VAGC_deriv_grad_Theorem5.6_xh.wxm

A short development document for Geometric Algebra with wxMaxima
just to test some calculus functions within the GAwxM environment,
contains...
Initialization
Loading of functions (intrinsic and GA specific)
Pseudoscalar definition (specifies the space dimension) and
Calculation of the inverse pseudoscalar used to generate the dual of a multivector
Enumeration of the standard basis for the specified dimension

Theorem 5.6, VAGC page 59 to show how the directional derivative is related to the
gradient using the vector notation for the direction, h and the coordinate, x
Also indicates how a gradient operator may be formed

Initialization

```
(%i29) ext:["wxm"]$
       file_type_maxima:append(ext,file_type_maxima)$
       batchload("initialize_fns")$
```

the pseudoscalar and its inverse
the lowest useable dimension pseudoscalar should be {e1,e2} i.e. Plen = 2
e.g. for four dimensions edit Pseudos:{e1,e2,e3}$ to Pseudos:{e1,e2,e3,e4}$

```
(%i1)  Pseudos:{e1,e2,e3}$
       Pvar:listofvars(Pseudos)$
       Plen:length(Pvar)$
       I:Pseudos$
       ni:(Plen-1)*Plen/2$
       Ii:(-1)^ni*I$
       kill(ni)$
       ldisplay(Pvar)$

 (%t8) Pvar=[e1,e2,e3]

(%i9)  batchload("initialize_lsts")$

 (%t9) lstblds=[[{e1},{e2},{e3}],[{e1,e2},{e1,e3},{e2,e3}],[{e1,e2,e3}]]
(%t10) allblds=[{e1},{e2},{e3},{e1,e2},{e1,e3},{e2,e3},{e1,e2,e3}]
(%t11) invblds=[{e1},{e2},{e3},-{e1,e2},-{e1,e3},-{e2,e3},-{e1,e2,e3}]
```

end of Initialization

set derivabbrev:false$

```
(%i12) derivabbrev:false$
```

Theorem 5.6,
VAGC page 59

coefficients of the vector, h

```
(%i13) hlst:[h1,h2,h3,0,0,0,0]$
```

coefficients of the vector, x

```
(%i14) xlst:[x1,x2,x3,0,0,0,0]$
```

form the coordinate and direction vectors from the lists of coefficients

```
(%i15) eJ:allblds$
       lenlst:2^Plen-1$
       x:0$
       h:0$
       for j:1 thru lenlst do
       block(x:x+xlst[j]*eJ[j],
       h:h+hlst[j]*eJ[j])$
       ldisplay(x,h)$

(%t20) x={e3}*x3+{e2}*x2+{e1}*x1
(%t21) h={e3}*h3+{e2}*h2+{e1}*h1
```

form a function, f(x)

```
(%i22) f(x):=x&*(x+3)$
       fx:ev(f(x))$
       ldisplay(fx)$

(%t24)/R/ fx=x3^2+3*{e3}*x3+x2^2+3*{e2}*x2+x1^2+3*{e1}*x1
```

use the function mvderiv() to form Dhf(x)

```
(%i25) fstr:"f(x)"$
       derivf:mvderiv(fstr,xlst,hlst)$
       ldisplay(derivf)$

(%t27) derivf=h3*(d/(d*x3)*f(x))+h2*(d/(d*x2)*f(x))+h1*(d/(d*x1)*f(x))
```

evaluate the directional derivative of f(x) without using the calculated value, fx

```
(%i28) Dhf:ev(derivf,diff)$
       ldisplay(Dhf)$

(%t29)/R/ Dhf=2*h3*x3+2*h2*x2+2*h1*x1+3*h3*{e3}+3*h2*{e2}+3*h1*{e1}
```

confirm that the calculated directional derivative is related to the gradient
by Equation 5.2

use the function mvgrad() to form the gradient of g, as yet undefined

```
(%i30) fstr:"g"$
       gradg:mvgrad(fstr,xlst)$
       ldisplay(gradg)$

(%t32) gradg={e3}&*(d/(d*x3)*g)+{e2}&*(d/(d*x2)*g)+{e1}&*(d/(d*x1)*g)
```

if the inner product code for &. treats the above gradient as a sum of terms of the form
base &* (scalar), then the gradient operator, gradg, may be used like a vector

form (h.grad)g

```
(%i33) hinnergrad:h&.gradg;

(%o33)/R/ (d/(d*x3)*g)*h3+(d/(d*x2)*g)*h2+(d/(d*x1)*g)*h1
```

evaluate (h.grad)g for g=f(x)

```
(%i34) g:f(x)$
       lhs:ev(hinnergrad,diff)$
       ldisplay(lhs)$

(%t36)/R/ lhs=2*h3*x3+2*h2*x2+2*h1*x1+3*{e3}*h3+3*h2*{e2}+3*h1*{e1}
```

confirm one of the equalities in Equation 5.2

```
(%i37) is(equal(lhs,Dhf));

(%o37) true
```