NAG Library Routine Document

C09EZF

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

C09EZF inserts a selected set of two-dimensional discrete wavelet transform (DWT) coefficients into the full set of coefficients stored in compact form, which may be later used as input to the multi-level reconstruction routine C09EDF.

2 Specification

```
SUBROUTINE C09EZF (ILEV, CINDEX, LENC, C, D, LDD, ICOMM, IFAIL)
INTEGER            ILEV, CINDEX, LENC, LDD, ICOMM(180), IFAIL
REAL (KIND=nag_wp) C(LENC), D(LDD,*)
```

3 Description

C09EZF inserts a selected set of two-dimensional DWT coefficients into the full set of coefficients stored in compact form in a one-dimensional array C. It is required that C09EZF is preceded by a call to the initialization routine C09ABF and the forward multi-level transform routine C09ECF.

Given an initial two-dimensional data set A, a prior call to C09ECF computes the approximation coefficients (at the highest requested level) and three sets of detail coeficients at all levels and stores these in compact form in a one-dimensional array C. C09EYF can then extract either the approximation coefficients or one of the sets of detail coefficients at one of the levels into a two-dimensional array, D. Following some calculation on this set of coefficients (for example, denoising), the updated coefficients in D are inserted back into the full set C using C09EZF. Several extractions and insertions may be performed at different levels. C09EDF can then be used to reconstruct a manipulated data set $\tilde{A}$. The dimensions of D depend on the level extracted and are available from the arrays DWTLVM and DWTLVN as returned by C09ECF which contain the first and second dimensions respectively. See Section 2.1 in the C09 Chapter Introduction for a discussion of the multi-level two-dimensional DWT.

4 References

None.

5 Arguments

Note: the following notation is used in this section:

n_{cm} is the number of wavelet coefficients in the first dimension, which, at level ILEV, is equal to DWTLVM(NWL − ILEV + 1) as returned by a call to C09ECF transforming NWL levels.

n_{cn} is the number of wavelet coefficients in the second dimension, which, at level ILEV, is equal to DWTLVN(NWL − ILEV + 1) as returned by a call to C09ECF transforming NWL levels.

1: ILEV – INTEGER *Input*

On entry: the level at which coefficients are to be inserted.

Constraints:

$1 \leq \mathrm{ILEV} \leq \mathrm{NWL}$, where NWL is as used in a preceding call to C09ECF;
if CINDEX = 0, ILEV = NWL.

2: CINDEX – INTEGER *Input*

On entry: identifies which coefficients to insert. The coefficients are identified as follows:

CINDEX $= 0$
: The approximation coefficients, produced by application of the low pass filter over columns and rows of the original matrix (LL). The approximation coefficients are present only for ILEV $=$ NWL, where NWL is the value used in a preceding call to C09ECF.

CINDEX $= 1$
: The vertical detail coefficients produced by applying the low pass filter over columns of the original matrix and the high pass filter over rows (LH).

CINDEX $= 2$
: The horizontal detail coefficients produced by applying the high pass filter over columns of the original matrix and the low pass filter over rows (HL).

CINDEX $= 3$
: The diagonal detail coefficients produced by applying the high pass filter over columns and rows of the original matrix (HH).

Constraint: $0 \leq$ CINDEX ≤ 3 when ILEV $=$ NWL as used in C09ECF, otherwise $1 \leq$ CINDEX ≤ 3.

3: LENC – INTEGER *Input*

On entry: the dimension of the array C as declared in the (sub)program from which C09EZF is called.

Constraint: LENC must be unchanged from the value used in the preceding call to C09ECF..

4: C(LENC) – REAL (KIND=nag_wp) array *Input/Output*

On entry: contains the DWT coefficients inserted by previous calls to C09EZF, or computed by a previous call to C09ECF.

On exit: contains the same DWT coefficients provided on entry except for those identified by ILEV and CINDEX, which are updated with the values supplied in D, inserted into the correct locations as expected by the reconstruction routine C09EDF.

5: D(LDD, $*$) – REAL (KIND=nag_wp) array *Input*

Note: the second dimension of the array D must be at least n_{cn}.

On entry: the coefficients to be inserted.

If ILEV $=$ NWL (as used in C09ECF) and CINDEX $= 0$, the n_{cm} by n_{cn} manipulated approximation coefficients a_{ij} must be stored in D(i,j), for $i = 1,2,\ldots,n_{\mathrm{cm}}$ and $i = 1,2,\ldots,n_{\mathrm{cn}}$.

