Subject: Fast linear least sqares fitting beta test
Posted by dudley on Sat, 28 May 1994 06:58:15 GMT
View Forum Message <> Reply to Message

The following is a translation from Fortran 90 of the fast fitting program
given in a new Rybicki and Press preprint. Please let me know if you find
and bugs.

Chris

```
;+
; NAME:
;
;Fastfit
;
; PURPOSE:
;
;Linear least squares type fitting of basis to data with arbitrary
;absissa.
;
; CATEGORY:
;
;New numerical recipies based on Ribicki and Press 5/94 preprint "A
;Class of Fast Methods for Processing Irregularly Samples or Otherwise
;Inhomogenious One-Dimentional Data". The preprint can be obtained by sending
;mail to comp-gas@xyz.lanl.gov with the subject line 'get 9405004'.
;
; CALLING SEQUENCE:
;
; fit=fast_fitting(x,y,sig,psig,pw,xel,el,q,chisq)
;
; INPUTS:
;
; x,y,sig: data x,y, and yerror bars.
; psig: estimated population sigma
; pw: estimated inverse decorrelation distance.
; xel: xvalues for basis matrix el
; el: basis matrix
;
; OPTIONAL INPUTS:
;
;
;
; KEYWORD PARAMETERS:
;
; /usesvd specifies the use of sigular value decomposition and back
; substitution rather than Gauss-Jordan elimination.
;
```

```
; OUTPUTS:
;
; fit: y values of fit at xel
; q: solution vector (fit=el#q)
; chisq: fit chi^2
;
; OPTIONAL OUTPUTS:
;
;
;
;
; COMMON BLOCKS:
;
;
;
;
; SIDE EFFECTS:
;
;
;
;
; RESTRICTIONS:
;
; You are responsible to see that x and xel are unique and that the
; basis of el are independent.  This is a beta test.  Please let me
; know if other problems arise.
;
; PROCEDURE:
;
; Suppose you have an irregularly sampled spectrum with an SRF FWHM of
; .2 microns.  guess psig=<sig> and pw=5.
;
; EXAMPLE:
;
; s=23
; a=findgen(1000)+randomu(s,1000)-.5
; c=randomu(s,1000)
; d=randomu(s,20)*5
; d=rebin(abs(d-2.5),1000)
; b=smooth(sin(a/30.)+(c-.5)*d,4.)
;
; el=fltarr(1000,4)
; el(*,0)=1.
; el(*,1)=sin(findgen(1000)/25.)
; el(*,2)=sin(findgen(1000)/30.)
; el(*,3)=sin(findgen(1000)/35.)
; fit=fastfit(a,b,d,.6,.25,findgen(1000),el,q,chisq)
; plot,a,b
; oplot,fit
;
; MODIFICATION HISTORY:
```

```
; adapted from Rybicki and Press preprint May 23 1994 by C. dudley
; dudley@galileo.ifa.hawaii.edu
;-


PRO Gaussj, a, b
n = size(a)
m = size(b)

indxc = indgen(n(1))
indxr = indgen(n(1))
ipiv = intarr(n(1))

FOR i = 0, n(1)-1 DO BEGIN
   big = 0.
   FOR j = 0, n(1)-1 DO BEGIN
      IF ipiv(j) NE 1 THEN BEGIN
         FOR k = 0, n(1)-1 DO BEGIN
            IF ipiv(k) EQ 0 THEN BEGIN
               IF abs(a(j, k)) GE big THEN BEGIN
                  big = abs(a(j, k))
                  irow = j
                  icol = k
               ENDIF
            ENDIF ELSE IF ipiv(k) GT 1 THEN BEGIN
               print, 'singular matrix-1'
               return
            ENDIF

         ENDFOR
      ENDIF
   ENDFOR

   ipiv(icol) = ipiv(icol)+1


   IF irow NE icol THEN BEGIN
      dumm = a(irow, *)
      a(irow, *) = a(icol, *)
      a(icol, *) = dumm
      dumm = b(irow, *)
      b(irow, *) = b(icol, *)
      b(icol, *) = dumm
   ENDIF
   pivinv = 1./a(icol, icol)
   a(icol, icol) = 1.
