NAG Toolbox

nag_roots_contfn_interval_rcomm (c05av)

1 Purpose

nag_roots_contfn_interval_rcomm (c05av) attempts to locate an interval containing a simple zero of a continuous function using a binary search. It uses reverse communication for evaluating the function.

2 Syntax

```
[x, h, y, c, ind, ifail] = nag_roots_contfn_interval_rcomm(x, fx, h, boundl, boundu, y, c, ind)
[x, h, y, c, ind, ifail] = c05av(x, fx, h, boundl, boundu, y, c, ind)
```

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 an initial point **x** and a step **h**. nag_roots_contfn_interval_rcomm (c05av) attempts to locate a short interval $[\mathbf{x}, \mathbf{y}] \subset [\mathbf{boundl}, \mathbf{boundu}]$ containing a simple zero of $f(x)$.

(On exit we may have $\mathbf{x} > \mathbf{y}$; **x** is determined as the first point encountered in a binary search where the sign of $f(x)$ differs from the sign of $f(x)$ at the initial input point **x**.) The function attempts to locate a zero of $f(x)$ using **h**, $0.1 \times \mathbf{h}$, $0.01 \times \mathbf{h}$ and $0.001 \times \mathbf{h}$ in turn as its basic step before quitting with an error exit if unsuccessful.

nag_roots_contfn_interval_rcomm (c05av) 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_interval_rcomm (c05av) again.

4 References

None.

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)

On initial entry: the best available approximation to the zero.

Constraint: **x** must lie in the closed interval $[\mathbf{boundl}, \mathbf{boundu}]$ (see below).

2: **fx** – REAL (KIND=nag_wp)

On initial entry: if $\mathbf{ind} = 1$, **fx** need not be set.

If $\mathbf{ind} = -1$, **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**.

3: **h** – REAL (KIND=nag_wp)

On initial entry: a basic step size which is used in the binary search for an interval containing a zero. The basic step sizes $\mathbf{h}, 0.1 \times \mathbf{h}$, $0.01 \times \mathbf{h}$ and $0.001 \times \mathbf{h}$ are used in turn when searching for the zero.

Constraint: either $\mathbf{x} + \mathbf{h}$ or $\mathbf{x} - \mathbf{h}$ must lie inside the closed interval $[\mathbf{boundl}, \mathbf{boundu}]$.

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

4: **boundl** – REAL (KIND=nag_wp)

5: **boundu** – REAL (KIND=nag_wp)

On initial entry: **boundl** and **boundu** must contain respectively lower and upper bounds for the interval of search for the zero.

Constraint: **boundl** $<$ **boundu**.

6: **y** – REAL (KIND=nag_wp)

On initial entry: need not be set.

7: **c**(**11**) – REAL (KIND=nag_wp) array

On initial entry: need not be set.

8: **ind** – INTEGER

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

ind $= 1$
fx need not be set.

ind $= -1$
fx must contain $f(\mathbf{x})$.

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

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **x** – REAL (KIND=nag_wp)

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

On final exit: contains one end of an interval containing the zero, the other end being in **y**, unless an error has occurred. If **ifail** $= 4$, **x** and **y** are the end points of the largest interval searched. If a zero is located exactly, its value is returned in **x** (and in **y**).

2: **h** – REAL (KIND=nag_wp)

On final exit: is undefined.

3: **y** – REAL (KIND=nag_wp)

On final exit: contains the closest point found to the final value of **x**, such that $f(\mathbf{x}) \times f(\mathbf{y}) \leq 0.0$. If a value **x** is found such that $f(\mathbf{x}) = 0$, then $\mathbf{y} = \mathbf{x}$. On final exit with **ifail** $= 4$, **x** and **y** are the end points of the largest interval searched.

4: **c**(**11**) – REAL (KIND=nag_wp) array

On final exit: if **ifail** $= 0$ or 4, $\mathbf{c}(1)$ contains $f(\mathbf{y})$.

5: **ind** – INTEGER

On intermediate exit: contains 2 or 3. The calling program must evaluate f at **x**, storing the result in **fx**, and re-enter nag_roots_contfn_interval_rcomm (c05av) with all other arguments unchanged.

On final exit: contains 0.

