Subject: minor error in SVDFIT
Posted by murthy_j on Thu, 23 Apr 1992 12:46:09 GMT
View Forum Message <> Reply to Message

There is a minor error in SVDFIT (in the user library). YFIT, the vector
of calculated y's, needs to be multiplied by the weight. The following
code corrects that and also uses the intrinsic function svdksb instead of
doing the backsubstitution by hand.

Jayant Murthy, Dept of Physics, JHU

```
FUNCTION SVDFIT,X,Y,M, YFIT = yfit, WEIGHT = weight, CHISQ = chisq, $
 SINGULAR = sing, VARIANCE = var, COVAR = covar, Funct = funct
;+
; NAME:
; SVDFIT
;
; PURPOSE:
; Perform a general least squares fit with optional error estimates.
;
; This version uses SVD.  A user-supplied function or a built-in
; polynomial is fit to the data.
;
; CATEGORY:
; Curve fitting.
;
; CALLING SEQUENCE:
; Result = SVDFIT(X, Y, M)
;
; INPUTS:
; X: A vector representing the independent variable.
;
; Y: Dependent variable vector.  This vector should be same length
;  as X.
;
; M: The number of coefficients in the fitting function.  For
;  polynomials, M is equal to the degree of the polynomial + 1.
;
; OPTIONAL INPUTS:
; Weight: A vector of weights for Y(i).  This vector should be the same
;  length as X and Y.
;
;  If this parameter is ommitted, 1 is assumed.  The error for
;  each term is weighted by Weight(i) when computing the fit.
;  Frequently, Weight(i) = 1./Sigma(i) where Sigma is the
;  measurement error or standard deviation of Y(i).
```

```
;
; Funct: A string that contains the name of an optional user-supplied
;  basis function with M coefficients. If omitted, polynomials
;  are used.
;
;  The function is called:
;   R = FUNCT(X,M)
;  where X is an N element vector, and the function value is an
;  (N, M) array of the N inputs, evaluated with the M basis
;  functions.  M is analogous to the degree of the polynomial +1
;  if the basis function is polynomials.  For example, see the
;  function COSINES, in the IDL User Library, which returns a
;  basis function of:
;   R(i,j) = cos(j*x(i)).
;  For more examples, see Numerical Recipes, page 519.
;
;  The basis function for polynomials, is R(i,j) = x(i)^j.
;
; OUTPUTS:
; SVDFIT returns a vector of M coefficients.
;
; OPTIONAL OUTPUT PARAMETERS:
; NOTE:  In order for an optional keyword output parameter
; to be returned, it must be defined before calling SVDFIT.
; The value or structure doesn't matter.  For example:
;
;  YF = 1    ;Define output variable yf.
;  C = SVDFIT(X, Y, M, YFIT = YF)  ;Do SVD, fitted Y vector is now
;      ;returned in variable YF.
;
; YFIT: Vector of calculated Y's.
;
; CHISQ: Sum of squared errors multiplied by weights if weights
;  are specified.
;
; COVAR: Covariance matrix of the coefficients.
;
;    VARIANCE: Sigma squared in estimate of each coeff(M).
;
;    SINGULAR: The number of singular values returned.  This value should
;  be 0.  If not, the basis functions do not accurately
;  characterize the data.
;
; COMMON BLOCKS:
; None.
;
; SIDE EFFECTS:
; None.
```

```
;
; MODIFICATION HISTORY:
; Adapted from SVDFIT, from the book Numerical Recipes, Press,
; et. al., Page 518.
; minor error corrected April, 1992 (J.Murthy)
;-
; ON_ERROR,2  ;RETURN TO CALLER IF ERROR
; set variables
 THRESH = 1.0E-9  ;Threshold used in editing singular values


 XX = X*1.  ;BE SURE X IS FLOATING OR DOUBLE
 N = N_ELEMENTS(X)  ;SIZE
 IF N NE N_ELEMENTS(Y) THEN BEGIN ;SAME # OF DATA POINTS.
   message, 'X and Y must have same # of elements.'
   ENDIF

 if n_elements(weight) ne 0 then begin
  if n_elements(weight) ne n then begin
    message, 'Weights have wrong number of elements.'
    endif
   b = y * weight ;Apply weights
   endif else b = y ;No weights
;

 if n_elements(funct) eq 0 then begin ;Use polynomial?
   A = FLTARR(N,M)   ;COEFF MATRIX
   if n_elements(weight) ne 0 then xx = float(weight) $
   else xx = replicate(1.,n) ;Weights are 1.
   for i=0,m-1 do begin  ;Make design matrix
  a(0,i) = xx
  xx = xx * x
  endfor
 endif else begin  ;Call user's function
  z = execute('a='+funct+'(x,m)')
  if z ne 1 then begin
   message, 'Error calling user fcn: ' + funct
   endif
  if n_elements(weight) ne 0 then $
    a = a * (weight # replicate(1.,m)) ;apply wts to A
 endelse

 svd,a,w,u,v   ;Do the svd

 good = where(w gt (max(w) * thresh), ng) ;Cutoff for sing values
 sing = m - ng  ;# of singular values
 if sing ne 0 then begin
  message, 'Warning:' + strcompress(sing, /REMOVE) + $
```

```
   ' singular values found.'
;modified J.M.
small=where(w le max(w)*thresh)
w(Small)=0
;
  if ng eq 0 then return,undefined
  endif
;modified J.M

svbksb,u,w,v,b,coeff
;
 wt = fltarr(m)
 wt(good)=1./w(good)

 if (n_elements(yfit) ne 0) or (n_elements(chisq) ne 0) then begin
   if n_elements(funct) eq 0 then yfit = poly(x,coeff) $
  else begin
  yfit = fltarr(n)
  for i=0,m-1 do yfit = yfit + coeff(i) * a(*,i) $
   / weight ;corrected J.M.
  endelse
   endif

 if n_elements(chisq) ne 0 then begin ;Compute chisq?
  chisq = (y - yfit)
  if n_elements(weight) ne 0 then chisq = chisq * weight
  chisq = total(chisq ^ 2)
  endif

 wt = wt*wt  ;Use squared w

 if n_elements(covar) ne 0 then begin ;Get covariance?
  covar = fltarr(m,m)
  for i=0,m-1 do for j=0,i do begin
    s = 0.
    for k=0,m-1 do s = s + wt(k) * v(i,k) * v(k,j)
    covar(i,j) = s
    covar(j,i) = s
  endfor
   endif

 if n_elements(var) ne 0 then begin
  var = fltarr(m)
  for j=0,m-1 do for i=0,m-1 do $
     var(j) = var(j) + v(j,i)^2 * wt(i)
  endif

 return,coeff
```

end