Otherwise the n_{cm} by n_{cn} manipulated level ILEV detail coefficients (of type specified by CINDEX) d_{ij} must be stored in D(i,j), for $i = 1,2,\ldots,n_{\mathrm{cm}}$ and $j = 1,2,\ldots,n_{\mathrm{cn}}$.

6: LDD – INTEGER *Input*

On entry: the first dimension of the array D as declared in the (sub)program from which C09EZF is called.

Constraint: LDD $\geq n_{\mathrm{cm}}$.

7: ICOMM(180) – INTEGER array *Communication Array*

On entry: contains details of the discrete wavelet transform and the problem dimension as setup in the call to the initialization routine C09ABF.

8: IFAIL – INTEGER *Input/Output*

On 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, if you are not familiar with this argument, 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, ILEV $= \langle value \rangle$.
Constraint: ILEV ≥ 1.

On entry, ILEV $= \langle value \rangle$ and NWL $= \langle value \rangle$.
Constraint: ILEV $\leq$ NWL, where NWL is the number of levels used in the call to C09ECF.

IFAIL $= 2$

On entry, CINDEX $= \langle value \rangle$.
Constraint: CINDEX ≤ 3.

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

IFAIL $= 3$

On entry, LENC $= \langle value \rangle$ and $n_{\mathrm{ct}} = \langle value \rangle$.
Constraint: LENC $\geq n_{\mathrm{ct}}$, where n_{ct} is the number of DWT coefficients computed in a previous call to C09ECF.

IFAIL $= 4$

On entry, LDD $= \langle value \rangle$ and $n_{\mathrm{cm}} = \langle value \rangle$.
Constraint: LDD $\geq n_{\mathrm{cm}}$, where n_{cm} is the number of DWT coefficients in the first dimension at the selected level ILEV.

IFAIL $= 5$

On entry, ILEV $= \langle value \rangle$ and NWL $= \langle value \rangle$, but CINDEX $= 0$.
Constraint: CINDEX > 0 when ILEV $<$ NWL in the preceding call to C09ECF.

IFAIL $= 6$

Either the initialization routine has not been called first or ICOMM has been corrupted.

Either the initialization routine was called with WTRANS = 'S' or ICOMM has been corrupted.

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

Not applicable.

8 Parallelism and Performance

C09EZF is not threaded in any implementation.

9 Further Comments

None.

10 Example

The following example demonstrates using the coefficient extraction and insertion routines in order to apply denoising using a thresholding operation. The original input data, which is horizontally striped, has artificial noise introduced to it, taken from a normal random number distribution. Reconstruction then takes place on both the noisy data and denoised data. The Mean Square Errors (MSE) of the two reconstructions are printed along with the reconstruction of the denoised data. The MSE of the denoised reconstruction is less than that of the noisy reconstruction.

