NAG Library Function Document

nag_zero_cont_func_cntin_rcomm (c05axc)

1 Purpose

nag_zero_cont_func_cntin_rcomm (c05axc) attempts to locate a zero of a continuous function using a continuation method based on a secant iteration. It uses reverse communication for evaluating the function.

2 Specification

```
#include <nag.h>
#include <nagc05.h>
```

```
void nag_zero_cont_func_cntin_rcomm (double *x, double fx, double tol,
     Nag_ErrorControl ir, double scal, double c[], Integer *ind,
     NagError *fail)
```

3 Description

nag_zero_cont_func_cntin_rcomm (c05axc) uses a modified version of an algorithm given in Swift and Lindfield (1978) to compute a zero α of a continuous function $f(x)$. The algorithm used is based on a continuation method in which a sequence of problems

$$f(x) - \theta_r f(x_0), \quad r = 0, 1, \ldots, m$$

are solved, where $1 = \theta_0 > \theta_1 > \cdots > \theta_m = 0$ (the value of m is determined as the algorithm proceeds) and where x_0 is your initial estimate for the zero of $f(x)$. For each θ_r the current problem is solved by a robust secant iteration using the solution from earlier problems to compute an initial estimate.

You must supply an error tolerance **tol**. **tol** is used directly to control the accuracy of solution of the final problem ($\theta_m = 0$) in the continuation method, and $\sqrt{\textbf{tol}}$ is used to control the accuracy in the intermediate problems ($\theta_1, \theta_2, \ldots, \theta_{m-1}$).

4 References

Swift A and Lindfield G R (1978) Comparison of a continuation method for the numerical solution of a single nonlinear equation *Comput. J.* **21** 359–362

5 Arguments

Note: this function uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the argument **ind**. Between intermediate exits and re-entries, **all arguments other than fx must remain unchanged**.

1: **x** – double * *Input/Output*

On initial entry: an initial approximation to the zero.

On intermediate exit: the point at which f must be evaluated before re-entry to the function.

On final exit: the final approximation to the zero.

2: **fx** – double *Input*

On initial entry: if **ind** $= 1$, **fx** need not be set.

If **ind** $= -1$, **fx** must contain $f(\mathbf{x})$ for the initial value of **x**.

On intermediate re-entry: must contain $f(\mathbf{x})$ for the current value of **x**.

3: **tol** – double *Input*

On initial entry: a value that controls the accuracy to which the zero is determined. **tol** is used in determining the convergence of the secant iteration used at each stage of the continuation process. It is used directly when solving the last problem ($\theta_m = 0$ in Section 3), and $\sqrt{\textbf{tol}}$ is used for the problem defined by θ_r, $r < m$. Convergence to the accuracy specified by **tol** is not guaranteed, and so you are recommended to find the zero using at least two values for **tol** to check the accuracy obtained.

Constraint: **tol** > 0.0.

4: **ir** – Nag_ErrorControl *Input*

On initial entry: indicates the type of error test required, as follows. Solving the problem defined by θ_r, $1 \leq r \leq m$, involves computing a sequence of secant iterates $x_r^0, x_r^1, \ldots$. This sequence will be considered to have converged only if:

for **ir** = Nag_Mixed,

$$\left|x_r^{(i+1)} - x_r^{(i)}\right| \leq eps \times \max\left(1.0, \left|x_r^{(i)}\right|\right),$$

for **ir** = Nag_Absolute,

$$\left|x_r^{(i+1)} - x_r^{(i)}\right| \leq eps,$$

for **ir** = Nag_Relative,

$$\left|x_r^{(i+1)} - x_r^{(i)}\right| \leq eps \times \left|x_r^{(i)}\right|,$$

for some $i > 1$; here eps is either **tol** or $\sqrt{\textbf{tol}}$ as discussed above. Note that there are other subsidiary conditions (not given here) which must also be satisfied before the secant iteration is considered to have converged.

Constraint: **ir** = Nag_Mixed, Nag_Absolute or Nag_Relative.

5: **scal** – double *Input*

On initial entry: a factor for use in determining a significant approximation to the derivative of $f(x)$ at $x = x_0$, the initial value. A number of difference approximations to $f'(x_0)$ are calculated using

$$f'(x_0) \sim (f(x_0 + h) - f(x_0))/h$$

where $|h| < |\textbf{scal}|$ and h has the same sign as **scal**. A significance (cancellation) check is made on each difference approximation and the approximation is rejected if insignificant.

Suggested value: $\sqrt{\epsilon}$, where ϵ is the ***machine precision*** returned by nag_machine_precision (X02AJC).

Constraint: **scal** must be sufficiently large that $\mathbf{x} + \textbf{scal} \neq \mathbf{x}$ on the computer.

6: **c**[**26**] – double *Communication Array*

(**c**[4] contains the current θ_r, this value may be useful in the event of an error exit.)

7: **ind** – Integer * *Input/Output*

On initial entry: must be set to 1 or -1.

ind = 1
  fx need not be set.

ind = -1
  fx must contain $f(\mathbf{x})$.

On intermediate exit: contains 2, 3 or 4. The calling program must evaluate f at **x**, storing the result in **fx**, and re-enter nag_zero_cont_func_cntin_rcomm (c05axc) with all other arguments unchanged.

On final exit: contains 0.

Constraint: on entry **ind** $= -1$, 1, 2, 3 or 4.

