Package 'hypergeo'

July 22, 2025

Title The Gauss Hypergeometric Function

Version 1.2-14

Depends R (>= 3.1.0),

Imports elliptic (>= 1.3-5), contfrac (>= 1.1-9), deSolve

Description The Gaussian hypergeometric function for complex numbers.

Maintainer Robin K. S. Hankin <hankin.robin@gmail.com>

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2025-03-24 14:30:02 UTC

Author Robin K. S. Hankin [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5982-0415>)

Contents

`hypergeo-package` *The hypergeometric function*

Description

The hypergeometric function for the whole complex plane

Details

Package:	hypergeo
Type:	Package
Version:	1.0
Date:	2008-04-16
License:	GPL

The front end function of the package is `hypergeo()`: depending on the value of the parameters, this executes one or more of many sub-functions.

Author(s)

Robin K. S. Hankin

Maintainer: <hankin.robin@gmail.com>

References

M. Abramowitz and I. A. Stegun 1965. *Handbook of mathematical functions*. New York: Dover

Examples

```
hypergeo(1.1,2.3,1.9 , 1+6i)

options(showHGcalls = TRUE)    # any non-null value counts as TRUE
hypergeo(4.1, 3.1, 5.1, 1+1i)  # shows trace back
options(showHGcalls = FALSE)   # reset
```

buhring *Evaluation of the hypergeometric function using Buhring's method*

Description

Expansion of the hypergeometric function using the residue theorem; useful for when the primary argument is close to the critical points $1/2 \pm i\sqrt{3}/2$

Usage

```
hypergeo_buhring(A,B,C,z,z0=1/2,tol=0,maxiter=2000,use11=TRUE)
buhring_eqn11(n,S,A,B,C,z0=1/2)
buhring_eqn12(n,S,A,B,C,z0=1/2)
buhring_eqn5_factors(A,B,C,z,z0=1/2)
buhring_eqn5_series(S,A,B,C,z,z0=1/2,use11=FALSE,tol=0,maxiter=2000)
```

Arguments

A, B, C	Parameters (real)
S	Parameter taken to be either A or B
n	Term to calculate in buhring_eqn11() or buhring_eqn12()
z	Primary complex argument
z0	Centre of circle of non-convergence; series expressed in powers of $1/\left(z-z_0\right)^n$
tol, maxiter	tolerance and maximum number of iterations (as in hypergeo())
use11	Boolean with default TRUE meaning to use buhring_eqn11() and FALSE meaning to use buhring_eqn12()

Details

The functions are direct transcriptions of Buhring 1987. The basic idea is to expand the hypergeometric function in powers of $(z-z_0)^{-1}$.

Functions buhring_eqn11() and buhring_eqn12() return the coefficients d_n given by equations 11 and 12 of Buhring 1987.

The series do not converge satisfactorily near the critical points due to some sort of numerical instability. But they seem to work OK if $|z-1/2|$ is large.

Note

There is some issue which prevents the series from converging correctly, also sometimes the sequence converges to a demponstrably incorrect value.

Author(s)

Robin K. S. Hankin

References

- W. Buhring 1987. "An analytic continuation of the hypergeometric series", *Siam J. Math. Anal.* 18(3)

See Also

residue

Examples

```
# should be identical:
buhring_eqn11(n=0:10,S=1/2,A=1/2,B=1/3,C=pi)
buhring_eqn12(n=0:10,S=1/2,A=1/2,B=1/3,C=pi)
# but differ in one element


a <- hypergeo(1/2,1/3,4,1+8i,maxiter=90)
b <- hypergeo_buhring(1/2,1/3,4,1+8i,maxiter=90)
# should be identical but are not


# following command fails due to numerical instability:
## Not run:
hypergeo_buhring(1/2,1/3,pi,z=1/2 + 1i*sqrt(3)/2)

## End(Not run)
```

---

complex_gamma *Gamma function for complex arguments*

Description

Gamma and factorial functions for complex arguments

Usage

```
complex_gamma(z, log = FALSE)
complex_factorial(z, log = FALSE)
lanczos(z,log=FALSE)
```

Arguments

z	Primary argument, a complex vector
log	Boolean, with default FALSE meaning to return the function value and TRUE meaning to return its logarithm

Details

Method follows that of Lanczos, coefficients identical to those of the GSL

Author(s)

Robin K. S. Hankin

References

Lanczos, C. 1964. "A precision approximation of the gamma function". *Journal of the society for industrial and applied mathematics series B*, Volume 1, pp86-96

M. Galassi et al, GNU Scientific Library Reference Manual (3rd Ed.), ISBN 0954612078.

Examples

```
complex_gamma(5)  # should be 4!=24

complex_gamma(1+1i)   # takes complex arguments

complex_gamma(-5/2) + sqrt(pi)*8/15  # should be small

z <- pi + 1i*sqrt(2)
complex_gamma(z+1)-z*complex_gamma(z)   # should be small


complex_gamma(z)*complex_gamma(1-z) - pi/sin(pi*z)  # small
```

---

f15.3.1 *Hypergeometric function using Euler's integral representation*

Description

Hypergeometric function using Euler's integral representation, evaluated using numerical contour integrals.

