NAG Library Function Document

nag_mldwt_2d (c09ecc)

1 Purpose

nag_mldwt_2d (c09ecc) computes the two-dimensional multi-level discrete wavelet transform (DWT). The initialization function nag_wfilt_2d (c09abc) must be called first to set up the DWT options.

2 Specification

```
#include <nag.h>
#include <nagc09.h>
void nag_mldwt_2d (Integer m, Integer n, const double a[], Integer lda,
     Integer lenc, double c[], Integer nwl, Integer dwtlvm[],
     Integer dwtlvn[], Integer icomm[], NagError *fail)
```

3 Description

nag_mldwt_2d (c09ecc) computes the multi-level DWT of two-dimensional data. For a given wavelet and end extension method, nag_mldwt_2d (c09ecc) will compute a multi-level transform of a matrix A, using a specified number, n_{fwd}, of levels. The number of levels specified, n_{fwd}, must be no more than the value $l_{\max}$ returned in **nwlmax** by the initialization function nag_wfilt_2d (c09abc) for the given problem. The transform is returned as a set of coefficients for the different levels (packed into a single array) and a representation of the multi-level structure.

The notation used here assigns level 0 to the input matrix, A. Level 1 consists of the first set of coefficients computed: the vertical (v_1), horizontal (h_1) and diagonal (d_1) coefficients are stored at this level while the approximation (a_1) coefficients are used as the input to a repeat of the wavelet transform at the next level. This process is continued until, at level n_{fwd}, all four types of coefficients are stored. The output array, C, stores these sets of coefficients in reverse order, starting with $a_{n_{\text{fwd}}}$ followed by $v_{n_{\text{fwd}}}, h_{n_{\text{fwd}}}, d_{n_{\text{fwd}}}, v_{n_{\text{fwd}}-1}, h_{n_{\text{fwd}}-1}, d_{n_{\text{fwd}}-1}, \ldots, v_1, h_1, d_1$.

4 References

None.

5 Arguments

1: **m** – Integer *Input*

On entry: number of rows, m, of data matrix A.

Constraint: this must be the same as the value **m** passed to the initialization function nag_wfilt_2d (c09abc).

2: **n** – Integer *Input*

On entry: number of columns, n, of data matrix A.

Constraint: this must be the same as the value **n** passed to the initialization function nag_wfilt_2d (c09abc).

3: **a**[**lda** × **n**] – const double *Input*

Note: the (i,j)th element of the matrix A is stored in **a**$[(j-1) \times$ **lda** $+ i - 1]$.

On entry: the m by n data matrix A.

4: **lda** – Integer *Input*

On entry: the stride separating matrix row elements in the array **a**.

Constraint: **lda** $\geq$ **m**.

5: **lenc** – Integer *Input*

On entry: the dimension of the array **c**. **c** must be large enough to contain, n_{ct}, wavelet coefficients. The maximum value of n_{ct} is returned in **nwct** by the call to the initialization function nag_wfilt_2d (c09abc) and corresponds to the DWT being continued for the maximum number of levels possible for the given data set. When the number of levels, n_{fwd}, is chosen to be less than the maximum, $l_{\max}$, then n_{ct} is correspondingly smaller and **lenc** can be reduced by noting that the vertical, horizontal and diagonal coefficients are stored at every level and that in addition the approximation coefficients are stored for the final level only. The number of coefficients stored at each level is given by $3 \times \lceil \bar{m}/2 \rceil \times \lceil \bar{n}/2 \rceil$ for **mode** = Nag_Periodic in nag_wfilt_2d (c09abc) and $3 \times \lfloor (\bar{m} + n_f - 1)/2 \rfloor \times \lfloor (\bar{n} + n_f - 1)/2 \rfloor$ for **mode** = Nag_HalfPointSymmetric, Nag_WholePointSymmetric or Nag_ZeroPadded, where the input data is of dimension $\bar{m} \times \bar{n}$ at that level and n_f is the filter length **nf** provided by the call to nag_wfilt_2d (c09abc). At the final level the storage is $4/3$ times this value to contain the set of approximation coefficients.

Constraint: **lenc** $\geq n_{\mathrm{ct}}$, where n_{ct} is the total number of coefficients that correspond to a transform with **nwl** levels.

6: **c**[**lenc**] – double *Output*

On exit: the coefficients of the discrete wavelet transform. If you need to access or modify the approximation coefficients or any specific set of detail coefficients then the use of nag_wav_2d_coeff_ext (c09eyc) or nag_wav_2d_coeff_ins (c09ezc) is recommended. For completeness the following description provides details of precisely how the coefficient are stored in **c** but this information should only be required in rare cases.

