NAG Library Routine Document

C09FZF

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

C09FZF inserts a selected set of three-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 reconstruction routines C09FBF or C09FDF.

2 Specification

```
SUBROUTINE C09FZF (ILEV, CINDEX, LENC, C, D, LDD, SDD, ICOMM, IFAIL)
INTEGER            ILEV, CINDEX, LENC, LDD, SDD, ICOMM(260), IFAIL
REAL (KIND=nag_wp) C(LENC), D(LDD,SDD,*)
```

3 Description

C09FZF inserts a selected set of three-dimensional DWT coefficients into the full set of coefficients stored in compact form in a one-dimensional array C. It is required that C09FZF is preceded by a call to the initialization routine C09ACF and either the forwards transform routine C09FAF or multi-level forwards transform routine C09FCF.

Given an initial three-dimensional data set A, a prior call to C09FAF or C09FCF computes the approximation coefficients (at the highest requested level in the case of C09FCF) and, seven sets of detail coefficients (at all levels in the case of C09FCF) and stores these in compact form in a one-dimensional array C. C09FYF can then extract either the approximation coefficients or one of the sets of detail coefficients (at one of the levels following C09FCF) into a three-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 C09FZF. Several extractions and insertions may be performed. C09FBF or C09FDF 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 either: the arrays DWTLVM, DWTLVN and DWTLVFR as returned by C09FCF if this was called first; or, otherwise from NWCT, NWCN and NWCFR as returned by C09ACF. See Section 2.1 in the C09 Chapter Introduction for a discussion of the three-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. Following a call to C09FAF (i.e., when $\text{ILEV} = 0$) this is equal to $\text{NWCT}/(8 \times \text{NWCN} \times \text{NWCFR})$ as returned by C09ACF. Following a call to C09FCF transforming NWL levels, and when inserting at level $\text{ILEV} > 0$, this is equal to $\text{DWTLVM}(\text{NWL} - \text{ILEV} + 1)$.

n_{cn} is the number of wavelet coefficients in the second dimension. Following a call to C09FAF (i. e., when $\text{ILEV} = 0$) this is equal to NWCN as returned by C09ACF. Following a call to C09FCF transforming NWL levels, and when inserting at level $\text{ILEV} > 0$, this is equal to $\text{DWTLVN}(\text{NWL} - \text{ILEV} + 1)$.

n_{cfr} is the number of wavelet coefficients in the third dimension. Following a call to C09FAF (i. e., when ILEV $= 0$) this is equal to NWCFR as returned by C09ACF. Following a call to C09FCF transforming NWL levels, and when inserting at level ILEV > 0, this is equal to DWTLVFR(NWL $-$ ILEV $+ 1$).

1: ILEV – INTEGER *Input*

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

If ILEV $= 0$, it is assumed that the coefficient array C was produced by a preceding call to the single level routine C09FAF.

If ILEV > 0, it is assumed that the coefficient array C was produced by a preceding call to the multi-level routine C09FCF.

Constraints:

ILEV $= 0$ (following a call to C09FAF);
$0 \leq$ ILEV $\leq$ NWL, where NWL is as used in a preceding call to C09FCF;
if CINDEX $= 0$, ILEV $=$ NWL (following a call to C09FCF).

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, rows and frames of A (LLL). After a call to the multi-level transform routine C09FCF (which implies that ILEV > 0) the approximation coefficients are present only for ILEV $=$ NWL, where NWL is the value used in a preceding call to C09FCF.

CINDEX $= 1$
: The detail coefficients produced by applying the low pass filter over columns and rows of A and the high pass filter over frames (LLH).

CINDEX $= 2$
: The detail coefficients produced by applying the low pass filter over columns, high pass filter over rows and low pass filter over frames of A (LHL).

CINDEX $= 3$
: The detail coefficients produced by applying the low pass filter over columns of A and high pass filter over rows and frames (LHH).

CINDEX $= 4$
: The detail coefficients produced by applying the high pass filter over columns of A and low pass filter over rows and frames (HLL).

CINDEX $= 5$
: The detail coefficients produced by applying the high pass filter over columns, low pass filter over rows and high pass filter over frames of A (HLH).

CINDEX $= 6$
: The detail coefficients produced by applying the high pass filter over columns and rows of A and the low pass filter over frames (HHL).