Usage

```
f15.3.1(A, B, C, z, h = 0)
```

Arguments

A, B, C	Parameters
z	Primary complex argument
h	specification for the path to be taken; see details

Details

Argument h specifies the path to be taken (the path has to avoid the point $1/z$). If h is real and of length 1, the path taken comprises two straight lines: one from 0 to $0.5 + hi$ and one from $0.5 + hi$ to 1 (if $h = 0$ the integration is performed over a single segment).

Otherwise, the integration is performed over `length(h)+1` segments: 0 to `h[1]`, then `h[i]` to `h[i+1]` for $1 \leq i \leq n - 1$ and finally `h[n]` to 1.

See examples and notes sections below.

Note

The Mellin-Barnes form is not yet coded up.

Author(s)

Robin K. S. Hankin

References

M. Abramowitz and I. A. Stegun 1965. *Handbook of mathematical functions*. New York: Dover

See Also

hypergeo

Examples

```
# For |z| <1 the path can be direct:
f15.3.1(2,1,2,-1/2) -2/3

# cf identity 07.23.03.0046.01 of Hypergeometric2F1.pdf with b=1




f <- function(h){f15.3.1(1,2,3, z=2, h=h)}

# Winding number [around 1/z] matters:
f(0.5)
f(c(1-1i, 1+1i, -2i))

# Accuracy isn't too bad; compare numerical to analytical result :
f(0.5) - (-1+1i*pi/2)
```

f15.3.10 *Transformations of the hypergeometric function*

Description

Transformations of the hypergeometric function detailed in AMS-55, page 559-560.

Usage

```
f15.3.10       (A, B,    z, tol = 0, maxiter = 2000, method = "a")
f15.3.10_a     (A, B,    z, tol = 0, maxiter = 2000              )
f15.3.10_b     (A, B,    z, tol = 0, maxiter = 2000              )
f15.3.11       (A, B, m, z, tol = 0, maxiter = 2000, method = "a")
f15.3.11_bit1  (A, B, m, z, tol = 0                              )
f15.3.11_bit2_a(A, B, m, z, tol = 0, maxiter = 2000              )
f15.3.11_bit2_b(A, B, m, z, tol = 0, maxiter = 2000              )
f15.3.12       (A, B, m, z, tol = 0, maxiter = 2000, method = "a")
f15.3.12_bit1  (A, B, m, z, tol = 0                              )
f15.3.12_bit2_a(A, B, m, z, tol = 0, maxiter = 2000              )
f15.3.12_bit2_b(A, B, m, z, tol = 0, maxiter = 2000              )
f15.3.13       (A, C,    z, tol = 0, maxiter = 2000, method = "a")
f15.3.13_a     (A, C,    z, tol = 0, maxiter = 2000              )
f15.3.13_b     (A, C,    z, tol = 0, maxiter = 2000              )
f15.3.14       (A, C, m, z, tol = 0, maxiter = 2000, method = "a")
f15.3.14_bit1_a(A, C, m, z, tol = 0, maxiter = 2000              )
f15.3.14_bit1_b(A, C, m, z, tol = 0, maxiter = 2000              )
f15.3.14_bit2  (A, C, m, z, tol = 0                              )
f15.3.13_14    (A, C, m, z, tol = 0, maxiter = 2000, method = "a")
f15.3.10_11_12 (A, B, m, z, tol = 0, maxiter = 2000, method = "a")
f15.1.1        (A, B, C, z, tol = 0, maxiter = 2000              )
```

Arguments

A, B, C	Parameters of the hypergeometric function
m	Integer linking A, B, C as set out in AMS-55, page 559,560
z	primary complex argument
tol, maxiter	numerical parameters
method	Length 1 character vector specifying the method. See details

Details

Naming scheme (functions and arguments) follows AMS-55, pages 559-560.

The method argument to (eg) f15.3.14() specifies whether to use psigamma() directly (method "a"), or the recurrence 6.3.5 (method "b"). Press et al recommend method "b", presumably on the grounds of execution speed. I'm not so sure (method "a" seems to be more accurate in the sense that it returns values closer to those of Maple).

Method "c" means to use a totally dull, slow, direct (but clearly correct) summation, for the purposes of debugging. This is only used for the functions documented under `wolfram.Rd`

Functions `f15.3.13_14()` and `f15.3.10_11_12()` are convenience wrappers. For example, function `f15.3.13_14()` dispatches to either `f15.3.13()` or `f15.3.14()` depending on the value of `m`.

Note

These functions are not really designed to be called by the user: use `hypergeo()` instead, or `hypergeo_cover[123]()` for specific cases.

Author(s)

Robin K. S. Hankin

References

M. Abramowitz and I. A. Stegun 1965. *Handbook of mathematical functions*. New York: Dover

See Also

`hypergeo`,`wolfram`,`hypergeo_cover1`

Examples

```
f15.3.10_11_12(A=1.1, B=pi, m= +3, z=.1+.1i)
f15.3.10_11_12(A=1.1, B=pi, m= -3, z=.1+.1i)
```

---

f15.3.3 *Various transformation formulae for the hypergeometric function*

Description

Transformations of the hypergeometric function: equations 15.3.3 to 15.3.9

Usage

```
f15.3.3(A, B, C, z, tol = 0, maxiter = 2000)
f15.3.4(A, B, C, z, tol = 0, maxiter = 2000)
f15.3.5(A, B, C, z, tol = 0, maxiter = 2000)
f15.3.6(A, B, C, z, tol = 0, maxiter = 2000)
f15.3.7(A, B, C, z, tol = 0, maxiter = 2000)
f15.3.8(A, B, C, z, tol = 0, maxiter = 2000)
f15.3.9(A, B, C, z, tol = 0, maxiter = 2000)
```

