NAG Library Routine Document

c05mdf

1 Purpose

c05mdf is a comprehensive reverse communication routine that finds a solution of a system of nonlinear equations by fixed-point iteration using Anderson acceleration.

2 Specification

Fortran Interface

```
Subroutine c05mdf (irevcm, n, x, fvec, atol, rtol, m, cndtol, astart,      &
                   iwsav, rwsav, ifail)
Integer, Intent (In)                 :: n, m, astart
Integer, Intent (Inout)              :: irevcm, iwsav(14+m), ifail
Real (Kind=nag_wp), Intent (In)      :: atol, rtol, cndtol
Real (Kind=nag_wp), Intent (Inout)   :: x(n), fvec(n),                     &
                                        rwsav(2*m*n+m*m+m+2*n+1+min(m,1)   &
                                        *max(n,3*m))
```

3 Description

The system of equations is defined as:

$$f_k(x_1, x_2, \ldots, x_n) = 0, \quad k = 1, 2, \ldots, n.$$

This homogeneous system can readily be reformulated as

$$g(x) = x, \quad x \in R^n.$$

A standard fixed-point iteration approach is to start with an approximate solution $\hat{x}_0$ and repeatedly apply the function g until possible convergence; i.e., $\hat{x}_{i+1} = g(\hat{x}_i)$, until $\|\hat{x}_{i+1} - \hat{x}_i\| < \text{tol}$. Anderson acceleration uses up to m previous values of $\hat{x}$ to obtain an improved estimate $\hat{x}_{i+1}$. If a standard fixed-point iteration converges, then Anderson acceleration usually results in convergence in far fewer iterations (and therefore using far fewer function evaluations).

Full details of Anderson acceleration are provided in Anderson (1965). In summary, the previous m iterates are combined to form a succession of least squares problems. These are solved using a QR decomposition, which is updated at each iteration.

You are free to choose any value for m, provided $m \leq n$. A typical choice is $m = 4$.

4 References

Anderson D G (1965) Iterative Procedures for Nonlinear Integral Equations *J. Assoc. Comput. Mach.* **12** 547–560

5 Arguments

Note: this routine uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the argument **irevcm**. Between intermediate exits and re-entries, **all arguments other than fvec must remain unchanged**.

1: **irevcm** – Integer *Input/Output*

On initial entry: must have the value 0.

On intermediate exit: specifies what action you must take before re-entering **c05mdf** with **irevcm unchanged**. The value of **irevcm** should be interpreted as follows:

irevcm $= 1$

Indicates the start of a new iteration. No action is required by you, but **x** and **fvec** are available for printing, and a limit on the number of iterations can be applied.

irevcm $= 2$

Indicates that before re-entry to **c05mdf**, **fvec** must contain the function values $f(\hat{x}_i)$.

On final exit: **irevcm** $= 0$ and the algorithm has terminated.

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

Note: any values you return to **c05mdf** as part of the reverse communication procedure should not include floating-point NaN (Not a Number) or infinity values, since these are not handled by **c05mdf**. If your code inadvertently **does** return any NaNs or infinities, **c05mdf** is likely to produce unexpected results.

2: **n** – Integer *Input*

On entry: n, the number of equations.

Constraint: **n** > 0.

3: **x**(**n**) – Real (Kind=nag_wp) array *Input/Output*

On initial entry: an initial guess at the solution vector, $\hat{x}_0$.

On intermediate exit: contains the current point.

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

4: **fvec**(**n**) – Real (Kind=nag_wp) array *Input/Output*

On initial entry: need not be set.

On intermediate re-entry: if **irevcm** $= 1$, **fvec** must not be changed.

If **irevcm** $= 2$, **fvec** must be set to the values of the functions computed at the current point **x**, $f(\hat{x}_i)$.

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

5: **atol** – Real (Kind=nag_wp) *Input*

On initial entry: the absolute convergence criterion; see below.

Suggested value: $\sqrt{\epsilon}$, where ϵ is the ***machine precision*** returned by **x02ajf**.

Constraint: **atol** ≥ 0.0.

6: **rtol** – Real (Kind=nag_wp) *Input*

On initial entry: the relative convergence criterion. At each iteration $\|f(\hat{x}_i)\|$ is computed. The iteration is deemed to have converged if $\|f(\hat{x}_i)\| \leq \max(\mathbf{atol}, \mathbf{rtol} \times \|f(\hat{x}_0)\|)$.

