

NAG Toolbox for MATLAB

c05nd

1 Purpose

c05nd is a comprehensive reverse communication function to find a solution of a system of nonlinear equations by a modification of the Powell hybrid method.

2 Syntax

```
[irevcm, x, fvec, diag, fjac, r, qtf, w, ifail] = c05nd(irevcm, x, fvec,
ml, mu, diag, mode, fjac, r, qtf, w, 'n', n, 'xtol', xtol, 'epsfcn',
epsfcn, 'factor', factor, 'lr', lr)
```

3 Description

The system of equations is defined as:

$$f_i(x_1, x_2, \dots, x_n) = 0, \quad i = 1, 2, \dots, n.$$

c05nd is based on the MINPACK routine HYBRD (see Moré *et al.* 1980). It chooses the correction at each step as a convex combination of the Newton and scaled gradient directions. Under reasonable conditions this guarantees global convergence for starting points far from the solution and a fast rate of convergence. The Jacobian is updated by the rank-1 method of Broyden. At the starting point the Jacobian is approximated by forward differences, but these are not used again until the rank-1 method fails to produce satisfactory progress. For more details see Powell 1970.

4 References

Moré J J, Garbow B S and Hillstom K E 1980 User guide for MINPACK-1 *Technical Report ANL-80-74* Argonne National Laboratory

Powell M J D 1970 A hybrid method for nonlinear algebraic equations *Numerical Methods for Nonlinear Algebraic Equations* (ed P Rabinowitz) Gordon and Breach

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 **parameter IREVCN**. Between intermediate exits and re-entries, **all parameters other than fvec must remain unchanged**.

5.1 Compulsory Input Parameters

1: **irevcm** – int32 scalar

On initial entry: must have the value 0.

Constraint: **irevcm** = 0, 1 or 2.

2: **x(n)** – double array

On initial entry: an initial guess at the solution vector.

3: **fvec(n)** – double array

On initial entry: need not be set.

On intermediate re-entry: if **irevcm** ≠ 2, **fvec** must not be changed.

If **irevcm** = 2, **fvec** must be set to the values of the functions computed at the current point **x**.

4: **ml – int32 scalar**

On initial entry: the number of subdiagonals within the band of the Jacobian matrix. (If the Jacobian is not banded, or you are unsure, set **ml** = **n** – 1.)

Constraint: **ml** \geq 0.

5: **mu – int32 scalar**

On initial entry: the number of superdiagonals within the band of the Jacobian matrix. (If the Jacobian is not banded, or you are unsure, set **mu** = **n** – 1.)

Constraint: **mu** \geq 0.

6: **diag(n) – double array**

On initial entry: if **mode** = 2, **diag** must contain multiplicative scale factors for the variables.

Constraint: **diag**(*i*) > 0.0, for *i* = 1, 2, ..., *n*.

7: **mode – int32 scalar**

On initial entry: indicates whether or not you have provided scaling factors in **diag**. If **mode** = 2 the scale factors must be supplied in **diag**. Otherwise, the variables will be scaled internally.

8: **fjac(ldfjac,n) – double array**

ldfjac, the first dimension of the array, must be at least **n**.

On initial entry: need not be set.

9: **r(lr) – double array**

On initial entry: need not be set.

10: **qtf(n) – double array**

On initial entry: need not be set.

11: **w(n,4) – double array****5.2 Optional Input Parameters**1: **n – int32 scalar**

Default: The dimension of the arrays **x**, **fvec**, **diag**, **fjac**, **qtf**, **w**. (An error is raised if these dimensions are not equal.)

On initial entry: *n*, the number of equations.

Constraint: **n** > 0.

2: **xtol – double scalar**

On initial entry: the accuracy in **x** to which the solution is required.

Suggested value: the square root of the *machine precision*.

Default: $\sqrt{}$

Constraint: **xtol** \geq 0.0.

3: **epsfcn – double scalar**

On initial entry: the order of the largest relative error in the functions. It is used in determining a suitable step for a forward difference approximation to the Jacobian. If **epsfcn** is less than *machine precision* then *machine precision* is used. Consequently a value of 0.0 will often be suitable.

Suggested value: **epsfcn** = 0.0.

Default: 0.0

4: **factor – double scalar**

On initial entry: a quantity to be used in determining the initial step bound. In most cases, **factor** should lie between 0.1 and 100.0. (The step bound is $\mathbf{factor} \times \|\mathbf{diag} \times \mathbf{x}\|_2$ if this is nonzero; otherwise the bound is **factor**.)

Suggested value: **factor** = 100.0.

Default: 100.0

Constraint: **factor** > 0.0.

5: **lr – int32 scalar**

Default: The dimension of the array **r**.

On initial entry:

Constraint: **lr** $\geq \mathbf{n} \times (\mathbf{n} + 1)/2$.

5.3 Input Parameters Omitted from the MATLAB Interface

ldfjac

5.4 Output Parameters

1: **irevcn – int32 scalar**

On intermediate exit: specifies what action you must take before re-entering c05nd with **irevcn** **unchanged**. The value of **irevcn** should be interpreted as follows:

irevcn = 1

Indicates the start of a new iteration. No action is required by you, but **x** and **fvec** are available for printing.

irevcn = 2

Indicates that before re-entry to c05nd, **fvec** must contain the function values $f_i(x)$.