Arguments

A, B, C	Parameters of the hypergeometric function
z	Primary complex argument
tol, maxiter	parameters passed to genhypergeo()

Details

The naming scheme follows that of Abramowitz and Stegun

Author(s)

Robin K. S. Hankin

References

M. Abramowitz and I. A. Stegun 1965. "Handbook of mathematical functions". New York: Dover

See Also

hypergeo

Examples

```
f15.3.4(1.1,2.2,3.4,-1+0.1i)
```

---

f15.5.1 *Hypergeometric functions via direct numerical integration*

Description

The hypergeometric function may be evaluated using Gauss's differential equation 15.5.1:

$$z(1-z)w'' + (c-(a+b+1)z)w' - abw = 0$$

using a start value away from the three singular points. This page documents a suite of related functionality.

Usage

```
hypergeo_press(A,B,C,z, ...)
f15.5.1(A, B, C, z, startz, u, udash, give=FALSE, ...)
hypergeo_func(Time, State, Pars, u, udash)
to_real(o)
to_complex(p)
complex_ode(y, times, func, parms=NA, method=NULL, u, udash, ...)
semicircle(t,z0,z1,clockwise=TRUE)
semidash(t,z0,z1,clockwise=TRUE)
straight(t,z0,z1)
straightdash(t,z0,z1)
```

Arguments

A, B, C, z	Standard parameters for the hypergeometric function
u, udash	Functions to specify the path of integration, and its derivative
give	In function f15.5.1(), Boolean with TRUE meaning to return extra information from ode() and default FALSE meaning to return only the evaluated function
startz	In function f15.5.1(, the start position of the path
...	Further arguments passed to ode()
o, p	Real and complex objects to be coerced to each other in to_real() and to_complex()
y, times, func, parms, method	In function complex_ode(), arguments matching those of ode()
t, z0, z1, clockwise	Arguments for the standard path functions semicircle() et seq: u is the primary argument (real, $0 \leq u \leq 1$); z0 and z1 are the start and end points of the path; and clockwise is Boolean, indicating whether the path proceeds clockwise or not
Time, State, Pars	arguments matchin those of standard examples in the deSolve package

Details

Function hypergeo_press() is the most user-friendly of the functions documented here. It performs integration of Gauss's ODE, along a straight line path from the start-point to z. It follows Press et al's suggestion of start-point.

Function f15.5.1() is a little more flexible in that it allows the user to choose a start point and an integration path.

Function complex_ode() is a complex generalization of ode() of package **deSolve**; function hypergeo_func is an internal function, designed for use with complex_ode(), that specifies the Gauss ODE which is satisified by the hypergeometric function.

Functions to_real() and to_complex() are internal functions which coerce from real to complex and back; they are needed because ode() deals only with real values.

Functions semicircle() and straight() are helper functions which specify straight or semicircular paths from z0 to z1; note that f(0)=z0 and f(1)=z1. Functions semidash() and straightdash() provide the differential of the path.

Note

Accuracy is low compared with the other methods in the package.

Author(s)

Robin K. S. Hankin

References

W. H. Press et al. 1997. *Numerical Recipes in C*. Cambridge University Press, Second Edition.

See Also

hypergeo_residue

Examples

```
hypergeo_press(A=pi,B=sqrt(2),C=1.4,z=1-2i)
hypergeo      (A=pi,B=sqrt(2),C=1.4,z=1-2i)


jj1 <-
f15.5.1(
    A=1.1, B=2.2, C=3.3, z=3+0.5i, startz=0.5,
        u    =function(u){semicircle(u,0.5,3+0.5i,FALSE)},
        udash=function(u){semidash(u,0.5,3+0.5i,FALSE)}
        )

jj2 <-
f15.5.1(
    A=1.1, B=2.2, C=3.3, z=3+0.5i, startz=0.5,
        u    =function(u){semicircle(u,0.5,3+0.5i,TRUE)},
        udash=function(u){semidash(u,0.5,3+0.5i,TRUE)}
        )



jj3 <- hypergeo(    A=1.1, B=2.2, C=3.3, z=3+0.5i)
## First one agrees with jj3=hypergeo(...), the second one does not


# Now try the Airy Ai function;  satisfies f'' =  zf:

pars <- c(kay = 1+1i, ell = 0.1+0.2i)  # not actually used
airy_ai_func <- function(Time, State, Pars, u, udash) {
    with(as.list(c(to_complex(State), to_complex(Pars))), {

      z <- u(Time)
      dz <- udash(Time)

      dF <- Fdash*dz
      dFdash <-  z*F*dz # could use kay and ell from pars here if necessary

        ## coerce back to real:
        out <- to_real(c(dF,dFdash))
        names(out) <- names(State)
        return(list(out))
    })
}

complex_ode(
    y     = c(F = 1/3^(2/3)/gamma(2/3), Fdash= -1/3^(1/3)/gamma(1/3)),
    times = seq(0,1,by=0.1),
    func  = airy_ai_func,
```

```
    parms = pars,
    u     = function(t){straight(t,0,1)},
    udash = function(t){straightdash(t,0,1)}
)

# Look at the last line for the value at 1.
# compare gsl: Ai(1) = 0.1352924 ; Ai'(1) = -0.1591474

# ...although in this case there is actually a hypergeometric series
#  representation:

f <- function(z){
    return(
        +genhypergeo(U=NULL,L=2/3,z^3/9)/3^(2/3)/gamma(2/3)
        -genhypergeo(U=NULL,L=4/3,z^3/9)/3^(1/3)/gamma(1/3)*z
        )
}

f(1)
```

---

genhypergeo *The generalized hypergeometric function*

Description

The generalized hypergeometric function, using either the series expansion or the continued fraction expansion.