CINDEX $= 7$
: The detail coefficients produced by applying the high pass filter over columns, rows and frames of A (HHH).

Constraints:

if ILEV $= 0$, $0 \leq$ CINDEX ≤ 7;
if ILEV $=$ NWL, following a call to C09FCF transforming NWL levels, $0 \leq$ CINDEX ≤ 7;
otherwise $1 \leq$ CINDEX ≤ 7.

3: LENC – INTEGER *Input*

On entry: the dimension of the array C as declared in the (sub)program from which C09FZF is called.

Constraint: LENC must be unchanged from the value used in the preceding call to either C09FAF or C09FCF..

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

On entry: contains the DWT coefficients inserted by previous calls to C09FZF, or computed by a previous call to either C09FAF or C09FCF.

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 one of the reconstruction routines C09FBF (if C09FAF was called previously) or C09FDF (if C09FCF was called previously).

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

Note: the last dimension of the array D must be at least n_{cfr}.

On entry: the coefficients to be inserted.

If the DWT coefficients were computed by C09FAF then

if $\mathrm{CINDEX} = 0$, the approximation coefficients must be stored in $\mathrm{D}(i,j,k)$, for $i = 1,2,\ldots,n_{\mathrm{cm}}$, $j = 1,2,\ldots,n_{\mathrm{cn}}$ and $k = 1,2,\ldots,n_{\mathrm{cfr}}$;

if $1 \leq \mathrm{CINDEX} \leq 7$, the detail coefficients, as indicated by CINDEX, must be stored in $\mathrm{D}(i,j,k)$, for $i = 1,2,\ldots,n_{\mathrm{cm}}$, $j = 1,2,\ldots,n_{\mathrm{cn}}$ and $k = 1,2,\ldots,n_{\mathrm{cfr}}$.

If the DWT coefficients were computed by C09FCF then

if $\mathrm{CINDEX} = 0$ and $\mathrm{ILEV} = \mathrm{NWL}$, the approximation coefficients must be stored in $\mathrm{D}(i,j,k)$, for $i = 1,2,\ldots,n_{\mathrm{cm}}$, $j = 1,2,\ldots,n_{\mathrm{cn}}$ and $k = 1,2,\ldots,n_{\mathrm{cfr}}$;

if $1 \leq \mathrm{CINDEX} \leq 7$, the detail coefficients, as indicated by CINDEX, for level ILEV must be stored in $\mathrm{D}(i,j,k)$, for $i = 1,2,\ldots,n_{\mathrm{cm}}$, $j = 1,2,\ldots,n_{\mathrm{cn}}$ and $k = 1,2,\ldots,n_{\mathrm{cfr}}$.

6: LDD – INTEGER *Input*

On entry: the first dimension of the array D as declared in the (sub)program from which C09FZF is called.

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

7: SDD – INTEGER *Input*

On entry: the second dimension of the array D as declared in the (sub)program from which C09FZF is called.

Constraint: $\mathrm{SDD} > n_{\mathrm{cn}}$.

8: ICOMM(260) – 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 C09ACF.

9: 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 $= 0$ following a call to the single level routine C09FAF.

On entry, ILEV $= \langle value \rangle$.
Constraint: ILEV > 0 following a call to the multi-level routine C09FCF.

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 C09FCF.

IFAIL $= 2$

On entry, CINDEX $= \langle value \rangle$.
Constraint: CINDEX ≤ 7.

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

IFAIL $= 3$

On entry, LENC $= \langle value \rangle$ and $n_{ct} = \langle value \rangle$.
Constraint: LENC $\geq n_{ct}$, where n_{ct} is the number of DWT coefficients computed in a previous call to C09FAF.

On entry, LENC $= \langle value \rangle$ and $n_{ct} = \langle value \rangle$.
Constraint: LENC $\geq n_{ct}$, where n_{ct} is the number of DWT coefficients computed in a previous call to C09FCF.

IFAIL $= 4$

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

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

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

On entry, SDD $= \langle value \rangle$ and $n_{cn} = \langle value \rangle$.
Constraint: SDD $\geq n_{cn}$, where n_{cn} is the number of DWT coefficients in the second dimension following the single level transform.

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 C09FCF.