10.1 Program Text

```
    Program c09ezfe

!     C09EZF Example Program Text
!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: c09abf, c09ecf, c09edf, c09eyf, c09ezf, dnrm2,    &
                             nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: mse, thresh
      Integer                          :: cindex, denoised, i, ifail, ilev, j, &
                                          lda, ldb, ldd, lenc, m, n, nf, nwcn, &
                                          nwct, nwl
      Character (10)                   :: mode, wavnam, wtrans
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:), an(:,:), b(:,:), c(:),       &
                                          d(:,:), e(:,:)
      Integer, Allocatable             :: dwtlvm(:), dwtlvn(:)
      Integer                          :: icomm(180)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: abs, log, real, sqrt
!     .. Executable Statements ..
      Write (nout,*) 'C09EZF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
```

```
!     Read problem parameters
      Read (nin,*) m, n
      Read (nin,*) wavnam, mode
      Write (nout,99999) wavnam, mode, m, n

!     Allocate arrays to hold the original data, A, original data plus noise,
!     AN, reconstruction using denoised coefficients, B, and randomly
!     generated noise, X.
      lda = m
      ldb = m
      Allocate (a(lda,n),an(lda,n),b(ldb,n),e(m,n))

!     Read in the original data
      Do i = 1, m
        Read (nin,*) a(i,1:n)
      End Do

!     Output the original data
      Write (nout,99997)
      Do i = 1, m
        Write (nout,99998) a(i,1:n)
      End Do

!     Fill the array AN with the original data in A plus some noise
!     and return a VisuShrink denoising threshold, thresh.
      Call create_noise(a,an,lda,m,n,thresh)

!     Output the noisy data
      Write (nout,99996)
      Do i = 1, m
        Write (nout,99998) an(i,1:n)
      End Do

!     Query wavelet filter dimensions
!     For Multi-Resolution Analysis, decomposition, wtrans = 'M'
      wtrans = 'Multilevel'
      ifail = 0
      Call c09abf(wavnam,wtrans,mode,m,n,nwl,nf,nwct,nwcn,icomm,ifail)

!     Allocate arrays to hold the coefficients, C, and the dimensions
!     of the coefficients at each level, DWTLVM, DWTLVN
      lenc = nwct
      Allocate (c(lenc),dwtlvm(nwl),dwtlvn(nwl))

!     Perform a forwards multi-level transform on the noisy data
      ifail = 0
      Call c09ecf(m,n,an,lda,lenc,c,nwl,dwtlvm,dwtlvn,icomm,ifail)

!     Reconstruct without thresholding of detail coefficients
      ifail = 0
      Call c09edf(nwl,lenc,c,m,n,b,ldb,icomm,ifail)

!     Calculate the Mean Square Error of the noisy reconstruction
      e(:,:) = a(:,:) - b(:,:)
      mse = dnrm2(m*n,e,1)
      mse = mse**2
      mse = mse/real(m*n,kind=nag_wp)
      Write (nout,99995) mse

!     Now perform the denoising by extracting each of the detail
!     coefficients at each level and applying hard thresholding

!     Allocate a 2D array to hold the detail coefficients
      ldd = dwtlvm(nwl)
      Allocate (d(ldd,dwtlvn(nwl)))

      denoised = 0
!     For each level
      Do ilev = nwl, 1, -1

!       Select detail coefficients
```

```
        Do cindex = 1, 3

!         Extract coefficients into the 2D array D
          ifail = 0
          Call c09eyf(ilev,cindex,lenc,c,d,ldd,icomm,ifail)

!         Perform the hard thresholding operation
          Do j = 1, dwtlvn(nwl-ilev+1)
            Do i = 1, dwtlvm(nwl-ilev+1)
              If (abs(d(i,j))<thresh) Then
                d(i,j) = 0.0_nag_wp
                denoised = denoised + 1
              End If
            End Do
          End Do

!         Insert the denoised coefficients back into C
          ifail = 0
          Call c09ezf(ilev,cindex,lenc,c,d,ldd,icomm,ifail)

        End Do

      End Do

!     Output the number of coefficients that were set to zero
      Write (nout,99994) denoised, nwct - dwtlvm(1)*dwtlvn(1)

!     Reconstruct original data following thresholding of detail coefficients
      ifail = 0
      Call c09edf(nwl,lenc,c,m,n,b,ldb,icomm,ifail)

!     Calculate the Mean Square Error of the denoised reconstruction
      e(:,:) = a(:,:) - b(:,:)
      mse = dnrm2(m*n,e,1)
      mse = mse**2
      mse = mse/real(m*n,kind=nag_wp)
      Write (nout,99993) mse

!     Output the denoised reconstruction
      Write (nout,99992)
      Do i = 1, m
        Write (nout,99998) b(i,1:n)
      End Do

99999 Format (1X,' MLDWT :: Wavelet  : ',A,/,1X,'          End mode : ',A,/,   &
        1X,'          M        : ',I4,/,1X,'          N        : ',I4)
99998 Format (8(F8.4,1X),:)
99997 Format (/,1X,' Original data            A  : ')
99996 Format (/,1X,' Original data plus noise AN : ')
99995 Format (/,1X,' Without denoising Mean Square Error is ',F9.6)
99994 Format (/,1X,' Number of coefficients denoised is ',I3,' out of ',I3)
99993 Format (/,1X,' With denoising Mean Square Error is ',F9.6)
99992 Format (/,1X,' Reconstruction of denoised input D : ')

    Contains

!     Subroutine fills the output array AN with the data in A
!     plus some noise taken from a normal distribution, and
!     returns the VisuShrink denoising threshold, thresh.

      Subroutine create_noise(a,an,lda,m,n,thresh)

!       .. Use Statements ..
        Use nag_library, Only: g05kff, g05skf
!       .. Parameters ..
        Integer, Parameter             :: lseed = 1
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: thresh
        Integer, Intent (In)           :: lda, m, n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: a(lda,n)
```

```
        Real (Kind=nag_wp), Intent (Out) :: an(lda,n)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: var, xmu
        Integer                        :: genid, i, ifail, lstate, subid
!       .. Local Arrays ..
        Real (Kind=nag_wp), Allocatable :: x(:,:)
        Integer                        :: seed(lseed)
        Integer, Allocatable           :: state(:)
!       .. Executable Statements ..

!       Set up call to g05skf in order to create some random noise from
!       a normal distribution to add to the original data.
!       Initial call to RNG initializer to get size of STATE array
        seed(1) = 642521
        genid = 3
        subid = 0
        lstate = 0
        Allocate (state(lstate))
        ifail = 0
        Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

!       Reallocate STATE
        Deallocate (state)
        Allocate (state(lstate))

!       Initialize the generator to a repeatable sequence
        ifail = 0
        Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

!       Set the distribution parameters for the random noise.
        xmu = 0.0_nag_wp
        var = 0.1E-3_nag_wp

        Allocate (x(m,n))

!       Generate the noise variates
        ifail = 0
        Do i = 1, n
          Call g05skf(m,xmu,var,state,x(1,i),ifail)
        End Do

!       Add the noise to the original input and save in AN
        an(:,:) = a(:,:) + x(:,:)

!       Calculate the threshold based on VisuShrink denoising
        thresh = sqrt(var)*sqrt(2._nag_wp*log(real(m*n,kind=nag_wp)))

      End Subroutine create_noise

    End Program c09ezfe
```