On final exit: **irevcn** = 0, and the algorithm has terminated.

2: **x(n) – double array**

On intermediate exit: contains the current point.

On final exit: the final estimate of the solution vector.

3: **fvec(n) – double array**

On final exit: the function values at the final point, **x**.

4: **diag(n) – double array**

On intermediate exit: the scale factors actually used (computed internally if **mode** \neq 2).

5: **fjac(ldfjac,n) – double array**

On intermediate exit: should not be changed.

On final exit: the orthogonal matrix Q produced by the QR factorization of the final approximate Jacobian.

6: **r(lr)** – double array

On intermediate exit: should not be changed.

On final exit: the upper triangular matrix R produced by the QR factorization of the final approximate Jacobian, stored row-wise.

7: **qtf(n)** – double array

On intermediate exit: should not be changed.

On final exit: the vector $Q^T f$.

8: **w(n,4)** – double array

9: **ifail** – int32 scalar

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

On entry, **n** ≤ 0 ,
 or **xtol** < 0.0 ,
 or **ml** < 0 ,
 or **mu** < 0 ,
 or **factor** ≤ 0.0 ,
 or **ldfjac** $< \mathbf{n}$,
 or **lr** $< \mathbf{n} \times (\mathbf{n} + 1)/2$,
 or **mode** = 2 and **diag**(i) ≤ 0.0 for some i , $i = 1, 2, \dots, \mathbf{n}$.

ifail = 2

On entry, **irevcm** < 0 or **irevcm** > 2 .

ifail = 3

No further improvement in the approximate solution \mathbf{x} is possible; **xtol** is too small.

ifail = 4

The iteration is not making good progress, as measured by the improvement from the last five Jacobian evaluations.

ifail = 5

The iteration is not making good progress, as measured by the improvement from the last 10 iterations.

The values **ifail** = 4 and **ifail** = 5 may indicate that the system does not have a zero, or that the solution is very close to the origin (see Section 7). Otherwise, rerunning c05nd from a different starting point may avoid the region of difficulty.

7 Accuracy

If \hat{x} is the true solution and D denotes the diagonal matrix whose entries are defined by the array **diag**, then c05nd tries to ensure that

$$\|D(x - \hat{x})\|_2 \leq \mathbf{xtol} \times \|D\hat{x}\|_2.$$

If this condition is satisfied with **xtol** = 10^{-k} , then the larger components of Dx have k significant decimal

digits. There is a danger that the smaller components of Dx may have large relative errors, but the fast rate of convergence of c05nd usually avoids this possibility.

If **xtol** is less than *machine precision* and the above test is satisfied with the *machine precision* in place of **xtol**, then the function exits with **ifail** = 3.

Note: this convergence test is based purely on relative error, and may not indicate convergence if the solution is very close to the origin.

The test assumes that the functions are reasonably well behaved. If this condition is not satisfied, then c05nd may incorrectly indicate convergence. The validity of the answer can be checked, for example, by rerunning c05nd with a tighter tolerance.

8 Further Comments

The time required by c05nd to solve a given problem depends on n , the behaviour of the functions, the accuracy requested and the starting point. The number of arithmetic operations executed by c05nd to process the evaluation of functions in the main program in each exit is about $11.5 \times n^2$. The timing of c05nd will be strongly influenced by the time spent in the evaluation of the functions.

Ideally the problem should be scaled so that, at the solution, the function values are of comparable magnitude.

The number of function evaluations required to evaluate the Jacobian may be reduced if you can specify **ml** and **mu**.

9 Example

```
irevcm = int32(0);
x = [-1; -1; -1; -1; -1; -1; -1; -1; -1];
fvec = zeros(9,1);
ml = int32(1);
mu = int32(1);
diag = [1; 1; 1; 1; 1; 1; 1; 1; 1];
mode = int32(2);
fjac = zeros(9,9);
r = zeros(45,1);
qtf = zeros(9,1);
w = zeros(9, 4);
[irevcm, x, fvec, diag, fjac, r, qtf, w, ifail] = ...
    c05nd(irevcm, x, fvec, ml, mu, diag, mode, fjac, r, qtf, w);
while (irevcm > 0)
    if (irevcm == 2)
        for i=1:9
            fvec(i) = (3-2*x(i))*x(i)+1;
            if (i > 1)
                fvec(i) = fvec(i) - x(i-1);
            end
            if (i < 9)
                fvec(i) = fvec(i) - 2*x(i+1);
            end
        end
    end
    [irevcm, x, fvec, diag, fjac, r, qtf, w, ifail] = ...
        c05nd(irevcm, x, fvec, ml, mu, diag, mode, fjac, r, qtf, w);
end
x
```

```
x =
    -0.5707
    -0.6816
    -0.7017
    -0.7042
    -0.7014
```

-0.6919
-0.6658
-0.5960
-0.4164