Suggested value: $\sqrt{\epsilon}$, where ϵ is the ***machine precision*** returned by **x02ajf**.

Constraint: **rtol** ≥ 0.0.

7: **m** – Integer *Input*

On initial entry: m, the number of previous iterates to use in Anderson acceleration. If $m = 0$, Anderson acceleration is not used.

Suggested value: **m** $= 4$.

Constraint: $0 \leq$ **m** $\leq$ **n**.

8: **cndtol** – Real (Kind=nag_wp) *Input*

On initial entry: the maximum allowable condition number for the triangular *QR* factor generated during Anderson acceleration. At each iteration, if the condition number exceeds **cndtol**, columns are deleted until it is sufficiently small.

If $\mathbf{cndtol} = 0.0$, no condition number tests are performed.

Suggested value: $\mathbf{cndtol} = 0.0$. If condition number tests are required, a suggested value is $\mathbf{cndtol} = 1.0/\sqrt{\epsilon}$.

Constraint: $\mathbf{cndtol} \geq 0.0$.

9: **astart** – Integer *Input*

On initial entry: the number of iterations by which to delay the start of Anderson acceleration.

Suggested value: $\mathbf{astart} = 0$.

Constraint: $\mathbf{astart} \geq 0$.

10: $\mathbf{iwsav}(14 + \mathbf{m})$ – Integer array *Communication Array*

11: $\mathbf{rwsav}(2 \times \mathbf{m} \times \mathbf{n} + \mathbf{m}^2 + \mathbf{m} + 2 \times \mathbf{n} + 1 + \min(\mathbf{m}, 1) \times \max(\mathbf{n}, 3 \times \mathbf{m}))$
– Real (Kind=nag_wp) array *Communication Array*

The arrays **iwsav** and **rwsav must not** be altered between calls to **c05mdf**.

The size of **rwsav** is bounded above by $3 \times \mathbf{n} \times (\mathbf{m} + 2) + 1$.

12: **ifail** – Integer *Input/Output*

On initial entry: **ifail** must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, because for this routine the values of the output arguments may be useful even if $\mathbf{ifail} \neq 0$ on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of ifail on exit.**

On final exit: $\mathbf{ifail} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry $\mathbf{ifail} = 0$ or -1, explanatory error messages are output on the current error message unit (as defined by **x04aaf**).

Errors or warnings detected by the routine:

$\mathbf{ifail} = 1$

On initial entry, $\mathbf{irevcm} = \langle\mathit{value}\rangle$.
Constraint: $\mathbf{irevcm} = 0$.

On intermediate entry, $\mathbf{irevcm} = \langle\mathit{value}\rangle$.
Constraint: $\mathbf{irevcm} = 1$ or 2.

$\mathbf{ifail} = 2$

On entry, $\mathbf{n} = \langle\mathit{value}\rangle$.
Constraint: $\mathbf{n} > 0$.

ifail $= 3$

On entry, **atol** $= \langle$*value*$\rangle$.
Constraint: **atol** ≥ 0.0.

ifail $= 4$

On entry, **rtol** $= \langle$*value*$\rangle$.
Constraint: **rtol** ≥ 0.0.

ifail $= 5$

On entry, **m** $= \langle$*value*$\rangle$ and **n** $= \langle$*value*$\rangle$.
Constraint: $0 \leq$ **m** $\leq$ **n**.

ifail $= 6$

On entry, **cndtol** $= \langle$*value*$\rangle$.
Constraint: **cndtol** ≥ 0.0.

ifail $= 7$

On entry, **astart** $= \langle$*value*$\rangle$.
Constraint: **astart** ≥ 0.

ifail $= 8$

An error occurred in evaluating the QR decomposition during Anderson acceleration. This may be due to slow convergence of the iteration. Try setting the value of **cndtol**. If condition number tests are already performed, try decreasing **cndtol**.

ifail $= 9$

The iteration is not making good progress, as measured by the reduction in the norm of $f(x)$ in the last $\langle$*value*$\rangle$ iterations.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

ifail $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

There are no theoretical guarantees of global or local convergence for Anderson acceleration. However, extensive numerical tests show that, in practice, Anderson acceleration leads to significant improvements over the underlying fixed-point methods (which may only converge linearly), and in some cases can even alleviate divergence.

