}8\text{we}9
    + 2 \text{e}1\text{we}2\text{we}3\text{we}4\text{we}6\text{we}9 + 2 \text{e}1\text{we}4\text{we}7\text{we}9, 3) = −4 \text{e}2\text{we}3\text{we}8

\text{vectorpart}(-4 \text{Id} − 4 \text{e}2\text{we}3\text{we}8 + 2 \text{e}2\text{we}3\text{we}4\text{we}5\text{we}6\text{we}8\text{we}9 − 4 \text{e}1\text{we}2\text{we}5\text{we}6\text{we}8\text{we}9
    + 2 \text{e}1\text{we}2\text{we}3\text{we}4\text{we}6\text{we}9 + 2 \text{e}1\text{we}4\text{we}7\text{we}9, 4) = 2 \text{e}1\text{we}4\text{we}7\text{we}9

\text{vectorpart}(-4 \text{Id} − 4 \text{e}2\text{we}3\text{we}8 + 2 \text{e}2\text{we}3\text{we}4\text{we}5\text{we}6\text{we}8\text{we}9 − 4 \text{e}1\text{we}2\text{we}5\text{we}6\text{we}8\text{we}9
    + 2 \text{e}1\text{we}2\text{we}3\text{we}4\text{we}6\text{we}9 + 2 \text{e}1\text{we}4\text{we}7\text{we}9, 5) = 0

\text{vectorpart}(-4 \text{Id} − 4 \text{e}2\text{we}3\text{we}8 + 2 \text{e}2\text{we}3\text{we}4\text{we}5\text{we}6\text{we}8\text{we}9 − 4 \text{e}1\text{we}2\text{we}5\text{we}6\text{we}8\text{we}9
    + 2 \text{e}1\text{we}2\text{we}3\text{we}4\text{we}6\text{we}9 + 2 \text{e}1\text{we}4\text{we}7\text{we}9, 6) = −4 \text{e}1\text{we}2\text{we}5\text{we}6\text{we}8\text{we}9 + 2 \text{e}1\text{we}2\text{we}3\text{we}4\text{we}6\we9
```
6. Clifford product in Cl(B): cmulNUM and cmulRS

> restart; with(Clifford):

Upon loading CLIFFORD, one of the environmental variables displayed by Clifford:-CLIFFORD_ENV is _default_Clifford_product. The default value of that variable is Clifford:-cmulRS which gives the Clifford product computed using the Rota-Stein cliffordization technique based on a combinatorial and non-recursive approach (see package Bigebra for more help on the subject). The default values can be changed by the user. In particular, user can include additional Maple types in _scalartypes to allow computations with Clifford modules.

> CLIFFORD_ENV();

' >>> Global variables describing a default system defined in Clifford:-setup are now available and have these values: <<< '
' ************* Start *************
_processor_type = "Intel (R) Core (TM) 2 Duo CPU 2.19 GHz, 2.9 GB RAM"
_operating_system = "Win XP Professional (SP3)"
' ************* End *************

' >>> Global variables defined in Clifford:-setup are now available and have these values: <<< '
' ************* Start *************
dim_V = 9
_default_Clifford_product = Clifford:-cmulRS
_prolevel = false
_shortcut_in_minimalideal = true
_shortcut_in_Kfield = true
_shortcut_in_spinorKbasis = true
_shortcut_in_spinorKrepr = true
_warnings_flag = true
_scalartypes = {`^`, RootOf, complex, indexed, numeric, constant, function, mathfunc, rational}
_quatbasis = [[Id, e3we2, e1we3, e2we1], {`Maple has assigned qj:=e2we3, qj:e1we3, qk:=e1we2`)}
' ************* End *************

Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.
Global variables defined in Cliplus:-setup are now available and have these values: <<<

************* Start *************

macro(Clifford:-cmul = climul)
macro(Clifford:-cmulQ = climul)
macro(`&c` = climul)
macro(`&cQ` = climul)
macro(Clifford:-reversion = clirev)
macro(Clifford:-LC = LCbig)
macro(Clifford:-RC = RCBig)

`Warning, new definitions for type/climon and type/clipolynom now include &C`

************* End *************

<<< There are no new global variables or macros in GTP yet. <<<

************* End *************

Global variables defined in Octonion:-setup are now available and have these values: <<<

************* Start *************

_octbasis = [Id, e1, e2, e3, e4, e5, e6, e7]
_pureoctbasis = [e1, e2, e3, e4, e5, e6, e7]
_default_Fano_triples = [[1, 3, 7], [1, 2, 4], [1, 5, 6], [2, 3, 5], [2, 6, 7], [3, 4, 6], [4, 5, 7]]
_default_squares = [-Id, -Id, -Id, -Id, -Id, -Id]
_default_Clifford_product = Clifford:-cmulNUM

************* End *************

Procedure \texttt{cmulNUM} is based on Chevalley’s recursive definition of the Clifford product as a deformation of the Grassmann product in a Grassmann algebra. It uses a left contraction \texttt{LC} that has the following properties:

1. \( LC(x,y) = B(x,y)\cdot Id \) for any two 1-vectors \( x \) and \( y \) in \( V \), and a bilinear form \( B \) in \( V \).

\begin{verbatim}
> x:=a1*e1+a2*e2+a3*e3;
y:=b1*e1+b2*e2+b3*e3;
z:=c1*e1+c2*e2+c3*e3;
clicollect(LC(x,y));

x := a1 e1 + a2 e2 + a3 e3
\end{verbatim}
\[ y := b_1 e_1 + b_2 e_2 + b_3 e_3 \]
\[ z := c_1 e_1 + c_2 e_2 + c_3 e_3 \]
\[ (a_1 b_1 B_{1,1} + a_1 b_2 B_{1,2} + a_1 b_3 B_{1,3} + a_2 b_1 B_{2,1} + a_2 b_2 B_{2,2} + a_2 b_3 B_{2,3} + a_3 b_1 B_{3,1} \\
+ a_3 b_2 B_{3,2} + a_3 b_3 B_{3,3} ) ld \]

2. \( \text{LC}(x, y \wedge z) = \text{LC}(x, y) \wedge z - \text{LC}(x, z) \wedge y \) where \( x, y, z \) are 1-vectors, and with this property extended to \( \text{LC}(x, u) \) where \( u \) is any \( k \)-vector

\[
\text{> clicollect(\text{LC}(x, \text{wedge}(y, z)))}; \ #\text{lhs od 2}. \\
\text{\(-(a_2 b_2 c_1 B_{2,2} - a_1 b_2 c_1 B_{1,2} - a_2 b_3 c_1 B_{2,3} - a_1 b_3 c_1 B_{1,3} + a_2 b_1 c_2 B_{2,2} \\
+ a_1 b_1 c_2 B_{1,2} - a_3 b_3 c_1 B_{3,3} + a_3 b_1 c_3 B_{3,3} + a_2 b_1 c_3 B_{2,3} + a_1 b_1 c_3 B_{1,3} \\
- a_3 b_2 c_1 B_{3,2} + a_3 b_1 c_2 B_{3,2} ) e_1 + (a_2 b_3 c_2 B_{2,3} - a_1 b_2 c_3 B_{1,3} - a_1 b_2 c_1 B_{1,1} \\
- a_2 b_2 c_3 B_{2,3} + a_3 b_1 c_2 B_{3,1} + a_1 b_1 c_2 B_{1,1} + a_3 b_3 c_2 B_{3,3} + a_1 b_3 c_2 B_{1,3} \\
- a_2 b_2 c_1 B_{2,1} - a_3 b_2 c_3 B_{3,3} - a_3 b_2 c_1 B_{3,1} + a_2 b_1 c_2 B_{2,1} ) e_2 + (-a_1 b_3 c_2 B_{1,2} \\
+ a_1 b_2 c_3 B_{1,2} - a_3 b_3 c_1 B_{3,1} - a_2 b_3 c_1 B_{2,1} - a_1 b_3 c_1 B_{1,1} + a_1 b_1 c_3 B_{1,1} \\
- a_3 b_3 c_2 B_{3,2} - a_2 b_3 c_2 B_{2,2} + a_3 b_1 c_3 B_{3,1} + a_2 b_1 c_3 B_{2,1} + a_3 b_2 c_3 B_{3,2} \\
+ a_2 b_2 c_3 B_{2,2} ) e_3 \)
\]

\[
\text{> clicollect(\text{wedge}(\text{LC}(x, y)), z) - \text{wedge}(\text{LC}(x, z), y)); \\
\text{\(-(a_2 b_2 c_1 B_{2,2} - a_1 b_2 c_1 B_{1,2} - a_2 b_3 c_1 B_{2,3} - a_1 b_3 c_1 B_{1,3} + a_2 b_1 c_2 B_{2,2} \\
+ a_1 b_1 c_2 B_{1,2} - a_3 b_3 c_1 B_{3,3} + a_3 b_1 c_3 B_{3,3} + a_2 b_1 c_3 B_{2,3} + a_1 b_1 c_3 B_{1,3} \\
- a_3 b_2 c_1 B_{3,2} + a_3 b_1 c_2 B_{3,2} ) e_1 + (a_2 b_3 c_2 B_{2,3} - a_1 b_2 c_3 B_{1,3} - a_1 b_2 c_1 B_{1,1} \\
- a_2 b_2 c_3 B_{2,3} + a_3 b_1 c_2 B_{3,1} + a_1 b_1 c_2 B_{1,1} + a_3 b_3 c_2 B_{3,3} + a_1 b_3 c_2 B_{1,3} \\
- a_2 b_2 c_1 B_{2,1} - a_3 b_2 c_3 B_{3,3} - a_3 b_2 c_1 B_{3,1} + a_2 b_1 c_2 B_{2,1} ) e_2 + (-a_1 b_3 c_2 B_{1,2} \\
+ a_1 b_2 c_3 B_{1,2} - a_3 b_3 c_1 B_{3,1} - a_2 b_3 c_1 B_{2,1} - a_1 b_3 c_1 B_{1,1} + a_1 b_1 c_3 B_{1,1} \\
- a_3 b_3 c_2 B_{3,2} - a_2 b_3 c_2 B_{2,2} + a_3 b_1 c_3 B_{3,1} + a_2 b_1 c_3 B_{2,1} + a_3 b_2 c_3 B_{3,2} \\
+ a_2 b_2 c_3 B_{2,2} ) e_3 \)
\]

\[
\text{> %%%}; \\
0
\]

\[
\text{> uk:=e1we2we3we4we5we6we7we8we9; #basis 9-vector} \\
\text{LC(ei,uk);} \\
\text{uk := e1we2we3we4we5we6we7we8we9}
\]
B_{i,1} e2we3we4we5we6we7we8we9 - B_{i,2} e1we3we4we5we6we7we8we9 + B_{i,3} e1we2we4we5we6we7we8we9 - B_{i,4} e1we2we3we5we6we7we8we9 + B_{i,5} e1we2we3we4we6we7we8we9 - B_{i,6} e1we2we3we4we5we7we8we9 + B_{i,7} e1we2we3we4we5we6we8we9 - B_{i,8} e1we2we3we4we5we6we7we9 + B_{i,9} e1we2we3we4we5we6we7we8

3. \text{LC}(u \wedge v, w) = \text{LC}(u, \text{LC}(v, w)) \text{ where } u, v, w \text{ are arbitrary } k-, l-, \text{ and } m\text{-vectors}

with all these properties extended by linearity and bilinearity to all 1-vectors x, y, z, and all multivectors (Clifford polynomials in Grassmann basis) u, v, and w.

\[
\begin{align*}
U &:= -4 e2 + 5 \text{Id}; \\
V &:= \text{Id} + 2 e1we3 - e2we5 + 2 e3we4 - e4we5; \\
W &:= -3 \text{Id} + 2 e2we5 - 3 e1we3 - 3 e1we4we5 - 3 e2we3we4we5; \\
\text{LC}(\wedge(U, V), W) &- \text{LC}(U, \text{LC}(V, W));
\end{align*}
\]

\[
U := -4 e2 + 5 \text{Id} \\
V := \text{Id} + 2 e1we3 - e2we5 + 2 e3we4 - e4we5 \\
W := -3 \text{Id} + 2 e2we5 - 3 e1we3 - 3 e1we4we5 - 3 e2we3we4we5
\]

0

Left contraction \text{LC} can be used with an explicitly specified bilinear form (matrix) K. Right contraction \text{RC} defined in a similar way in CLIFFORD.

The Clifford associative product \text{cmul} is defined now as

\[
\text{cmul}(x, U) = \text{LC}(x, U) + \wedge(x, U)
\]

and it is extended by higher order terms in the first slot via recursion, and, at the end, it is made bilinear. For example, when x is replaced with a 2-vector \( x \wedge y = \wedge(x, y) \), we get

\[
\text{cmul}(x \wedge y, U) = \text{cmul}(\wedge(x, y), U) = \text{cmul}((\text{cmul}(x, y) - \text{LC}(x, y) \text{Id}, U) = \text{cmul}(\text{cmul}(x, y), U) - \text{LC}(x, y) \text{cmul}(\text{Id}, U) = \text{cmul}(x, \text{cmul}(y, U)) - \text{LC}(x, y) \text{U}
\]

> \text{useproduct}(\text{cmulNUM}); #selecting 'cmulNUM'

Warning, \text{cmul} will use \text{cmulNUM}; for help see pages \?cmul, \?Clifford:-intro, or
> out1 := clicollect(cmul(wedge(x, y), U));

\[
\text{out1} := -4 \left( -a_2 b_1 B_{2,2} + a_1 b_3 B_{3,2} + a_1 b_2 B_{2,2} - a_3 b_1 B_{3,2} \right) e_1 \\
+ 4 \left( -a_2 b_1 B_{1,2} - a_2 b_3 B_{3,2} + a_3 b_2 B_{3,2} + a_1 b_2 B_{1,2} \right) e_2 \\
+ 4 \left( a_2 b_3 B_{2,2} - a_3 b_1 B_{1,2} + a_3 b_3 B_{1,2} - a_3 b_2 B_{2,2} \right) e_3 + 5 \left( a_1 b_2 - a_2 b_1 \right) e_1 \text{we} 2 \\
+ 5 \left( a_1 b_3 - a_3 b_1 \right) e_2 \text{we} 3 + 5 \left( -a_3 b_2 + a_2 b_3 \right) e_2 \text{we} 3 + 4 \left( a_1 b_3 - a_3 b_1 \right) e_1 \text{we} 2 \text{we} 3
\]

> out2 := cmul(x, cmul(y, U)) - cmul(LC(x, y), U); # notice extra terms in out2 that get cancelled in out3

\[
\text{out2} := -4 \left( -a_2 b_1 B_{2,2} + a_1 b_3 B_{3,2} + a_1 b_2 B_{2,2} - a_3 b_1 B_{3,2} \right) e_1 - 4 \left( a_2 b_1 B_{1,2} \\
+ a_2 b_1 B_{2,1} + a_3 b_3 B_{3,3} + a_2 b_3 B_{3,2} + a_1 b_3 B_{1,3} + a_2 b_2 B_{2,2} + a_3 b_1 B_{3,1} + a_1 b_1 B_{1,1} \\
+ a_2 b_3 B_{2,3} \right) e_2 + 4 \left( a_2 b_3 B_{2,2} - a_3 b_1 B_{1,2} + a_1 b_3 B_{1,2} - a_3 b_2 B_{2,2} \right) e_3 \\
+ 5 \left( a_1 b_2 - a_2 b_1 \right) e_1 \text{we} 2 + 5 \left( a_1 b_3 - a_3 b_1 \right) e_1 \text{we} 3 + 5 \left( -a_3 b_2 + a_2 b_3 \right) e_2 \text{we} 3 \\
+ 4 \left( a_1 b_3 - a_3 b_1 \right) e_1 \text{we} 2 \text{we} 3 + 4 \left( a_1 b_3 B_{1,3} + a_2 b_3 B_{2,3} + a_2 b_2 B_{2,2} + a_3 b_3 B_{3,3} \\
+ a_3 b_2 B_{3,2} + a_1 b_1 B_{1,1} + a_3 b_1 B_{3,1} + a_1 b_2 B_{1,2} + a_2 b_1 B_{2,1} \right) e_2
\]

> out3 := clicollect(out2);

\[
\text{out3} := -4 \left( -a_2 b_1 B_{2,2} + a_1 b_3 B_{3,2} + a_1 b_2 B_{2,2} - a_3 b_1 B_{3,2} \right) e_1 \\
+ 4 \left( -a_2 b_1 B_{1,2} - a_2 b_3 B_{3,2} + a_3 b_2 B_{3,2} + a_1 b_2 B_{1,2} \right) e_2 \\
+ 4 \left( a_2 b_3 B_{2,2} - a_3 b_1 B_{1,2} + a_3 b_3 B_{1,2} - a_3 b_2 B_{2,2} \right) e_3 + 5 \left( a_1 b_2 - a_2 b_1 \right) e_1 \text{we} 2 \\
+ 5 \left( a_1 b_3 - a_3 b_1 \right) e_1 \text{we} 3 + 5 \left( -a_3 b_2 + a_2 b_3 \right) e_2 \text{we} 3 + 4 \left( a_1 b_3 - a_3 b_1 \right) e_1 \text{we} 2 \text{we} 3
\]

> out1-out3;

\[
0
\]

The above clearly shows that the recursion definition of cmul has resulted in 34 monomial terms in out2 of which 14 got cancelled out (or got combined) to produce 20 monomial terms in out3 that equals out1. Generation of these extra terms slows down CLIFFORD considerably when \(\dim V = 5\) and \(B\) is purely symbolic. In that case it is recommended to use \texttt{cmulRS} that uses Rota-Stein's combinatorial approach via Hopf algebra. In this approach, no redundant terms are generated: cmulRS generates exactly those terms that must appear in the final answer. \texttt{cmulNUM} is faster than \texttt{cmulRS} in \(\dim V < 5\) and when the bilinear form \(B\) is numeric with lost of zeros, e.g., diagonal.

That's why the default is cmulRS. Note that \texttt{cmulNUM} and \texttt{cmulRS} can accept
Grassmann monomials as input whereas cmul is a wrapper function that is multilinear.

> useproduct(cmulRS); #selecting 'cmulRS'

Warning, cmul will use cmulRS; for help see pages ?cmul, ?Clifford:-intro, or ?cmulRS

We can do a small benchmarking of both algorithms.

> N:=5:
    TNUMRS:=table():
    _processor_type; ###<<<--- reassign to this global variable information about your processor as a Maple string
    _operating_system; ###<<<--- reassign to this global variable information about your OS as a Maple string

    "Intel (R) Core (TM) 2 Duo CPU 2.19 GHz, 2.9 GB RAM"

    "Win XP Professional (SP3)"

> for n from 2 to N do
    cbas:=cbasis(n):
    start:=time():
    seq(seq(cmulNUM(mm,nn,B),mm=cbas),nn=cbas):
    tNUM:=time()-start:
    start:=time():
    seq(seq(cmulRS(mm,nn,B),mm=cbas),nn=cbas):
    tRS:=time()-start:
    TNUMRS[n]:=[tNUM,tRS]:
    printf("In dimension %a times are: %g for cmulNUM and %g for cmulRS which is %g times faster running on %s under %s\n", n,TNUMRS[n][1],TNUMRS[n][2],TNUMRS[n][1]/TNUMRS[n][2],_processor_type,_operating_system);
end do:

In dimension 2 times are: 0.032 for cmulNUM and 0 for cmulRS which is Inf times faster running on Intel (R) Core (TM) 2 Duo CPU 2.19 GHz, 2.9 GB RAM under Win XP Professional (SP3)
In dimension 3 times are: 0.265 for cmulNUM and 0.078 for cmulRS which is 3.39744 times faster running on Intel (R) Core (TM) 2 Duo CPU 2.19 GHz, 2.9 GB RAM under Win XP Professional (SP3)
In dimension 4 times are: 3.157 for cmulNUM and 1.062 for cmulRS which is 2.97269 times faster running on Intel (R) Core (TM) 2 Duo CPU 2.19 GHz, 2.9 GB RAM under Win XP Professional (SP3)
In dimension 5 times are: 51.36 for cmulNUM and 13.125 for cmulRS which is 3.91...
Thus, starting with $\dim V = 5$, $\text{cmulRS}$ is at least 9-10 times faster than $\text{cmulNUM}$ for a symbolic $B$. For a numeric $B$, time differences are not that pronounced. This is because Maple evaluates and simplifies constantly intermediate expressions for a numeric $B$.

```maple
N := 5:
TNUMRSN := table():
for n from 2 to N do
    cbas := cbasis(n):
    B := linalg:-diag(1$n):
    start := time():
    seq(seq(cmulNUM(mm, nn, B), mm = cbas), nn = cbas):
    tNUM := time() - start:
    start := time():
    seq(seq(cmulRS(mm, nn, B), mm = cbas), nn = cbas):
    tRS := time() - start:
    TNUMRSN[n] := [tNUM, tRS]:
    printf("In dimension %a times are: %g for cmulNUM and %g
          for cmulRS which is %g times faster on %s running under %s
          \n", n, TNUMRSN[n][1], TNUMRSN[n][2], TNUMRSN[n][1]/TNUMRSN[n][2], 
          processor_type, _operating_system);
end do:
```

In dimension 2 times are: 0 for cmulNUM and 0 for cmulRS which is NaN times faster on Intel (R) Core (TM) 2 Duo CPU 2.19 GHz, 2.9 GB RAM running under Win XP Professional (SP3)
In dimension 3 times are: 0.125 for cmulNUM and 0.031 for cmulRS which is 4.032 26 times faster on Intel (R) Core (TM) 2 Duo CPU 2.19 GHz, 2.9 GB RAM running under Win XP Professional (SP3)
In dimension 4 times are: 0.25 for cmulNUM and 0.14 for cmulRS which is 1.78571 times faster on Intel (R) Core (TM) 2 Duo CPU 2.19 GHz, 2.9 GB RAM running under Win XP Professional (SP3)
In dimension 5 times are: 2.063 for cmulNUM and 1.391 for cmulRS which is 1.483 11 times faster on Intel (R) Core (TM) 2 Duo CPU 2.19 GHz, 2.9 GB RAM running under Win XP Professional (SP3)

```maple
> save TNUMRSN, "TNUMRSN.m";
```

Changing the procedure that is used to compute a product of two Grassmann monomials can be done as follows:

```maple
> _default_Clifford_product; #current value of the global variable
useproduct(cmulNUM); #selecting 'cmulNUM'
```
Clifford product; #current value of the global variable
cmulRS

Warning, cmul will use cmulNUM; for help see pages ?cmul, ?Clifford:-intro, or ?cmulNUM

cmulNUM

> useproduct(cmul_user_defined);

Warning, no computations with cmul can be performed yet since cmul_user_defined has not been defined as procedure. Select cmulRS, cmulNUM, or a new procedure as argument to useproduct.

For example, we could define the following new Clifford product (for more help, see Clifford:-cmul_user_defined):

> cmul_user_defined:=proc(p1::{cliscalar,clibasmon,climon,clipolynom},
p2::{cliscalar,clibasmon,climon,clipolynom},
K::{name,symbol,matrix,array}) local a1,a2,p;
a1:=scalarpart(p1)+vectorpart(p1,1):#assign to a1 only the scalar and the 1-vector parts of p1
a2:=scalarpart(p1)+vectorpart(p2,1):#assign to a2 only the scalar and the 1-vector parts of p2
RETURN(LC(a1,a2,K));
end:

As it can be guessed from the above code, the result of cmul_user_defined(p1,p2,K) is just the left contraction of the scalar and 1-vector part of p2 by the scalar and the 1-vector part of p1 where the contraction is taken with respect to some form K.

> useproduct(cmul_user_defined);

Warning, cmul will use cmul_user_defined; for help see pages ?cmul, ?Clifford:-intro, or ?cmul_user_defined

> cmul(e1,e2+e3we4);

0
7. Algebraic operations in Cl(B)

```plaintext
> restart:with(Clifford):with(linalg):
```

In any Clifford algebra, one can perform the following algebraic operations:

- `cmul` - Clifford product
- `wedge` - wedge product
- `cexp` - exponentation of a Clifford number
- `cinv` - finding a symbolic inverse, if it exists, of a Clifford number
- `climinpoly` - finding a minimal polynomial of a Clifford number
- `clisolve` - solve algebraic equations for a Clifford number
- `conjugation` - Clifford conjugation as a composition of grade involution and reversion
- `findbasis` - finding linearly independent monomials or polynomials that span a subspace or an ideal
- `gradeinv` - grade involution (the main automorphism of Cl(B))
- `reversion` - reversion (the main antiautomorphism of Cl(B))
- `rot3d` - rotations in three dimensions using quaternions
- `sexp` - finding an exponential of a Clifford number modulo its minimal polynomial
- `subs_clipolynom` - substituting a Clifford number into a polynomial
- `wexp` - finding an exterior exponential of a Clifford number
- `matKrepr` - find a matrix representation of Cl(Q) over a field K in a minimal ideal S
- `minimalideal` - finding generators for a left or right minimal ideal generated by a primitive idempotent
- `spinorKbasis` - find a spinor basis over a field K in a minimal ideal S of Cl(Q)
- `spinorKrepr` - find a spinor representation of Cl(Q) over a field K in a minimal ideal S

There are additional procedures for Vahlen matrices, computing group-invariant spinor forms in spinor ideals, manipulating with matrices over double rings, computing with quaternions, etc. Additional functionality,
as mentioned earlier is achieved through the supplementary packages.

Here we show only a few examples.

```maple
> B:=diag(1,1,1):
> dat:=clidata();
f:=dat[4];
dat := [complex, 2, simple, \frac{Id}{2} + \frac{e1}{2}, [Id, e2, e3, e2we3], [Id, e2we3], [Id, e2]]
f := \frac{Id}{2} + \frac{e1}{2}
> p:=climinpoly(f); #minimal polynomial of the idempotent f;
subs_clipolynom(f,p); #substituting f into its minimal polynomial
g:=2e3+2*Id-3e1we2we3-3e2we3-3e1; #generating a random polynomial in Cl(3)
subs_clipolynom(g,p); #substituting g into polynomial p
p2:=climinpoly(g); #minimal polynomial of g
subs_clipolynom(g,p2); #substituting g into polynomial p2
cexp(g,6); #finding truncated power series of g up to and including degree 6
sexp(g,6); #finding truncated power series of g up to and including degree 6 modulo a minimal polynomial of g
wexp(g,6); #finding truncated exterior power series of g up to and including degree 6
ginv:=cinv(g); #inverse of g
cmul(ginv,g); #this should be Id
reversion(g); #reversion of g
gradeinv(g); #grade involution of g
conjugation(g); #conjugation of g
```

Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

\[ p := x^2 - x \]

\[ g := 2 e3 + 2 Id - 3 e1we2we3 - 3 e2we3 - 3 e1 \]
\[ 6 e3 - 3 Id - 12 e1we2 - 27 e1 + 9 e2we3 + 9 e1we2we3 \]
\[
p^2 := x^4 - 8x^3 + 34x^2 - 288x + 981
\]

\[
\begin{align*}
& \quad 42653 \text{Id} - 52633 \text{e1} + 829 \text{e3} - 709 \text{e1we2} - 79 \text{e1we2we3} + 559 \text{e2we3} \\
& \quad \text{720} - 60 + 36 - 20 - 10 + 30
\end{align*}
\]

Warning, since B has been (re-)assigned, value of dim_V has been reduced by 'we dge' to 3

\[
\begin{align*}
& \quad 331 \text{Id} - 109 \text{e1} + 218 \text{e3} - 109 \text{e2we3} + 206 \text{e1we2we3} \\
& \quad \text{45} - 5 + 15 - 5 + 5
\end{align*}
\]

\[
\text{ginv} := \frac{8 \text{Id}}{109} - \frac{13 \text{e1}}{109} + \frac{2 \text{e3}}{109} - \frac{20 \text{e1we2}}{327} + \frac{7 \text{e2we3}}{109} + \frac{29 \text{e1we2we3}}{327}
\]

\[
\text{Id}
\]

\[
2 \text{Id} - 3 \text{e1} + 2 \text{e3} + 3 \text{e2we3} + 3 \text{e1we2we3}
\]

\[
2 \text{Id} + 3 \text{e1} - 2 \text{e3} - 3 \text{e2we3} + 3 \text{e1we2we3}
\]

\[
2 \text{Id} + 3 \text{e1} - 2 \text{e3} + 3 \text{e2we3} - 3 \text{e1we2we3}
\]

Note that when the form B is not diagonal, reversion is quite complicated as it includes off-diagonal terms of B:

\[
> \text{B} := \text{'B'};
\]

\[
> \text{reversion(e1we2we3)};
\]

\[
-B_{2,3} \text{e1} + B_{3,2} \text{e1} - B_{3,1} \text{e2} + B_{1,3} \text{e2} + B_{2,1} \text{e3} - B_{1,2} \text{e3} - e1we2we3
\]

\[
> \text{reversion(g)};
\]

\[
2 \text{Id} + 3 B_{2,3} \text{Id} - 3 B_{3,2} \text{Id} + 3 B_{2,3} \text{e1} - 3 \text{e1} - 3 B_{3,2} \text{e1} + 3 B_{3,1} \text{e2} - 3 B_{1,3} \text{e2} + 2 \text{e3} \\
- 3 B_{2,1} \text{e3} + 3 B_{1,2} \text{e3} + 3 \text{e2we3} + 3 \text{e1we2we3}
\]

For other procedures, see help pages.

Appendix
8. Dotted and undotted Grassmann bases in quantum Clifford algebras

(This work has been developed with Bertfried Fauser, Universitat Konstanz)

> restart:with(Clifford):with(Cliplus):

Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

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

When the bilinear form \( B \) has an antisymmetric part \( F \), it is convinient to split \( B = g + F \), where \( g \) is the symmetric part of \( B \), and to introduce the so called ``dotted Grassmann basis'' (Ablamowicz and Fauser [1], and Ablamowicz and Lounesto [2] and references therein) and the dotted wedge product. See also Helmstetter [3] for mathematical foundations of deformations of exterior algebras).

The old Grassmann basis will be referred to here as ``undotted Grassmann basis''. In CLIFFORD, the wedge product is denoted by 'wedge' and `&w`, while the dotted wedge product is given by 'dwedge' and `&dw`.

According to the Chevalley definition of the Clifford product &c, we have

\[
x \ &c \ u = \text{LC}(x,u,B) + x \ &w \ u
\]

for a 1-vector \( x \) and an arbitrary element \( u \) of \( Cl(B) \). Here, \( \text{LC}(x,u,B) \) denotes the left contraction of \( u \) by \( x \) with respect to the entire bilinear form \( B \). For more information, see LC. However, when \( B = g + F \), then

\[
\text{LC}(x,u,B) = \text{LC}(x,u,g) + \text{LC}(x,u,F)
\]

and

\[
x \ &c \ u = \text{LC}(x,u,B) + x \ &w \ u = \text{LC}(x,u,g) + \text{LC}(x,u,F) + x \ &w \ u = \text{LC}(x,u,g) +
\]
\( x \& dw u \) where

\( x \& dw u = x \& w u + LC(x,u,F) \). 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 being discussed next.

**Example 1: Simple examples**

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 among the undotted and dotted bases. When no index is used, the default is \( F \):

