NAG Toolbox

nag_roots_contfn_brent_rcomm (c05az)

1 Purpose

nag_roots_contfn_brent_rcomm (c05az) locates a simple zero of a continuous function in a given interval by using Brent's method, which is a combination of nonlinear interpolation, linear extrapolation and bisection. It uses reverse communication for evaluating the function.

2 Syntax

```
[x, y, c, ind, ifail] = nag_roots_contfn_brent_rcomm(x, y, fx, tolx, c, ind,
'ir', ir)
[x, y, c, ind, ifail] = c05az(x, y, fx, tolx, c, ind, 'ir', ir)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:

At Mark 23: **fx** is no longer an output parameter.

3 Description

You must supply **x** and **y** to define an initial interval $[a, b]$ containing a simple zero of the function $f(x)$ (the choice of **x** and **y** must be such that $f(\mathbf{x}) \times f(\mathbf{y}) \leq 0.0$). The function combines the methods of bisection, nonlinear interpolation and linear extrapolation (see Dahlquist and BjÎrck (1974)), to find a sequence of sub-intervals of the initial interval such that the final interval $[\mathbf{x}, \mathbf{y}]$ contains the zero and $|\mathbf{x} - \mathbf{y}|$ is less than some tolerance specified by **tolx** and **ir** (see Section 5). In fact, since the intermediate intervals $[\mathbf{x}, \mathbf{y}]$ are determined only so that $f(\mathbf{x}) \times f(\mathbf{y}) \leq 0.0$, it is possible that the final interval may contain a discontinuity or a pole of f (violating the requirement that f be continuous). nag_roots_contfn_brent_rcomm (c05az) checks if the sign change is likely to correspond to a pole of f and gives an error return in this case.

A feature of the algorithm used by this function is that unlike some other methods it guarantees convergence within about $(\log_2[(b-a)/\delta])^2$ function evaluations, where δ is related to the argument **tolx**. See Brent (1973) for more details.

nag_roots_contfn_brent_rcomm (c05az) returns to the calling program for each evaluation of $f(x)$. On each return you should set $\mathbf{fx} = f(\mathbf{x})$ and call nag_roots_contfn_brent_rcomm (c05az) again.

The function is a modified version of procedure ‘zeroin’ given by Brent (1973).

4 References

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

Bus J C P and Dekker T J (1975) Two efficient algorithms with guaranteed convergence for finding a zero of a function *ACM Trans. Math. Software* **1** 330–345

Dahlquist G and BjÎrck Đ (1974) *Numerical Methods* Prentice–Hall

5 Parameters

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**.

5.1 Compulsory Input Parameters

1: **x** – REAL (KIND=nag_wp)

2: **y** – REAL (KIND=nag_wp)

On initial entry: **x** and **y** must define an initial interval $[a, b]$ containing the zero, such that $f(\mathbf{x}) \times f(\mathbf{y}) \leq 0.0$. It is not necessary that $\mathbf{x} < \mathbf{y}$.

3: **fx** – REAL (KIND=nag_wp)

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**.

4: **tolx** – REAL (KIND=nag_wp)

On initial entry: the accuracy to which the zero is required. The type of error test is specified by **ir**.

Constraint: **tolx** > 0.0.

5: **c**(**17**) – REAL (KIND=nag_wp) array

On initial entry: if **ind** $= 1$, no elements of **c** need be set.

If **ind** $= -1$, **c**(1) must contain $f(\mathbf{y})$, other elements of **c** need not be set.

6: **ind** – INTEGER

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

ind $= 1$
: **fx** and **c**(1) need not be set.

ind $= -1$
: **fx** and **c**(1) must contain $f(\mathbf{x})$ and $f(\mathbf{y})$ respectively.

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

5.2 Optional Input Parameters

1: **ir** – INTEGER

Suggested value: **ir** $= 0$.

Default: 0

On initial entry: indicates the type of error test.

ir $= 0$
: The test is: $|\mathbf{x} - \mathbf{y}| \leq 2.0 \times \mathbf{tolx} \times \max(1.0, |\mathbf{x}|)$.

ir $= 1$
: The test is: $|\mathbf{x} - \mathbf{y}| \leq 2.0 \times \mathbf{tolx}$.

ir $= 2$
: The test is: $|\mathbf{x} - \mathbf{y}| \leq 2.0 \times \mathbf{tolx} \times |\mathbf{x}|$.

Constraint: **ir** $= 0$, 1 or 2.

5.3 Output Parameters

1: **x** – REAL (KIND=nag_wp)
2: **y** – REAL (KIND=nag_wp)

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

On final exit: **x** and **y** define a smaller interval containing the zero, such that $f(\mathbf{x}) \times f(\mathbf{y}) \leq 0.0$, and $|\mathbf{x} - \mathbf{y}|$ satisfies the accuracy specified by **tolx** and **ir**, unless an error has occurred. If **ifail** $= 4$, **x** and **y** generally contain very good approximations to a pole; if **ifail** $= 5$, **x** and **y** generally contain very good approximations to the zero (see Section 6). If a point **x** is found such that $f(\mathbf{x}) = 0.0$, then on final exit $\mathbf{x} = \mathbf{y}$ (in this case there is no guarantee that **x** is a simple zero). In all cases, the value returned in **x** is the better approximation to the zero.

3: **c**(**17**) – REAL (KIND=nag_wp) array

On final exit: is undefined.

4: **ind** – INTEGER

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_roots_contfn_brent_rcomm (c05az) with all other arguments unchanged.

On final exit: contains 0.

5: **ifail** – INTEGER

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

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, $f(\mathbf{x})$ and $f(\mathbf{y})$ have the same sign with neither equalling 0.0.

ifail $= 2$

On entry, **ind** $\neq -1$, 1, 2, 3 or 4.

ifail $= 3$

On entry, **tolx** ≤ 0.0,
or **ir** $\neq 0$, 1 or 2.

ifail $= 4$ (*warning*)

An interval $[\mathbf{x}, \mathbf{y}]$ has been determined satisfying the error tolerance specified by **tolx** and **ir** and such that $f(\mathbf{x}) \times f(\mathbf{y}) \leq 0$. However, from observation of the values of f during the calculation of $[\mathbf{x}, \mathbf{y}]$, it seems that the interval $[\mathbf{x}, \mathbf{y}]$ contains a pole rather than a zero. Note that this error exit is not completely reliable: the error exit may be taken in extreme cases when $[\mathbf{x}, \mathbf{y}]$ contains a zero, or the error exit may not be taken when $[\mathbf{x}, \mathbf{y}]$ contains a pole. Both these cases occur most frequently when **tolx** is large.

ifail $= 5$ (*warning*)

The tolerance **tolx** is too small for the problem being solved. This indicator is only set when the interval containing the zero has been reduced to one of relative length at most 2ϵ, where ϵ is the ***machine precision***, but the exit condition specified by **ir** is not satisfied. It is unsafe to continue reducing the interval beyond this point, but the final values of **x** and **y** returned are accurate approximations to the zero.

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 accuracy of the final value **x** as an approximation of the zero is determined by **tolx** and **ir** (see Section 5). A relative accuracy criterion (**ir** $= 2$) should not be used when the initial values **x** and **y** are of different orders of magnitude. In this case a change of origin of the independent variable may be appropriate. For example, if the initial interval $[\mathbf{x}, \mathbf{y}]$ is transformed linearly to the interval $[1, 2]$, then the zero can be determined to a precise number of figures using an absolute (**ir** $= 1$) or relative (**ir** $= 2$) error test and the effect of the transformation back to the original interval can also be determined. Except for the accuracy check, such a transformation has no effect on the calculation of the zero.

8 Further Comments

For most problems, the time taken on each call to nag_roots_contfn_brent_rcomm (c05az) will be negligible compared with the time spent evaluating $f(x)$ between calls to nag_roots_contfn_brent_rcomm (c05az).

If the calculation terminates because $f(\mathbf{x}) = 0.0$, then on return **y** is set to **x**. (In fact, $\mathbf{y} = \mathbf{x}$ on return only in this case and, possibly, when **ifail** $= 5$.) There is no guarantee that the value returned in **x** corresponds to a **simple** root 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 a zero of $e^{-x} - x$ with an initial interval $[0, 1]$, **tolx** $= 1.0\text{e}{-5}$ and a mixed error test.

9.1 Program Text

