NAG Library Function Document

nag_specfun_1f1_real_scaled (s22bbc)

1 Purpose

nag_specfun_1f1_real_scaled (s22bbc) returns a value for the confluent hypergeometric function ${}_1F_1(a;b;x)$, with real parameters a and b and real argument x. The solution is returned in the scaled form ${}_1F_1(a;b;x) = m_f \times 2^{m_s}$. This function is sometimes also known as Kummer's function $M(a,b,x)$.

2 Specification

```
#include <nag.h>
#include <nags.h>
void nag_specfun_1f1_real_scaled (double ani, double adr, double bni,
     double bdr, double x, double *frm, Integer *scm, NagError *fail)
```

3 Description

nag_specfun_1f1_real_scaled (s22bbc) returns a value for the confluent hypergeometric function ${}_1F_1(a;b;x)$, with real parameters a and b and real argument x, in the scaled form ${}_1F_1(a;b;x) = m_f \times 2^{m_s}$, where m_f is the real scaled component and m_s is the integer power of two scaling. This function is unbounded or not uniquely defined for b equal to zero or a negative integer.

The confluent hypergeometric function is defined by the confluent series,

$$ {}_1F_1(a;b;x) = M(a,b,x) = \sum_{s=0}^{\infty} \frac{(a)_s x^s}{(b)_s s!} = 1 + \frac{a}{b}x + \frac{a(a+1)}{b(b+1)2!}x^2 + \cdots $$

where $(a)_s = 1(a)(a+1)(a+2)\ldots(a+s-1)$ is the rising factorial of a. $M(a,b,x)$ is a solution to the second order ODE (Kummer's Equation):

$$ x\frac{d^2M}{dx^2} + (b-x)\frac{dM}{dx} - aM = 0. \tag{1} $$

Given the parameters and argument (a,b,x), this function determines a set of safe values $\{(\alpha_i,\beta_i,\zeta_i) \mid i \leq 2\}$ and selects an appropriate algorithm to accurately evaluate the functions $M_i(\alpha_i,\beta_i,\zeta_i)$. The result is then used to construct the solution to the original problem $M(a,b,x)$ using, where necessary, recurrence relations and/or continuation.

For improved precision in the final result, this function accepts a and b split into an integral and a decimal fractional component. Specifically $a = a_i + a_r$, where $|a_r| \leq 0.5$ and $a_i = a - a_r$ is integral. b is similarly deconstructed.

Additionally, an artificial bound, $arbnd$ is placed on the magnitudes of a_i, b_i and x to minimize the occurrence of overflow in internal calculations. $arbnd = 0.0001 \times I_{\max}$, where $I_{\max} =$ X02BBC. It should, however, not be assumed that this function will produce an accurate result for all values of a_i, b_i and x satisfying this criterion.

Please consult the NIST Digital Library of Mathematical Functions or the companion (2010) for a detailed discussion of the confluent hypergeometric function including special cases, transformations, relations and asymptotic approximations.

4 References

NIST Handbook of Mathematical Functions (2010) (eds F W J Olver, D W Lozier, R F Boisvert, C W Clark) Cambridge University Press

Pearson J (2009) Computation of hypergeometric functions *MSc Dissertation, Mathematical Institute, University of Oxford*

5 Arguments

1: **ani** – double *Input*

On entry: a_i, the nearest integer to a, satisfying $a_i = a - a_r$.

Constraints:

ani $= \lfloor$**ani**$\rfloor$;
$|$**ani**$| \leq arbnd$.

2: **adr** – double *Input*

On entry: a_r, the signed decimal remainder satisfying $a_r = a - a_i$ and $|a_r| \leq 0.5$.

Constraint: $|$**adr**$| \leq 0.5$.

Note: if $|$**adr**$| < 100.0\epsilon$, $a_r = 0.0$ will be used, where ϵ is the ***machine precision*** as returned by nag_machine_precision (X02AJC).

3: **bni** – double *Input*

On entry: b_i, the nearest integer to b, satisfying $b_i = b - b_r$.

Constraints:

bni $= \lfloor$**bni**$\rfloor$;
$|$**bni**$| \leq arbnd$;
if **bdr** $= 0.0$, **bni** > 0.

4: **bdr** – double *Input*

On entry: b_r, the signed decimal remainder satisfying $b_r = b - b_i$ and $|b_r| \leq 0.5$.

Constraint: $|$**bdr**$| \leq 0.5$.

Note: if $|$**bdr** $-$ **adr**$| < 100.0\epsilon$, $a_r = b_r$ will be used, where ϵ is the ***machine precision*** as returned by nag_machine_precision (X02AJC).

5: **x** – double *Input*

On entry: the argument x of the function.

