

Function: Clifford:-version - display information about the current version of the 'CLIFFORD' package

Calling Sequence:

version();

Parameters:

no parameters needed

Description:

- Procedure 'version' displays information about the current version of the 'CLIFFORD' package.

Examples:

```
[ > restart:with(Clifford) :  
  > version() ;  
  
  ++++++  
  CLIFFORD - A Maple 12 Package for Clifford Algebras with "Bigebra"  
  (Version 12 with environmental variables given by CLIFFORD_ENV())  
  Last revised: December 20, 2009 (Source file: clifford_M12_12.mws)  
  Copyright 1995-2009 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.  
  ++++++This is CLIFFORD version 12+++++
```

[There is a supplementary package '[Cliplus](#)' that extends 'CLIFFORD':
[

```
[ > with(Cliplus) ;
```

```
Clipus has been loaded. Definitions for type/climon and type/clipolynom now in  
clude &C and &C[K]. Type ?cliprod for help.
```

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

```
[ >
```

```
[ >
```

 **See Also:** [CLIFFORD_ENV](#), [Clifford:-makealiases](#)

(c) Copyright 1995-2009, by Rafal Ablamowicz & Bertfried Fauser, all rights reserved.

Last modified: December 20, 2008, RA/BF.

- Function: Clifford:-wexp - exterior exponential in the Clifford algebra Cl(B)

Calling Sequence:

wexp(p,k);

Parameters:

p - an expression of type 'cliscalar' or 'clipolynom' or 'numeric'

k - a non-negative integer

- Description:

- Procedure 'wexp' computes the Clifford exponential of a Clifford polynomial p in Cl(B) up to the order specified by the second argument k which is expected to be a nonnegative integer.
- If k = 0 then the procedure returns 1 or 'Id' depending whether p is of type 'cliscalar'. See [`type/cliscalar`](#) and [`type/clipolynom`](#) for more help on these basic types.
- Use 'cexp' to compute the Clifford exponential in Cl(B) and 'cexpQ' to compute the Clifford exponential in Cl(Q). See [cexp](#), [cexpQ](#) and [sexp](#) for more help.
- Note that one may exponentiate not only polynomials but also polynomials times some parameter.
- It is not necessary that the form Q (or B) be defined.
- Use 'clicollect' to collect terms with respect to Clifford monomials. See [clicollect](#) for more help.

- Examples:

```
[ > restart:with(Clifford) :
[ > wexp(e1we2*t,3):clicollect(%);
  Cliplus has been loaded. Definitions for type/climon and type/clipolynom now in
  clude &C and &C[K]. Type ?cliprod for help.
                                     Id + t e1we2
[ > p:=e1we2+e1we3+e2we3: #define p
[ > wexp(p,0);
                                     1
[ > wexp(p,2);
                                     e1we2 + e1we3 + e2we3 + 1
[ > wexp(p*t,0);
                                     1
[ > wexp((e1we2+e3we4)*s,3):clicollect(%);
                                     Id + s e1we2 + s e3we4 + s^2 e1we2we3we4
[ > wexp(2*alpha,0);
                                     1
[ > wexp(2*alpha,7);
```


Function: `&c`, `&cQ`, `&w`, `&q`, `&cm`, `&cQm`, `&om`, `&wm`, `&qm`, `rm`, `&C`

Calling Sequence:

`&c(p1,p2,...pn)` or `p1 &c p2 &c ... &c pn`
`&c[K](p1,p2,...pn)`
`&C(p1,p2,...pn)` or `p1 &C p2 &C ... &C pn`
`&C[K](p1,p2,...pn)`
`&cQ(p1,p2,...pn)` or `p1 &cQ p2 &cQ ... &cQ pn`
`&cQ[K](p1,p2,...pn)`
`&w(p1,p2,...pn)` or `p1 &w p2 &w ... &w pn`
`&q(q1,q2,...qn)` or `q1 &q q2 &q ... &q qn`

`&cm(M1,M2,...,Mn)` or `M1 &cm M2 &cm ... &cm Mn`
`&cm[K](M1,M2,...,Mn)`
`&cQm(M1,M2,...,Mn)` or `M1 &cQm M2 &cQm ... &cQm Mn`
`&cQm[K](M1,M2,...,Mn)`
`&om(o1,o2,...,on)` (octonionic multiplication of matrices - requires package 'Octonion')
`&wm(M1,M2,...,Mn)` or `M1 &wm M2 &wm ... &wm Mn`
`&qm(Q1,Q2,...,Qn)` or `Q1 &qm Q2 &qm ... &qm Qn`
`&rm(M1,M2,...,Mn)` or `M1 &rm M2 &rm ... &rm Mn`
`&rm[K](M1,M2,...,Mn)`

Parameters:

`p1, p2, ..., pn` - expressions of the type 'cliscalar', 'clibasmon', 'climon', or 'clipolynom'
`q1, q2, ..., qn` - expressions of the type 'quaternion'
`o1, o2, ..., on` - expressions of type 'octonion' (see package 'Octonion' for more information)
`M1, M2, ..., Mn` - matrices of the type 'climatrix'
`Q1, Q2, ..., Qn` - matrices of the type 'climatrix' with quaternionic entries, that is, entries of type 'quaternion'
`K` - (optional) bilinear form of type 'name', 'symbol', 'matrix', 'array', or '&*' (numeric, {name,symbol,array,matrix})

Description:

- CLIFFORD defines the following infix forms upon loading:
 - `&c` is the infix form for Clifford multiplication 'cmul' in $Cl(B)$ (see [cmul](#) for more help),
 - `&C` is the inert form for Clifford multiplication 'cmul' in $Cl(B)$ (see [type/cliprod](#) for more help),
 - `&cQ` is the infix form for Clifford multiplication 'cmulQ' in $Cl(Q)$ (see [cmulQ](#) for more help),
 - `&w` is the infix form for wedge (exterior) multiplication 'wedge' in $Cl(B)$ (see [wedge](#) for more help),
 - `&q` is the infix form for quaternionic multiplication 'qmul' in $Cl([1,1,1])$ (see [qmul](#) for more help),

