NAG Library Function Document

nag_wav_2d_coeff_ins (c09ezc)

1 Purpose

nag_wav_2d_coeff_ins (c09ezc) 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 function nag_imldwt_2d (c09edc).

2 Specification

```
#include <nag.h>
#include <nagc09.h>
void nag_wav_2d_coeff_ins (Integer ilev, Integer cindex, Integer lenc,
     double c[], const double d[], Integer pdd, Integer icomm[],
     NagError *fail)
```

3 Description

nag_wav_2d_coeff_ins (c09ezc) 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 nag_wav_2d_coeff_ins (c09ezc) is preceded by a call to the initialization function nag_wfilt_2d (c09abc) and the forward multi-level transform function nag_mldwt_2d (c09ecc).

Given an initial two-dimensional data set A, a prior call to nag_mldwt_2d (c09ecc) 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**. nag_wav_2d_coeff_ext (c09eyc) can then extract either the approximation coefficients or one of the sets of detail coefficients at one of the levels as two-dimensional data into the 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 nag_wav_2d_coeff_ins (c09ezc). Several extractions and insertions may be performed at different levels. nag_imldwt_2d (c09edc) can then be used to reconstruct a manipulated data set $\tilde{A}$. The dimensions of the two-dimensional data stored in **d** depend on the level extracted and are available from the arrays **dwtlvm** and **dwtlvn** as returned by nag_mldwt_2d (c09ecc) 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**] as returned by a call to nag_mldwt_2d (c09ecc) 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**] as returned by a call to nag_mldwt_2d (c09ecc) transforming **nwl** levels.

1: **ilev** – Integer *Input*

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

Constraints:

$1 \leq$ **ilev** $\leq$ **nwl**, where **nwl** is as used in a preceding call to nag_mldwt_2d (c09ecc); 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 nag_mldwt_2d (c09ecc).

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 nag_mldwt_2d (c09ecc), otherwise $1 \leq$ **cindex** ≤ 3.

3: **lenc** – Integer *Input*

On entry: the dimension of the array **c**.

Constraint: **lenc** must be unchanged from the value used in the preceding call to nag_mldwt_2d (c09ecc)..

4: **c**[**lenc**] – double *Input/Output*

On entry: contains the DWT coefficients inserted by previous calls to nag_wav_2d_coeff_ins (c09ezc), or computed by a previous call to nag_mldwt_2d (c09ecc).

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 function nag_imldwt_2d (c09edc).

5: **d**[dim] – const double *Input*

Note: the dimension, *dim*, of the array **d** must be at least **pdd** $\times n_{\mathrm{cn}}$.

On entry: the coefficients to be inserted.

If **ilev** $=$ **nwl** (as used in nag_mldwt_2d (c09ecc)) and **cindex** $= 0$, the n_{cm} by n_{cn} manipulated approximation coefficients a_{ij} must be stored in $\mathbf{d}[(j-1) \times \mathbf{pdd} + i - 1]$, 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 $\mathbf{d}[(j-1) \times \mathbf{pdd} + i - 1]$, for $i = 1, 2, \ldots, n_{\mathrm{cm}}$ and $j = 1, 2, \ldots, n_{\mathrm{cn}}$.

6: **pdd** – Integer *Input*

On entry: the stride separating row elements in the two-dimensional data stored in the array **d**.

Constraint: **pdd** $\geq n_{\mathrm{cm}}$.

7: **icomm**[**180**] – Integer *Communication Array*

On entry: contains details of the discrete wavelet transform and the problem dimension as setup in the call to the initialization function nag_wfilt_2d (c09abc).

8: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INITIALIZATION

Either the initialization function has not been called first or **icomm** has been corrupted.

Either the initialization function was called with **wtrans** = Nag_SingleLevel or **icomm** has been corrupted.

NE_INT

On entry, **cindex** = ⟨*value*⟩.
Constraint: **cindex** ≤ 3.

On entry, **cindex** = ⟨*value*⟩.
Constraint: **cindex** ≥ 0.

On entry, **ilev** = ⟨*value*⟩.
Constraint: **ilev** ≥ 1.

NE_INT_2

On entry, **ilev** = ⟨*value*⟩ and **nwl** = ⟨*value*⟩.
Constraint: **ilev** $\leq$ **nwl**, where **nwl** is the number of levels used in the call to nag_mldwt_2d (c09ecc).

