Module 3.6: nag_ell_intg

Elliptic Integrals

`nag_ell_intg` contains procedures for approximating symmetrised variants of the classic elliptic integrals.

Contents

Introduction

This module contains procedures for approximating symmetrised variants (Carlson [2], Carlson [3], Carlson [4]) of the classic elliptic integrals.

- `nag_ell_rf` computes the standard integral of the first kind

  $$R_F(x,y,z) = \frac{1}{2}\int_0^\infty \frac{dt}{\sqrt{(t+x)(t+y)(t+z)}},$$

  where $x, y, z \geq 0$ and at most one may be equal to zero.

- `nag_ell_rc` computes the following degenerate form of R_F

  $$R_C(x,y) = \frac{1}{2}\int_0^\infty \frac{dt}{\sqrt{t+x}\,(t+y)}, \quad x \geq 0,\ y \neq 0.$$

- `nag_ell_rd` computes the standard integral of the second kind

  $$R_D(x,y,z) = \frac{3}{2}\int_0^\infty \frac{dt}{\sqrt{(t+x)(t+y)(t+z)^3}},$$

  where $z > 0$, $x \geq 0$ and $y \geq 0$ but only one of x or y may be zero.

- `nag_ell_rj` computes the standard integral of the third kind

  $$R_J(x,y,z,\rho) = \frac{3}{2}\int_0^\infty \frac{dt}{(t+\rho)\sqrt{(t+x)(t+y)(t+z)}},$$

  with $\rho \neq 0$, $x, y, z \geq 0$ with at most one equality holding.

For further details of elliptic integrals see Abramowitz and Stegun [1], Chapter 17.

Procedure: nag_ell_rf

1 Description

`nag_ell_rf` evaluates an approximation to the symmetrised elliptic integral of the first kind

$$R_F(x,y,z) = \frac{1}{2}\int_0^\infty \frac{dt}{\sqrt{(t+x)(t+y)(t+z)}},$$

where x, y, $z \geq 0$ and at most one is zero. The normalisation factor, $\frac{1}{2}$, is chosen so as to make $R_F(x,x,x) = 1/\sqrt{x}$.

2 Usage

```
USE nag_ell_intg
[value =] nag_ell_rf(x, y, z  [, optional arguments])
```

The function result is a scalar, of type real(kind=*wp*), containing $R_F(x,y,z)$.

3 Arguments

3.1 Mandatory Argument

x — real(kind=*wp*), intent(in)
y — real(kind=*wp*), intent(in)
z — real(kind=*wp*), intent(in)

Input: the arguments x, y, and z of the function, respectively.

Constraints: `x`, `y`, `z` $\geq$ 0.0 and only one of `x`, `y` and `z` may be zero.

3.2 Optional Argument

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.

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

If $x = y$, $x = z$ or $y = z$, the function reduces to the elementary integral R_C, computed by `nag_ell_rc`.

6.2 Algorithmic Detail

The basic algorithm, which is due to Carlson [5] and Carlson [6], is to reduce the arguments recursively towards their mean by the rule:

$x_0 = \min(x, y, z)$, $z_0 = \max(x, y, z)$,
$y_0 =$ remaining third intermediate value argument.

(This ordering, which is possible because of the symmetry of the function, is done for technical reasons related to the avoidance of overflow and underflow.)

$$\begin{aligned}
\mu_n &= (x_n + y_n + z_n)/3 \\
X_n &= 1 - x_n/\mu_n \\
Y_n &= 1 - y_n/\mu_n \\
Z_n &= 1 - z_n/\mu_n \\
\lambda_n &= \sqrt{x_n y_n} + \sqrt{y_n z_n} + \sqrt{z_n x_n} \\
x_{n+1} &= (x_n + \lambda_n)/4 \\
y_{n+1} &= (y_n + \lambda_n)/4 \\
z_{n+1} &= (z_n + \lambda_n)/4
\end{aligned}$$

$\varepsilon_n = \max(|X_n|, |Y_n|, |Z_n|)$ and the function may be approximated adequately by a fifth-order power series:

$$R_F(x, y, z) = \frac{1}{\sqrt{\mu_n}} \left(1 - \frac{E_2}{10} + \frac{E_2^2}{24} - \frac{3E_2E_3}{44} + \frac{E_3}{14} \right)$$

where $E_2 = X_nY_n + Y_nZ_n + Z_nX_n, E_3 = X_nY_nZ_n$.

The truncation error involved in using this approximation is bounded by $\varepsilon_n^6/4(1 - \varepsilon_n)$. The recursive process is stopped when this truncation error is negligible compared with `EPSILON(1.0_`*wp*`)`.