At each iteration, **c05mdf** checks whether $\|f(\hat{x}_i)\| \leq \max(\mathbf{atol}, \mathbf{rtol} \times \|f(\hat{x}_0)\|)$. If the inequality is satisfied, then the iteration is deemed to have converged. The validity of the answer may be checked by inspecting the value of **fvec** on final exit from **c05mdf**.

8 Parallelism and Performance

c05mdf makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

During each iteration, Anderson acceleration updates the factors of a *QR* decomposition and uses the decomposition to solve a linear least squares problem. This involves an additional $O(mn)$ floating-point operations per iteration compared with the unaccelerated fixed-point iteration.

c05mdf does not count the number of iterations. Thus, it is up to you to add a limit on the number of iterations and check if this limit has been exceeded when **c05mdf** is called. This is illustrated in the example program below.

10 Example

This example determines the values $x_1, \ldots, x_4$ which satisfy the equations

$$\begin{aligned} \cos x_3 - x_1 &= 0, \\ \sqrt{1 - x_4^2} - x_2 &= 0, \\ \sin x_1 - x_3 &= 0, \\ x_2^2 - x_4 &= 0. \end{aligned}$$

10.1 Program Text

```
    Program c05mdfe

!     C05MDF Example Program Text

!     Mark 26.1 Release. NAG Copyright 2017.

!     .. Use Statements ..
      Use nag_library, Only: c05mdf, dnrm2, nag_wp, x02ajf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter              :: imax = 50, n = 4, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: atol, cndtol, fnorm, rtol
      Integer                         :: astart, i, icount, ifail, irevcm, m
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable :: fvec(:), rwsav(:), x(:)
      Integer, Allocatable            :: iwsav(:)
!     .. Intrinsic Procedures ..
      Intrinsic                       :: cos, max, min, sin, sqrt
!     .. Executable Statements ..

!     .. Executable Statements ..
      Write (nout,*) 'C05MDF Example Program Results'

      m = 2
      Allocate (fvec(n),iwsav(14+m),x(n),rwsav(2*m*n+m*m+m+2*n+1+min(m,        &
        1)*max(n,3*m)))

!     The following starting values provide a rough solution.

      x(1) = 2.0E0_nag_wp
      x(2) = 0.5E0_nag_wp
      x(3) = 2.0E0_nag_wp
      x(4) = 0.5E0_nag_wp

      atol = sqrt(x02ajf())
      rtol = sqrt(x02ajf())
      cndtol = 0.0_nag_wp
      astart = 0
      icount = 0
      irevcm = 0
      ifail = -1


revcomm: Do

        Call c05mdf(irevcm,n,x,fvec,atol,rtol,m,cndtol,astart,iwsav,rwsav,     &
          ifail)

        Select Case (irevcm)
        Case (1)

          If (icount==imax) Then
            Write (nout,*) 'Exiting after the maximum number of iterations'
            Exit revcomm
          End If

          icount = icount + 1

!         Insert print statements here to monitor progress if desired.

          Cycle revcomm
        Case (2)

!          Evaluate functions at given point

          fvec(1) = cos(x(3)) - x(1)
          fvec(2) = sqrt(1.0_nag_wp-x(4)**2) - x(2)
          fvec(3) = sin(x(1)) - x(3)
          fvec(4) = x(2)**2 - x(4)

          Cycle revcomm
        Case Default
          Exit revcomm
        End Select

      End Do revcomm

      If (ifail==0 .Or. icount==imax) Then
!         The NAG name equivalent of dnrm2 is f06ejf
        fnorm = dnrm2(n,fvec,1)
        Write (nout,*)
        Write (nout,99999) 'Final 2-norm of the residuals after', icount,      &
          ' iterations is ', fnorm
        Write (nout,*)
        Write (nout,*) 'Final approximate solution'
        Write (nout,*)
        Write (nout,99998)(x(i),i=1,n)
      End If

99999 Format (1X,A,I4,A,E12.4)
99998 Format (1X,4F12.4)
    End Program c05mdfe
```

10.2 Program Data

None.

10.3 Program Results

```
 C05MDF Example Program Results

 Final 2-norm of the residuals after  31 iterations is   0.2476E-07

 Final approximate solution

       0.7682      0.7862      0.6948      0.6180
```