On entry, **lenc** = ⟨*value*⟩ and n_{ct} = ⟨*value*⟩.
Constraint: **lenc** $\geq n_{\mathrm{ct}}$, where n_{ct} is the number of DWT coefficients computed in a previous call to nag_mldwt_2d (c09ecc).

On entry, **pdd** = ⟨*value*⟩ and n_{cm} = ⟨*value*⟩.
Constraint: **pdd** $\geq n_{\mathrm{cm}}$, where n_{cm} is the number of DWT coefficients in the first dimension at the selected level **ilev**.

NE_INT_3

On entry, **ilev** = ⟨*value*⟩ and **nwl** = ⟨*value*⟩, but **cindex** = 0.
Constraint: **cindex** > 0 when **ilev** $<$ **nwl** in the preceding call to nag_mldwt_2d (c09ecc).

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_wav_2d_coeff_ins (c09ezc) is not threaded in any implementation.

9 Further Comments

None.

10 Example

The following example demonstrates using the coefficient extraction and insertion functions 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

```
/* nag_wav_2d_coeff_ins (c09ezc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagc09.h>
#include <nagg05.h>

#define A(I,J) a[(J-1)*lda + I-1]
#define AN(I,J) an[(J-1)*lda + I-1]
#define B(I,J) b[(J-1)*ldb + I-1]
#define D(I,J) d[(J-1)*ldd + I-1]

int main(void)
{
  /* Scalars */
  Integer exit_status = 0;
  Integer lstate = 1, lseed = 1;
  Integer i, j, k, lda, ldb, ldd, lenc, m, n, mn, nf, nwcn, nwct, nwl;
  Integer subid, denoised, cindex, ilev;
  Nag_BaseRNG genid;
  double mse, thresh, var, xmu;
  /* Arrays */
  char mode[25], wavnam[25];
  double *a = 0, *an = 0, *b = 0, *c = 0, *d = 0, *x = 0;
  Integer *dwtlvm = 0, *dwtlvn = 0, *state = 0;
```

```
  Integer icomm[180], seed[1];
  /* Nag Types */
  Nag_Wavelet wavnamenum;
  Nag_WaveletMode modenum;
  Nag_MatrixType matrix = Nag_GeneralMatrix;
  Nag_OrderType order = Nag_ColMajor;
  Nag_DiagType diag = Nag_NonUnitDiag;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_wav_2d_coeff_ins (c09ezc) Example Program Results\n\n");
  /* Skip heading in data file and read problem parameters. */
#ifdef _WIN32
  scanf_s("%*[^\n] %" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &m, &n);
#else
  scanf("%*[^\n] %" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &m, &n);
#endif
#ifdef _WIN32
  scanf_s("%24s%24s%*[^\n] ", wavnam, (unsigned)_countof(wavnam), mode,
          (unsigned)_countof(mode));
#else
  scanf("%24s%24s%*[^\n] ", wavnam, mode);
#endif

  printf("MLDWT :: Wavelet  : %s\n", wavnam);
  printf("         End mode : %s\n", mode);
  printf("         m  : %4" NAG_IFMT "\n", m);
  printf("         n  : %4" NAG_IFMT "\n\n", n);
  fflush(stdout);

  /* 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;
  if (!(a = NAG_ALLOC((lda) * (n), double)) ||
      !(an = NAG_ALLOC((lda) * (n), double)) ||
      !(b = NAG_ALLOC((ldb) * (n), double)) ||
      !(x = NAG_ALLOC((m * n), double)))
  {
    printf("Allocation failure\n");
    exit_status = 1;
    goto END;
  }

  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value.
   */
  wavnamenum = (Nag_Wavelet) nag_enum_name_to_value(wavnam);
  modenum = (Nag_WaveletMode) nag_enum_name_to_value(mode);

  /* Read in the original data. */
  for (i = 1; i <= m; i++)
#ifdef _WIN32
    for (j = 1; j <= n; j++)
      scanf_s("%lf", &A(i, j));
#else
    for (j = 1; j <= n; j++)
      scanf("%lf", &A(i, j));
#endif

  /* Output the original data. */
  nag_gen_real_mat_print_comp(order, matrix, diag, m, n, a, lda, "%11.4e",
                              "Input data :", Nag_NoLabels, 0,
                              Nag_NoLabels, 0, 80, 0, 0, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_real_mat_print_comp (x04cbc).\n%s\n",
           fail.message);
