Module 11.4: nag_quad_util
Numerical Integration Utilities

nag_quad_util provides utility procedures for computation involving integration of functions, e.g., the computation of the weights and abscissae for Gaussian integration rules.

Contents

Introduction

This module provides procedures which perform tasks in numerical integration. Currently, one procedure is available which computes the weights (normal or adjusted) and abscissae for six different types of Gaussian integration rules.

Procedure: nag_quad_gs_wt_absc

1 Description

nag_quad_gs_wt_absc computes the weights (normal or adjusted) and abscissae associated with six different Gauss rules. These weights and abscissae may be used for the approximation of definite integrals (see Stroud and Secrest [2], or pages 73–105 of Davis and Rabinowitz [1]).

2 Usage

```
USE nag_quad_util
CALL nag_quad_gs_wt_absc(rule, a, b, wt, x  [, optional arguments])
```

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '$\mathbf{x}(n)$' is used in the argument descriptions to specify that the array x must have exactly n elements.

This procedure derives the value of the following problem parameter from the shape of the supplied arrays.

$n \geq 1$ — the number of Gaussian weights and abscissae

3.1 Mandatory Arguments

rule — integer, intent(in)

Input: indicates the quadrature rule to be used as specified in Section 6.1:

if rule = 0 for the Gauss–Legendre rule;
if rule = 1 for the Gauss–Jacobi rule with normal weights;
if rule = −1 for the Gauss–Jacobi rule with adjusted weights;
if rule = 2 for the Gauss–Exponential rule with normal weights;
if rule = −2 for the Gauss–Exponential rule with adjusted weights;
if rule = 3 for the Gauss–Laguerre rule with normal weights;
if rule = −3 for the Gauss–Laguerre rule with adjusted weights;
if rule = 4 for the Gauss–Hermite rule with normal weights;
if rule = −4 for the Gauss–Hermite rule with adjusted weights;
if rule = 5 for the Gauss–Rational rule with normal weights;
if rule = −5 for the Gauss–Rational rule with adjusted weights.

Constraints: $-5 \leq$ rule ≤ 5.

a — real(kind=*wp*), intent(in)

b — real(kind=*wp*), intent(in)

Input: the parameters a and b which occur in the quadrature formulae as specified in Section 6.1.

Constraints:

If rule = 0, ±1 or ±2, then a < b;
if rule = ±3, then b ≠ 0;
if rule = ±4, then b > 0;
if rule = ±5, then a + b ≠ 0.

wt(n) — real(kind=*wp*), intent(out)

x(n) — real(kind=*wp*), intent(out)

Output: the n weights and abscissae of the Gauss rule.

Note: if `rule` $= -2$ or -4 and the optional argument `c` $\neq 0.0$, then an odd number of weights and abscissae, n, may raise an error condition (see `error%code` = 204 in Section 4).

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

c — real(kind=*wp*), intent(in), optional

d — real(kind=*wp*), intent(in), optional

Input: the parameters c and d which occur in some of the quadrature formulae (see Section 6.1). For some rules `c` and `d` must not be too large (see Section 4).

Note:

If `rule` $= 0$, then `c` and `d` should not be present;

if `rule` $= \pm 2, \pm 3$ or ± 4, then `d` should not be present.

Constraints:

if `rule` $= \pm 1$, then `c` > -1 and `d` > -1;

if `rule` $= \pm 2, \pm 3$ or ± 4, then `c` > -1;

if `rule` $= \pm 5$, then `c` > -1 and `d` $>$ `c` $+1$.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document `nag_error_handling` (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to `nag_set_error` before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.
303	Array arguments have inconsistent shapes.
305	Invalid absence of an optional argument.

Failures (error%level = 2):

error%code	Description
201	Convergence not achieved. `nag_sym_tridiag_eig_all` has failed to obtain convergence.
202	One or more of the weights are larger than `HUGE(1.0_wp)`, the largest floating-point number on this machine. The overflowing weights are returned as `HUGE(1.0_wp)`. Possible solutions are to use a smaller value of n; or, if using adjusted weights, to change to normal weights. To examine the values of the weights after this failure, you must supply the *optional* argument `error` and initialise it using `nag_set_error` and set `halt_level` to 3 (see the Essential Introduction or the module `nag_error_handling` for handling error exits).

203 One or more of the weights are too small to be distinguished from zero on this machine.

The underflowing weights are returned as zero, which may be a usable approximation. Possible solutions are to use a smaller value of n; or, if using normal weights, to change to adjusted weights. To examine the values of the weights after this failure, you must supply the *optional* argument `error` and initialise it using `nag_set_error` and set `halt_level` to 3 (see the Essential Introduction or the module `nag_error_handling` for handling error exits).