```plaintext
> x:=x1*e1+x2*e2+x3*e3; u:=e4we5;
> LC(x,u,F);
> dwedge[F] (x,u);
> dwedge[K] (e1+2*e1we3, e4+3*e1we2);
> &dw(e1+2*e1we3, e4+3*e1we2); #default index in `&dw` is F

\[
x := x_1 e_1 + x_2 e_2 + x_3 e_3
\]

\[
u := e_4 e_5
\]

\[
x_1 F_{1,4} e_5 - x_1 F_{1,5} e_4 + x_2 F_{2,4} e_5 - x_2 F_{2,5} e_4 + x_3 F_{3,4} e_5 - x_3 F_{3,5} e_4
\]

\[
x_1 e_1 w e 4 e_5 + (x_1 F_{1,4} + x_2 F_{2,4} + x_3 F_{3,4}) e_5 - (x_1 F_{1,5} + x_2 F_{2,5} + x_3 F_{3,5}) e_4
\]

\[
+ x_2 e_2 w e_4 e_5 + x_3 e_3 w e_4 e_5
\]

\[
e_1 w e 4 - (-K_{1,4} + 6 K_{1,3} K_{1,2}) \text{Id} + 2 e_1 w e_3 w e_4 - (3 K_{1,2} - 2 K_{3,4}) e_1 - 2 K_{1,4} e_3
\]

\[- 6 K_{1,2} e_1 w e_3 - 6 K_{1,3} e_1 w e_2
\]

\[-(3 F_{1,2} - 2 F_{3,4}) e_1 - 2 F_{1,4} e_3 + e_1 w e 4 - (-F_{1,4} + 6 F_{1,3} F_{1,2}) \text{Id} + 2 e_1 w e_3 w e_4
\]

\[- 6 F_{1,2} e_1 w e_3 - 6 F_{1,3} e_1 w e_2
\]

**Example 2: Conversion from the undotted wedge basis to the dotted wedge basis**

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 identity:

\[
\text{convert(e1we2w...wen,wedge_to_dwedge,F) = dwedge[F](e1,e2,...,en)}
\]

which can be shown as follows in dim_V =2, 3:

> F:=array(1..9,1..9,antisymmetric):

> ##when dim_V = 2:
> simplify(dwedge[F](e1,e2)=convert(wedge(e1,e2),wedge_to_dwedge,F));

\[e1we2 + F_{1,2} \text{Id} = e1we2 + F_{1,2} \text{Id}\]

> ##when dim_V = 3:
> simplify(dwedge[F](e1,e2,e3)=convert(wedge(e1,e2,e3),wedge_to_dwedge,F));

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

> ##when dim_V = 4:
> simplify(dwedge[F](e1,e2,e3,e4)=convert(wedge(e1,e2,e3,e4),wedge_to_dwedge,F));

\[F_{2,3} e1we4 + F_{2,3} F_{1,4} \text{Id} - F_{1,3} F_{2,4} \text{Id} + F_{1,2} F_{3,4} \text{Id} - F_{2,4} e1we3 + F_{3,4} e1we2 + e1we2we3we4 + F_{1,2} e2we4 + F_{1,4} e2we3 - F_{1,3} F_{2,4} \text{Id} + F_{1,2} F_{3,4} \text{Id} - F_{2,4} e1we3 + F_{3,4} e1we2 + e1we2we3we4 + F_{1,4} e2we3 - F_{1,3} e2we4 + F_{1,2} e3we4\]

Example 3: Dotted and undotted wedge bases - conversions both ways

First we expand the basis of the original wedge into the dotted wedge and back. For this purpose we choose dim_V=4 and consider Cl(C,B) where the antisymmetric part of B is denoted by F (and its negative by FT), while the symmetric part of B is denoted by g.

> dim_V:=4:
> F:=array(1..dim_V,1..dim_V,antisymmetric):
\( g := \text{array}(1..\text{dim}_V, 1..\text{dim}_V, \text{symmetric}) : \)
\( B := \text{evalm}(g + F) : \)
\( FT := \text{evalm}(-F) : \)
\( F, FT = \text{evalm}(F), \text{evalm}(FT) ; \)
\( g, B = \text{evalm}(g), \text{evalm}(B) ; \)
\( w_{\text{bas}} := \text{cbasis}(\text{dim}_V) ; \) ## the wedge basis

\[
F, FT = \\
\begin{bmatrix}
0 & F_{1,2} & F_{1,3} & F_{1,4} \\
-F_{1,2} & 0 & F_{2,3} & F_{2,4} \\
-F_{1,3} & -F_{2,3} & 0 & F_{3,4} \\
-F_{1,4} & -F_{2,4} & -F_{3,4} & 0
\end{bmatrix}
\]
\[
g, B = \\
\begin{bmatrix}
g_{1,1} & g_{1,2} & g_{1,3} & g_{1,4} \\
g_{1,2} & g_{2,2} & g_{2,3} & g_{2,4} \\
g_{1,3} & g_{2,3} & g_{3,3} & g_{3,4} \\
g_{1,4} & g_{2,4} & g_{3,4} & g_{4,4}
\end{bmatrix}
\begin{bmatrix}
g_{1,1} & g_{1,2} + F_{1,2} & g_{1,3} + F_{1,3} & g_{1,4} + F_{1,4} \\
g_{1,2} - F_{1,2} & g_{2,2} & g_{2,3} + F_{2,3} & g_{2,4} + F_{2,4} \\
g_{1,3} - F_{1,3} & g_{2,3} - F_{2,3} & g_{3,3} & g_{3,4} + F_{3,4} \\
g_{1,4} - F_{1,4} & g_{2,4} - F_{2,4} & g_{3,4} - F_{3,4} & g_{4,4}
\end{bmatrix}
\]

\( w_{\text{bas}} := \{ \text{Id} , e_1, e_2, e_3, e_4, e_1\text{we}2, e_1\text{we}3, e_1\text{we}4, e_2\text{we}4, e_3\text{we}4, e_1\text{we}2\text{we}3, e_1\text{we}2\text{we}4, e_1\text{we}3\text{we}4, e_2\text{we}3\text{we}4, e_1\text{we}2\text{we}3\text{we}4 \} \)

Now we map the convert function onto this basis to get the dotted-wedge basis (and back to test that this device works properly)

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

\( d_{\text{bas}} := \{ \text{Id} , e_1, e_2, e_3, e_4, e_1\text{we}2 + F_{1,2} \text{Id}, e_1\text{we}3 + F_{1,3} \text{Id}, F_{1,4} \text{Id} + e_1\text{we}4, F_{2,3} \text{Id} + e_2\text{we}3, F_{2,4} \text{Id} + e_2\text{we}4, F_{3,4} \text{Id} + e_3\text{we}4, F_{2,3} e_1 - F_{1,3} e_2 + F_{1,2} e_3 + e_1\text{we}2\text{we}3, F_{2,4} e_1 - F_{1,4} e_2 + F_{1,2} e_4 + e_1\text{we}2\text{we}4, F_{3,4} e_1 - F_{1,4} e_3 + F_{1,3} e_4 + e_1\text{we}3\text{we}4, F_{3,4} e_2 - F_{2,4} e_3 + F_{2,3} e_4 + e_2\text{we}3\text{we}4, e_1\text{we}2\text{we}3\text{we}4 + F_{2,3} F_{1,4} \text{Id} - F_{1,3} F_{2,4} \text{Id} + F_{1,2} F_{3,4} e_1 + F_{1,2} e_3 - e_2 e_4 - F_{2,4} e_1\text{we}3 + F_{1,4} e_2\text{we}3 + F_{2,3} e_1\text{we}4 - F_{1,3} e_2\text{we}4 \} \)

\( \text{test}_w_{\text{bas}} := \{ \text{Id} , e_1, e_2, e_3, e_4, e_1\text{we}2, e_1\text{we}3, e_1\text{we}4, e_2\text{we}4, e_3\text{we}4, e_1\text{we}2\text{we}3, e_1\text{we}2\text{we}4, e_1\text{we}3\text{we}4, e_2\text{we}3\text{we}4, e_1\text{we}2\text{we}3\text{we}4 \} \)

Note that only the scalar Id and the one vector basis elements \( e_i \) are unaltered and that the other basis elements of higher grade pick up additional terms of lower grade (which preserves the filtration).

It is possible to define aliases for the dotted wedge basis "monomials"
similar to the Grassmann basis monomials used by 'CLIFFORD'. For example, we could denote the element $e_1w_2 + F[1,2]*Id$ by $e_1de_2$ or $e_1We_2$, and similarly for other elements:

```maple
> alias(e1We2=e1we2 + F[1,2]*Id, e1We3=e1we3 + F[1,3]*Id, e1We4=e1we4 + F[1,4]*Id,
    e2We3=e2we3 + F[2,3]*Id, e2We4=e2we4 + F[2,4]*Id,
    e1We2We3=e1we2we3+F[2,3]*e1-F[1,3]*e2+F[1,2]*e3,
    e1We2We4=e1we2we4+F[2,4]*e1-F[1,4]*e2+F[1,2]*e4,
    e1We3We4=e1we3we4+F[3,4]*e1-F[1,4]*e3+F[1,3]*e4,
    e2We3We4=e2we3we4+F[3,4]*e2-F[2,4]*e3+F[2,3]*e4,
    e1We2We3We4=e1we2we3we4+F[1,2]*e3we4+F[1,2]*F[3,4]*Id-F[2,4]*e1
    we3-F[1,3]*F[2,4]*Id+F[1,4]*e2we3+F[2,3]*F[1,4]*Id+F[3,4]*e1we2
    +F[2,3]*e1we4-F[1,3]*e2we4);
```

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

```maple
> d_bas;
[Id, e1, e2, e3, e1We2, e1We3, e1We4, e2We3, e2We4, e3We4, e1We2We3, e1We2We4,
  e1We3We4, e1We2We3We4]
```

While command 'cbasis' displays basis elements in the Grassmann basis by default, it is not difficult to write a new procedure that would display the dotted basis instead. For example, procedure 'dottedbasis' returns such basis. Since we have defined aliases above, output from 'dottedcbasis' will be automatically converted to aliases:

```maple
> dottedcbasis[F](3);
[Id, e1, e2, e3, e1We2, e1We3, e1We4, e2We3, e2We4, e3We4, e1We2We3, e1We2We4,
  e1We3We4, e2We3We4, e1We2We3We4]
```

```maple
> dottedcbasis[F](3, 'even');
[Id, e1We2, e1We3, e2We3]
```

```maple
> dottedcbasis[F](3, 2);
[e1We2, e1We3, e2We3]
```

With the procedure 'findbasis' which returns linearly independent elements
from a list, we can verify that the above lists contain linearly independent elements:

```plaintext
> findbasis(dottedcbasis[F](3));
[Id, e1, e2, e3, e1We2, e1We3, e2We3, e1We2We3]
> findbasis(dottedcbasis[F](3,'even'));
[Id, e1We2, e1We3, e2We3]
> findbasis(dottedcbasis[F](3,2));
[e1We2, e1We3, e2We3]
```

Example 4: Commutative Diagram 1: Contraction in dotted and undotted bases

The contraction w.r.t. any bilinear form works on both sets in the same manner which can be seen if we re-convert the dotted-wedge basis after the computation into the wedge basis. In a reasonable setting, the antisymmetric bilinear form F would be the antisymmetric part of B. To read more about the left contraction LC in Cl(B), go to the help page `Clifford:-LC` or see [1, 2, 4]. To illustrate this fact, we first compute left contraction by e1 of every element in the standard Grassmann wedge basis w_bas with respect to the entire form B:

```plaintext
> 'w_bas'=w_bas; #standard Grassmann wedge basis in Cl(B)

w_bas = [Id, e1, e2, e3, e1we2, e1we3, e1we4, e2we3, e2we4, e3we4, e1we2we3, e1we2we4, e1we3we4, e2we3we4, e1we2we3we4]
> w_wout:=map2(LC,e1,w_bas,B);#left contraction LC in Cl(B)

w_r.t. B in wedge basis

w_wout := [0, g_{1,1} Id, (g_{1,2} + F_{1,2}) Id, (g_{1,3} + F_{1,3}) Id, (g_{1,4} + F_{1,4}) Id,
           g_{1,1} e2 - (g_{1,2} + F_{1,2}) e1, g_{1,1} e3 - (g_{1,3} + F_{1,3}) e1, g_{1,1} e4 - (g_{1,4} + F_{1,4}) e1,
           (g_{1,2} + F_{1,2}) e3 - (g_{1,3} + F_{1,3}) e2, (g_{1,2} + F_{1,2}) e4 - (g_{1,4} + F_{1,4}) e2,
           (g_{1,3} + F_{1,3}) e4 - (g_{1,4} + F_{1,4}) e3, g_{1,1} e2we3 - (g_{1,2} + F_{1,2}) e1we3 + (g_{1,3} + F_{1,3}) e1we2,
           g_{1,1} e2we4 - (g_{1,2} + F_{1,2}) e1we4 + (g_{1,4} + F_{1,4}) e1we2,
           g_{1,1} e3we4 - (g_{1,3} + F_{1,3}) e1we4 + (g_{1,4} + F_{1,4}) e1we3,
           (g_{1,2} + F_{1,2}) e3we4 - (g_{1,3} + F_{1,3}) e2we4 + (g_{1,4} + F_{1,4}) e2we3,
```

43
Next, we compute left contraction by \( e_1 \) of every element in the dotted wedge basis \( \text{d}_\text{bas} \) with respect to the entire form \( B \). Recall from the above that conversion from the wedge basis to the dotted wedge basis used the antisymmetric part \( F \) of \( B \):

\[
g_{1,1} e_2 e_3 e_4 - (g_{1,2} + F_{1,2}) e_1 e_3 e_4 + (g_{1,3} + F_{1,3}) e_1 e_2 e_4 - (g_{1,4} + F_{1,4}) e_1 e_2 e_3
\]

We compute left contraction by \( e_1 \) of every element in the dotted wedge basis used the antisymmetric part \( F \) of \( B \):

\[
d_{\text{bas}} := \text{map}(\text{convert}, \text{d}_{\text{out}}, \text{dwedge}_\text{to}_\text{wedge}, -F); \quad \# \text{converting back to undotted basis}
\]

Notice that in the above coefficients of \( g \), the symmetric part of \( B \), are mixed with the coefficients of the antisymmetric part \( F \) of \( B \). To remove the \( F \) coefficients, we need to convert back the above result to the un-dotted standard Grassmann basis using the negative \( -F \), that is, the negative of the antisymmetric part of \( B \) in the conversion process:

\[
w_{\text{dout}} := \text{map}(\text{convert}, w_{\text{dout}1}, \text{dwedge}_\text{to}_\text{wedge}, -F); \quad \# \text{converting back to undotted basis}
\]
This computation shows clearly the isomorphism of both pictures. To show that the new structure is nevertheless valuable for other reasons, we proceed with Clifford products.

Example 5: Commutative Diagram 2: Clifford product in dotted and undotted bases

We can build a Clifford algebra \( \text{Cl}(B) \) over each basis set, that is, \( \text{w}_\text{wout} \) or \( \text{w}_\text{dout} \), but with different bilinear forms: when \( B=g \) and when \( B=g+F \) (following notation from [1, 2, 4]), where \( g \) is the symmetric part of \( B \) and \( F \) is the antisymmetric part of \( B \):

\[
B, g, F = \begin{bmatrix}
ge_{1,1} & g_{1,2} + F_{1,2} & g_{1,3} + F_{1,3} & g_{1,4} + F_{1,4} \\
g_{1,2} - F_{1,2} & g_{2,2} & g_{2,3} + F_{2,3} & g_{2,4} + F_{2,4} \\
g_{1,3} - F_{1,3} & g_{2,3} - F_{2,3} & g_{3,3} & g_{3,4} + F_{3,4} \\
g_{1,4} - F_{1,4} & g_{2,4} - F_{2,4} & g_{3,4} - F_{3,4} & g_{4,4}
\end{bmatrix}
\]

Let us compute a few Clifford products using the facility of `Clifford:-cmul` to
take a bilinear form (here in matrix form) as index. We will show an example with the following two elements:

\[
\begin{align*}
  w_{p1} &= e_1 w e_2; \\
  w_{p2} &= a e_3 + b e_2 w e_3;
\end{align*}
\]

We can then define Clifford product 'cmulg' with respect to the symmetric part \(g\), and another Clifford product 'cmulB' with respect to the entire form \(B\):

\[
\begin{align*}
  \text{cmulg} &:= \text{proc}() \ \text{RETURN}(\text{cmul}[g](\text{args})) \ \text{end}; \\
  \text{cmulB} &:= \text{proc}() \ \text{RETURN}(\text{cmul}[B](\text{args})) \ \text{end};
\end{align*}
\]

Thus, we are ready to perform computations around our second commutative diagram.

First, we compute Clifford product \(\text{cmulg}\) in \(\text{Cl}(g)\), that is, with respect to the symmetric part \(g\) of the bilinear form \(B\), of the two above defined elements \(w_{p1}\) and \(w_{p2}\) expressed in undotted Grassmann basis.

\[
\begin{align*}
  w_{out1} &:= \text{cmulg}(w_{p1}, w_{p2}); \quad \# \text{ Clifford product w.r.t. } g \text{ in } \text{Cl}(g) \text{ in wedge basis}
\end{align*}
\]

Now, we convert each element \(p1\) and \(p2\) to the dotted wedge basis:

\[
\begin{align*}
  d_{p1} &:= \text{convert}(w_{p1}, \text{wedge\_to\_dwedge}, F); \\
  d_{p2} &:= \text{convert}(w_{p2}, \text{wedge\_to\_dwedge}, F); \quad \# \text{incomplete conversion to } e_1 w e_2, \text{ etc. basis}
\end{align*}
\]

We now compute the Clifford product of \(d_{p1}\) and \(d_{p2}\) in \(\text{Cl}(B)\) in the dotted wedge basis:

\[
\begin{align*}
  d_{out1} &:= \text{cmulB}(d_{p1}, d_{p2}); \quad \# \text{ Clifford product w.r.t. } B=g+F \text{ in } \text{Cl}(B) \text{ in dwedge basis}
\end{align*}
\]
We now convert the above result back to the un-dotted wedge basis:

```maple
w_out2 := convert(d_out1, dwedge_to_wedge, -F);  ## convert result
dwedge-> wedge
```

Finally, we show that this result is the same as before when we computed Clifford product of \( p_1 \) and \( p_2 \) in \( \text{Cl}(g) \):

```maple
simplify(w_out1-w_out2);        ## show equality !
```

This shows (one can prove this) that the Clifford algebra \( \text{Cl}(g) \) of the symmetric part \( g \) of \( B \) using the undotted exterior basis is isomorphic, as an associative algebra, to the Clifford algebra \( \text{Cl}(B) \) 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 form \( F \) that is used to connect the two basis sets (cf. [1, 2, 4]).

References


9. More on associativity of the dotted wedge

(This work has been developed with Bertfried Fauser, Universitat Konstanz)

Example 6: Operation 'dwedge' is associative with Id as a unit

\[ \text{dwedge}[F](\text{dwedge}[F](e_1,e_2),e_3); \]
\[ \text{dwedge}[F](e_1,\text{dwedge}[F](e_2,e_3)); \]
\[
\begin{align*}
e_1 & We_2 We_3 \\
e_1 & We_2 We_3
\end{align*}
\]

\[ \text{dwedge}[F](\text{dwedge}[F](e_1,e_2 we_3),e_4); \]
\[ \text{dwedge}[F](e_1,\text{dwedge}[F](e_2 we_3,e_4)); \]
\[ F, 1 2 \\
F, 1 3 \\
F, 2 4 \\
F, 1 2 \\
F, 3 4 \\
F, 1 4 \\
e_2 we_3 \\
e_1 we_2 \]
\[ F, 1 3 \\
F, 2 4 \\
F, 1 2 \\
F, 3 4 \\
F, 1 4 \\
e_2 we_4 \\
e_1 we_3 \]
\[
\begin{align*}
F_{1,2} e_3 we_4 + (-F_{1,3} F_{2,4} + F_{1,2} F_{3,4}) Id + F_{1,4} e_2 we_3 + F_{3,4} e_1 we_2 + e_1 we_2 we_3 we_4 \\
- F_{1,3} e_2 we_4 - F_{2,4} e_1 we_3 \\
F_{1,2} e_3 we_4 + (-F_{1,3} F_{2,4} + F_{1,2} F_{3,4}) Id + F_{1,4} e_2 we_3 + F_{3,4} e_1 we_2 + e_1 we_2 we_3 we_4 \\
- F_{1,3} e_2 we_4 - F_{2,4} e_1 we_3 \\
0
\end{align*}
\]

Finally, for some arbitrary random Clifford polynomials expressed in Grassmann undotted basis:

\[ u := \text{Id} + e_3 we_4 - e_1 we_4 - e_1 we_3 - e_2 we_3 - e_1; \]
\[
u := \text{Id} + e_3 we_4 - e_1 we_4 - e_1 we_3 - e_2 we_3 - e_1
\]
\[ v := 5 \text{Id} + 5 e_2 we_4 + 5 e_2 we_3 we_4; \]
\[
v := 5 \text{Id} + 5 e_2 we_4 + 5 e_2 we_3 we_4
\]
\[ z := -6 \text{Id} + 2e4 + 2e2we4; \]
\[ dwedge[F]\text{ (Id, u)} = u; \quad \text{unity} \]
\[ dwedge[F]\text{ (u, Id)} = u; \]
\[ \text{Id} + e3we4 - e1we4 - e1we3 - e2we3 - e1 = \text{Id} + e3we4 - e1we4 - e1we3 - e2we3 - e1 \]
\[ \text{Id} + e3we4 - e1we4 - e1we3 - e2we3 - e1 = \text{Id} + e3we4 - e1we4 - e1we3 - e2we3 - e1 \]
\[ dwedge[F]\text{ (dwedge[F] (u, v), z)}: \#\text{associativity} \]
\[ dwedge[F]\text{ (u, dwedge[F] (v, z))}: \%\%-%; \]

We also have the following **Commutative Diagram 3: Wedge in undotted and dwedge in dotted bases:**

\[
\text{wedge(u, v)} = \text{convert(dwedge( convert(u, wedge_to_dwedge, F), convert(v, wedge_to_dwedge, F)), dwedge_to_wedge, -F)}
\]

which we show as follows:

\[
\text{uu} := \text{convert (u, wedge_to_dwedge, F)}; \quad \#\text{u converted to dotted basis} \\
\text{vv} := \text{convert (v, wedge_to_dwedge, F)}; \quad \#\text{v converted to dotted basis} \\
\text{uu} := -e1we4 + F_{3,4} \text{Id} - F_{1,4} \text{Id} + \text{Id} - F_{1,3} \text{Id} - F_{2,3} \text{Id} - e1we3 - e2we3 - e1 + e3we4 \\
\text{vv} := 5e2we4 + 5 \text{Id} + 5F_{2,4} \text{Id} + 5e2we3we4 + 5F_{3,4} e2 - 5F_{2,4} e3 + 5F_{2,3} e4 \\
\text{out1} := \text{dwedge[F] (uu, vv)}; \quad \#\text{dwedge computed w.r.t. F} \\
\text{out2} := \text{convert (out1, dwedge_to_wedge, -F)}; \quad \#\text{previous result converted back to undotted basis} \\
\text{out2} := 5 \text{Id} + 5e3we4 - 5e1we4 - 5e1we3 - 5e2we3 - 5e1 + 5e2we4 - 5e1we2we4 + 5e2we3we4 \\
\text{out3} := \text{wedge(u, v)}; \quad \#\text{direct computation of the wedge product in undotted basis} \\
\text{out3} := 5 \text{Id} + 5e3we4 - 5e1we4 - 5e1we3 - 5e2we3 - 5e1 + 5e2we4 - 5e1we2we4 + 5e2we3we4 \\
\text{out2-out3}; \quad \#\text{the same results!}
10. Reversion in dotted and undotted bases

(This work has been developed with Bertfried Fauser, Universitat Konstanz)

Example 7: Commutative Diagram 3: Reversion in dotted and undotted bases

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 e.g. the Clifford reversion. Only if the bilinear
form is symmetric, we find that the reversion is grade preserving,
otherwise it reflects only the filtration (i.e., is in general a sum 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)

\[-2 F_{1,2} \text{Id} - e1we2\]
\[-e1we2\]

Observe in the above that only when \(B[1,2]=B[2,1]\), the result is \(-e1we2\)
known from the theory of classical Clifford algebras. Likewise,

> cbas:=cbasis(3);

\[\text{cbas} := [\text{Id}, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3]\]

> map(reversion,cbas,B);

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

If instead of \(B\) we use a symmetric matrix 'g' defined above, we obtain
instead

> map(reversion,cbas,g);

\[[\text{Id}, e1, e2, e3, -e1we2, -e1we3, -e2we3, -e1we2we3]\]

Convert now \(e1we2\) to the dotted basis and call it \(e1We2\):
Apply reversion to \( \text{e1We2} \) with respect to \( F \) to get the reversed element in the dotted basis:

\[
\text{reversed}_\text{e1We2} := \text{reversion} (\text{e1We2}, F);
\]

\[
\text{reversed}_\text{e1We2} := -F_{1,2} \text{Id} - \text{e1we2}
\]

Observe, that the above element is equal to the negative of \( \text{e1We2} \) just like reversing \( \text{e1we2} \) with respect to the symmetric part \( g \) of \( B \):

\[
\text{reversed}_\text{e1We2} + \text{e1We2};
\]

\[
0
\]

Finally, convert \( \text{reversed}_\text{e1We2} \) to the un-dotted standard Grassmann basis to get \( -\text{e1we2} \):

\[
\text{convert} (\text{reversed}_\text{e1We2}, \text{dwedge_to_wedge}, -F);
\]

\[
-\text{e1we2}
\]

The above, of course, can be obtained by applying reversion to \( \text{e1we2} \) with respect to the symmetric part of \( B \):

\[
\text{reversion} (\text{e1we2}, g); \quad \# \text{reversion with respect to the symmetric part \( g \) of \( B \)}
\]

\[
-\text{e1we2}
\]

11. Spinor representations of \( \text{Cl}(Q) \) in a minimal left ideal \( S = \text{Cl}(Q) f \)

To shorten output, we will use aliases. Since below we will not exceed dimension 6, we can define aliases first:

\[
\text{restart:with(Clifford)};
\]

Thus, for example,
\[ e_{12} + B_{1,2} \text{Id} \]

where \( e_{12} = e_1 e_2 \). When the bilinear form \( B \) is diagonal, then \( e_{12} = e_1 e_2 = e_1 w e_2 \), etc.

