
Subject: Re: Confluent Hypergeometric Function of the First Kind
Posted by [noahh.schwartz](#) on Fri, 22 Feb 2008 10:36:05 GMT

[View Forum Message](#) <> [Reply to Message](#)

On 21 fév, 17:26, Dan Larson <dlar...@aecom.yu.edu> wrote:

> On Feb 21, 9:30 am, Spon <christoph.b...@gmail.com> wrote:

>

>

>

>> On Feb 20, 3:44 pm, noahh.schwa...@gmail.com wrote:

>

>>> Hello everyone,

>

>>> I am looking for the ConfluentHypergeometricFunction of the First
>>> Kind in the DLMath Library but it does not seem to be implemented!

>

>>> I would like to use a function similar to the Hypergeometric1F1[a, b,
>>> z] of Mathematica [[http://reference.wolfram.com/mathematica/ref/
>>> Hypergeometric1F1.html](http://reference.wolfram.com/mathematica/ref/Hypergeometric1F1.html)].

>

>>> I have not found what I was looking for, and so decided to try to code
>>> it my self... [sigh...]. Being a fresh beginner in IDL this is a hard
>>> task!

>

>>> Would anybody know how to code an infinite series expansion like the
>>> Hypergeometric1F1?

>

>>> Thank you in advance for your time!

>>> Noah

>

>> Noah,

>

>> here's my attempt. It accepts only scalar inputs for A and B, while Z
>> can be a vector. I've tested it for the examples on the mathematica
>> site and it seems to give correct results, and works correctly for
>> complex input too as far as I can tell. 'Precision' is an input
>> variable to specify how close two successive iterations have to be
>> before the function assumes they are the same and aborts the while
>> loop. Default is 7 (i.e. stop when results differ by 10^{-7} or less).
>> If you're finding this programme is running very slow, try decreasing
>> the precision (I was surprised how fast it runs despite the while
>> loop, actually!)

>

>> Ideally the input parameters should all be double precision before you
>> make the call to the function, but the function converts them if
>> they're not.

>

>> If you want all your inputs to be vectors (not just Z), I'm sure it

```

>> can be done, but it'd be a bit more complicated. :-)
>
>> Take care,
>> Chris
>
>> FUNCTIONHYPERGEOMETRICONEFONE, A, B, Z, $
>> PRECISION = Precision, $
>> K = K ; K is an output parameter to count No. of WHILE loops
>> performed.
>
>> ; References:
>> ; http://reference.wolfram.com/mathematica/ref/Hypergeometric1F1.html
>> ; http://en.wikipedia.org/wiki/Confluent\_hypergeometric\_functi on
>
>> IF N_PARAMS() NE 3 THEN MESSAGE, 'Must input A, B & Z as 3 input
>> parameters.'
>> IF N_ELEMENTS(A) GT 1 THEN MESSAGE, 'Variable A must be a scalar.'
>> IF N_ELEMENTS(B) GT 1 THEN MESSAGE, 'Variable B must be a scalar.'
>
>> A *= 1.0D ; Double precision or double complex scalar
>> B *= 1.0D ; Double precision or double complex scalar
>> Z *= 1.0D ; Double precision or double complex scalar or vector
>> IF N_ELEMENTS(Precision) EQ 0 THEN $
>> Precision = 7L ELSE $
>> Precision = (LONG(Precision))[0]
>> Cutoff = 10D^(-1D * Precision) > (MACHAR()).EPS ; Cutoff can't be
>> smaller than machine accuracy!
>
>> K = 0L
>> ThisResult = REPLICATE(0D, N_ELEMENTS(Z))
>> WHILE (N_ELEMENTS>LastResult) EQ 0) || (MAX(ABS>LastResult -
>> ThisResult)) GT Cutoff) DO BEGIN
>> LastResult = ThisResult
>> AK = GAMMA(A + K) / GAMMA(A) ; Define (A)k
>> BK = GAMMA(B + K) / GAMMA(B) ; Define (B)k
>> F = (AK * Z^K) / (BK * FACTORIAL(K)) ; Evaluatefunction.
>> ThisResult = LastResult + F
>> K += 1
>> ENDWHILE ; Until result is good to Precision
>
>> ; Error if not enough while loops to give accurate results.
>> IF K LE 1 THEN MESSAGE, 'Functionfailed. Try greater precision.'
>
>> RETURN, ThisResult
>> END- Hide quoted text -
>
>> - Show quoted text -
>