- `&cm` is the infix form for multiplication of Clifford matrices, that is, matrices with entries in $Cl(B)$ when Clifford multiplication '`cmul`' is applied to matrix entries (see [`type/climatrix`](#) and [`rmulm`](#) for more help),
- `&cQm` is the infix form for multiplication of Clifford matrices, that is, matrices with entries in $Cl(Q)$ when Clifford multiplication '`cmulQ`' is applied to matrix entries (see [`type/climatrix`](#) and [`rmulm`](#) for more help),
- `&wm` is the infix form for multiplication of Clifford matrices, that is, matrices with entries in $Cl(B)$ when wedge (exterior) multiplication '`wedge`' is applied to matrix entries (see [`type/climatrix`](#) and [`rmulm`](#) for more help),
- `&qm` is the infix form for multiplication of Clifford matrices, that is, matrices with entries in $Cl(B)$ of type 'quaternion' when quaternionic multiplication '`qmul`' is applied to matrix entries (see [quaternion](#), [`qmul`](#), [`type/climatrix`](#) and [`rmulm`](#) for more help),
- `&om` is the infix form for multiplication of Clifford matrices, that is, matrices with entries in $Cl(B)$ of type 'octonion' when octonionic multiplication '`omul`' is applied to matrix entries (see [octonion](#), [`omul`](#), [`type/climatrix`](#) and [`rmulm`](#) for more help),
- `&rm` is the infix form for multiplication of Clifford matrices, that is, matrices with entries in $Cl(B)$ (matrices of type '`climatrix`') when when some generic, yet-to-be defined, multiplication '`r`' is applied to matrix entries (see [`type/climatrix`](#) and [`rmulm`](#) for more help),
- The infix forms `&c`, `&cQ`, `&cm`, `&rm`, and `&C` can accept an optional index of type `'name'`, `'symbol'`, `'matrix'`, `'array'`, or `'&*' (numeric, {name,symbol,array,matrix})` as in, for example, `&c[K](p1,p2,...,pn)`, `&c[-K](p1,p2,...,pn)`.
- NOTE: When index `K` has been assigned a matrix, double quotes must be used around `K` to stop premature evaluation of the matrix by Maple as in `&c["K"](p1,p2,...,pn)`, `&c["-K"](p1,p2,...,pn)`, `&c["2*K"](p1,p2,...,pn)`, `&c["-2*K"](p1,p2,...,pn)`. See examples below.

Examples:

```
> restart:
bench:=time():
with(Clifford):
```

Example 1: There are two different ways to use the infix form:

A. Infix form of '`cmul`':

```
> cmul(1+e1we2+2*e3, e2+3*e4);      #long form in Cl(B)
cmul[B](1+e1we2+2*e3, e2+3*e4);   #long form in Cl(B)
cmul[K](1+e1we2+2*e3, e2+3*e4);   #long form in Cl(K)
cmul[-K](1+e1we2+2*e3, e2+3*e4);  #long form in Cl(-K)
```

Cliplus has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.

$$2(B_{3,2} + 3B_{3,4})Id - 2e_2we_3 - (-1 + B_{1,2} + 3B_{1,4})e_2 + 3e_4 + (B_{2,2} + 3B_{2,4})e_1 + 3e_1we_2we_4 + 6e_3we_4$$

$$2(B_{3,2} + 3B_{3,4})Id + (B_{2,2} + 3B_{2,4})e1 - (-1 + B_{1,2} + 3B_{1,4})e2 + 3e4 - 2e2we3 + 6e3we4 + 3e1we2we4$$

$$2(K_{3,2} + 3K_{3,4})Id + (K_{2,2} + 3K_{2,4})e1 - (-1 + K_{1,2} + 3K_{1,4})e2 + 3e4 - 2e2we3 + 6e3we4 + 3e1we2we4$$

$$-2(K_{3,2} + 3K_{3,4})Id - (K_{2,2} + 3K_{2,4})e1 + (1 + K_{1,2} + 3K_{1,4})e2 + 3e4 - 2e2we3 + 6e3we4 + 3e1we2we4$$

```
> &c(1+e1we2+2*e3,e2+3*e4); #infix form for cmul in Cl(B)
(1+e1we2+2*e3) &c(e2+3*e4); #infix form for cmul in Cl(B)
&c[K](1+e1we2+2*e3,e2+3*e4); #infix form for cmul in Cl(K)
&c[-K](1+e1we2+2*e3,e2+3*e4); #infix form for cmul in Cl(-K)
```

$$2(B_{3,2} + 3B_{3,4})Id + (B_{2,2} + 3B_{2,4})e1 - (-1 + B_{1,2} + 3B_{1,4})e2 + 3e4 - 2e2we3 + 6e3we4 + 3e1we2we4$$

$$2(B_{3,2} + 3B_{3,4})Id + (B_{2,2} + 3B_{2,4})e1 - (-1 + B_{1,2} + 3B_{1,4})e2 + 3e4 - 2e2we3 + 6e3we4 + 3e1we2we4$$

$$2(K_{3,2} + 3K_{3,4})Id + (K_{2,2} + 3K_{2,4})e1 - (-1 + K_{1,2} + 3K_{1,4})e2 + 3e4 - 2e2we3 + 6e3we4 + 3e1we2we4$$

$$-2(K_{3,2} + 3K_{3,4})Id - (K_{2,2} + 3K_{2,4})e1 + (1 + K_{1,2} + 3K_{1,4})e2 + 3e4 - 2e2we3 + 6e3we4 + 3e1we2we4$$

[K can be a matrix::

```
> K:=linalg[diag](1$4):
```

```
> &c['K'](1+e1we2+2*e3,e2+3*e4); #infix form for cmul in Cl(K)
e1 + e2 + 3 e4 - 2 e2we3 + 6 e3we4 + 3 e1we2we4
```

```
> &c['K'](1+e1we2+2*e3,e2+3*e4); #<<<- Intended error, infix form
for cmul in Cl(K) / single quotes are not enough
```

Error, (in Clifford:-cliparse) check spelling of sparse or define it as a constant or an alias

```
> &c[K](1+e1we2+2*e3,e2+3*e4); #<<<- Intended error, infix form
for cmul in Cl(K)
```