**Example 1:** Clifford algebra \( \text{Cl}(2,0) \) is isomorphic to \( \mathbb{R}(2) \)

\[
\text{dim} := 2; \text{B} := \text{linear}[\text{diag}](1,1); \quad \# \text{define the bilinear form } B \text{ for } \text{Cl}(2,0)
\]

\[
\text{clibasis} := \text{cbasis}(\text{dim}); \quad \# \text{compute a Clifford basis for } \text{Cl}(B)
\]

\[
\text{data} := \text{clidata}(B); \quad \# \text{retrieve and display data about } \text{Cl}(B)
\]

\[
data := \begin{bmatrix}
\text{real}, 2, \text{simple}, \\
\frac{e_1}{2}, \frac{e_2}{2}, [\text{Id}, e_2], [\text{Id}], [\text{Id}, e_2]
\end{bmatrix}
\]

\[
f := \text{data}[4]; \quad \# \text{assign pre-stored idempotent to } f \text{ or use your own here}
\]

\[
\text{sbasis} := \text{minimalideal}(\text{clibasis}, f, '\text{left}'); \quad \# \text{compute a real basis in } \text{Cl}(B)f
\]

\[
sbasis := \begin{bmatrix}
\frac{\text{Id}}{2} + \frac{e_1}{2}, \frac{e_2}{2}, -\frac{e_{12}}{2}
\end{bmatrix}, [\text{Id}, e_2], \text{left}
\]

\[
\text{Kbasis} := \text{Kfield}(\text{sbasis}, f); \quad \# \text{compute a basis for the field } K
\]

\[
\text{SBgens} := \text{sbasis}[2]; \quad \# \text{generators for a real basis in } S
\]

\[
\text{FBgens} := \text{Kbasis}[2]; \quad \# \text{generator for } K \text{ is only one since } K = \mathbb{R}
\]

\[
\text{FBgens} := \{ \text{Id} \}
\]

\[
\text{Kbasis} := \text{spinorKbasis}(\text{SBgens}, f, \text{FBgens}, '\text{left}'); \quad \# K\text{-basis for } S
\]

\[
K\_basis := \begin{bmatrix}
\frac{\text{Id}}{2} + \frac{e_1}{2}, \frac{e_2}{2}, -\frac{e_{12}}{2}
\end{bmatrix}, [\text{Id}, e_2], \text{left}
\]

Here are matrices representing basis monomials of \( \text{Cl}(B) \):

\[
\text{M0, M1, M2, M3} := \text{op} (\text{map(}\text{spinorKrepr, clibasis, K\_basis}[1], \text{FBgens}, '\text{left}'));
\]
Since the spinor representation is a homomorphism from Cl(2,0) to R(2), matrix $M_{12}$ 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' (see \texttt{rmulm} for more help) which can also be entered in its infix form as:

\begin{verbatim}
> M12 := M1 &cm M2;
\end{verbatim}

Notice that these matrices have the same algebraic properties as the basis elements they represent. For example:

\begin{verbatim}
> e1 &c e2 + e2 &c e1, evalm(M1 &cm M2 + M2 &cm M1);
\end{verbatim}

Let's find a matrix representing an arbitrary Clifford polynomial $p$ in Cl(2,0):

\begin{verbatim}
> p := a0 + a1*e1 + a2*e2 + a12*e12;
\end{verbatim}

\begin{verbatim}
> spinorKrepr(p,K_basis[1],FBgens,'left'); # matrix of $p$ in S
\end{verbatim}

Of course, this matrix is the same as the following one:

\begin{verbatim}
> evalm(a0*M0 + a1*M1 + a1*M2 + a12*M12);
\end{verbatim}

The simplest way to compute that matrix is to use 'matKrepr' (see \texttt{matKrepr} for more help):

\begin{verbatim}
> matKrepr(p);
\end{verbatim}
Example 2: Clifford algebra $\text{Cl}(3,0)$ is isomorphic to $\mathbb{C}(2)$: the Pauli matrices with complex entries.

\begin{verbatim}
> dim:=3; B:=linalg[diag](1,1,1): #define the bilinear form B for Cl(3,0)
clibasis:=cbasis(dim): #compute a Clifford basis for Cl(B)
> data:=clidata(B); #retrieve and display data about Cl(B)
    data := [complex, 2, simple, \( \frac{\text{Id} + e_1}{2} \), \([\text{Id}, e_2, e_3], [\text{Id}, e_23], [\text{Id}, e_2]\)]
> 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(B)
> Kbasis:=Kfield(sbasis,f); #compute a basis for the field K
    Kbasis := [\[\frac{\text{Id} + e_1}{2}, \frac{e_{23}}{2} + \frac{e_{123}}{2}\], [\text{Id}, e_{23}]]
> SBgens:=sbasis[2]: #generators for a real basis in S
> FBgens:=Kbasis[2]; #generators for K are two since K=C
    FBgens := [\text{Id}, e_{23}]
> K_basis:=spinorKbasis(SBgens,f,FBgens,'left');
    K_basis := [\[\frac{\text{Id} + e_1}{2}, \frac{e_{2}}{2} - \frac{e_{12}}{2}\], [\text{Id}, e_{2}], .left]
\end{verbatim}

Here are the matrices representing some basis monomials of Cl(B). Matrices $\sigma[1]$, $\sigma[2]$, and $\sigma[3]$ are the well-known Pauli matrices with entries in the field $K$:

\begin{verbatim}
> 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 & -e_{23} \\ e_{23} & 0 \end{bmatrix}
\end{verbatim}

Let's find matrices representing the two basis elements in S:

\begin{verbatim}
> f1, f2:=K_basis[1][1], K_basis[1][2];
\end{verbatim}
\( f_1, f_2 := \frac{\text{Id}}{2} + \frac{e_1}{2} + \frac{e_2}{2} - \frac{e_{12}}{2} \)

\[
\begin{pmatrix}
1 & 0 \\
0 & 0
\end{pmatrix}
\]

Thus, a spinor \( s \) from \( S = \text{Cl}(3,0) \) is a complex vector written in terms of the basis \([f_1, f_2]\) and its one-column complex matrix with entries in \( K = \langle \text{Id}, e_{23} \rangle \) is:

\[
\psi_1, \psi_2 := a \cdot \text{Id} + b \cdot e_{23}, c \cdot \text{Id} + d \cdot e_{23};
\]

\[
\begin{pmatrix}
\psi_1 \\
\psi_2
\end{pmatrix}
\]

\( s := f_1 \& c \psi_1 + f_2 \& c \psi_2 \); #remember that \( S \) is a right \( K \)-vector space

\[
\begin{pmatrix}
\frac{a \cdot \text{Id}}{2} + \frac{a \cdot e_1}{2} + \frac{b \cdot e_{23}}{2} + \frac{b \cdot e_{123}}{2} + \frac{c \cdot e_2}{2} + \frac{d \cdot e_3}{2} - \frac{c \cdot e_{12}}{2} - \frac{d \cdot e_{13}}{2} \\
\frac{b \cdot e_{23}}{2} + \frac{d \cdot e_{123}}{2}
\end{pmatrix}
\]

One column matrix \( sm \) that represents \( s \) has entries in \( K = \langle \text{Id}, e_{23} \rangle \).

**Example 3:** Clifford algebra \( \text{Cl}(1,3) \) isomorphic to \( H(2) \): the gamma matrices with quaternionic entries.

\[
\begin{pmatrix}
quaternionic, 2, \text{simple}, \frac{\text{Id}}{2} + \frac{e_{14}}{2}, [\text{Id}, e_1, e_2, e_3, e_{12}, e_{13}, e_{23}, e_{123}], \\
[\text{Id}, e_2, e_3, e_{23}], [\text{Id}, e_1]
\end{pmatrix}
\]
sbasis:=minimalideal(clibasis,f,'left'): # compute a real basis in Cl(B)f

Kbasis:=Kfield(sbasis,f); # compute a basis for the field K

\[
\begin{bmatrix}
\frac{Id}{2} + \frac{e14}{2}, \\
\frac{e2}{2} - \frac{e124}{2}, \\
\frac{e3}{2} - \frac{e134}{2}, \\
\frac{e23}{2} + \frac{e1234}{2}
\end{bmatrix} \cdot [Id, e2, e3, e23]
\]

SBgens:=sbasis[2]: # generators for a real basis in S

FBgens:=Kbasis[2]; # generators for K are four since K=H

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

Here are the matrices representing some basis monomials of Cl(B). Those that represent the 1-vectors are usually referred to as gamma matrices (since 'gamma' is a protected Maple name, we will use here 'Gamma' instead):

\[
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix},
\begin{bmatrix}
e2 & 0 \\
0 & -e2
\end{bmatrix},
\begin{bmatrix}
e3 & 0 \\
0 & -e3
\end{bmatrix},
\begin{bmatrix}
0 & -1 \\
1 & 0
\end{bmatrix}
\]

Matrix of the unit pseudoscalar is:

\[
\begin{bmatrix}
e23 & 0 \\
0 & -e23
\end{bmatrix}
\]

Since the spinor representation is a homomorphism from Cl(1,3) to H(2), the matrix Gamma[1234] is a product of matrices Gamma[1],..., Gamma[4] with Clifford multiplication applied to their entries. Procedure which handles multiplication of such matrices is called 'rmulm' (see rmulm for more help). For example,

\[
\begin{bmatrix}
e23 & 0 \\
0 & -e23
\end{bmatrix}
\]
Let's find matrices representing the two basis vectors in $S$:

\[ f_1, f_2 := \text{K\_basis}[1][1], \text{K\_basis}[1][2]; \]

\[
f_1, f_2 := \frac{\text{Id}}{2} + \frac{e_{14}}{2}, \frac{e_1}{2} + \frac{e_4}{2}
\]

\[
F_1, F_2 := \text{op}(\text{map}(\text{spinorKrepr}, [f_1, f_2], \text{K\_basis}[1], \text{FBgens}, '\text{left}'));
\]

Thus, a spinor $s$ from $S = \text{Cl}(1,3)$ is a two component vector with quaternionic entries. It can be written in terms of the basis $[f_1, f_2]$ and its quaternionic entries in $K = <\text{Id}, e_2, e_3, e_{23}>_R$ (written on the right) as follows:

\[
\psi_1, \psi_2 := a_0 \text{Id} + a_1 e_2 + a_2 e_3 + a_3 e_{23}, b_0 \text{Id} + b_1 e_2 + b_2 e_3 + b_3 e_{23};
\]

\[
s := f_1 \& c \psi_1 + f_2 \& c \psi_2; \quad \#\text{remember that } S \text{ is a right } K\text{-vector space}
\]

Finally, we can find the one-column matrix $s_m$ representing $s$:

\[
s_m := \text{matKrepr}(s); \quad \#\text{matrix of } s
\]

One column matrix $s_m$ that represents $s$ has entries in $K = <\text{Id}, e_2, e_3, e_{23}>_R$. Let's find a matrix representing an arbitrary Clifford polynomial $p$ in $\text{Cl}(1,3)$:

\[
p := \text{add}(a \mid (i-1)\*\text{clibasis}[i], i=1..\text{nops(\text{clibasis})});
\]

\[
P := \text{map}(\text{clicollect}, \text{spinorKrepr}(p, \text{K\_basis}[1], \text{FBgens}, '\text{left}')); \quad \#\text{matrix of } p \text{ in } S
\]

\[
P :=
\begin{pmatrix}
(a_0 + a_7) \text{Id} + (a_8 + a_{15}) e_{23} - (-a_2 + a_{12}) e_2 - (-a_3 + a_{13}) e_3, \\
(a_1 - a_4) \text{Id} + (a_{11} - a_{14}) e_{23} - (a_9 + a_5) e_2 - (a_{10} + a_6) e_3 \\
((a_1 + a_4) \text{Id} + (a_{11} + a_{14}) e_{23} + (-a_9 + a_5) e_2 - (a_{10} - a_6) e_3, \\
(a_0 - a_7) \text{Id} - (-a_8 + a_{15}) e_{23} - (a_2 + a_{12}) e_2 - (a_3 + a_{13}) e_3
\end{pmatrix}
\]

**Example 4**: Clifford algebra $\text{Cl}(3,1)$ is isomorphic to $\text{R}(4)$: the gamma
matrices are the well-known \textit{Dirac matrices}. 

\begin{verbatim}
> dim:=4: B:=linalg[diag](1,1,1,-1): #define form B for Cl(3,1)
> clibasis:=cbasis(dim): #compute a Clifford basis for Cl(B)
> data:=clidata(B); #retrieve and display data about Cl(B)

data :=
\[
\begin{bmatrix}
\text{real}, 4, \text{simple}, '\text{cmulQ}' \left(\frac{\text{Id}}{2} + \frac{\text{e1}}{2}, \frac{\text{Id}}{2} + \frac{\text{e34}}{2}\right), [\text{Id}, \text{e2}, \text{e3}, \text{e23}], [\text{Id}], [\text{Id}, \text{e2}, \text{e3}, \text{e23}] \\
\end{bmatrix}
\]

> 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(B)f

> Kbasis:=Kfield(sbasis,f); #compute a basis for the field K

\[
Kbasis := \left[\begin{bmatrix}
\frac{\text{Id}}{4} + \frac{\text{e1}}{4} + \frac{\text{e34}}{4}, \frac{\text{e134}}{4}
\end{bmatrix}, [\text{Id}]\right]
\]

> SBgens:=sbasis[2]: #generators for a real basis in S

> FBgens:=Kbasis[2]; #generator for K is only one since K=R

\[
\text{FBgens} := [\text{Id}]
\]

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

\[
K_{basis} := \left[\begin{bmatrix}
\frac{\text{Id}}{4} + \frac{\text{e1}}{4} + \frac{\text{e34}}{4}, \frac{\text{e134}}{4}, \frac{\text{e12}}{4} - \frac{\text{e234}}{4} - \frac{\text{e134}}{4}, \frac{\text{e13}}{4} + \frac{\text{e4}}{4} - \frac{\text{e14}}{4}, \\
\frac{\text{e23}}{4} + \frac{\text{e123}}{4} + \frac{\text{e24}}{4} + \frac{\text{e124}}{4}
\end{bmatrix}, [\text{Id}, \text{e2}, \text{e3}, \text{e23}], \text{left}\right]
\]

Here are the matrices representing some basis monomials of Cl(B). Those that represent the 1-vectors are usually referred to as \textit{Dirac gamma matrices} (since 'gamma' is a protected Maple name, we will use here 'Gamma' instead):

\begin{verbatim}
> Gamma[1], Gamma[2], Gamma[3], Gamma[4]: = op(map(spinorKrepr, [e1,e2, e3,e4], K_basis[1], FBgens, 'left'));
\end{verbatim}
\end{verbatim}
\[ \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4 := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

Matrix of the unit pseudoscalar is:
\[ \Gamma_{1234} := \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \]

Let's find matrices representing the four basis vectors in \( S \):
\[ \Gamma_{123} := \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \]

Thus, a spinor \( s \) from \( S = \text{Cl}(3,1)f \) is a four component vector with real entries. It can be written in terms of the basis \([f_1,f_2,f_3,f_4]\) and its real entries in \( K = \langle \text{Id} \rangle_R \) as follows:
\[ s := f_1 \psi_1 + f_2 \psi_2 + f_3 \psi_3 + f_4 \psi_4 \]

Finally, we can find the one-column matrix \( sm \) representing \( s \). (Note: it is faster to use 'matKrepr' than 'spinorKrepr').
\[ sm := \text{subs}(\text{Id}=1, \text{matKrepr}(s)); \]
Let's find a matrix representing an arbitrary Clifford polynomial \( p \) in \( \text{Cl}(3,1) \):

\[
\begin{bmatrix}
\psi_1 & 0 & 0 & 0 \\
\psi_2 & 0 & 0 & 0 \\
\psi_3 & 0 & 0 & 0 \\
\psi_4 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
sm := \begin{bmatrix}
\psi_1 & 0 & 0 & 0 \\
\psi_2 & 0 & 0 & 0 \\
\psi_3 & 0 & 0 & 0 \\
\psi_4 & 0 & 0 & 0 \\
\end{bmatrix}
\]

References


12. Scalar products beta_plus and beta_minus in spinor ideals

- Procedure

\[\text{-- beta_plus(psi, phi, f, 's')} \text{ - computes a scalar product of two spinors 'psi' and 'phi' in S=Cl(Q)f when one of them is subject to reversion anti-automorphism where S is a spinor ideal in Cl(Q) generated by a primitive idempotent 'f'; argument 's' is optional}\]
-- beta_minus(psi, phi, f, 's') - computes a scalar product of two spinors 'psi' and 'phi' in S=Cl(Q)f when one of them is subject to conjugation anti-automorphism where S is a spinor ideal in Cl(Q) generated by a primitive idempotent 'f'; argument 's' is optional

- The fourth optional argument 's' is of `type/name`: it will be a placeholder for an invertible element $s$ in Cl(Q) such that $\text{cmul}(f, s) = \text{cmul}(s, \text{auto}(f))$ for a simple Clifford algebra Cl(Q) (when the signature of Q is (p,q) and $p - q \not= 1 \mod 4$), or $\text{cmul}(g, s) = \text{cmul}(g, \text{auto}(g))$ for a semisimple Clifford algebra Cl(Q) (when the signature of Q is (p,q) and $p - q = 1 \mod 4$). Here, \texttt{cmul} denotes Clifford product in Cl(Q), $g = f + \text{gradeinv}(f)$, and 'auto' is either 'reversion' or 'conjugation'.

- Invertible element $s$ is needed so that $\text{cmul}(s, \text{auto}(psi), phi)$ belongs to $K, K^\wedge$ or $K + K^\wedge$, depending on the chosen (anti-) automorphism and whether Cl(Q) is simple or semisimple. Here $K$ is a subalgebra of Cl(Q) isomorphic with the reals, complex numbers, or quaternions depending on the signature of Q and the hat in $K^\wedge$ denotes grade involution in Cl(Q). The grade involution is computed with the procedure \texttt{gradeinv}. Upon exiting, 'beta' stores the value of $s$ under the name 's' supplied by the user. These special elements $s$ in Cl(Q) are called 'purespinors'. It must be entered as 's', that is, as an unevaluated name. Otherwise, an error message will appear if $s$ has already been assigned some other basis monomial as a purespinor since basis monomials are protected in 'CLIFFORD'.

**Example 1:** Let's compute the two bilinear forms $\beta_+$ and $\beta_-$ on $S = \text{Cl}(Q)f$ where Cl(Q)=Cl(3,0) is the Clifford algebra of the Euclidean space $R^3$. To shorten output, procedure \texttt{makealiases} is used.

```plaintext
> restart:with(Clifford):with(linalg):B:=diag(1,1,1); # define the diagonal form 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;

Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

    f1 := \frac{Id}{2} + \frac{e1}{2}
    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 'psi' and 'phi' in S = Cl(Q)f = C^2. Notice, that psi1, psi2, phi1, and phi2 belong to a subalgebra K of Cl(Q) spanned by {Id, e2we3} that is isomorphic to C since (e2we3)^2 = -Id. Recall also that the left minimal ideal S = Cl(D)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; #entries in K

    ψ1 := ψ11 \text{Id} + ψ12 e23
    ψ2 := ψ21 \text{Id} + ψ22 e23

> phi1:=phi11 * Id + phi12 * e23; phi2:=phi21 * Id + phi22 * e23; #entries in K

    φ1 := φ11 \text{Id} + φ12 e23
    φ2 := φ21 \text{Id} + φ22 e23

> psi:='f1 &c psi1' + 'f2 &c psi2'; #here psi1 ... are 'complex' components of psi

    psi := 'f1 &c psi1' + 'f2 &c psi2'; #here psi1 ... are 'complex' components of psi

    phi:='f1 &c phi1' + 'f2 &c phi2'; #here phi1 ... are 'complex' components of phi

62
\[ \psi := \text{climul}(f1, \psi1) + \text{climul}(f2, \psi2) \]

\[ \phi := \text{climul}(f1, \phi1) + \text{climul}(f2, \phi2) \]

Now, we compute \( \beta_+ \) on \( \psi \) and \( \phi \), while we store the purespinor under the name 'purespinor1'. Notice, that \( \beta_+ \) is invariant under the unitary group \( U(2) \) (for more information see [1], [4]).

\[ > \text{x} := \text{beta_plus}(\psi, \phi, f, 'purespinor1'); \text{purespinor1}; \ #\beta_+ \text{ is invariant under } U(2) \]

\[ x := (\psi11 \phi11 + \psi22 \phi22 + \psi12 \phi12 + \psi21 \phi21) \text{Id} \]
\[ + (\psi12 \phi11 + \psi11 \phi12 + \psi21 \phi22 - \psi22 \phi21) e23 \]

\[ \text{Id} \]

\[ > \psi1c := \text{reversion}(\psi1); \psi2c := \text{reversion}(\psi2); \]

\[ \psi1c := \psi11 \text{Id} - \psi12 e23 \]

\[ \psi2c := \psi21 \text{Id} - \psi22 e23 \]

\[ > \text{clicollect}(\psi1c \&c \phi1 + \psi2c \&c \phi2); \ #\text{this equals } x \text{ and is invariant under } U(2) \]

\[ (\psi11 \phi11 + \psi22 \phi22 + \psi12 \phi12 + \psi21 \phi21) \text{Id} \]
\[ - (\psi22 \phi21 + \psi12 \phi11 - \psi11 \phi12 - \psi21 \phi22) e23 \]

Now, we compute \( \beta_- \) on \( \psi \) and \( \phi \), while we store the purespinor under the name 'purespinor2'. Notice, that \( \beta_- \) is invariant under the complex symplectic group \( \text{Sp}(2,\mathbb{C}) \) (for more information see [1], [4]).

\[ > \text{y} := \text{beta_minus}(\psi, \phi, f, 'purespinor2'); \text{purespinor2}; \ #\beta_- \text{ is invariant under } \text{Sp}(2,\mathbb{C}) \]

\[ y := (-\psi12 \phi22 + \psi11 \phi21 - \psi21 \phi11 + \psi22 \phi12) \text{Id} \]
\[ + (\psi11 \phi22 - \psi21 \phi12 - \psi22 \phi11 + \psi12 \phi21) e23 \]

\[ e2 \]

\[ > \text{clicollect}(\psi1 \&c \phi2 - \psi2 \&c \phi1); \ #\text{this equals } y \text{ and is invariant under } \text{Sp}(2,\mathbb{C}) \]

\[ -(\psi12 \phi22 - \psi11 \phi21 + \psi21 \phi11 - \psi22 \phi12) \text{Id} \]
\[ - (\psi11 \phi22 + \psi21 \phi12 + \psi22 \phi11 - \psi12 \phi21) e23 \]

Observe that 'purespinor1' and 'purespinor2' have the desired commuting
properties with the idempotent $f$:

\begin{verbatim}
> u:=purespinor1; f &c u - u &c reversion(f);
    u := Id
    0

> u:=purespinor2; f &c u - u &c conjugation(f);
    u := e2
    0
\end{verbatim}

**Example 2:** Scalar products of spinors of the Clifford algebra $\text{Cl}(1,3)$ that is isomorphic to $\text{H}(2)$.

\begin{verbatim}
> restart:with(Clifford):with(linalg):
    B:=linalg[diag](1,-1,-1,-1): #define the bilinear form $B$ for $\text{Cl}(1,3)$

> dim:=coldim(B):eval(makealiases(dim)):

> data:=clidata(B); #retrieve and display data about $\text{Cl}(B)$

    data := 
    [quaternionic, 2, simple, $\frac{\text{Id} + \frac{e14}{2}}{2}$, 
        $\{\text{Id}, e1, e2, e3, e12, e13, e23, e123\}$, 
        $\{\text{Id}, e2, e3, e23\}$, $\{\text{Id}, e1\}$]

> 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;

Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

\begin{verbatim}
    f1 := $\frac{\text{Id}}{2} + \frac{e14}{2}$

    f2 := $\frac{e1}{2} + \frac{e4}{2}$
\end{verbatim}

Elements $f1$ and $f2$ computed above give a $K$-basis in $S = \text{Cl}(1,3)f$ where $K = \text{span}\{\text{Id}, e2, e3, e2we3\}$ is a subalgebra of $\text{Cl}(1,3)$ isomorphic with the quaternionic ring $\text{H}$. The elements spanning $K$ are stored in data[6] displayed above. One can really check that these elements form a
quaternionic basis and form a list of `type/genquatbasis`:

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

    Kbasis := [Id, e2, e3, e23]
```

> type(Kbasis, genquatbasis);

    true

It is also possible to check it directly by computing all possible products and then displaying them in a matrix:

```plaintext
> M := matrix(4, 4, (i, j) -> cmul(Kbasis[i], Kbasis[j]));

    M :=
    [   Id   e2   e3   e23 ]
    [ e2  -Id  e23  -e3  ]
    [ e3  -e23  -Id   e2  ]
    [ e23  e3   -e2  -Id  ]
```

Like in the example above, we first define arbitrary quaternionic coefficients for two spinors 'psi' and 'phi' in \(S = \text{Cl}(1,3)\):

```plaintext
> psi1 := psi11 * Id + psi12 * e2 + psi13 * e3 + psi14 * e23; # first coefficient of psi in K

psi1 := ψ11 * Id + ψ12 * e2 + ψ13 * e3 + ψ14 * e23

> psi2 := psi21 * Id + psi22 * e2 + psi23 * e3 + psi24 * e23; # second coefficient of psi in K

psi2 := ψ21 * Id + ψ22 * e2 + ψ23 * e3 + ψ24 * e23

> phi1 := phi11 * Id + phi12 * e2 + phi13 * e3 + phi14 * e23; # first coefficient of phi in K

phi1 := φ11 * Id + φ12 * e2 + φ13 * e3 + φ14 * e23

> phi2 := phi21 * Id + phi22 * e2 + phi23 * e3 + phi24 * e23; # second coefficient of phi in K

phi2 := φ21 * Id + φ22 * e2 + φ23 * e3 + φ24 * e23
```

Now, we define spinors 'psi' and 'phi':

```plaintext
> psi := 'f1 &c psi1' + 'f2 &c psi2'; # here psi1 ... are quaternionic components of psi

psi := ψ11 * Id + ψ12 * e2 + ψ13 * e3 + ψ14 * e23 + ψ21 * Id + ψ22 * e2 + ψ23 * e3 + ψ24 * e23

> phi := 'f1 &c phi1' + 'f2 &c phi2'; # here phi1 ... are quaternionic components of phi

phi := φ11 * Id + φ12 * e2 + φ13 * e3 + φ14 * e23 + φ21 * Id + φ22 * e2 + φ23 * e3 + φ24 * e23
```

ψ := climul(f1, ψ1) + climul(f2, ψ2)
\[ \phi := \text{climul}(f1, \phi1) + \text{climul}(f2, \phi2) \]

The beta\(_{+}\) and the beta\(_{-}\) forms on S x S are invariant under the symplectic group Sp(2,2) as can be checked directly from the output below:

\[ x := \text{beta\(_{+}\)}(\psi, \phi, f, 'purespinor1'); \text{purespinor1}; \text{#beta\(_{+}\) is invariant under Sp(2,2)} \]

\[ x := \]
\[ (\psi12 \phi22 + \psi23 \phi13 + \psi14 \phi24 + \psi24 \phi14 + \psi21 \phi11 + \psi22 \phi12 + \psi11 \phi21 + \psi13 \phi23) Id \]
\[ + (\psi22 \phi11 - \psi23 \phi14 - \psi12 \phi21 + \psi24 \phi13 - \psi13 \phi24 + \psi14 \phi23 + \psi11 \phi22 + \psi21 \phi12) e2 \]
\[ + (\psi22 \phi14 - \psi24 \phi12 - \psi23 \phi11 + \psi11 \phi23 + \psi12 \phi24 - \psi13 \phi21 + \psi21 \phi13 - \psi14 \phi22) e3 \]
\[ + (\psi24 \phi11 + \psi21 \phi14 + \psi11 \phi24 + \psi23 \phi12 - \psi14 \phi21 - \psi22 \phi13 + \psi13 \phi22 - \psi12 \phi23) e23 \]
\[ e1 \]

\[ y := \text{beta\(_{-}\)}(\psi, \phi, f, 'purespinor2'); \text{purespinor2}; \text{#beta\(_{-}\) is invariant under Sp(2,2)} \]

\[ y := \]
\[ (\psi23 \phi13 + \psi11 \phi21 - \psi24 \phi14 - \psi21 \phi11 + \psi22 \phi12 - \psi13 \phi23 - \psi12 \phi22 + \psi14 \phi24) Id \]
\[ + (\psi14 \phi23 - \psi21 \phi12 + \psi12 \phi21 - \psi22 \phi11 - \psi23 \phi14 - \psi24 \phi13 + \psi13 \phi24 + \psi11 \phi22) e2 \]
\[ + (-\psi14 \phi22 + \psi11 \phi23 - \psi23 \phi11 + \psi13 \phi21 - \psi21 \phi13 - \psi12 \phi24 + \psi24 \phi12 + \psi22 \phi14) e3 \]
\[ + (-\psi14 \phi21 - \psi22 \phi13 + \psi23 \phi12 + \psi12 \phi23 + \psi24 \phi11 + \psi11 \phi24 - \psi13 \phi22 - \psi21 \phi14) e23 \]
\[ e1 \]

**Example 3:** Scalar products in the spinor spaces of the Clifford algebra Cl(3,1), that is isomorphic to R(4), are both invariant under the symplectic group Sp(4).

\[ \text{restart}: \text{bench} := \text{time}(): \text{with(Clifford)}: \text{dim} := 4: \text{eval} \text{(makealiases(dim))}: \]
\[ B := \text{linalg[diag]}(1, 1, 1, -1): \text{#define form B for Cl(3,1)} \]
\[ \text{data} := \text{clidata}(B); \text{#retrieve and display data about Cl}(3,1) \]
\[ data := \begin{bmatrix} \text{real, 4, simple, 'cmulQ} \left( \frac{\text{Id}}{2} + \frac{e1}{2}, \frac{\text{Id}}{2} + \frac{e34}{2} \right), \left[ \text{Id}, e2, e3, e23 \right], \left[ \text{Id} \right], \left[ \text{Id}, e2, e3, e23 \right] \end{bmatrix} \]

\[ > f := \text{data}[4]; \# \text{assign pre-stored idempotent to } f \text{ or use your own here} \]

Here are the matrices representing some basis monomials of Cl(B). Those that represent the 1-vectors are usually referred to as \textit{Dirac gamma matrices} (since 'gamma' is a protected Maple name, we will use here 'g' instead):

\[ > \text{matKrepr(Bsignature());} \]

\text{Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.}

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

Let's find matrices representing the four basis vectors in S:

\[ > \text{for } i \text{ from 1 to 4 do } f||i := \text{data}[7][i] \&c f \text{ od;} \]

\[ f1 := \frac{\text{Id}}{4} + \frac{e1}{4} + \frac{e34}{4} + \frac{e134}{4} \]

\[ f2 := \frac{e2}{4} - \frac{e12}{4} - \frac{e1234}{4} + \frac{e234}{4} \]

\[ f3 := \frac{e3}{4} + \frac{e4}{4} - \frac{e13}{4} - \frac{e14}{4} \]

\[ f4 := \frac{e23}{4} + \frac{e24}{4} + \frac{e123}{4} + \frac{e124}{4} \]

Thus, a spinor psi from S=Cl(3,1)f is a four component vector with real entries. It can be written in terms of the basis [f1,f2,f3,f4] and its real entries in K = <Id>_R as follows:

\[ > \text{psi} := \text{psi1} \ast 'f1' + \text{psi2} \ast 'f2' + \text{psi3} \ast 'f3' + \text{psi4} \ast 'f4'; \]
\[ \phi := \text{phi1} \ast 'f1' + \text{phi2} \ast 'f2' + \text{phi3} \ast 'f3' + \text{phi4} \ast 'f4'; \]

\[ \psi := \psi1 f1 + \psi2 f2 + \psi3 f3 + \psi4 f4 \]
\[ \phi := \phi1 f1 + \phi2 f2 + \phi3 f3 + \phi4 f4 \]
x := \beta_+(\psi, \phi, f, 's1'); s1; \text{ #automorphism group of } \beta_+ \text{ is } \text{Sp}(4, \mathbb{R})

\begin{align*}
x &:= (-\psi_1 \phi_4 - \psi_2 \phi_3 + \psi_4 \phi_1 + \psi_3 \phi_2) \text{Id} \\
&= e_{23}
\end{align*}

y := \beta_-(\psi, \phi, f, 's2'); s2; \text{ #automorphism group of } \beta_- \text{ is } \text{Sp}(4, \mathbb{R})

\begin{align*}
y &:= (\psi_1 \phi_3 - \psi_2 \phi_4 + \psi_4 \phi_2 - \psi_3 \phi_1) \text{Id} \\
&= e_{3}
\end{align*}

Since the coefficients of '\psi' and '\phi' are here assumed to be real, it doesn't matter that they were written on the left side of the basis elements \{f_1, f_2, f_3, f_4\}.