Usage

```
genhypergeo(U, L, z, tol=0, maxiter=2000, check_mod=TRUE,
    polynomial=FALSE, debug=FALSE, series=TRUE)
genhypergeo_series(U, L, z, tol=0, maxiter=2000, check_mod=TRUE,
    polynomial=FALSE, debug=FALSE)
genhypergeo_contfrac(U, L, z, tol = 0, maxiter = 2000)
```

Arguments

U, L	Upper and lower arguments respectively (real or complex)
z	Primary complex argument (see notes)
tol	tolerance with default zero meaning to iterate until additional terms to not change the partial sum
maxiter	Maximum number of iterations to perform
check_mod	Boolean, with default TRUE meaning to check that the modulus of z is less than 1

polynomial	Boolean, with default FALSE meaning to evaluate the series until converged, or return a warning; and TRUE meaning to return the sum of maxiter terms, whether or not converged. This is useful when either A or B is a nonpositive integer in which case the hypergeometric function is a polynomial
debug	Boolean, with TRUE meaning to return debugging information and default FALSE meaning to return just the evaluate
series	In function genhypergeo(), Boolean argument with default TRUE meaning to return the result of genhypergeo_series() and FALSE the result of genhypergeo_contfrac()

Details

Function genhypergeo() is a wrapper for functions genhypergeo_series() and genhypergeo_contfrac().

Function genhypergeo_series() is the workhorse for the whole package; every call to hypergeo() uses this function except for the (apparently rare—but see the examples section) cases where continued fractions are used.

The generalized hypergeometric function [here genhypergeo()] appears from time to time in the literature (eg Mathematica) as

$$F(U,L;z) = \sum_{n=0}^{\infty} \frac{(u_1)_n (u_2)_n \dots (u_i)_n}{(l_1)_n (l_2)_n \dots (l_j)_n} \cdot \frac{z^n}{n!}$$

where $U = (u_1, \dots, u_i)$ and $L = (l_1, \dots, l_i)$ are the "upper" and "lower" vectors respectively. The radius of convergence of this formula is 1.

For the Confluent Hypergeometric function, use genhypergeo() with length-1 vectors for arguments U and V.

For the ${}_0F_1$ function (ie no "upper" arguments), use genhypergeo(NULL,L,x).

See documentation for genhypergeo_contfrac() for details of the continued fraction representation.

Note

The radius of convergence for the series is 1 but under some circumstances, analytic continuation defines a function over the whole complex plane (possibly cut along $(1, \infty)$). Further work would be required to implement this.

Author(s)

Robin K. S. Hankin

References

M. Abramowitz and I. A. Stegun 1965. *Handbook of mathematical functions*. New York: Dover

See Also

hypergeo,genhypergeo_contfrac

Examples

```
genhypergeo(U=c(1.1,0.2,0.3), L=c(10.1,pi*4), check_mod=FALSE, z=1.12+0.2i)
genhypergeo(U=c(1.1,0.2,0.3), L=c(10.1,pi*4),z=4.12+0.2i,series=FALSE)
```

`gosper` *Evaluation of the hypergeometric function using Gosper's method*

Description

Evaluation of the hypergeometric function using Gosper's method

Usage

```
hypergeo_gosper(A, B, C, z, tol = 0, maxiter = 2000)
```

Arguments

`A`, `B`, `C`	Parameters (real or complex)
`z`	Complex argument
`tol`	tolerance (passed to `GCF()`)
`maxiter`	maximum number of iterations

Details

Gosper provides a three-term recurrence which converges when z is close to a critical point.

Bill Gosper asserts that the recursion holds for values of z which are inside the cardioid `(sqrt(8)*cos(t)-2*cos(2t),sqrt(` (see examples section).

It is suggested that the recursion should only be used when the auxiliary parameters A, B,C are all ≤ 12 in absolute value.

Author(s)

R code by Robin K. S. Hankin, transcribed from maxima code posted by Richard Fateman, who credited Bill Gosper

References

Original email was archived at `https://www.ma.utexas.edu/pipermail/maxima/2006/000126.html` but does not appear there now; and the wayback machine doesn't find it either.

See Also

`hypergeo_contfrac`

Examples

```
hypergeo_gosper(1.1,5.1,3.1,crit())

# Compare MMA: -0.192225 + 0.692328 I

t <- seq(from=0,to=2i*pi,len=100)
plot(exp(t)*(sqrt(8)-exp(t)),asp=1,type='l')
points(crit())
```

`hypergeo` *The hypergeometric function*

Description

The Hypergeometric and generalized hypergeometric functions as defined by Abramowitz and Stegun. Function `hypergeo()` is the user interface to the majority of the package functionality; it dispatches to one of a number of subsidiary functions.

