Subject: Re: Voigt
Posted by agraps on Tue, 10 Sep 1996 07:00:00 GMT
View Forum Message <> Reply to Message

mike@atm.ch.cam.ac.uk (Mike Burton) writes:

> I would like to use the in built VOIGT function in IDL 4.0.1
> on Solaris 2 but for certain lorentzian and doppler halfwidths
> the function fails. Is there another algorithm that someone
> else has coded in IDL?

Here is an IDL routine for computing the Voigt function via the complex error function and rational polynomials. The Voigt function is the real part of the complex error function.

The code snippet was from the following reference:

Hui, Armstong, and Wray, JQSRT, vol 19, pg 509, 1978.

(I know that there are references more recent than this, but I don't have them handy. There were many in JQSRT when I did a literature search on Voigt function computation 3 or 4 years ago.)

Since Dave Stern wrote the first version, this procedure may be just a variation of the IDL built-in Voigt, with more control of baseline, easy calculation of derivatives, etc. I used this routine to fit spectra via Marquardt-Levenberg curvefitting.

I haven't used this procedure in a few years but it still worked the last time I tried it.

Hope this helps,

Amara

```
;********************************************************** ****
PRO VOIGT2, x0, v, vmask, pder, yfit
;+
; Name:
; VOIGT2
;
; PURPOSE:
;   Evaluates approximation to Voigt function and generates a fit to
;       one Voigt line plus linear baseline (transformed from
;       direct to transmission space).
;
; CATEGORY:
```

```
;       E2 - Curve And Surface Fitting.
;
; CALLING SEQUENCE:
;       VOIGT2, x0, v, vmask, yfit, pder
;
; INPUTS:
;       x0 = Values of independent wavenumber variable.
;       v = Params of equation described below.
;       vmask = Mask of values which are "1" where pder is to be calculated.
;
; OUTPUTS:
;       yfit = Value of function at each x0(i).
;
; OPTIONAL OUTPUT PARAMETERS:
;       Pder = (N_ELEMENTS(x0),TOTAL(vmask)) array containing the
;               partial derivatives.  Pder(i,j) = derivative
;               at ith point w/respect to jth parameter.
;
; COMMON BLOCKS:
;       None
;
; SIDE EFFECTS:
;       None.
;
; RESTRICTIONS:
;       None.
;
; PROCEDURE:
; As described in Hui, Armstrong and Wray; JQSRT, vol. 19 pp. 509; 1978
;       R = Num/Den   =   Rational Fraction ~ To Cmplx Error Function.
; F = Double(R) extracts just the real part & makes it double precision.
; Imaginary(R) gives you the partial derivatives.
;
;MODIFICATION HISTORY:
;   Written, DMS, RSI, Sept, 1982.
;   Revised By T. Sauke, May 1990.
;   Revised By T. Sauke, V. Dinh & T. Gutierrez, April 1991.
;   Revised by A. Graps to output covariance matrix, make compatible
;   with my Marquardt algorithm (marquardt.pro) and vfitx.pro (x=1,2,3)
;   August 1992
;
;Amara Graps Sterling Software/NASA-Ames Updated: 10-16-92
;
;-
;Define coefficients for rational function approximation
a0 = 122.607931777104326D0
a1 = 214.382388694706425D0
a2 = 181.928533092181549D0
```

```
a3 = 93.155580458138441D0
a4 = 30.180142196210589D0
a5 = 5.912626209773153D0
a6 = 0.564189583562615D0
b0 = 122.607931773875350D0
b1 = 352.730625110963558D0
b2 = 457.334478783897737D0
b3 = 348.703917719495792D0
b4 = 170.354001821091472D0
b5 = 53.992906912940207D0
b6 = 10.479857114260399D0

;Compute yfit- the Voigt Function
sqln2 = SQRT(ALOG(2.0D0))
onbyrtpi = 1.0D0/SQRT(!dpi)
y = DBLARR(N_ELEMENTS(x0))+v(2)*sqln2/v(3)
x = DOUBLE((x0-v(1))*sqln2/v(3))
zh = COMPLEX(y,-x)

num = ((((((a6*zh+a5)*zh+a4)*zh+a3)*zh+a2)*zh+a1)*zh+a0)
den = (((((((zh+b6)*zh+b5)*zh+b4)*zh+b3)*zh+b2)*zh+b1)*zh+b0)
r = num/den       ; complex error function.
u = DOUBLE(r)   ;extracts just the real part and makes it dbl precision
sf = sqln2 * onbyrtpi
expu = EXP(-v(0)*(sf/v(3))*u)
yfit = (v(4)+v(5)*x0)*expu

;Compute Partial Derivatives: pder(,)

vi = IMAGINARY(r)
dudx = -2*(x*u-y*vi)
dudy = -2*(onbyrtpi-x*vi-y*u)

jj=0

;dFdv0...
IF vmask(0) eq 1 THEN pder(0,jj) = -sf / v(3) * yfit * u
jj=jj+vmask(0)

;dFdv1...
IF vmask(1) eq 1 THEN BEGIN
 t1 = v(3)^2
 pder(0,jj) = v(0) * yfit * sf / t1 * dudx * sqln2
 END ;dFdv1
jj=jj+vmask(1)

;dFdv2...
IF vmask(2) EQ 1 THEN BEGIN
```

```
 t1 = v(3)^2
 pder(0,jj) = -v(0) * yfit * sf / t1 * sqln2 * dudy
 END ;dFdv2
jj=jj+vmask(2)

;dFdv3...
IF vmask(3) EQ 1 THEN BEGIN
 t1 = v(3)^2
 pder(0,jj) = yfit * v(0)* sf / t1 * (dudy*y + dudx*x + u)
 END ;dFdv3
jj=jj+vmask(3)


;dFdv4...
IF vmask(4) eq 1 THEN pder(0,jj) = EXP(-v(0) * sf / v(3) * u)
jj=jj+vmask(4)

;dFdv5...
IF vmask(5) EQ 1 THEN pder(0,jj) = x0 * EXP(-v(0) * sf / v(3) * u)
jj=jj+vmask(5)

END ;of procedure VOIGT2

 ;************************************************************ ****

--

 ************************************************************* *************
Amara Graps                    email: agraps@netcom.com
Computational Physics             vita:  finger agraps@best.com
Multiplex Answers              URL:   http://www.amara.com/
 ************************************************************* *************
```

Subject: Re: Voigt
Posted by Amara Graps on Tue, 10 Sep 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Sorry, my previous explanation wasn't complete. You would
need to know what the Voigt function parameters are.

```
v = dblarr(6) ;holds estimates of Voigt function parameters

v(0) Line strength
v(1) Line position in terms of wavenumber array
v(2) Lorentz width
v(3) Doppler width
```

v(4) Line intercept
v(5) Line slope

x0(i) = array of wavenumbers

Then
yfit = (v(4) + v(5)* x0) * expu

Where "expu" is the rational function approximation
to the Voigt function described in the previous code.

Amara

--

 ** *****
Amara Graps amara@quake.stanford.edu
Solar Oscillation Investigations Stanford University
http://quake.stanford.edu/~amara/amara.html
 ** *****