Subject: Re: Using Errorf in IDL
Posted by jackel[1] on Sun, 25 May 1997 07:00:00 GMT
View Forum Message <> Reply to Message

In article <5lsqed$efs@agate.berkeley.edu> karl@gojira.berkeley.edu (Karl Young) writes:

> I recently tried to feed a complex argument to Errorf, but it would have
> none of it. With Mathematica you get a complex numerical result
> if you feed Erf a complex numerical argument. I assume Mathematica is
> calculating the principal value of the integral or something like
> that. So my basic question is, how can I do the same thing most
> efficiently in IDL? (If I have to I can numerically integrate
> Fresnel integral euivalents or something else equally disgusting
> but I'm praying for something simple)

Somebody else has already posted a response, but here's my
two cents worth:

```
;Distribute freely, use at your own risk
;Bug reports cheerfully accepted: jackel@danlon.physics.uwo.ca
;+
; NAME:        ErrorfW
;
; PURPOSE:      Evaluates the "W" function (Abramowitz & Stegun 7.1.3)
;               for complex arguments.  The "W" function is useful for
;               calculating the plasma dispersion function, and can also
;               be used as a basis for calculating other functions.
;
;               For example, the Error function for complex arguments is
;               related to the "W" function by
;
;                    W(z)= EXP(-z^2) * ( 1.0 - Erf(-iz) )
;
;               As well, the Fresnel integrals can be expressed in terms
;               of the "W" function.
;
; CATEGORY:     Math, Special functions
;
;
; CALLING SEQUENCE:     Result= ErrorfW(Arguement,[Order])
;
; INPUTS:   Argument   the (possibly complex) argument for the "W" function.
;                      May be a scalar or array.
;
; OPTIONAL INPUTS: Order   the number of terms in the series expansion
;                      used to calculate the result (must be power of 2).
;                      Default is 16.
;
```

```
; OUTPUTS:      Result     the (probably complex) result.
;
;
; KEYWORDS:     ERRORF     if this keyword is set, then the function returns
;                          the error function for complex arguments.
;
;             FRESNEL    if this keyword is set, then the function returns
;                          the C and S Fresnel integrals as the real and
;                          imaginary parts of the result.
;
;             Note: if both these keywords are set, the result will be
;                  the Fresnel integrals.
;
;             DOUBLE     if set, forces double precision arithmetic
;
;             HELP   if set, DOC_LIBRARY is called to return this documentation.
;
;
; COMMON BLOCKS:  ERRORFW,coefficients     which contains the coefficients for
;                                  the series.
;
; RESTRICTIONS: may lose precision near the zeros in the negative imaginary
;             portion of the complex plane.  Note that with purely real
;             arguments and the /ERRORF switch set the results are identical
;             to the intrinsic routine ERRORF(), but take about 10x as long.
;
; PROCEDURE:    As given in "Computation of the Complex Error Function"
;                          by J.A.C. Weideman, SIAM J. Numer. Anal.
;                          Vol. 31, No. 5, pp. 1497-1518, October 1994
;
;             Basically a series expansion in a scaled variable, with a
;             neat trick of using an FFT to get coefficient values.  Only
;             works in the upper half plane (positive imaginary values),
;             so the relation w(-z)= 2*exp(-z^2)-w(z) may have to be used.
;             Highly vectorized, so useful in IDL.
;
; EXAMPLES:
;
;The simplest example:
;
;     z= COMPLEX(0.1,0.2)
;     result= ErrorfW(z)              ;(0.802567,0.080029) in Abromowitz & Stegun
;
;To evaluate the first zero of the error function:
;
;     z= COMPLEX(1.45061616,1.88094300)
;     result= ErrorfW(z,/ERRORF)              ;should be within 1.0E-6 of zero
;
```

```
;To get the Fresnel integrals over a range of real arguments
;
;     x= FINDGEN(80)/20.0
;     result= ErrorfW(x,/FRESNEL)
;     c= FLOAT(result)                    ; the "C" Fresnel integral
;     s= IMAGINARY(result)                    ; the "S" Fresnel integral
;     PLOT,x,c & OPLOT,x,s                    ; Figure 7.5 in Abromowitz & Stegun
;
;
; MODIFICATION HISTORY:
; Brian Jackel    July 3 1993
; University of Western Ontario, jackel@danlon.physics.uwo.ca
;          Using "Efficient Computation of the Complex Error Function"
;          Walter Gautschi, SIAM J. Numer. Anal.
;          Vol 7. No. 1, March 1970, pp 187-198.
;
; Brian Jackel  March 30 1995
;          Replaced Gautschi algorithm with Weideman's.  This greatly
;          simplifies the code, and speeds up execution by a factor