Usage

```
hypergeo(A, B, C, z, tol = 0, maxiter=2000)
```

Arguments

`A`, `B`, `C`	Parameters for `hypergeo()`
`z`	Primary argument, complex
`tol`	absolute tolerance; default value of zero means to continue iterating until the result does not change to machine precision; strictly positive values give less accuracy but faster evaluation
`maxiter`	Integer specifying maximum number of iterations

Details

The hypergeometric function as defined by Abramowitz and Stegun, equation 15.1.1, page 556 is

$$ {}_2F_1(a,b;c;z) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n}{(c)_n} \cdot \frac{z^n}{n!} $$

where $(a)_n = a(a+1)\ldots(a+n-1) = \Gamma(a+n)/\Gamma(a)$ is the Pochammer symbol (6.1.22, page 256).

Function `hypergeo()` is the front-end for a rather unwieldy set of back-end functions which are called when the parameters `A`, `B`, `C` take certain values.

The general case (that is, when the parameters do not fall into a "special" category), is handled by `hypergeo_general()`. This applies whichever of the transformations given on page 559 gives the smallest modulus for the argument `z`.

Sometimes `hypergeo_general()` and all the transformations on page 559 fail to converge, in which case `hypergeo()` uses the continued fraction expansion `hypergeo_contfrac()`.

If this fails, the function uses integration via `f15.3.1()`.

Note

Abramowitz and Stegun state:

"The radius of convergence of the Gauss hypergeometric series ... is $|z| = 1$" (AMS-55, section 15.1, page 556).

This reference book gives the correct radius of convergence; use the ratio test to verify it. Thus if $|z| > 1$, the hypergeometric series will diverge and function `genhypergeo()` will fail to converge.

However, the hypergeometric function is defined over the whole of the complex plane, so analytic continuation may be used if appropriate cut lines are used. A cut line must join $z = 1$ to (complex) infinity; it is conventional for it to follow the real axis in a positive direction from $z = 1$ but other choices are possible.

Note that in using the package one sometimes draws a "full precision not achieved" warning from `gamma()`; and complex arguments are not allowed. I would suggest either ignoring the warning (the error of `gamma()` is unlikely to be large) or to use one of the bespoke functions such as `f15.3.4()` and tolerate the slower convergence, although this is not always possible.

Author(s)

Robin K. S. Hankin

References

Abramowitz and Stegun 1955. *Handbook of mathematical functions with formulas, graphs and mathematical tables* (AMS-55). National Bureau of Standards

See Also

`hypergeo_powerseries`, `hypergeo_contfrac`, `genhypergeo`

Examples

```
#  equation 15.1.3, page 556:
f1 <- function(x){-log(1-x)/x}
f2 <- function(x){hypergeo(1,1,2,x)}
f3 <- function(x){hypergeo(1,1,2,x,tol=1e-10)}
x <- seq(from = -0.6,to=0.6,len=14)
f1(x)-f2(x)
f1(x)-f3(x)  # Note tighter tolerance

# equation 15.1.7, p556:
g1 <- function(x){log(x + sqrt(1+x^2))/x}
g2 <- function(x){hypergeo(1/2,1/2,3/2,-x^2)}
```

```
g1(x)-g2(x)  # should be small
abs(g1(x+0.1i) - g2(x+0.1i))  # should have small modulus.

# Just a random call, verified by Maple [ Hypergeom([],[1.22],0.9087) ]:
genhypergeo(NULL,1.22,0.9087)


# Little test of vectorization (warning: inefficient):
hypergeo(A=1.2+matrix(1:10,2,5)/10, B=1.4, C=1.665, z=1+2i)


# following calls test for former bugs:
hypergeo(1,2.1,4.1,1+0.1i)
hypergeo(1.1,5,2.1,1+0.1i)
hypergeo(1.9, 2.9, 1.9+2.9+4,1+0.99i) # c=a+b+4; hypergeo_cover1()
```

`hypergeo_A_nonpos_int` *Hypergeometric functions for integer arguments*

Description

Hypergeometric functions for A and/or B being integers

Usage

```
hypergeo_A_nonpos_int(A, B, C, z, tol = 0)
hypergeo_AorB_nonpos_int(A, B, C, z, tol = 0)
```

Arguments

`A`, `B`, `C`	Parameters for the hypergeometric function
`z`	Primary complex argument
`tol`	tolerance

Details

The "point" of these functions is that if A and C (or B and C) are identical nonpositive integers, a warning needs to be given because the function is defined as the appropriate limit and one needs to be sure that both A and C approach that limit at the same speed.

Function `hypergeo_AorB_nonpos_int()` is a convenience wrapper for `hypergeo_A_nonpos_int()`.

Author(s)

Robin K. S. Hankin

References

M. Abramowitz and I. A. Stegun 1965. *Handbook of mathematical functions*. New York: Dover

See Also

hypergeo

Examples

```
jjR1 <- hypergeo(-4, pi, 2.2 , 1+6i)
jjR2 <- hypergeo(pi, -4, 2.2 , 1+6i)  # former bug
jjM <- 3464.1890402837334002-353.94143580568566281i  # value given by Mathematica
```

---

hypergeo_contfrac *Continued fraction expansion of the hypergeometric function*

Description

Continued fraction expansion of the hypergeometric and generalized hypergeometric functions using continued fraction expansion.

