NAG Toolbox for MATLAB

c05nc

1 Purpose

c05nc is a comprehensive function to find a solution of a system of nonlinear equations by a modification of the Powell hybrid method.

2 Syntax

```
[x, fvec, diag, nfev, fjac, r, qtf, ifail] = c05nc(fcn, x, ml, mu, diag,
mode, nprint, lr, 'n', n, 'xtol', xtol, 'maxfev', maxfev, 'epsfcn',
epsfcn, 'factor', factor)
```

3 Description

The system of equations is defined as:

$$f_i(x_1, x_2, \ldots, x_n) = 0, \qquad \text{for } i = 1, 2, \ldots, n.$$

c05nc 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 Hillstrom 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

5.1 Compulsory Input Parameters

1: **fcn – string containing name of m-file**

fcn must return the values of the functions f_i at a point x.

Its specification is:

```
[fvec, iflag] = fcn(n, x, fvec, iflag)
```

Input Parameters

1: **n – int32 scalar**

n, the number of equations.

2: **x(n) – double array**

The components of the point x at which the functions must be evaluated.

3: **fvec(n) – double array**

If **iflag** $= 0$, **fvec** contains the function values $f_i(x)$ and must not be changed.

If **iflag** > 0 on entry, **fvec** must contain the function values $f_i(x)$ (unless **iflag** is set to a negative value by **fcn**).

4: **iflag – int32 scalar**

iflag ≥ 0.

iflag $= 0$

x and **fvec** are available for printing (see **nprint** below).

iflag > 0

fvec must be updated.

In general **iflag** should not be reset by **fcn**. If, however, you wish to terminate execution (perhaps because some illegal point **x** has been reached), then **iflag** should be set to a negative integer. This value will be returned through **ifail**.

Output Parameters

1: **fvec(n) – double array**

If **iflag** $= 0$, **fvec** contains the function values $f_i(x)$ and must not be changed.

If **iflag** > 0 on entry, **fvec** must contain the function values $f_i(x)$ (unless **iflag** is set to a negative value by **fcn**).

2: **iflag – int32 scalar**

iflag ≥ 0.

iflag $= 0$

x and **fvec** are available for printing (see **nprint** below).

iflag > 0

fvec must be updated.

In general **iflag** should not be reset by **fcn**. If, however, you wish to terminate execution (perhaps because some illegal point **x** has been reached), then **iflag** should be set to a negative integer. This value will be returned through **ifail**.

2: **x(n) – double array**

An initial guess at the solution vector.

3: **ml – int32 scalar**

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** ≥ 0.

4: **mu – int32 scalar**

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** ≥ 0.

5: **diag(n) – double array**

If **mode** $= 2$, **diag** must contain multiplicative scale factors for the variables.

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

6: **mode – int32 scalar**

Indicates whether or not you have provided scaling factors in **diag**. If **mode** $= 2$ the scaling must have been specified in **diag**. Otherwise, the variables will be scaled internally.

7: **nprint – int32 scalar**

Indicates whether special calls to the user-supplied (sub)program **fcn**, with **iflag** set to *0*, are to be made for printing purposes.

nprint ≤ 0

No calls are made.

nprint > 0

user-supplied (sub)program **fcn** is called at the beginning of the first iteration, every **nprint** iterations thereafter and immediately prior to the return from c05nc.

8: **lr – int32 scalar**

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

5.2 Optional Input Parameters

1: **n – int32 scalar**

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

n, the number of equations.

Constraint: **n** > 0.

2: **xtol – double scalar**

The accuracy in **x** to which the solution is required.

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

Default: $\sqrt{}$

Constraint: **xtol** ≥ 0.0.

3: **maxfev – int32 scalar**

The maximum number of calls to user-supplied (sub)program **fcn** with **iflag** $\neq 0$. c05nc will exit with **ifail** $= 2$, if, at the end of an iteration, the number of calls to **fcn** exceeds **maxfev**.

Suggested value: **maxfev** $= 200 \times$ (**n** $+ 1)$.

Default: $200 \times$ (**n** $+ 1)$

Constraint: **maxfev** > 0.

4: **epsfcn – double scalar**

A rough estimate 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

5: **factor – double scalar**

Must specify 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: $\mathbf{factor} = 100.0$.

Default: 100.0

Constraint: $\mathbf{factor} > 0.0$.

5.3 Input Parameters Omitted from the MATLAB Interface

ldfjac, w

5.4 Output Parameters

1: **x(n) – double array**

The final estimate of the solution vector.

2: **fvec(n) – double array**

The function values at the final point, **x**.

3: **diag(n) – double array**

The scale factors actually used (computed internally if $\mathbf{mode} \neq 2$).

4: **nfev – int32 scalar**

The number of calls made to user-supplied (sub)program **fcn**.