   a(icol, *) = a(icol, *)*pivinv
   b(icol, *) = b(icol, *)*pivinv
```

```
   FOR ll = 0, n(1)-1 DO BEGIN
      IF ll NE icol THEN BEGIN
         dum = a(ll, icol)
         a(ll, icol) = 0.0
         a(ll, *) = a(ll, *) -a(icol, *)*dum
         b(ll, *) = b(ll, *) -b(icol, *)*dum
      ENDIF
   ENDFOR
ENDFOR



FOR l = n(1)-1, 0, -1 DO BEGIN
   IF indxr(l) NE indxc(l) THEN BEGIN
      dumm = a(*, indxr(l))
      a(*, indxr(l)) = a(*, indxc(l))
      a(*, indxc(l)) = dumm
   ENDIF
ENDFOR

END


FUNCTION Tridiag, a, b, c, r

n = n_elements(b)

bet = b(0)
IF bet EQ 0. THEN BEGIN
   print, 'error 1 in tridiag'
   return, -1
ENDIF ELSE BEGIN
   gam = fltarr(n)
   u = fltarr(n)
   u(0) = r(0)/bet
   eflag = 0
   FOR j = 1, n-1 DO BEGIN
      gam(j) = c(j-1)/bet
      bet = b(j)-a(j-1)*gam(j)
      IF bet EQ 0. THEN eflag = 1 ELSE u(j) = (r(j)-a(j-1)*u(j-1))/bet
   ENDFOR
   IF eflag eq 1 THEN BEGIN
      print, 'error 2 in tridiag'
      return, -1
   ENDIF
   FOR j = n-2, 0, -1 DO u(j) = u(j)-gam(j+1)*u(j+1)

   return, u
```

```
ENDELSE
END ;tridiag

FUNCTION symtrimul, a, b, u

n = n_elements(u)

result = fltarr(n)

result(0) = b(0)*u(0)+a(0)*u(1)
result(1:n-2) = a(0:n-3)*u(0:n-3)+b(1:n-2)*u(1:n-2)+a(1:n-2)*u(2:n-1)
result(n-1) = a(n-2)*u(n-2)+b(n-1)*u(n-1)
return, result
END ;symtrimul



FUNCTION Apply_cinv, x, y, sig, psig, pw

n = n_elements(x)
m = size(y)
tmp = sig^2

b = fltarr(n)
r = exp(-abs(pw*(x(1:n-1)-x(0:n-2))))
g = where(1./r-r GT 1e-4)
ng = where(1./r-r LE 1e-4, count)
a = r*0
a(g) = -1./(1./r(g)-r(g))
IF count NE 0 THEN a(ng) = -1./1e-4
r = r*a
b(0) = 1.-r(0)
b(1:n-2) = 1.-r(1:n-2)-r(0:n-3)
b(n-1) = 1.-r(n-2)
a = a/psig^2
b = b/psig^2
;stop
aa = a*tmp(1:n-1)
bb = b*tmp+1.
cc = a*tmp(0:n-2)
result = y*0.
FOR j = 0, m(2)-1 DO BEGIN
   temp = tridiag(aa, bb, cc, y(*, j))
;tridag, [[0], aa], bb, [cc, [0]], y(*, j), temp
   result(*, j) = symtrimul(a, b, temp)
ENDFOR
return, result
```

```
END ;apply_cinv

FUNCTION Fastfit, x, y, sig, psig, pw, xel, el, q, chisq, usesvd=usesvd

n = n_elements(x)
nl = n_elements(xel)
m = size(el)

IF (n_elements(y) NE n OR n_elements(sig) NE n OR m(1) NE nl ) THEN BEGIN
print, 'size errors', n,nl, m
return, -1
ENDIF

y = y(sort(x))
sig = sig(sort(x))
x = x(sort(x))
el = el(sort(xel), *)
xel = xel(sort(xel))
nt = n+nl
fl = bytarr(nt)
fl(0:n-1) = 1
xx = [x, xel]
order = sort(xx)
xx = xx(order)
ell = fltarr(nt, m(2)+1)
ell(0:n-1, m(2)) = y
ell(n:*, 0:m(2)-1) = el
ell(*, *) = ell(order, *)
sigg = [sig, fltarr(nl)]
sigg = sigg(order)

cil = apply_cinv(xx, ell, sigg, psig, pw)
qq = fltarr(m(2), 1)
qq(*, 0) = transpose(ell(*, 0:m(2)-1))#cil(*, m(2))
;stop
IF keyword_set(usesvd) THEN BEGIN
   svd, transpose(ell(*, 0:m(2)-1))#cil(*, 0:m(2)-1), w, u, v

   small = where(w LT max(w)*1e-6, count)
   IF count NE 0 THEN w(small) = 0.
   svbksb, u, w, v, -qq, q
ENDIF ELSE BEGIN
   gaussj, transpose(ell(*, 0:m(2)-1))#cil(*, 0:m(2)-1), qq
   q = -qq(*, 0)
ENDELSE


ell(*, 0) = ell(*, m(2))+ell(*, 0:m(2)-1)#q
```

```
cil(*, 0) = cil(*, m(2))+cil(*, 0:m(2)-1)#q
chisq = total(ell(*, 0)*cil(*, 0))
return, el#q
END
```