204 Gauss–Exponential or Gauss–Hermite adjusted weights with n odd and $\mathtt{c} \neq 0.0$.

Theoretically, in these cases:

for $\mathtt{c} > 0.0$, the central adjusted weight is infinite, and the exact function $f(x)$ is zero at the central abscissa;

for $\mathtt{c} < 0.0$, the central adjusted weight is zero, and the exact function $f(x)$ is infinite at the central abscissa.

In either case, the contribution of the central abscissa to the summation is indeterminate.

In practice, the central weight may not have overflowed or underflowed, if there is sufficient rounding error in the value of the central abscissa.

If the soft fail option is used, the weights and abscissa returned may be usable; the user must be particularly careful not to *round* the central abscissa to its true value without simultaneously *rounding* the central weight to zero or ∞ as appropriate, or the summation will suffer. It would be preferable to use normal weights, if possible.

It is important to remember that, when switching from normal weights to adjusted weights or vice versa, redefinition of $f(x)$ is involved.

Warnings (error%level = 1):

error%code	Description
101	Optional argument present but not used

If `rule` = 0, `c` and `d` should not be present;
if `rule` = ±2 or ±3 or ±4, `d` should not be present.

5 Examples of Usage

A complete example of the use of this procedure appears in Example 1 of this module document.

6 Further Comments

6.1 Mathematical Background

This procedure computes the weights w_i and abscissae x_i for use in the summation

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

which approximates a definite integral (see Stroud and Secrest [2], or pages 73–105 of Davis and Rabinowitz [1]). The following types are provided:

1. Gauss–Legendre:

$$S \simeq \int_a^b f(x)\,dx, \quad \text{exact for } \; f(x) = P_{2n-1}(x).$$

Constraint: $b > a$.

2. Gauss–Jacobi:

normal weights:

$$S \simeq \int_a^b (b-x)^c (x-a)^d f(x)\,dx, \quad \text{exact for } \; f(x) = P_{2n-1}(x),$$

adjusted weights:

$$S \simeq \int_a^b f(x)\,dx, \quad \text{exact for } \; f(x) = (b-x)^c (x-a)^d P_{2n-1}(x).$$

Constraint: $c > -1$, $d > -1$, $b > a$.

3. Gauss–Exponential:

normal weights:

$$S \simeq \int_a^b \left| x - \frac{a+b}{2} \right|^c f(x)\,dx, \quad \text{exact for } \; f(x) = P_{2n-1}(x),$$

adjusted weights:

$$S \simeq \int_a^b f(x)\,dx, \quad \text{exact for } \; f(x) = \left| x - \frac{a+b}{2} \right|^c P_{2n-1}(x).$$

Constraint: $c > -1$, $b > a$.

4. Gauss–Laguerre:

normal weights:

$$\begin{aligned} S \; &\simeq \int_a^\infty |x-a|^c e^{-bx} f(x)\,dx \;\; (b > 0), \\ &\simeq \int_{-\infty}^a |x-a|^c e^{-bx} f(x)\,dx \;\; (b < 0), \quad \text{exact for } \; f(x) = P_{2n-1}(x), \end{aligned}$$

adjusted weights:

$$\begin{aligned} S \; &\simeq \int_a^\infty f(x)\,dx \;\; (b > 0), \\ &\simeq \int_{-\infty}^a f(x)\,dx \;\; (b < 0), \quad \text{exact for } \; f(x) = |x-a|^c e^{-bx} P_{2n-1}(x). \end{aligned}$$

Constraint: $c > -1$, $b \neq 0$.

5. Gauss–Hermite:

normal weights:

$$S \simeq \int_{-\infty}^{+\infty} |x-a|^c e^{-b(x-a)^2} f(x)\,dx, \quad \text{exact for } \; f(x) = P_{2n-1}(x),$$

adjusted weights:

$$S \simeq \int_{-\infty}^{+\infty} f(x)\,dx, \quad \text{exact for } \; f(x) = |x-a|^c e^{-b(x-a)^2} P_{2n-1}(x).$$

Constraint: $c > -1$, $b > 0$.

6. Gauss–Rational:

   normal weights:

$$S \simeq \int_a^\infty \frac{|x-a|^c}{|x+b|^d} f(x)\,dx \quad (a+b>0),$$

$$\simeq \int_{-\infty}^a \frac{|x-a|^c}{|x+b|^d} f(x)\,dx \quad (a+b<0), \text{ exact for } f(x) = P_{2n-1}\left(\frac{1}{x+b}\right),$$

   adjusted weights:

$$S \simeq \int_a^\infty f(x)\,dx \quad (a+b>0),$$

$$\simeq \int_{-\infty}^a f(x)\,dx \quad (a+b<0), \text{ exact for } f(x) = \frac{|x-a|^c}{|x+b|^d} P_{2n-1}\left(\frac{1}{x+b}\right).$$

   Constraint: $c > -1$, $d > c+1$, $a+b \neq 0$.