13. Continuous families of idempotents in Clifford algebras

In this example we will show how it is possible to find continuous families of idempotents in a Clifford algebra \text{Cl}(B) by finding a general solution to \( f^2 = \)}
f in Cl(B). In our example B = Q = diag(1,1), that is, Cl(Q) = Cl(2,0) is the Clifford algebra of the Euclidean plane. We will verify that CLIFFORD gives a correct answer by using Groebner basis approach to solve a system of four polynomial equations in variables \{x_0, x_1, x_2, x_{12}\} with lex order x_0 > x_1 > x_2 > x_{12}:

```maple
restart: with(linalg): with(Clifford): with(Groebner):
B := diag(1,1): dim := coldim(B): eval(makealiases(dim)): clibasis := cbasis(dim):
clidata();
```

Let u be the most general element in Cl(2,0). We require that u be an idempotent so it must satisfy equation \(u^2 = u\):

```maple
u := x_0*Id + x_1*e_1 + x_2*e_2 + x_{12}*e_{12};

eq := clicollect(cmul(u,u)-u);
```

Thus, we have to solve the following system of four polynomial equations in variables \(\{x_0, x_1, x_2, x_{12}\}\):  

```maple
for i from 1 to 2^dim do p||i := factor(coeff(eq, clibasis[i]))
end do;
sys := \{p||1..2^dim\}:
vars := \{seq(coeff(u, clibasis[i]), i=1..2^dim)\}:

p1 := x_0^2 + x_2^2 - x_{12}^2 + x_1^2 - x_0
p2 := x_1 \(2 \cdot x_0 - 1\)

p3 := x_2 \(2 \cdot x_0 - 1\)

p4 := x_{12} \(2 \cdot x_0 - 1\)
```

The continuous family of idempotents in Cl(2,0) is determined by an affine variety \(V(p_1, p_2, p_3, p_4)\) in \(R^4\). First, we just tell Maple to solve the system
\[ p_1 = p_2 = p_3 = p_4 = 0: \]

\[ \text{sol} := \text{remove(has, map(allvalues, \{solve(sys, vars)\}), I);} \]

\[ \text{sol := \{ x0 = 0, x1 = 0, x12 = 0, x2 = 0 \}, x0 = 1, x1 = 0, x12 = 0, x2 = 0 \}, } \]

\[ \begin{align*} 
\{ x0 &= \frac{1}{2}, x1 = 0, x12 = -\sqrt{1 - 4x_2^2} x_2 = x_2 \}, \\
\{ x0 &= \frac{1}{2}, x1 = 0, x12 = \sqrt{1 - 4x_2^2} x_2 = x_2 \}, \\
\{ x0 &= \frac{1}{2}, x1 = -\sqrt{1 - 4x_2^2 + 4x_12^2} x_12 = x_12, x2 = x_2 \}, \\
\{ x0 &= \frac{1}{2}, x1 = \sqrt{1 - 4x_2^2 + 4x_12^2} x_12 = x_12, x2 = x_2 \} \}
\]

\[ \text{seq(subs(sol[i], \{p1, p2, p3, p4\}), i = 1 .. nops(sol)); \# verification that these solutions are correct} \]

\[ \{ 0, 0, 0, 0, 0, 0, 0, 0 \} \]

\[ \text{for i from 1 to nops(sol) do f||i := subs(sol[i], u) end do; \# displaying all idempotent types} \]

\[ f1 := 0 \]

\[ f2 := \text{Id} \]

\[ f3 := \frac{\text{Id}}{2} + x_2 e_2 - \frac{\sqrt{1 - 4x_2^2} e_{12}}{2} \]

\[ f4 := \frac{\text{Id}}{2} + x_2 e_2 + \frac{\sqrt{1 - 4x_2^2} e_{12}}{2} \]

\[ f5 := \frac{\text{Id}}{2} - \frac{\sqrt{1 - 4x_2^2 + 4x_12^2} e_1}{2} + x_2 e_2 + x_12 e_{12} \]

\[ f6 := \frac{\text{Id}}{2} + \frac{\sqrt{1 - 4x_2^2 + 4x_12^2} e_1}{2} + x_2 e_2 + x_12 e_{12} \]

\[ \text{seq(cmul(f||i, f||i) - f||i, i = 1 .. nops(sol)); \# verifying that f1, ..., f8 are all idempotents in Cl(2, 0):} \]

\[ 0, 0, 0, 0, 0, 0 \]
We can also solve the above system of equations using CLIFFORD's built-in procedure 'clisolve' which automatically solves for the unknown parameters that appear in u. We will use now the Groberner basis approach and verify that the above solutions are all possible real solutions. We will use lex order with x0>x1>x2>x12.

First, we find the Groebner basis for the variety \(V(p_1,p_2,p_3,p_4)\):

\[
\begin{align*}
&GB1 := gbasis([p||(1..4)], plex(x0,x1,x2,x12)) ; nops(GB1); \\
&\text{for } i \text{ from } 1 \text{ to } 7 \text{ do } g||i := \text{factor}(GB1[i]) \text{ end do};
\end{align*}
\]

Warning, Groebner[gbasis] is deprecated. Please, use Groebner[Basis].

\[
\begin{align*}
GB1 := [4 x12 x2^2 - x12 - 4 x12^3 + 4 x12 x1^2, 4 x2^3 - x2 - 4 x12^2 x2 + 4 x2 x1^2, \\
4 x1 x2^2 - x1 - 4 x12^2 x1 + 4 x1^3, 2 x12 x0 - x12, -x2 + 2 x2 x0, 2 x1 x0 - x1, \\
x0^2 + x2^2 - x12^2 + x1^2 - x0]
\end{align*}
\]

\[
\begin{align*}
g1 &:= x12 (4 x2^2 - 1 - 4 x12^2 + 4 x1^2) \\
g2 &:= x2 (4 x2^2 - 1 - 4 x12^2 + 4 x1^2) \\
g3 &:= x1 (4 x2^2 - 1 - 4 x12^2 + 4 x1^2) \\
g4 &:= x12 (2 x0 - 1) \\
g5 &:= x2 (2 x0 - 1) \\
g6 &:= x1 (2 x0 - 1) \\
g7 &:= x0^2 + x2^2 - x12^2 + x1^2 - x0
\end{align*}
\]

Thus, \(V(p_1,p_2,p_3,p_4) = <g1,g2,g3,g4,g5,g6,g7>\) where g1,g2,...,g7 provide a Groebner basis.

Observe, that the above system can be easily solved. One solution is obtained when \(x0=1/2\):

\[
\begin{align*}
g11 &:= \text{subs}(x0=1/2, g1); \\
g21 &:= \text{subs}(x0=1/2, g2); \\
g31 &:= \text{subs}(x0=1/2, g3); \\
g71 &:= 4*\text{subs}(x0=1/2, g7); \\
H &:= [\text{solve}(g71, x1)];
\end{align*}
\]
\[
g_{11} := x_{12} \left( 4x^2 - 1 - 4x^2 - 4x^2 \right)
g_{21} := x_2 \left( 4x^2 - 1 - 4x^2 - 4x^2 \right)
g_{31} := x_1 \left( 4x^2 - 1 - 4x^2 - 4x^2 \right)
g_{71} := 4x^2 - 1 - 4x^2 - 4x^2
\]

\[
H := \begin{bmatrix}
\frac{\sqrt{1 - 4x^2 + 4x_{12}^2}}{2} & -\frac{\sqrt{1 - 4x^2 + 4x_{12}^2}}{2}
\end{bmatrix}
\]

which gives the following two idempotents that differ only in one sign:

\[
h_1 := \text{subs} \{x_0 = 1/2, x_1 = H[1], x_2 = x_2, x_{12} = x_{12}, u\};
h_2 := \text{subs} \{x_0 = 1/2, x_1 = H[2], x_2 = x_2, x_{12} = x_{12}, u\};
\]

\[
\text{cmul}(h_1, h_1) - h_1, \text{cmul}(h_2, h_2) - h_2;
\]

\[
h_1 := \frac{\text{Id}}{2} + \frac{\sqrt{1 - 4x^2 + 4x_{12}^2} \cdot e_1}{2} + x_2 \cdot e_2 + x_{12} \cdot e_{12}
\]

\[
h_2 := \frac{\text{Id}}{2} - \frac{\sqrt{1 - 4x^2 + 4x_{12}^2} \cdot e_1}{2} + x_2 \cdot e_2 + x_{12} \cdot e_{12}
\]

Second solution we get when \(x_0 \not= 1/2\) then \(x_1 = x_2 = x_{12} = 0\):

\[
g_{72} := \text{factor} \left( \text{subs} \{x_1 = 0, x_2 = 0, x_{12} = 0\}, g_7 \right);
\]

\[
g_{72} := x_0 \left( -1 + x_0 \right)
\]

so \(x_0 = 0\) or \(x_0 = 1\), and we get trivial idempotents \text{Id} and 0. So, \(h_1\) and \(h_2\) are the nontrivial ones only provided \(1 + 4x_{12}^2 - 4x_2^2 \geq 0\).

Assume that \(1 + 4x_{12}^2 - 4x_2^2 = 0\) and solve for \(x_2\):

\[
K := [\text{solve}(1 + 4x_{12}^2 - 4x_2^2 = 0, x_2)];
\]

\[
K := \begin{bmatrix}
\frac{\sqrt{1 + 4x_{12}^2}}{2} & -\frac{\sqrt{1 + 4x_{12}^2}}{2}
\end{bmatrix}
\]

\[
h_3 := \text{subs} \{x_0 = 1/2, x_1 = 0, x_2 = K[1], x_{12} = x_{12}, u\};
h_4 := \text{subs} \{x_0 = 1/2, x_1 = 0, x_2 = K[2], x_{12} = x_{12}, u\};
\]

\[
\text{cmul}(h_3, h_3) - h_3, \text{cmul}(h_4, h_4) - h_4;
\]
Thus, the idempotents $h_1$ and $h_2$ are the same most general non-trivial idempotents as $f_1$ and $f_2$ obtained before.

Let's graph this variety $V(p_1, p_2, p_3, p_4)$ in $\mathbb{R}^3$ when $x_0 = 1/2$ which is given implicity via the polynomial

$$g_{71} := -1 + 4 x_1^2 + 4 x_2^2 - 4 x_12^2$$

We will plot this variety in cylindrical coordinates chosen as follows:

$x_1 = r \cos(\theta)$;
$x_2 = r \sin(\theta)$;
$x_{12} = z$

then we get

$$4r^2 - 4z^2 = 1$$

```plaintext
> with(plots):
> eq := -4*x12^2 + 4*x1^2 + 4*x2^2 - 1;
> eq := 4 x2^2 - 1 - 4 x12^2 + 4 x1^2
> plot3d(sqrt(1+4*z^2)/4,
        theta=0..2*Pi, z=-3..3, coords=cylindrical, axes=boxed, labels=[x1, x2, x12], title=`Idempotent variety for Cl(2,0) (hyperboloid -4*x12^2+4*x1^2+4*x2^2=1 and its inside)`);
```
All nontrivial idempotents $h_1$ and $h_2$ found above are ungraded, i.e., they are neither odd nor even. If we set $x_2=x_{12}=0$, then the above two idempotents become the classical ones:
However, \( x_2 e_2 + \frac{1}{2} \left( \frac{1}{2} \right) e_1 + x_{12} e_{12} + \frac{1}{2} \) gives a two-parameter family of idempotents in \( \text{Cl}(2,0) \) as long as \( 1 - 4x_2^2 + 4x_{12}^2 \geq 0 \). The classical (discrete) idempotents occupy the center \((0,0)\) of the parameter region \( 4x_1^2 + 4x_2^2 = 4 \) in the \( x_1 \times x_2 \)-plane.

Similar analysis can be performed in the neutral Clifford algebra with the roles of \( x_{12} \) and \( x_2 \) parameters being switched: one gets the hyperboloid

\[
4x_{12}^2 + 4x_1^2 - 4x_2^2 = 1
\]

instead of

\[
-4x_{12}^2 + 4x_1^2 + 4x_2^2 = 1.
\]

For the Clifford algebra \( \text{Cl}(0,2) \) of the anti-Euclidean plane one of course gets only the trivial idempotents. This is because \( \text{Cl}(0,2) = \mathbb{H} \).

Reference


### 14. Vahlen matrices

- Procedure 'isVahlenmatrix' determines if the given 2 x 2 Clifford matrix 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 x 2 matrix $V$ with entries in a Clifford algebra $\text{Cl}(p, q)$ such that if

$$V := \text{matrix}(2, 2, [a, b, c, d]);$$

- and $a, b, c, d$ are elements in $\text{Cl}(p, q)$, then the following conditions must be met:
  1. $a, b, c, d$ are products of 1-vectors,
  2. the pseudodeterminant of $V$ is $+1$ or $-1$ (or, $\text{Id}$ or $-\text{Id}$ in the algebra),
  3. $a \& c \text{reversion}(b)$, $\text{reversion}(b) \& c \, d$, $d \& c \text{reversion}(c)$, and $\text{reversion}(c) \& c \, a$ are all 1-vectors.

- See `reversion` for more information on this Clifford algebra anti-automorphism and `\&c` for information about the Clifford product in $\text{Cl}(B)$. For information about the pseudodeterminant see `pseudodet`.

- Condition (1) above implies that $a, b, c, d$ are elements of the Lipschitz group of $R^{(p,q)}$. 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 Mobius transformations and are compositions of rotations, translations, dilations and transversions (called also special conformal transformations). A Mobius transformation in $R^{(p,q)}$ can be written in the form

$$x \rightarrow (a \, x + b) \, (c \, x + d)^{-1}$$

- where $x$ is a 1-vector that belongs to $R^{(p,q)}$, $a, b, c, d$ belong to $\text{Cl}(p,q)$, and, the products and the inverse are taken in $\text{Cl}(p,q)$. This transformation may be represented by the Vahlen matrix $V$ defined above.

- Rotations, translations, dilations, and transversions will then be represented as follows:

  - rotations $x \rightarrow a \, x \, a^{(-1)}$ where $a$ is in $\text{Spin}^{+(p,q)}$, $V = \text{matrix}(2, 2, [a, 0, 0, a])$
  - translations $x \rightarrow x + b$ where $b$ is in $R^{(p,q)}$, $V =$
matrix(2, 2, [1, b, 0, 1])
- dilations \( x \rightarrow s \times \) where \( s > 0 \), \( V = \)
  matrix(2, 2, [sqrt(s), 0, 0, 1/sqrt(s)])
- transversions \( x \rightarrow (x + x^2 \cdot c)/(1 + 2 \cdot x \cdot c + x^2 \cdot c^2) \), \( V = \)
  matrix(2, 2, [1, 0, c, 1]),
where \( c \) is in \( \mathbb{R}^{(p,q)} \), and \( x \cdot c \) is the dot product in \( \mathbb{R}^{(3,1)} \).

**Example 1:** Simple cases in the signature \((3,1)\).

(1) Rotations:

```plaintext
> restart:with(Clifford):eval(makealiases(6)):
> 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]);
> isVahlenmatrix(R);
```

```plaintext
true
```

(2) Translations:

```plaintext
> b:=e1+2*e3; #vector in \( \mathbb{R}^{(3,1)} \)
> T:=linalg[matrix](2,2,[1,b,0,1]);
> isVahlenmatrix(T);
```

```plaintext
b := e1 + 2 e3
T :=
\[
\begin{bmatrix}
1 & e1 + 2 e3 \\
0 & 1
\end{bmatrix}
\]```
(3) Dilations:

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

Dil :=
[ 1     0 ]
[ 2     2 ]
true
```

(4) Transversions:

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

c := 2 e1 - e3
Tv :=
[ 1           0 ]
[ 2 e1 - e3   1 ]
true
```

**Example 2**: If we now take a product of these four matrices above, we will obtain an element of the conformal group in R^(3,1):

```plaintext> conf:=R &cQm T &cQm Dil &cQm Tv;
Conf :=
[ e12        10 e23          -2 e2 + 4 e123 ]
[ -4 e2 - 2 e123  2 e12   ]

true
```

Since in the product above each matrix appeared exactly once, the diagonal entries of 'conf' must be invertible:

```plaintext> cinv(conf[1,1]); #inverse of conf[1,1]
- 2 e12
- 10 e23
- 40 e2
- 401

> cmul(%,'conf[1,1]'); #let's verify the inverse of conf[1,1]
Id
```
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.

**Example 3:** (Johannes Maks) Example of a Vahlen matrix $W$ without any invertible entries. Matrix $W$ defined below represents an element in the identity component of the conformal group of $\mathbb{R}^{3,1}$.

$$W := \text{evalm}((1/2) \times \text{linalg} \{\text{matrix}\}(2,2,[1-e14,-e1+e4,e1+e4,1+e14]));$$

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

$$\text{type}(W[1,1], \text{idempotent}); \quad \text{element (1,1) of } W \text{ is an idempotent}$$

$$\text{type}(W[2,2], \text{idempotent}); \quad \text{element (2,2) of } W \text{ is an idempotent}$$

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

$$\text{cmul}(W[1,2], W[1,2]), \text{cmul}(W[2,1], W[2,1]);$$

$$0, 0$$

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

$$\text{isVahlenmatrix}(W);$$

$$\text{true}$$
However, $W$ is an element of the identity component of the conformal group in $R^{\wedge}(3,1)$ 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 not a product of just one rotation, one translation, one dilation, and/or one transversion:

$$Tv := \text{linalg}[\text{matrix}](2,2,[1,0,(e1+e4)/2,1]);$$

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

$$T := \text{linalg}[\text{matrix}](2,2,[1,(-e1+e4)/2,0,1]);$$

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

$$Tv \&cQm T \&cQm Tv, \text{displayid}(W); \ # W = Tv \&cQm T \&cQm Tv$$

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

$$\text{pseudodet}(W); \ # \text{computing pseudo-determinant of } W$$

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

**Example 4:** This 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 $R^{\wedge}(3,1)$.

$$W := \text{evalm}((1/2)*\text{linalg}[\text{matrix}](2,2,[1-e24,-e2+e4,e2+e4,1+e24]));$$

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

Notice that the diagonal elements of $W$ are non-trivial idempotents in
Cl(3,1), hence they are not invertible in Cl(3,1):

```plaintext
> 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:

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

0, 0
```

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

```plaintext
> 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:

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

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

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

T :=
\[
\begin{bmatrix}
1 & -\frac{e2 + e4}{2} \\
\frac{e2}{2} + \frac{e4}{2} & 1
\end{bmatrix}
\]

> Tv &cQm T &cQm Tv,map(displayid,W); # W = Tv &cQm T &cQm Tv

\[
\begin{bmatrix}
\frac{1}{2} & -\frac{e2 + e4}{2} & \frac{1}{2} & -\frac{e2 + e4}{2} \\
\frac{e2 + e4}{2} & \frac{1}{2} & -\frac{e2 + e4}{2} & \frac{1}{2}
\end{bmatrix}
\]

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

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

81
Thus, the above computation confirms that $W = T v \& c Q m T \& c Q m T v$ and that the pseudo-determinant of $W$ is 1.

References


15. Singular Value Decomposition and Clifford algebra

We can check that the following procedure gives an anti-automorphism $t p$ of $C l(p, q)$ that for every element $u$ of $C l(p, q)$ and its matrix representation $U$ under a spinor representation of $C l(p, q)$, the matrix that corresponds to $t p(u)$ is the matrix transpose of $U$. On homogeneous elements of $C l(p, q)$, $t p$ is defined as follows:

$$t p: e_1 \wedge e_2 \wedge e_3 \wedge ... \wedge e_n \rightarrow Q[i, i]^{*} Q[j, j]^{*} Q[k, k]^{*} ... Q[n, n]^{*} (e_n \wedge ... \wedge e_k \wedge j_k \wedge e_i) = Q[i, i]^{*} Q[j, j]^{*} Q[k, k]^{*} ... Q[n, n]^{*} r e v e r s i o n (e_i \wedge e_j \wedge e_k \wedge ... \wedge e_n)$$

and it is extended by linearity to all elements of $C l(p, q)$.

```maple
> restart: with(linalg): with(Clifford): with(asvd): alias(t=transpose):

We can check that the following procedure gives an anti-automorphism $t p$ of $C l(p, q)$ that for every element $u$ of $C l(p, q)$ and its matrix representation $U$ under a spinor representation of $C l(p, q)$, the matrix that corresponds to $t p(u)$ is the matrix transpose of $U$. On homogeneous elements of $C l(p, q)$, $t p$ is defined as follows:

$$t p: e_1 \wedge e_2 \wedge e_3 \wedge ... \wedge e_n \rightarrow Q[i, i]^{*} Q[j, j]^{*} Q[k, k]^{*} ... Q[n, n]^{*} (e_n \wedge ... \wedge e_k \wedge j_k \wedge e_i) = Q[i, i]^{*} Q[j, j]^{*} Q[k, k]^{*} ... Q[n, n]^{*} r e v e r s i o n (e_i \wedge e_j \wedge e_k \wedge ... \wedge e_n)$$

and it is extended by linearity to all elements of $C l(p, q)$.

```
Example 1: Let's find the Singular Value Decomposition of a 2 x 2 matrix of rank 2 and the SVD of its isomorphic image in Cl(2,0).

> A:=matrix(2,2,[2,3,1,2]);#defining A
m:=rowdim(A); #number of rows of A is m
n:=coldim(A); #number of columns of A is n
r:=rank(A);   #rank of A is r

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

\[
m := 2 \\
n := 2 \\
r := 2
\]

We can work either with Cl(2,0) or Cl(1,1) as in each case these Clifford algebras are isomorphic to Mat(2,R). We will work with Cl(2,0):

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

[[1,1],[2,0]]

> dim:=2:B:=diag(1,1);

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

> eval(makealiases(dim));

Recall information about Cl(2,0):

> data:=clidata([2,0]);

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

Define a Grassmann basis in Cl(2,0), assign a primitive idempotent to f, and generate a spinor basis in Cl(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)
    
    f := \frac{Id}{2} + \frac{e1}{2}

> SBgens:=data[5]; #generators for a real basis in S
    
    SBgens := [Id,e2]

> FBgens:=data[6]; #generators for K
    
    FBgens := [Id]

Here 'SBgens' is a K-basis for S=Cl(B)f modulo f. Since for the current signature (2,0) we have that K=R, S=R(2), and Cl(2,0)=Mat(2,R), the output from 'spinorKbasis' shown below has four basis vectors and their generators modulo f:

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

Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

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

Note: the second list in 'Kbasis' gives generators of the elements in the first list in 'Kbasis'. Thus, the real spinor basis in S consists of the following four polynomials:

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

    f1 := \frac{Id}{2} + \frac{e1}{2}

    f2 := \frac{e2}{2} - \frac{e12}{2}

Now, we compute matrices m1, m2, m3, m4 representing each basis element in Cl(2,0) isomorphic with Mat(2,R) (real 2 x 2 matrices).

> for i from 1 to nops(clibas) do
    lprint(`The basis element`,clibas[i],`is represented by the`);  
    M[i] := subs(Id=1,matKrepr(clibas[i])); 
    od;
The basis element, \( \text{Id} \), is represented by the following matrix:

\[
M_1 := \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

The basis element, \( e_1 \), is represented by the following matrix:

\[
M_2 := \begin{bmatrix}
1 & 0 \\
0 & -1
\end{bmatrix}
\]

The basis element, \( e_2 \), is represented by the following matrix:

\[
M_3 := \begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\]

The basis element, \( e_{12} \), is represented by the following matrix:

\[
M_4 := \begin{bmatrix}
0 & 1 \\
-1 & 0
\end{bmatrix}
\]

We will use the procedure 'phi' which gives an isomorphism from Mat(2,R) to Cl(2,0). This way we can find the image 'p' in Cl(2,0) of any real 2 x 2 real matrix \( A \). Knowing the image of each matrix \( m_1, m_2, \ldots, m_4 \) in terms of the Clifford polynomials in Cl(2,0), we can easily find the image of \( A \).

Notice that in Cl(2,0), reversion corresponds to matrix transposition. WARNING: This is not always the case! However, transposition corresponds to the antiautomorphism \( \text{tp} \) defined above which in this case reduces to the reversion.

Let \( AA \) be a general 2 x 2 real matrix and let \( pAA = \phi(AA,M) \) be its isomorphic image in Cl(2,0) in the basis \([M1,M2,M3,M4]\) defined above.

\[
AA := \begin{bmatrix} a & b \\ c & d \end{bmatrix}
\]

\[
pAA := \text{Id} \left( \frac{a + d}{2} \right) + \left( -\frac{d + a}{2} \right) e1 + \left( \frac{b + c}{2} \right) e2 + \left( -\frac{c + b}{2} \right) e12
\]

Let's transpose \( AA \) and find the image \( pAAt \) of the transposed matrix \( t(AA) = AAt \) in Cl(2,0):

\[
AAt := \begin{bmatrix} a & c \\ b & d \end{bmatrix}
\]
\[ p\text{AA}t := \text{Id} \left( \frac{a}{2} + \frac{d}{2} \right) + \left( -\frac{d}{2} + \frac{a}{2} \right) e1 + \left( \frac{b}{2} + \frac{c}{2} \right) e2 + \left( \frac{c}{2} - \frac{b}{2} \right) e12 \]

In signature (2,0), the reversion and the tp-antiautomorphism are identical, hence

\[ \text{simplify}(\text{reversion}(p\text{AA}) - p\text{AA}t), \text{simplify}(\text{tp}(p\text{AA}) - p\text{AA}t); \]

0, 0

Let's find the image of A defined above:

\[ p := \phi(A, M); \quad \text{# finding image of A in Cl}(2,0) \]

\[ p := 2 \text{Id} + 2 e2 + e12 \]

\[ pT := \phi(t(A), M); \quad \text{# finding image of } A^T \text{ in } Cl(2,0) \]

\[ pT := 2 \text{Id} + 2 e2 - e12 \]

\[ \text{ATA} := \text{evalm}(t(A) &* A); \quad \text{# finding matrix ATA} \]

\[ \text{ATA} := \begin{bmatrix} 5 & 8 \\ 8 & 13 \end{bmatrix} \]

\[ pTp := \phi(\text{ATA}, M); \quad \text{# finding image of ATA in Cl}(2,0) \]

\[ pTp := 9 \text{Id} - 4 e1 + 8 e2 \]

which should be the same as the Clifford product of pT and p:

\[ \text{cmul}(pT, p); \]

\[ 9 \text{Id} - 4 e1 + 8 e2 \]

Notice that we can solve the eigenvalue problem for the Clifford number pTp directly in Cl(2,0). The minimal polynomials of pTp and ATA are the same, of course:

\[ \text{pol} := \text{charpoly}(\text{ATA}, x); \quad \text{# characteristic polynomial of ATA} \]

\[ \text{pol} := \text{climinpoly}(pTp); \]

\[ \text{pol} := x^2 - 18 x + 1 \]

\[ \text{pol} := x^2 - 18 x + 1 \]

Let's define a general eigenspinor as element of \( S = \text{Cl}(2,0)f \), set up the eigen equation, and solve it using the command 'clisolve':
> eigenspinor := x1*f1 + x2*f2;
eigeneq := clicollect (expand (cmul (pTp, eigenspinor) - lam*eigenspinor));

\[
eigenspinor := x1 \left( \frac{Id}{2} + \frac{e1}{2} \right) + x2 \left( \frac{e2}{2} - \frac{e12}{2} \right)
\]

\[
eigeneq := -\frac{(-5 \times x1 - 8 \times x2 + lam \times x1) \times Id}{2} - \frac{(-5 \times x1 - 8 \times x2 + lam \times x1) \times e1}{2} - \frac{(lam \times x2 - 13 \times x2 - 8 \times x1) \times e2}{2} + \frac{(lam \times x2 - 13 \times x2 - 8 \times x1) \times e12}{2}
\]

> sol := remove (has, map (allvalues, clisolve (eigeneq, [lam, x1, x2])), lam = lam); # removes trivial solutions

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

Observe, that the Clifford number pTp shown above has two eigenvalues and two corresponding eigenvectors (eigenspinors) parametrized by a real parameter x2 (the values of x2 in each case could be different). Let's find these eigenspinors and their corresponding matrices:

> lam1, lam2 := 9 + 4*sqrt(5), 9 - 4*sqrt(5);

\[
lam1, lam2 := 9 + 4 \sqrt{5}, 9 - 4 \sqrt{5}
\]

> eigenspinor1 := subs (sol[1], eigenspinor);
eigenspinor2 := subs (sol[2], eigenspinor);

\[
eigenspinor1 := x1 \left( \frac{Id}{2} + \frac{e1}{2} \right) + x1 \times (4 - 4 \sqrt{5}) \times \left( \frac{e2}{2} - \frac{e12}{2} \right)
\]

\[
eigenspinor2 := x1 \left( \frac{Id}{2} + \frac{e1}{2} \right) + x1 \times (4 + 4 \sqrt{5}) \times \left( \frac{e2}{2} - \frac{e12}{2} \right)
\]

We just check that the eigenequation is satisfied by the Clifford numbers pTp, lam1, lam2, eigenspinor1 and eigenspinor1:

> simplify (cmul (pTp, eigenspinor1) - lam1*eigenspinor1), simplify (cmu
\[
1(\rho_T, \text{eigenspinor}_2) - \lambda_2 \cdot \text{eigenspinor}_2);
\]
\[
-4 x_1 \text{Id} \sqrt{5} - 4 x_1 e_1 \sqrt{5} - 2 x_1 \sqrt{5} e_2 + 10 x_1 e_2 + 2 x_1 \sqrt{5} e_{12} - 10 x_1 e_{12},
4 x_1 \text{Id} \sqrt{5} + 4 x_1 e_1 \sqrt{5} + 2 x_1 \sqrt{5} e_2 + 10 x_1 e_2 - 2 x_1 \sqrt{5} e_{12} - 10 x_1 e_{12}
\]

Observe that as expected both eigenspinors are one column matrices. These vectors vec1 and vec2 should be the eigenvectors of the matrix \( \text{ATA} \):

\[
\text{vec11} := \text{matKrepr} (\text{eigenspinor}_1); \text{vec22} := \text{matKrepr} (\text{eigenspinor}_2);
\]

\[
\text{vec11} := \begin{bmatrix}
x_I \\
\frac{1}{2} x_I - \frac{1}{2} x_I \sqrt{5}
\end{bmatrix},
\text{vec22} := \begin{bmatrix}
x_I \\
\frac{1}{2} x_I + \frac{1}{2} x_I \sqrt{5}
\end{bmatrix}
\]

\[
\text{vec1} := \text{delcols} (\text{matKrepr} (\text{eigenspinor}_1), 2..2); \text{vec2} := \text{delcols} (\text{matKrepr} (\text{eigenspinor}_2), 2..2);
\]

\[
\text{vec1} := \begin{bmatrix}
x_I \\
\frac{1}{2} x_I - \frac{1}{2} x_I \sqrt{5}
\end{bmatrix},
\text{vec2} := \begin{bmatrix}
x_I \\
\frac{1}{2} x_I + \frac{1}{2} x_I \sqrt{5}
\end{bmatrix}
\]

\[
\text{map} (\text{simplify}, \text{evalm} (\text{ATA} \ &* \ \text{vec1} - \lambda_1 \cdot \text{vec1})), \text{map} (\text{simplify}, \text{evalm} (\text{ATA} \ &* \ \text{vec2} - \lambda_2 \cdot \text{vec2}));
\]

\[
\begin{bmatrix}
-8 x_I \sqrt{5} \\
20 x_I - 4 x_I \sqrt{5}
\end{bmatrix}, \begin{bmatrix}
8 x_I \sqrt{5} \\
20 x_I + 4 x_I \sqrt{5}
\end{bmatrix}
\]

Let's compare these results with a similar computation in the matrix language applied to the matrix \( \text{ATA} \):

\[
\text{L} := \text{sort} ([\text{eigenvects} (\text{ATA})], \text{byeigenvals}); \text{finding eigenvectors of ATA}
\]

\[
\text{L} := \left[ \begin{bmatrix} 9 + 4 \sqrt{5}, 1, \{1, \frac{1}{2} + \frac{\sqrt{5}}{2} \} \end{bmatrix}, \begin{bmatrix} 9 - 4 \sqrt{5}, 1, \{1, \frac{1}{2} - \frac{\sqrt{5}}{2} \} \end{bmatrix} \right]
\]

We assign eigenvectors of \( \text{ATA} \) to vect1 and vect2, while the corresponding values we assign to \( \lambda_1 \) and \( \lambda_2 \). Up to the constant, the
eigenvectors are the same as previously found vec1 and vect

\[
\begin{align*}
vect1, vect2 &= \begin{bmatrix} 1 + \frac{\sqrt{5}}{2} \\ 2 \\ \frac{1}{2} \end{bmatrix}, \begin{bmatrix} 1 - \frac{\sqrt{5}}{2} \\ 2 \\ \frac{1}{2} \end{bmatrix} \\
\lambda1, \lambda2 &= L[1][1], L[2][1];
\end{align*}
\]

We can verify that indeed the eigenvectors and the eigenvalues have been assigned correctly:

\[
\begin{align*}
\text{map}(\text{simplify}, \text{evalm}((ATA - \lambda1) &* vect1)); \\
\text{map}(\text{simplify}, \text{evalm}((ATA - \lambda2) &* vect2));
\end{align*}
\]

Similar verification can be done in Cl(2,0) since one can view the 1-column eigenvector vect1 (resp. vect2) as one-column spinor spinor1 (resp. spinor2) in S.

\[
\begin{align*}
\text{spinorbasis}:=["f1", "f2"]:
\text{spinor1} &= \text{convert}(\text{vect1}, \text{spinor}, \text{spinorbasis});
\text{spinor2} &= \text{convert}(\text{vect2}, \text{spinor}, \text{spinorbasis});
\end{align*}
\]

Thus, spinors spinor1 and spinor2 are eigenspinors of pTp.

\[
\begin{align*}
\text{simplify}((pTp - \lambda1) &c \text{ spinor1}); \\
\text{simplify}((pTp - \lambda2) &c \text{ spinor2});
\end{align*}
\]

Let's assign the normalized eigenvectors to v1 and v2:
for i from 1 to nops(L) do
    for j from 1 to L[i][2] do
        k:=i+j-1;
        v||k:=normalize(L[i][3][j]); #assigning the eigenvectors to v1 and v2
    od; od;

Normalized spinors are s1 and s2:

sv||1:=convert(v1,spinor,spinorbasis);
sv||2:=convert(v2,spinor,spinorbasis);

\[
\begin{align*}
sv1 &:= \frac{2f_1}{\sqrt{10 + 2\sqrt{5}}} + \frac{2\left(\frac{1}{2} + \sqrt{\frac{5}{2}}\right)f_2}{\sqrt{10 + 2\sqrt{5}}} \\
sv2 &:= \frac{2f_1}{\sqrt{10 - 2\sqrt{5}}} + \frac{2\left(\frac{1}{2} - \sqrt{\frac{5}{2}}\right)f_2}{\sqrt{10 - 2\sqrt{5}}}
\end{align*}
\]

simplify((pTp - lambda1) &c sv1);
simplify((pTp - lambda2) &c sv2);

0
0

Let's define a 2 x 2 matrix V built out of these two normalized eigenvectors v1 and v2:

V:=radsimplify(augment(v||1..n)); #defining matrix V

\[
V := \begin{bmatrix}
\frac{2}{\sqrt{10 + 2\sqrt{5}}} & \frac{2}{\sqrt{10 - 2\sqrt{5}}} \\
\frac{1+\sqrt{5}}{\sqrt{10 + 2\sqrt{5}}} & \frac{1-\sqrt{5}}{\sqrt{10 - 2\sqrt{5}}}
\end{bmatrix}
\]

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

\[
pV := -\text{Id} \left( -\sqrt{10 + 2\sqrt{5}} + \sqrt{5} \sqrt{10 + 2\sqrt{5}} - 2\sqrt{10 - 2\sqrt{5}} \right)
\]

\[
+ \left( -\sqrt{10 + 2\sqrt{5}} + \sqrt{5} \sqrt{10 + 2\sqrt{5}} + 2\sqrt{10 - 2\sqrt{5}} \right) e1
\]

\[
= -\frac{1}{2\sqrt{10 + 2\sqrt{5}}} \left( 2\sqrt{10 + 2\sqrt{5}} \sqrt{10 - 2\sqrt{5}} \right)
\]
The images of V and its transpose t(V) in Cl(2,0) are the same as they are related by the reversion which stabilizes the 1-vectors e1 and e2.

It happens in Cl(2) that reversion corresponds to translation:

\[
pVt := \phi(t(V), M); \quad \# \text{finding image of } t(V) \text{ in } Cl(2,0)
\]

\[
pVt := \frac{-\sqrt{10 + 2\sqrt{5} + \sqrt{5} \sqrt{10 + 2\sqrt{5} - 2\sqrt{10 - 2\sqrt{5}}}}}{2\sqrt{10 + 2\sqrt{5} \sqrt{10 - 2\sqrt{5}}}}
\]

\[
+ \frac{\sqrt{10 - 2\sqrt{5} + \sqrt{5} \sqrt{10 - 2\sqrt{5} + 2\sqrt{10 + 2\sqrt{5}}}}}{2\sqrt{10 + 2\sqrt{5} \sqrt{10 - 2\sqrt{5}}}} e1
\]

\[
+ \frac{\sqrt{10 + 2\sqrt{5} - \sqrt{5} \sqrt{10 - 2\sqrt{5} + 2\sqrt{10 + 2\sqrt{5}}}}}{2\sqrt{10 + 2\sqrt{5} \sqrt{10 - 2\sqrt{5}}}} e2
\]

\[
+ \frac{\sqrt{10 - 2\sqrt{5} - \sqrt{5} \sqrt{10 - 2\sqrt{5} + 2\sqrt{10 + 2\sqrt{5}}}}}{2\sqrt{10 + 2\sqrt{5} \sqrt{10 - 2\sqrt{5}}}} e12
\]

The images of V and its transpose t(V) in Cl(2,0) are the same as they are related by the reversion which stabilizes the 1-vectors e1 and e2.

It happens in Cl(2) that reversion corresponds to translation:

\[
> \text{simplify}(pVt - \text{reversion}(pV));
\]

\[
0
\]

\[
> \text{radsimplify(evlm(t(V) \&* V));}\quad \# V \text{ is an orthogonal matrix}
\]

\[
> \text{radsimplify(evlm(V \&* t(V)));}\quad \# V \text{ is an orthogonal matrix}
\]

\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

\[
> \text{expand(simplify(cmul(pVt,pV))}, \text{expand(simplify(cmul(pV,pVt))});
\]

\[
Id, Id
\]

\[
> \text{AAT:=evalm(A \&* transpose(A)); } \quad \# \text{computing } AAT
\]

\[
AAT := \begin{bmatrix}
13 & 8 \\
8 & 5
\end{bmatrix}
\]
\[
ppT := \phi(A_{AT}, M); \quad \text{#finding image of } A_{AT} \text{ in } Cl(2, 0)
\]
\[
ppT := 9Id + 4e1 + 8e2
\]
\[
pol2 := \text{charpoly}(A_{AT}, \lambda); \quad \text{#finding the characteristic polynomial of } A_{AT}
\]
\[
pol2 := \lambda^2 - 18\lambda + 1
\]
\[
\text{climinpoly}(ppT);
\]
\[
x^2 - 18x + 1
\]
Notice that the characteristic polynomials of \(ATA\) and \(A_{AT}\) are the same; hence the eigenvalues will be the same. There is no need to look for them again. We define therefore the \textbf{singular values} \(\sigma_1\) and \(\sigma_2\) of \(A\) as follows:
\[
\sigma1 := \sqrt{\lambda_1}; \sigma2 := \sqrt{\lambda_2};
\]
\[
\sigma1 := \sqrt{5} + 2
\]
\[
\sigma2 := -2 + \sqrt{5}
\]
Notice that when we compute the eigenvectors \(u_1\) and \(u_2\) of \(A_{AT}\), not necessarily we will have \(A \times v_i = \sigma_i \times u_i, i=1,2\). This is because the choice of \(u_1\) and \(u_2\) usually is not consistent with the choice of \(v_1\) and \(v_2\). See below:
\[
L := \text{sort}([\text{eigenvects}(A_{AT})], \text{byeigenvals}); \quad \text{#finding eigenvectors of } A_{AT}
\]
\[
L := \left[ \left[ 9 + 4\sqrt{5}, 1, \left\{ \frac{1}{2} + \frac{\sqrt{5}}{2}, 1 \right\} \right], \left[ 9 - 4\sqrt{5}, 1, \left\{ \frac{1}{2} - \frac{\sqrt{5}}{2}, 1 \right\} \right] \right]
\]
\[
\text{for } i \text{ from 1 to nops(L) do}
\]
\[
\text{for } j \text{ from 1 to } L[i][2] \text{ do}
\]
\[
k := i+j-1;
\]
\[
\text{u}||k := \text{normalize}(L[i][3][j]) \text{ od od}; \quad \text{#assigning the eigenvectors to } u_1 \text{ and } u_2
\]
\[
u1 := \text{eval}(u1); u2 := \text{eval}(u2); \quad \text{#here are the eigenvectors of } A_{AT}
\]
\[
\text{chosen by Maple}
\]
\[
\begin{align*}
\mathbf{u}_1 &= \left[ \frac{2 \left( \frac{1}{2} + \frac{\sqrt{5}}{2} \right)}{\sqrt{10 + 2 \sqrt{5}}, \sqrt{10 + 2 \sqrt{5}}} \right] \\
\mathbf{u}_2 &= \left[ \frac{2 \left( \frac{1}{2} - \frac{\sqrt{5}}{2} \right)}{\sqrt{10 - 2 \sqrt{5}}, \sqrt{10 - 2 \sqrt{5}}} \right]
\end{align*}
\]

Notice that not necessarily \( \mathbf{A} \cdot \mathbf{v}_i = \sigma_i \mathbf{u}_i, i=1,2: \)

\[
\text{radsimplify}\left(\text{evalm}(\mathbf{A} \cdot \mathbf{v}_1 - \sigma_1 \mathbf{u}_1)\right); \quad \text{#this one checks out}
\]

\[
\left[ 0, 0 \right]
\]

\[
\text{radsimplify}\left(\text{evalm}(\mathbf{A} \cdot \mathbf{v}_2 - \sigma_2 \mathbf{u}_2)\right); \quad \text{#this one does not check out}
\]

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

Notice that the set \( \{\mathbf{u}_1, \mathbf{u}_2\} \) is an orthonormal set, but so is \( \{\mathbf{u}_1, -\mathbf{u}_2\} \). Let's re-define \( \mathbf{u}_2 \) as \(-\mathbf{u}_2\) and call it \( \mathbf{u}_{22} \). For completeness we redefine \( \mathbf{u}_1 \) as \( \mathbf{u}_1 \) and call it \( \mathbf{u}_{11} \):

\[
\text{u}_{11} := \text{evalm}(\mathbf{u}_1); \quad \text{u}_{22} := \text{evalm}(-\mathbf{u}_2);
\]

\[
\begin{align*}
\mathbf{u}_{11} &= \left[ \frac{2 \left( \frac{1}{2} + \frac{\sqrt{5}}{2} \right)}{\sqrt{10 + 2 \sqrt{5}}, \sqrt{10 + 2 \sqrt{5}}} \right] \\
\mathbf{u}_{22} &= \left[ \frac{-2 \left( \frac{1}{2} - \frac{\sqrt{5}}{2} \right)}{\sqrt{10 - 2 \sqrt{5}}, \sqrt{10 - 2 \sqrt{5}}} \right]
\end{align*}
\]

\[
\text{radsimplify}\left(\text{evalm}(\mathbf{A} \cdot \mathbf{v}_1 - \sigma_1 \mathbf{u}_{11})\right); \quad \text{#this one checks out as before}
\]

\[
\left[ 0, 0 \right]
\]

\[
\text{radsimplify}\left(\text{evalm}(\mathbf{A} \cdot \mathbf{v}_2 - \sigma_2 \mathbf{u}_{22})\right); \quad \text{#this one checks out now too}
\]

\[
\left[ 0, 0 \right]
\]
In the Clifford algebra $\text{Cl}(2,0)$, we have similar computation:

```plaintext
> su|1:=convert(u11,spinor,spinorbasis);
> su|2:=convert(u22,spinor,spinorbasis);

\[
\begin{align*}
\text{su1} &= \frac{2}{\sqrt{10 + 2 \sqrt{5}}} \left( \frac{1}{2} + \frac{\sqrt{5}}{2} \right) f1 + \frac{2}{\sqrt{10 + 2 \sqrt{5}}} f2, \\
\text{su2} &= -\frac{2}{\sqrt{10 - 2 \sqrt{5}}} \left( \frac{1}{2} - \frac{\sqrt{5}}{2} \right) f1 - \frac{2}{\sqrt{10 - 2 \sqrt{5}}} f2.
\end{align*}
\]
```

The above verification can be done in $\text{Cl}(2,0)$ too:

```plaintext
> simplify(p &c sv1-sigma1*su1);
> simplify(p &c sv2-sigma2*su2);

0
0
```

Now we may define a $2 \times 2$ matrix $U$: vectors $u11$ and $u22$ will form columns of $U$:

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

\[
U := \begin{bmatrix}
\frac{1 + \sqrt{5}}{\sqrt{10 + 2 \sqrt{5}}} & -\frac{1 + \sqrt{5}}{\sqrt{10 - 2 \sqrt{5}}} \\
\frac{\sqrt{10 + 2 \sqrt{5}}}{2} & \frac{\sqrt{10 - 2 \sqrt{5}}}{2}
\end{bmatrix}
\]
```

```plaintext
> pU:=phi(U,M);

\[
pU := \frac{\left( \sqrt{10 - 2 \sqrt{5}} + \sqrt{5} \sqrt{10 - 2 \sqrt{5}} - 2 \sqrt{10 + 2 \sqrt{5}} \right)}{2 \sqrt{10 + 2 \sqrt{5}} \sqrt{10 - 2 \sqrt{5}}} \text{Id} \\
+ \frac{\left( \sqrt{10 - 2 \sqrt{5}} + \sqrt{5} \sqrt{10 - 2 \sqrt{5}} + 2 \sqrt{10 + 2 \sqrt{5}} \right)}{2 \sqrt{10 + 2 \sqrt{5}} \sqrt{10 - 2 \sqrt{5}}} e1 \\
+ \frac{\left( -\sqrt{10 + 2 \sqrt{5}} + \sqrt{5} \sqrt{10 + 2 \sqrt{5}} + 2 \sqrt{10 - 2 \sqrt{5}} \right)}{2 \sqrt{10 - 2 \sqrt{5}} \sqrt{10 + 2 \sqrt{5}}} e2.
\]
```
\[ (+\sqrt{10 + 2\sqrt{5}} + \sqrt{5}\sqrt{10 + 2\sqrt{5}} - 2\sqrt{10 - 2\sqrt{5}}) e_{12} \]
\[ \frac{2\sqrt{10 + 2\sqrt{5}}}{\sqrt{10 - 2\sqrt{5}}} \]

\[ p_{Ut} := \phi(t(U), M); \]

\[ p_{Ut} := \frac{(-\sqrt{10 - 2\sqrt{5}} + \sqrt{5}\sqrt{10 - 2\sqrt{5}} - 2\sqrt{10 + 2\sqrt{5}})}{2\sqrt{10 + 2\sqrt{5}}\sqrt{10 - 2\sqrt{5}}} I_d \]

\[ + \frac{(-\sqrt{10 - 2\sqrt{5}} + \sqrt{5}\sqrt{10 - 2\sqrt{5}} + 2\sqrt{10 + 2\sqrt{5}})}{2\sqrt{10 + 2\sqrt{5}}\sqrt{10 - 2\sqrt{5}}} e_1 \]

\[ + \frac{(-\sqrt{10 + 2\sqrt{5}} + \sqrt{5}\sqrt{10 + 2\sqrt{5}} + 2\sqrt{10 - 2\sqrt{5}})}{2\sqrt{10 + 2\sqrt{5}}\sqrt{10 - 2\sqrt{5}}} e_2 \]

\[ - \frac{(-\sqrt{10 + 2\sqrt{5}} + \sqrt{5}\sqrt{10 + 2\sqrt{5}} - 2\sqrt{10 - 2\sqrt{5}})}{2\sqrt{10 + 2\sqrt{5}}\sqrt{10 - 2\sqrt{5}}} e_{12} \]

\[ > \text{simplify}(p_{U} - \text{reversion}(p_{Ut})); \]

\[ 0 \]

Let's check that \( U \) is an orthogonal matrix too:

\[ radsimplify(\text{evalm}(t(U) \&* U)); \text{#U is an orthogonal matrix} \]

\[ radsimplify(\text{evalm}(U \&* t(U))); \text{#U is an orthogonal matrix} \]

\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

\[ > \text{expand(simplify}(p_{Ut} \& c p_{U})), \text{expand(simplify}(p_{U} \& c p_{Ut})); \]

\[ \text{Id}, \text{Id} \]

Now, define matrices \( \Sigma \) using a procedure 'makediag' and then STS and SST. In this example matrices STS and SST are the same since \( \Sigma \) is a square diagonal matrix. Normally these matrices are different.

\[ > \Sigma := \text{makediag}(m, n, [\text{sigma1, sigma2}]); \]

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

\[ > \text{pSigma := phi}(\Sigma, M); \]
\[ p_{\Sigma} := \text{Id} \sqrt{5} + 2 \, e1 \]

> \text{STS} := \text{evalm}(\text{t}(\Sigma) \times \Sigma) ;

\[
\text{STS} := \begin{bmatrix}
(\sqrt{5} + 2)^2 & 0 \\
0 & (-2 + \sqrt{5})^2
\end{bmatrix}
\]

> \text{pSTS} := \text{phi}(\text{STS}, M, \text{FBgens}) ;

\[ p_{\text{STS}} := 9 \, \text{Id} + 4 \, e1 \sqrt{5} \]

> \text{SST} := \text{evalm}(\Sigma \times \text{t}(\Sigma)) ;

\[
\text{SST} := \begin{bmatrix}
(\sqrt{5} + 2)^2 & 0 \\
0 & (-2 + \sqrt{5})^2
\end{bmatrix}
\]

> \text{pSST} := \text{phi}(\text{SST}, M) ;

\[ p_{\text{SST}} := 9 \, \text{Id} + 4 \, e1 \sqrt{5} \]

We should be able to verify now these two factorizations of \( AAT \) and \( ATA \). Here is factorization of the matrix \( ATA \) and its corresponding Clifford number \( p_{Tp} \):

\[
\text{evalm}(\text{ATA}) = \text{radsimplify}(\text{evalm}(V \times \text{STS} \times \text{t}(V))) ; \quad \# \text{factorization of matrix } \text{ATA}
\]

\[
\begin{bmatrix}
5 & 8 \\
8 & 13
\end{bmatrix} = \begin{bmatrix}
5 & 8 \\
8 & 13
\end{bmatrix}
\]

> \text{pTp} = pV \& c pSTS \& c pVt ; \quad \# \text{factorization of Clifford number } p_{Tp}

\[ 9 \, \text{Id} - 4 \, e1 + 8 \, e2 = 9 \, \text{Id} - 4 \, e1 + 8 \, e2 \]

where

> \text{'pTp' = pTp ; 'pV' = pV ; 'pSTS' = pSTS ; 'pVt' = pVt ;}

\[ p_{Tp} = 9 \, \text{Id} - 4 \, e1 + 8 \, e2 \]

\[
p_{V} = -\text{Id} \left( \sqrt{10 + 2 \sqrt{5}} + \sqrt{5} \sqrt{10 + 2 \sqrt{5}} - 2 \sqrt{10 - 2 \sqrt{5}} \right) \\
\quad \div \left( 2 \sqrt{10 + 2 \sqrt{5}} \sqrt{10 - 2 \sqrt{5}} \right)
\]

\[
+ \left( \sqrt{10 + 2 \sqrt{5}} + \sqrt{5} \sqrt{10 + 2 \sqrt{5}} + 2 \sqrt{10 - 2 \sqrt{5}} \right) \, e1 \\
\quad \div \left( 2 \sqrt{10 - 2 \sqrt{5}} \sqrt{10 + 2 \sqrt{5}} \right)
\]
Here is factorization of the matrix $AAT$ and its corresponding Clifford number $ppT$:

\[
\begin{align*}
\text{evalm}(AAT) &= \text{radsimplify}(\text{evalm}(U \ast \text{SST} \ast t(U))) \; \# \text{factorization of matrix } AAT \\
\begin{bmatrix}
13 & 8 \\
8 & 5
\end{bmatrix} &= \begin{bmatrix}
13 & 8 \\
8 & 5
\end{bmatrix}
\end{align*}
\]

\[
ppT = pU \ast c \text{pSST} \ast c \text{pUt};
\]

\[
pV_t = -\frac{\text{Id} \left( -\sqrt{5} + 2 \sqrt{5} \sqrt{10 + 2 \sqrt{5} - 2 \sqrt{10 - 2 \sqrt{5}}} \right)}{2 \sqrt{10 - 2 \sqrt{5} \sqrt{10 + 2 \sqrt{5}}} \sqrt{10 - 2 \sqrt{5}}}
\]

\[
+pV_t = \frac{(-\sqrt{5} + 2 \sqrt{5} \sqrt{10 + 2 \sqrt{5} + 2 \sqrt{10 - 2 \sqrt{5}}}) \text{e} \cdot \text{Id} + 8 \text{e} \cdot 2}{2 \sqrt{10 - 2 \sqrt{5} \sqrt{10 + 2 \sqrt{5}}} \sqrt{10 - 2 \sqrt{5}}}
\]

\[
pU = \frac{(-\sqrt{5} + 2 \sqrt{5} \sqrt{10 + 2 \sqrt{5} + 2 \sqrt{10 - 2 \sqrt{5}}}) \text{e} \cdot \text{Id} + 8 \text{e} \cdot 2}{2 \sqrt{10 + 2 \sqrt{5} \sqrt{10 - 2 \sqrt{5}}}}
\]

\[
\]

\[
\]

\[
ppT = 9 \text{Id} + 4 \text{e} \cdot 1 + 8 \text{e} \cdot 2
\]

where

\[
'ppT'=ppT;'pU'=pU;'pSST'=pSST;'pUt'=pUt; \; \# \text{factorization of Clifford number } ppT
\]

\[
ppT = 9 \text{Id} + 4 \text{e} \cdot 1 + 8 \text{e} \cdot 2
\]
Finally, we verify the **Singular Value Decomposition** of A which is

\[
A = U \cdot \Sigma \cdot V^T
\]

as follows:

\[
\text{evalm}(A) = \text{radsimplify}(	ext{evalm}(U \cdot \Sigma \cdot t(V))); \quad \text{#matrix language}
\]

\[
\begin{bmatrix}
2 & 3 \\
1 & 2
\end{bmatrix} = \begin{bmatrix}
2 & 3 \\
1 & 2
\end{bmatrix}
\]

The **Singular Value Decomposition** of the Clifford number \(p\) is then

\[
p = pU \& c p\Sigma \& c pVt
\]

as follows:

\[
p = pU \& c p\Sigma \& c pVt; \quad \text{#Clifford algebra language}
\]

\[
2 Id + 2 e2 + e12 = 2 Id + 2 e2 + e12
\]
where

\[ P = 2 \text{Id} + 2 e_2 + e_{12} \]

\[ P_U = \frac{1}{2 \sqrt{10 + 2 \sqrt{5} \sqrt{10 - 2 \sqrt{5}}} \sqrt{10 - 2 \sqrt{5}}} \]
\[ \left( \sqrt{10 - 2 \sqrt{5}} + \sqrt{5} \sqrt{10 - 2 \sqrt{5}} + 2 \sqrt{10 + 2 \sqrt{5}} \right) \text{Id} \]
\[ + \frac{1}{2 \sqrt{10 + 2 \sqrt{5} \sqrt{10 - 2 \sqrt{5}}} \sqrt{10 - 2 \sqrt{5}}} \]
\[ \left( \sqrt{10 - 2 \sqrt{5}} + \sqrt{5} \sqrt{10 - 2 \sqrt{5}} + 2 \sqrt{10 + 2 \sqrt{5}} \right) e_1 \]
\[ + \frac{1}{2 \sqrt{10 + 2 \sqrt{5} \sqrt{10 - 2 \sqrt{5}}} \sqrt{10 - 2 \sqrt{5}}} \]
\[ \left( -\sqrt{10 + 2 \sqrt{5}} + \sqrt{5} \sqrt{10 + 2 \sqrt{5}} + 2 \sqrt{10 - 2 \sqrt{5}} \right) e_2 \]
\[ + \frac{1}{2 \sqrt{10 + 2 \sqrt{5} \sqrt{10 - 2 \sqrt{5}}} \sqrt{10 - 2 \sqrt{5}}} \]
\[ \left( -\sqrt{10 + 2 \sqrt{5}} + \sqrt{5} \sqrt{10 + 2 \sqrt{5}} + 2 \sqrt{10 - 2 \sqrt{5}} \right) e_{12} \]

\[ P_{\text{Sigma}} = \text{Id} \sqrt{5} + 2 e_1 \]

\[ P_{\text{Vt}} = -\frac{1}{2 \sqrt{10 + 2 \sqrt{5} \sqrt{10 - 2 \sqrt{5}}} \sqrt{10 - 2 \sqrt{5}}} \]
\[ \left( \sqrt{10 + 2 \sqrt{5}} + \sqrt{5} \sqrt{10 + 2 \sqrt{5}} - 2 \sqrt{10 - 2 \sqrt{5}} \right) \]
\[ + \frac{1}{2 \sqrt{10 + 2 \sqrt{5} \sqrt{10 - 2 \sqrt{5}}} \sqrt{10 - 2 \sqrt{5}}} \]
\[ \left( -\sqrt{10 + 2 \sqrt{5}} + \sqrt{5} \sqrt{10 + 2 \sqrt{5}} + 2 \sqrt{10 - 2 \sqrt{5}} \right) e_1 \]
\[ + \frac{1}{2 \sqrt{10 + 2 \sqrt{5} \sqrt{10 - 2 \sqrt{5}}} \sqrt{10 - 2 \sqrt{5}}} \]
\[ \left( \sqrt{10 - 2 \sqrt{5}} + \sqrt{5} \sqrt{10 - 2 \sqrt{5}} + 2 \sqrt{10 + 2 \sqrt{5}} \right) e_2 \]
\[ + \frac{1}{2 \sqrt{10 + 2 \sqrt{5} \sqrt{10 - 2 \sqrt{5}}} \sqrt{10 - 2 \sqrt{5}}} \]
\[ \left( \sqrt{10 - 2 \sqrt{5}} + \sqrt{5} \sqrt{10 - 2 \sqrt{5}} - 2 \sqrt{10 + 2 \sqrt{5}} \right) e_{12} \]

References


16. Finding generators of $H(2,F)$ in $\text{Cl}(B)$

(This work has been developed with Bertfried Fauser, Universitat Konstanz)

We will try to find realization of the generators $b_1$ and $b_2$ of the Hecke algebra $H(2,F)$ in the even part of $\text{Cl}(B)$ where $B$ is an appropriate bilinear form in a four dimensional real vector space $V$. These generators must satisfy three defining relations:

1. $b_i^2 = (1 - q)b_i + q$, $i = 1,2,$
2. $b_i b_j = b_j b_i$ when $|i - j| \geq 2$ (trivially satisfied in this case)
3. $b_i b_{i+1} b_i = b_{i+1} b_i b_{i+1}$

We will find not necessarily disjoint families of these generators and verify that they satisfy (1), (2), and (3).

