Module 3.9: nag_kelvin_fun
Kelvin Functions

`nag_kelvin_fun` contains procedures for approximating Kelvin functions.

Contents

Introduction

This module contains procedures for approximating Kelvin functions.

- `nag_kelvin_ber` approximates the Kelvin function ber x

  $$\mathrm{ber}_\nu\ x = \mathrm{Re}(J_\nu(xe^{3\pi i/4}))$$

- `nag_kelvin_bei` approximates the Kelvin function bei x

  $$\mathrm{bei}_\nu\ x = \mathrm{Im}(J_\nu(xe^{3\pi i/4}))$$

- `nag_kelvin_ker` approximates the Kelvin function ker x

  $$\mathrm{ker}_\nu\ x = \mathrm{Re}(e^{-\nu\pi i/2}K_\nu(xe^{\pi i/4}))$$

- `nag_kelvin_kei` approximates the Kelvin function kei x

  $$\mathrm{kei}_\nu\ x = \mathrm{Im}(e^{-\nu\pi i/2}K_\nu(xe^{\pi i/4}))$$

Only procedures for $\nu = 0$ are provided.

For further details of Kelvin functions, see Abramowitz and Stegun [1], Chapter 9.

The approximations are based on expansions in terms of Chebyshev polynomials. Further details appear in Section 6.1 of the individual procedure documents.

Procedure: nag_kelvin_ber

1 Description

nag_kelvin_ber evaluates an approximation to the Kelvin function ber x.

2 Usage

```
USE nag_kelvin_fun
[value =] nag_kelvin_ber(x  [, optional arguments])
```

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

3 Arguments

3.1 Mandatory Argument

x — real(kind=*wp*), intent(in)

Input: the argument x of the function.

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

Failures (error%level = 2):

error%code	Description
201	Total loss of accuracy. |x| is too large. No computation has been performed as the error amplification factor grows exponentially such that all the precision in the result returned would be lost.

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

Since ber $(-x)$ = ber x, we need only consider the case $x \geq 0$.

- For $0 \leq x \leq 5$, the procedure uses a Chebyshev expansion of the form

$$\text{ber } x = {\sum_{r=0}}' a_r T_r(t), \quad \text{with} \quad t = 2\left(\frac{x}{5}\right)^4 - 1.$$

- For $x > 5$, it uses

$$\begin{aligned}\text{ber } x = \quad & \frac{e^{x/\sqrt{2}}}{\sqrt{2\pi x}}\left[\left(1+\frac{1}{x}a(t)\right)\cos\alpha + \frac{1}{x}b(t)\sin\alpha\right] \\ & + \frac{e^{-x/\sqrt{2}}}{\sqrt{2\pi x}}\left[\left(1+\frac{1}{x}c(t)\right)\sin\beta + \frac{1}{x}d(t)\cos\beta\right]\end{aligned}$$

  where $\alpha = \frac{x}{\sqrt{2}} - \frac{\pi}{8}$, $\beta = \frac{x}{\sqrt{2}} + \frac{\pi}{8}$,

  and $a(t)$, $b(t)$, $c(t)$, and $d(t)$ are expansions in the variable $t = \frac{10}{x} - 1$.

- For x near zero, the result is set directly to ber $(0) = 1.0$.

- For large x, there is a danger of the result being totally inaccurate, as the error amplification factor grows in an essentially exponential manner; therefore the procedure must fail.

6.2 Accuracy

Since the function is oscillatory, the absolute error rather than the relative error is important. Let E be the absolute error in the result and δ be the relative error in the argument. If δ is somewhat larger than `EPSILON(1.0_wp)`, then we have:

$$E \simeq \left| x/\sqrt{2}(\text{ber}_1\, x + \text{bei}_1\, x)\right| \delta$$

(provided E is within machine bounds).

For small x the error amplification is insignificant and thus the absolute error is effectively bounded by `EPSILON(1.0_wp)`.

For medium and large x, the error behaviour is oscillatory and its amplitude grows like $\sqrt{\frac{x}{2\pi}}e^{x/\sqrt{2}}$. Therefore it is not possible to calculate the function with any accuracy when $\sqrt{x}e^{x/\sqrt{2}} > \frac{\sqrt{2\pi}}{\delta}$. Note that this value of x is much smaller than the minimum value of x for which the function overflows.

Procedure: nag_kelvin_bei

1 Description

nag_kelvin_bei evaluates an approximation to the Kelvin function bei x.

2 Usage

```
USE nag_kelvin_fun
[value =] nag_kelvin_bei(x  [, optional arguments])
```

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

3 Arguments

3.1 Mandatory Argument

x — real(kind=*wp*), intent(in)

Input: the argument x of the function.

