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

```
#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 α is determined so that at least one of the following criteria is satisfied:

(i) $|x - \alpha| \leq$ **eps**,

(ii) $|f(x)| \leq$ **eta**.

4 References

Brent R P (1973) *Algorithms for Minimization Without Derivatives* Prentice–Hall

5 Arguments

1: **a** – double *Input*

On entry: a, the lower bound of the interval.

2: **b** – double *Input*

On entry: b, the upper bound of the interval.

Constraint: **b** $\neq$ **a**.

3: **eps** – double *Input*

On entry: the termination tolerance on x (see Section 3).

Constraint: **eps** > 0.0.

4: **eta** – double *Input*

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

5: **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 *

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_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 * *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 * *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_FUNC_END_VAL

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

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_PROBABLE_POLE

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

NE_REAL

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

NE_REAL_2

On entry, **a** $= \langle value \rangle$ and **b** $= \langle value \rangle$.
Constraint: **a** $\neq$ **b**.

NE_TOO_SMALL

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

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_TOO_SMALL, 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_brent (c05ayc) depends primarily on the time spent evaluating **f** (see Section 5).

10 Example

This example calculates an approximation to the zero of $e^{-x} - x$ within the interval $[0, 1]$ using a tolerance of **eps** $=$ 1.0e−5.

10.1 Program Text

```
/* nag_zero_cont_func_brent (c05ayc) Example Program.
 *
 * Copyright 2014 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>

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

```
}
#endif

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\n", 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
```