

NAG Library Function Document

nag_zero_cont_func_contin (c05awc)

1 Purpose

nag_zero_cont_func_contin (c05awc) attempts to locate a zero of a continuous function using a continuation method based on a secant iteration.

2 Specification

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

void nag_zero_cont_func_contin (double *x, double eps, double eta,
    double (*f)(double x, Nag_Comm *comm),
    Integer nfmmax, Nag_Comm *comm, NagError *fail)
```

3 Description

nag_zero_cont_func_contin (c05awc) attempts to obtain an approximation to a simple zero α of the function $f(x)$ given an initial approximation x to α . The zero is found by a call to nag_zero_cont_func_contin_rcomm (c05axc) whose specification should be consulted for details of the method used.

The approximation x to the zero α is determined so that at least one of the following criteria is satisfied:

- (i) $|x - \alpha| \sim \mathbf{eps}$,
- (ii) $|f(x)| < \mathbf{eta}$.

4 References

None.

5 Arguments

1: **x** – double * *Input/Output*

On entry: an initial approximation to the zero.

On exit: if **fail.code** = NE_NOERROR, NE_SECANT_ITER_FAILED or NE_TOO_MANY_CALLS it contains the approximation to the zero, otherwise it contains no useful information.

2: **eps** – double *Input*

On entry: an absolute tolerance to control the accuracy to which the zero is determined. In general, the smaller the value of **eps** the more accurate **x** will be as an approximation to α . Indeed, for very small positive values of **eps**, it is likely that the final approximation will satisfy $|x - \alpha| < \mathbf{eps}$. You are advised to call the function with more than one value for **eps** to check the accuracy obtained.

Constraint: **eps** > 0.0.

3: **eta** – double *Input*

On entry: a value such that if $|f(x)| < \mathbf{eta}$, x is accepted as the zero. **eta** may be specified as 0.0 (see Section 7).

- 4: **f** – function, supplied by the user *External Function*
f must evaluate the function f whose zero is to be determined.

The specification of **f** is:

```
double f (double x, Nag_Comm *comm)
```

1: **x** – double *Input*

On entry: the point at which the function must be evaluated.

2: **comm** – Nag_Comm * *Communication Structure*

Pointer to structure of type Nag_Comm; the following members are relevant to **f**.

user – double *

iuser – Integer *

p – Pointer

The type Pointer will be void *. Before calling nag_zero_cont_func_cntin (c05awc) you may allocate memory and initialize these pointers with various quantities for use by **f** when called from nag_zero_cont_func_cntin (c05awc) (see Section 3.2.1.1 in the Essential Introduction).

- 5: **nfmax** – Integer *Input*

On entry: the maximum permitted number of calls to **f** from nag_zero_cont_func_cntin (c05awc). If **f** is inexpensive to evaluate, **nfmax** should be given a large value (say > 1000).

Constraint: **nfmax** > 0.

- 6: **comm** – Nag_Comm * *Communication Structure*

The NAG communication argument (see Section 3.2.1.1 in the Essential Introduction).

- 7: **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 $\langle value \rangle$ had an illegal value.

NE_INT

On entry, **nfmax** = $\langle value \rangle$.

Constraint: **nfmax** > 0.

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.

A serious error occurred in an internal call to an auxiliary function.

Internal scale factor invalid for this problem. Consider using nag_zero_cont_func_cntin_rcomm (c05axc) instead and setting **scal**.

NE_REAL

On entry, **eps** = $\langle value \rangle$.

Constraint: **eps** > 0.0.

NE_SECANT_ITER_FAILED

Either **f** has no zero near **x** or too much accuracy has been requested. Check the coding of **f** or increase **eps**.

NE_TOO_MANY_CALLS

More than **nfmax** calls have been made to **f**.

***nfmax** may be too small for the problem (because **x** is too far away from the zero), or **f** has no zero near **x**, or too much accuracy has been requested in calculating the zero. Increase **nfmax**, check the coding of **f** or increase **eps**.*

7 Accuracy

The levels of accuracy depend on the values of **eps** and **eta**. If full machine accuracy is required, they may be set very small, resulting in an exit with **fail.code** = NE_SECANT_ITER_FAILED or NE_TOO_MANY_CALLS, although this may involve many more iterations than a lesser accuracy. You are recommended to set **eta** = 0.0 and to use **eps** to control the accuracy, unless you have considerable knowledge of the size of $f(x)$ for values of x near the zero.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_zero_cont_func_cntin (c05awc) depends primarily on the time spent evaluating the function f (see Section 5) and on how close the initial value of **x** is to the zero.

If a more flexible way of specifying the function f is required or if you wish to have closer control of the calculation, then the reverse communication function nag_zero_cont_func_cntin_rcomm (c05axc) is recommended instead of nag_zero_cont_func_cntin (c05awc).

10 Example

This example calculates the zero of $f(x) = e^{-x} - x$ from a starting value **x** = 1.0. Two calculations are made with **eps** = 1.0e−3 and 1.0e−4 for comparison purposes, with **eta** = 0.0 in both cases.

10.1 Program Text

```

/* nag_zero_cont_func_cntin (c05awc) Example Program.
 *
 * Copyright 2011 Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */

#include <nag.h>
#include <nagx04.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <math.h>
#include <nagc05.h>
#include <nagx02.h>

#ifdef __cplusplus
extern "C" {
#endif
static double NAG_CALL f(double x, Nag_Comm *comm);
#ifdef __cplusplus
}
#endif

```

```

int main(void)
{
    /* Scalars */
    Integer      nfmmax, exit_status = 0;
    double       eps, eta, x, i;
    /* Arrays */
    static double ruser[1] = {-1.0};

    NagError      fail;
    Nag_Comm      comm;

    printf("nag_zero_cont_func_cntin (c05awc) Example Program Results\n");

    /* For communication with user-supplied functions: */
    comm.user = ruser;

    for (i = 3; i <= 4; i++)
    {
        eps = pow(10.0, -i);
        x = 1.0;
        eta = 0.0;
        nfmmax = 200;

        INIT_FAIL(fail);

        /* nag_zero_cont_func_cntin (c05awc).
         * Locates a zero of a continuous function.
         */
        nag_zero_cont_func_cntin(&x, eps, eta, f, nfmmax, &comm, &fail);

        if (fail.code == NE_NOERROR)
        {
            printf("\nWith eps = %10.2e, root is %14.5f\n", eps, x);
        }
        else
        {
            printf(
                "Error from nag_zero_cont_func_cntin (c05awc) %s\n",
                fail.message);

            if (fail.code == NE_TOO_MANY_CALLS ||
                fail.code == NE_SECANT_ITER_FAILED)
            {
                printf("\nWith eps = %10.2e, final value is %14.5f\n",
                    eps, x);
            }

            exit_status = 1;
            goto END;
        }
    }

    END:

    return exit_status;
}

static double NAG_CALL f(double x, Nag_Comm *comm)
{
    if (comm->user[0] == -1.0)
    {
        printf("(User-supplied callback f, first invocation.)\n");
        comm->user[0] = 0.0;
    }
    return exp(-x)-x;
}

```

10.2 Program Data

None.

10.3 Program Results

nag_zero_cont_func_cntin (c05awc) Example Program Results
(User-supplied callback f, first invocation.)

With eps = 1.00e-03, root is 0.56715

With eps = 1.00e-04, root is 0.56715