```maple
restart:with(Clifford):with(Cliplus):
read "mydefs.m": #contains additional functions that simplify output

Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

We specify the dimension of $V$ to be 4 and define the bilinear form $B$ as follows:

```
for i from 1 to n-1 do
B[i+n,i+1]:=1:  # lambda:
B[i+n+1,i]:=q:  # q/lambda:
B[i,i+n+1]:=-1-q:  # -(lambda^2+q)/lambda:
B[i+1,i+n]:=-1-q:  # -(lambda^2+q)/lambda:
od:
B=evalm(B);

\[
B = \begin{bmatrix}
0 & 0 & q & -1-q \\
0 & 0 & -1-q & q \\
1 & 1 & 0 & 0 \\
q & 1 & 0 & 0
\end{bmatrix}
\]

We will find b1 and b2 in a subalgebra of Cl(B) spanned by the elements in
the list span which are generated, in our design, by suitable 1-vectors
\{e.i,e.(i+n)\} whose indices differ by n=2:

> for i from 1 to n do
g[i]:=&w(cat(e,i),cat(e,n+i));
g||i:=g[i]:
od:
gens:=\{seq(g||i,i=1..n)\};
span:=gens:
for xx in gens do
for yy in gens do
span:=span union cliterms(cmul(xx,yy));
od od;
span:=sort(convert(span,list),bygrade);

Warning, since B has been (re-)assigned, value of dim_V has been reduced by 'we
dge' to 4

gens := \{e13, e24\}

span := [Id, e13, e14, e23, e24, e1234]

We take all monomials in the span except the one of the highest order:

> x:=add(a[i]*span[i],i=1..nops(span)-1);

\[ x := a_1 Id + a_2 e13 + a_3 e14 + a_4 e23 + a_5 e24 \]

We now require that x must satisfy the quadratic equation (1) from above.
Maple returns, depending on a session, 8 or 10 families of solutions. Not all families are disjoint.
We will explicitly list all real solutions:

```plaintext
> sol := clisolve(CS(cmul(x, x) - (1-q)*x - q*Id), [seq(a[i], i=1..nops(span))]):
> sol_all := map(allvalues, sol):
> sol_real := remove(has, sol_all, I):
> nops(sol_real);
10

```
Let's verify that indeed $x_1, x_2, \ldots, $ etc. satisfy condition (1):

\[
\begin{align*}
x_7 & := (1 - a_2 + q + q^2 - q a_2 - a_2 q^2) \text{Id} + a_2 e_{13} + (-1 - q + a_2 + q a_2) e_{14} \\
x_8 & := (2 q - a_2 - q^2 - q a_2 - a_2 q^2) \text{Id} + a_2 e_{13} + (1 + q + a_2 + q a_2) e_{14} \\
x_9 & := \text{Id} \\
x_{10} & := -q \text{Id}
\end{align*}
\]

We pick one family of solutions:

\[
x := -(-a_2 - a_4)^2 + a_2^2 + a_4 a_2 + a_2^2 q - a_4 a_2 q^2 - 2 a_4^2 q - q a_2 - a_2 q^2 + q^2 a_2^2) \text{Id} \\
\quad -q a_4 - a_4 + a_2 \\
\quad + a_2 (-q a_4 - 1 + a_2 - q + q a_2) e_{14} \\
\quad + a_4 e_{23} + (q a_4 - 1 + a_2 - q + q a_2) a_4 e_{24} \\
\quad -q a_4 - a_4 + a_2
\]

Observe that the simplest solution $b_1$ is just given by the monomial $e_{13} = e_{1}w_{3}$:

\[
b_1 := \text{subs} \{ \{ a_2 = 1, a_4 = 0 \}, x \};
\]

We will now look for the second generator $b_2$ that will satisfy not only equation (1) but also, together with our choice for $b_1$, equation (3) from above. This can be accomplished by assigning the most general solution $x$ to $b_2$ first:

\[
b_2 := x;
\]

A quick check that $b_1$ and $b_2$ both satisfy (1):

\[
\begin{align*}
\text{map}(\text{evalb}, \text{seq}(\text{simplify}(\text{CS(cmul(b||i, b||i) - (1-q)*b||i*Id})))) = 0, \\
i = 1 \ldots n)\}
\end{align*}
\]
and then by solving (3) for the unknown parameters \(a[2]\) and \(a[4]\) in \(b2\):

\[
\text{sol} := \text{csolve}(\text{cmul}(b1, \text{cmul}(b2, b1)) - \text{cmul}(\text{cmul}(b2, b1), b2), [a[2], a[4]])
\]

\[
\begin{align*}
\text{sol} &= \{(a_2 = 1, a_4 = 0), (a_2 = \frac{a_4^2 q}{q a_4 + a_4 - 1}, a_4 = a_4)\}
\end{align*}
\]

Since the form of \(b2\) included \(b1\), we shouldn't be surprised that one of the solutions is \(b1\) itself (it is given by the choice \([a[4] = 0, a[2] = 1]\)). We are interested in the second choice which we substitute into \(x\) to find \(b2\):

\[
\text{b2} := \text{CS}(\text{subs}(\text{sol}[2], x))
\]

We verify that \(b1\) and \(b2\) satisfy the tangle equation (3):

\[
\text{map(evalb@factor@normal, [seq(CS(\text{cmul(\text{cmul(b|i, cat(b,i+1)), b|i}) - cmul(\text{cat(b,i+1), b|i), cat(b,i+1))})=0, i=1..n-1])]);}
\]

One specific choice for \(b2\) is:

\[
\text{b2'} = \text{subs}(a[4]=1, \text{b||2})
\]

\[
b2 = -\frac{(q^2 + 2 q a_4 - 1 + a_4) Id}{q a_4 + a_4 - 1} + \frac{a_4^2 q e13}{q a_4 + a_4 - 1} + a_4 e23 + \frac{a_4 q e14}{q a_4 + a_4 - 1} + e24
\]

Another, even simpler choice is just

\[
\text{b2'} = \text{subs}(a[4]=0, \text{b||2})
\]

\[
b2 = e24
\]

Guided by these solutions, we pick for \(b1, b2, ..., bn\), that represent generators \(t1, t2, ..., tn\) of \(H(n,F)\):

\[
\text{for i from 1 to n do}
\text{b[i]} := w(\text{cat(e,i)}, \text{cat(e,n+i))};
\text{b||i} := b[i];
\text{od};
\[
\text{seq}(b|i, i=1..n); \\
\text{seq}((\text{cliexpand}(b|i), i=1..n); \\

\quad e13, e24 \\
\quad \quad \quad \quad \quad \quad \quad -q \text{ &C}(Id) + (e1 \text{ &C} e3), -q \text{ &C}(Id) + (e2 \text{ &C} e4)
\]

\begin{verbatim}
> map(evalb@factor@normal,
    [seq(CS(cmul(cmul(b|i, cat(b,i+1)),b|i) -
        cmul(cmul(cat(b,i+1),b|i),cat(b,i+1)))=0, i=1..n-1)]);

[ true ]

> map(evalb,[seq(simplify(CS(cmul(b|i,b|i) - (1-q)*b|i-q*Id))=0,
    i=1..n)]);

[ true, true ]
\end{verbatim}

Reference


17. q-Young Clifford idempotents

(This work has been developed with Bertfried Fauser, Universitat Konstanz)

The distinguishing feature of Ablamowicz and Fauser [1] is that the full antisymmetrizers are constructed by reversing the corresponding full symmetrizers in the appropriate Clifford algebra. This contrasts the purely Hecke algebraic approach of King and Wybourne (KW) [2] who do not embed Hecke algebras into Clifford algebras.

\begin{verbatim}
> restart:with(Clifford):with(Cliplus):read "mydefs.m":

Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

> R||12:=CS(q*Id+b1);
C||12:=CS(Id-b1);
# test, if reversion(C.12) = R.12
CS(reversion(C||12) - R||12);
\end{verbatim}
Observe, that $R(12)$ is the reversion of $C(12)$, that is, $R(12) = C(12)\tilde{~}$ where $\tilde{~}$ is the reversion in the Clifford algebra $\cl_{1,1}$. The following is the verification that $C(12)$ and $R(12)$ are mutually annihilating almost idempotents.

$$R_{12} := q \text{Id} + e1we5$$

$$C_{12} := \text{Id} - e1we5$$

0

We look now for the intertwining element(s) $T$ in the algebraic span of the two Young operators $Y^{(2)}_{12}$ and $Y^{(11)}_{12}$, that is, element $T$ such that $T Y^{(2)}_{12} = Y^{(11)}_{12} T$, and we find none except
0.

> \( T := \text{CS}(a \cdot \text{Id} + b \cdot b_1); \)

\[
T := a \cdot \text{Id} + b \cdot \text{elwe5}
\]

> \( \text{clisolve}(\text{cmul}(T, Y_{2|12}) - \text{cmul}(Y_{1|12}, T), [a, b]); \)

\[
\{ \{ a = 0, b = 0 \} \}
\]

In the Clifford algebra \( \cl_{1,1} \), we can construct various primitive idempotents. Since in our construction the Hecke algebra \( H_{\bf F}(2, q) \) is a subalgebra of the even part \( \cl^+_{1,1} \) of the Clifford algebra \( \cl_{1,1} \), the idempotents of the Hecke algebra must be even Clifford elements. The following computation shows that the only two nontrivial mutually annihilating idempotents in the Hecke algebra are the Young operators.

> \( \text{xx} := \text{CS}(a \cdot \text{Id} + b \cdot b_1); \)

\[
\text{CS}(\text{cmul}(\text{xx}, \text{xx}) - \text{xx});
\]

\[
\text{sol} := [\text{solve}([\text{coeffs}(\%), \text{cliterms}(\%)], \{a, b\})];
\]

\[
\text{xx} := a \cdot \text{Id} + b \cdot \text{elwe5}
\]

\[
(a^2 + b^2 - q - a) \cdot \text{Id} + b \cdot (2 + b - 2 q) \cdot \text{elwe5}
\]

\[
\text{sol} := \left[ \{ a = 0, b = 0 \}, \{ a = 1, b = 0 \}, \{ a = \frac{1}{1 + q}, b = -\frac{1}{1 + q} \}, \{ a = \frac{q}{1 + q}, b = \frac{1}{1 + q} \} \right]
\]

> \( f_{1|1}, f_{1|2} := \text{subs}([\text{sol}[4], \text{xx}], \text{subs}([\text{sol}[3], \text{xx}]); \)

\[
f_{1|1}, f_{1|2} := \frac{q \cdot \text{Id}}{1 + q} + \frac{\text{elwe5}}{1 + q}, \frac{\text{Id}}{1 + q} - \frac{\text{elwe5}}{1 + q}
\]

```
f_{1|1} + f_{1|2} = CS(f_{1|1} + f_{1|2}), f_{1} & c f_{2} = CS(f_{1} & c f_{2}), f_{2} & c f_{1} = CS(f_{2} & c f_{1}); evalb(f_{1} & c f_{1} = f_{1}), evalb(f_{2} & c f_{2} = f_{2}); f_{1} + f_{2} = \text{Id}, f_{1} & c f_{2} = 0, f_{2} & c f_{1} = 0
```

true, true

> \( \text{bset} := [\text{Id}, b_1, b_2, b_{12}, b_{21}, b_{121}]; \)

\[
\text{XX} := \text{bexpand}(\text{add}(K[i] \cdot \text{bset}[i], i = 1 .. 6)); \]

\[
\text{sollist} := \text{clisolve}((\text{XX} + \text{reversion}(\text{XX}) - \text{Id}, [\text{seq}(K[i], i = 1 .. 6)]));
\]

\[
\text{out} := \text{bexpand}(\text{subs}([\text{sollist}[1], \text{XX}])
\]
This element belongs to a family parameterized by four real or complex parameters, and it is shown above in the Hecke algebra basis. Next we demanded that this most general element also satisfies (eq.2).

We found eight sets of solutions, each two of them related by the reversion. Two different kinds of solutions can be distinguished: one of them is probably parameterized by complex numbers satisfying two similar but different quadratic equations. Denoting their roots as $\alpha$ and $\beta$, we obtain the following representatives of all families again expressed in the Hecke algebra basis.
\[
\text{solve} := \left\{ K_1 = K_1, K_2 = K_5 q, K_3 = -K_5, K_4 = K_4, K_5 = K_5, K_6 = -\frac{K_5 q^2 - K_5 + 1}{q (1 + q)} \right\},
\]
\[
\left\{ K_1 = K_1, K_2 = K_5 q, K_3 = -K_5, K_4 = K_4, K_5 = K_5, K_6 = -\frac{K_5 q^2 - K_5 - 1}{q (1 + q)} \right\},
\]
\[
\left\{ K_1 = K_1, K_2 = -K_5, K_3 = K_5 q, K_4 = K_4, K_5 = K_5, K_6 = -\frac{K_5 q^2 - K_5 + 1}{q (1 + q)} \right\},
\]
\[
\left\{ K_1 = K_1, K_2 = -K_5, K_3 = K_5 q, K_4 = K_4, K_5 = K_5, K_6 = -\frac{K_5 q^2 - K_5 - 1}{q (1 + q)} \right\},
\]
\[
\left\{ K_1 = K_1, K_2 = \alpha, K_3 = K_3, K_4 = K_4, K_5 = K_5, K_6 = -\frac{\alpha K_3 + K_5^2 q}{q (K_3 + \alpha + K_5 - K_5 q)} \right\},
\]
\[
\left\{ K_1 = K_1, K_2 = \kappa, K_3 = K_3, K_4 = K_4, K_5 = K_5, K_6 = -\frac{\kappa K_3 + K_5^2 q}{q (K_3 + \kappa + K_5 - K_5 q)} \right\}
\]

\[
x_{\text{solve}} := \left[
\begin{array}{l}
\frac{\text{Id}}{1 + q} + K_5 q b1 - K_5 b2 - \frac{(K_5 - 1 + K_5 q^3 + q) b12}{q (1 + q)} + K_5 b21 - \frac{(K_5 q^2 - K_5 + 1) b121}{q (1 + q)}, \\
q \frac{\text{Id}}{1 + q} + K_5 q b1 - K_5 b2 - \frac{(K_5 + 1 + K_5 q^3 - q) b12}{q (1 + q)} + K_5 b21 - \frac{(K_5 q^2 - K_5 - 1) b121}{q (1 + q)}, \\
\frac{\text{Id}}{1 + q} - K_5 b1 + K_5 q b2 - \frac{(K_5 - 1 + K_5 q^3 + q) b12}{q (1 + q)} + K_5 b21 - \frac{(K_5 q^2 - K_5 + 1) b121}{q (1 + q)}, \\
q \frac{\text{Id}}{1 + q} - K_5 b1 + K_5 q b2 - \frac{(K_5 + 1 + K_5 q^3 - q) b12}{q (1 + q)} + K_5 b21 - \frac{(K_5 q^2 - K_5 - 1) b121}{q (1 + q)}, \\
\frac{\text{Id}}{1 + q} + \alpha b1 + K_3 b2 - \frac{(q + q^2 K_3 + q^2 \alpha - 1 - K_3 - \alpha + K_4 q^2 + K_5 q) b12}{q (1 + q)} + K_5 b21, \\
\frac{(1 + K_3 q + \alpha q + K_3 + \alpha) b121}{q (1 + q)} + q \frac{\text{Id}}{1 + q} + \kappa b1 + K_3 b2 \\
- \frac{(-q + q^2 K_3 + q^2 \kappa + 1 - K_3 - \kappa + K_5 q^2 + K_5 q) b12}{q (1 + q)} + K_5 b21, \\
\frac{(-1 + K_3 q + \kappa q + K_3 + \kappa) b121}{q (1 + q)}
\end{array}
\right]
\]
Since the representatives are related by the reversion, we seek four linearly independent elements. For example the first, second, third and sixth elements are linearly independent and are denoted as $f.1 \cdots f.4$.

```maple
> nops(findbasis({xxlist[1],xxlist[2],xxlist[3],xxlist[6]}));
f||1:=bexpand(xxlist[1]);
f||2:=bexpand(xxlist[2]);
f||3:=bexpand(xxlist[3]);
f||4:=bexpand(xxlist[6]);
```

Warning, unable to find a provably non-zero pivot

Warning, unable to find a provably non-zero pivot

Warning, unable to find a provably non-zero pivot

\[
\begin{align*}
f1 &:= \frac{\text{Id}}{1 + q} + K_5\frac{q b1 - K_5 b2 - (K_5 - 1 + K_5 q^3 + q) b12}{q (1 + q)} + K_5 b21 - \frac{(K_5 q^2 - K_5 + 1) b12}{q (1 + q)} \\
\end{align*}
\]

\[
\begin{align*}
f2 &:= \frac{q \text{Id}}{1 + q} + K_5\frac{q b1 - K_5 b2 - (K_5 + 1 + K_5 q^3 - q) b12}{q (1 + q)} + K_5 b21 - \frac{(K_5 q^2 - K_5 - 1) b12}{q (1 + q)} \\
\end{align*}
\]

\[
\begin{align*}
f3 &:= \frac{\text{Id}}{1 + q} - K_5 b1 + K_5\frac{q b2 - (K_5 - 1 + K_5 q^3 + q) b12}{q (1 + q)} + K_5 b21 - \frac{(K_5 q^2 - K_5 + 1) b12}{q (1 + q)} \\
\end{align*}
\]

\[
\begin{align*}
f4 &:= \frac{q \text{Id}}{1 + q} + \kappa b1 + K_3\frac{b2 - (-q + q^2 K_3 + q^2 \kappa + 1 - K_3 - \kappa + K_5 q^2 + K_5 q) b12}{q (1 + q)} + K_5 b21 \\
\end{align*}
\]

\[
\begin{align*}
\quad - \frac{(-1 + K_3 q + \kappa q + K_3 + \kappa) b12}{q (1 + q)} \\
\end{align*}
\]