Error, (in Clifford:-cliparse) check spelling of sparse or define it as a constant or an alias

```
> &c['-K'](1+e1we2+2*e3,e2+3*e4); #infix form for cmul in Cl(K)
-e1 + e2 + 3 e4 - 2 e2we3 + 6 e3we4 + 3 e1we2we4
```

```
> &c['-2*K'](1+e1we2+2*e3,e2+3*e4); #infix form for cmul in
Cl(K)
```

$$-2e1 + e2 + 3e4 - 2e2we3 + 6e3we4 + 3e1we2we4$$

```
> K:='K': #unassigning K for next examples
```

```
> cmul(e1,e2-e5,e3-2*e1); cmul[K](e1,e2-e5,e3-2*e1); cmul[-K](e1,e2
-e5,e3-2*e1);
```

```

-(B5,3 - 2 B1,5 + 2 B2,1 - B2,3 + 2 B1,2 - 2 B5,1) e1 + (2 B1,1 - B1,3) e2 + (B1,2 - B1,5) e3
- (2 B1,1 - B1,3) e5 + e1e2e3 + e1e3e5
-(K5,3 - 2 K1,5 + 2 K2,1 - K2,3 + 2 K1,2 - 2 K5,1) e1 + (2 K1,1 - K1,3) e2 + (K1,2 - K1,5) e3
- (2 K1,1 - K1,3) e5 + e1e2e3 + e1e3e5
(K5,3 - 2 K1,5 + 2 K2,1 - K2,3 + 2 K1,2 - 2 K5,1) e1 - (2 K1,1 - K1,3) e2 - (K1,2 - K1,5) e3
+ (2 K1,1 - K1,3) e5 + e1e2e3 + e1e3e5
> &c(e1, e2, e3);
                                     B2,3 e1 - B1,3 e2 + B1,2 e3 + e1e2e3
> cmul[B](e1, e2, e3);
                                     B2,3 e1 - B1,3 e2 + B1,2 e3 + e1e2e3
> &c('B')(e1+3*e3, e2-e3, e4);
(B2,4 - B3,4) e1 - (3 B3,4 + B1,4) e2 + (3 B2,4 + B1,4) e3 + (B1,2 - 3 B3,3 + 3 B3,2 - B1,3) e4
+ e1e2e4 - e1e3e4 - 3 e2e3e4
> &c('B')(e1, e2);
                                     B1,2 Id + e1e2
> &c('B')(e1+3*e3, e2-e3, e4);
(B2,4 - B3,4) e1 - (3 B3,4 + B1,4) e2 + (3 B2,4 + B1,4) e3 + (B1,2 - 3 B3,3 + 3 B3,2 - B1,3) e4
+ e1e2e4 - e1e3e4 - 3 e2e3e4
> e1 &c e2;
                                     B1,2 Id + e1e2
> (e1+e2) &c (e3-2*e4); &c[K]((e1+e2), (e3-2*e4));
                                     (B1,3 + B2,3 - 2 B1,4 - 2 B2,4) Id + e1e3 - 2 e1e4 + e2e3 - 2 e2e4
                                     (K1,3 + K2,3 - 2 K1,4 - 2 K2,4) Id + e1e3 - 2 e1e4 + e2e3 - 2 e2e4
> (e1+e2) &c (e3-2*e4) &c (2*e9); &c[K]((e1+e2), (e3-2*e4), 2*e9);
2(-2 B4,9 + B3,9) e1 + 2(-2 B4,9 + B3,9) e2 - 2(B2,9 + B1,9) e3 + 4(B2,9 + B1,9) e4
+ 2(B1,3 + B2,3 - 2 B1,4 - 2 B2,4) e9 + 2 e1e3e9 - 4 e1e4e9 + 2 e2e3e9
- 4 e2e4e9
2(-2 K4,9 + K3,9) e1 + 2(-2 K4,9 + K3,9) e2 - 2(K2,9 + K1,9) e3 + 4(K2,9 + K1,9) e4
+ 2(K1,3 + K2,3 - 2 K1,4 - 2 K2,4) e9 + 2 e1e3e9 - 4 e1e4e9 + 2 e2e3e9
- 4 e2e4e9
B. Infix form of 'cmulQ':
> B:=linalg[diag](1$9);
> cmulQ(e1+e2, e1-2*e4); #long form
cmulQ[B](e1+e2, e1-2*e4); #long form
cmulQ[K](e1+e2, e1-2*e4); #optional parameter in cmulQ

```

```
cmulQ[-K](e1+e2,e1-2*e4); #optional parameter in cmulQ
```

$Id - e1we2 - 2 e1we4 - 2 e2we4$

$Id - e1we2 - 2 e1we4 - 2 e2we4$

$(K_{1,1} + K_{2,1} - 2 K_{1,4} - 2 K_{2,4}) Id - e1we2 - 2 e1we4 - 2 e2we4$

$-(K_{1,1} + K_{2,1} - 2 K_{1,4} - 2 K_{2,4}) Id - e1we2 - 2 e1we4 - 2 e2we4$

```
> &cQ(e1+e2,e1-2*e4); #infix form for cmulQ
```

```
&cQ[''B''](e1+e2,e1-2*e4); #infix form for cmulQ
```

```
(e1+e2) &cQ (e1-2*e4); #infix form for cmulQ
```

```
&cQ[K](e1+e2,e1-2*e4); #infix form for cmulQ
```

```
&cQ[-K](e1+e2,e1-2*e4); #infix form for cmulQ
```

$Id - e1we2 - 2 e1we4 - 2 e2we4$

$Id - e1we2 - 2 e1we4 - 2 e2we4$

$Id - e1we2 - 2 e1we4 - 2 e2we4$

$(K_{1,1} + K_{2,1} - 2 K_{1,4} - 2 K_{2,4}) Id - e1we2 - 2 e1we4 - 2 e2we4$

$-(K_{1,1} + K_{2,1} - 2 K_{1,4} - 2 K_{2,4}) Id - e1we2 - 2 e1we4 - 2 e2we4$

Also in `&cQ` double quotes must be used around a name used as index if that name has been assigned a matrix:

```
> K:=linalg[diag](1$4):
```

```
> &cQ[K](e1+e2,e1-2*e4); #<<<Intended error, infix form for cmulQ
```

```
Error, (in Clifford:-cliparse) check spelling of sparse or define it as a constant or an alias
```

```
> &cQ['K'](e1+e2,e1-2*e4); #<<< Intended error, infix form for cmulQ / single quotes are not enough
```

```
Error, (in Clifford:-cliparse) check spelling of sparse or define it as a constant or an alias
```

```
> &cQ[''K''](e1+e2,e1-2*e4); #infix form for cmulQ / double quotes are needed
```

$Id - e1we2 - 2 e1we4 - 2 e2we4$

```
> e1 &cQ e1;
```

Id

C. Infix form of 'wedge':

```
> wedge(e1+e2,e3-2*e4); #long form
```

$e1we3 + e2we3 - 2 e1we4 - 2 e2we4$

```
> &w(e1+e2,e3-2*e4); #short form for wedge
```

$e1we3 + e2we3 - 2 e1we4 - 2 e2we4$

```
> (e1+e2) &w (e3-2*e4);
```

$e1we3 + e2we3 - 2 e1we4 - 2 e2we4$

There is a dwedge procedure in 'Clipus' package for computation of dotted wedge.