Within the domain of definition, the function value is itself representable for all representable values of its arguments. However, for values of the arguments near the extremes the above algorithm must be modified so as to avoid causing underflows or overflows in intermediate steps. In extreme regions, arguments are pre-scaled away from the extremes and compensating scaling of the result is done before returning to the calling program.

6.3 Accuracy

In principle the procedure is capable of producing accuracy of the order of `EPSILON(1.0_`*wp*`)`. However, round-off errors in internal arithmetic will result in slight loss of accuracy. This loss should never be excessive as the algorithm does not involve any significant amplification of round-off error. It is reasonable to assume that the result is accurate to within a small multiple of `EPSILON(1.0_`*wp*`)`.

Procedure: nag_ell_rc

1 Description

nag_ell_rc calculates an approximate value for the elementary integral

$$R_C(x,y) = \frac{1}{2}\int_0^\infty \frac{dt}{\sqrt{t+x}\,(t+y)},$$

where $x \geq 0$ and $y \neq 0$. This integral occurs as a degenerate case of the elliptic integral $R_F(x,y,z)$ of the first kind.

2 Usage

```
USE nag_ell_intg
[value =] nag_ell_rc(x, y  [, optional arguments])
```

The function result is a scalar, of type real(kind=*wp*), containing $R_C(x,y)$.

3 Arguments

3.1 Mandatory Argument

x — real(kind=*wp*), intent(in)

y — real(kind=*wp*), intent(in)

Input: the arguments x and y of the function, respectively.

Constraints: x $\geq$ 0.0 and y $\neq$ 0.0.

3.2 Optional Argument

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.

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 integral, which is related to the logarithm or inverse hyperbolic functions for $y < x$ and to inverse circular functions if $x < y$, arises as a degenerate form of the elliptic integral R_F of the first kind.

If $y < 0$, the result computed is the Cauchy principal value of the integral.

6.2 Algorithmic Detail

The basic algorithm, which is due to Carlson [5] and Carlson [6], is to reduce the arguments recursively towards their mean by the system:

$$\begin{array}{rclrcl} x_0 &=& x, & y_0 &=& y \\ \mu_n &=& (x_n + 2y_n)/3, & S_n &=& (y_n - x_n)/3\mu_n \\ \lambda_n &=& y_n + 2\sqrt{x_n y_n} & & & \\ x_{n+1} &=& (x_n + \lambda_n)/4, & y_{n+1} &=& (y_n + \lambda_n)/4. \end{array}$$

The quantity $|S_n|$, for $n = 0, 1, 2, 3, \ldots$, decreases with increasing n; eventually $|S_n| \sim 1/4^n$. For small enough S_n the required function value can be approximated by the first few terms of the Taylor series about the mean. That is

$$R_C(x,y) = \left(1 + \frac{3S_n^2}{10} + \frac{S_n^3}{7} + \frac{3S_n^4}{8} + \frac{9S_n^5}{22}\right) \bigg/ \sqrt{\mu_n}.$$

The truncation error involved in using this approximation is bounded by $16|S_n|^6/(1 - 2|S_n|)$ and the recursive process is stopped when S_n is small enough for this truncation error to be negligible compared to `EPSILON(1.0_`*wp*`)`.

Within the domain of definition, the function value is itself representable for all representable values of its arguments. However, for values of the arguments near the extremes the above algorithm must be modified so as to avoid causing underflows or overflows in intermediate steps. In extreme regions, arguments are pre-scaled away from the extremes and compensating scaling of the result is done before returning to the calling program.

6.3 Accuracy

In principle the procedure is capable of producing accuracy of the order of `EPSILON(1.0_`*wp*`)`. However, round-off errors in internal arithmetic will result in slight loss of accuracy. This loss should never be excessive as the algorithm does not involve any significant amplification of round-off error. It is reasonable to assume that the result is accurate to within a small multiple of `EPSILON(1.0_`*wp*`)`.

Procedure: nag_ell_rd

1 Description

`nag_ell_rd` evaluates an approximation to the symmetrised elliptic integral of the second kind

$$R_D(x,y,z) = \frac{3}{2}\int_0^\infty \frac{dt}{\sqrt{(t+x)(t+y)(t+z)^3}},$$

where x, $y \geq 0$, at most one of x and y is zero, and $z > 0$. The normalisation factor, $\frac{3}{2}$, is chosen so as to make $R_D(x,x,x) = 1/(x\sqrt{x})$.