```maple
> CS(f||1+reversion(f||1)-Id);
```

0

The $q$-Young operators we are seeking are now linear combinations of the above four elements. Due to the fact that the representation spaces which correspond to the antisymmetric and the symmetric Young operators respectively are one dimensional, they cannot have any free parameters beside $q$. The full symmetrizer can be given according to KW as the $q$ weighted sum of all six Hecke basis elements. However, due to our construction the full antisymmetrizer is defined as the reversion of the full symmetrizer, as it was done in dimension two.
Each of the above defined $f.i$ elements generates a three dimensional one sided ideal in the Hecke algebra, or, in other words, they pairwise decompose the unit element in $H_{\bf F}(3,q)$. Our objective is to further split each of those ideals into one and two dimensional vector spaces. The one dimensional vector spaces will be generated by the full symmetrizer and the full antisymmetrizer respectively. The two dimensional vector spaces will be generated by the mixed type Young operators. So each of the $f:i$ elements has to be a sum of a full (anti)symmetrizer and a mixed type Young operator. If we pick $f.1$ we notice that it must contain the full antisymmetrizer, because when the parameter $K_5$ is replaced with $1/(q+1)$ then $f:i$ reduces to an expression with alternating signs in the Hecke basis.

Therefore by subtracting the full antisymmetrizer from $f.1$ we find our first Young operator of the mixed type.
Y21132 := \frac{q \text{Id}}{q^2 + q + 1} + \frac{(K_5 q^4 + 2 K_5 q^3 + K_5 q + 2 K_5 q^2 + 1) b1}{2 q + 2 q^2 + q^3 + 1}
- \frac{(K_5 q^3 + 2 K_5 q^2 + 2 K_5 q - 1 + K_5) b2}{2 q + 2 q^2 + q^3 + 1}
- \frac{(K_5 q^5 + K_5 q^4 + K_5 q^3 + q^3 + K_5 q^2 + K_5 q + q - 1 + K_5) b12}{2 q + 2 q^2 + q^3 + 1}
+ \frac{q (2 q + 2 q^2 + q^3 + 1)}{2 q + 2 q^2 + q^3 + 1}
- \frac{(K_5 q^3 + 2 K_5 q^2 + 2 K_5 q - 1 + K_5) b21}{2 q + 2 q^2 + q^3 + 1}
- \frac{(K_5 q^4 + K_5 q^3 + q^2 - K_5 q + 1 - K_5) b121}{2 q + 2 q^2 + q^3 + 1}
+ \frac{q (q^2 + q + 1) (1 + q)}{2 q + 2 q^2 + q^3 + 1}

outl := 0

> ybas := [Y3||123, Y111||123, Y21||123, reversion(Y21||123)]:
ybasb := \text{map}(\text{bexpand}, ybas);

ybasb := \begin{bmatrix}
\frac{q^3 \text{Id}}{2 q + 2 q^2 + q^3 + 1} + \frac{q^2 b1}{(q^2 + q + 1) (1 + q)} + \frac{q^2 b2}{(q^2 + q + 1) (1 + q)} + \frac{\text{Id}}{b12}
+ \frac{2 q + 2 q^2 + q^3 + 1}{b1}
- \frac{(q^2 + q + 1) (1 + q)}{b2}
+ \frac{2 q + 2 q^2 + q^3 + 1}{b21}
+ \frac{(q^2 + q + 1) (1 + q)}{b121}
+ \frac{q \text{Id}}{q (K_5 q^3 + 2 K_5 q + 2 K_5 q^2 + q + K_5) b1}
+ \frac{(K_5 q^3 + 2 K_5 q - q^2 + 2 K_5 q^2 + K_5) b2}{2 q + 2 q^2 + q^3 + 1}
\end{bmatrix}
\[
\begin{align*}
&\left( K_5 q^5 + K_5 q^4 + K_5 q^3 + q^3 - q^2 + K_5 q^2 + K_5 q - 1 + K_5 \right) b12 \\
&+ \frac{q \left( 2 q + 2 q^2 + q^3 + 1 \right)}{2 q + 2 q^2 + q^3 + 1}
\end{align*}
\]
\[
\begin{align*}
&\left( K_5 q^3 + 2 K_5 q + 2 K_5 q^2 + q + K_5 \right) b21 \\
&+ \frac{q \left( K_5 q^4 + K_5 q^3 + q^2 - K_5 q + 1 - K_5 \right) b21}{q \left( q^2 + q + 1 \right)}
\end{align*}
\]
\[
\begin{align*}
&\left( K_5 q^3 + 2 K_5 q^2 + 2 K_5 q - 1 + K_5 \right) b2 \\
&- \frac{q \left( K_5 q^5 + K_5 q^4 + K_5 q^3 + q^3 + K_5 q^2 + K_5 q + q - 1 + K_5 \right) b12}{q \left( 2 q + 2 q^2 + q^3 + 1 \right)}
\end{align*}
\]
\[
\begin{align*}
&\left( K_5 q^3 + 2 K_5 q^2 + 2 K_5 q - 1 + K_5 \right) b21 \\
&+ \frac{q \left( K_5 q^4 + K_5 q^3 + q^2 - K_5 q + 1 - K_5 \right) b21}{q \left( q^2 + q + 1 \right)}
\end{align*}
\]

> \text{CS}(Y3||123+Y111||123+Y21||123+reversion(Y21||123));

\[
\begin{align*}
&\text{Id}
\end{align*}
\]

> \text{L}:= 'L': i:= 'i':

map(bexpand, bset); \# Hecke basis

\[
\begin{align*}
YY:= \text{bexpand}(add(P[i]*bset[i], i=1..6)): \\
YY:= \text{bexpand}(op(clisolve(YY+reversion(YY)-Id, YY)));
\end{align*}
\]

\[
\begin{align*}
R||13:= \text{bexpand}(op(clisolve(Y21||132-cmul(YY,f||1), YY)));
\end{align*}
\]

\[
\begin{align*}
\left[ \text{Id}, b1, b2, b12, b21, b121 \right]
\end{align*}
\]

\[
\begin{align*}
YY := \left( \frac{-1}{2} P_2 + \frac{1}{2} P_2 q - \frac{1}{2} P_3 + \frac{1}{2} P_3 q + \frac{1}{2} P_6 q^2 - \frac{1}{2} P_6 q \right) \text{Id} + P_2 b1 + P_3 b2 \\
+ \left( -P_5 + P_6 q - P_6 \right) b12 + P_5 b21 + P_6 b121
\end{align*}
\]

\[
\begin{align*}
R13 := \frac{q \text{Id}}{1+q} + \frac{\left( P_5 q^3 + P_5 q^2 + P_5 q + 1 \right) b1}{q^2 + q + 1} - \frac{\left( P_5 q^2 + P_5 q - q + P_5 \right) b2}{q^2 + q + 1}
\end{align*}
\]

\[
\begin{align*}
- \frac{\left( P_5 q^2 + P_5 q^3 + q^4 P_5 + P_5 q + q^5 P_5 + q^2 - q + P_5 \right) b12}{2 q + 2 q^2 + q^3 + 1} + P_5 b21 \\
- \frac{\left( q^4 P_5 + P_5 q^3 - P_5 q + q - P_5 \right) b121}{2 q + 2 q^2 + q^3 + 1}
\end{align*}
\]

> \text{CS}(Y21||132-cmul(R||13,f||1)); \# Y21.132 = R.13 & c f.1
Obviously, we have the following relations

\[
C(13) := R(13) \tilde{~}
\]
and

\[
C(12) := f.1,
\]
\[
R(12) := C(12) \tilde{~}.
\]

For more information see [1].

References


18. Computations with octonions

\[
\Phi, \text{associator, commutator, def.omultable, o.conjug, oinv, omul, omultable, onorm, oversion, purevectorpart, realpart}
\]

The following is the default octonionic multiplication table:

\[
\text{omultable();}
\]
\[
\begin{bmatrix}
-1d & e4 & e7 & -e2 & e6 & -e5 & -e3 \\
-e4 & -1d & e5 & e1 & -e3 & e7 & -e6 \\
-e7 & -e5 & -1d & e6 & e2 & -e4 & e1 \\
e2 & -e1 & -e6 & -1d & e7 & e3 & -e5 \\
e6 & e3 & -e2 & -e7 & -1d & e1 & e4 \\
e5 & -e7 & e4 & -e3 & -e1 & -1d & e2 \\
e3 & e6 & -e1 & e5 & -e4 & -e2 & -1d
\end{bmatrix}
\]

\[o1, o2, o3 := 1 - 2*e1 + 3*e3 + e4 - e6 + e7, 2 + e3 - 4*e6 + e7, 2 - 3*e1 + e5 - e7;\]

\[o1, o2, o3 := 1 - 2*e1 + 3*e3 + e4 - e6 + e7, 2 + e3 - 4*e6 + e7, 2 - 3*e1 + e5 - e7;\]

\> `type(o1, octonion), type(o2, octonion);`

 Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

`true, true`

\> `omul(o1, o2);`

\[-6 \Id - 2*e1 + 5*e3 + 13*e4 - 7*e6 + e7 - 9*e5 + 3*e2\]

Octonionic multiplication is not commutative:

\> `o1 \&o o2; o2 \&o o1; evalb(%%=%);`

\[-6 \Id - 2*e1 + 5*e3 + 13*e4 - 7*e6 + e7 - 9*e5 + 3*e2\]

\[-6 \Id + 9*e3 - 5*e6 + 5*e7 - 6*e1 + 9*e5 - 9*e4 - 3*e2\]

`false`

We show now that it is not associative either:

\> `(o1 \&o o2) \&o o3; o1 \&o (o2 \&o o3); %%-%;`

\[-8 \Id + 16*e1 - 21*e2 + 2*e3 + 43*e4 + 10*e5 - 40*e6 + 36*e7\]

\[-8 \Id + 6*e1 + 47*e2 + 8*e3 + 5*e4 - 18*e5 - 38*e6 + 38*e7\]

\[10*e1 - 68*e2 - 6*e3 + 38*e4 + 28*e5 - 2*e6 - 2*e7\]

The difference between (o1 \&o o2) \&o o3 and o1 \&o (o2 \&o o3) is measured by an associator, or see Octonion:-associator

\> `associator(o1, o2, o3);`

\[10*e1 - 68*e2 - 6*e3 + 38*e4 + 28*e5 - 2*e6 - 2*e7\]

The difference between o1 \&o o2 and o2 \&o o1 is measured by a
commutator, or see Octonion:-commutator

> commutator(o1,o2);

$$4e1 - 4e3 + 22e4 - 2e6 - 4e7 - 18e5 + 6e2$$

Changing octonionic multiplication table via Fano triples:

> restart:with(Clifford):with(Octonion);

[\text{\Phi, associator, commutator, def\_omultable, o\_conj, oinv, omul, omultable, onorm, oversion, purevectorpart, realpart}]

> _default_Fano_triples; \# default Fano triples

[[1, 3, 7], [1, 2, 4], [1, 5, 6], [2, 3, 5], [2, 6, 7], [3, 4, 6], [4, 5, 7]]

For example, the first list implies the following about \{e1,e3,e7\}:

> 'omul(e1,e3)'=omul(e1,e3);'omul(e3,e7)'=omul(e3,e7);'omul(e7,e1)'=omul(e7,e1);

\[
\begin{align*}
\text{omul}(e1, e3) &= e7 \\
\text{omul}(e3, e7) &= e1 \\
\text{omul}(e7, e1) &= e3
\end{align*}
\]

and

> 'omul(e3,e1)'=omul(e3,e1);'omul(e7,e3)'=omul(e7,e3);'omul(e1,e7)'=omul(e1,e7);

\[
\begin{align*}
\text{omul}(e3, e1) &= -e7 \\
\text{omul}(e7, e3) &= -e1 \\
\text{omul}(e1, e7) &= -e3
\end{align*}
\]

and so on.

However, the following is another valid list of Fano triples:

> new_Fano_triples:=[[6, 2, 5], [6, 3, 4], [6, 7, 1], [2, 3, 7], [3, 1, 5], [2, 4, 1], [4, 5, 7]];

\[
\begin{align*}
\text{new\_Fano\_triples} & := [[6, 2, 5], [6, 3, 4], [6, 7, 1], [2, 3, 7], [3, 1, 5], [2, 4, 1], [4, 5, 7]]
\end{align*}
\]

> type(new_Fano_triples,Fano_triples);
true

while the following is not:

> another_Fano_triples:=[[4,2,5],[6,3,4],[6,7,1],[2,3,7],[3,1,5],[2,4,1],[4,5,7]]

another_Fano_triples :=
[[4, 2, 5], [6, 3, 4], [6, 7, 1], [2, 3, 7], [3, 1, 5], [2, 4, 1], [4, 5, 7]]

> type(another_Fano_triples,Fano_triples);

false

The reason is that '4' appears in four lists. Octonionic multiplication table can now be changed:

> def_omultable(new_Fano_triples): #re-define now the octonionic multiplication table.

> 'omul(e3,e1)'='omul(e3,e1);'omul(e7,e3)'='omul(e7,e3);'omul(e1,e7)'='omul(e1,e7);

omul(e3, e1) = e5
omul(e7, e3) = -e2
omul(e1, e7) = -e6

19. Rotations with quaternions

> restart:bench:=time():with(Clifford):_prolevel:=true:eval(makealiases(3));

B:=linalg[diag](1,1,1):_quatbasis;


[[Id, e32, e13, e21], {Maple has assigned qi:=-e2we3, qj:=e1we3, qk:=-e1we2 }]

Example 1: Rotations in coordinate planes.

Let's define first unit quaternions responsible for the rotations in the coordinate planes. These are counter-clockwise rotations when looking down the rotation axis.

> q12:=(cos(alpha/2)+sin(alpha/2)*'qk'; #rotation in the xy-plane
\[q_{12} := \cos \left(\frac{\alpha}{2}\right) + \sin \left(\frac{\alpha}{2}\right) q_k\]

> \(q_{12i} := \text{qinv}(q_{12});\)  

#inverse of \(q_{12}\)

Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include \&C and \&C[K]. Type ?cliprod for help.

\[q_{12i} := \cos \left(\frac{\alpha}{2}\right) - \sin \left(\frac{\alpha}{2}\right) q_k\]

> \(q_{13} := \cos(\beta/2) + \sin(\beta/2) 'qj';\)  

#rotation in the xz-plane

\[q_{13} := \cos \left(\frac{\beta}{2}\right) + \sin \left(\frac{\beta}{2}\right) q_j\]

> \(q_{13i} := \text{qinv}(q_{13});\)  

#inverse of \(q_{13}\)

\[q_{13i} := \cos \left(\frac{\beta}{2}\right) - \sin \left(\frac{\beta}{2}\right) q_j\]

> \(q_{23} := \cos(\gamma/2) + \sin(\gamma/2) 'qi';\)  

#rotation in the yz-plane

\[q_{23} := \cos \left(\frac{\gamma}{2}\right) + \sin \left(\frac{\gamma}{2}\right) q_i\]

> \(q_{23i} := \text{qinv}(q_{23});\)  

#inverse of \(q_{23}\)

\[q_{23i} := \cos \left(\frac{\gamma}{2}\right) - \sin \left(\frac{\gamma}{2}\right) q_i\]

> \(\text{qnorm}(q_{12}), \text{qnorm}(q_{13}), \text{qnorm}(q_{23});\)  

# these are unit quaternions

1, 1, 1

> \(\text{type}(q_{12, \text{quaternion}}), \text{type}(q_{13, \text{quaternion}}), \text{type}(q_{23, \text{quaternion}});\)

true, true, true

> \(q_{12} &\& q_{12i}, q_{13} &\& q_{13i}, q_{23} &\& q_{23i};\)

1, 1, 1

Notice that to rotate by an angle \(n\cdot\alpha\) it is enough to find the \(n\)-th Clifford power of the appropriate quaternion:

> \(q_{12} &\& q_{12};\)  

#rotation by the angle \(2\cdot\alpha\)

\[\cos(\alpha) + \sin(\alpha) q_k\]

> \(q_{12} &\& q_{12} &\& q_{12};\)  

#rotation by the angle \(3\cdot\alpha\)
\[
\cos\left(\frac{3\alpha}{2}\right) + \sin\left(\frac{3\alpha}{2}\right) qk
\]

> q12 &q q12 &q q12 &q q12; #rotation by the angle 4*alpha

\[
\cos(2\alpha) + \sin(2\alpha) qk
\]

> q12 &q q12 &q q12 &q q12 &q q12; #rotation by the angle 5*alpha

\[
\cos\left(\frac{5\alpha}{2}\right) + \sin\left(\frac{5\alpha}{2}\right) qk
\]

Example 2: Let's see how basis vectors \(\{e_1, e_2, e_3\}\) are rotated by the quaternions \(q_{12}\), \(q_{13}\), and \(q_{23}\) in the coordinate planes \(xy-, xz-,\) and \(yz-\) respectively.

> e11:=rot3d(e1,q12); #rotation of \(e_1\) by the quaternion \(q_{12}\)

\[
e_{11} := \cos(\alpha) e_1 + \sin(\alpha) e_2
\]

> e22:=rot3d(e2,q12); #rotation of \(e_2\) by the quaternion \(q_{12}\)

\[
e_{22} := -\sin(\alpha) e_1 + \cos(\alpha) e_2
\]

> e33:=rot3d(e3,q12); #rotation of \(e_3\) by the quaternion \(q_{12}\)

\[
e_{33} := e_3
\]

We can also compose rotations. For example, let's follow rotation \(q_{12}\) with the rotation \(q_{13}\):

> e111:=rot3d(e1,q13 &q q12); #rotation of \(e_1\) by the quaternion \(q_{13} \& q q_{12}\)

\[
e_{111} := \sin(\alpha) e_2 + e_1 \cos(\beta) \cos(\alpha) - e_3 \sin(\beta) \cos(\alpha)
\]

> e222:=rot3d(e2,q13 &q q12); #rotation of \(e_2\) by the quaternion \(q_{13} \& q q_{12}\)

\[
e_{222} := \cos(\alpha) e_2 - e_1 \sin(\alpha) \cos(\beta) + e_3 \sin(\beta) \sin(\alpha)
\]

> e333:=rot3d(e3,q13 &q q12); #rotation of \(e_3\) by the quaternion \(q_{13} \& q q_{12}\)

\[
e_{333} := \sin(\beta) e_1 + \cos(\beta) e_3
\]

It is not difficult to see that the sets \(\{e_{11}, e_{22}, e_{33}\}\) and \(\{e_{111}, e_{222}, e_{333}\}\) are orthonormal. Let's verify that assertion for the latter set:

> A:=array(1..3,1..3,[]):

for i from 1 to 3 do for j from 1 to 3 do
A[i,j]:=scalarpart((e||i||i||i &c e||j||j||j + e||j||j||j &c
\[ e^{i\|i\|i/2} \text{ end do end do; } \]

\[ \text{print(A);} \]

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

We can check orientation of the set \{e_{111}, e_{222}, e_{333}\} by checking that the wedge products \(e_{111} \& w e_{222}\), \(e_{222} \& w e_{333}\), and \(e_{333} \& w e_{111}\) are dual to the vectors \(e_{333}\), \(e_{111}\), and \(e_{222}\) respectively. Remembering that \(e_1 w e_2\) is dual to \(e_3\), \(e_2 w e_3\) is dual to \(e_1\), and \(e_3 w e_1\) is dual to \(e_2\), we have the following dual pars:

\[ e_{111} \& w e_{222}, e_{333}; \]

\[
\cos(\beta) e_1 e_2 + \sin(\beta) e_2 e_3, \sin(\beta) e_1 + \cos(\beta) e_3
\]

\[ e_{222} \& w e_{333}, e_{111}; \]

\[
-\sin(\beta) \cos(\alpha) e_1 e_2 - \sin(\alpha) e_1 e_3 + \cos(\beta) \cos(\alpha) e_2 e_3, \\
\sin(\alpha) e_2 + e_1 \cos(\beta) \cos(\alpha) - e_3 \sin(\beta) \cos(\alpha)
\]

\[ e_{333} \& w e_{111}, e_{222}; \]

\[
\sin(\beta) \sin(\alpha) e_1 e_2 - \cos(\alpha) e_1 e_3 - \sin(\alpha) \cos(\beta) e_2 e_3, \\
\cos(\alpha) e_2 - e_1 \sin(\alpha) \cos(\beta) + e_3 \sin(\beta) \sin(\alpha)
\]

Another way to check that the sets \(\{e_1, e_2, e_3\}\), \(\{e_{11}, e_{22}, e_{33}\}\), and \(\{e_{111}, e_{222}, e_{333}\}\) have the same orientation is to verify that their wedge products are all equal:

\[ \text{wedge(e}_{1}, e_{2}, e_{3}), \text{wedge(e}_{11}, e_{22}, e_{33}), \text{wedge(e}_{111}, e_{222}, e_{333}); \]

\[ e_{123}, e_{123}, e_{123} \]

**Example 3**: Let's see now how these basis rotations in the coordinate planes act on an arbitrary vector.

\[ v := a e_1 + b e_2 + c e_3; \]

\[ v := a e_1 + b e_2 + c e_3 \]

The length of \(v\) can be found as follows:

\[ \text{vlength := sqrt(scalarpart(v \& c Q v));} \]

\[ \text{vlength :=} \sqrt{a^2 + b^2 + c^2} \]

Certainly, rotations do not change length. For example, let's rotate \(v\) by \(q_{13} \& q_{12}\):

\[ v_{123} := \text{rot3d(v, q_{13} \& q_{12});} \# \text{rotation } q_{12} \text{ followed by } q_{13} \]
Thus, the length of $v_{123}$ is the same as the length of $v$. However, rotations do not commute. We will show that by applying quaternion $q_{12}$ & $q_{13}$ to $v$ and comparing with $q_{123}$:

\[
\begin{align*}
v_{132} &= \text{rot3d}(v, q_{12} \& q_{13}); \quad \# \text{rotation } q_{13} \text{ followed by } q_{12} \\
v_{132} &= (c \sin(\beta) - b \sin(\alpha) \cos(\beta) - a \cos(\beta) \cos(\alpha)) e\dot{1} + e_2 (c \cos(\beta) - a \sin(\beta)) e_3 \\
&\quad + (a \sin(\beta) - b \sin(\alpha) + a \cos(\beta) \cos(\alpha)) e\dot{2} + e_3 (c \sin(\beta) + b \cos(\beta)) e_3 + e_2 (c \cos(\beta) - a \sin(\beta)) e_3 \\
c\text{licollect}(v_{123} - v_{132}); \\
&\quad -(c \sin(\beta) + b \sin(\alpha) \cos(\beta) - a \cos(\beta) \cos(\alpha)) e\dot{1} \\
&\quad - e_2 (c \sin(\beta) + b \cos(\beta)) e_2 - e_3 (c \cos(\beta) - a \sin(\beta)) e_3 \\
\end{align*}
\]

As it can be seen, $v_{123} \neq v_{132}$.

**Example 4:** Counter-clockwise rotation by an angle 'alpha' around the given vector 'axis'.

Suppose we want to rotate vector 'v' around some given unit vector 'axis' by the angle 'alpha'. This rotation will be counter-clockwise when looking down the axis towards the origin $(0,0,0)$ of the coordinate system.

\[
\begin{align*}
\text{quatbasis;} \\
[[I\dot{d}, e_{32}, e_{13}, e_{21}], \{\text{Maple has assigned } q_i := -e_2 we_3, q_j := e_1 we_3, q_k := -e_1 we_2 \}] \\
\text{axis} := (a_1 e_1 + a_2 e_2 + a_3 e_3); \text{alias(}N = \sqrt{\text{scalarpart} (\text{axis} \& c\text{axis})}) ; \\
\text{axis} := a_1 e_1 + a_2 e_2 + a_3 e_3 \\
\text{axis} := \text{axis}/N; \# \text{we normalize axis} \\
\text{axis} := \frac{a_1 e_1 + a_2 e_2 + a_3 e_3}{\sqrt{a_1^2 + a_2^2 + a_3^2}} \\
\text{qaxis} := \text{subs}\{(e_3 = -e_1 w_2, e_1 = -e_2 w_3, e_2 = e_1 w_3), \text{axis}\}; \# \text{define dual quaternion}
\end{align*}
\]
Notice that to find the dual quaternion 'qaxis' we could have also multiplied the axis vector 'axis' by the unit pseudoscalar \( e_1 e_2 e_3 \) (unit volume element) on the right as follows:

\[
qaxis := \text{axis} \& c (-e_1 e_2 e_3);
\]

\[
qaxis := \frac{-a_3 e_{12}}{\sqrt{a_1^2 + a_2^2 + a_3^2}} + \frac{a_2 e_{13}}{\sqrt{a_1^2 + a_2^2 + a_3^2}} - \frac{a_1 e_{23}}{\sqrt{a_1^2 + a_2^2 + a_3^2}}
\]

\[
qaxis := \text{qdisplay}(qaxis);
\]

\[
qaxis := \frac{a_3 qk}{\sqrt{a_1^2 + a_2^2 + a_3^2}} + \frac{a_2 qj}{\sqrt{a_1^2 + a_2^2 + a_3^2}} + \frac{a_1 qi}{\sqrt{a_1^2 + a_2^2 + a_3^2}}
\]

\[
qrot := \cos(\alpha/2) + \sin(\alpha/2) * qaxis;
\]

Let's define various rotation quaternions:

\[
q100 := \text{subs}\{\{a_1=1, a_2=0, a_3=0\}, qrot\}; \quad \text{# rotation about the axis } (1,0,0)
\]

\[
q100 := \cos\left(\frac{\alpha}{2}\right) - \sin\left(\frac{\alpha}{2}\right) e_{23}
\]

\[
q010 := \text{subs}\{\{a_1=0, a_2=1, a_3=0\}, qrot\}; \quad \text{# rotation about the axis } (0,1,0)
\]

\[
q010 := \cos\left(\frac{\alpha}{2}\right) + \sin\left(\frac{\alpha}{2}\right) e_{13}
\]

\[
q001 := \text{subs}\{\{a_1=0, a_2=0, a_3=1\}, qrot\}; \quad \text{# rotation about the axis } (0,0,1)
\]

\[
q001 := -\sin\left(\frac{\alpha}{2}\right) e_{12} + \cos\left(\frac{\alpha}{2}\right)
\]

\[
q101 := \text{subs}\{\{a_1=1, a_2=0, a_3=1\}, qrot\}; \quad \text{# rotation about the axis } (1,0,1)
\]

\[
q101 := \cos\left(\frac{\alpha}{2}\right) + \sin\left(\frac{\alpha}{2}\right) \left( -\frac{\sqrt{2} e_{12}}{2} - \frac{\sqrt{2} e_{23}}{2} \right)
\]

\[
qaxis := \frac{-a_1 e_{23} + a_2 e_{13} - a_3 e_{12}}{\sqrt{a_1^2 + a_2^2 + a_3^2}}
\]
Let's try to rotate first a basis vector $e_1$ around various axes listed above by some angle $\alpha$. After we find a general formula for the components of the rotated $e_1$, we will verify it for various angles.

```plaintext
> v1:='v1':v2:='v2':v3:='v3':v:=e1;
  
> vnew:=rot3d(v,q100); #the result of the rotation around the (100) axis
  
> eval(subs(alpha=Pi/2,vnew));
  
> vnew:=rot3d(v,q010); #the result of the rotation around the (010) axis
  
> eval(subs(alpha=Pi/2,vnew));
  
> vnew:=rot3d(v,q001); #the result of the rotation around the (001) axis
```

```plaintext
q011 := subs({a1=0,a2=1,a3=1},qrot);#rotation about the axis (0,1,1)

\[
q011 := \cos\left(\frac{\alpha}{2}\right) + \sin\left(\frac{\alpha}{2}\right)\left(\frac{\sqrt{2} e_{12}}{2} + \frac{\sqrt{2} e_{13}}{2}\right)
\]

q110 := subs({a1=1,a2=1,a3=0},qrot);#rotation about the axis (1,1,0)

\[
q110 := \cos\left(\frac{\alpha}{2}\right) + \sin\left(\frac{\alpha}{2}\right)\left(\frac{\sqrt{2} e_{13}}{2} - \frac{\sqrt{2} e_{23}}{2}\right)
\]

q111 := subs({a1=1,a2=1,a3=1},qrot);#rotation about the axis (1,1,1)