IFAIL = 6

Either the initialization routine has not been called first 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

C09FZF 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 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 c09fzfe

!     C09FZF Example Program Text
!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: c09acf, c09fcf, c09fdf, c09fyf, c09fzf, 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, fr, i, ifail,      &
                                          ilev, j, k, lda, ldb, ldd, lenc, m,  &
                                          n, nf, nwcfr, nwcn, nwct, nwl, sda,  &
                                          sdb, sdd
      Character (10)                   :: mode, wavnam, wtrans
!     .. Local Arrays ..
```

```
      Real (Kind=nag_wp), Allocatable  :: a(:,:,:), an(:,:,:), b(:,:,:), c(:), &
                                          d(:,:,:), e(:,:,:)
      Integer, Allocatable             :: dwtlvfr(:), dwtlvm(:), dwtlvn(:)
      Integer                          :: icomm(260)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: abs, log, real, sqrt
!     .. Executable Statements ..
      Write (nout,*) 'C09FZF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
!     Read problem parameters
      Read (nin,*) m, n, fr
      Read (nin,*) wavnam, mode
      Write (nout,99999) wavnam, mode, m, n, fr

!     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
      sda = n
      sdb = n
      Allocate (a(lda,lda,fr),an(lda,lda,fr),b(ldb,ldb,fr),e(m,n,fr))

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

!     Output the original data
      Write (nout,99997)
      Do k = 1, fr
        Write (nout,99991) k
        Do i = 1, m
          Write (nout,99998) a(i,1:n,k)
        End Do
      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,sda,m,n,fr,thresh)

!     Output the noisy data
      Write (nout,99996)
      Do k = 1, fr
        Write (nout,99991) k
        Do i = 1, m
          Write (nout,99998) an(i,1:n,k)
        End Do
      End Do

!     Query wavelet filter dimensions
!     For Multi-Resolution Analysis, decomposition, wtrans = 'M'
      wtrans = 'Multilevel'
      ifail = 0
      Call c09acf(wavnam,wtrans,mode,m,n,fr,nwl,nf,nwct,nwcn,nwcfr,icomm,      &
        ifail)

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

!     Perform a forwards multi-level transform on the noisy data
      ifail = 0
```

```
      Call c09fcf(m,n,fr,an,lda,sda,lenc,c,nwl,dwtlvm,dwtlvn,dwtlvfr,icomm,    &
        ifail)

!     Reconstruct without thresholding of detail coefficients
      ifail = 0
      Call c09fdf(nwl,lenc,c,m,n,fr,b,ldb,sdb,icomm,ifail)

!     Calculate the Mean Square Error of the noisy reconstruction
      e(:,:,:) = a(:,:,:) - b(:,:,:)
      mse = dnrm2(m*n*fr,e,1)
      mse = mse**2
      mse = mse/real(m*n*fr,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 3D array to hold the detail coefficients
      ldd = dwtlvm(nwl)
      sdd = dwtlvn(nwl)
      Allocate (d(ldd,sdd,dwtlvn(nwl)))

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

!       Select detail coefficients
        Do cindex = 1, 7

!         Extract coefficients into the 3D array D
          ifail = 0
          Call c09fyf(ilev,cindex,lenc,c,d,ldd,sdd,icomm,ifail)

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

!         Insert the denoised coefficients back into C
          ifail = 0
          Call c09fzf(ilev,cindex,lenc,c,d,ldd,sdd,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)*dwtlvfr(1)

!     Reconstruct original data following thresholding of detail coefficients
      ifail = 0
      Call c09fdf(nwl,lenc,c,m,n,fr,b,ldb,sdb,icomm,ifail)

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

!     Output the denoised reconstruction
      Write (nout,99992)
      Do k = 1, fr
        Write (nout,99991) k
```

```
        Do i = 1, m
          Write (nout,99998) b(i,1:n,k)
        End Do
      End Do

99999 Format (1X,' MLDWT :: Wavelet  : ',A,/,1X,'          End mode : ',A,/,   &
        1X,'          M        : ',I4,/,1X,'          N        : ',I4,/,1X,    &
        '          FR       : ',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 : ')
99991 Format (1X,' Frame ',I2,' : ')

    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,sda,m,n,fr,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)           :: fr, lda, m, n, sda
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: a(lda,sda,fr)
        Real (Kind=nag_wp), Intent (Out) :: an(lda,sda,fr)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: var, xmu
        Integer                        :: genid, i, ifail, j, 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,fr))