```

```

> Noah,
>
> Here is my implementation, both of the series expansion of the hypergeometric function and the
integral representation. Depending on
> the parameters, I have found that one may be more stable than the
> other. Both of these are based on Arfken and Weber, Mathematical
> Methods for Physicists.
>
> best,
> dan
>
> ; chss
> ; confluent hypergeometric series solution
> ; calculates the solution to the differential equation:
> ;  $xy''(x) + (c - x)y'(x) - ay(x) = 0$ 
> ; (see Arfken and Weber, p. 801-2)
> ;
> ; inputs:
> ; n: number of terms to calculate. Due to limits in IDL architecture n
> must be between 1 and 170.
> ; a, c: vector of constants - see above. They MUST have the same
> number of elements
> ; x: input value
> ;
> ; outputs:
> ; y: output vector
> ;
> ; Dan Larson, 2007.10.11
>
> function chss, n, a, c, x
>   y = DBLARR((SIZE(a))[1])
>   ; first term of series expansion is 1, so we initialize the output
>   y[*] = 1
>
>   FOR i = 0, (SIZE(a))[1]-1 DO BEGIN
>     ; initialize constant series products
>     a_p = 1
>     c_p = 1
>     FOR j = 0, n DO BEGIN
>
>       a_p *= (a[i] + j)
>       c_p *= (c[i] + j)
>       d = FACTORIAL(j + 1)
>
>       y[i] += (a_p/c_p)*(x^(j + 1))/d
>
>     ENDFOR
>   ENDFOR

```

```

> RETURN, y
> END
>
> ; chins
> ; confluenthypergeometricintegral solution
> ; calculates the solution to the differential equation:
> ;  $xy''(x) + (c - x)y'(x) - ay(x) = 0$ 
> ; (see Afken and Weber, p. 801-2)
> ;
> ; inputs:
> ; a, c: vector constants - see above
> ; x: input value
> ;
> ; outputs:
> ; y: output vector
> ;
> ;Dan larson, 2007.10.11
>
> Functionchins, a, c, x
> ; curve point resolution, change this if your results look like
> crap
> b = 1000
> ; initialize t vector
> t = DINDGEN(b + 1)/b
> ; initialize vector of curve heights
> h = DBLARR(b + 1)
> ; initialize output
> y=dblarr(n_elements(c))
>
> g1 = GAMMA(c)
> g2 = GAMMA(a) * GAMMA(c - a)
>
> FOR i = 0, (SIZE(a))[1]-1 DO BEGIN
>   FOR j = 0, b DO BEGIN
>     h[j] = exp(x*t[j]) * ((t[j])^(a[i] - 1.0)) * ((1.0 -
> t[j])^(c[i] - a[i] - 1.0))
>   ENDFOR
>
>   y[i] = (g1[i]/g2[i]) * int_tabulated(t, h, /double)
> ENDFOR
>
> RETURN, y
> END
>
> ; chcmp
> ; compares the results between the integral and series form of CHF
> ;
> ; inputs:

```

```

> ; a, c: constants
> ; x: input value
> ;
> ; output:
> ; none
> ; (solution is printed to screen)
>
> PRO chcmp, a, c, x, epsilon
>   y1 = chins(a, c, x)
>
>   lastVal = 0.0000
>
>   FOR n = 1, 170 DO BEGIN
>     y2 = chss( n, a, c, x)
>
>     IF ( ABS(y1 - y2)/y1 LT epsilon) THEN BEGIN
>       print, "Number of necessary terms for ep = ", epsilon, " n =
> ", n
>       BREAK
>     ENDIF
>
>     IF( ABS(lastVal - y2)/y2 EQ 0) THEN BEGIN
>       print, "Series converged before matching integral!"
>       BREAK
>     ENDIF
>
>     lastVal = y2
>   ENDFOR
>
>   IF n EQ 169 THEN PRINT, "Equations did not meet!"
>
>   RETURN
> END

```

Thank you all for your answers. It really made my day!
 I've tested the 3 methods for the "Mathematica" example (i.e. for
 $1F1(1,2,[-5..5])$)
 HYPERGEOMETRICONEFONE, CHSS and CHINS seem to work fine for this
 example. HYPERGEOMETRICONEFONE seems a bit faster than the other ones.
 Unfortunately I am more interested in evaluating something like $1F1(-.$
 $75,1,40)$:

```
IDL> print,hypergeometriconefone(-0.75D,1D,-40D)
      NaN
```

```
IDL> print, chins([-0.75D],[1D],-40D)
```

NaN

```
IDL> print, chss(169D,[-0.75D],[1D],-40D)  
17.478776
```

The mathematica website gives [<http://functions.wolfram.com/webMathematica/FunctionEvaluation.jsp?name=Hypergeometric1F1>]:

Thanks,
Noah