Usage

```
hypergeo_contfrac(A, B, C, z, tol = 0, maxiter = 2000)
genhypergeo_contfrac_single(U, L, z, tol = 0, maxiter = 2000)
```

Arguments

A, B, C	Parameters (real or complex)
U, L	In function genhypergeo_contfrac(), upper and lower arguments as in genhypergeo()
z	Complex argument
tol	tolerance (passed to GCF())
maxiter	maximum number of iterations

Details

These functions are included in the package in the interests of completeness, but it is not clear when it is advantageous to use continued fraction form rather than the series form.

Note

The continued fraction expression is the RHS identity 07.23.10.0001.01 of Hypergeometric2F1.pdf.

The function sometimes fails to converge to the correct value but no warning is given.

Function genhypergeo_contfrac() is documented under genhypergeo.Rd.

Author(s)

Robin K. S. Hankin

References

- M. Abramowitz and I. A. Stegun 1965. *Handbook of mathematical functions*. New York: Dover
- http://functions.wolfram.com/Hypergeometric2F1.pdf

See Also

genhypergeo

Examples

```
hypergeo_contfrac(0.3 , 0.6 , 3.3 , 0.1+0.3i)
# Compare Maple: 1.004280829477551 1972+0.17044041575976110947e-1i

genhypergeo_contfrac_single(U=0.2 , L=c(9.9,2.7,8.7) , z=1+10i)
# (powerseries does not converge)
# Compare Maple: 1.0007289707983569879 + 0.8625071421725183731 7e-2i
```

---

hypergeo_cover1 *Hypergeometric functions for special values of the parameters*

Description

Hypergeometric functions for special values of the parameters

Usage

```
hypergeo_cover1(A, B, m, z, tol = 0, maxiter = 2000, method = "a", give = FALSE)
hypergeo_cover2(A, C, m, z, tol = 0, maxiter = 2000, method = "a", give = FALSE)
hypergeo_cover3(A, n, m, z, tol = 0, maxiter = 2000, method = "a", give = FALSE)
```

Arguments

A, B, C	parameters for the hypergeometric function
m, n	Integers (positive or negative)
z	Primary complex argument
tol, maxiter	Numerical arguments passed to genhypergeo()
method	Method, passed to f15.3.10() (qv)
give	Boolean with TRUE meaning to return the choice of helper function used (eg f15.3.7()), and default FALSE meaning to return the hypergeometric function's value

Details

These functions deal with the exceptional cases listed on page 559-560.

- Function `hypergeo_cover1()` deals with the case $C = A + B \pm m, m = 0, 1, 2, \ldots$
- Function `hypergeo_cover2()` deals with the case $B = A \pm m, m = 0, 1, 2, \ldots$
- Function `hypergeo_cover3()` deals with the case $C - A = 0, -1, -2, \ldots$ [elementary] and $C - A = 1, 2, \ldots$ [not covered by AMS-55]

Note

Function `hypergeo_cover3()` is required because the "limiting process" mentioned on p560, just after 15.3.14, is not explicit. Which is why it dispatches to `w07.23.06.0026.01()` and `w07.23.06.0031.01()`, documented at `wolfram`.

Author(s)

Robin K. S. Hankin

References

M. Abramowitz and I. A. Stegun 1965. *Handbook of mathematical functions*. New York: Dover

See Also

`hypergeo`,`f15.3.10`,`wolfram`

Examples

```
# Test hypergeo_cover1():
jjR <- hypergeo(pi,pi/2,3*pi/2-4, z=0.1+0.2i)
jjM <- 0.53745229690249593045 + 1.89174564732405156 64i


# Test hypergeo_cover2():
jjM <- -0.15888831928748121465e-5 + 0.40339599711492215912e-4i
jjR <- hypergeo(pi,pi+2, 1.1 , 1+10i)  # This is 15.3.13
stopifnot(Mod(jjR-jjM)<1e-10)


# Test hypergeo_cover3()
jjM <- -0.24397135980533720308e-1 + 0.28819643319432922231i
jjR <- hypergeo(pi, 1.4, pi+4, 1+6i)
stopifnot(Mod(jjR-jjM)<1e-10)
```

`hypergeo_powerseries` *The hypergeometric function as determined by power series*

Description

The hypergeometric function as determined by infinite (`hypergeo_powerseries()`) or finite (`hypergeo_taylor()`) power series

Usage

```
hypergeo_powerseries(A, B, C, z, tol = 0, maxiter = 2000)
```

Arguments

`A`, `B`, `C`	Parameters of the hypergeometric function
`z`	Primary complex argument
`tol`, `maxiter`	Numerical arguments

Details

Function `hypergeo_powerseries()` is the primary decision-making function of the package. It is this function that detects degenerate cases of the three parameters and dispatches accordingly. Non-degenerate cases are sent to function `hypergeo_general()`.

Function `hypergeo_taylor()` deals with cases where the hypergeometric function is a polynomial.