Let $q(i)$ denote the number of coefficients (of each type) at level i, for $i = 1, 2, \ldots, n_{\mathrm{fwd}}$, such that $q(i) =$ **dwtlvm**$[n_{\mathrm{fwd}} - i] \times$ **dwtlvn**$[n_{\mathrm{fwd}} - i]$. Then, letting $k_1 = q(n_{\mathrm{fwd}})$ and $k_{j+1} = k_j + q(n_{\mathrm{fwd}} - \lceil j/3 \rceil + 1)$, for $j = 1, 2, \ldots, 3n_{\mathrm{fwd}}$, the coefficients are stored in **c** as follows:

c$[i-1]$, for $i = 1, 2, \ldots, k_1$
Contains the level n_{fwd} approximation coefficients, $a_{n_{\mathrm{fwd}}}$.

c$[i-1]$, for $i = k_j + 1, \ldots, k_{j+1}$
Contains the level $n_{\mathrm{fwd}} - \lceil j/3 \rceil + 1$ vertical, horizontal and diagonal coefficients. These are:

vertical coefficients if $j \bmod 3 = 1$;

horizontal coefficients if $j \bmod 3 = 2$;

diagonal coefficients if $j \bmod 3 = 0$,

for $j = 1, \ldots, 3n_{\mathrm{fwd}}$.

7: **nwl** – Integer *Input*

On entry: the number of levels, n_{fwd}, in the multi-level resolution to be performed.

Constraint: $1 \leq$ **nwl** $\leq l_{\max}$, where $l_{\max}$ is the value returned in **nwlmax** (the maximum number of levels) by the call to the initialization function nag_wfilt_2d (c09abc).

8: **dwtlvm**[**nwl**] – Integer *Output*

On exit: the number of coefficients in the first dimension for each coefficient type at each level. **dwtlvm**$[i-1]$ contains the number of coefficients in the first dimension (for each coefficient type computed) at the $(n_{\mathrm{fwd}} - i + 1)$th level of resolution, for $i = 1, 2, \ldots, n_{\mathrm{fwd}}$. Thus for the first $n_{\mathrm{fwd}} - 1$ levels of resolution, **dwtlvm**$[n_{\mathrm{fwd}} - i]$ is the size of the first dimension of the matrices of

vertical, horizontal and diagonal coefficients computed at this level; for the final level of resolution, **dwtlvm**[0] is the size of the first dimension of the matrices of approximation, vertical, horizontal and diagonal coefficients computed.

9: **dwtlvn**[**nwl**] – Integer *Output*

On exit: the number of coefficients in the second dimension for each coefficient type at each level. **dwtlvn**$[i-1]$ contains the number of coefficients in the second dimension (for each coefficient type computed) at the $(n_{\text{fwd}}-i+1)$th level of resolution, for $i = 1, 2, \ldots, n_{\text{fwd}}$. Thus for the first $n_{\text{fwd}}-1$ levels of resolution, **dwtlvn**$[n_{\text{fwd}}-i]$ is the size of the second dimension of the matrices of vertical, horizontal and diagonal coefficients computed at this level; for the final level of resolution, **dwtlvn**[0] is the size of the second dimension of the matrices of approximation, vertical, horizontal and diagonal coefficients computed.

10: **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).

On exit: contains additional information on the computed transform.

11: **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, **m** = ⟨*value*⟩.
Constraint: **m** = ⟨*value*⟩, the value of **m** on initialization (see nag_wfilt_2d (c09abc)).

On entry, **n** = ⟨*value*⟩.
Constraint: **n** = ⟨*value*⟩, the value of **n** on initialization (see nag_wfilt_2d (c09abc)).

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

NE_INT_2

On entry, **lda** = ⟨*value*⟩ and **m** = ⟨*value*⟩.
Constraint: **lda** $\geq$ **m**.

On entry, **lenc** = ⟨*value*⟩.
Constraint: **lenc** $\geq$ ⟨*value*⟩, the total number of coefficents to be generated.

On entry, **nwl** = ⟨*value*⟩ and **nwlmax** = ⟨*value*⟩ in nag_wfilt_2d (c09abc).
Constraint: **nwl** ≤ **nwlmax** in nag_wfilt_2d (c09abc).

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

The accuracy of the wavelet transform depends only on the floating-point operations used in the convolution and downsampling and should thus be close to ***machine precision***.

8 Parallelism and Performance

nag_mldwt_2d (c09ecc) is not threaded in any implementation.