2 Usage

```
USE nag_ell_intg
[value =] nag_ell_rd(x, y, z  [, optional arguments])
```

The function result is a scalar, of type real(kind=*wp*), containing $R_D(x,y,z)$.

3 Arguments

3.1 Mandatory Argument

x — real(kind=*wp*), intent(in)
y — real(kind=*wp*), intent(in)
z — real(kind=*wp*), intent(in)

Input: the arguments x, y, and z of the function, respectively.

Constraints: `x`, `y` $\geq$ 0.0, with at most one of `x` and `y` zero, and `z` $>$ 0.0.

3.2 Optional Argument

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.

Failures (error%level = 2):

error%code	Description
201	Possibility of overflow. Either `z` is too close to zero or both `x` and `y` are too close to zero. There is a danger of overflow.
202	Possibility of underflow. At least one of `x`, `y` or `z` is too large. There is a danger of underflow. Zero is returned.

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 Algorithmic Detail

The basic algorithm, which is due to Carlson [5] and Carlson [6], is to reduce the arguments recursively towards their mean by the rule:

$$\begin{aligned}
x_0 &= x, \; y_0 = y, \; z_0 = z \\
\mu_n &= (x_n + y_n + 3z_n)/5 \\
X_n &= 1 - x_n/\mu_n \\
Y_n &= 1 - y_n/\mu_n \\
Z_n &= 1 - z_n/\mu_n \\
\lambda_n &= \sqrt{x_n y_n} + \sqrt{y_n z_n} + \sqrt{z_n x_n} \\
x_{n+1} &= (x_n + \lambda_n)/4 \\
y_{n+1} &= (y_n + \lambda_n)/4 \\
z_{n+1} &= (z_n + \lambda_n)/4.
\end{aligned}$$

For n sufficiently large,

$$\varepsilon_n = \max(|X_n|, |Y_n|, |Z_n|) \sim 1/4^n$$

and the function may be approximated adequately by a fifth order power series

$$\begin{aligned}
R_D(x,y,z) &= 3\sum_{m=0}^{n-1} \frac{4^{-m}}{(z_m + \lambda_n)\sqrt{z_m}} \\
&\quad + \frac{4^{-n}}{\sqrt{\mu_n^3}} \left(1 + \frac{3}{7}S_n^{(2)} + \frac{1}{3}S_n^{(3)} + \frac{3}{22}(S_n^{(2)})^2 + \frac{3}{11}S_n^{(4)} + \frac{3}{13}S_n^{(2)}S_n^{(3)} + \frac{3}{13}S_n^{(5)}\right)
\end{aligned}$$

where

$$S_n^{(m)} = \left(X_n^m + Y_n^m + 3Z_n^m\right)/2m.$$

The truncation error in this expansion is bounded by $3\varepsilon_n^6/\sqrt{(1-\varepsilon_n)^3}$ and the recursive process is terminated when this quantity is negligible compared with EPSILON(1.0_*wp*).

The procedure may fail either because it has been called with arguments outside the domain of definition, or with arguments so extreme that there is an unavoidable danger of setting overflow.

If at least one of x, y or z is too large, such that there is an unavoidable danger of setting underflow, this procedure returns zero.

Note. $R_D(x,x,x) = x^{-3/2}$, so there exists a region of extreme arguments for which the function value is not representable.

6.2 Accuracy

In principle the procedure is capable of producing accuracy of the order of EPSILON(1.0_*wp*). However, round-off errors in internal arithmetic will result in slight loss of accuracy. This loss should never be excessive as the algorithm does not involve any significant amplification of round-off error. It is reasonable to assume that the result is accurate to within a small multiple of EPSILON(1.0_*wp*).

Procedure: nag_ell_rj

1 Description

nag_ell_rj evaluates an approximation to the symmetrised elliptic integral of the third kind

$$R_J(x,y,z,\rho) = \frac{3}{2}\int_0^\infty \frac{dt}{(t+\rho)\sqrt{(t+x)(t+y)(t+z)}},$$

where x, y, $z \geq 0$, $\rho \neq 0$ and at most one of x, y and z is zero. The normalisation factor, $\frac{3}{2}$, is chosen so as to make $R_J(x,x,x,x) = 1/(x\sqrt{x})$.

2 Usage

```
USE nag_ell_intg
[value =] nag_ell_rj(x, y, z, r  [, optional arguments])
```

The function result is a scalar, of type real(kind=*wp*), containing $R_J(x,y,z,\rho)$.

3 Arguments

3.1 Mandatory Argument

x — real(kind=*wp*), intent(in)
y — real(kind=*wp*), intent(in)
z — real(kind=*wp*), intent(in)
r — real(kind=*wp*), intent(in)

Input: the arguments x, y, z and ρ of the function, respectively.

