VAGC_grad_vector.wxm

A development document for Geometric Algebra using wxMaxima

Exercise 5.14, VAGC page 67 for the gradient of a vector function in 2D only

Initialization

```
(%i26) 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}$
       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]

(%i9)  batchload("initialize_lsts")$

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

end of Initialization

set derivabbrev:false$

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

Exercise 5.14
VAGC page 67

form the coordinate vector, x from the lists of coefficients in 2D only!

```
(%i13) xstr:"x"$
       xlst:lstvector(xstr)$
       ldisplay(xlst)$

(%t15) xlst=[x1,x2,0]

(%i16) x:makevector(xlst)$
       ldisplay(x)$

(%t17) x={e2}*x2+{e1}*x1
```

form the vector valued function

```
(%i18) f(x1,x2):=f1(x1,x2)*{e1}+f2(x1,x2)*{e2}$
       fx:ev(f(x1,x2))$
       ldisplay(fx)$

(%t20) fx={e2}*f2(x1,x2)+{e1}*f1(x1,x2)
```

Exercise 5.14, part a)

```
(%i21) Fstr:"fx"$
       divF:mvdiv(Fstr,xlst)$
       ldisplay(divF)$

(%t23) divF={e2} &. (d/(d*x2)*fx)+{e1} &. (d/(d*x1)*fx)

(%i24) divf:ev(divF,diff)$
       ldisplay(divf)$

(%t25)/R/ divf=d/(d*x1)*f1(x1,x2)+d/(d*x2)*f2(x1,x2)
```

Exercise 5.14, part b)

```
(%i26) Fstr:"fx"$
       curlF:mvcurl(Fstr,xlst)$
       ldisplay(curlF)$

(%t28) curlF={e2} &^ (d/(d*x2)*fx)+{e1} &^ (d/(d*x1)*fx)

(%i29) ev(curlF,diff)$
       curlf:facsum(%,allblds)$
       ldisplay(curlf)$

(%t31) curlf={e1,e2}*(d/(d*x1)*f2(x1,x2)-d/(d*x2)*f1(x1,x2))
```

N.B. gradF = divF + curlF for vectors, so test the function mvgrad() once again

```
(%i32) Fstr:"fx"$
       gradF:mvgrad(Fstr,xlst)$
       ldisplay(gradF)$

(%t34) gradF={e2} &* (d/(d*x2)*fx)+{e1} &* (d/(d*x1)*fx)

(%i35) ev(gradF,diff)$
       gradf:facsum(%,allblds)$
       ldisplay(gradf)$

(%t37) gradf=d/(d*x2)*f2(x1,x2)+{e1,e2}*(d/(d*x1)*f2(x1,x2)-d/(d*x2)*f1(x1,x2))+
d/(d*x1)*f1(x1,x2)

(%i38) divf+curlf$
       is(equal(gradf,%));

(%o39) true
```