D. Infix form of 'qmul'. Defining B to be 3 x 3 diagonal matrix so that we could use quaternions:

```
> B:=linalg[diag](1,1,1):
> q1:=2+2*e1we2+e1we3+e2we3;q2:=-2*e2we3+3+e1we2;
      q1 := 2 + 2 e1we2 + e1we3 + e2we3
      q2 := -2 e2we3 + 3 + e1we2
> type(q1,quaternion),type(q2,quaternion);
      true, true
> qmul(q1,q2); # long form of the quaternionic product
&q(q1,q2); # short form of the quaternionic product
q1 &q q2; # short form of the quaternionic product
      6 - 10 qk - 2 qj
      6 - 10 qk - 2 qj
      6 - 10 qk - 2 qj
```

This is how Maple will display quaternions if you use 'qdisplay':

```
> _quatbasis;
[[Id, e3we2, e1we3, e2we1], {Maple has assigned qi:=-e2we3, qj:=e1we3, qk:=-e1we2}]
> q1:=qdisplay(q1);q2:=qdisplay(q2);
      q1 := 2 - 2 qk + qj - qi
      q2 := 3 - qk + 2 qi
> type(q1,quaternion);type(q2,quaternion);
      true
      true
```

Quaternion multiplication is done with 'qmul' or '&q':

```
> qmul(q1,q2);
&q(q1,q2);
      6 - 10 qk - 2 qj
      6 - 10 qk - 2 qj
```

Let's find the multiplication table for H:

```
> Q:=[1,'qi','qj','qk'];
      Q := [1, qi, qj, qk]
> A := array(1..4,1..4):
for i from 1 to 4 do for j from 1 to 4 do A[i,j]:=Q[i] &q Q[j]
od od;
> print(A);
```

$$\begin{bmatrix} 1 & q_i & q_j & q_k \\ q_i & -1 & q_k & -q_j \\ q_j & -q_k & -1 & q_i \\ q_k & q_j & -q_i & -1 \end{bmatrix}$$

E. Infix form `&rm` for user-defined product `&r` when applied to matrices:

```
> M1:=matrix(2,2,[e1+e2,e3-3,e2+4,Id]);
```

$$M1 := \begin{bmatrix} e1 + e2 & e3 - 3 \\ e2 + 4 & Id \end{bmatrix}$$

```
> type(M1,climatrix);
```

true

```
> &rm(M1,M1); #implicitly Clifford product is computed in Cl(B)
```

$$\begin{bmatrix} \&r_B(e1 + e2, e1 + e2) + \&r_B(e3 - 3, e2 + 4) & \&r_B(e1 + e2, e3 - 3) + \&r_B(e3 - 3, 1) \\ \&r_B(e2 + 4, e1 + e2) + \&r_B(1, e2 + 4) & \&r_B(e2 + 4, e3 - 3) + \&r_B(1, 1) \end{bmatrix}$$

```
> evalm(B);
```

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

Since B has been assigned a matrix, we must use double quotes as follows:

```
> &rm['-B'](M1,M1); #explicitly Clifford product is computed in Cl(B)
```

$$\begin{bmatrix} \&r_{-B}(e1 + e2, e1 + e2) + \&r_{-B}(e3 - 3, e2 + 4) & \&r_{-B}(e1 + e2, e3 - 3) + \&r_{-B}(e3 - 3, 1) \\ \&r_{-B}(e2 + 4, e1 + e2) + \&r_{-B}(1, e2 + 4) & \&r_{-B}(e2 + 4, e3 - 3) + \&r_{-B}(1, 1) \end{bmatrix}$$

If double quotes are not used, an error message will result:

```
> &rm['B'](M1,M1); #<<< Intended error: explicitly Clifford product is computed in Cl(B)
```

Warning, enclose index in double quotes as in `&rm['B']` or `&rm['-B']` when B has been assigned a matrix to avoid the following:

$$\&rm\left(\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)\right)$$

```
> &rm['-B'](M1,M1); #<<< Intended error: single quotes are not enough
```

Error, (in decindex) unable to determine index or wrong index type for `&rm`, try enclosing name of the index in double quotes as in `&rm['B']` or `&rm['-B']`

```
> &rm['-B'](M1,M1); #explicitly Clifford product is computed in Cl(B)
```

$$\begin{bmatrix} \&r_{-B}(e1 + e2, e1 + e2) + \&r_{-B}(e3 - 3, e2 + 4) & \&r_{-B}(e1 + e2, e3 - 3) + \&r_{-B}(e3 - 3, 1) \\ \&r_{-B}(e2 + 4, e1 + e2) + \&r_{-B}(1, e2 + 4) & \&r_{-B}(e2 + 4, e3 - 3) + \&r_{-B}(1, 1) \end{bmatrix}$$

Example 2: Multiplication of matrices with entries in $Cl(B)$ is done as follows. As an example, we define B to have both the symmetric and antisymmetric parts.

E. Infix form of the Clifford product applied to matrices with entries in a Clifford algebra.

```
> B:=matrix(3,3,[ 1, a, b,
                  -a, 1, c,
                  -b,-c, 1]);
```

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

```
> eval(makealiases(3)):
```

```
> M1:=matrix(2,2,[1+2*e3,e3+e2,e1we2we3,e2+2*e3]);
```

$$M1 := \begin{bmatrix} 1 + 2 e3 & e3 + e2 \\ e123 & e2 + 2 e3 \end{bmatrix}$$

```
> M2:=matrix(2,2,[2*e1we2,-e1+e2,2*e2we3,e2-2*e3]);
```

$$M2 := \begin{bmatrix} 2 e12 & -e1 + e2 \\ 2 e23 & e2 - 2 e3 \end{bmatrix}$$

```
> type(M1,climatrix);type(M2,climatrix);
```

true

true

```
> M1 &cm M2; #Clifford product 'cmul' is applied to matrix
entries
```

```
[2 e12 + 4 e123 - 4 b e2 + 4 c e1 - 2 e3 c + 2 e3 - 2 e2 - 2 e2 c,
 -5 c + 2 b + 2 e13 - 5 e23 + e2 - e1 - 1]
[-2 e3 a^2 + 2 e2 a b - 4 e2 c - 2 e1 b - 2 e1 c a - 4 a e123 - 4 e3 c - 4 e2,
 -a e13 - e13 - 5 e23 + a e23 - c e12 + b e12 - 3 - 4 c]
```

```
> map(clicollect, %);
```

```
[2 e12 - 2 (c - 1) e3 - 2 (2 b + c + 1) e2 + 4 c e1 + 4 e123,
 (-5 c + 2 b - 1) Id + 2 e13 - 5 e23 + e2 - e1]
[-2 (a^2 + 2 c) e3 + 2 (a b - 2 - 2 c) e2 - 2 (b + c a) e1 - 4 a e123,
 -(3 + 4 c) Id - (a + 1) e13 + (-5 + a) e23 + (-c + b) e12]
```

