NAG Toolbox

nag_quad_1d_gauss_vec (d01ua)

1 Purpose

nag_quad_1d_gauss_vec (d01ua) computes an estimate of the definite integral of a function of known analytical form, using a Gaussian quadrature formula with a specified number of abscissae. Formulae are provided for a finite interval (Gauss–Legendre), a semi-infinite interval (Gauss–Laguerre, rational Gauss), and an infinite interval (Gauss–Hermite).

2 Syntax

```
[dinest, user, ifail] = nag_quad_1d_gauss_vec(key, a, b, n, f, 'user', user)
[dinest, user, ifail] = d01ua(key, a, b, n, f, 'user', user)
```

3 Description

3.1 General

nag_quad_1d_gauss_vec (d01ua) evaluates an estimate of the definite integral of a function $f(x)$, over a finite or infinite range, by n-point Gaussian quadrature (see Davis and Rabinowitz (1975), FrÎberg (1970), Ralston (1965) or Stroud and Secrest (1966)). The integral is approximated by a summation of the product of a set of weights and a set of function evaluations at a corresponding set of abscissae x_i. For adjusted weights, the function values correspond to the values of the integrand f, and hence the sum will be

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

where the w_i are called the weights, and the x_i the abscissae. A selection of values of n is available. (See Section 5.)

Where applicable, normal weights may instead be used, in which case the corresponding weight function ω is factored out of the integrand as $f(x) = \omega(x)g(x)$ and hence the sum will be

$$\sum_{i=-1}^{n} \bar{w}_i g(x),$$

where the normal weights $\bar{w}_i = w_i \omega(x_i)$ are computed internally.

nag_quad_1d_gauss_vec (d01ua) uses a vectorized **f** to evaluate the integrand or normalized integrand at a set of abscissae, x_i, for $i = 1, 2, \ldots, n_x$. If adjusted weights are used, the integrand $f(x_i)$ must be evaluated otherwise the normalized integrand $g(x_i)$ must be evaluated.

3.2 Both Limits Finite

$$\int_a^b f(x)\, dx.$$

The Gauss–Legendre weights and abscissae are used, and the formula is exact for any function of the form:

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

The formula is appropriate for functions which can be well approximated by such a polynomial over

$[a, b]$. It is inappropriate for functions with algebraic singularities at one or both ends of the interval, such as $(1 + x)^{-1/2}$ on $[-1, 1]$.

3.3 One Limit Infinite

$$\int_a^\infty f(x)\,dx \quad \text{or} \quad \int_{-\infty}^a f(x)\,dx.$$

Two quadrature formulae are available for these integrals.

(a) The Gauss–Laguerre formula is exact for any function of the form:

$$f(x) = e^{-bx}\sum_{i=0}^{2n-1} c_i x^i.$$

This formula is appropriate for functions decaying exponentially at infinity; the argument b should be chosen if possible to match the decay rate of the function.

If the adjusted weights are selected, the complete integrand $f(x)$ should be provided through **f**.

If the normal form is selected, the contribution of e^{-bx} is accounted for internally, and **f** should only return $g(x)$, where $f(x) = e^{-bx}g(x)$.

If $b < 0$ is supplied, the interval of integration will be $[a, \infty)$. Otherwise if $b > 0$ is supplied, the interval of integration will be $(-\infty, a]$.

(b) The rational Gauss formula is exact for any function of the form:

$$f(x) = \sum_{i=2}^{2n+1} \frac{c_i}{(x+b)^i} = \frac{\sum_{i=0}^{2n-1} c_{2n+1-i}(x+b)^i}{(x+b)^{2n+1}}.$$

This formula is likely to be more accurate for functions having only an inverse power rate of decay for large x. Here the choice of a suitable value of b may be more difficult; unfortunately a poor choice of b can make a large difference to the accuracy of the computed integral.

Only the adjusted form of the rational Gauss formula is available, and as such, the complete integrand $f(x)$ must be supplied in **f**.

If $a + b < 0$, the interval of integration will be $[a, \infty)$. Otherwise if $a + b > 0$, the interval of integration will be $(-\infty, a]$.

3.4 Both Limits Infinite

$$\int_{-\infty}^{+\infty} f(x)\,dx.$$

The Gauss–Hermite weights and abscissae are used, and the formula is exact for any function of the form:

$$f(x) = e^{-b(x-a)^2}\sum_{i=0}^{2n-1} c_i x^i,$$

where $b > 0$. Again, for general functions not of this exact form, the argument b should be chosen to match if possible the decay rate at $\pm\infty$.

If the adjusted weights are selected, the complete integrand $f(x)$ should be provided through **f**.

If the normal form is selected, the contribution of $e^{-b(x-a)^2}$ is accounted for internally, and **f** should only return $g(x)$, where $f(x) = e^{-b(x-a)^2} g(x)$.

