

NAG Library Routine Document

C05ZDF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details.

1 Purpose

C05ZDF checks the user-supplied gradients of a set of nonlinear functions in several variables, for consistency with the functions themselves. The routine must be called twice.

2 Specification

```
SUBROUTINE C05ZDF (MODE, M, N, X, FVEC, FJAC, XP, FVECP, ERR, IFAIL)
INTEGER          MODE, M, N, IFAIL
REAL (KIND=nag_wp) X(N), FVEC(M), FJAC(M,N), XP(N), FVECP(M), ERR(M)
```

3 Description

C05ZDF is based on the MINPACK routine CHKDER (see Moré *et al.* (1980)). It checks the i th gradient for consistency with the i th function by computing a forward-difference approximation along a suitably chosen direction and comparing this approximation with the user-supplied gradient along the same direction. The principal characteristic of C05ZDF is its invariance under changes in scale of the variables or functions.

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

5 Parameters

- 1: MODE – INTEGER *Input*
On entry: the value 1 on the first call and the value 2 on the second call of C05ZDF.
Constraint: MODE = 1 or 2.
- 2: M – INTEGER *Input*
On entry: m , the number of functions.
Constraint: $M \geq 1$.
- 3: N – INTEGER *Input*
On entry: n , the number of variables. For use with C05RBF, C05RCF and C05RDF, $M = N$.
Constraint: $N \geq 1$.
- 4: X(N) – REAL (KIND=nag_wp) array *Input*
On entry: the components of a point x , at which the consistency check is to be made. (See Section 7.)

- 5: FVEC(M) – REAL (KIND=nag_wp) array Input
On entry: if MODE = 2, FVEC must contain the value of the functions evaluated at x . If MODE = 1, FVEC is not referenced.
- 6: FJAC(M,N) – REAL (KIND=nag_wp) array Input
On entry: if MODE = 2, FJAC must contain the value of $\frac{\partial f_i}{\partial x_j}$ at the point x , for $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$. If MODE = 1, FJAC is not referenced.
- 7: XP(N) – REAL (KIND=nag_wp) array Output
On exit: if MODE = 1, XP is set to a point neighbouring X. If MODE = 2, XP is undefined.
- 8: FVECP(M) – REAL (KIND=nag_wp) array Input
On entry: if MODE = 2, FVECP must contain the value of the functions evaluated at XP (as output by a preceding call to C05ZDF with MODE = 1). If MODE = 1, FVECP is not referenced.
- 9: ERR(M) – REAL (KIND=nag_wp) array Output
On exit: if MODE = 2, ERR contains measures of correctness of the respective gradients. If MODE = 1, ERR is undefined. If there is no loss of significance (see Section 7), then if ERR(i) is 1.0 the i th user-supplied gradient $\frac{\partial f_i}{\partial x_j}$, for $j = 1, 2, \dots, n$ is correct, whilst if ERR(i) is 0.0 the i th gradient is incorrect. For values of ERR(i) between 0.0 and 1.0 the categorisation is less certain. In general, a value of ERR(i) > 0.5 indicates that the i th gradient is probably correct.
- 10: IFAIL – INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**
On exit: 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 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:

IFAIL = 1

On entry, MODE \neq 1 or 2.

IFAIL = 2

On entry, M < 1.

IFAIL = 3

On entry, N < 1.

7 Accuracy

C05ZDF does not perform reliably if cancellation or rounding errors cause a severe loss of significance in the evaluation of a function. Therefore, none of the components of x should be unusually small (in particular, zero) or any other value which may cause loss of significance. The relative differences between corresponding elements of FVECP and FVEC should be at least two orders of magnitude greater than the *machine precision* returned by X02AJF.

8 Further Comments

The time required by C05ZDF increases with M and N.

9 Example

This example checks the Jacobian matrix for a problem with 15 functions of 3 variables (sometimes referred to as the Bard problem).

