NAG Library Function Document

nag_zero_cont_func_brent (c05ayc)

1 Purpose

nag_zero_cont_func_brent (c05ayc) locates a simple zero of a continuous function in a given interval using Brent’s method, which is a combination of nonlinear interpolation, linear extrapolation and bisection.

2 Specification

```c
#include <nag.h>
#include <nagc05.h>
void nag_zero_cont_func_brent (double a, double b, double eps, double eta,
     double (*f)(double x, Nag_Comm *comm),
     double *x, Nag_Comm *comm, NagError *fail)
```

3 Description

nag_zero_cont_func_brent (c05ayc) attempts to obtain an approximation to a simple zero of the function \( f(x) \) given an initial interval \([a, b]\) such that \( f(a) \times f(b) \leq 0 \).

The approximation \( x \) to the zero \( \alpha \) is determined so that at least one of the following criteria is satisfied:

(i) \( |x - \alpha| \leq \text{eps} \),

(ii) \( |f(x)| \leq \text{eta} \).

4 References


5 Arguments

1: \( a \) – double

\( \text{Input} \)

\( \text{On entry:} \ a, \ \text{the lower bound of the interval.} \)

2: \( b \) – double

\( \text{Input} \)

\( \text{On entry:} \ b, \ \text{the upper bound of the interval.} \)

\( \text{Constraint:} \ b \neq a. \)

3: \( \text{eps} \) – double

\( \text{Input} \)

\( \text{On entry:} \ \text{the termination tolerance on} \ x \ \text{(see Section 3).} \)

\( \text{Constraint:} \ \text{eps} > 0.0. \)

4: \( \text{eta} \) – double

\( \text{Input} \)

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

5: \( f \) – function, supplied by the user

\( \text{External Function} \)

\( \text{f must evaluate the function} \ f \ \text{whose zero is to be determined.} \)
The specification of f is:

double f (double x, Nag_Comm *comm)

1: x – double
   \textit{Input}
   On entry: the point at which the function must be evaluated.

2: comm – Nag_Comm *
   Pointer to structure of type Nag_Comm; the following members are relevant to f.
   \begin{itemize}
   \item user – double *
   \item iuser – Integer *
   \item p – Pointer
\end{itemize}
   The type Pointer will be \texttt{void *}. Before calling nag_zero_cont_func_brent (c05ayc) you may allocate memory and initialize these pointers with various quantities for use by f when called from nag_zero_cont_func_brent (c05ayc) (see Section 3.2.1.1 in the Essential Introduction).

6: x – double *
   \textit{Output}
   On exit: if fail.code = NE_NOERROR or NE_TOO_SMALL, x is the final approximation to the zero. If fail.code = NE_PROBABLE_POLE, x is likely to be a pole of \(f(x)\). Otherwise, x contains no useful information.

7: comm – Nag_Comm *
   The NAG communication argument (see Section 3.2.1.1 in the Essential Introduction).

8: fail – NagError *
   \textit{Input/Output}
   The NAG error argument (see Section 3.6 in the Essential Introduction).

6 \textbf{Error Indicators and Warnings}

\textbf{NE_ALLOC_FAIL}
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}
On entry, argument \(\langle\text{value}\rangle\) had an illegal value.

\textbf{NE_FUNC_END_VAL}
On entry, \(f(a)\) and \(f(b)\) have the same sign with neither equalling 0.0: \(f(a) = \langle\text{value}\rangle\) and \(f(b) = \langle\text{value}\rangle\).

\textbf{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.

\textbf{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.
The function values in the interval \([a, b]\) might contain a pole rather than a zero. Reducing \(\text{eps}\) may help in distinguishing between a pole and a zero.

On entry, \(\text{eps} = \langle\text{value}\rangle\).
Constraint: \(\text{eps} > 0.0\).

On entry, \(a = \langle\text{value}\rangle\) and \(b = \langle\text{value}\rangle\).
Constraint: \(a \neq b\).

No further improvement in the solution is possible. \(\text{eps}\) is too small: \(\text{eps} = \langle\text{value}\rangle\). The final value of \(x\) returned is an accurate approximation to the zero.

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

The time taken by nag_zero_cont_func_brent (c05ayc) depends primarily on the time spent evaluating \(f\) (see Section 5).

This example calculates an approximation to the zero of \(e^{-x} - x\) within the interval \([0, 1]\) using a tolerance of \(\text{eps} = 1.0e-5\).
int main(void)
{
    static double ruser[1] = {-1.0};
    Integer exit_status = 0;
    double a, b;
    double x, eta, eps;
    NagError fail;
    Nag_Comm comm;

    INIT_FAIL(fail);

    printf("nag_zero_cont_func_brent (c05ayc) Example Program Results\n");

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

    a = 0.0;
    b = 1.0;
    eps = 1e-05;
    eta = 0.0;

    /* nag_zero_cont_func_brent (c05ayc).
    * Zero of a continuous function using Brent’s algorithm
    */
    nag_zero_cont_func_brent(a, b, eps, eta, f, &x, &comm, &fail);
    if (fail.code == NE_NOERROR)
    {
        printf("Zero = %12.5f\n", x);
    } else
    {
        printf("%s", fail.message);
        if (fail.code == NE_TOO_SMALL ||
            fail.code == NE_PROBABLE_POLE)
        {
            printf("Final point = %12.5f\n", 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_brent (c05ayc) Example Program Results
(User-supplied callback f, first invocation.)
Zero = 0.56714