Constraints: x, y, z $\geq$ 0.0 and r $\neq$ 0.0 and at most one of x, y and z may be zero.

3.2 Optional Argument

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.
399	An unexpected Library error. Please report this error to NAG.

Failures (error%level = 2):

error%code	Description
201	Possibility of overflow.
	Either `r` is too close to zero, or any two of `x`, `y` and `z` are too close to zero. There is a danger of overflow.
202	Possibility of underflow.
	At least one of `x`, `y`, `z` or `r` is too large. There is a danger of underflow. Zero is returned.

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

If $\rho = x, y$ or z, is equal to any of the other arguments, the function reduces to the integral R_D, computed by `nag_ell_rd`.

If $\rho < 0$, the result computed is the Cauchy principal value of the integral.

6.2 Algorithmic Detail

The basic algorithm, which is due to Carlson [5] and Carlson [6], is to reduce the arguments recursively towards their mean by the rule:

$$\begin{array}{rcl}
x_0 & = & x,\ \ y_0 \ = \ y,\ \ z_0 \ = \ z,\ \ \rho_0 \ = \ \rho \\
\mu_n & = & (x_n + y_n + z_n + 2\rho_n)/5 \\
X_n & = & 1 - x_n/\mu_n \\
Y_n & = & 1 - y_n/\mu_n \\
Z_n & = & 1 - z_n/\mu_n \\
P_n & = & 1 - \rho_n/\mu_n \\
\lambda_n & = & \sqrt{x_n y_n} + \sqrt{y_n z_n} + \sqrt{z_n x_n} \\
x_{n+1} & = & (x_n + \lambda_n)/4 \\
y_{n+1} & = & (y_n + \lambda_n)/4 \\
z_{n+1} & = & (z_n + \lambda_n)/4 \\
\rho_{n+1} & = & (\rho_n + \lambda_n)/4 \\
\alpha_n & = & [\rho_n(\sqrt{x_n} + \sqrt{y_n} + \sqrt{z_n}) + \sqrt{x_n y_n z_n}]^2 \\
\beta_n & = & \rho_n(\rho_n + \lambda_n)^2.
\end{array}$$

For n sufficiently large,

$$\varepsilon_n = \max(|X_n|, |Y_n|, |Z_n|, |P_n|) \sim 1/4^n$$

and the function may be approximated by a fifth-order power series

$$\begin{array}{rcl}
R_J(x,y,z,\rho) & = & 3\displaystyle\sum_{m=0}^{n-1} 4^{-m} R_C(\alpha_m, \beta_m) \\
& & + \dfrac{4^{-n}}{\sqrt{\mu_n^3}}\left(1 + \dfrac{3}{7}S_n^{(2)} + \dfrac{1}{3}S_n^{(3)} + \dfrac{3}{22}(S_n^{(2)})^2 + \dfrac{3}{11}S_n^{(4)} + \dfrac{3}{13}S_n^{(2)}S_n^{(3)} + \dfrac{3}{13}S_n^{(5)}\right),
\end{array}$$

where $S_n^{(m)} = (X_n^m + Y_n^m + Z_n^m + 2P_n^m)/2m$.

The truncation error in this expansion is bounded by $3\varepsilon_n^6/\sqrt{(1-\varepsilon_n)^3}$ and the recursion process is terminated when this quantity is negligible compared with EPSILON(1.0_*wp*). The procedure may fail either because it has been called with arguments outside the domain of definition or with arguments so extreme that there is an unavoidable danger of setting overflow.

If at least one of x, y, z or ρ is too large, such that there is an unavoidable danger of setting underflow, nag_ell_rd returns zero.

Note. $R_J(x,x,x,x) = x^{-3/2}$, so there exists a region of extreme arguments for which the function value is not representable.

6.3 Accuracy

In principle the procedure is capable of producing accuracy of the order of EPSILON(1.0_*wp*). However, round-off errors in internal arithmetic will result in slight loss of accuracy. This loss should never be excessive as the algorithm does not involve any significant amplification of round-off error. It is reasonable to assume that the result is accurate to within a small multiple of EPSILON(1.0_*wp*).

Example 1: Evaluation of elliptic integrals

This example program simply generates a small set of non-extreme arguments which are used with the four procedures nag_ell_rc, nag_ell_rf, nag_ell_rd and nag_ell_rj to produce the table of low accuracy results.

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_ell_intg_ex01

  ! Example Program Text for nag_ell_intg
  ! NAG fl90, Release 3. NAG Copyright 1997.

