NAG Toolbox for MATLAB

c05pd

1 Purpose

c05pd is a comprehensive reverse communication function to find a solution of a system of nonlinear equations by a modification of the Powell hybrid method. You must provide the Jacobian.

2 Syntax

```
[irevcm, x, fvec, fjac, diag, r, qtf, w, ifail] = c05pd(irevcm, x, fvec,
fjac, diag, mode, r, qtf, w, 'n', n, 'xtol', xtol, 'factor', factor,
'lr', lr)
```

3 Description

The system of equations is defined as:

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

c05pd is based on the MINPACK routine HYBRJ (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. The Jacobian is requested to be supplied at the start of the computations, but it is not requested again. 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

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

5.1 Compulsory Input Parameters

1: **irevcm – int32 scalar**

On initial entry: must have the value 0.

Constraint: **irevcm** $= 0$, 1, 2 or 3.

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** $\neq 2$, **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: **fjac(ldfjac,n) – double array**

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

On initial entry: must be set to the values of the Jacobian evaluated at the initial point **x**.

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

On intermediate re-entry: if $\mathbf{irevcm} \neq 3$, **fjac** must not be changed.

If $\mathbf{irevcm} = 3$, **fjac** must be set to the value of the Jacobian computed at the current point **x**.

5: **diag(n) – double array**

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

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

6: **mode – int32 scalar**

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

7: **r(lr) – double array**

On initial entry: need not be set.

8: **qtf(n) – double array**

On initial entry: need not be set.

9: **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: $\mathbf{n} > 0$.

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: $\mathbf{xtol} \geq 0.0$.

3: **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: $\mathbf{factor} = 100.0$.

Default: 100.0

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

4: **lr – int32 scalar**

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

On initial entry:

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

5.3 Input Parameters Omitted from the MATLAB Interface

ldfjac

5.4 Output Parameters

1: **irevcm – int32 scalar**

On intermediate exit: specifies what action you must take before re-entering c05pd **with irevcm unchanged**. The value of **irevcm** should be interpreted as follows:

$\mathbf{irevcm} = 1$

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

$\mathbf{irevcm} = 2$

Indicates that before re-entry to c05pd, **fvec** must contain the function value $f_i(x)$.

$\mathbf{irevcm} = 3$

Indicates that before re-entry to c05pd, $\mathbf{fjac}(i,j)$ must contain the value of $\frac{\partial f_i}{\partial x_j}$ at the point x, for $i,j = 1, 2, \ldots, n$.

On final exit: $\mathbf{irevcm} = 0$, and the algorithm has terminated.

2: $\mathbf{x}(\mathbf{n})$ **– double array**

On intermediate exit: contains the current point.

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

3: $\mathbf{fvec}(\mathbf{n})$ **– double array**

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

4: $\mathbf{fjac}(\mathbf{ldfjac},\mathbf{n})$ **– double array**

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

5: $\mathbf{diag}(\mathbf{n})$ **– double array**

On intermediate exit: the scale factors actually used (computed internally if $\mathbf{mode} \neq 2$).

6: $\mathbf{r}(\mathbf{lr})$ **– 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: $\mathbf{qtf}(\mathbf{n})$ **– double array**

On intermediate exit: should not be changed.

On final exit: the vector $Q^{\mathrm{T}}f$.

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 **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 $= 2$

On entry, **irevcm** < 0 or **irevcm** > 3.

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 **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 c05pd 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 c05pd 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 c05pd 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 and the Jacobian are coded consistently and that the functions are reasonably well behaved. If these conditions are not satisfied, then c05pd may incorrectly indicate convergence. The coding of the Jacobian can be checked using c05za. If the Jacobian is coded correctly, then the validity of the answer can be checked by rerunning c05pd with a tighter tolerance.

8 Further Comments

The time required by c05pd 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 c05pd is about $11.5 \times n^2$ to process each evaluation of the functions and about $1.3 \times n^3$ to process each evaluation of the Jacobian. The timing of c05pd is strongly influenced by the time spent in the evaluation of the functions and the Jacobian.

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

9 Example

```
irevcm = int32(0);
x = [-1;
     -1;
     -1;
     -1;
     -1;
     -1;
     -1;
     -1;
     -1];
fvec = zeros(9, 1);
fjac = zeros(9,9);
diag = [1; 1; 1; 1; 1; 1; 1; 1; 1];
mode = int32(2);
r = zeros(45,1);
qtf = zeros(9,1);
w = zeros(9, 4);
lwsav = false(5, 1);
iwsav = zeros(15, 1, 'int32');
rwsav = zeros(10, 1);
[irevcm, x, fvec, fjac, diag, r, qtf, w, lwsav, iwsav, rwsav, ifail] =
...
       c05pd(irevcm, x, fvec, fjac, diag, mode, r, qtf, w, lwsav, iwsav,
rwsav);
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, fjac, diag, r, qtf, w, lwsav, iwsav, rwsav, ifail] =
...
       c05pd(irevcm, x, fvec, fjac, diag, mode, r, qtf, w, lwsav, iwsav,
rwsav);
end
  x
```

```
x =
   -0.5707
   -0.6816
   -0.7017
   -0.7042
   -0.7014
   -0.6919
   -0.6658
   -0.5960
   -0.4164
```