    exit_status = 2;
```

```
  goto END;
}
printf("\n");
fflush(stdout);

/* Set up call to nag_rand_normal (g05skc) in order to create some
 * randomnoise from a normal distribution to add to the original data.
 * Initial call to RNG initializer to get size of STATE array.
 */
seed[0] = 642521;
genid = Nag_MersenneTwister;
subid = 0;
if (!(state = NAG_ALLOC((lstate), Integer)))
{
  printf("Allocation failure\n");
  exit_status = 3;
  goto END;
}

/* nag_rand_init_repeatable (g05kfc).
 * Query the size of state.
 */
lstate = 0;
nag_rand_init_repeatable(genid, subid, seed, lseed, state, &lstate, &fail);
if (fail.code != NE_NOERROR) {
  printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n",
         fail.message);
  exit_status = 4;
  goto END;
}

/* Reallocate STATE */
NAG_FREE(state);
if (!(state = NAG_ALLOC((lstate), Integer)))
{
  printf("Allocation failure\n");
  exit_status = 5;
  goto END;
}

/* nag_rand_init_repeatable (g05kfc).
 * Initialize the generator to a repeatable sequence.
 */
nag_rand_init_repeatable(genid, subid, seed, lseed, state, &lstate, &fail);
if (fail.code != NE_NOERROR) {
  printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n",
         fail.message);
  exit_status = 6;
  goto END;
}

/* Set the distribution parameters for the random noise. */
xmu = 0.0;
var = 0.1E-3;

/* Generate the noise variates */

/* nag_rand_normal (g05skc).
 * Generates a vector of pseudorandom numbers from a Normal distribution.
 */
mn = n * m;
nag_rand_normal(mn, xmu, var, state, x, &fail);
if (fail.code != NE_NOERROR) {
  printf("Error from nag_rand_normal (g05skc).\n%s\n", fail.message);
  exit_status = 7;
  goto END;
}

/* Add the noise to the original input and save in AN */
k = 0;
for (j = 1; j <= n; j++) {
```

```
    for (i = 1; i <= m; i++) {
      AN(i, j) = A(i, j) + x[k];
      k = k + 1;
    }
  }

  /* Output the noisy data */
  nag_gen_real_mat_print_comp(order, matrix, diag, m, n, an, lda, "%11.4e",
                              "Original data plus noise :", Nag_NoLabels, 0,
                              Nag_NoLabels, 0, 80, 0, 0, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_real_mat_print_comp (x04cbc).\n%s\n",
           fail.message);
    exit_status = 8;
    goto END;
  }
  printf("\n");

  /* nag_wfilt_2d (c09abc).
   * Two-dimensional wavelet filter initialization.
   */
  nag_wfilt_2d(wavnamenum, Nag_MultiLevel, modenum, m, n, &nwl, &nf, &nwct,
               &nwcn, icomm, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_wfilt_2d (c09abc).\n%s\n", fail.message);
    exit_status = 9;
    goto END;
  }

  /* Allocate arrays to hold the coefficients, c, and the dimensions
   * of the coefficients at each level, dwtlvm, dwtlvn.
   */
  lenc = nwct;
  if (!(c = NAG_ALLOC((lenc), double)) ||
      !(dwtlvm = NAG_ALLOC((nwl), Integer)) ||
      !(dwtlvn = NAG_ALLOC((nwl), Integer)))
  {
    printf("Allocation failure\n");
    exit_status = 10;
    goto END;
  }

  /* Perform a forwards multi-level transform on the noisy data. */

  /* nag_mldwt_2d (c09ecc).
   * Two-dimensional multi-level discrete wavelet transform.
   */
  nag_mldwt_2d(m, n, an, lda, lenc, c, nwl, dwtlvm, dwtlvn, icomm, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_mldwt_2d (c09ecc).\n%s\n", fail.message);
    exit_status = 11;
    goto END;
  }

  /* Reconstruct without thresholding of detail coefficients. */

  /* nag_imldwt_2d (c09edc).
   * Two-dimensional inverse multi-level discrete wavelet transform.
   */
  nag_imldwt_2d(nwl, lenc, c, m, n, b, ldb, icomm, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_imldwt_2d (c09edc).\n%s\n", fail.message);