5: **fjac(ldfjac,n) – double array**

The orthogonal matrix Q produced by the QR factorization of the final approximate Jacobian.

6: **r(lr) – double array**

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

7: **qtf(n) – double array**

The vector $Q^{\mathrm{T}}f$.

8: **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:

$\mathbf{ifail} < 0$

This indicates an exit from c05nc because you have set **iflag** negative in the user-supplied (sub)program **fcn**. The value of **ifail** will be the same as your setting of **iflag**.

ifail $= 1$

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

ifail $= 2$

There have been at least **maxfev** evaluations of user-supplied (sub)program **fcn**. Consider restarting the calculation from the final point held in **x**.

ifail $= 3$

No further improvement in the approximate solution **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 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 c05nc 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 c05nc 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 c05nc 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 c05nc may incorrectly indicate convergence. The validity of the answer can be checked, for example, by rerunning c05nc with a tighter tolerance.

8 Further Comments

The time required by c05nc 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 c05nc to process each call of user-supplied (sub)program **fcn** is about $11.5 \times n^2$. Unless **fcn** can be evaluated quickly, the timing of c05nc will be strongly influenced by the time spent in **fcn**.

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

```
c05nc_fcn.m
```

```
function [fvec, iflag] = c05nc_fcn(n,x,fvec,iflag)
  for k = 1:n
    fvec(k) = (3.0-2.0*x(k))*x(k)+1.0;
    if k > 1
      fvec(k) = fvec(k) - x(k-1);
    end
    if k < n
      fvec(k) = fvec(k) - 2*x(k+1);
    end
  end
```

```
x = [-1;
     -1;
     -1;
     -1;
     -1;
     -1;
     -1;
     -1;
     -1];
ml = int32(1);
mu = int32(1);
diag = [1;
     1;
     1;
     1;
     1;
     1;
     1;
     1;
     1];
mode = int32(2);
nprint = int32(0);
lr = int32(45);
[xOut, fvec, diagOut, nfev, fjac, r, qtf, ifail] = c05nc('c05nc_fcn', x,
...
     ml, mu, diag, mode, nprint, lr)
```

```
xOut =
   -0.5707
   -0.6816
   -0.7017
   -0.7042
   -0.7014
   -0.6919
   -0.6658
   -0.5960
   -0.4164
fvec =
   1.0e-08 *
    0.6560
   -0.4175
   -0.5193
   -0.2396
    0.2022
    0.4818
    0.2580
   -0.3884
   -0.0136
diagOut =
     1
     1
     1
     1
     1
     1
     1
     1
     1
nfev =
          14
fjac =
  Columns 1 through 7
   -0.9691   -0.2148   -0.0209    0.0470    0.0611    0.0339   -0.0188
    0.2268   -0.9561   -0.1558    0.0078    0.0423    0.0328   -0.0064
   -0.0174    0.1584   -0.9720   -0.1533   -0.0206    0.0090    0.0087
   -0.0478   -0.0482    0.1554   -0.9653   -0.1728   -0.0306    0.0139
   -0.0414   -0.0486   -0.0103    0.1910   -0.9579   -0.1638   -0.0068
   -0.0072   -0.0143   -0.0058    0.0102    0.1882   -0.9588   -0.1373
    0.0361    0.0408    0.0197    0.0011   -0.0011    0.1731   -0.9455
    0.0591    0.0849    0.0606    0.0302    0.0175    0.0320    0.2366
    0.0164    0.0304    0.0474    0.0704    0.1040    0.1402    0.1748
  Columns 8 through 9
   -0.0678   -0.0471
   -0.0621   -0.0582
   -0.0254   -0.0720
    0.0072   -0.0918
    0.0214   -0.1198
   -0.0024   -0.1614
   -0.1429   -0.2288
   -0.8900   -0.3679
    0.4218   -0.8676
r =
   -5.9002
    4.1274
   -0.4204
   -0.4522
   -0.5225
   -0.3437
   -0.0653
    0.0185
   -0.5519
   -6.0563
    3.2570
   -0.5570
   -0.3043
   -0.2274
   -0.0452
    0.1224
   -0.3490
   -6.4766
    3.2423
   -0.2317
   -0.0327
   -0.0923
    0.0258
    0.0501
   -6.3881
    3.5140
   -0.1806
   -0.1332
   -0.1071
    0.4337
   -6.2931
    3.4096
   -0.4194
   -0.1511
    0.7698
   -6.4309
    3.1398
   -0.3188
    0.9972
   -6.5080
    3.4679
    0.7938
   -6.0582
    4.8522
   -3.8733
qtf =
   1.0e-07 *
   -0.3574
    0.0823
    0.2291
    0.1836
    0.0005
   -0.2262
   -0.2399
    0.1980
    0.0074
ifail =
           0
```