Subject: Re: Fitting curves
Posted by larkum on Tue, 06 Sep 1994 08:03:30 GMT
View Forum Message <> Reply to Message

Thanks to the people that responded and particularly to Amara Graps for
the example procedures that solved my problem very quickly.

I thought I would post a follow-up as some may be in the same position
as I was and others may find the experience interesting. My problem
was that somehow I hadn't got the message that you were actually
supposed to _rewrite_ the built-in FUNCT procedure with the appropriate
function. It still seems kind of clumsy to me.

Anyway, during the search for information I down-loaded some archives of
comp.lang.idl-pvwave from IDLmeteo (ftp.sma.ch, /pub/idlmeteo/News_Archives)
which I will go straight to next time, since it is a source of
information better even than the FAQ. I found a discussion in which
David Hembroff had made a similar plea for help and had been given a
succint and useful reply from Eric Korpela. From his reply it appears to
me that there are different versions of CURVEFIT floating around.
Specifically, his procedure uses the keyword FUNCTION_NAME which is a
string that gives the name of the procedure to be used in place of FUNCT.
This is a much better idea, but it isn't part of the procedure we got
with our version of PV-Wave version 4.2.

On the other hand, I liked the modifications made by Amara Graps in his
version of CURVEFIT, so I put in the (trivial) code to include the
FUNCTION_NAME keyword into CURVEFIT and the resulting MYCURVEFIT is
given below.

Once again, thank you to all that replied.

```
-------------------------- Cut Here --------------------------------
;
; $Id: curvefit.pro,v 1.1 1991/03/29 12:27:07 jeffry Exp $
;
PRO MYCURVEFIT, X, Y, W, A, SIGMAA, YFIT, COVAR, function_name=function_name
;+
; NAME:
; MYCURVEFIT
; PURPOSE:
; Non-linear least squares fit to a function of an
; arbitrary number of parameters.
; Function may be any non-linear function where
; the partial derivatives are known or can be approximated.
; CATEGORY:
; E2 - Curve and Surface Fitting
; CALLING SEQUENCE:
```

```
; MYCURVEFIT,X,Y,W,A,SIGMAA,YFIT,COVAR
; INPUTS:
; X = Row vector of independent variables.
; Y = Row vector of dependent variable, same length as x.
; W = Row vector of weights, same length as x and y.
;  For no weighting
;  w(i) = 1., instrumental weighting w(i) =
;  1./y(i), etc.
; A = Vector of nterms length containing the initial estimate
;  for each parameter.  If A is double precision, calculations
;  are performed in double precision, otherwise in single prec.
;
; INPUT PARAMETERS:
; FUNCTION_NAME = Fitting function to be used.  The default
;  is the Gaussian function defined in FUNCT.
; OUTPUTS:
; A = Vector of parameters containing fit.
; Function result = YFIT = Vector of calculated
;  values.
;       Covariance matrix= error of YFIT showing correlations
; Sigmaa = Vector of standard deviations for parameters
;  A.
;
; COMMON BLOCKS:
; NONE.
; SIDE EFFECTS:
; The function to be fit must be defined and called FUNCT.
; For an example see FUNCT in the IDL User's Library.
; Call to FUNCT is:
; FUNCT,X,A,F,PDER
; Alternatively, the FUNCTION_NAME keyword can be used to
; specify a different procedure definition
; where:
; X = Vector of NPOINT independent variables, input.
; A = Vector of NTERMS function parameters, input.
; F = Vector of NPOINT values of function, y(i) = funct(x(i)), output.
; PDER = Array, (NPOINT, NTERMS), of partial derivatives of funct.
;  PDER(I,J) = Derivative of function at ith point with
;  respect to jth parameter.  Optional output parameter.
;  PDER should not be calculated if parameter is not
;  supplied in call (Unless you want to waste some time).
; RESTRICTIONS:
; NONE.
; PROCEDURE:
; Copied from "CURFIT", least squares fit to a non-linear
; function, pages 237-239, Bevington, Data Reduction and Error
; Analysis for the Physical Sciences.
;
```

```
; "This method is the Gradient-expansion algorithm which
; compines the best features of the gradient search with
; the method of linearizing the fitting function."
;
; Iterations are perform until the chi square changes by
; only 0.1% or until 20 iterations have been performed.
;
; The initial guess of the parameter values should be
; as close to the actual values as possible or the solution
; may not converge.
;
; MODIFICATION HISTORY:
; Written, DMS, RSI, September, 1982.
; Modified ;to output covariance matrix, ALG ,August 1987
; Modified ; function_name keyword, Matthew Larkum, September, 1994
;
 ;---------------------------------------------------------- -------------
;-
 if not keyword_set(function_name) then function_name = 'FUNCT'
 ON_ERROR,2  ;RETURN TO CALLER IF ERROR
 A = 1.*A  ;MAKE PARAMS FLOATING
 NTERMS = N_ELEMENTS(A) ;# OF PARAMS.
 NFREE = (N_ELEMENTS(Y)<N_ELEMENTS(X))-NTERMS ;Degs of freedom
 IF NFREE LE 0 THEN STOP,'Curvefit - not enough points.'
 FLAMBDA = 0.001  ;Initial lambda
 DIAG = INDGEN(NTERMS)*(NTERMS+1) ;SUBSCRIPTS OF DIAGONAL ELEMENTS
;
 FOR ITER = 1,20 DO BEGIN ;Iteration loop
;
;  EVALUATE ALPHA AND BETA MATRICIES.
;
 ok = execute(function_name + ',X,A,YFIT,PDER') ;COMPUTE FUNCTION AT A.
 BETA = (Y-YFIT)*W # PDER
 ALPHA = TRANSPOSE(PDER) # (W # (FLTARR(NTERMS)+1)*PDER)
;
 CHISQ1 = TOTAL(W*(Y-YFIT)^2)/NFREE ;PRESENT CHI SQUARED
;
; INVERT MODIFIED CURVATURE MATRIX TO FIND NEW PARAMETERS.
;
 REPEAT BEGIN
  C = SQRT(ALPHA(DIAG) # ALPHA(DIAG))
  ARRAY = ALPHA/C
  ARRAY(DIAG) = (1.+FLAMBDA)
  ARRAY = INVERT(ARRAY)
  B = A+ ARRAY/C # TRANSPOSE(BETA) ;NEW PARAMS
  ok = execute(function_name+',X,B,YFIT') ;EVALUATE FUNCTION
  CHISQR = TOTAL(W*(Y-YFIT)^2)/NFREE ;NEW CHISQR
  FLAMBDA = FLAMBDA*10. ;ASSUME FIT GOT WORSE
```

```
 ENDREP UNTIL CHISQR LE CHISQ1
;
      FLAMBDA = FLAMBDA/100. ;DECREASE FLAMBDA BY FACTOR OF 10
 A=B   ;SAVE NEW PARAMETER ESTIMATE.
 PRINT,'ITERATION =',ITER,' ,CHISQR =',CHISQR
 PRINT,A
 IF CHISQ1 eq 0 or ((CHISQ1-CHISQR)/CHISQ1) LE .001 THEN GOTO,DONE ;Finished?
 ENDFOR   ;ITERATION LOOP
;
 PRINT,'CURVEFIT - Failed to converge'
;
DONE: ok = execute(function_name+',X,A,YFIT,PDER')
 ALPHA = TRANSPOSE(PDER) # (W # (FLTARR(NTERMS)+1)*PDER)
 COVAR = INVERT(ALPHA)
 SIGMAA = SQRT(COVAR(DIAG))  ;RETURN SIGMA'S

END
```