To use infix form, make sure to enclose the name of the index in double quotes if it has been assigned a matrix. For example,

```
> &cm[X](M1,M2); # No need for 'X' since X is a name, not a
matrix
```

```
[2 e12 + 4 e123 + 4 X3,1 e2 - 4 X3,2 e1 + 2 X3,2 e3 + 2 X2,2 e3 - 2 X3,3 e2 - 2 X2,3 e2,
 3 X3,2 - 2 X3,1 + 2 e13 - 5 e23 + e2 - e1 - 2 X2,3 + X2,2 - 2 X3,3]
[2 e3 X2,1 X1,2 - 2 e3 X2,2 X1,1 - 2 e2 X3,1 X1,2 + 2 e2 X3,2 X1,1 + 2 e1 X3,1 X2,2
 - 2 e1 X3,2 X2,1 + 2 e123 X2,1 - 2 e123 X1,2 + 2 X2,2 e3 + 4 X3,2 e3 - 2 X2,3 e2 - 4 X3,3 e2,
 X2,1 e13 - X2,2 e13 - X1,1 e23 + X1,2 e23 + X3,2 e12 - X3,1 e12 + X2,2 + 2 X3,2 - 2 X2,3]
```

```

-4 X3,3 - 4 e23]
> &cm[''B''] (M1,M2); # Need for ''B'' since B is a matrix
[2 e12 + 4 e123 - 4 b e2 + 4 c e1 - 2 e3 c + 2 e3 - 2 e2 - 2 e2 c ,
-5 c + 2 b + 2 e13 - 5 e23 + e2 - e1 - 1]
[-2 e3 a2 + 2 e2 a b - 4 e2 c - 2 e1 b - 2 e1 c a - 4 a e123 - 4 e3 c - 4 e2 ,
-a e13 - e13 - 5 e23 + a e23 - c e12 + b e12 - 3 - 4 c]
> &cm[''-B''] (M1,M2); # Need for ''B'' since B is a matrix
[2 e12 + 4 e123 + 4 b e2 - 4 c e1 + 2 e3 c - 2 e3 + 2 e2 + 2 e2 c ,
5 c - 2 b + 2 e13 - 5 e23 + e2 - e1 + 1]
[-2 e3 a2 - 4 e3 + 2 e2 a b - 2 e1 b - 2 e1 c a + 4 a e123 + 4 e3 c + 4 e2 ,
a e13 + e13 - 3 e23 - a e23 + c e12 - b e12 + 3 + 4 c]

```

The same applies to `&cQm`:

```

> M1 &cQm M2; #Clifford product 'cmulQ' is applied to matrix
entries
[2 e12 + 4 e123 - 2 e2 + 2 e3 -e1 + 2 e13 + e2 - 5 e23 - 1]
[-4 e2 -5 e23 - e13 - 3]
> &cQm(M1,M2); #default matrix B is used implicitly
[2 e12 + 4 e123 - 2 e2 + 2 e3 -e1 + 2 e13 + e2 - 5 e23 - 1]
[-4 e2 -5 e23 - e13 - 3]
> &cQm[''B''] (M1,M2); #correct result when ''B'' is used
[2 e12 + 4 e123 - 2 e2 + 2 e3 -e1 + 2 e13 + e2 - 5 e23 - 1]
[-4 e2 -5 e23 - e13 - 3]
> &cQm[''-B''] (M1,M2); #correct result
[2 e12 + 4 e123 + 2 e2 - 2 e3 -e1 + 2 e13 + e2 - 5 e23 + 1]
[-4 e3 + 4 e2 -3 e23 + e13 + 3]

```

>

Procedure `&wm` applies the wedge product to the matrix entries and as such it does not need an index.

```

> M1 &wm M2; #wedge (exterior) 'wedge' is applied to matrix
entries
Warning, since B has been (re-)assigned, value of dim_V has been reduced by 'wedge' to 3
[2 e12 + 4 e123 -e1 + 2 e13 + e2 - 5 e23]
[0 -4 e23]
> &wm(M1,M2); #wedge (exterior) 'wedge' is applied to matrix
entries
[2 e12 + 4 e123 -e1 + 2 e13 + e2 - 5 e23]
[0 -4 e23]

```

Note: In this last example, an arbitrary, yet-to-be-defined by the user product 'r' has been applied to matrix entries.

```

[ > B:=linalg[diag](1,1,1):
[ > q1:=2+2*qi+qj+3*qk;q2:=-2*qi+3+qk;q3:=-2-2*qk+qj;q4:=-2*qj+3*qi
+qk;
      q1 := 2 - 2 e23 + e13 - 3 e12
      q2 := 2 e23 + 3 - e12
      q3 := -2 + 2 e12 + e13
      q4 := -2 e13 - 3 e23 - e12
[ > type(q1,quaternion),type(q2,quaternion),
type(q3,quaternion),type(q4,quaternion);
      true, true, true, true
[ > Q1:=matrix(2,2,[q1,q2,q3,q4]);Q2:=matrix(2,2,[q2,-q1,q4,-q3]);
      Q1 := [ 2 - 2 e23 + e13 - 3 e12    2 e23 + 3 - e12 ]
            [ -2 + 2 e12 + e13        -2 e13 - 3 e23 - e12 ]
      Q2 := [ 2 e23 + 3 - e12    -2 + 2 e23 - e13 + 3 e12 ]
            [ -2 e13 - 3 e23 - e12    2 - 2 e12 - e13 ]
[ > type(Q1,climatrix),type(Q2,climatrix);
      true, true
[ > Q1 &qm Q2;
      [ 12 + 20 qk - 6 qj + 14 qi   14 - 2 qk - 3 qj - 11 qi ]
      [ -18 - 6 qk + 7 qj + 5 qi   -5 + 11 qk - 6 qj + 2 qi ]
[ > printf("Worksheet took %f seconds to compute on Pentium M 2.13
GHz 2GB machine with Win XP Professional\n",time()-bench);
Worksheet took 3.250000 seconds to compute on Pentium M 2.13 GHz 2GB machine wi
th Win XP Professional
[ >
[ >

```

See Also: [GTP:-`&t`](#), [Clifford:-rmulm](#), [Clifford:-`type/clipolynom`](#), [Cliplus:-dwedge](#), [Clifford:-useproduct](#)

(c) Copyright 1995-2009, by Rafal Ablamowicz & Bertfried Fauser, all rights reserved.
 Last modified: December 20, 2008, RA/BF.