  ! .. Use Statements ..
  USE nag_examples_io, ONLY : nag_std_out
  USE nag_ell_intg, ONLY : nag_ell_rc, nag_ell_rf, nag_ell_rd, nag_ell_rj
  ! .. Implicit None Statement ..
  IMPLICIT NONE
  ! .. Intrinsic Functions ..
  INTRINSIC KIND
  ! .. Parameters ..
  INTEGER, PARAMETER :: wp = KIND(1.0D0)
  ! .. Local Scalars ..
  INTEGER :: ix, iy, iz
  REAL (wp) :: r, rc, rd, rf, rj, x, y, z
  ! .. Executable Statements ..

  WRITE (nag_std_out,*) 'Example Program Results for nag_ell_intg_ex01'

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,*) '     x      y         rc'
  y = 1.0_wp
  DO ix = 1, 3
    x = ix*0.5_wp

    rc = nag_ell_rc(x,y)

    WRITE (nag_std_out,fmt='(1X,2F7.2,F12.4)') x, y, rc
  END DO

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,*) '     x      y      z         rf'
  DO ix = 1, 3
    x = ix*0.5_wp
    y = x + 0.5_wp
    z = y + 0.5_wp

    rf = nag_ell_rf(x,y,z)

    WRITE (nag_std_out,fmt='(1X,3F7.2,F12.4)') x, y, z, rf
  END DO

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,*) '     x      y      z         rd'
  z = 1.0_wp
  DO ix = 1, 3
    x = ix*0.5_wp
    DO iy = ix, 3
      y = iy*0.5_wp
```

```
        rd = nag_ell_rd(x,y,z)

        WRITE (nag_std_out,fmt='(1X,3F7.2,F12.4)') x, y, z, rd
      END DO
    END DO

    WRITE (nag_std_out,*)
    WRITE (nag_std_out,*) '     x      y      z      r         rj'
    r = 2.0_wp
    DO ix = 1, 3
      x = ix*0.5_wp
      DO iy = ix, 3
        y = iy*0.5_wp
        DO iz = iy, 3
          z = iz*0.5_wp

          rj = nag_ell_rj(x,y,z,r)

          WRITE (nag_std_out,fmt='(1X,4F7.2,F12.4)') x, y, z, r, rj
        END DO
      END DO
    END DO

  END PROGRAM nag_ell_intg_ex01
```

2 Program Data

None.

3 Program Results

```
 Example Program Results for nag_ell_intg_ex01

     x      y         rc
   0.50   1.00      1.1107
   1.00   1.00      1.0000
   1.50   1.00      0.9312

     x      y      z         rf
   0.50   1.00   1.50      1.0281
   1.00   1.50   2.00      0.8260
   1.50   2.00   2.50      0.7116

     x      y      z         rd
   0.50   0.50   1.00      1.4787
   0.50   1.00   1.00      1.2108
   0.50   1.50   1.00      1.0611
   1.00   1.00   1.00      1.0000
   1.00   1.50   1.00      0.8805
   1.50   1.50   1.00      0.7775

     x      y      z      r         rj
   0.50   0.50   0.50   2.00      1.1184
   0.50   0.50   1.00   2.00      0.9221
   0.50   0.50   1.50   2.00      0.8115
   0.50   1.00   1.00   2.00      0.7671
   0.50   1.00   1.50   2.00      0.6784
   0.50   1.50   1.50   2.00      0.6017
   1.00   1.00   1.00   2.00      0.6438
   1.00   1.00   1.50   2.00      0.5722
   1.00   1.50   1.50   2.00       0.5101
   1.50   1.50   1.50   2.00       0.4561
```

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_ell_intg_ex02`

Evaluation of the elliptic integral R_F of the first kind.

`nag_ell_intg_ex03`

Evaluation of the elliptic integral R_D of the second kind.

`nag_ell_intg_ex04`

Evaluation of the elliptic integral R_J of the third kind.

`nag_ell_intg_ex05`

Evaluation of the degenerate form R_C of the elliptic integral of the first kind.

References

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

[2] Carlson B C (1965) On computing elliptic integrals and functions *J. Math. Phys.* **44** 36–51

[3] Carlson B C (1977) Elliptic integrals of the first kind *SIAM J. Math. Anal.* **8** 231–242

[4] Carlson B C (1977) *Special Functions of Applied Mathematics* Academic Press

[5] Carlson B C (1978) Computing elliptic integrals by duplication *Preprint* Department of Physics, Iowa State University

[6] Carlson B C (1988) A table of elliptic integrals of the third kind *Math. Comput.* **51** 267–280