```
function c05az_example

fprintf('c05az example results\n\n');

x = 0;
y = 1;
fx = 0;
tolx = 1e-05;
c = zeros(17,1);
ind = nag_int(1);
while (ind ~= 0)
  [x, y, c, ind, ifail] = c05az(x, y, fx, tolx, c, ind);
  fx = exp(-x) - x;
  fprintf('x = %8.5f fx = %12.4e ind = %d\n', x, fx, ind);
end
if ifail == 0
  fprintf('\nSolution: x = %8.5f y = %8.5f\n', x, y);
elseif ifail == 4 || ifail == 5
  fprintf('\nx = %8.5f y = %8.5f\n', x, y);
end
```

9.2 Program Results

```
     c05az example results

x =  0.00000 fx =   1.0000e+00 ind = 2
x =  1.00000 fx =  -6.3212e-01 ind = 3
x =  0.61270 fx =  -7.0814e-02 ind = 4
x =  0.56707 fx =   1.1542e-04 ind = 4
x =  0.56714 fx =  -9.4481e-07 ind = 4
x =  0.56713 fx =   1.4727e-05 ind = 4
x =  0.56714 fx =  -9.4481e-07 ind = 4
x =  0.56714 fx =  -9.4481e-07 ind = 0

Solution: x =  0.56714 y =  0.56713
```