- Function: Clifford:-adfmatrix, Clifford:-mdfmatrix - add or multiply matrices over double field

Calling Sequence:

adfmatrix(m1,m2) - add two matrices m1 and m2 of `type/dfmatrix` and of the same dimension n x n,

mdfmatrix(m1,m2) - multiply two matrices m1 and m2 of `type/dfmatrix` and of the same dimension n x n,

Parameters:

m1,m2 - matrices of `type/dfmatrix` of the same size

- Description:

- A matrix M is of type 'ddfmatrix' if it is a square matrix whose entries are two element lists. These matrices arise when faithful matrix representations of semi-simple Clifford algebras are calculated. Use [all_sigs](#) to display signatures of semisimple Clifford algebras in dimensions 1 through 9.
- Faithful spinor representations of semisimple Clifford algebras have been precomputed. They are stored in a library file and can be retrieved with the procedure [clidata](#).

- Examples:

```
> restart:bench:=time():  
with(Clifford):with(linalg):  
_default_Clifford_product;  
#useproduct(cmulNUM);  
_default_Clifford_product;
```

cmulRS

cmulRS

Let's display signatures of semi-simple Clifford algebras in dimensions 1 through 9 which are isomorphic to rings of matrices over double real field R+R:

```
> all_sigs(1..9,'real','semisimple');
```

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

Let's display signatures of semi-simple Clifford algebras in dimensions 1 through 9 which are isomorphic to rings of matrices over double quaternionic field H+H:

```
> all_sigs(1..9,'quat','semisimple');
```

```
[[0, 3], [1, 4], [2, 5], [3, 6], [5, 0], [6, 1], [7, 2]]
```

There are no semisimple Clifford algebras that would be isomorphic to rings of matrices of double complex field C+C:

```
> all_sigs(1..9,'complex','semisimple');
```

```
[ ]
```

More information about Clifford algebra Cl(Q) of the quadratic form of signature (2,1) can be

found using procedure [clidata](#) as follows:

```
> clidata([2,1]);
```

$$\left[\text{real}, 2, \text{semisimple}, 'cmulQ\left(\frac{Id}{2} + \frac{e1}{2}, \frac{Id}{2} + \frac{e2we3}{2}\right), [Id, e2], [Id], [Id, e2] \right]$$

Example 1: Let's view matrices $m[i]$, $i=1..3$, representing 1-vectors $\{e1, e2, e3\}$ in $Cl(2,1) = Mat(2,2, R+R)$. These matrices are of type `'type/dfmatrix'` and have been precomputed. They can be displayed with the procedure [matKrepr](#):

```
> pq:=[2,1];
```

```
B:=diag(1$pq[1], -1$pq[2]);
```

$$pq := [2, 1]$$

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

```
> L:=matKrepr();
```

Cliplus has been loaded. Definitions for type/climon and type/clipolynom now include `&C` and `&C[K]`. Type `?cliprod` for help.

$$L := \left[e1 = \begin{bmatrix} [1, -1] & [0, 0] \\ [0, 0] & [-1, 1] \end{bmatrix}, e2 = \begin{bmatrix} [0, 0] & [1, 1] \\ [1, 1] & [0, 0] \end{bmatrix}, e3 = \begin{bmatrix} [0, 0] & [-1, -1] \\ [1, 1] & [0, 0] \end{bmatrix} \right]$$

Let's assign these matrices to $m[i]$, $i=1..3$:

```
> for i from 1 to nops(L) do m[i]:=rhs(L[i]) od;
```

$$m_1 := \begin{bmatrix} [1, -1] & [0, 0] \\ [0, 0] & [-1, 1] \end{bmatrix}$$

$$m_2 := \begin{bmatrix} [0, 0] & [1, 1] \\ [1, 1] & [0, 0] \end{bmatrix}$$

$$m_3 := \begin{bmatrix} [0, 0] & [-1, -1] \\ [1, 1] & [0, 0] \end{bmatrix}$$

Observe that matrices $m[1]$, $m[2]$, and $m[3]$ satisfy appropriate relations:

(A) Squares of $m[1]$, $m[2]$, and $m[3]$ are:

```
> mdfmatrix(m[1],m[1]),mdfmatrix(m[2],m[2]),mdfmatrix(m[3],m[3]);
```

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

(B) Pair-wise, matrices $m[1]$, $m[2]$, and $m[3]$ anticommute:

```
> adfmatrix(mdfmatrix(m[1],m[2]),mdfmatrix(m[2],m[1]));
```

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

```
> adfmatrix(mdfmatrix(m[1],m[3]),mdfmatrix(m[3],m[1]));
```

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

```
> adfmatrix(mdfmatrix(m[2],m[3]),mdfmatrix(m[3],m[2]));
```

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

Thus, we can find a unique matrix representing each basis Grassmann monomial in $Cl(Q)$ as follows. Let's define a homomorphism f from $Cl(Q)$ to $Mat(2,2,R+R)$ as a Maple function:

```
> clibas:=cbasis(3);
      clibas := [Id, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3]
> f:=proc() end:
f(Id):=mdfmatrix(m[1],m[1]):
f(e1):=evalm(m[1]):
f(e2):=evalm(m[2]):
f(e3):=evalm(m[3]):
f(e1we2):=mdfmatrix(m[1],m[2]):
f(e1we3):=mdfmatrix(m[1],m[3]):
f(e2we3):=mdfmatrix(m[2],m[3]):
f(e1we2we3):=mdfmatrix(mdfmatrix(m[1],m[2]),m[3]):
```

