# NAG Toolbox

# nag_roots_contfn_brent_interval (c05au)

## 1 Purpose

nag_roots_contfn_brent_interval (c05au) locates a simple zero of a continuous function from a given starting value. It uses a binary search to locate an interval containing a zero of the function, then Brent's method, which is a combination of nonlinear interpolation, linear extrapolation and bisection, to locate the zero precisely.

## 2 Syntax

```
[x, a, b, user, ifail] = nag_roots_contfn_brent_interval(x, h, eps, eta, f,
'user', user)
```

```
[x, a, b, user, ifail] = c05au(x, h, eps, eta, f, 'user', user)
```

## 3 Description

nag_roots_contfn_brent_interval (c05au) attempts to locate an interval $[a, b]$ containing a simple zero of the function $f(x)$ by a binary search starting from the initial point $x = \mathbf{x}$ and using repeated calls to nag_roots_contfn_interval_rcomm (c05av). If this search succeeds, then the zero is determined to a user-specified accuracy by a call to nag_roots_contfn_brent (c05ay). The specifications of functions nag_roots_contfn_interval_rcomm (c05av) and nag_roots_contfn_brent (c05ay) should be consulted for details of the methods used.

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

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

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

## 4 References

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

## 5 Parameters

### 5.1 Compulsory Input Parameters

1: **x** – REAL (KIND=nag_wp)

An initial approximation to the zero.

2: **h** – REAL (KIND=nag_wp)

A step length for use in the binary search for an interval containing the zero. The maximum interval searched is $[\mathbf{x} - 256.0 \times \mathbf{h}, \mathbf{x} + 256.0 \times \mathbf{h}]$.

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

3: **eps** – REAL (KIND=nag_wp)

The termination tolerance on $x$ (see Section 3).

*Constraint*: $\mathbf{eps} > 0.0$.

4: **eta** – REAL (KIND=nag_wp)

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** – REAL (KIND=nag_wp) FUNCTION, supplied by the user.

**f** must evaluate the function $f$ whose zero is to be determined.

```
[result, user] = f(x, user)
```

**Input Parameters**

1: **x** – REAL (KIND=nag_wp)

The point at which the function must be evaluated.

2: **user** – INTEGER array

**f** is called from nag_roots_contfn_brent_interval (c05au) with the object supplied to nag_roots_contfn_brent_interval (c05au).

**Output Parameters**

1: **result**

The value of $f$ evaluated at **x**.

2: **user** – INTEGER array

## 5.2 Optional Input Parameters

1: **user** – INTEGER array

**user** is not used by nag_roots_contfn_brent_interval (c05au), but is passed to **f**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

## 5.3 Output Parameters

1: **x** – REAL (KIND=nag_wp)

If **ifail** $= 0$ or $4$, **x** is the final approximation to the zero.

If **ifail** $= 3$, **x** is likely to be a pole of $f(x)$.

Otherwise, **x** contains no useful information.

2: **a** – REAL (KIND=nag_wp)
3: **b** – REAL (KIND=nag_wp)

The lower and upper bounds respectively of the interval resulting from the binary search. If the zero is determined exactly such that $f(x) = 0.0$ or is determined so that $|f(x)| \leq$ **eta** at any stage in the calculation, then on exit **a** $=$ **b** $= x$.

4: **user** – INTEGER array

5: **ifail** – INTEGER

**ifail** $= 0$ unless the function detects an error (see Section 5).

## 6 Error Indicators and Warnings

Errors or warnings detected by the function:

**ifail** $= 1$

Constraint: **eps** $> 0.0$.

Constraint: $\mathbf{x} + \mathbf{h} \neq \mathbf{x}$ (to machine accuracy).

**ifail** $= 2$

An interval containing the zero could not be found. Increasing **h** and calling nag_roots_contfn_brent_interval (c05au) again will increase the range searched for the zero. Decreasing **h** and calling nag_roots_contfn_brent_interval (c05au) again will refine the mesh used in the search for the zero.

**ifail** $= 3$ (*warning*)

Solution may be a pole rather than a zero.

**ifail** $= 4$ (*warning*)

The tolerance **eps** has been set too small for the problem being solved.

**ifail** $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

**ifail** $= -399$

Your licence key may have expired or may not have been installed correctly.

**ifail** $= -999$

Dynamic memory allocation failed.

## 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 **ifail** $= 4$, 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 Further Comments

The time taken by nag_roots_contfn_brent_interval (c05au) depends primarily on the time spent evaluating **f** (see Section 5). The accuracy of the initial approximation **x** and the value of **h** will have a somewhat unpredictable effect on the timing.

If it is important to determine an interval of relative length less than $2 \times$ **eps** containing the zero, or if **f** is expensive to evaluate and the number of calls to **f** is to be restricted, then use of nag_roots_contfn_interval_rcomm (c05av) followed by nag_roots_contfn_brent_rcomm (c05az) is recommended. Use of this combination is also recommended when the structure of the problem to be solved does not permit a simple **f** to be written: the reverse communication facilities of these functions are more flexible than the direct communication of **f** required by nag_roots_contfn_brent_interval (c05au).

If the iteration terminates with successful exit and $\mathbf{a} = \mathbf{b} = \mathbf{x}$ there is no guarantee that the value returned in **x** corresponds to a simple zero and you should check whether it does.

One way to check this is to compute the derivative of $f$ at the point **x**, preferably analytically, or, if this is not possible, numerically, perhaps by using a central difference estimate. If $f'(\mathbf{x}) = 0.0$, then **x** must correspond to a multiple zero of $f$ rather than a simple zero.

# 9 Example

This example calculates an approximation to the zero of $x - e^{-x}$ using a tolerance of **eps** $= 1.0\text{e}{-5}$ starting from **x** $= 1.0$ and using an initial search step **h** $= 0.1$.

## 9.1 Program Text

```
function c05au_example

fprintf('c05au example results\n\n');

x = 1;
h = 0.1;
eps = 1e-5;
eta = 0;
[x, a, b, user, ifail] = c05au(x, h, eps, eta, @f);

fprintf('Root is               %8.5f\n',x);
fprintf('Interval searched is [%8.5f, %8.5f]\n',a,b);

function [result, user] = f(x, user)
  result = x - exp(-x);
```

## 9.2 Program Results

```
c05au example results

Root is               0.56714
Interval searched is [ 0.50000,  0.90000]
```