9 Further Comments

The wavelet coefficients at each level can be extracted from the output array **c** using the information contained in **dwtlvm** and **dwtlvn** on exit (see the descriptions of **c**, **dwtlvm** and **dwtlvn** in Section 5). For example, given an input data set, A, denoising can be carried out by applying a thresholding operation to the detail (vertical, horizontal and diagonal) coefficients at every level. The elements $\mathbf{c}[k_1]$ to $\mathbf{c}[k_{n_{\text{fwd}}+1} - 1]$, as described in Section 5, contain the detail coefficients, $\hat{c}_{ij}$, for $i = n_{\text{fwd}}, n_{\text{fwd}} - 1, \ldots, 1$ and $j = 1, 2, \ldots, 3q(i)$, where $q(i)$ is the number of each type of coefficient at level i and $\hat{c}_{ij} = c_{ij} + \sigma\epsilon_{ij}$ and $\sigma\epsilon_{ij}$ is the transformed noise term. If some threshold parameter α is chosen, a simple hard thresholding rule can be applied as

$$\bar{c}_{ij} = \begin{cases} 0, & \text{if } \left|\hat{c}_{ij}\right| \leq \alpha \\ \hat{c}_{ij}, & \text{if } \left|\hat{c}_{ij}\right| > \alpha, \end{cases}$$

taking $\bar{c}_{ij}$ to be an approximation to the required detail coefficient without noise, c_{ij}. The resulting coefficients can then be used as input to nag_imldwt_2d (c09edc) in order to reconstruct the denoised signal. See Section 10 in nag_wav_2d_coeff_ins (c09ezc) for a simple example of denoising.

See the references given in the introduction to this chapter for a more complete account of wavelet denoising and other applications.

10 Example

This example performs a multi-level resolution transform of a dataset using the Daubechies wavelet (see **wavnam** = Nag_Daubechies2 in nag_wfilt_2d (c09abc)) using half-point symmetric end extensions, the maximum possible number of levels of resolution, where the number of coefficients in each level and the coefficients themselves are not changed. The original dataset is then reconstructed using nag_imldwt_2d (c09edc).