    exit_status = 12;
    goto END;
  }

  /* Calculate the Mean Square Error of the noisy reconstruction. */
  mse = 0.0;
  for (j = 1; j <= n; j++)
    for (i = 1; i <= m; i++)
      mse = mse + pow((A(i, j) - B(i, j)), 2);
```

```
  mse = mse / (double) (m * n);
  printf("Without denoising Mean Square Error is %11.4e\n\n", mse);
  fflush(stdout);

  /* 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 - 1];
  if (!(d = NAG_ALLOC((ldd) * (dwtlvn[nwl - 1]), double)))
  {
    printf("Allocation failure\n");
    exit_status = 13;
    goto END;
  }

  /* Calculate the threshold based on VisuShrink denoising. */
  thresh = sqrt(var) * sqrt(2. * log((double) (m * n)));
  denoised = 0;
  /* For each level */
  for (ilev = nwl; ilev >= 1; ilev -= 1) {
    /* Select detail coefficients */
    for (cindex = 1; cindex <= 3; cindex++) {
      /* Extract coefficients into the 2D array d */

      /* nag_wav_2d_coeff_ext (c09eyc).
       * Two-dimensional discrete wavelet transform coefficient extraction.
       */
      nag_wav_2d_coeff_ext(ilev, cindex, lenc, c, d, ldd, icomm, &fail);
      if (fail.code != NE_NOERROR) {
        printf("Error from nag_wav_2d_coeff_ext (c09eyc).\n%s\n",
               fail.message);
        exit_status = 14;
        goto END;
      }

      /* Perform the hard thresholding operation */
      for (j = 1; j <= dwtlvn[nwl - ilev]; j++)
        for (i = 1; i <= dwtlvm[nwl - ilev]; i++)
          if (fabs(D(i, j)) < thresh) {
            D(i, j) = 0.0;
            denoised = denoised + 1;
          }

      /* Insert the denoised coefficients back into c. */

      /* nag_wav_2d_coeff_ins (c09ezc).
       * Two-dimensional discrete wavelet transform coefficient insertion.
       */
      nag_wav_2d_coeff_ins(ilev, cindex, lenc, c, d, ldd, icomm, &fail);
      if (fail.code != NE_NOERROR) {
        printf("Error from nag_wav_2d_coeff_ins (c09ezc).\n%s\n",
               fail.message);
        exit_status = 15;
        goto END;
      }

    }
  }

  /* Output the number of coefficients that were set to zero */
  printf("Number of coefficients denoised is %4" NAG_IFMT " out of %4"
         NAG_IFMT "\n\n", denoised, nwct - dwtlvm[0] * dwtlvn[0]);
  fflush(stdout);

  /* Reconstruct original data following thresholding of detail coefficients */

  /* nag_imldwt_2d (c09edc).
   * Two-dimensional inverse multi-level discrete wavelet transform.
   */
  nag_imldwt_2d(nwl, lenc, c, m, n, b, ldb, icomm, &fail);
```

```
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_imldwt_2d (c09edc).\n%s\n", fail.message);
    exit_status = 16;
    goto END;
  }

  /* Calculate the Mean Square Error of the denoised reconstruction. */
  mse = 0.0;
  for (j = 1; j <= n; j++)
    for (i = 1; i <= m; i++)
      mse = mse + pow((A(i, j) - B(i, j)), 2);
  mse = mse / (double) (m * n);
  printf("With denoising Mean Square Error is %11.4e \n\n", mse);
  fflush(stdout);

  /* Output the denoised reconstruction. */
  nag_gen_real_mat_print_comp(order, matrix, diag, m, n, b, ldb, "%11.4e",
                              "Reconstruction of denoised input :",
                              Nag_NoLabels, 0, Nag_NoLabels, 0, 80, 0, 0,
                              &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_real_mat_print_comp (x04cbc).\n%s\n",
           fail.message);
    exit_status = 17;
    goto END;
  }

END:
  NAG_FREE(a);
  NAG_FREE(an);
  NAG_FREE(b);
  NAG_FREE(c);
  NAG_FREE(d);
  NAG_FREE(x);
  NAG_FREE(dwtlvm);
  NAG_FREE(dwtlvn);
  NAG_FREE(state);
  return exit_status;
}
```