Author(s)

Robin K. S. Hankin

References

M. Abramowitz and I. A. Stegun 1965. *Handbook of mathematical functions*. New York: Dover

See Also

`hypergeo`,`genhypergeo`

Examples

```
jjR <- hypergeo(pi,-4,2.2,1+5i)
jjM <- 1670.8287595795885335 - 204.81995157365381258i
```

`i15.3.6` *Helper functions*

Description

Helper functions for equations 15.3.6-15.3.9

Usage

```
i15.3.6(A, B, C)
i15.3.7(A, B, C)
i15.3.8(A, B, C)
i15.3.9(A, B, C)
j15.3.6(A, B, C)
j15.3.7(A, B, C)
j15.3.8(A, B, C)
j15.3.9(A, B, C)
```

Arguments

`A`, `B`, `C`	Parameters of the hypergeometric function

Details

These functions are named for their equation numbers in Abramowitz and Stegun.

Functions `i15.3.?()` return the factors at the beginning of equations 15.3.6-9. These functions return zero if the denominator is infinite (because it includes a gamma function of a nonpositive integer).

Functions `j15.3.?()` check for the appropriate arguments of the gamma function being nonpositive integers.

Author(s)

Robin K. S. Hankin

References

M. Abramowitz and I. A. Stegun 1965. *Handbook of mathematical functions*. New York: Dover

See Also

`hypergeo`

Examples

```
i15.3.6(1.1, 3.2, pi)
```

`is.nonpos` *Various utilities*

Description

Various utilities needing nonce functions

Usage

```
is.near_integer(i, tol=getOption("tolerance"))
is.nonpos(i)
is.zero(i)
isgood(x, tol)
thingfun(z, complex=FALSE)
crit(...)
lpham(x,n)
```

Arguments

`i`	Numerical vector of suspected integers
`tol`	Tolerance
`x`	Argument to `isgood()` and `lpham()`
`z`	Complex vector
`complex`	In function `thingfun()`, Boolean with default `FALSE` meaning to return the modulus of the transforms and `TRUE` meaning to return the complex values themselves
`n`	second argument to `lpham()`
`...`	Ignored

Details

- Function `is.near_integer(i)` returns `TRUE` if `i` is "near" [that is, within `tol`] an integer; if the option is unset then `1e-11` is used.
- Function `is.nonpos()` returns `TRUE` if `i` is near a nonpositive integer
- Function `is.zero()` returns `TRUE` if `i` is, er, near zero
- Function `isgood()` checks for all elements of `x` having absolute values less than `tol`
- Function `thingfun()` transforms input vector `z` by each of the six members of the anharmonic group, viewed as a subgroup of the Mobius group of functions. It returns a real six-column matrix with columns being the modulus of $z, z/(z-1), 1-z, 1/z, 1/(1-z), 1-1/z$. These six columns correspond to the primary argument in equations 15.3.3 to 15.3.9, p551 of AMS-55
- Function `crit()` returns the two critical points, $\frac{1}{2} \pm \frac{\sqrt{3}i}{2}$. These points have unit modulus as do their six transforms by `thingfun()`
- Function `lpham()` returns the log of the Pochhammer function $log\left(\Gamma(x+n)/\Gamma(x)\right)$

Note

Function `isgood()` uses zero as the default tolerance (argument `tol` passed in from `hypergeo()`); compare the different meaning of `tol` used in `is.near_integer()`.

Here, "integer" means one of the sequence $0, \pm 1, \pm 2, \ldots$ [ie *not* the Gaussian integers].

Author(s)

Robin K. S. Hankin

Examples

```
is.near_integer(-3)

is.zero(4)
```

`residue` *Evaluation of the hypergeometric function using the residue theorem*

Description

Expansion of the hypergeometric function using the residue theorem; useful for when the primary argument is close to the critical points $1/2 \pm i\sqrt{3}/2$

Usage

```
hypergeo_residue_general(A, B, C, z, r, O=z, tol=0, maxiter=2000)
hypergeo_residue_close_to_crit_single(A, B, C, z, strategy='A', tol=0, maxiter=2000)
hypergeo_residue_close_to_crit_multiple(A, B, C, z, strategy='A', tol=0, maxiter=2000)
```

Arguments

`A`, `B`, `C`	Parameters (real or complex)
`z`	Complex argument
`tol`, `maxiter`	tolerance and maximum number of iterations (passed to `hypergeo()`)
`r`, `O`	Radius and center of circle to integrate over
`strategy`	Indicates which strategy to use. Strategy 'A' means to use the critical point as the centre of the circle and strategy 'B' means to use z

Details

These functions are not really intended for the user; `hypergeo()` uses `hypergeo_residue_close_to_crit_multiple()` when $|z - c|$ is less than 0.1 (hardwired) for c being either of the two critical points. Infinite regress is avoided because the contour is always more than this distance from the critical points.

These functions use the residue theorem $f(z_0) = \oint_C \frac{f(z)\,dz}{z - z_0}$ to evaluate the hypergeometric function near the two critical points $1/2 \pm i\sqrt{3}/2$. These points are problematic because all of the transformations listed under `thingfun()` take the points either to themselves or each other.

At these points the ratio of successive terms in the hypergeometric series tends to one and thus numerical summation is difficult.

The hypergeometric function, however, is not at all badly behaved near these critical points (see examples); but OTOH there do not seem to be any identities for the hypergeometric function at these points.

I have not investigated in detail whether strategy 'A' or 'B' is better. I would expect that 'A' is faster but 'B' more accurate, on the grounds that 'A' uses a contour whose closest approach to the critical point is further than that of 'B'; but 'B' uses a contour which does not vary in distance from z.

