

Some TI-92+ utilities for calculus III

curl computes the curl of a vector field.

The variables are always x, y, and z.

Syntax: `curl(r)` or `curl([x*y,y/z,z^2])`

Result: $[y/z^2 \ 0 \ -x]$

Code: `curl(v)`

$[[d(v[1,3],y)-d(v[1,2],z), d(v[1,1],z)-d(v[1,3],x), d(v[1,2],x)-d(v[1,1],y)]]$

div computes the divergence of a vector field.

The variables are always x, y, and z.

Syntax: `div(r)` or `div([x*y,y/z,z^2])`

Result: $y+2z+1/z$

Code: `div(v)`

$d(v[1,1],x)+d(v[1,2],y)+d(v[1,3],z)$

Note: You can enter this in your calculator by typing

define `div(v)= d(v[1,1],x)+d(v[1,2],y)+d(v[1,3],z)`

on the command line in the home screen. You can also use the program editor.

dss computes dS , for surface integrals

You supply the parameters for the surface.

Syntax: `dss(r,u,v)` or `dss([x*y,x^2,y^2],x,y)`

Result: $[4x^2y \ -2y^2 \ -2x^2]$

Code: `dss(r,x,y)`

`CrossP(d(r,x), d(r,y))`

flux computes $\mathbf{F} \bullet \mathbf{dS} = \text{dotp}(\mathbf{f}, \mathbf{dss})$. You supply the variables.

Syntax: `flux(f,r,u,v)` or `flux([u^2,v^2,u*v],[u+v,u-v,u],u,v)`

Result: $u^2-2uv+v^2$

Code: `flux(f,r,x,y)`

`DotP(f,dss(r,x,y))`

grad computes the gradient of a function

The variables are always x, y, and z.

Syntax: `grad(w)` or `grad(x^2*y-x*y^2)`

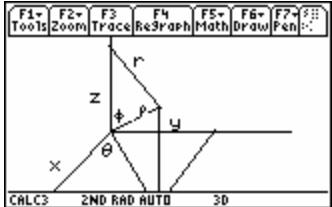
Result: $[2xy-y^2 \ x^2-2xy \ 0]$

Code: `grad(f)`

$[[d(f,x), d(f,y), d(f,z)]]$

Note: You can replace `[[` by `[` and `]]` by `]` when you type this in.

picture displays the picture that shows the relationships among rectangular, cylindrical, and spherical coordinates.



pot computes potential functions for vector fields, whether they exist or not. (Computes "potential potential functions"?) The variables are always x , y , and z .

Syntax: `pot(r)` or `pot([x,y,z])`
 Result: $(x^2+y^2+z^2)/2$
 Code: `pot(vv)`
`potfunc(vv[1,1],vv[1,2],vv[1,3])`

potfunc a utility function called by **pot**; you can also use it on its own.
 The variables are always x , y , and z .

Syntax: `potfunc(x,y,z)`
 Result: $(x^2+y^2+z^2)/2$
 Code: `potfunc(aa,bb,cc)`
 $\int(aa,x)+\int(bb,y)+\int(cc,z)-\int(\int(d(aa,y),y),x)-\int(\int(d(aa,z),z),x)-$
 $\int(\int(d(bb,z),z),y)+\int(\int(\int(d(aa,y),z),z),y),x)$

Note: The " \int " is the integral symbol, 2nd 7.

stretch computes $|dS| = v1(dss(r,u,v))$ for surface integrals.
 You supply the parameters for the surface.

Syntax: `stretch([u^2,v^2,u*v],u,v)`
 Result: $2\sqrt{x^4+4x^2y^2+y^4}$
 Code: `stretch(r,x,y)`
`v1(dss(r,x,y))`

v1 computes the length of a vector in 3-space.
 Avoids the complex conjugation of the built-in function **norm**

Syntax: `v1([3,4,t])`
 Result: $\sqrt{25+t^2}$
 Code: `v1(r)`
`\sqrt{r[1,1]^2+r[1,2]^2+r[1,3]^2}`

vsolve solves equations involving vectors
You can use this in conjunction with integration operations to solve projectile motion problems. It handles vectors with any (finite) number of components, or matrices.

Syntax: `vsolve([a^2,b,3]=[b,2,a]+[1,1,1],{a,b})`

Result: `a=2 and b=3`

Syntax: `vsolve(a*[100,<b]=[500,300],{a,b})`

Result: `a=5.83095 and b=.54042`

Code: `vsolve(veqn,ls)`

```
Func
Local vv,i,n,e
breakup(veqn)→vv
dim(ls)→n
"solve(vv,{ls[1]}→e
For i,2,n
e&","ls["&string(i)&"]"→e
EndFor
e&"}")"→e
expr(e)
EndFunc
```

breakup is a utility program called by **vsolve**
I haven't found any uses for it on its own.

Code: `breakup(veqn)`

```
Func
Local a,m,n,i,j
dim(veqn)[1]→m
dim(veqn)[2]→n
true→a
For i,1,m
For j,1,n
a and veqn[i,j]→a
EndFor
EndFor
EndFunc
```