10.2 Program Data

```
nag_wav_2d_coeff_ins (c09ezc) Example Program Data
 7  6                          : m, n
 Nag_Daubechies6 Nag_Periodic : wavnam, mode
0.1000E-01 0.1000E-01 0.1000E-01 0.1000E-01 0.1000E-01 0.1000E-01
0.1000E+01 0.1000E+01 0.1000E+01 0.1000E+01 0.1000E+01 0.1000E+01
0.1000E-01 0.1000E-01 0.1000E-01 0.1000E-01 0.1000E-01 0.1000E-01
0.1000E+01 0.1000E+01 0.1000E+01 0.1000E+01 0.1000E+01 0.1000E+01
0.1000E-01 0.1000E-01 0.1000E-01 0.1000E-01 0.1000E-01 0.1000E-01
0.1000E+01 0.1000E+01 0.1000E+01 0.1000E+01 0.1000E+01 0.1000E+01
0.1000E-01 0.1000E-01 0.1000E-01 0.1000E-01 0.1000E-01 0.1000E-01
```

10.3 Program Results

```
nag_wav_2d_coeff_ins (c09ezc) Example Program Results

MLDWT :: Wavelet  : Nag_Daubechies6
         End mode : Nag_Periodic
         m  :    7
         n  :    6

 Input data :
   1.0000e-02  1.0000e-02  1.0000e-02  1.0000e-02  1.0000e-02  1.0000e-02
   1.0000e+00  1.0000e+00  1.0000e+00  1.0000e+00  1.0000e+00  1.0000e+00
   1.0000e-02  1.0000e-02  1.0000e-02  1.0000e-02  1.0000e-02  1.0000e-02
   1.0000e+00  1.0000e+00  1.0000e+00  1.0000e+00  1.0000e+00  1.0000e+00
   1.0000e-02  1.0000e-02  1.0000e-02  1.0000e-02  1.0000e-02  1.0000e-02
   1.0000e+00  1.0000e+00  1.0000e+00  1.0000e+00  1.0000e+00  1.0000e+00
   1.0000e-02  1.0000e-02  1.0000e-02  1.0000e-02  1.0000e-02  1.0000e-02
```

```
 Original data plus noise :
   1.3549e-02  1.6995e-02 -4.9390e-03 -8.5383e-04  1.8706e-04  1.2313e-02
   1.0015e+00  9.8963e-01  9.9833e-01  1.0044e+00  1.0097e+00  9.8469e-01
  -1.7004e-03  1.0657e-02  1.9363e-02 -8.3748e-03  1.1364e-02 -5.6284e-04
   9.8993e-01  1.0038e+00  1.0005e+00  9.9208e-01  9.9231e-01  9.9820e-01
  -9.3452e-03  1.4871e-02  9.3840e-03  1.5987e-02  5.8176e-03  2.5670e-02
   9.8416e-01  1.0278e+00  9.9907e-01  9.9562e-01  1.0113e+00  9.9112e-01
   1.3876e-02 -1.0616e-03  1.7986e-02  1.8721e-02  1.0583e-02  1.1846e-02

Without denoising Mean Square Error is  9.7832e-05

Number of coefficients denoised is   32 out of   48

With denoising Mean Square Error is  1.8134e-05

 Reconstruction of denoised input :
   1.2651e-02  9.3672e-03  2.9873e-03  6.6813e-04  9.2377e-04  6.5269e-03
   9.9126e-01  9.9404e-01  9.9996e-01  1.0027e+00  1.0032e+00  9.9764e-01
   8.4483e-03  8.5768e-03  7.1934e-03  4.8007e-03  2.7522e-03  5.0164e-03
   1.0009e+00  9.9979e-01  9.9655e-01  9.9419e-01  9.9300e-01  9.9650e-01
   6.0974e-03  6.9809e-03  1.0318e-02  1.3407e-02  1.5362e-02  1.1390e-02
   1.0034e+00  1.0036e+00  1.0028e+00  1.0011e+00  9.9964e-01  1.0011e+00
   1.3530e-02  1.1252e-02  9.2970e-03  1.1447e-02  1.4711e-02  1.4839e-02
```