Subject: Re: Multidimensional curve fitting
Posted by rivers on Sat, 20 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

In article <1995May19.085532.18482@rahman.earth.ox.ac.uk>, keith@earth.ox.ac.uk (Keith Refson) writes:
> The usual IDL fitting routines svdfit and curvefit only deal with 1-d
> functions. There is a function "sfit" which claims to perform surface
> fitting, but this can not provide uncertainties in the fit, nor even
> take account of the numerical values of x1, x2.
>
Here is an example which illustrates my previous post. It computes and fits a 2-D surface. It is necessary to "REFORM" the array to 1-D before passing it to CURVEFIT, but this is only a minor nuisance. This program uses the new version of CURVEFIT which does not require derivatives, but the old version will work the same in terms of fitting N-dimensional data.

Mark Rivers (312) 702-2279 (office)
CARS (312) 702-9951 (secretary)
Univ. of Chicago (312) 702-5454 (FAX)
5640 S. Ellis Ave. (708) 922-0499 (home)
Chicago, IL 60637 rivers@cars3.uchicago.edu (Internet)


```
**********************************************************
pro fit_curve, ind, g, pred
nx = 10
ny = 8
x = rebin(findgen(nx), nx, ny)
y = rebin(transpose(findgen(ny)), nx, ny)
pred = g(0)*(x-g(1))^2 + g(2)*(y-g(3))^2    ; Predicted surface based on
                                  ; current fit parameters
pred = reform(pred, nx*ny, /overwrite)      ; Convert back to 1-D array
end


; Main program - compute a 2-D surface and fit it
nx = 10
ny = 8
x = rebin(findgen(nx), nx, ny)
y = rebin(transpose(findgen(ny)), nx, ny)
a = [4.5, 4.7, 6, 5]                    ; Actual coefficients
g = [4.0, 3.0, 5.4, 5.9]                  ; Guess of coefficients
obs = a(0)*(x-a(1))^2 + a(2)*(y-a(3))^2     ; 2-D surface
obs = reform(obs, nx*ny, /overwrite)        ; Reform to 1-D for CURVEFIT
w = fltarr(nx*ny) + 1.                    ; Weights, all 1
ind = findgen(nx*ny)                      ; Independent variable, dummy
print, 'Actual coefficients   = ', a
```

```
print, 'Initial guess          = ', g
fit = curvefit(ind, obs, w, g, funct='fit_curve', /noderivative)
print, 'Best fit coefficients = ', g
end
```