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

Failures (error%level = 2):

error%code	Description
201	Total loss of accuracy. |x| is too large. No computation has been performed as the error amplification factor grows exponentially such that all the precision in the result returned would be lost.

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

Since bei $(-x)$ = bei x, we need only consider the case $x \geq 0$.

- For $0 \leq x \leq 5$, the procedure uses a Chebyshev expansion of the form

$$\text{bei } x = \frac{x^2}{4}{\sum_{r=0}}' a_r T_r(t), \quad \text{with} \quad t = 2\left(\frac{x}{5}\right)^4 - 1.$$

- For $x > 5$, it uses

$$\begin{aligned}\text{bei } x = \quad & \frac{e^{x/\sqrt{2}}}{\sqrt{2\pi x}}\left[\left(1+\frac{1}{x}a(t)\right)\sin\alpha - \frac{1}{x}b(t)\cos\alpha\right] \\ & + \frac{e^{x/\sqrt{2}}}{\sqrt{2\pi x}}\left[\left(1+\frac{1}{x}c(t)\right)\cos\beta - \frac{1}{x}d(t)\sin\beta\right]\end{aligned}$$

  where $\alpha = \frac{x}{\sqrt{2}} - \frac{\pi}{8}$, $\beta = \frac{x}{\sqrt{2}} + \frac{\pi}{8}$,

  and $a(t)$, $b(t)$, $c(t)$, and $d(t)$ are expansions in the variable $t = \frac{10}{x} - 1$.

- For x near zero, the result is computed as $\text{bei}\, x = \frac{x^2}{4}$. If this result would underflow, the result returned is $\text{bei}\, x = 0.0$.

- For large x, there is a danger of the result being totally inaccurate, as the error amplification factor grows in an essentially exponential manner; therefore the procedure must fail.

6.2 Accuracy

Since the function is oscillatory, the absolute error rather than the relative error is important. Let E be the absolute error in the function, and δ be the relative error in the argument. If δ is somewhat larger than `EPSILON(1.0_wp)`, then we have:

$$E \simeq \left| x/\sqrt{2}(-\,\text{ber}_1\, x + \text{bei}_1\, x)\right| \delta$$

(provided E is within machine bounds).

For small x the error amplification is insignificant and thus the absolute error is effectively bounded by `EPSILON(1.0_wp)`.

For medium and large x, the error behaviour is oscillatory and its amplitude grows like $\sqrt{\frac{x}{2\pi}}e^{x/\sqrt{2}}$. Therefore it is impossible to calculate the functions with any accuracy when $\sqrt{x}e^{x/\sqrt{2}} > \frac{\sqrt{2\pi}}{\delta}$. Note that this value of x is much smaller than the minimum value of x for which the function overflows.

Procedure: nag_kelvin_ker

1 Description

nag_kelvin_ker evaluates an approximation to the Kelvin function ker x.

2 Usage

```
USE nag_kelvin_fun
[value =] nag_kelvin_ker(x  [, optional arguments])
```

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

3 Arguments

3.1 Mandatory Argument

x — real(kind=*wp*), intent(in)

Input: the argument x of the function.

Constraints: x > 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 underflow. Argument x is too large. No computation has been performed as the value of the function becomes so small that it cannot be computed without 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

- For $x < 0$, the function ker x is undefined and the procedure will fail for such arguments.
- At $x = 0$, the function ker 0.0 is infinite and the procedure will fail for this argument.
- For $0 < x \leq 1$,

  $$\ker x = -f(t)\log x + \frac{\pi}{16}x^2 g(t) + y(t)$$

  where $f(t)$, $g(t)$ and $y(t)$ are expansions in the variable $t = 2x^4 - 1$.
- For $1 < x \leq 3$,

  $$\ker x = \exp\left(-\frac{11}{16}x\right) q(t)$$

  where $q(t)$ is an expansion in the variable $t = x - 2$.
- For $x > 3$,

  $$\ker x = \sqrt{\frac{\pi}{2x}} e^{-x/\sqrt{2}} \left[\left(1 + \frac{1}{x}c(t)\right)\cos\beta - \frac{1}{x}d(t)\sin\beta\right]$$

  where $\beta = \frac{x}{\sqrt{2}} + \frac{\pi}{8}$, and $c(t)$ and $d(t)$ are expansions in the variable $t = \frac{6}{x} - 1$.
- For x near zero, the result is computed as

  $$\ker x = -\gamma - \log\left(\frac{x}{2}\right) + \left(\pi - \frac{3}{8}x^2\right)\frac{x^2}{16}$$

  where γ is the Euler contant.
- When x is even closer to zero, it uses

  $$\ker x = -\gamma - \log\left(\frac{x}{2}\right)$$

  where γ is the Euler contant.
- For large x, ker x is asymptotically given by $\sqrt{\frac{\pi}{2x}} e^{-x/\sqrt{2}}$ and this becomes so small that it cannot be computed without underflow, and the procedure returns zero.

