NAG Library Function Document

nag_bessel_zeros (s17alc)

1 Purpose

nag_bessel_zeros (s17alc) determines the leading **n** zeros of one of the Bessel functions $J_\alpha(x)$, $Y_\alpha(x)$, $J'_\alpha(x)$ or $Y'_\alpha(x)$ for real x and non-negative α.

2 Specification

```
#include <nag.h>
#include <nags.h>

void nag_bessel_zeros (double a, Integer n, Integer mode, double rel,
     double x[], NagError *fail)
```

3 Description

nag_bessel_zeros (s17alc) attempts to find the leading N zeros of one of the Bessel functions $J_\alpha(x)$, $Y_\alpha(x)$, $J'_\alpha(x)$ or $Y'_\alpha(x)$, where x is real. When α is real, these functions each have an infinite number of real zeros, all of which are simple with the possible exception of $x = 0$. If $\alpha \geq 0$, the nth positive zero is denoted by $j_{\alpha,n}, j'_{\alpha,n}, y_{\alpha,n}$ and $y'_{\alpha,n}$, respectively, for $n = 1, 2, \ldots, N$, except that $x = 0$ is counted as the first zero of $J'_\alpha(x)$ when $\alpha = 0$. Since $J'_0(x) = -J_1(x)$, it therefore follows that $j'_{0,1} = 0$ and $j'_{0,n} = -j_{1,n-1}$ for $n = 2, 3, \ldots, N - 1$. Further details can be found in Section 9.5 of Abramowitz and Stegun (1972).

nag_bessel_zeros (s17alc) is based on Algol 60 procedures given by Temme (1979). Initial approximations to the zeros are computed from asymptotic expansions. These are then improved by higher-order Newton iteration making use of the differential equation for the Bessel functions.

4 References

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

Temme N M (1976) On the numerical evaluation of the ordinary Bessel function of the second kind *J. Comput. Phys.* **21** 343–350

Temme N M (1979) An algorithm with Algol 60 program for the computation of the zeros of ordinary Bessel functions and those of their derivatives *J. Comput. Phys.* **32** 270–279

5 Arguments

1: **a** – double *Input*

On entry: the order α of the function.

Constraint: $0.0 \leq \mathbf{a} \leq 100000.0$.

2: **n** – Integer *Input*

On entry: the number N of zeros required.

Constraint: $\mathbf{n} \geq 1$.

3: **mode** – Integer *Input*

On entry: specifies the form of the function whose zeros are required.

mode $= 1$
The zeros of $J_\alpha(x)$ are required.

mode $= 2$
The zeros of $Y_\alpha(x)$ are required;

mode $= 3$
The zeros of $J'_\alpha(x)$ are required;

mode $= 4$
The zeros of $Y'_\alpha(x)$ are required.

Constraint: $1 \leq$ **mode** ≤ 4.

4: **rel** – double *Input*

On entry: the relative accuracy to which the zeros are required.

Suggested value: the square root of the ***machine precision***.

Constraint: **rel** > 0.0.

5: **x**[**n**] – double *Output*

On exit: the N required zeros of the function specified by **mode**.

6: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **mode** = ⟨*value*⟩.
Constraint: **mode** ≤ 4.

On entry, **mode** = ⟨*value*⟩.
Constraint: **mode** ≥ 1.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 1.

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.

NE_REAL

On entry, **a** = ⟨*value*⟩.
Constraint: **a** ≤ 100000.0.

On entry, **a** = ⟨*value*⟩.
Constraint: **a** ≥ 0.0.

On entry, **rel** = ⟨*value*⟩.
Constraint: **rel** > 0.0.

7 Accuracy

If the value of **rel** is set to 10^{-d}, then the required zeros should have approximately d correct significant digits.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example determines the leading five positive zeros of the Bessel function $J_0(x)$.

10.1 Program Text

```
/* nag_bessel_zeros (s17alc) Example Program.
 *
 * Copyright 2000 Numerical Algorithms Group.
 *
 * NAG C Library
 *
 * Mark 6, 2000.
 * Mark 7, revised, 2001.
 *
 */

#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nags.h>
#include <nagx02.h>

int main(void)
{

#define NMAX 100

  Integer  exit_status = 0, i, mode, n;
  NagError fail;
  double   a, rel, *x = 0;

  INIT_FAIL(fail);

  /* Skip heading in data file */
  scanf("%*[^\n]");
  printf("nag_bessel_zeros (s17alc) Example Program Results\n\n");

  if (!(x = NAG_ALLOC(NMAX, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
    }
  /* nag_machine_precision (x02ajc).
   * The machine precision
   */
  rel = sqrt(nag_machine_precision);
  scanf("%lf %ld %ld", &a, &n, &mode);
  /* nag_bessel_zeros (s17alc).
   * Zeros of Bessel functions J_alpha(x), (J_alpha')(x),
   * Y_alpha(x) or (Y_alpha')(x)
   */
```

```
  nag_bessel_zeros(a, n, mode, rel, x, &fail);

  if (fail.code == NE_NOERROR)
    {
      printf("   a   n   mode\n");
      printf(" %4.1f%3ld%6ld\n\n", a, n, mode);
      if (mode == 1)
        printf("Leading n positive zeros of J\n");
      else if (mode == 2)
        printf("Leading n positive zeros of Y\n");
      else if (mode == 3)
        {
          if (a == 0.0)
            printf("Leading n non-negative zeros of J’\n");
          else
            printf("Leading n positive zeros of J’\n");
        }
      else if (mode == 4)
        printf("Leading n positive zeros of Y’\n\n");
      for (i = 0; i <= n-1; ++i)
        printf("  x = %13.4e\n", x[i]);
      printf("\n");
    }
  else
    {
      printf("Error from nag_bessel_zeros (s17alc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }
 END:
  NAG_FREE(x);
  return exit_status;
}
```

10.2 Program Data

```
nag_bessel_zeros (s17alc) Example Program Data
 0.0  5  1 : Values of a, n and mode
```

10.3 Program Results

```
nag_bessel_zeros (s17alc) Example Program Results

   a   n   mode
  0.0  5     1

Leading n positive zeros of J
  x =   2.4048e+00
  x =   5.5201e+00
  x =   8.6537e+00
  x =   1.1792e+01
  x =   1.4931e+01
```