Then, we can apply f to the entries of $clibas$:

```
> for x in clibas do
x, " is mapped by f to ", f(x);
od;
      Id, " is mapped by f to ",  $\begin{bmatrix} [1,1] & [0,0] \\ [0,0] & [1,1] \end{bmatrix}$ 
      e1, " is mapped by f to ",  $\begin{bmatrix} [1,-1] & [0,0] \\ [0,0] & [-1,1] \end{bmatrix}$ 
      e2, " is mapped by f to ",  $\begin{bmatrix} [0,0] & [1,1] \\ [1,1] & [0,0] \end{bmatrix}$ 
      e3, " is mapped by f to ",  $\begin{bmatrix} [0,0] & [-1,-1] \\ [1,1] & [0,0] \end{bmatrix}$ 
      e1we2, " is mapped by f to ",  $\begin{bmatrix} [0,0] & [1,-1] \\ [-1,1] & [0,0] \end{bmatrix}$ 
      e1we3, " is mapped by f to ",  $\begin{bmatrix} [0,0] & [-1,1] \\ [-1,1] & [0,0] \end{bmatrix}$ 
      e2we3, " is mapped by f to ",  $\begin{bmatrix} [1,1] & [0,0] \\ [0,0] & [-1,-1] \end{bmatrix}$ 
      e1we2we3, " is mapped by f to ",  $\begin{bmatrix} [1,-1] & [0,0] \\ [0,0] & [1,-1] \end{bmatrix}$ 
```

>

Example 2: Let's view matrices $m[i]$, $i=1..3$, representing 1-vectors $\{e1,e2,e3\}$ in $Cl(0,3) = Mat(1,1,H+H)$. These matrices are of type `type/dfmatrix` and have been precomputed. They can be displayed with the procedure [matKrepr](#):

```
> pq:=[0,3];
B:=diag(1$pq[1],-1$pq[2]);
```

$$pq := [0, 3]$$

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

```
> L:=matKrepr();
```

$$L := [e1 = [[e1, e1]], e2 = [[e2, e2]], e3 = [[-e1we2, e1we2]]]$$

Let's assign these matrices to m[i], i=1..3:

```
> for i from 1 to nops(L) do m[i]:=rhs(L[i]) od;
```

$$m_1 := [[e1, e1]]$$

$$m_2 := [[e2, e2]]$$

$$m_3 := [[-e1we2, e1we2]]$$

```
>
```

Observe that matrices m[1], m[2], and m[3] satisfy appropriate relations:

(A) Squares of m[1], m[2], and m[3] are:

```
> mdfmatrix(m[1],m[1]), mdfmatrix(m[2],m[2]), mdfmatrix(m[3],m[3]);
```

$$[[-1, -1]], [[-1, -1]], [[-1, -1]]$$

(B) Pair-wise, matrices m[1], m[2], and m[3] anticommute:

```
> adfmatrix(mdfmatrix(m[1],m[2]), mdfmatrix(m[2],m[1]));
```

$$[[0, 0]]$$

```
> adfmatrix(mdfmatrix(m[1],m[3]), mdfmatrix(m[3],m[1]));
```

$$[[0, 0]]$$

```
> adfmatrix(mdfmatrix(m[2],m[3]), mdfmatrix(m[3],m[2]));
```

$$[[0, 0]]$$

Thus, we can find a unique matrix representing each basis Grassmann monomial in Cl(Q) as follows. Let's define a homomorphism h from Cl(Q) to Mat(1,1,H+H) as a Maple function:

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

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

```
> h:=proc() end:
```

```
h(Id):=mdfmatrix(m[1],m[1]):
```

```
h(e1):=evalm(m[1]):
```

```
h(e2):=evalm(m[2]):
```

```
h(e3):=evalm(m[3]):
```

```
h(e1we2):=mdfmatrix(m[1],m[2]):
```

```
h(e1we3):=mdfmatrix(m[1],m[3]):
```

```
h(e2we3):=mdfmatrix(m[2],m[3]):
```

```
h(e1we2we3):=mdfmatrix(mdfmatrix(m[1],m[2]),m[3]):
```

Then, we can apply f to the entries of clibas:

```
> for x in clibas do
```

```
  x, " is mapped by h to ", h(x);
```

od;

Id , " is mapped by h to ", $[[-1, -1]]$
 $e1$, " is mapped by h to ", $[[e1, e1]]$
 $e2$, " is mapped by h to ", $[[e2, e2]]$
 $e3$, " is mapped by h to ", $[[-e1we2, e1we2]]$
 $e1we2$, " is mapped by h to ", $[[e1we2, e1we2]]$
 $e1we3$, " is mapped by h to ", $[[e2, -e2]]$
 $e2we3$, " is mapped by h to ", $[[-e1, e1]]$
 $e1we2we3$, " is mapped by h to ", $[[1, -1]]$

It is easy to see from the above display, that in order for f to be an isomorphism, there is a need to represent each element in the Grassmann basis by a pair of matrices, or, equivalently, by a matrix in a double fields. For example, let's split into a pair of matrices from $\text{Mat}(1,1,H)$ double-field matrices representing $e3$ and $e1we2$ elements:

```
> ddfmatrix(h(e3)) ; ddfmatrix(h(e1we2)) ;  
       $[[ -e1we2, e1we2 ]]$   
       $[[ e1we2, e1we2 ]]$ 
```

Notice, that the second matrix $[e1we2]$ is identical in both pairs, or, equivalently, element $e1we2$ is present in both matrices

```
> h(e3) , h(e1we2) ;  
       $[[ -e1we2, e1we2 ]]$ ,  $[[ e1we2, e1we2 ]]$ 
```

Thus, the assignment $e3 \rightarrow e1we2$, $e1we2 \rightarrow e1we2$ would not be an isomorphism. There is a need for another element to distinguish these two images by adding a second element. That is, $e3 \rightarrow [-e1we2, e1we2]$ and $e1we2 \rightarrow [e1we2, e1we2]$.