;          of 10.  For standard precision, the results are identical
;          to 6 decimal places, which is sufficient for IDL.
;
; Brian Jackel  June 9 1995
;          Included /DOUBLE keyword which forces double precision arithmetic,
;          added tests for double precision complex values (IDL 4.0 and later).
;
; POSSIBLE IMPROVEMENTS:
;    -introduce a test for regions where more terms may be required,
;     then recursively pass these values to a higher order evaluator.
;
;-
FUNCTION  ErrorfW,argument,order,ERRORF=errorf,FRESNEL=fresnel,DOUBLE=
double,HELP=help,TEST=test

 ON_ERROR,2

 IF KEYWORD_SET(HELP) THEN BEGIN
   DOC_LIBRARY,"ErrorfW"
   RETURN,-1
 END

 IF KEYWORD_SET(TEST) THEN BEGIN
    x= RANDOMN(seed,512)
    y= RANDOMN(seed,512)
    z= DCOMPLEX(x,y)
    diff= ERRORFW(z,test) - ERRORFW(z,16)
    RETURN,[[z],[diff]]
 ENDIF
```

```
 IF (N_PARAMS() LT 1) THEN MESSAGE,"This function requires at least one input value"

 siz= SIZE(argument)
 typecode= siz( siz(0) + 1)
 IF (typecode EQ 5) OR (typecode EQ 9) OR KEYWORD_SET(DOUBLE) THEN BEGIN
    one= 1.0D0
    two= 2.0D0
    pi= !dpi
 ENDIF ELSE BEGIN
    one= 1.0
    two= 2.0
    pi= !pi
 ENDELSE

 i= COMPLEX(0.0,one)          ;square root of negative one
 z= COMPLEX(argument)          ;make sure that the input value is complex


  ;------------------------------------------------------------ -------------
 ;For a given order (# of terms to sum), we only need to calculate the
 ;coefficients once.  So, we keep them in a common block, and only
 ;re-calculate them if the order changes.
 ;
 COMMON ERRORFW,coefficients

 IF (N_PARAMS() GT 1) THEN n= (ABS(order) > 4) < 128 ELSE n=16

 L= SQRT(n/SQRT(2.0))   ;a scaling parameter used to calculate the coefficients
 IF (N_ELEMENTS(coefficients) NE n) THEN BEGIN
   m= two*n
   k= FINDGEN(2*m-1) - (m-1)
   theta= pi*k/m
   t= L*tan(theta/two)
   f= EXP(-t^2)*(L^2+t^2)
   f= [0,f]
   a= FFT( SHIFT(f,m) ,-1)
   coefficients= a(1:n)
   dummy= CHECK_MATH()     ;We expect an underflow error in the EXP above, catch it and
throw it away
 ENDIF
  ;------------------------------------------------------------ ------------


  ;------------------------------------------------------------ ------------
 ;The Error function for complex argument is related to W(z) by:
 ; erf(z) = 1 - exp(-z^2)*W(iz)
 ;so if we want it, we just have to multiply the argument by i now,
```

```
;and then do the remaining calculations at the end.
;
;The Fresnel integrals are most easily expressed in terms of the
;Error function with a scaled argument.  So, we scale z, then
;do the Error function multiplication by i as well.
;
IF KEYWORD_SET(FRESNEL) THEN z= ( SQRT(pi)/two*(one-i) ) * z
IF (KEYWORD_SET(ERRORF) OR KEYWORD_SET(FRESNEL)) THEN z= i*z
 ;------------------------------------------------------------ ------------



 ;------------------------------------------------------------ ------------
;Since this algorithm only applies for non-negative imaginary
;values, we have to flip all the negative imaginary values here
;(keeping track of what we did), calculate W(z), then use
;W(-z)= 2.0*EXP(-z^2) - W(z) wherever necessary
;
neglist= WHERE( IMAGINARY(z) LT 0.0, nneg)
IF (nneg NE 0) THEN z(neglist)= -z(neglist)
 ;------------------------------------------------------------ -------------



 ;------------------------------------------------------------ -------------
;Here's the actual calculation.  We construct a scaled version
;of z, then sum the series.  Note that we sum in reverse order,
;in the hopes that the (smaller) high order terms will add up
;to something substantial before the (much bigger) low order
;terms swamp them.
;
one_over_lminus= one / (L - i*z)
z_scaled= (L+i*z) * one_over_lminus
w= coefficients(n-1)
FOR indx=n-2,0,-1 DO w= z_scaled*TEMPORARY(w) + coefficients(indx)
w= two*w * one_over_lminus^2 + (one/SQRT(pi))*one_over_lminus
 ;------------------------------------------------------------ -----------------



 ;------------------------------------------------------------ -----------------
;Now we clean up a bit.
;First correct all W(z) that originally had z's in the lower half plane (negative imaginary parts).
;Then, if either ERRORF or FRESNEL were set, then convert W(z) to ERF(z).
; Note here that (-z)^2 = z^2, so we don't have to restore the original z's.
;Finally, if FRESNEL was set, then convert ERF(z) to C(z) + iS(z)
;
IF (nneg NE 0) THEN w(neglist)= two*EXP(-z(neglist)^2) - w(neglist)
;IF (nneg NE 0) THEN z(neglist)= -z(neglist)
IF (KEYWORD_SET(ERRORF) OR KEYWORD_SET(FRESNEL)) THEN w= one - EXP(z^2)*w
IF KEYWORD_SET(FRESNEL) THEN w= ((one+i)/two) * w
```

```
;------------------------------------------------------- -----------------

RETURN,w
END
```