# NAG C 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

```
void nag_legendre_p (Integer mode, double x, Integer m, Integer nl, double p[],
          NagError *fail)
```

## 3 Description

This routine 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

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

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

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

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 is based on a standard recurrence relation given by Abramowitz and Stegun (Abramowitz and Stegun (1972), 8.5.3). 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 4) to zero whenever the required function is not defined for certain values of $m$ and $n$ (e.g., $m = -5$ and $n = 3$).

## 4 Parameters

1: **mode** – Integer *Input*

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

if **mode** = 1, then the sequence of function values is returned unnormalized;

if **mode** = 2, then 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:* abs(**x**) $\leq$ 1.0.

3: **m** – Integer *Input*

*On entry:* the order $m$ of the function.

*Constraint:* abs(**m**) $\leq$ 27.

4: **nl** – Integer *Input*

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

*Constraints:*

**nl** $\geq$ 0,

**nl** $\leq$ 100 when **m** $=$ 0,

**nl** $\leq$ 55 $-$ abs(**m**) when **m** $\neq$ 0.

5: **p[nl+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 parameter (see the Essential Introduction).

# 5 Error Indicators and Warnings

**NE_REAL**

On entry, **x** $=$ <*value*>.
Constraint: abs(**x**) $\leq$ 1.0.

**NE_INT**

On entry, **mode** $=$ <*value*>.
Constraint: **mode** $=$ 1 or 2.

On entry, **nl** $=$ <*value*>.
Constraint: **nl** $\geq$ 0.

On entry, **m** $=$ <*value*>.
Constraint: abs(**m**) $\leq$ 27.

**NE_INT_2**

On entry, **nl** $=$ <*value*>, **m** $=$ <*value*>.
Constraint: **nl** $\leq$ 100 when **m** $=$ 0.

On entry, **nl** $=$ <*value*>, **m** $=$ <*value*>.
Constraint: **nl** $\leq$ 55 $-$ abs(**m**) when **m** $\neq$ 0.

# 6 Further Comments

## 6.1 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.

### 6.2 References

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

## 7 See Also

None.

## 8 Example

The following program 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.

### 8.1 Program Text

```
/* nag_legendre_p (s22aac) Example Program.
 *
 * Copyright 2000 Numerical Algorithms Group.
 *
 * NAG C Library
 *
 * Mark 6, 2000.
 */

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

int main(void)
{
  const char fmt_99998[] = "%2ld %12.4e\n";
  const char fmt_99999[] = "%3ld    %5.1f%6ld%6ld\n\n";
  char str[80];
  double p[101];
  double x;
  Integer exit_status=0;
  NagError fail;
  Integer m, mode, n, nl;

  INIT_FAIL(fail);
  Vprintf("s22aac Example Program Results\n\n");

  /*     Skip heading in data file */
  Vscanf("%*[^\n] ");
  Vscanf("%ld %lf %ld %ld", &mode, &x, &m, &nl);
  if (mode == 1)
    {
      if (m == 0)
 {
   Vstrcpy(str, "Unnormalized Legendre function values\n");
 }
      else
 {
   Vstrcpy(str, "Unnormalized associated Legendre function values\n");
 }
    }
  else if (mode == 2)
    {
```

```
      if (m == 0)
 {
   Vstrcpy(str, "Normalized Legendre function values\n");
 }
      else
 {
   Vstrcpy(str, "Normalized associated Legendre function values\n");
 }
    }

  s22aac  (mode, x, m, nl, p, &fail);
  Vprintf("mode      x      m    nl\n\n");
  Vprintf(fmt_99999, mode, x, m, nl);

  if (fail.code == NE_NOERROR)
    {
      Vprintf(str);
      Vprintf("\n");
      Vprintf(" n      P(n)\n");
      for (n = 0; n <= nl; ++n)
        {
          Vprintf(fmt_99998,n,p[n]);
        }
    }
  else
    {
      Vprintf("Error from s22aac.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
 END:
  return exit_status;
}
```

## 8.2 Program Data

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

## 8.3 Program Results

```
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
```