[6.53] Implicit derivatives

An implicit derivative results from differentiating an implicit function of the form f(x,y) = 0. Implicit differentiation is covered in all basic calculus texts, so I will skip the details here, and focus on finding implicit derivatives with the TI-89/TI-92+.

The most straight-forward method to find the implicit derivative of f(x,y) is to specify y = y(x) in the expression and use the built-in differentiation function $d()$. For example, to find the implicit derivative of

$$f(x,y) = 4 \cdot x^2 - 6 \cdot x \cdot y + 3 \cdot y^2$$

use `d(4*x^2-6*x*y(x)+3*(y(x))^2,x)`

or `d(4*x^2-6*x*y+3*y^2,x)|y=y(x)`

which both return $(6 \cdot y(x) - 6 \cdot x) \cdot \frac{d}{dx}(y(x)) - 6 \cdot y(x) + 8 \cdot x$

Note that the derivative occurs as a single term, so you can solve for it. First, copy the expression from the history area, and substitute for the derivative; I use the temporary variable *yp*:

`(6*y(x)-6*x)*d(y(x),x)-6*y(x)+8*x|d(y(x),x)=yp`

which returns `(6*yp-6)*y(x)+(8-6*yp)*x`

Now set the expression equal to zero and solve for the derivative:

`solve((6*yp-6)*y(x)+(8-6*yp)*x=0,yp)`

which returns $yp = \frac{3 \cdot y(x) - 4 \cdot x}{3 \cdot (y(x) - x)}$

You can repeat this process to find higher-order derivatives.

If you need to solve many implicit derivatives, particularly of higher orders, it is worthwhile to define a function. The following routine is a modified version of routines by David Stoutemyer and Don Phillips.

```
impdifn(ü,x,y,ñ)
Func
Local  e,e1,i

If  part(ü,0)="="            © Convert equation to expression if necessary
left(ü)-right(ü)→ü

-d(ü,x)/(d(ü,y))→e1          © Find first derivative
e1→e

For  i,2,ñ                   © Loop to find higher-order derivatives
 d(e,x)+e1*d(e,y)→e
EndFor

e                            © Return derivative.
                             © (Replace with factor(e) to automatically factor result.)
EndFunc
```

The arguments are

```
impdifn(expression, independent variable, dependent variable, order)
```

expression may be an algebraic expression, or an equation. *order* is the order of the derivative: 1 for the first derivative, 2 for the second and so on.

For example, the call to find the second derivative of x^4 + y^4 = 0 is

```
impdifn(x^4+y^4,x,y,2)
```

which returns $\frac{-3 \cdot x^6}{y^7} - \frac{3 \cdot x^2}{y^3}$

You can apply *factor()* to this expression to get $\frac{-3 \cdot x^2 \cdot (x^4 + y^4)}{y^7}$

which is zero, since x^4 + y^4 = 0.

Don Phillips modified David Stoutemyer's *impdifn()* to handle equations as input arguments, and automatically factor the result. I modified Don's routine by removing the automatic factoring of the result (you can do this manually, when needed), and converting a *Loop ... EndLoop* to a *For* loop.

If you use the first, manual method to find the derivative of an equation, this warning may be shown in the status line: *Warning: Differentiating an equation may produce a false equation*.

(Credit to Bhuvanesh Bhatt and Kevin Kofler)