

## NAG C Library Function Document

### nag\_polygamma\_deriv (s14adc)

#### 1 Purpose

nag\_polygamma\_deriv (s14adc) returns a sequence of values of scaled derivatives of the psi function  $\psi(x)$ .

#### 2 Specification

```
void nag_polygamma_deriv (double x, Integer n, Integer m, double ans[],  
NagError *fail)
```

#### 3 Description

nag\_polygamma\_deriv (s14adc) computes  $m$  values of the function

$$w(k, x) = \frac{(-1)^{k+1} \psi^{(k)}(x)}{k!},$$

for  $x > 0$ ,  $k = n, n + 1, \dots, n + m - 1$ , where  $\psi$  is the psi function

$$\psi(x) = \frac{d}{dx} \ln \Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)},$$

and  $\psi^{(k)}$  denotes the  $k$ th derivative of  $\psi$ .

The function is derived from the routine PSIFN in Amos (1983). The basic method of evaluation of  $w(k, x)$  is the asymptotic series

$$w(k, x) \sim \epsilon(k, x) + \frac{1}{2x^{k+1}} + \frac{1}{x^k} \sum_{j=1}^{\infty} B_{2j} \frac{(2j+k-1)!}{(2j)! k! x^{2j}}$$

for large  $x$  greater than a machine-dependent value  $x_{\min}$ , followed by backward recurrence using

$$w(k, x) = w(k, x+1) + x^{-k-1}$$

for smaller values of  $x$ , where  $\epsilon(k, x) = -\ln x$  when  $k = 0$ ,  $\epsilon(k, x) = \frac{1}{kx^k}$  when  $k > 0$ , and  $B_{2j}$ ,  $j = 1, 2, \dots$ , are the Bernoulli numbers.

When  $k$  is large, the above procedure may be inefficient, and the expansion

$$w(k, x) = \sum_{j=1}^{\infty} \frac{1}{(x+j)^{k+1}},$$

which converges rapidly for large  $k$ , is used instead.

#### 4 References

Amos D E (1983) Algorithm 610: A portable FORTRAN subroutine for derivatives of the psi function  
*ACM Trans. Math. Software* **9** 494–502

Abramowitz M and Stegun I A (1972) *Handbook of Mathematical Functions* (3rd Edition) Dover Publications

## 5 Parameters

- 1: **x** – double *Input*  
*On entry:* the argument  $x$  of the function.  
*Constraint:*  $x > 0.0$ .
- 2: **n** – Integer *Input*  
*On entry:* the index of the first member  $n$  of the sequence of functions.  
*Constraint:*  $n \geq 0$ .
- 3: **m** – Integer *Input*  
*On entry:* the number of members  $m$  required in the sequence  $w(k, x)$ , for  $k = n, n+1, \dots, n+m-1$ .  
*Constraint:*  $m \geq 1$ .
- 4: **ans[m]** – double *Output*  
*On exit:* the first  $m$  elements of **ans** contain the required values  $w(k, x)$ , for  $k = n, n+1, \dots, n+m-1$ .
- 5: **fail** – NagError \* *Input/Output*  
The NAG error parameter (see the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_INT

On entry,  $\mathbf{m} = \langle value \rangle$ .  
Constraint:  $\mathbf{m} \geq 1$ .

On entry,  $\mathbf{n} = \langle value \rangle$ .  
Constraint:  $\mathbf{n} \geq 0$ .

### NE\_INTERNAL\_WORKSPACE

There is not enough internal workspace to continue computation.  $\mathbf{m}$  is probably too large.

### NE\_OVERFLOW\_LIKELY

Computation abandoned due to the likelihood of overflow.

### NE\_REAL

On entry,  $\mathbf{x} = \langle value \rangle$ .  
Constraint:  $\mathbf{x} > 0.0$ .

### NE\_UNDERFLOW\_LIKELY

Computation abandoned due to the likelihood of underflow.

### NE\_BAD\_PARAM

On entry, parameter  $\langle value \rangle$  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 consult NAG for assistance.

## 7 Accuracy

All constants in nag\_polygamma\_deriv (s14adc) are given to approximately 18 digits of precision. Calling the number of digits of precision in the floating-point arithmetic being used  $t$ , then clearly the maximum number of correct digits in the results obtained is limited by  $p = \min(t, 18)$ . Empirical tests of nag\_polygamma\_deriv (s14adc), taking values of  $x$  in the range  $0.0 < x < 50.0$ , and  $n$  in the range  $1 \leq n \leq 50$ , have shown that the maximum relative error is a loss of approximately two decimal places of precision. Tests with  $n = 0$ , i.e., testing the function  $-\psi(x)$ , have shown somewhat better accuracy, except at points close to the zero of  $\psi(x)$ ,  $x \approx 1.461632$ , where only absolute accuracy can be obtained.

## 8 Further Comments

The time taken for a call of nag\_polygamma\_deriv (s14adc) is approximately proportional to  $m$ , plus a constant. In general, it is much cheaper to call nag\_polygamma\_deriv (s14adc) with  $m$  greater than 1 to evaluate the function  $w(k, x)$ , for  $k = n, n + 1, \dots, n + m - 1$ , rather than to make  $m$  separate calls of nag\_polygamma\_deriv (s14adc).

## 9 Example

The example program reads values of the argument  $x$  from a file, evaluates the function at each value of  $x$  and prints the results.

### 9.1 Program Text

```
/* nag_polygamma_deriv (s14adc) Example Program
 *
 * Copyright 2002 Numerical Algorithms Group.
 *
 * Mark 7, 2002.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdl�.h>
#include <nags.h>

int main(void)
{
    double x, w[4];
    int n, m;

    /* Skip heading in data file */
    vscanf("%*[^\n]");
    Vprintf("s14adc Example Program Results\n");
    Vprintf("      x      w(0,x)      w(1,x)      w(2,x)      w(3,x)\n");
    while (scanf("%lf", &x) != EOF)
    {
        n = 0;
        m = 4;
        s14adc(x, n, m, w, NAGERR_DEFAULT);
        Vprintf("%12.4e %12.4e %12.4e %12.4e %12.4e\n",
                x, w[0], w[1], w[2], w[3]);
    }
    return EXIT_SUCCESS;
}
```

### 9.2 Program Data

```
s14adc Example Program Data
0.1
0.5
3.6
8.0
```

### 9.3 Program Results

s14adc Example Program Results

x	w(0,x)	w(1,x)	w(2,x)	w(3,x)
1.0000e-01	1.0424e+01	1.0143e+02	1.0009e+03	1.0001e+04
5.0000e-01	1.9635e+00	4.9348e+00	8.4144e+00	1.6235e+01
3.6000e+00	-1.1357e+00	3.1988e-01	5.0750e-02	1.0653e-02
8.0000e+00	-2.0156e+00	1.3314e-01	8.8498e-03	7.8321e-04