Constraint: $|$**x**$| \leq arbnd$.

6: **frm** – double * *Output*

On exit: m_f, the scaled real component of the solution satisfying $m_f = M(a,b,x) \times 2^{-m_s}$.

Note: if overflow occurs upon completion, as indicated by **fail**.**code** = NW_OVERFLOW_WARN, the value of m_f returned may still be correct. If overflow occurs in a subcalculation, as indicated by **fail**.**code** = NE_OVERFLOW, this should not be assumed.

7: **scm** – Integer * *Output*

On exit: m_s, the scaling power of two, satisfying $m_s = \log_2\left(\frac{M(a,b,x)}{m_f}\right)$.

Note: if overflow occurs upon completion, as indicated by **fail.code** = NW_OVERFLOW_WARN, then $m_s \geq I_{\max}$, where $I_{\max}$ is the largest representable integer (see nag_max_integer (X02BBC)). If overflow occurs during a subcalculation, as indicated by **fail.code** = NE_OVERFLOW, m_s may or may not be greater than $I_{\max}$. In either case, **scm** = nag_max_integer will have been returned.

8: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

NE_OVERFLOW

Overflow occurred in a subcalculation of $M(a, b, x)$.
The answer may be completely incorrect.

NE_REAL

On entry, **adr** = ⟨*value*⟩.
Constraint: $|\mathbf{adr}| \leq 0.5$.

On entry, **bdr** = ⟨*value*⟩.
Constraint: $|\mathbf{bdr}| \leq 0.5$.

NE_REAL_2

On entry, $b = \mathbf{bni} + \mathbf{bdr} =$ ⟨*value*⟩.
$M(a, b, x)$ is undefined when b is zero or a negative integer.

NE_REAL_ARG_NON_INTEGRAL

ani is non-integral.
On entry, **ani** = ⟨*value*⟩.
Constraint: **ani** = ⌊**ani**⌋.

bni is non-integral.
On entry, **bni** = ⟨*value*⟩.
Constraint: **bni** = ⌊**bni**⌋.

NE_REAL_RANGE_CONS

On entry, **ani** = ⟨*value*⟩.
Constraint: $|\mathbf{ani}| \leq arbnd =$ ⟨*value*⟩.

On entry, **bni** = ⟨*value*⟩.
Constraint: $|\mathbf{bni}| \leq arbnd =$ ⟨*value*⟩.

On entry, **x** = ⟨*value*⟩.
Constraint: $|\mathbf{x}| \leq arbnd =$ ⟨*value*⟩.

NE_TOTAL_PRECISION_LOSS

All approximations have completed, and the final residual estimate indicates no accuracy can be guaranteed.
Relative residual $= \langle value \rangle$.

NW_OVERFLOW_WARN

On completion, overflow occurred in the evaluation of $M(a, b, x)$.

NW_SOME_PRECISION_LOSS

All approximations have completed, and the final residual estimate indicates some precision may have been lost.
Relative residual $= \langle value \rangle$.

NW_UNDERFLOW_WARN

Underflow occurred during the evaluation of $M(a, b, x)$.
The returned value may be inaccurate.

7 Accuracy

In general, if **fail.code** = NE_NOERROR, the value of M may be assumed accurate, with the possible loss of one or two decimal places. Assuming the result does not under or overflow, an error estimate res is made internally using equation (1). If the magnitude of res is sufficiently large a different **fail.code** will be returned. Specifically,

fail.code = NE_NOERROR	$res \leq 1000\epsilon$
fail.code = NW_SOME_PRECISION_LOSS	$1000\epsilon < res \leq 0.1$
fail.code = NE_TOTAL_PRECISION_LOSS	$res > 0.1$

A further estimate of the residual can be constructed using equation (1), and the differential identity,

$$\frac{dM(a,b,x)}{dx} = \frac{a}{b}M(a+1,b+1,x),$$

$$\frac{d^2M(a,b,x)}{dx^2} = \frac{a(a+1)}{b(b+1)}M(a+2,b+2,x).$$

This estimate is however dependent upon the error involved in approximating $M(a+1,b+1,x)$ and $M(a+2,b+2,x)$.

8 Parallelism and Performance

nag_specfun_1f1_real_scaled (s22bbc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_specfun_1f1_real_scaled (s22bbc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The values of m_f and m_s are implementation dependent. In most cases, if ${}_1F_1(a;b;x) = 0$, $m_f = 0$ and $m_s = 0$ will be returned, and if ${}_1F_1(a;b;x) = 0$ is finite, the fractional component will be bound by $0.5 \leq \left|m_f\right| < 1$, with m_s chosen accordingly.