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

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

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

NE_CONTIN_AWAY_NOT_POSS

Continuation away from the initial point is not possible. This error exit will usually occur if the problem has not been properly posed or the error requirement is extremely stringent.

NE_CONTIN_PROB_NOT_SOLVED

Current problem in the continuation sequence cannot be solved. Perhaps the original problem had no solution or the continuation path passes through a set of insoluble problems: consider refining the initial approximation to the zero. Alternatively, **tol** is too small, and the accuracy requirement is too stringent, or too large and the initial approximation too poor.

NE_FINAL_PROB_NOT_SOLVED

Final problem (with $\theta_m = 0$) cannot be solved. It is likely that too much accuracy has been requested, or that the zero is at $\alpha = 0$ and **ir** $=$ Nag_Relative.

NE_INT

On initial entry, **ind** $=$ ⟨*value*⟩.
Constraint: **ind** $= -1$ or 1.

On intermediate entry, **ind** $=$ ⟨*value*⟩.
Constraint: **ind** $=$ 2, 3 or 4.

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.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_REAL

On entry, **scal** $=$ ⟨*value*⟩.
Constraint: $\mathbf{x} + \mathbf{scal} \neq \mathbf{x}$ (to machine accuracy).

On entry, **tol** $= \langle value \rangle$.
Constraint: **tol** > 0.0.

NE_SIGNIF_DERIVS_NOT_COMPUT

Significant derivatives of f cannot be computed. This can happen when f is almost constant and nonzero, for any value of **scal**.

7 Accuracy

The accuracy of the approximation to the zero depends on **tol** and **ir**. In general decreasing **tol** will give more accurate results. Care must be exercised when using the relative error criterion (**ir** $= 2$).

If the zero is at **x** $= 0$, or if the initial value of **x** and the zero bracket the point **x** $= 0$, it is likely that an error exit with **fail.code** = NE_CONTIN_AWAY_NOT_POSS, NE_CONTIN_PROB_NOT_SOLVED or NE_FINAL_PROB_NOT_SOLVED will occur.

It is possible to request too much or too little accuracy. Since it is not possible to achieve more than machine accuracy, a value of **tol** $\ll$ ***machine precision*** should not be input and may lead to an error exit with **fail.code** = NE_CONTIN_AWAY_NOT_POSS, NE_CONTIN_PROB_NOT_SOLVED or NE_FINAL_PROB_NOT_SOLVED. For the reasons discussed under **fail.code** = NE_CONTIN_PROB_NOT_SOLVED in Section 6, **tol** should not be taken too large, say no larger than **tol** $= 1.0\mathrm{e}{-3}$.

8 Parallelism and Performance

Not applicable.

9 Further Comments

For most problems, the time taken on each call to nag_zero_cont_func_cntin_rcomm (c05axc) will be negligible compared with the time spent evaluating $f(x)$ between calls to nag_zero_cont_func_cntin_rcomm (c05axc). However, the initial value of **x** and the choice of **tol** will clearly affect the timing. The closer that **x** is to the root, the less evaluations of f required. The effect of the choice of **tol** will not be large, in general, unless **tol** is very small, in which case the timing will increase.

10 Example

This example calculates a zero of $x - e^{-x}$ with initial approximation $x_0 = 1.0$, and **tol** $= 1.0\mathrm{e}{-3}$ and $1.0\mathrm{e}{-4}$.

10.1 Program Text

```
/* nag_zero_cont_func_cntin_rcomm (c05axc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 9, 2009.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <math.h>
#include <nagc05.h>
#include <nagx02.h>

int main(void)
{
  /* Scalars */
  Integer          exit_status = 0;
```

```
  double           fx, tol, x, scal, i;
  Integer          ind;
  Nag_ErrorControl ir;
  /* Arrays */
  double           c[26];
  NagError         fail;

  INIT_FAIL(fail);

  printf("nag_zero_cont_func_cntin_rcomm (c05axc) Example Program Results\n");

  scal = sqrt(nag_machine_precision);
  ir = Nag_Mixed;

  for (i = 3; i <= 4; i++)
    {
      tol = pow(10.0, -i);
      printf("\ntol = %13.4e\n\n", tol);
      x = 1.0;
      ind = 1;
      fx = 0.0;

      /* nag_zero_cont_func_cntin_rcomm (c05axc).
       * Locates a zero of a continuous function.
       * Reverse communication.
       */

      while (ind != 0)
        {
          nag_zero_cont_func_cntin_rcomm(&x, fx, tol, ir, scal, c, &ind, &fail);
          if (ind != 0)
            fx = x - exp(-x);
        }

      if (fail.code == NE_NOERROR)
        {
          printf("Root is %14.5f\n", x);
        }
      else
        {
          printf(
                  "Error from nag_zero_cont_func_cntin_rcomm (c05axc) %s\n",
                  fail.message);

          if (fail.code == NE_CONTIN_PROB_NOT_SOLVED ||
              fail.code == NE_FINAL_PROB_NOT_SOLVED)
            {
              printf("Final value = %14.5f, theta = %10.2f\n", x, c[4]);
            }

          exit_status = 1;
          goto END;
        }
    }

 END:

  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_zero_cont_func_cntin_rcomm (c05axc) Example Program Results

tol = 1.0000e-03

Root is        0.56715

tol = 1.0000e-04

Root is        0.56715
```