In the above formulae, $P_{2n-1}(x)$ stands for any polynomial of degree $2n-1$ or less in x.

6.2 Algorithmic Detail

The method used is to calculate the abscissae, using `nag_sym_tridiag_eig_all` to find the eigenvalues of the appropriate tridiagonal matrix (see Golub and Welsch [3]). The weights are then determined by the formula:

$$w_i = \left(\sum_{j=0}^{n-1} P_j^*(x_i)^2\right)^{-1}$$

where $P_j^*(x)$ is the jth orthogonal polynomial with respect to the weight function over the appropriate interval.

6.3 Accuracy

The accuracy depends mainly on n, with increasing loss of accuracy for larger values of n. Typically, one or two decimal digits may be lost from machine accuracy with $n \simeq 20$, and three or four decimal digits may be lost for $n \simeq 100$.

6.4 Timing

The major portion of the time is taken up by `nag_sym_tridiag_eig_all` during the calculation of the abscissae, where the time is roughly proportional to n^3.

Example 1: Abscissae and weights by Gauss–Legendre formula

Generates abscissae and weights using the Gauss–Legendre 7-point formula.

1 Program Text

Note. The listing of the example program presented below is double precision. Single precision users are referred to Section 5.2 of the Essential Introduction for further information.

```
PROGRAM nag_quad_util_ex01
  ! Example Program Text for nag_quad_util
  ! NAG fl90, Release 3. NAG Copyright 1997.
  ! .. Use Statements ..
  USE nag_examples_io, ONLY : nag_std_out
  USE nag_quad_util, ONLY : nag_quad_gs_wt_absc
  ! .. Implicit None Statement ..
  IMPLICIT NONE
  ! .. Intrinsic Functions ..
  INTRINSIC KIND
  ! .. Parameters ..
  INTEGER, PARAMETER :: n = 7
  INTEGER, PARAMETER :: wp = KIND(1.0D0)
  ! .. Local Scalars ..
  INTEGER :: j, rule
  REAL (wp) :: a, b
  ! .. Local Arrays ..
  REAL (wp) :: wt(n), x(n)
  ! .. Executable Statements ..

  WRITE (nag_std_out,*) 'Example Program Results for nag_quad_util_ex01'
  a = -1.0_wp
  b = 1.0_wp
  rule = 0

  CALL nag_quad_gs_wt_absc(rule,a,b,wt,x)

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,'(1X,A,I3,A)') 'Legendre formula,', n, ' points'
  WRITE (nag_std_out,*)
  WRITE (nag_std_out,'(6X,A,8X,A)') 'Abscissae', 'Weights'
  WRITE (nag_std_out,*)
  WRITE (nag_std_out,'(2F15.5)') (x(j),wt(j),j=1,n)
END PROGRAM nag_quad_util_ex01
```

2 Program Data

None.

3 Program Results

```
 Example Program Results for nag_quad_util_ex01

 Legendre formula,  7 points

      Abscissae        Weights

       -0.94911        0.12948
       -0.74153        0.27971
       -0.40585        0.38183
```

```
0.00000         0.41796
0.40585         0.38183
0.74153         0.27971
0.94911         0.12948
```

Additional Examples

Not all example programs supplied with NAG *fl*90 appear in full in this module document. The following additional examples, associated with this module, are available.

`nag_quad_util_ex02`

Generates abscissae with both weights and adjusted weights using the Gauss–Jacobi 7-point formula.

`nag_quad_util_ex03`

Generates abscissae with both weights and adjusted weights using the Gauss–Exponential 7-point formula.

`nag_quad_util_ex04`

Generates abscissae with both weights and adjusted weights using the Gauss–Laguerre 7-point formula.

`nag_quad_util_ex05`

Generates abscissae with both weights and adjusted weights using the Gauss–Hermite 7-point formula.

`nag_quad_util_ex06`

Generates abscissae with both weights and adjusted weights using the Gauss–Rational 7-point formula.

References

[1] Davis P J and Rabinowitz P (1975) *Methods of Numerical Integration* Academic Press

[2] Stroud A H and Secrest D (1966) *Gaussian Quadrature Formulas* Prentice-Hall

[3] Golub G H and Welsch J H (1969) Calculation of Gauss quadrature rules *Math. Comput.* **23** 221–230