NAG Library Function Document

nag_legendre_p (s22aac)

1   Purpose

nag_legendre_p (s22aac) returns a sequence of values for either the unnormalized or normalized Legendre functions of the first kind $P_n^m(x)$ or $\overline{P_n^m}(x)$ for real x of a given order m and degree $n = 0, 1, \ldots, N$.

2   Specification

```
#include <nag.h>
#include <nags.h>
void nag_legendre_p (Integer mode, double x, Integer m, Integer nl,
    double p[], NagError *fail)
```

3   Description

nag_legendre_p (s22aac) evaluates a sequence of values for either the unnormalized or normalized Legendre ($m = 0$) or associated Legendre ($m \neq 0$) functions of the first kind $P_n^m(x)$ or $\overline{P_n^m}(x)$, where x is real with $-1 \leq x \leq 1$, of order m and degree $n = 0, 1, \ldots, N$ defined by

$$P_n^m(x) \quad = \quad (1 - x^2)^{m/2} \frac{d^m}{dx^m} P_n(x) \qquad\qquad \text{if } m \geq 0,$$

$$P_n^m(x) \quad = \quad \frac{(n+m)!}{(n-m)!} P_n^{-m}(x) \qquad\qquad \text{if } m < 0 \quad \text{and}$$

$$\overline{P_n^m}(x) \quad = \quad \sqrt{\frac{(2n+1)}{2} \frac{(n-m)!}{(n+m)!}} P_n^m(x)$$

respectively; $P_n(x)$ is the (unassociated) Legendre polynomial of degree n given by

$$P_n(x) \equiv P_n^0(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n$$

(the *Rodrigues formula*). Note that some authors (e.g., Abramowitz and Stegun (1972)) include an additional factor of $(-1)^m$ (the *Condon–Shortley Phase*) in the definitions of $P_n^m(x)$ and $\overline{P_n^m}(x)$. They use the notation $P_{mn}(x) \equiv (-1)^m P_n^m(x)$ in order to distinguish between the two cases.

nag_legendre_p (s22aac) is based on a standard recurrence relation described in Section 8.5.3 of Abramowitz and Stegun (1972). Constraints are placed on the values of m and n in order to avoid the possibility of machine overflow. It also sets the appropriate elements of the array **p** (see Section 5) to zero whenever the required function is not defined for certain values of m and n (e.g., $m = -5$ and $n = 3$).

4   References

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

5 Arguments

1: **mode** – Integer Input

On entry: indicates whether the sequence of function values is to be returned unnormalized or normalized.

mode $= 1$
The sequence of function values is returned unnormalized.

mode $= 2$
The sequence of function values is returned normalized.

Constraint: **mode** $= 1$ or 2.

2: **x** – double Input

On entry: the argument x of the function.

Constraint: $\text{abs}(\mathbf{x}) \le 1.0$.

3: **m** – Integer Input

On entry: the order m of the function.

Constraint: $\text{abs}(\mathbf{m}) \le 27$.

4: **nl** – Integer Input

On entry: the degree N of the last function required in the sequence.

Constraints:

$\mathbf{nl} \ge 0$;
if $\mathbf{m} = 0$, $\mathbf{nl} \le 100$;
if $\mathbf{m} \ne 0$, $\mathbf{nl} \le 55 - \text{abs}(\mathbf{m})$.

5: **p**$[\mathbf{nl} + \mathbf{1}]$ – double Output

On exit: the required sequence of function values as follows:

if **mode** $= 1$, **p**$[n]$ contains $P_n^m(x)$, for $n = 0, 1, \ldots, N$;

if **mode** $= 2$, **p**$[n]$ contains $\overline{P_n^m}(x)$, for $n = 0, 1, \ldots, N$.

6: **fail** – NagError * Input/Output

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT

On entry, $|\mathbf{m}| = \langle value \rangle$.
Constraint: $|\mathbf{m}| \le 27$.

On entry, **mode** $= \langle value \rangle$.
Constraint: **mode** ≤ 2.

On entry, **mode** $= \langle value \rangle$.
Constraint: **mode** ≥ 1.