\[
q111 := \cos\left(\frac{\alpha}{2}\right) + \sin\left(\frac{\alpha}{2}\right)\left(\frac{\sqrt{3} e_{12}}{3} + \frac{\sqrt{3} e_{13}}{3} - \frac{\sqrt{3} e_{23}}{3}\right)
\]

```
\begin{verbatim}
> eval(subs(alpha=Pi/2,vnew));

\text{e2}

> vnew:=rot3d(v,q101); \# the result of the rotation around the (101) axis

\text{vnew} := \frac{1}{2} (\cos(\alpha) + 1) e1 + \frac{1}{2} \sin(\alpha) \sqrt{2} e2 - \frac{1}{2} (-1 + \cos(\alpha)) e3

> eval(subs(alpha=Pi/2,vnew));

\frac{e1}{2} + \frac{\sqrt{2} e2}{2} + \frac{e3}{2}

> vnew:=rot3d(v,q011); \# the result of the rotation around the (011) axis

\text{vnew} := \cos(\alpha) e1 + \frac{1}{2} \sin(\alpha) \sqrt{2} e2 - \frac{1}{2} \sin(\alpha) \sqrt{2} e3

> eval(subs(alpha=Pi/2,vnew));

\frac{\sqrt{2} e2}{2} - \frac{\sqrt{2} e3}{2}

> vnew:=rot3d(v,q110); \# the result of the rotation around the (110) axis

\text{vnew} := \frac{1}{2} (\cos(\alpha) + 1) e1 - \frac{1}{2} (-1 + \cos(\alpha)) e2 - \frac{1}{2} \sin(\alpha) \sqrt{2} e3

> eval(subs(alpha=Pi/2,vnew));

\frac{e1}{2} + \frac{e2}{2} - \frac{\sqrt{2} e3}{2}

> vnew:=rot3d(v,q111); \# the result of the rotation around the (111) axis

\text{vnew} :=
\frac{1}{3} e1 (2 \cos(\alpha) + 1) - \frac{1}{3} (-\sin(\alpha) \sqrt{3} - 1 + \cos(\alpha)) e2 - \frac{1}{3} (\sin(\alpha) \sqrt{3} - 1 + \cos(\alpha)) e3

> eval(subs(alpha=Pi/2,vnew));
\end{verbatim}
\[
\begin{align*}
\frac{e_1}{3} - \frac{(\sqrt{3} - 1) e_2}{3} - \frac{(\sqrt{3} - 1) e_3}{3}
\end{align*}
\]

\[
\text{eval(subs(alpha=Pi,vnew));}
\]

\[
- \frac{e_1}{3} + \frac{2 e_2}{3} + \frac{2 e_3}{3}
\]

**Example 5:** Finally, we derive a general formula for a rotation of an arbitrary vector 'v' around an arbitrary axis called 'axis' and by an arbitrary angle 'alpha'. Recall that the general quaternion defined in terms of the 'axis' components was defined in Example 3 above as 'qrot'.

\[
\text{alias(N=N)}:
\]

\[
v1 := 'v1': v2 := 'v2': v3 := 'v3': v := v1*e1 + v2*e2 + v3*e3; \quad \# \text{an arbitrary vector}
\]

\[
\text{qrot} = qrot;
\]

\[
v := v1*e1 + v2*e2 + v3*e3
\]

\[
\text{qnew} := \text{rot3d}(v, qrot);
\]

\[
\begin{align*}
\text{qnew} := & - e_1 (a_3 v_3 a_2^2 \sin(\alpha) - a_3 v_2 a_3^2 \sin(\alpha) - a_2 v_3 a_3^2 \sin(\alpha) + a_3 v_2 a_1^2 \sin(\alpha)) \\
& - a_2 v_3 a_1^2 \sin(\alpha) - a_3 v_3 \sin(\alpha) - a_3^2 v_1 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha) \\
& - a_2^2 v_1 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha) - a_1^2 v_2 \sqrt{a_1^2 + a_2^2 + a_3^2} \\
& + a_3 a_1 v_3 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha) - a_3 a_1 v_3 \sqrt{a_1^2 + a_2^2 + a_3^2} \\
& + a_2 a_1 v_2 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha) - a_2 a_1 v_2 \sqrt{a_1^2 + a_2^2 + a_3^2} \right) \left( a_1^2 + a_2^2 + a_3^2 \right)^{3/2} \\
& + e_2 \left( a_3 a_2 v_3 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha) - a_3 a_2 v_3 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha) + a_3 v_1 a_1^2 \sin(\alpha) \\
& + a_3 v_1 a_2^2 \sin(\alpha) + a_3^3 v_2 \sin(\alpha) + a_1^2 v_2 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha) \\
& + a_3^2 v_2 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha) + a_2 a_1 v_1 \sqrt{a_1^2 + a_2^2 + a_3^2} \\
& - a_2 a_1 v_1 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha) - a_1^3 v_3 \sin(\alpha) - a_1 v_3 a_2^2 \sin(\alpha) - a_1 v_3 a_3^2 \sin(\alpha) \\
& + v_2 \sqrt{a_1^2 + a_2^2 + a_3^2} (a_1^2 + a_2^2 + a_3^2) \right) + e_3 \left( a_3^2 v_3 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha) + a_2 a_1 v_3 \sqrt{a_1^2 + a_2^2 + a_3^2} \\
& + a_2^2 v_3 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha) + a_1^3 v_2 \sin(\alpha) + a_1 v_2 a_2^2 \sin(\alpha) + a_1 v_2 a_3^2 \sin(\alpha) \\
& + a_1 v_3 a_2^2 \sin(\alpha) - a_2 v_1 a_1^2 \sin(\alpha) - a_2 v_1 a_3^2 \sin(\alpha) \\
& + a_3 a_2 v_2 \sqrt{a_1^2 + a_2^2 + a_3^2} - a_3 a_2 v_2 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha)
\end{align*}
\]
\[ \frac{1}{4} + a_3 a_1 v_1 \sqrt{a_1^2 + a_2^2 + a_3^2} - a_3 a_1 v_1 \sqrt{a_1^2 + a_2^2 + a_3^2} \cos(\alpha) \sqrt{a_1^2 + a_2^2 + a_3^2}^{3/2} \]

For example, let's rotate \( v = e_1 + 2e_2 + 3e_3 \) around the axis \((1,2,3)\) by any angle \( \alpha \). Certainly, since the vector \( v \) is on the axis of rotation, it should not change:

\[
\text{subs}\{v_1=1,v_2=2,v_3=3,a_1=1,a_2=2,a_3=3\},qnew;\]

\[ e_1 + 2e_2 + 3e_3 \]

Finally, let's rotate \( v = e_1 - 2e_2 + 4e_3 \) around the axis \((2,-3,4)\) by an angle \( \alpha = \pi/4 \).

\[
\text{eval}\left(\text{subs}\{v_1=1,v_2=-2,v_3=4,a_1=2,a_2=-3,a_3=4,\alpha=\pi/4\},qnew\right);\]

\[
e_1 \left( \frac{58\sqrt{2}}{2} + \frac{19\sqrt{29}\sqrt{2}}{2} - 48\sqrt{29} \right) \sqrt{29} \]

\[
- \frac{\sqrt{29}}{841} \left( 72\sqrt{29} + 7\sqrt{29} \sqrt{2} - 58\sqrt{2} \right) \\
+ \frac{e_2 \sqrt{29}}{841} \left( 96\sqrt{29} + 10\sqrt{29} \sqrt{2} - 29\sqrt{2} \right) \\
+ \frac{e_3 \sqrt{29}}{841} \left( 96\sqrt{29} + 10\sqrt{29} \sqrt{2} - 29\sqrt{2} \right) \]

\[
\text{printf}("\text{Worksheet took }%f\text{ seconds to compute on Intel Pentium M 2.13 GHz 2GB RAM with Win XP Professional}\n",\text{time()}\text{-bench});\]

Worksheet took 6.281000 seconds to compute on Intel Pentium M 2.13 GHz 2GB RAM with Win XP Professional

\[ \]

20. Sample help pages

CLIFFORD's Main Page:

+++++++++++++++++++++++++++

CLIFFORD - A Maple 11 Package for Clifford Algebras with "Bigebra"

(Version 10 with environmental variables given by CLIFFORD_ENV())

Last revised: February 18, 2008 (Source file: clifford_M11_10.mws)

Copyright 1995-2008 by Rafal Ablamowicz (*) and Bertfried Fauser ($)

(*) Department of Mathematics, Box 5054

Tennessee Technological University, Cookeville, TN 38505
Calling Sequence:
function(args)
Clifford:-function](args)

Description:

- This package allows the user to perform a variety of algebraic computations in the Clifford algebra \( Cl(V,B) \) on a real vector space \( V \) endowed with a bilinear form \( B \). The dimension of \( V \) is must be between 1 and 9, while the bilinear form \( B \) may be undefined, numeric, symbolic, symmetric, or antisymmetric. In fact, in most computations the form \( B \) doesn't even have to be assigned in which case coefficients of \( B \) appear in the Maple output. To use any function from the package, load the package using the command with(Clifford).

- Upon loading the package, default dimension of \( V \) is set to 9 (maximum) by assigning value of 9 to the global variable \( \text{dim V} \) in the initialization procedure Clifford:-setup. Then, computations can be immediately performed in \( Cl(V,B) \) where
B is unspecified and \( \dim(V) = 9 \). This and other environmental variables can be displayed by the procedure `Clifford:-CLIFFORD_ENV`. User can change the value of \( \dim_V \) by assigning a square matrix to \( B \) of size \( n \) by \( n \), in which case \( \dim_V \) is assigned value of \( n \) and 'CLIFFORD' performs computations in \( \Cl(V,B) \) assuming that \( \dim(V) = n \). Another way to change value of \( \dim_V \) is to simply assign a positive integer \( p \) to \( \dim_V \) such that \( 1 \leq p \leq 9 \). The value of \( \dim_V \) is used to determine whether an index larger than \( \dim_V \) has been entered by the user with parsing done by a procedure `Clifford:-cliparse`. Also, when symbolic or mixed, symbolic and numeric, indices are used in Grassmann basis monomials, procedure wedge, which computes the wedge product, suppresses monomial terms in the output of grade higher than \( \dim_V \).

- In this version, user can select one of two procedures that compute Clifford product of any two multivectors \( p_1, p_2 \) in \( \Cl(B) \): `cmulRS` and `cmulNUM`. Both procedures require that an additional parameter \( K \) of type 'name', 'symbol' that is passed on to them from `cmul` as in `cmul(p1,p2,K)`. Then, the contraction is internally computed with respect to this parameter. Procedure 'cmulRS' uses Rota-Stein cliffordization and is faster for symbolic \( B \). Procedure 'cmulNUM' uses Chevalley's recursive definition of the Clifford product in \( \Cl(B) \) and is faster when \( B \) is numeric. In addition, user can supply his/her own procedure in place of these two under a generic name `cmul_user_defined`. The current choice is displayed by `Clifford:-CLIFFORD_ENV` and is stored in a global variable `_default_Clifford_product`. See also procedure `Clifford:-useproduct` which is used to select the internal Clifford product.

- In comparison with release 4, several procedures in this release have been revised to achieve greater speed. Some have obtained new optional features, and some have been revised to save memory. In addition, a new package 'Bigebra' has been added to supplement 'CLIFFORD'. Other supplementary packages that extend abilities of 'CLIFFORD' are 'Cliplus', 'GTP', and 'Octonion'.

- 'CLIFFORD' performs computations in \( \Cl(V,B) \) using Grassmann basis monomials written as \( \Id, e_1, e_1\wedge e_2, e_1\wedge e_3\wedge e_5, \) etc., with \( \Id \) denoting the identity element in \( \Cl(V,B) \). Thus, the fundamental types include `Clifford:-`type/clisscalar`, `Clifford:-`type/clibasmon`, `Clifford:-`type/climon` and `Clifford:-`type/clipolynom`. Computations in Clifford basis with Clifford monomials written as \( \Id, e_1, e_2, e_1 \&C e_2, \&C(e_1,e_2,e_3), \) etc. can be performed upon loading the supplementary package 'Cliplus'. Note that the indices of basis monomials range from 1 to 9; symbolic indices as strings of length 1 are also allowed, and can be mixed with numeric indices.

- Clifford product in \( \Cl(V,B) \) is given by the procedure `Clifford:-cmul`, while the wedge product is given by the procedure `Clifford:-wedge`. For example, to multiply two (Grassmann) monomials or polynomials in \( \Cl(V,B) \) enter `cmul(p1,p2)` or `p1 \&c p2`, where \( p_1, p_2 \) are of one of these types: `Clifford:-`type/clibasmon`, `Clifford:-`type/climon` or `Clifford:-`type/clipolynom`. Likewise, the wedge product is entered as `wedge(p1,p2)` or `p1 \&w p2`. Multiple inputs are also allowed, e.g.,
cmul(p1,p2,p3,p4), or, &c(p1,p2,p3,p4), or, wedge(p1,p2,p3,p4), or &w(p1,p2,p3,p4). It is also possible to compute with Clifford matrices of 
\texttt{Clifford:-`type/climatrix`}, and apply various products to the matrix elements. See procedure \texttt{Clifford:-rmulm} for more help.

• 'CLIFFORD' uses Claude Chevalley's definition of the Clifford product or one based on Rota-Stein cliffordization technique. For more information, see page for \texttt{Clifford:-cmul}. All computations performed internally by 'CLIFFORD' are purely symbolic, that is, they do not involve any matrices. While matrices in the regular or spinor representations of Cl(V,B) could be used, 'CLIFFORD' performs operations on strings such as e1we2, e1we3we4, etc. that is, on Grassmann monomials whose type is \texttt{Clifford:-`type/clibasmon`}.

• For more information on a particular 'function' see its help page. In a Maple worksheet, type \texttt{?Clifford, function, or, ?function, at the Maple prompt. It is also possible to access all help pages for 'CLIFFORD' via the Maple browser by typing \texttt{?Clifford} and following links in the browser.

• This version includes a global variable \_prolevel which is set to its default value of 'false' by the 'CLIFFORD' initialization file \texttt{Clifford:-setup}. When \_prolevel=false, all type checking is done and user's input is parsed for errors. This can take considerable time though. User can set \_prolevel to 'true' in which case procedure 'cliparse' (see \texttt{Clifford:-cliparse}) always returns 'true' and no parsing of user input is done.

• There are additional packages available below which extend the 'CLIFFORD' package. They need to be loaded separately and each requires that 'CLIFFORD' be loaded first. Help pages for these packages are available also. For example:

- 'Cliplus' is a collection of additional procedures; it allows, for example, to use a Clifford basis instead of a Grassmann basis,
- 'Bigebra' is a collection of procedures to compute with coproducts,
- 'GTP' - Graded Tensor Package allows for certain computations in graded tensor products of Clifford algebras,
- 'Octonion' allows for computations with octonions considered as para-vectors in Cl(0,7),

This is how to load the main package and the supplementary packages:

\begin{verbatim}
> restart:with(Clifford);

[ &m, Bsignment, CLIFFORD_ENV, Kfield, LC, LCQ, RC, RCQ, RHnumber, adfmatrix, all_sigs, beta_minus, beta_plus, buildm, bygrade, c_conjug, cba, cis, cdfmatrix, cexp, cexpQ, cinv, clibilinear, clicollect,clidata, cilinear, climinpoly, cliparse, cliparse, cliremove,
\end{verbatim}
clisolve, clisort, cliterms, cmul, cmulNUM, cmulQ, cmulRS, cmulgen, cocycle, commutingelements, conjugation, ddfmatrix, diagonalize, displayid, extract, factoridempotent, find1str, findbasis, gradeinv, init, isVahlenmatrix, isproduct, makealiases, makeclibasmon, matKrepr, maxgrade, maxindex, mdfmatrix, minimalideal, ord, permsign, pseudodet, q_conjug, qdisplay, qinv, qmul, qnorm, rd_clibasmon, rd_climon, rd_clipolynom, reorder, reversion, rmulm, rot3d, scalarpart, sexp, specify_constants, spinorKbasis, spinorKrepr, squaremodf, subs_clipolynom, useproduct, vectorpart, version, wedge, wexp ]

> version();

+++++++++++++++++++++++++++++++++++++++++++++++++++++

CLIFFORD - A Maple 13 Package for Clifford Algebras with "Bigebra"

(Version 13.3 with environmental variables given by CLIFFORD_ENV())

Last revised: June 17, 2012 (Source file: clifford_M13_3.mws)

Copyright 1995-2012 by Rafal Ablamowicz (*) and Bertfried Fauser ($)

(*) Department of Mathematics, Box 5054
Tennessee Technological University, Cookeville, TN 38505
tel: USA (931) 372-3569, fax: USA (931) 372-6353
rablamowicz@tntech.edu

http://math.tntech.edu/rafal/

($) Universit"at Konstanz, Fachbereich Physik, Fach M678
78457 Konstanz, Germany
Bertfried.Fauser@uni-konstanz.de

http://kaluza.physik.uni-konstanz.de/~fauser/

If you are a Clifford algebra pro, assign 'true' to '_prolevel' and see
how much faster your computations will be! But watch your syntax!
Use 'useproduct' to change value of _default_Clifford_product in Cl(B) from
cmulRS when B is symbolic to cmulNUM when B is numeric. Type ?cmul for help.

Type CLIFFORD_ENV() to see current values of environmental variables.

cmulNUM, cmulRS, cmulWalsh3, and reorder are thread-safe in this version

+++++++++++++This is CLIFFORD version 13.3+++++++++++++

To load 'CLIFFORD' and the 'GTP' package type:

> restart:with(Clifford):with(GTP);

[cmulB, gbasis, gcollect, gprod, grade, gradedprod, tensorrank]

To load 'CLIFFORD' and the 'Octonion' package, type:

> restart:with(Clifford):with(Octonion);

[Φ, associator, commutator, def_omultable, o_conjug, oinv, omul, omultable, onorm, oversion, purevectorpart, realpart]

To load 'CLIFFORD' and the 'Clplus' package, type:

> restart:with(Clifford):with(Clplus);

Clplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

[LCbig, RCbig, clibasis, clieval, cliexpand, climul, clirev, dotteddbasis, dwedge, makeclialias]

To load 'CLIFFORD' and the 'Bigebra' package, type:

> restart:with(Clifford):with(Bigebra);

Increase verbosity by infolevel[`function`=val -- use online help > ?Bigebra[help]

[&cco, &gco, &gco_d, &gco_pl, &map, &v, EV, VERSION, bracket, contract, drop_t, eps, gantipode, gco_unit, gswitch, hodge, linop, linop2, lists2mat, lists2mat2, make_BI_Id, mapop, mapop2, meet, op2mat, op2mat2, pairing, peek, poke, remove_eq, switch, tcollect, tsolve1]

> CLIFFORD_ENV();

`>>> Global variables describing a default system defined in Clifford:-setup are now available and have these values: <<<'

`************* Start *************
_processor_type = "Intel (R) Core (TM) 2 Duo CPU 2.19 GHz, 2.9 GB RAM"
_operating_system = "Win XP Professional (SP3)"
`************* End *************`
Global variables defined in Clifford:-setup are now available and have these values: <<<

`************* Start *************`

dim_V = 9
_defClifford_product = Clifford:-cmulRS
_prolevel = false
_shortcut_in_minimalideal = true
_shortcut_in_Kfield = true
_shortcut_in_spinorKbasis = true
_shortcut_in_spinorKrepr = true
_warnings_flag = true
_scalartypes = {'^', RootOf, complex, indexed, numeric, constant, function, mathfunc, rational}

_quatbasis = [[Id, e3we2, e1we3, e2we1], {'Maple has assigned qi:-e2we3, qj:=e1we3, qk:=--elwe2'}]

`************* End *************`

Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

Global variables defined in Cliplus:-setup are now available and have these values: <<<

`************* Start *************`

macro(Clifford:-cmul = climul)
macro(Clifford:-cmulQ = climul)
macro(`&c` = climul)
macro(`&cQ` = climul)
macro(Clifford:-reversion = clirev)
macro(Clifford:-LC = LCbig)
macro(Clifford:-RC = RCbig)

`Warning, new definitions for type/climon and type/clipolynom now include &C`

`************* End *************`

There are no new global variables or macros in GTP yet. <<<

`************* Start *************`

`************* End *************`

Global variables defined in Octonion:-setup are now available and have these values: <<<

`************* Start *************`

_octbasis = [Id, e1, e2, e3, e4, e5, e6, e7]
_pureoctbasis = [e1, e2, e3, e4, e5, e6, e7]
_default_Fano_triples = [[1, 3, 7], [1, 2, 4], [1, 5, 6], [2, 3, 5], [2, 6, 7], [3, 4, 6], [4, 5, 7]]
_default_squares = [-Id, -Id, -Id, -Id, -Id, -Id, -Id]
_defClifford_product = Clifford:-cmulNUM
Listing of available procedures in 'CLIFFORD':

\&c - infix form of Clifford product
\&cm - multiplication of Clifford matrices with the Clifford product \&c applied to matrix entries
\&cQm - multiplication of Clifford matrices with the Clifford product \&cQ applied to matrix entries
\&om - multiplication of Clifford matrices with the octonionic product omul (ampersand form \&om) applied to matrix entries
\&q - quaternionic multiplication
\&qm - multiplication of quaternionic matrices with the quaternion product \&q applied to matrix entries
\&rm - multiplication of Clifford matrices with a user-defined product \&r applied to matrix entries
\&r - a possibly non-commutative user-defined product applied to entries in Cl(B)
\&w - infix form of the exterior product
\&wm - multiplication of matrices with the wedge product \&w applied to matrix entries
Bsignature - to display signature of the current form Q
CLIFFORD_ENV - to display current values of environmental variables in the worksheet
Kfield - to find a field K of the spinor representation of Cl(Q)
LC - left contraction in Cl(B)
LCQ - left contraction in Cl(Q)
RC - right contraction in Cl(B)
RCQ - right contraction in Cl(Q)
RHnumber - Radon-Hurwitz function
adfmatrix - add matrices over double fields
all_sigs - display all signatures of Clifford algebras
beta_minus - scalar product in S = Cl(Q)f related to reversion
beta_plus - scalar product in S = Cl(Q)f related to conjugation
buildm - finding a left-regular representation
bygrade - sorting by grade
c_conjug - complex conjugation in complexified Cl(B)
cbasis - create a canonical (Grassmann) basis in Cl(B)
cdfmatrix - create a matrix over a double field from two matrices in a list
cexp - Clifford exponentiation in Cl(B)
cexpQ - Clifford exponentiation in Cl(Q)
cinv - inverse in Clifford algebra Cl(B)
cilibilinear - making a user-defined procedure bilinear or multilinear
clicollect - collecting with respect to Clifford basis monomials
clidata - to find basis information about Cl(Q) for the given signature
cilinear - making a user-defined procedure linear
climinpoly - finding minimal polynomial of any Clifford polynomial
cliparse - checking user input for syntax errors
cliremove - deleting some parts of a string (internal use mostly)
clisolve - solves equations in Cl(V,B)
clisort - sorting with respect to Clifford basis monomials
cliterms - finding Clifford basis monomials in the given polynomial
cmul - Clifford multiplication in Cl(B) with ampersand form &c
cmulgen - generic (dummy) Clifford product
cmulNUM - Clifford multiplication of two multivectors using Chevalley’s recursive definition
cmulQ - Clifford multiplication in Cl(Q) with ampersand form &cQ
cmulRS - Clifford multiplication of two multivectors using Rota-Stein cliffordization technique
cmul_user_defined - dummy (user defined) Clifford product
cocycle - finding element x in Cl(V,B) that connects two given polynomials
commutingelements - finding commuting basis monomials that square to 1
conjugation - conjugation in Cl(B)
convert/dfmatrix - converts a list of two matrices into a matrix with entries in a double field
convert/mlist - converting matrix entries to a list
convert/str_to_int - converting numerical strings to digits
ddfmatrix - decompose a matrix over a double field into a list of two matrices
diagonalize - procedure to diagonalize a matrix
displayid - display algebra unit as Id in the given polynomial
extract - extract indices from the given basis monomial (mostly internal use)
factoridempotent - factor an idempotent into a product of simple idempotents
find1str - finds locations of string of length one in another string (mostly internal use)
findbasis - finding a basis in a linear span
gradeinv - grade involution in Cl(B)
intro - this help page
isproduct - checking if the given polynomial is a product of 1-vectors
isVahlenmatrix - procedure checking if a 2 x 2 matrix is a Vahlen matrix
makealiases - alias basis monomials for faster entry
makeclibasmon - makes a Grassmann monomial with the given indices
matKrepr - matrix representation of Cl(Q) using pre-stored matrices
maxgrade - finding the largest grade in the given polynomial
maxindex - finding the largest index in the given polynomial
mdfmatrix - multiply two matrices over a double field
minimalideal - finding a real basis in a left or right minimal ideal
ord - finding location of indices in a monomial string (internal use)
permsign - computes sign of a permutation
pseudodet - computing a pseudodeterminant of a matrix
g_conjug - quaternion conjugation
gdisplay - displaying quaternion in the standard basis \{1, qi, qj, qk\}
ginv - quaternionic inverse
gmul - quaternionic multiplication with the infix form &g
gnorm - quaternionic norm
rd_clibasmon - random Grassmann basis monomial
rd_climon - random Grassmann monomial
rd_clipolynom - random Grassmann polynomial
reorder - reorder monomial indices in a polynomial or monomial
reversion - reversion in Cl(B)
rmulm - various ways to multiply matrices with entries in Cl(B)
rot3d - rotation in 3 dimensions with quaternions
scalarp - finding the scalar part in a polynomial
setup - initialization procedure for the 'CLIFFORD' package
sexp - exponentiation in Cl(B) modulo minimal polynomial
specify_constants - to specify new constants by the user
spinorKbasis - finding a basis in a minimal (spinor) ideal over a field K
spinorKrepr - finding a matrix over K in spinor representation of Cl(Q)
squaremodf - computing the square of an element in an ideal Cl(Q)f or fCl(Q) modulo f
subs_clipolynom - substitution of a Clifford polynomial into a polynomial in one variable
useproduct - procedure that allows user to switch between these Clifford products: cmulRS, cmulNUM, cmulgen, or cmul_user_defined
scalarp - finding a k-vector part in a polynomial
version - display current version of the package and copyrights
wedge - wedge/exterior multiplication in Cl(B)
wexp - wedge/exterior exponentiation in Cl(B)

To see code of any of these procedures, follow the example below.

**Types in 'CLIFFORD':**

type/antisymmatrix - an antisymmetric matrix
type/clibasmon - a basis monomial in Cl(B)
type/climatrix - matrix of with entries in Cl(B)
type/climon - a monomial in Cl(B)
type/clipolynom - a polynomial in Cl(B)
type/cliprod - basis monomial in a Clifford basis, used in the 'Cliplus' package described below
type/cliscalar - a scalar in Cl(B)
type/dfmatrix - matrix over a double field
type/diagmatrix - a diagonal matrix
type/evenelement - even element in Cl(B)
type/fieldelement - a basis element in Cl(Q) or fCl(Q) which is in some field K
type/gencomplex - a generalized complex number
type/genquatbasis - a generalized quaternion basis
type/genquaternion - a generalized quaternion
type/idempotent - an idempotent element in Cl(B)
type/nilpotent - a nilpotent element in Cl(B)
type/oddelement - an odd element in Cl(B)
type/primitiveidemp - a primitive idempotent in Cl(B)
type/purequatbasis - a basis for pure quaternions
type/quaternion - quaternion type
type/symmatrix - a symmetric matrix
type/tensorprod - place holders of tensor products

'GTP' - Graded Tensor Product of Clifford Algebras

Procedures in 'GTP':
cmulB - Clifford product in Cl(B) when B is specified as the third argument (may be also used in the package 'CLIFFORD' upon reading in the 'GTP' package)
gbasis - standard basis in a graded tensor product
gcollect - collection or terms in a graded tensor product
gprod - product of basis monomials in a graded tensor product
grade - grade of an element in a graded tensor product
gradedprod - product of polynomials in a graded tensor product
setup - initialization procedure for 'GTP' (no help page)
tensorrank - gives rank of a tensor
&t - tensor product

Types in 'GTP':
type/gradedeven - even element in GTP
type/gradedodd - odd element in GTP
type/gradedmonom - basis monomial in GTP
type/gradedpolynom - a polynomial in GTP

'Octonion' - Package for Computations with Octonions

Procedures in 'Octonion':
Phi - associative 3-form
&om - octonionic multiplication applied to matrix entries (procedure from 'CLIFFORD')
associator - associator of three octonions
commutator - commutator of two octonions
def_omultable - procedure to define an octonion multiplication table different than a default one
o_conjug - octonionic conjugation
oinv - octonionic inverse
omul - octonionic non-associative multiplication with infix form `&o`
omultable - procedure that displays current octonionic multiplication table
onorm - octonionic norm
oversion - displays current version of the package
purevectorpart - pure vector part of an octonion
realpart - real part of an octonion
setup - initialization procedure for 'Octonion'

Types in 'Octonion':

type/Fano_triples - type Fano triples [needed to define octonionic multiplication using the Fano plane concepts]
type/octonion - type 'octonion'

'Cliplus' - Package that extends certain functions from 'CLIFFORD'

Procedures in 'Cliplus':

&dw - dotted wedge
LCbig - procedure that extends left contraction LC from 'CLIFFORD'
RCbig - procedure that extends right contraction procedure RC from 'CLIFFORD'

clibasis - procedure that defines a Clifford basis consisting of Clifford monomials, e.g., e1 &C e2, &C(e1,e2,e3), etc.
clieval - procedure that converts polynomials in Cl(B) from the Clifford basis to the Grassmann basis
cliexpand - procedure that converts polynomials in Cl(B) from the Grassmann basis to the Clifford basis
climul - procedure that extends Clifford product cmul to polynomials expressed in terms of the Clifford basis
clirev - procedure that extends reversion to polynomials expressed in terms of the Clifford basis
convert/dwedge_to_wedge - procedure that converts from the undotted wedge basis to the dotted wedge basis
convert/wedge_to_dwedge - procedure that converts from the dotted wedge basis to the undotted wedge basis
dottedcbasis - procedure that returns a dotted wedge basis for Cl(B)
\textbf{Bigebra' - Package for Computations with Hopf Algebras}

\textit{Procedures in 'Bigebra':}

- \texttt{&cco} - Clifford co-product
- \texttt{&gco} - Grassmann co-product
- \texttt{&gco\_d} - Grassmann co-product w.r.t. the dotted wedge product computed in the undotted basis!
- \texttt{&gco\_pl} - Grassmann-Plücker co-product acting on hyperplanes in Plücker coordinatization
- \texttt{&map} - maps a product of adjacent slots onto a tensor polynomial
- \texttt{&v} - the vee (meet) product
- \texttt{EV} - the evaluation map
- \texttt{VERSION} - displays version of 'Bigebra'
- \texttt{bracket} - the bracket of Peano space
- \texttt{contract} - contraction of adjacent slots in tensor
- \texttt{define} - partially patched version of Maple' original 'define' facility
- \texttt{drop\_t} - drops the tensor symbol from tensors of rank one
- \texttt{gantipode} - the antipode map for Grassmann Hopf algebra
- \texttt{gco\_unit} - Grassmann co-unit
- \texttt{gswitch} - graded switch of tensor slots
- \texttt{help} - the main help page for 'Bigebra'
- \texttt{linop} - defines a linear operator on $\Lambda V$
- \texttt{linop2} - defines a linear operator on $\Lambda V \times \Lambda V$
- \texttt{lists2mat} - computes a matrix representation from the action of an operator $O : V^\wedge \rightarrow V^\wedge$ given on two lists of source and target elements
- \texttt{lists2mat2} - samee functionality as above, but for operators $O_2 : V^\wedge \&t V^\wedge \rightarrow V^\wedge \&t V^\wedge$
- \texttt{make\_BI\_Id} - initialize Clifford co-product
- \texttt{mapop} - maps a linear operator from \texttt{End} $\Lambda V$ onto tensor slots
- \texttt{mapop2} - maps a linear operator from \texttt{End} $(\Lambda V \&t \Lambda V)$ onto two tensor slots
- \texttt{meet} - the meet (vee) product
- \texttt{op2mat} - computes a matrix representation of an operator $O : V^\wedge \rightarrow V^\wedge$ given as function.
- \texttt{op2mat2} - sam as op2mat but for operators $O_2 : V^\wedge \&t V^\wedge \rightarrow V^\wedge \&t V^\wedge$.
- \texttt{pairing} - computes pairing w.r.t. a bilinear form $B$
- \texttt{peek} - picks elements from tensor slots
- \texttt{poke} - puts elements into tensor slots
- \texttt{remove\_eq} - helper function that removes tautologies from equation sets
switch - switch tensor slots
tcollect - collects coefficients of a tensor polynomial
tsolve1 - solves n --> 1 equations for endomorphisms

Types in 'Bigebra':
type/tensorbasmonom - new type: tensor basis monomial
type/tensormonom - new type: tensor monomial
type/tensorpolynom - new type: tensor polynomial

Sample code:

```
> restart:with(Clifford):interface(verbosityproc=2):
> eval(`&c`);
> `&c`;
```

See Also: Clifford:-makealiases, Clifford:-setup, Clifford:-cbasis, Clifford:-version

(c) Copyright October 8, 1995, by Rafal Ablamowicz & Bertfried Fauser, all rights reserved.
Last modified: September 17, 2005, RA/BF.

References


Help For:

```
> restart:with(Bigebra);
Increase verbosity by infolevel[`function`]=val -- use online help > ?Bigeb
Calling Sequence:

function(args)                      (if the package was loaded using with(Bigebra); )
Bigebra:-function(args)        (long form without loading the package)

Description:

- The BIGEBRA package supplements the CLIFFORD package Clifford version 8 for Maple 8. If BIGEBRA is loaded using with(Bigebra); it loads automatically the CLIFFORD package. BIGEBRA patches the Maple define/skeleton and define/multilinear routines of Maples define facility to allow a correct implementation of the tensor product.
The main purpose of the BIGEBRA package is to allow computations in tensor products of Clifford and Grassmann algebras. For this purpose, a tensor product `\&t` is defined which is linear with respect to all non-Clifford elements (constants). This allows to perform calculations in Grassmann/Clifford modules and Grassmann/Clifford bundles. Bi- and Hopf algebraic structures as co-units, co-products, switches etc. are employed. All structures of Grassmann Hopf algebra and Clifford biconvolution are implemented. However, using this device, Grassmann-Cayley algebras and bracket or Peano algebras are also supported. Especially the meet (of point fields and of plane fields in Plücker coordinatization) is implemented here in a very effective way. The join (of point fields) is implemented by the wedge of the CLIFFORD package.

There are several functions which allow the usage of linear operators given in a matrix representation w.r.t. the Grassmann basis. Such operators can act on a single tensor slot, i.e. they are from End \( \Lambda V \), or on two adjacent tensor slots, i.e. they are from End (\( \Lambda V \&t \Lambda V \)), where \( \Lambda V \) is the space underlying the Grassmann algebra.

The BIGEBRA package provides a facility to solve tangle equations [6] for linear operators applied to internal lines of the tangle if the tangle equation has \( n \) ingoing and one outgoing line (\( n \rightarrow 1 \) mapping). This simplifies e.g. the search for Clifford antipodes.

The Clifford product can be defined in terms of Hopf algebras [8]. BIGEBRA uses the Clifford product of CLIFFORD cmul which internally uses by default the cmulRS subroutine based on the Rota-Stein cliffordization technique and Hopf algebraic methods. The Clifford co-product is derived from co-cliffordization in the same way.

The Clifford co-product needs an additional bilinear form, called co-scalarproduct, which has to be defined as the global \( \text{dim}_V \times \text{dim}_V \) matrix \( BI \). The dimension has to be specified using the global variable \( \text{dim}_V \) of CLIFFORD. The Clifford co-product needs an initialization which is done by calling once the function \( \text{make_BI_Id} \). Some caution is needed here, since \( \text{dim}_V \) is set to the maximal value 9 by CLIFFORD and the initialization may take very long in this case, so that \( \text{dim}_V \) should be set to a smaller value if possible.
The BIGEBRA package makes use of some global variables, which are stored in the table _CLIENV. Currently in use are:
- _CLIENV[_SILENT], default = unassigned. If `true` it suppresses lots of startup output.
- _CLIENV[_fakenow], a flag used to detect if BIGEBRA was already loaded. Needed for patching define.
- _CLIENV[_QDEF_PREFAC], default = -1. Puts q-deformation into the Grassmann coproduct, (beware: ONLY there for now, the q-busines is not yet officially supported and not well tested).

BIGEBRA can also serve to provide the user a possibility to define various multilinear functions, i.e. tensor products over arbitrary rings, see define.

The help pages of BIGEBRA are part of the same Maple database file (maple.hdb) which contains help pages for 'CLIFFORD' and should be located in a directory in Maple's `libname[1]` variable. BIGEBRA is supposed to merge with CLIFFORD in a forthcoming version for Maple ver. 6/7.

BIGEBRA was already successfully used in deriving mathematically and physically relevant results [1,2,3]. Some references are added to provide information about Clifford Hopf gebras.

Literature:


Load Bigebra in the following way!

You can increase the verbosity level of Bigebra setting \texttt{infolevel[Bigebra]=3} or higher.

\begin{verbatim}
> restart:with(Clifford):infolevel(Bigebra)=3:with(Bigebra):

Increase verbosity by \texttt{infolevel[`function`]=val -- use online help > ?Bigebra[help]}

To initialize the Clifford coproduct type:

\begin{verbatim}
> dim_V:=2:
  BI:=linalg[\texttt{matrix}](\texttt{dim}_V,\texttt{dim}_V, [a,b,c,d]);
  make_BI_Id();

\texttt{BI:=}\begin{bmatrix} a & b \\ c & d \end{bmatrix}
\end{verbatim}

Cliplus ver. 13.2 of 3/24/2012 has been loaded. Definitions for type/climon
and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

\[ (Id \&t Id) + a(e1 \&t e1) + c(e2 \&t e1) + b(e1 \&t e2) + d(e2 \&t e2) + (c b - d a)(e1we2 \&t e1we2) \]

\(Bl\) is the \(\text{dim}_V \times \text{dim}_V\) matrix of the co-scalar product on co-one-vectors, from which the Clifford co-product \`&cco` is derived by Rota-Stein co-cliffordization, [2,7,8]. The tensor product \`\&t\` is already defined and ready to use:

```plaintext
> \&t(e1,\&t(e2,e3),e4);  ## associativity, i.e. drop 'parentheses'
&t(e1, e2, e3, e4)
```

```plaintext
> \&t(a*e1+sin(theta)*e3,b*e3-1/x*e1);  ## multilinearity
```

\[ a b(e1 \&t e3) - \frac{a(e1 \&t e1)}{x} + \sin(\theta) b(e3 \&t e3) - \frac{\sin(\theta)(e3 \&t e1)}{x} \]

Alphabetic listing of available procedures in 'BIGEBRA':

- &cco -- Clifford co-product on
- &gco -- The Grassmann co-product w.r.t. the wedge product.
- &gco_d -- dotted Grassmann co-product acting on the undotted wedge basis.
- &gpl_co -- Grassmann-Plücker co-product acting on hyperplanes in Plücker coordinatization.
- &map -- &map maps a product, i.e. a Clifford valued function of two Clifford polynoms (a 2->1 mapping) onto two adjacent slots of a tensor.
- &t -- The tensor product defined in BIGEBRA during loading of the package.
- &v -- Defined the vee-product, i.e. the meet.
- tensor polynoms.
- bracket -- Defines a bracket in the sense of a Peano space [8].
- cco_monom -- internal use only.
- contract -- Contract maps a cliscalar valued function of two Clifford polynoms onto two adjacent tensor slots.
- define -- Maple 8 'define' still has bugs, so 'define' had to be replaced by a patched code. **New option:** give a domain for k-multilinearity.
- drop_t -- Drops the tensor sign \&t in expressions like \&t(e1), projects on the first
argument in &t(p1,p2,...).

- **eps** -- no longer supported.
- **EV** -- EV is the evaluation of a multi-co-vector on a multivector. Multi-co-vectors are described currently (we are sorry to say) by the same Grassmann basis elements. The user is **responsible** to take care in which tensor slot co-vectors reside.

- **gantipode** -- Applies the Grassmann antipode to a tensor slot.
- **gco_unit** -- The Grassmann Hopf algebra co-unit.
- **gswitch** -- Graded switch of two adjacent slots of a tensor.
- **help** -- This page.
- **linop** -- Linop defines a linear operator acting on the Grassmann algebra, having a $2^\dim_V \times 2^\dim_V$ co-contra-variant matrix representing it.

- **linop2** -- Linop2 defines a linear operator acting on a tensor product of rank two of the Grassmann algebra, having a $4^\dim_V \times 4^\dim_V$ co-contra-variant matrix representing it.

- **list2mat** -- List2mat computes from two lists of elements from $V^\wedge$ which are connected as source and target of an linear operator a (possibly unfaithful reducible) matrix representation.

- **list2mat2** -- List2mat2 computes from two lists of elements from $V^\wedge \&t V^\wedge$ which are connected as source and target of an linear operator a (possibly unfaithful reducible) matrix representation.

- **make_BI_Id** -- Initialization routine for the Clifford co-product.

- **&map** -- &map maps a product, i.e. a Clifford valued function of two Clifford polynoms (a 2->1 mapping) onto two adjacent slots of a tensor.

- **mapop** -- Mapop applies a linear operator (element of End $V$) defined by linop onto one single slot of a tensor.

- **mapop2** -- Mapop2 applies a linear tensor-operator (element of End $V \&t V$) defined by linop2 onto two slots of a tensor.

- **meet** -- The meet is equivalent to the &v-(vee)-product.

- **op2mat** -- Op2mat returns a (possibly unfaithful reducible) matrix representation in $V^\wedge$ of a linear operator given as argument.

- **op2mat2** -- Op2mat2 returns a (possibly unfaithful reducible) matrix representation in $V^\wedge \&t V^\wedge$ of a linear operator given as argument.

- **pairing** -- A pairing of two Clifford polynoms.

- **peek** -- Peek gets a Clifford polynom from a tensor at a certain position.
• **poke** -- Poke puts a *Clifford polynom* into a *tensor* at a certain position.

• **remove_eq** -- Helper function, which allows to remove trivial equations if tangle equations are solved manually.

• **switch** -- Switch two adjacent slots of a *tensor* (Just a swap).

• **tcollect** -- Tcollect collects *cliscalar* coefficients in a *tensor* expression.

• **tsolve1** -- Tsolve1 solves tangle equations with n ingoing and one outgoing line (n--> 1 mappings). It has the ability to solve for operators applied to internal lines of the tangle. Such operators can be defined algebraically or using *linop* and *linop2*.

• **VERSION** -- Displays information about the current version of BIGEBRA.

**New Types in 'BIGEBRA':**

• **type/tensobasmonom** - A tensor basis monom having no prefactor.

• **type/tensormonom** - A tensor monom which may have a prefactor of type *cliscalar*.

• **type/tensorpolynom** - A sum of tensor monoms.

**See Also:** Clifford:-setup, Clifford:-version, Bigebra:-VERSION

(c) Copyright 1999 -- 2006, by Rafal Ablamowicz & Bertfried Fauser, all rights reserved. Last modified: version 9, September 17, 2005 /BF/RA.

Cookeville, August 11, 2008/June 28, 2012