6.2 Accuracy

Let E be the absolute error in the result, ϵ be the relative error in the result and δ be the relative error in the argument. If δ is somewhat larger than `EPSILON(1.0_wp)`, then we have:

$$E \simeq \left| x/\sqrt{2}(\ker_1 x + \mathrm{kei}_1 x)\right| \delta,$$

$$\epsilon \simeq \left| \frac{x}{\sqrt{2}} \frac{\ker_1 x + \mathrm{kei}_1 x}{\ker x}\right| \delta.$$

For very small x, the relative error amplification factor is approximately given by $\frac{1}{|\log x|}$, which implies a strong attenuation of relative error. However, ϵ in general cannot be less than `EPSILON(1.0_wp)`.

For small x, errors are damped by the function and hence are limited by `EPSILON(1.0_wp)`.

For medium and large x, the error behaviour, like the function itself, is oscillatory, and hence only the absolute accuracy for the function can be maintained. For this range of x, the amplitude of the absolute error decays like $\sqrt{\frac{\pi x}{2}} e^{-x/\sqrt{2}}$ which implies a strong attenuation of error. Eventually, ker x, which asymptotically behaves like $\sqrt{\frac{\pi}{2x}} e^{-x/\sqrt{2}}$, becomes so small that it cannot be calculated without causing underflow, and the procedure returns zero. Note that for large x the errors are dominated by those of the Fortran intrinsic function EXP.

Procedure: nag_kelvin_kei

1 Description

nag_kelvin_kei evaluates an approximation to the Kelvin function kei x.

2 Usage

```
USE nag_kelvin_fun
[value =] nag_kelvin_kei(x  [, optional arguments])
```

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

3 Arguments

3.1 Mandatory Argument

x — real(kind=*wp*), intent(in)

Input: the argument x of the function.

Constraints: x ≥ 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 underflow. Argument x is too large. No computation has been performed as the value of the function becomes so small that it cannot be computed without 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

- For $x < 0$, the function kei x is undefined and the procedure will fail for such arguments.
- For $0 \leq x \leq 1$,

  $$\text{kei } x = -\frac{\pi}{4} f(t) + \frac{x^2}{4}[-g(t)\log x + v(t)]$$

  where $f(t)$, $g(t)$ and $v(t)$ are expansions in the variable $t = 2x^4 - 1$.
- For $1 < x \leq 3$,

  $$\text{kei } x = \exp\left(-\frac{9}{8}x\right) u(t)$$

  where $u(t)$ is an expansion in the variable $t = x - 2$.
- For $x > 3$,

  $$\text{kei } x = \sqrt{\frac{\pi}{2x}} e^{-x/\sqrt{2}} \left[\left(1 + \frac{1}{x}\right) c(t)\sin\beta + \frac{1}{x} d(t)\cos\beta\right]$$

  where $\beta = \frac{x}{\sqrt{2}} + \frac{\pi}{8}$, and $c(t)$ and $d(t)$ are expansions in the variable $t = \frac{6}{x} - 1$.
- For x near zero, the result is computed as

  $$\text{kei } x = -\frac{\pi}{4} + \left(1 - \gamma - \log\left(\frac{x}{2}\right)\right)\frac{x^2}{4}$$

  where γ is the Euler contant.
- When x is even closer to zero, it uses

  $$\text{kei } x = -\frac{\pi}{4}.$$

- For large x, kei x is asymptotically given by $\sqrt{\frac{\pi}{2x}} e^{-x/\sqrt{2}}$ and this becomes so small that it cannot be computed without underflow and the procedure returns zero.

6.2 Accuracy

Let E be the absolute error in the result and δ be the relative error in the argument. If δ is somewhat larger than `EPSILON(1.0_wp)`, then we have:

$$E \simeq \left| x/\sqrt{2}(-\text{ker}_1\, x + \text{kei}_1\, x) \right| \delta.$$

For small x, errors are attenuated by the function and hence are limited by `EPSILON(1.0_wp)`.
For medium and large x, the error behaviour, like the function itself, is oscillatory and hence only absolute accuracy of the function can be maintained. For this range of x, the amplitude of the absolute error decays like $\sqrt{\frac{\pi x}{2}} e^{-x/\sqrt{2}}$, which implies a strong attenuation of error. Eventually, kei x, which is asymptotically given by $\sqrt{\frac{\pi}{2x}} e^{-x/\sqrt{2}}$, becomes so small that it cannot be calculated without causing underflow and therefore the procedure returns zero. Note that for large x, the errors are dominated by those of the Fortran intrinsic function EXP.