But both seem to be fairly accurate and fairly fast, and I have not systematically investigated the pros and cons.

Note

The residue theorem appears to be absurdly accurate for numerical evaluation

Author(s)

Robin K. S. Hankin

References

- W. Buhring 1987. "An analytic continuation of the hypergeometric series", *Siam J. Math. Anal.* 18(3)

See Also

buhring

Examples

```
c1 <- 1/2-sqrt(3)/2i
c2 <- 1/2+sqrt(3)/2i

a1_R <- hypergeo(1/2,1/3,pi,c1)
a1_M <- 1.0154051314906669 + 0.0544835896509068i

x <- y <- seq(from=-0.1,to=0.1,len=100)
elliptic::view(x,y,hypergeo(1/2,1,1/3,outer(x,1i*y,"+")))
```

---

shanks *Evaluation of the hypergeometric function using Shanks's method*

Description

Evaluation of the hypergeometric function using Shanks transformation of successive sums

Usage

```
hypergeo_shanks(A,B,C,z,maxiter=20)
genhypergeo_shanks(U,L,z,maxiter=20)
shanks(Last,This,Next)
```

Arguments

A, B, C	Parameters (real or complex)
U, L	Upper and lower vectors
z	Primary complex argument
maxiter	Maximum number of iterations
Last, This, Next	Three successive convergents

Details

The Shanks transformation of successive partial sums is

$$S(n) = \frac{A_{n+1}A_{n-1} - A_n^2}{A_{n+1} - 2A_n + A_{n-1}}$$

and if the A_n tend to a limit then the sequence $S(n)$ often converges more rapidly than A_n. However, the denominator is susceptible to catastrophic rounding under fixed-precision arithmetic and it is difficult to know when to stop iterating.

Note

The

Author(s)

Robin K. S. Hankin

References

- Shanks, D. (1955). "Non-linear transformation of divergent and slowly convergent sequences", *Journal of Mathematics and Physics* 34:1-42

See Also

buhring

Examples

```
hypergeo_shanks(1/2,1/3,pi,z= 0.1+0.2i)
```

wolfram *Various functions taken from the Wolfram Functions Site*

Description

Various functions taken from the Wolfram Functions Site

Usage

```
w07.23.06.0026.01(A, n, m, z, tol = 0, maxiter = 2000, method = "a")
w07.23.06.0026.01_bit1(A, n, m, z, tol = 0)
w07.23.06.0026.01_bit2(A, n, m, z, tol = 0, maxiter = 2000)
w07.23.06.0026.01_bit3_a(A, n, m, z, tol = 0)
w07.23.06.0026.01_bit3_b(A, n, m, z, tol = 0)
w07.23.06.0026.01_bit3_c(A, n, m, z, tol = 0)
w07.23.06.0029.01(A, n, m, z, tol = 0, maxiter = 2000)
w07.23.06.0031.01(A, n, m, z, tol = 0, maxiter = 2000)
w07.23.06.0031.01_bit1(A, n, m, z, tol = 0, maxiter = 2000)
w07.23.06.0031.01_bit2(A, n, m, z, tol = 0, maxiter = 2000)
```

Arguments

A	Parameter of hypergeometric function
m, n	Integers
z	Primary complex argument
tol, maxiter	Numerical arguments as per genhypergeo()
method	Character, specifying method to be used

Details

The method argument is described at f15.3.10. All functions' names follow the conventions in Hypergeometric2F1.pdf.

- Function w07.23.06.0026.01(A, n, m, z) returns ${}_2F_1(A, A+n, A+m, z)$ where m and n are nonnegative integers with $m \geq n$.
- Function w07.23.06.0029.01(A, n, m, z) returns ${}_2F_1(A, A+n, A-m, z)$.
- Function w07.23.06.0031.01(A, n, m, z) returns ${}_2F_1(A, A+n, A+m, z)$ with $m \leq n$.

Note

These functions use the psigamma() function which does not yet take complex arguments; this means that complex values for A are not supported. I'm working on it.

Author(s)

Robin K. S. Hankin

References

http://functions.wolfram.com/Hypergeometric2F1.pdf

See Also

f15.3.10,hypergeo

Examples

```
# Here we catch some answers from Maple (jjM) and compare it with R's:


jjM <- 0.95437201847068289095 + 0.808686874619544794391 # Maple's answer
jjR <- w07.23.06.0026.01(A=1.1 , n=1 , m=4 , z=1+1i)
# [In practice, one would type 'hypergeo(1.1, 2.1, 5.1, 1+1i)']

stopifnot(Mod(jjM - jjR) < 1e-10)


jjM <- -0.25955090546083991160e-3 - 0.59642767921444716242e-3i
jjR <- w07.23.06.0029.01(A=4.1 , n=1 , m=1 , z=1+4i)
# [In practice, one would type 'hypergeo(4.1, 3.1, 5.1, 1+1i)']

stopifnot(Mod(jjM - jjR) < 1e-15)

jjM <- 0.33186808222278923715e-1 - 0.40188208572232037363e-1i
jjR <- w07.23.06.0031.01(6.7,2,1,2+1i)
# [In practice, one would type 'hypergeo(6.7, 8.7, 7.7, 2+1i)']
stopifnot(Mod(jjM - jjR) < 1e-10)
```

Index