6: **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, **boundu** $\leq$ **boundl**,
or **x** $\notin$ [**boundl**, **boundu**],
or both **x** $+$ **h** and **x** $-$ **h** $\notin$ [**boundl**, **boundu**].

ifail $= 2$

On initial entry, **h** is too small to be used to perturb the initial value of **x** in the search.

ifail $= 3$

The argument **ind** is incorrectly set on initial or intermediate entry.

ifail $= 4$

nag_roots_contfn_interval_rcomm (c05av) has been unable to determine an interval containing a simple zero starting from the initial value of **x** and using the step **h**. If you have prior knowledge that a simple zero lies in the interval [**boundl**, **boundu**], you should vary **x** and **h** in an attempt to find it. (See also Section 9.)

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

nag_roots_contfn_interval_rcomm (c05av) is not intended to be used to obtain accurate approximations to the zero of $f(x)$ but rather to locate an interval containing a zero. This interval can then be used as input to an accurate rootfinder such as nag_roots_contfn_brent (c05ay) or nag_roots_contfn_brent_rcomm (c05az). The size of the interval determined depends somewhat unpredictably on the choice of **x** and **h**. The closer **x** is to the root and the **smaller** the initial value of **h**, then, in general, the smaller (more accurate) the interval determined; however, the accuracy of this statement depends to some extent on the behaviour of $f(x)$ near $x =$ **x** and on the size of **h**.

8 Further Comments

For most problems, the time taken on each call to nag_roots_contfn_interval_rcomm (c05av) will be negligible compared with the time spent evaluating $f(x)$ between calls to nag_roots_contfn_interval_rcomm (c05av). However, the initial value of **x** and **h** will clearly affect the timing. The closer **x** is to the root, and the **larger** the initial value of **h** then the less time taken. (However taking a large **h** can affect the accuracy and reliability of the function, see below.)

You are expected to choose **boundl** and **boundu** as physically (or mathematically) realistic limits on the interval of search. For example, it may be known, from physical arguments, that no zero of $f(x)$ of interest will lie outside [**boundl**, **boundu**]. Alternatively, $f(x)$ may be more expensive to evaluate for some values of **x** than for others and such expensive evaluations can sometimes be avoided by careful choice of **boundl** and **boundu**.

The choice of **boundl** and **boundu** affects the search only in that these values provide physical limitations on the search values and that the search is terminated if it seems, from the available information about $f(x)$, that the zero lies outside [**boundl**, **boundu**]. In this case (**ifail** $= 4$ on exit), only one of f(**boundl**) and f(**boundu**) may have been evaluated and a zero close to the other end of the interval could be missed. The actual interval searched is returned in the arguments **x** and **y** and you can call nag_roots_contfn_interval_rcomm (c05av) again to search the remainder of the original interval.

Though nag_roots_contfn_interval_rcomm (c05av) is intended primarily for determining an interval containing a zero of $f(x)$, it may be used to shorten a known interval. This could be useful if, for example, a large interval containing the zero is known and it is also known that the root lies close to one end of the interval; by setting **x** to this end of the interval and **h** small, a short interval will usually be determined. However, it is worth noting that once any interval containing a zero has been determined, a call to nag_roots_contfn_brent_rcomm (c05az) will usually be the most efficient way to calculate an interval of specified length containing the zero. To assist in this determination, the information in **fx** and in **x**, **y** and **c**(1) on successful exit from nag_roots_contfn_interval_rcomm (c05av) is in the correct form for a call to function nag_roots_contfn_brent_rcomm (c05az) with **ind** $= -1$.

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.) In this case, there is no guarantee that the value 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 finds a sub-interval of $[0.0, 4.0]$ containing a simple zero of $x^2 - 3x + 2$. The zero nearest to 3.0 is required and so we set $\mathbf{x} = 3.0$ initially.

9.1 Program Text

```
function c05av_example

fprintf('c05av example results\n\n');

x = 3;
fx = 0;
h = 0.1;
boundl = 0;
boundu = 4;
y = 0;
c = zeros(11, 1);
ind = nag_int(1);
while (ind ~= nag_int(0))
  [x, h, y, c, ind, ifail] = c05av(x, fx, h, boundl, boundu, y, c, ind);
```

```
  fx = x^2 -3*x + 2;
end

fprintf('Interval containing root is [%8.5f, %8.5f]\n',x,y);
fprintf('where f(%8.5f) = %8.5f and f(%8.5f) = %8.5f\n',x,fx,y,c(1));
```

9.2 Program Results

```
     c05av example results

Interval containing root is [ 1.70000,  2.50000]
where f( 1.70000) = -0.21000 and f( 2.50000) =  0.75000
```