Example 1: Evaluation of the Kelvin functions

This example program evaluates the four Kelvin functions `nag_kelvin_ber`, `nag_kelvin_bei`, `nag_kelvin_ker` and `nag_kelvin_kei` at a set of values of the argument `x`.

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_kelvin_fun_ex01

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

  ! .. Use Statements ..
  USE nag_examples_io, ONLY : nag_std_out
  USE nag_kelvin_fun, ONLY : nag_kelvin_ber, nag_kelvin_bei, &
   nag_kelvin_ker, nag_kelvin_kei
  ! .. Implicit None Statement ..
  IMPLICIT NONE
  ! .. Intrinsic Functions ..
  INTRINSIC KIND
  ! .. Parameters ..
  INTEGER, PARAMETER :: n = 7
  INTEGER, PARAMETER :: wp = KIND(1.0D0)
  ! .. Local Scalars ..
  INTEGER :: i
  REAL (wp) :: bei, ber, kei, ker
  ! .. Local Arrays ..
  REAL (wp) :: x(n)
  ! .. Executable Statements ..

  WRITE (nag_std_out,*) 'Example Program Results for nag_kelvin_fun_ex01'

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,*) '       x         ber x'

  x = (/ 0.1_wp, 1.0_wp, 2.5_wp, 5.0_wp, 10.0_wp, 15.0_wp, -1.0_wp/)
  DO i = 1, n

    ber = nag_kelvin_ber(x(i))

    WRITE (nag_std_out,'(1X,1P,2E12.3)') x(i), ber
  END DO

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,*) '       x         bei x'

  x = (/ 0.1_wp, 1.0_wp, 2.5_wp, 5.0_wp, 10.0_wp, 15.0_wp, -1.0_wp/)
  DO i = 1, n

    bei = nag_kelvin_bei(x(i))

    WRITE (nag_std_out,'(1X,1P,2E12.3)') x(i), bei
  END DO

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,*) '       x         ker x'

  x = (/ 0.01_wp, 0.1_wp, 1.0_wp, 2.5_wp, 5.0_wp, 10.0_wp, 15.0_wp/)
  DO i = 1, n
```

```
      ker = nag_kelvin_ker(x(i))

      WRITE (nag_std_out,'(1X,1P,2E12.3)') x(i), ker
    END DO

    WRITE (nag_std_out,*)
    WRITE (nag_std_out,*) '       x         kei x'

    x = (/ 0.0_wp, 0.1_wp, 1.0_wp, 2.5_wp, 5.0_wp, 10.0_wp, 15.0_wp/)
    DO i = 1, n

      kei = nag_kelvin_kei(x(i))

      WRITE (nag_std_out,'(1X,1P,2E12.3)') x(i), kei
    END DO

  END PROGRAM nag_kelvin_fun_ex01
```

2 Program Data

None.

3 Program Results

```
 Example Program Results for nag_kelvin_fun_ex01

       x         ber x
   1.000E-01   1.000E+00
   1.000E+00   9.844E-01
   2.500E+00   4.000E-01
   5.000E+00  -6.230E+00
   1.000E+01   1.388E+02
   1.500E+01  -2.967E+03
  -1.000E+00   9.844E-01

       x         bei x
   1.000E-01   2.500E-03
   1.000E+00   2.496E-01
   2.500E+00   1.457E+00
   5.000E+00   1.160E-01
   1.000E+01   5.637E+01
   1.500E+01  -2.953E+03
  -1.000E+00   2.496E-01

       x         ker x
   1.000E-02   4.721E+00
   1.000E-01   2.420E+00
   1.000E+00   2.867E-01
   2.500E+00  -6.969E-02
   5.000E+00  -1.151E-02
   1.000E+01   1.295E-04
   1.500E+01  -1.514E-08

       x         kei x
   0.000E+00  -7.854E-01
   1.000E-01  -7.769E-01
   1.000E+00  -4.950E-01
   2.500E+00  -1.107E-01
   5.000E+00   1.119E-02
   1.000E+01  -3.075E-04
```

```
    1.500E+01   7.963E-06
```

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_kelvin_fun_ex02`
: Evaluation of the Kelvin function ber x.

`nag_kelvin_fun_ex03`
: Evaluation of the Kelvin function bei x.

`nag_kelvin_fun_ex04`
: Evaluation of the Kelvin function ker x.

`nag_kelvin_fun_ex05`
: Evaluation of the Kelvin function kei x.

References

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