!       Generate the noise variates
        ifail = 0
```

```
        Do j = 1, fr
          Do i = 1, n
            Call g05skf(m,xmu,var,state,x(1,i,j),ifail)
          End Do
        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*fr,kind=nag_wp)))

      End Subroutine create_noise

    End Program c09fzfe
```

10.2 Program Data

```
C09FZF Example Program Data
 4, 4, 4   : m, n, fr
 Haar Period : wavnam, mode
0.01 0.01 0.01 0.01
1.00 1.00 1.00 1.00
0.01 0.01 0.01 0.01
1.00 1.00 1.00 1.00

1.00 1.00 1.00 1.00
0.01 0.01 0.01 0.01
1.00 1.00 1.00 1.00
0.01 0.01 0.01 0.01

0.01 0.01 0.01 0.01
1.00 1.00 1.00 1.00
0.01 0.01 0.01 0.01
1.00 1.00 1.00 1.00

1.00 1.00 1.00 1.00
0.01 0.01 0.01 0.01
1.00 1.00 1.00 1.00
0.01 0.01 0.01 0.01
```

10.3 Program Results

```
 C09FZF Example Program Results

  MLDWT :: Wavelet  : Haar
           End mode : Period
           M        :    4
           N        :    4
           FR       :    4

  Original data            A  :
  Frame  1 :
  0.0100   0.0100   0.0100   0.0100
  1.0000   1.0000   1.0000   1.0000
  0.0100   0.0100   0.0100   0.0100
  1.0000   1.0000   1.0000   1.0000
  Frame  2 :
  1.0000   1.0000   1.0000   1.0000
  0.0100   0.0100   0.0100   0.0100
  1.0000   1.0000   1.0000   1.0000
  0.0100   0.0100   0.0100   0.0100
  Frame  3 :
  0.0100   0.0100   0.0100   0.0100
  1.0000   1.0000   1.0000   1.0000
  0.0100   0.0100   0.0100   0.0100
  1.0000   1.0000   1.0000   1.0000
  Frame  4 :
  1.0000   1.0000   1.0000   1.0000
  0.0100   0.0100   0.0100   0.0100
```

```
 1.0000   1.0000   1.0000   1.0000
 0.0100   0.0100   0.0100   0.0100

 Original data plus noise AN :
 Frame  1 :
 0.0135  -0.0093  -0.0004   0.0378
 1.0015   0.9842   1.0007   0.9889
-0.0017   0.0139   0.0138  -0.0049
 0.9899   1.0070   1.0049   0.9983
 Frame  2 :
 1.0094   1.0080   0.9921   0.9902
 0.0105  -0.0009   0.0160   0.0197
 0.9994   1.0044   0.9956   1.0014
 0.0091  -0.0084   0.0187   0.0023
 Frame  3 :
 0.0058  -0.0053   0.0011   0.0159
 1.0113   0.9894   1.0018   0.9992
 0.0106   0.0082   0.0093   0.0153
 1.0023   1.0157   1.0084   0.9834
 Frame  4 :
 0.9969   1.0010   0.9904   0.9968
 0.0227   0.0022   0.0062   0.0214
 0.9948   0.9981   0.9951   0.9968
 0.0121   0.0103   0.0114   0.0206

 Without denoising Mean Square Error is  0.000081

 Number of coefficients denoised is  55 out of  63

 With denoising Mean Square Error is  0.000015

 Reconstruction of denoised input D :
 Frame  1 :
 0.0053   0.0053   0.0166   0.0166
 1.0026   1.0026   0.9913   0.9913
 0.0055   0.0055   0.0077   0.0077
 1.0025   1.0025   1.0003   1.0003
 Frame  2 :
 1.0026   1.0026   0.9913   0.9913
 0.0053   0.0053   0.0166   0.0166
 1.0025   1.0025   1.0003   1.0003
 0.0055   0.0055   0.0077   0.0077
 Frame  3 :
 0.0073   0.0073   0.0110   0.0110
 1.0006   1.0006   0.9969   0.9969
 0.0078   0.0078   0.0131   0.0131
 1.0002   1.0002   0.9949   0.9949
 Frame  4 :
 1.0006   1.0006   0.9969   0.9969
 0.0073   0.0073   0.0110   0.0110
 1.0002   1.0002   0.9949   0.9949
 0.0078   0.0078   0.0131   0.0131
```