On entry, **nl** $= \langle value \rangle$.
Constraint: **nl** ≥ 0.

NE_INT_2

On entry, **nl** $= \langle value \rangle$ and $|\mathbf{m}| = \langle value \rangle$.
Constraint: **nl** $+ |\mathbf{m}| \leq 55$.

On entry, **nl** $= \langle value \rangle$.
Constraint: **nl** ≤ 100 when $\mathbf{m} = 0$.

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.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_REAL

On entry, $|\mathbf{x}| = \langle value \rangle$.
Constraint: $|\mathbf{x}| \leq 1.0$.

7 Accuracy

The computed function values should be accurate to within a small multiple of the ***machine precision*** except when underflow (or overflow) occurs, in which case the true function values are within a small multiple of the underflow (or overflow) threshold of the machine.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example reads the values of the arguments x, m and N from a file, calculates the sequence of unnormalized associated Legendre function values $P_n^m(x), P_{n+1}^m(x), \ldots, P_{n+N}^m(x)$, and prints the results.

10.1 Program Text

```
/* nag_legendre_p (s22aac) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * NAG C Library
 *
 * Mark 6, 2000.
 * Mark 7 revised, 2001.
 * Mark 8 revised, 2004.
 *
```

```
 */

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

int main(void)
{
  Integer  exit_status = 0, m, mode, n, nl;
  NagError fail;
  char     *str = 0;
  double   *p = 0, x;

  INIT_FAIL(fail);

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  printf("nag_legendre_p (s22aac) Example Program Results\n");
#ifdef _WIN32
  scanf_s("%"NAG_IFMT" %lf %"NAG_IFMT" %"NAG_IFMT"", &mode, &x, &m, &nl);
#else
  scanf("%"NAG_IFMT" %lf %"NAG_IFMT" %"NAG_IFMT"", &mode, &x, &m, &nl);
#endif
  if (!(p = NAG_ALLOC(nl+1, double)) ||
      !(str = NAG_ALLOC(80, char)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  if (mode == 1)
    {
      if (m == 0)
#ifdef _WIN32
        strcpy_s(str, 80, "Unnormalized Legendre function values\n");
#else
        strcpy(str, "Unnormalized Legendre function values\n");
#endif
      else
#ifdef _WIN32
        strcpy_s(str, 80, "Unnormalized associated Legendre function values\n");
#else
        strcpy(str, "Unnormalized associated Legendre function values\n");
#endif
    }
  else if (mode == 2)
    {
      if (m == 0)
#ifdef _WIN32
        strcpy_s(str, 80, "Normalized Legendre function values\n");
#else
        strcpy(str, "Normalized Legendre function values\n");
#endif
      else
#ifdef _WIN32
        strcpy_s(str, 80, "Normalized associated Legendre function values\n");
#else
        strcpy(str, "Normalized associated Legendre function values\n");
#endif
    }

  /* nag_legendre_p (s22aac).
   * Legendre and associated Legendre functions of the first
   * kind with real arguments
   */
```

```
nag_legendre_p(mode, x, m, nl, p, &fail);
printf("mode      x      m    nl\n");
printf("%3"NAG_IFMT"    %5.1f%6"NAG_IFMT"%6"NAG_IFMT"\n\n", mode, x, m, nl);

if (fail.code == NE_NOERROR)
  {
    printf(str);
    printf("\n");
    printf(" n       P(n)\n");
    for (n = 0; n <= nl; ++n)
      printf("%2"NAG_IFMT" %13.4e\n", n, p[n]);
  }
else
  {
    printf("Error from nag_legendre_p (s22aac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
END:
 NAG_FREE(p);
 NAG_FREE(str);
 return exit_status;
}
```

10.2 Program Data

```
nag_legendre_p (s22aac) Example Program Data
 1   0.5   2   3  : Values of mode, x, m and nl
```

10.3 Program Results

```
nag_legendre_p (s22aac) Example Program Results
mode      x      m    nl
  1      0.5     2     3

Unnormalized associated Legendre function values

 n       P(n)
 0   0.0000e+00
 1   0.0000e+00
 2   2.2500e+00
 3   5.6250e+00
```