It may be worth recalling that these pairs of matrices are simply representations of the given element u from $\text{Cl}(Q)$ in two left (or right) minimal (spinor) ideals $S1$ and $S2$ generated by the primitive idempotents f and $\text{gradeinv}(f)$ respectively where 'gradeinv' is the grade involution in $\text{Cl}(Q)$. We can find f and generators for $S1 := \text{Cl}(Q)f$ from 'clidata':

```
> cdata:=clidata() ;  
cdata :=  
       $\left[ \text{quaternionic}, 1, \text{semisimple}, \frac{Id}{2} + \frac{e1we2we3}{2}, [Id, e1, e2, e3], [Id, e1, e2, e1we2], [Id] \right]$   
> f:=cdata[4] ; field:=cdata[6] ; gens:=cdata[7] ;  
       $f := \frac{Id}{2} + \frac{e1we2we3}{2}$   
       $field := [Id, e1, e2, e1we2]$   
       $gens := [Id]$ 
```

Notice that f is an idempotent:

```
> cmul(f, f) - f ;
```

0

and that f is a primitive idempotent:

> `type(f,primitiveidemp);`

true

Notice also that elements in the list 'field' generate a subalgebra in Cl(Q) isomorphic with the ring of quaternions. Let's display the multiplication table of these elements:

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

$$M := \begin{bmatrix} Id & e1 & e2 & e1we2 \\ e1 & -Id & e1we2 & -e2 \\ e2 & -e1we2 & -Id & e1 \\ e1we2 & e2 & -e1 & -Id \end{bmatrix}$$

Thus, the basis in $S1=Cl(Q)f$ over the $K = \text{gen}(\text{field}) = H$ is just f, that is, $S1=Cl(Q)f = \langle f \rangle$. Recall also that $S1 = Cl(Q)f$ is a right K module. Thus, we have

> `cmul(e3,f)=cmul(f,-e1we2); #representation in S1=Cl(Q)f`

$$\frac{e3}{2} - \frac{e1we2}{2} = \frac{e3}{2} - \frac{e1we2}{2}$$

which means that element '-e1we2' from K represents e3 in S1. Likewise, in $S2=Cl(Q)\text{gradeinv}(f)$. Recall also that $S1 = Cl(Q)\text{gradeinv}(f)$ is also a right K module.

> `fg:=gradeinv(f);`

$$fg := \frac{Id}{2} - \frac{e1we2we3}{2}$$

> `cmul(e3,fg)=cmul(fg,e1we2); #representation in S2=Cl(Q)fg`
`f(e3);`

$$\frac{e3}{2} + \frac{e1we2}{2} = \frac{e3}{2} + \frac{e1we2}{2}$$

$$\frac{1}{2} \text{Id}(e3) + \frac{1}{2} e1we2we3(e3)$$

Thus, the pair $(-e1we2, e1we2)$ represents e3 in $S1+S2$ (here '+' denotes the direct sum), which is the same as matrix m[3] above with an entry in $H+H$. Let's compute the pair representing e1we2:

> `cmul(e1we2,f)=cmul(f,e1we2); #representation in S1=Cl(Q)f`

$$-\frac{e3}{2} + \frac{e1we2}{2} = -\frac{e3}{2} + \frac{e1we2}{2}$$

> `cmul(e1we2,fg)=cmul(fg,e1we2); #representation in S2=Cl(Q)fg`

$$\frac{e3}{2} + \frac{e1we2}{2} = \frac{e3}{2} + \frac{e1we2}{2}$$

Thus, the pair $(e1we2, e1we2)$ represents e1we2 in $S1+S2$. Let's verify in a similar manner that matrices representing e1, e2, e1we3, e2we3, and e1we2we3 shown above are correct:

> `cmul(e1,f)=cmul(f,e1); cmul(e1,fg)=cmul(fg,e1); h(e1);`

$$\frac{e1}{2} - \frac{e2we3}{2} = \frac{e1}{2} - \frac{e2we3}{2}$$

$$\frac{e1}{2} + \frac{e2we3}{2} = \frac{e1}{2} + \frac{e2we3}{2}$$

$$[[e1, e1]]$$

Thus, the pair (e1,e1) represents e1 in S1+S2, which is the same as matrix m[1] above with an entry in H+H.

> **cmul (e2, f)=cmul (f, e2) ;cmul (e2, fg)=cmul (fg, e2) ;h (e2) ;**

$$\frac{e2}{2} + \frac{e1we3}{2} = \frac{e2}{2} + \frac{e1we3}{2}$$

$$\frac{e2}{2} - \frac{e1we3}{2} = \frac{e2}{2} - \frac{e1we3}{2}$$

$$[[e2, e2]]$$

Thus, the pair (e2,e2) represents e2 in S1+S2, which is the same as matrix m[2] above with an entry in H+H.

> **cmul (e1we3, f)=cmul (f, e2) ;cmul (e1we3, fg)=cmul (fg, -e2) ;h (e1we3) ;**

$$\frac{e2}{2} + \frac{e1we3}{2} = \frac{e2}{2} + \frac{e1we3}{2}$$

$$-\frac{e2}{2} + \frac{e1we3}{2} = -\frac{e2}{2} + \frac{e1we3}{2}$$

$$[[e2, -e2]]$$

Thus, the pair (e2,-e2) represents e1we3 in S1+S2.

> **cmul (e2we3, f)=cmul (f, -e1) ;cmul (e2we3, fg)=cmul (fg, e1) ;h (e2we3) ;**

$$-\frac{e1}{2} + \frac{e2we3}{2} = -\frac{e1}{2} + \frac{e2we3}{2}$$

$$\frac{e1}{2} + \frac{e2we3}{2} = \frac{e1}{2} + \frac{e2we3}{2}$$

$$[[-e1, e1]]$$

Thus, the pair (-e1,e1) represents e2we3 in S1+S2.

> **cmul (e1we2we3, f)=cmul (f, 1) ;cmul (e1we2we3, fg)=cmul (fg, -1) ;h (e1we2we3) ;**

$$\frac{Id}{2} + \frac{e1we2we3}{2} = \frac{Id}{2} + \frac{e1we2we3}{2}$$

$$-\frac{Id}{2} + \frac{e1we2we3}{2} = -\frac{Id}{2} + \frac{e1we2we3}{2}$$

$$[[1, -1]]$$

Thus, the pair (1,1) represents e1we2we3 in S1+S2.

> **printf("Worksheet took %f seconds to compute on Intel Pentium M 2.13 GHz 2GB RAM with Win XP Professional\n",time()-bench) ;**

Worksheet took 1.640000 seconds to compute on Intel Pentium M 2.13 GHz 2GB RAM with Win XP Professional

[>

[>

For more information how to find spinor representations in higher dimensions, see help pages for [Kfield](#), [gradeinv](#), [marKrepr](#), [minimalideal](#), [spinorKbasis](#), [spinorKrepr](#). See also [cbasis](#), [clidata](#), [`type/primitiveidemp`](#), [`type/idempotent`](#).

- See Also: [Clifford:-`type/dfmatrix`](#), [Clifford:-ddfmatrix](#), [Clifford:-cdfmatrix](#)

(c) Copyright 1995-2009, by Rafal Ablamowicz & Bertfried Fauser, all rights reserved.

Last modified: December 20, 2008, RA/BF.