NAG Library Function Document

nag_quad_1d_gauss_wset (d01tbc)

1 Purpose

nag_quad_1d_gauss_wset (d01tbc) returns the weights and abscissae appropriate to a Gaussian quadrature formula with a specified number of abscissae. The formulae provided are for Gauss–Legendre, rational Gauss, Gauss–Laguerre and Gauss–Hermite.

2 Specification

```c
#include <nag.h>
#include <nagd01.h>

void nag_quad_1d_gauss_wset (Nag_QuadType quad_type, double a, double b,
   Integer n, double weight[], double abscis[], NagError *fail)
```

3 Description

nag_quad_1d_gauss_wset (d01tbc) returns the weights and abscissae for use in the Gaussian quadrature of a function \( f(x) \). The quadrature takes the form

\[
S = \sum_{i=1}^{n} w_i f(x_i)
\]

where \( w_i \) are the weights and \( x_i \) are the abscissae (see Davis and Rabinowitz (1975), Fröberg (1970), Ralston (1965) or Stroud and Secrest (1966)).

Weights and abscissae are available for Gauss–Legendre, rational Gauss, Gauss–Laguerre and Gauss–Hermite quadrature, and for a selection of values of \( n \) (see Section 5).

(a) Gauss–Legendre Quadrature:

\[
S \approx \int_{a}^{b} f(x) \, dx
\]

where \( a \) and \( b \) are finite and it will be exact for any function of the form

\[
f(x) = \sum_{i=0}^{2n-1} c_i x^i.
\]

(b) Rational Gauss quadrature, adjusted weights:

\[
S \approx \int_{a}^{b} f(x) \, dx (a + b > 0) \quad \text{or} \quad S \approx \int_{-\infty}^{a} f(x) \, dx (a + b < 0)
\]

and will be exact for any function of the form

\[
f(x) = \sum_{i=0}^{2n-1} c_i (x + b)^i
\]

(c) Gauss–Laguerre quadrature, adjusted weights:

\[
S \approx \int_{0}^{\infty} f(x) \, dx (b > 0) \quad \text{or} \quad S \approx \int_{-\infty}^{a} f(x) \, dx (b < 0)
\]

and will be exact for any function of the form
\[ f(x) = e^{-bx} \sum_{i=0}^{2n-1} c_i x^i. \]

(d) Gauss–Hermite quadrature, adjusted weights:

\[ S \simeq \int_{-\infty}^{+\infty} f(x) \, dx \]

and will be exact for any function of the form

\[ f(x) = e^{-b(x-a)^2} \sum_{i=0}^{2n-1} c_i x^i \quad (b > 0). \]

(e) Gauss–Laguerre quadrature, normal weights:

\[ S \simeq \int_{a}^{\infty} e^{-bx} f(x) \, dx \quad (b > 0) \quad \text{or} \quad S \simeq \int_{-\infty}^{a} e^{-bx} f(x) \, dx \quad (b < 0) \]

and will be exact for any function of the form

\[ f(x) = \sum_{i=0}^{2n-1} c_i x^i. \]

(f) Gauss–Hermite quadrature, normal weights:

\[ S \simeq \int_{-\infty}^{+\infty} e^{-b(x-a)^2} f(x) \, dx \]

and will be exact for any function of the form

\[ f(x) = \sum_{i=0}^{2n-1} c_i x^i. \]

Note: the Gauss–Legendre abscissae, with \( a = -1, b = +1 \), are the zeros of the Legendre polynomials; the Gauss–Laguerre abscissae, with \( a = 0, b = 1 \), are the zeros of the Laguerre polynomials; and the Gauss–Hermite abscissae, with \( a = 0, b = 1 \), are the zeros of the Hermite polynomials.

### 4 References


### 5 Arguments

1. \texttt{quad_type} – \texttt{Nag QuadType} \hspace{1cm} Input

   *On entry:* indicates the quadrature formula.

   \texttt{quad_type} = \texttt{Nag Quad Gauss Legendre}

   Gauss–Legendre quadrature on a finite interval, using normal weights.

   \texttt{quad_type} = \texttt{Nag Quad Gauss Laguerre}

   Gauss–Laguerre quadrature on a semi-infinite interval, using normal weights.

   \texttt{quad_type} = \texttt{Nag Quad Gauss Laguerre Adjusted}

   Gauss–Laguerre quadrature on a semi-infinite interval, using adjusted weights.
quad_type = Nag_Quad_Gauss_Hermite
  Gauss–Hermite quadrature on an infinite interval, using normal weights.

quad_type = Nag_Quad_Gauss_Hermite_Adjusted
  Gauss–Hermite quadrature on an infinite interval, using adjusted weights.

quad_type = Nag_Quad_Gauss_Rational_Adjusted
  Rational Gauss quadrature on a semi-infinite interval, using adjusted weights.

Constraint: quad_type = Nag_Quad_Gauss_Legendre, Nag_Quad_Gauss_Laguerre,
Nag_Quad_Gauss_Laguerre_Adjusted, Nag_Quad_Gauss_Hermite,
Nag_Quad_Gauss_Hermite_Adjusted or Nag_Quad_Gauss_Rational_Adjusted.

2:  a – double
3:  b – double

On entry: the quantities a and b as described in the appropriate sub-section of Section 3.

Constraints:
  if quad_type = Nag_Quad_Gauss_Rational_Adjusted, a + b ≠ 0.0;
  if quad_type = Nag_Quad_Gauss_Laguerre or Nag_Quad_Gauss_Laguerre_Adjusted,
    b ≠ 0.0;
  if quad_type = Nag_Quad_Gauss_Hermite or Nag_Quad_Gauss_Hermite_Adjusted, b > 0.0.

