NAG Library Routine Document

C05ZAF

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

C05ZAF 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 C05ZAF (M, N, X, FVEC, FJAC, LDFJAC, XP, FVECP, MODE, ERR)

INTEGER            M, N, LDFJAC, MODE
REAL (KIND=nag_wp) X(N), FVEC(M), FJAC(LDFJAC,N), XP(*), FVECP(M), ERR(*)
```

3 Description

C05ZAF is based on the MINPACK routine CHKDER (see Moré *et al.* (1980)). It checks the ith gradient for consistency with the ith 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 C05ZAF is its invariance under changes in scale of the variables or functions.

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

5 Parameters

1: M – INTEGER *Input*

On entry: the number of functions.

2: N – INTEGER *Input*

On entry: the number of variables. For use with C05PBF/C05PBA and C05PCF/C05PCA, M = N.

3: 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 8.)

4: FVEC(M) – REAL (KIND=nag_wp) array *Input*

On entry: when MODE = 2, FVEC must contain the functions evaluated at x.

5: FJAC(LDFJAC,N) – REAL (KIND=nag_wp) array *Input*

On entry: when MODE = 2, FJAC must contain the user-supplied gradients. (The ith row of FJAC must contain the gradient of the ith function evaluated at the point x.)

6: LDFJAC – INTEGER *Input*

On entry: the first dimension of the array FJAC as declared in the (sub)program from which C05ZAF is called.

Constraint: LDFJAC $\geq$ M.

7: XP($*$) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array XP must be at least N if MODE $= 1$, and at least 1 otherwise.

On exit: when MODE $= 1$, XP is set to a neighbouring point to X.

8: FVECP(M) – REAL (KIND=nag_wp) array *Input*

On entry: when MODE $= 2$, FVECP must contain the functions evaluated at XP.

9: MODE – INTEGER *Input*

On entry: the value 1 on the first call and the value 2 on the second call of C05ZAF.

10: ERR($*$) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array ERR must be at least M if MODE $= 2$, and at least 1 otherwise.

On exit: when MODE $= 2$, ERR contains measures of correctness of the respective gradients. If there is no loss of significance (see Section 8), then if ERR(i) is 1.0 the ith user-supplied gradient is correct, whilst if ERR(i) is 0.0 the ith 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 ith gradient is probably correct.

6 Error Indicators and Warnings

If an error is detected in an input parameter C05ZAF will act as if a soft noisy exit has been requested (see Section 3.3.4 in the Essential Introduction).

7 Accuracy

See Section 8.

8 Further Comments

The time required by C05ZAF increases with M and N.

C05ZAF 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***.

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

```
!   C05ZAF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module c05zafe_mod

!     C05ZAF 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
      Integer, Parameter                  :: ldfjac = m
    Contains
      Subroutine get_fvec(m,n,x,fvec)

!       .. Scalar Arguments ..
        Integer, Intent (In)                :: m, n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out)    :: fvec(m)
        Real (Kind=nag_wp), Intent (In)     :: x(n)
!       .. 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(m,n,x,fjac,ldfjac)

!       .. Scalar Arguments ..
        Integer, Intent (In)                :: ldfjac, m, n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)  :: fjac(ldfjac,n)
        Real (Kind=nag_wp), Intent (In)     :: x(n)
!       .. 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) = (/-1.0_nag_wp,u*v*denom,u*w*denom/)
        End Do

        Return

      End Subroutine get_fjac
    End Module c05zafe_mod
    Program c05zafe

!     C05ZAF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: c05zaf, nag_wp
```

```
      Use c05zafe_mod, Only: get_fjac, get_fvec, ldfjac, m, n, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Integer                             :: i, mode
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable     :: err(:), fjac(:,:), fvec(:),      &
                                             fvecp(:), x(:), xp(:)
!     .. Intrinsic Procedures ..
      Intrinsic                           :: any
!     .. Executable Statements ..
      Write (nout,*) 'C05ZAF Example Program Results'

      Allocate (err(m),fjac(ldfjac,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

      Call c05zaf(m,n,x,fvec,fjac,ldfjac,xp,fvecp,mode,err)

      Call get_fvec(m,n,x,fvec)

      Call get_fvec(m,n,xp,fvecp)

      Call get_fjac(m,n,x,fjac,ldfjac)

      mode = 2

      Call c05zaf(m,n,x,fvec,fjac,ldfjac,xp,fvecp,mode,err)

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

      If (any(err(1:m)<=0.5E0_nag_wp)) Then

        Do i = 1, m

          If (err(i)<=0.5E0_nag_wp) Then
            Write (nout,99998) 'suspicious gradient number ', i, &
              ' with error measure ', err(i)
          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 c05zafe
```

9.2 Program Data

None.

9.3 Program Results

```
 C05ZAF Example Program Results

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