4 References

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

FrÎberg C E (1970) *Introduction to Numerical Analysis* Addison–Wesley

Ralston A (1965) *A First Course in Numerical Analysis* pp. 87–90 McGraw–Hill

Stroud A H and Secrest D (1966) *Gaussian Quadrature Formulas* Prentice–Hall

5 Parameters

5.1 Compulsory Input Parameters

1: **key** – INTEGER

Indicates the quadrature formula.

key $= 0$
Gauss–Legendre quadrature on a finite interval, using normal weights.

key $= 3$
Gauss–Laguerre quadrature on a semi-infinite interval, using normal weights.

key $= -3$
Gauss–Laguerre quadrature on a semi-infinite interval, using adjusted weights.

key $= 4$
Gauss–Hermite quadrature on an infinite interval, using normal weights.

key $= -4$
Gauss–Hermite quadrature on an infinite interval, using adjusted weights.

key $= -5$
Rational Gauss quadrature on a semi-infinite interval, using adjusted weights.

Constraint: **key** $= 0$, 3, -3, 4, -4 or -5.

2: **a** – REAL (KIND=nag_wp)
3: **b** – REAL (KIND=nag_wp)

The quantities a and b as described in the appropriate subsection of Section 3.

Constraints:

Rational Gauss: **a** + **b** $\neq 0.0$;
Gauss–Laguerre: **b** $\neq 0.0$;
Gauss–Hermite: **b** > 0.

4: **n** – INTEGER

n, the number of abscissae to be used.

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

If the soft fail option is used, the answer is evaluated for the largest valid value of **n** less than the requested value.

5: **f** – SUBROUTINE, supplied by the user.

f must return the value of the integrand f, or the normalized integrand g, at a specified point.

```
[fv, iflag, user] = f(x, nx, iflag, user)
```

Input Parameters

1: **x**(**nx**) – REAL (KIND=nag_wp) array

The abscissae, x_i, for $i = 1, 2, \ldots, n_x$ at which function values are required.

2: **nx** – INTEGER

n_x, the number of abscissae.

3: **iflag** – INTEGER

iflag $= 0$.

4: **user** – INTEGER array

f is called from nag_quad_1d_gauss_vec (d01ua) with the object supplied to nag_quad_1d_gauss_vec (d01ua).

Output Parameters

1: **fv**(**nx**) – REAL (KIND=nag_wp) array

If adjusted weights are used, the values of the integrand f. **fv**$(i) = f(x_i)$, for $i = 1, 2, \ldots, n_x$.

Otherwise the values of the normalized integrand g. **fv**$(i) = g(x_i)$, for $i = 1, 2, \ldots, n_x$.

2: **iflag** – INTEGER

Set **iflag** < 0 if you wish to force an immediate exit from nag_quad_1d_gauss_vec (d01ua) with **ifail** $= -1$.

3: **user** – INTEGER array

Some points to bear in mind when coding **f** are mentioned in Section 7.

5.2 Optional Input Parameters

1: **user** – INTEGER array

user is not used by nag_quad_1d_gauss_vec (d01ua), but is passed to **f**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

5.3 Output Parameters

1: **dinest** – REAL (KIND=nag_wp)

The estimate of the definite integral.

2: **user** – INTEGER array

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Note: nag_quad_1d_gauss_vec (d01ua) may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the function:

ifail $= 1$ (*warning*)

The n-point rule is not among those stored.

ifail $= 2$ (*warning*)

Underflow occurred in calculation of normal weights.

ifail $= 3$ (*warning*)

No nonzero weights were generated for the provided parameters.

ifail $= 11$

Constraint: **key** $= 0$, 3, -3, 4, -4 or -5.

ifail $= 12$

The value of **a** and/or **b** is invalid for the chosen **key**. Either:

Constraint: $|\mathbf{a} + \mathbf{b}| > 0.0$.

Constraint: $|\mathbf{b}| > 0.0$.

Constraint: $\mathbf{b} > 0.0$.

ifail $= 14$

Constraint: $\mathbf{n} > 0$.

ifail $= -1$ (*warning*)

Exit requested from **f** with **iflag** $= \langle$*value*$\rangle$.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

The accuracy depends on the behaviour of the integrand, and on the number of abscissae used. No tests are carried out in nag_quad_1d_gauss_vec (d01ua) to estimate the accuracy of the result. If such an estimate is required, the function may be called more than once, with a different number of abscissae each time, and the answers compared. It is to be expected that for sufficiently smooth functions a larger number of abscissae will give improved accuracy.

Alternatively, the range of integration may be subdivided, the integral estimated separately for each sub-interval, and the sum of these estimates compared with the estimate over the whole range.

The coding of **f** may also have a bearing on the accuracy. For example, if a high-order Gauss–Laguerre formula is used, and the integrand is of the form

$$f(x) = e^{-bx} g(x)$$

it is possible that the exponential term may underflow for some large abscissae. Depending on the machine, this may produce an error, or simply be assumed to be zero. In any case, it would be better to evaluate the expression with

$$f(x) = \mathrm{sgn}(g(x)) \times \exp(-bx + \ln|g(x)|)$$

Another situation requiring care is exemplified by

$$\int_{-\infty}^{+\infty} e^{-x^2} x^m \, dx = 0, \quad m \text{ odd}.$$

The integrand here assumes very large values; for example, when $m = 63$, the peak value exceeds 3×10^{33}. Now, if the machine holds floating-point numbers to an accuracy of k significant decimal digits, we could not expect such terms to cancel in the summation leaving an answer of much less than 10^{33-k} (the weights being of order unity); that is, instead of zero we obtain a rather large answer through rounding error. Such situations are characterised by great variability in the answers returned by formulae with different values of n.