The values returned in **frm** (m_f) and **scm** (m_s) may be used to explicitly evaluate $M(a,b,x)$, and may also be used to evaluate products and ratios of multiple values of M as follows,

$$\begin{aligned} M(a,b,x) &= m_f \times 2^{m_s} \\ M(a_1,b_1,x_1) \times M(a_2,b_2,x_2) &= \left(m_{f1} \times m_{f2}\right) \times 2^{(m_{s1}+m_{s2})} \\ \frac{M(a_1,b_1,x_1)}{M(a_2,b_2,x_2)} &= \frac{m_{f1}}{m_{f2}} \times 2^{(m_{s1}-m_{s2})} \\ \ln|M(a,b,x)| &= \ln\left|m_f\right| + m_s \times \ln(2) \end{aligned} .$$

10 Example

This example evaluates the confluent hypergeometric function at two points in scaled form using nag_specfun_1f1_real_scaled (s22bbc), and subsequently calculates their product and ratio without having to explicitly construct M.

10.1 Program Text

```
/* nag_specfun_1f1_real_scaled (s22bbc) Example Program.
 *
 * Copyright 2013 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */

#include <stdio.h>
#include <string.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nags.h>
#include <nagx02.h>

int main(void)
{
  /* Scalars */
  Integer  exit_status = 0;
  Integer  k, maxexponent, scm;
  double   ani, adr, bni, bdr, delta, frm, x;
  /* Arrays */
  double   frmv[2];
  Integer  scmv[2];
  /* Nag Types */
  NagError fail;

  maxexponent = X02BLC;
  printf("nag_specfun_1f1_real_scaled (s22bbc) Example Program Results\n\n");

  ani = -10.0;
  bni = 30.0;
  delta = 1.0E-4;
  adr = delta;
  bdr = -delta;
  x = 25.0;

  printf("%9s%10s%10s%12s%9s%12s\n", "a", "b", "x", "frm", "scm", "M(a,b,x)");
  for (k = 0; k < 2; k++)
    {
      INIT_FAIL(fail);
      /* Compute the real confluent hypergeometric function M(a,b,x) in scaled
       * form using nag_specfun_1f1_real_scaled (s22bbc).
       */
      nag_specfun_1f1_real_scaled(ani, adr, bni, bdr, x, &frm, &scm, &fail);
      switch (fail.code) {
      case NE_NOERROR:
      case NW_UNDERFLOW_WARN:
```

```
      case NW_SOME_PRECISION_LOSS:
        {
          if (scm < maxexponent)
            printf(" %9.4f %9.4f %9.4f %13.4e %5ld %13.4e\n",
                   ani+adr, bni+bdr, x, frm, scm, frm*pow(2.0, scm));
          else
            printf(" %9.4f %9.4f %9.4f %13.4e %5ld %17s\n",
                   ani+adr, bni+bdr, x, frm, scm, "Not Representable");
          frmv[k] = frm;
          scmv[k] = scm;
          break;
        }
      default:
        {
          /* Either the result has overflowed, no accuracy may be assumed,
           * or an input error has been detected.
           */
          printf(" %9.4f %9.4f %9.4f %17s\n", ani+adr, bni+bdr, x, "FAILED");
          exit_status = 1;
          goto END;
          break;
        }
      }
      adr = -adr;
      bdr = -bdr;
    }

  /* Calculate the product M1*M2*/
  frm = frmv[0] * frmv[1];
  scm = scmv[0] + scmv[1];
  printf("\n");
  if (scm < maxexponent)
    printf("%-30s%12.4e%6ld%12.4e\n",
           "Solution product", frm, scm, frm*pow(2.0, scm));
  else
    printf("%-30s%12.4e%6ld%17s\n",
           "Solution product", frm, scm, "Not Representable");

  /* Calculate the ratio M1/M2*/
  if (frmv[1] != 0.0)
    {
      frm = frmv[0]/frmv[1];
      scm = scmv[0] - scmv[1];
      printf("\n");
      if (scm < maxexponent)
        printf("%-30s%12.4e%6ld%12.4e\n",
               "Solution ratio", frm, scm, frm*pow(2.0, scm));
      else
        printf("%-30s%12.4e%6ld%17s\n",
               "Solution ratio", frm, scm, "Not Representable");
    }

 END:
  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_specfun_1f1_real_scaled (s22bbc) Example Program Results

        a         b         x          frm      scm    M(a,b,x)
   -9.9999   29.9999   25.0000   -7.7329e-01   -15   -2.3599e-05
  -10.0001   30.0001   25.0000   -7.7318e-01   -15   -2.3596e-05

Solution product               5.9789e-01   -30  5.5683e-10

Solution ratio                 1.0001e+00     0  1.0001e+00
```