10.2 Program Data

```
C09EZF Example Program Data
 7, 6   : m, n
 DB6 Period : wavnam, mode
0.01 0.01 0.01 0.01 0.01 0.01
1.00 1.00 1.00 1.00 1.00 1.00
0.01 0.01 0.01 0.01 0.01 0.01
1.00 1.00 1.00 1.00 1.00 1.00
0.01 0.01 0.01 0.01 0.01 0.01
1.00 1.00 1.00 1.00 1.00 1.00
0.01 0.01 0.01 0.01 0.01 0.01
```

10.3 Program Results

```
 C09EZF Example Program Results

  MLDWT :: Wavelet  : DB6
           End mode : Period
           M        :    7
           N        :    6

  Original data            A  :
  0.0100   0.0100   0.0100   0.0100   0.0100   0.0100
  1.0000   1.0000   1.0000   1.0000   1.0000   1.0000
  0.0100   0.0100   0.0100   0.0100   0.0100   0.0100
  1.0000   1.0000   1.0000   1.0000   1.0000   1.0000
  0.0100   0.0100   0.0100   0.0100   0.0100   0.0100
  1.0000   1.0000   1.0000   1.0000   1.0000   1.0000
  0.0100   0.0100   0.0100   0.0100   0.0100   0.0100

  Original data plus noise AN :
  0.0135   0.0170  -0.0049  -0.0009   0.0002   0.0123
  1.0015   0.9896   0.9983   1.0044   1.0097   0.9847
 -0.0017   0.0107   0.0194  -0.0084   0.0114  -0.0006
  0.9899   1.0038   1.0005   0.9921   0.9923   0.9982
 -0.0093   0.0149   0.0094   0.0160   0.0058   0.0257
  0.9842   1.0278   0.9991   0.9956   1.0113   0.9911
  0.0139  -0.0011   0.0180   0.0187   0.0106   0.0118

  Without denoising Mean Square Error is  0.000098

  Number of coefficients denoised is  32 out of  48

  With denoising Mean Square Error is  0.000018

  Reconstruction of denoised input D :
  0.0127   0.0094   0.0030   0.0007   0.0009   0.0065
  0.9913   0.9940   1.0000   1.0027   1.0032   0.9976
  0.0084   0.0086   0.0072   0.0048   0.0028   0.0050
  1.0009   0.9998   0.9966   0.9942   0.9930   0.9965
  0.0061   0.0070   0.0103   0.0134   0.0154   0.0114
  1.0034   1.0036   1.0028   1.0011   0.9996   1.0011
  0.0135   0.0113   0.0093   0.0114   0.0147   0.0148
```