In general, you should be aware of the order of magnitude of the integrand, and should judge the answer in that light.

8 Further Comments

The time taken by nag_quad_1d_gauss_vec (d01ua) depends on the complexity of the expression for the integrand and on the number of abscissae required.

9 Example

This example evaluates the integrals

$$\int_0^1 \frac{4}{1+x^2} dx = \pi$$

by Gauss–Legendre quadrature;

$$\int_2^{\infty} \frac{1}{x^2 \ln x} dx = 0.378671$$

by rational Gauss quadrature with $b = 0$;

$$\int_2^{\infty} \frac{e^{-x}}{x} dx = 0.048901$$

by Gauss–Laguerre quadrature with $b = 1$; and

$$\int_{-\infty}^{+\infty} e^{-3x^2-4x-1} \, dx = \int_{-\infty}^{+\infty} e^{-3(x+1)^2} e^{2x+2} dx = 1.428167$$

by Gauss–Hermite quadrature with $a = -1$ and $b = 3$.

The formulae with $n = 2, 4, 8, 16, 32$ and 64 are used in each case. Both adjusted and normal weights are used for Gauss–Laguerre and Gauss–Hermite quadrature.

9.1 Program Text

```
     function d01ua_example

fprintf('d01ua example results\n\n');

  for funid=1:6
    switch funid
      case 1
        fprintf('\nGauss-Legendre example\n');
        a = 0.0;
        b = 1.0;
        key = nag_int(0);
      case 2
        fprintf('\nRational Gauss example\n');
```

```
        a = 2.0;
        b = 0.0;
        key = nag_int(-5);
      case 3
        fprintf('\nGauss-Laguerre example (adjusted weights)\n');
        a = 2.0;
        b = 1.0;
        key = nag_int(-3);
      case 4
        fprintf('\nGauss-Laguerre example (normal weights)\n');
        a = 2.0;
        b = 1.0;
        key = nag_int(3);
      case 5
        fprintf('\nGauss-Hermite example (adjusted weights)\n');
        a = -1.0;
        b = 3.0;
        key = nag_int(-4);
      case 6
        fprintf('\nGauss-Hermite example (normal weights)\n');
        a = -1.0;
        b = 3.0;
        key = nag_int(4);
    end

    for i=1:6
      n = nag_int(2^i);
      [dinest, user, ifail] = d01ua(key, a, b, n, @f, 'user', funid);

      if ifail == 0 || ifail == 1
        fprintf('%2d Points   Answer = %10.5f\n', n, dinest);
      end
    end
  end

function [fv, iflag, user] = f(x, nx, iflag, user)
  switch user
    case 1
      fv = 4./(1+x.*x);
    case 2
      fv = 1./(x.*x.*log(x));
    case 3
      fv = exp(-x)./x;
    case 4
      fv = 1./x;
    case 5
      fv = exp(-3.*x.*x-4.*x-1);
    case 6
      fv = exp(2*x+2);
    otherwise
      fv = zeros(nx, 1);
      iflag = -1;
  end
```

9.2 Program Results

```
     d01ua example results

Gauss-Legendre example
 2 Points   Answer =    3.14754
 4 Points   Answer =    3.14161
 8 Points   Answer =    3.14159
16 Points   Answer =    3.14159
32 Points   Answer =    3.14159
64 Points   Answer =    3.14159

Rational Gauss example
 2 Points   Answer =    0.37989
 4 Points   Answer =    0.37910
 8 Points   Answer =    0.37876
```

```
16 Points   Answer =    0.37869
32 Points   Answer =    0.37867
64 Points   Answer =    0.37867

Gauss-Laguerre example (adjusted weights)
 2 Points   Answer =    0.04833
 4 Points   Answer =    0.04887
 8 Points   Answer =    0.04890
16 Points   Answer =    0.04890
32 Points   Answer =    0.04890
64 Points   Answer =    0.04890

Gauss-Laguerre example (normal weights)
 2 Points   Answer =    0.04833
 4 Points   Answer =    0.04887
 8 Points   Answer =    0.04890
16 Points   Answer =    0.04890
32 Points   Answer =    0.04890
64 Points   Answer =    0.04890

Gauss-Hermite example (adjusted weights)
 2 Points   Answer =    1.38381
 4 Points   Answer =    1.42803
 8 Points   Answer =    1.42817
16 Points   Answer =    1.42817
32 Points   Answer =    1.42817
64 Points   Answer =    1.42817

Gauss-Hermite example (normal weights)
 2 Points   Answer =    1.38381
 4 Points   Answer =    1.42803
 8 Points   Answer =    1.42817
16 Points   Answer =    1.42817
32 Points   Answer =    1.42817
64 Points   Answer =    1.42817
```