Constraints:
  Rational Gauss: a + b ≠ 0.0;
  Gauss–Laguerre: b ≠ 0.0;
  Gauss–Hermite: b > 0.

4:  n – Integer

On entry: n, the number of weights and abscissae to be returned.

Constraint: n = 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 16, 20, 24, 32, 48 or 64.

Note: if n > 0 and is not a member of the above list, the maximum value of n stored below n will be used, and all subsequent elements of abscis and weight will be returned as zero.

5:  weight[n] – double

On exit: the n weights.

6:  abscis[n] – double

On exit: the n abscissae.

7:  fail – NagError *

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
  The value of a and/or b is invalid for the chosen quad_type. Either:
On entry, argument \(\langle\text{value}\rangle\) had an illegal value.
The value of \(a\) and/or \(b\) is invalid for Gauss-Hermite quadrature.
On entry, \(\text{quad\_type} = \langle\text{value}\rangle\).
On entry, \(a = \langle\text{value}\rangle\) and \(b = \langle\text{value}\rangle\).
Constraint: \(b > 0.0\).
The value of \(a\) and/or \(b\) is invalid for Gauss-Laguerre quadrature.
On entry, \(\text{quad\_type} = \langle\text{value}\rangle\).
On entry, \(a = \langle\text{value}\rangle\) and \(b = \langle\text{value}\rangle\).
Constraint: \(|b| > 0.0\).
The value of \(a\) and/or \(b\) is invalid for rational Gauss quadrature.
On entry, \(\text{quad\_type} = \langle\text{value}\rangle\).
On entry, \(a = \langle\text{value}\rangle\) and \(b = \langle\text{value}\rangle\).
Constraint: \(|a + b| > 0.0\).

**NE_INT**

On entry, \(n = \langle\text{value}\rangle\).
Constraint: \(n > 0\).

**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_QUAD_GAUSS_NPTS_RULE**

The \(n\)-point rule is not among those stored.
On entry: \(n = \langle\text{value}\rangle\).
\(n\)-rule used: \(n = \langle\text{value}\rangle\).

**NE_TOO_SMALL**

Underflow occurred in calculation of normal weights.
Reduce \(n\) or use adjusted weights: \(n = \langle\text{value}\rangle\).

**NE_WEIGHT_ZERO**

No nonzero weights were generated for the provided parameters.

7 **Accuracy**
The weights and abscissae are stored for standard values of \(a\) and \(b\) to full machine accuracy.

8 **Parallelism and Performance**
Not applicable.

9 **Further Comments**
Timing is negligible.
10 Example

This example returns the abscissae and (adjusted) weights for the six-point Gauss–Laguerre formula.

10.1 Program Text

/* nag_quad_1d_gauss_wset (d01tbc) Example Program.
 * Copyright 2014 Numerical Algorithms Group.
 * Mark 23, 2011.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd01.h>

int main(void)
{
    Integer exit_status = 0;
    Integer n;
    double a, b;
    Integer i;
    Nag_QuadType quadtype;
    NagError fail;
    double *abscis = 0, *weight = 0;

    INIT_FAIL(fail);

    printf("nag_quad_1d_gauss_wset (d01tbc) Example Program Results\n");

    /* Skip heading in data file */
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif
    /* Input a, b and n */
    #ifdef _WIN32
        scanf("%lf %lf", &a, &b);
    #else
        scanf("%lf %lf", &a, &b);
    #endif
    #ifdef _WIN32
        scanf("%"NAG_IFMT"%*[\n] ", &n);
    #else
        scanf("%"NAG_IFMT"%*[\n] ", &n);
    #endif
    quadtype = Nag_Quad_Gauss_Laguerre_Adjusted;

    if (!(abscis = NAG_ALLOC(n, double)) ||
        !(weight = NAG_ALLOC(n, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* nag_quad_1d_gauss_wset (d01tbc).
     * Pre-computed weights and abscissae for
     * Gaussian quadrature rules, restricted choice of rule.
     */
    nag_quad_1d_gauss_wset(quadtype, a, b, n, weight, abscis, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_quad_1d_gauss_wset (d01tbc).\n%\n", fail.message);
        exit_status = 1;
        goto END;
    }

END:

return exit_status;
}
Laguerre formula, %3"NAG_IFMT" points

Abscissae    Weights

for (i = 0; i < n; i++)
{
    printf("%15.6e", abscis[i]);
    printf("%15.6e\n", weight[i]);
}
printf("\n");

END:
NAG_FREE(abscis);
NAG_FREE(weight);
return exit_status;

10.2 Program Data

None.

10.3 Program Results

nag_quad_1d_gauss_wset (d01tbc) Example Program Results

Laguerre formula, 6 points

<table>
<thead>
<tr>
<th>Abscissae</th>
<th>Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.228466e-01</td>
<td>5.735355e-01</td>
</tr>
<tr>
<td>1.188932e+00</td>
<td>1.369253e+00</td>
</tr>
<tr>
<td>2.992736e+00</td>
<td>1.369253e+00</td>
</tr>
<tr>
<td>5.775144e+00</td>
<td>3.350525e+00</td>
</tr>
<tr>
<td>9.837467e+00</td>
<td>4.886827e+00</td>
</tr>
<tr>
<td>1.598287e+01</td>
<td>7.849016e+00</td>
</tr>
</tbody>
</table>