9.1 Program Text

```
! C05ZDF Example Program Text
! Mark 24 Release. NAG Copyright 2012.

Module c05zdfe_mod

! C05ZDF Example Program Module:
! Parameters and User-defined Routines

! .. Use Statements ..
Use nag_library, Only: nag_wp
! .. Implicit None Statement ..
Implicit None
! .. Parameters ..
Integer, Parameter :: m = 15, n = 3, nout = 6
Contains
Subroutine get_fvec(x,fvec)

! .. Array Arguments ..
Real (Kind=nag_wp), Intent (Out) :: fvec(:)
Real (Kind=nag_wp), Intent (In) :: x(:)
! .. Local Scalars ..
Real (Kind=nag_wp) :: u, v, w
Integer :: i
! .. Local Arrays ..
Real (Kind=nag_wp), Allocatable :: y(:)
! .. Intrinsic Procedures ..
Intrinsic :: min, real
! .. Executable Statements ..
Allocate (y(m))

y(1:m) = real((/14,18,22,25,29,32,35,39,47,58,73,96,134,210,439/), &
  kind=nag_wp)
y(1:m) = y(1:m)*0.01_nag_wp

Do i = 1, m
  u = real(i,kind=nag_wp)
  v = real(m+1-i,kind=nag_wp)
  w = min(u,v)
  fvec(i) = y(i) - (x(1)+u/(v*x(2)+w*x(3)))
End Do

Return
End Subroutine get_fvec
Subroutine get_fjac(x,fjac)

! .. Array Arguments ..
Real (Kind=nag_wp), Intent (Inout) :: fjac(:,:)
```

```

      Real (Kind=nag_wp), Intent (In)      :: x(:)
!    .. Local Scalars ..
      Real (Kind=nag_wp)                  :: denom, u, v, w
      Integer                              :: i
!    .. Intrinsic Procedures ..
      Intrinsic                            :: min, real
!    .. Executable Statements ..
      Do i = 1, m
         u = real(i,kind=nag_wp)
         v = real(m+1-i,kind=nag_wp)
         w = min(u,v)
         denom = (v*x(2)+w*x(3))**(-2)
         fjac(i,1:n) = (/ -1.0E0_nag_wp, u*v*denom, u*w*denom /)
      End Do

      Return
End Subroutine get_fjac
End Module c05zdfe_mod
Program c05zdfe

!    C05ZDF Example Main Program

!    .. Use Statements ..
Use nag_library, Only: c05zdf, nag_wp
Use c05zdfe_mod, Only: get_fjac, get_fvec, m, n, nout
!    .. Implicit None Statement ..
Implicit None
!    .. Local Scalars ..
Integer                              :: i, ifail, mode
!    .. Local Arrays ..
Real (Kind=nag_wp), Allocatable       :: err(:), fjac(:, :), fvec(:),      &
                                       fvecp(:), x(:), xp(:)
!    .. Intrinsic Procedures ..
Intrinsic                              :: any
!    .. Executable Statements ..
Write (nout,*) 'C05ZDF Example Program Results'

Allocate (err(m), fjac(m,n), fvec(m), fvecp(m), x(n), xp(n))

!    Point at which to check gradients:

x(1:n) = (/ 0.92_nag_wp, 0.13_nag_wp, 0.54_nag_wp /)

mode = 1

ifail = 0
Call c05zdf(mode, m, n, x, fvec, fjac, xp, fvecp, err, ifail)

Call get_fvec(x, fvec)

Call get_fvec(xp, fvecp)

Call get_fjac(x, fjac)

mode = 2

ifail = 0
Call c05zdf(mode, m, n, x, fvec, fjac, xp, fvecp, err, ifail)

Write (nout,*)
Write (nout,99999) 'At point ', (x(i), i=1, n), ', '

If (any(err(1:m) <= 0.5_nag_wp)) Then

   Do i = 1, m

      If (err(i) <= 0.5_nag_wp) Then
         Write (nout,99998) 'suspicious gradient number ', i, &
            ' with error measure ', err(i)
      End If
   End Do
End If

```

```
End Do

Else
  Write (nout,99997) 'gradients appear correct'
End If

99999 Format (1X,A,3F12.4,A)
99998 Format (1X,A,I5,A,F12.4)
99997 Format (1X,A)
End Program c05zdf
```

9.2 Program Data

None.

9.3 Program Results

C05ZDF Example Program Results

```
At point      0.9200      0.1300      0.5400,
gradients appear correct
```