10.1 Program Text

```
/* nag_mldwt_2d (c09ecc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <string.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagc09.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer exit_status = 0;
  Integer nwcm, i, ilevel, itype_coeffs, j, lenc, m, n,
          nf, nwcn, nwct, nwlmax, nwl, nwlinv, pda, pdb;
  /* Arrays */
  char mode[24], wavnam[20], title[50];
  double *a = 0, *b = 0, *c = 0, *d = 0;
  Integer *dwtlvm = 0, *dwtlvn = 0;
  Integer icomm[180];
  /* 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);

  /* Output preamble */
  printf("nag_mldwt_2d (c09ecc) 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("%19s%23s%*[^\n] ", wavnam, (unsigned)_countof(wavnam), mode,
          (unsigned)_countof(mode));
#else
  scanf("%19s%23s%*[^\n] ", wavnam, mode);
#endif
  pda = m;
  pdb = m;
  if (!(a = NAG_ALLOC(pda * n, double)) || !(b = NAG_ALLOC(pdb * n, double))
        )
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  printf(" Parameters read from file :: \n");
  printf(" MLDWT :: Wavelet  : %s\n", wavnam);
  printf("          End mode : %s\n", mode);
  printf("          m        : %" NAG_IFMT "\n", m);
  printf("          n        : %" NAG_IFMT "\n\n", n);
  fflush(stdout);

  /* 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 data array and write it out */
#define A(I, J) a[(J-1)*pda + I-1]
  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

  nag_gen_real_mat_print_comp(order, matrix, diag, m, n, a, pda, "%8.4f",
                              "Input Data     A :", 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 = 1;
    goto END;
  }

  /* nag_wfilt_2d (c09abc).
   *  Two-dimensional wavelet filter initialization.
   */
  nag_wfilt_2d(wavnamenum, Nag_MultiLevel, modenum, m, n, &nwlmax, &nf, &nwct,
               &nwcn, icomm, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_nfilt_2d (c09abc).\n%s\n", fail.message);
    exit_status = 2;
    goto END;
  }
  lenc = nwct;
  if (!(c = NAG_ALLOC(lenc, double)) ||
      !(dwtlvm = NAG_ALLOC(nwlmax, Integer)) ||
      !(dwtlvn = NAG_ALLOC(nwlmax, Integer))
         )
  {
    printf("Allocation failure\n");
    exit_status = -2;
    goto END;
  }

  nwl = nwlmax;

  /* nag_mldwt_2d (c09ecc).
   * Two-dimensional multi-level discrete wavelet transform
   */
  nag_mldwt_2d(m, n, a, pda, 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 = 3;
    goto END;
  }

  /* Print decomposition */
  printf("\n Number of Levels : %" NAG_IFMT "\n", nwl);
  printf(" Number of coefficients in 1st dimension for each level :\n");
  for (j = 0; j < nwl; j++)
    printf("%8" NAG_IFMT "%s", dwtlvm[j], (j + 1) % 8 ? " " : "\n");

  printf("\n Number of coefficients in 2nd dimension for each level :\n");
  for (j = 0; j < nwl; j++)
    printf("%8" NAG_IFMT "%s", dwtlvn[j], (j + 1) % 8 ? " " : "\n");
  printf("\n\nWavelet coefficients C : \n");
  for (ilevel = nwl; ilevel > 0; ilevel -= 1) {
    nwcm = dwtlvm[nwl - ilevel];
    nwcn = dwtlvn[nwl - ilevel];
```

```
    if (!(d = NAG_ALLOC(nwcm * nwcn, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
    for (j = 0; j < 55; j++)
      printf("-");
    printf("\n Level : %" NAG_IFMT "; output is %" NAG_IFMT " by %" NAG_IFMT
           "\n", ilevel, nwcm, nwcn);
    for (j = 0; j < 55; j++)
      printf("-");
    printf("\n");
    fflush(stdout);
    for (itype_coeffs = 0; itype_coeffs <= 3; itype_coeffs++) {
      switch (itype_coeffs) {
      case 0:
#ifdef _WIN32
        if (ilevel == nwl)
          strcpy_s(title, (unsigned)_countof(title),
                   "Approximation coefficients  ");
#else
        if (ilevel == nwl)
          strcpy(title, "Approximation coefficients  ");
#endif
        break;
      case 1:
#ifdef _WIN32
        strcpy_s(title, (unsigned)_countof(title),
                 "Vertical coefficients       ");
#else
        strcpy(title, "Vertical coefficients       ");
#endif
        break;
      case 2:
#ifdef _WIN32
        strcpy_s(title, (unsigned)_countof(title),
                 "Horizontal coefficients     ");
#else
        strcpy(title, "Horizontal coefficients     ");
#endif
        break;
      case 3:
#ifdef _WIN32
        strcpy_s(title, (unsigned)_countof(title),
                 "Diagonal coefficients       ");
#else
        strcpy(title, "Diagonal coefficients       ");
#endif
      }
      if (itype_coeffs > 0 || ilevel == nwl) {
        /* nag_wav_2d_coeff_ext (c09aec).
         * Call the 2D extraction routine c09eac
         */
        nag_wav_2d_coeff_ext(ilevel, itype_coeffs, lenc, c, d, nwcm,
                             icomm, &fail);
        nag_gen_real_mat_print_comp(order, matrix, diag, nwcm, nwcn, d, nwcm,
                                    "%8.4f", title, 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 = 4;
          goto END;
        }
      }
    }
    NAG_FREE(d);
  }

  nwlinv = nwl;
```

```
  /* nag_imldwt_2d (c09edc).
   * Two-dimensional inverse multi-level discrete wavelet transform
   */
  nag_imldwt_2d(nwlinv, lenc, c, m, n, b, pdb, icomm, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_imldwt_2d (c09edc).\n%s\n", fail.message);
    exit_status = 5;
    goto END;
  }

  /* Print reconstruction */
  printf("\n");
  fflush(stdout);
#ifdef _WIN32
  strcpy_s(title, (unsigned)_countof(title),
           "Reconstruction          B :");
#else
  strcpy(title, "Reconstruction          B :");
#endif
  nag_gen_real_mat_print_comp(order, matrix, diag, m, n, b, pdb, "%8.4f",
                              title, 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 = 6;
    goto END;
  }

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

10.2 Program Data

```
nag_mldwt_2d (c09ecc) Example Program Data
  7        8                                                    : m, n
  Nag_Daubechies2   Nag_HalfPointSymmetric                      : wavnam, mode
  3.0000   7.0000   9.0000   1.0000   9.0000   9.0000   1.0000   0.0000
  9.0000   9.0000   3.0000   3.0000   4.0000   1.0000   2.0000   4.0000
  7.0000   8.0000   1.0000   3.0000   8.0000   9.0000   3.0000   3.0000
  1.0000   1.0000   1.0000   1.0000   2.0000   8.0000   4.0000   0.0000
  1.0000   2.0000   4.0000   6.0000   5.0000   6.0000   5.0000   4.0000
  2.0000   2.0000   5.0000   7.0000   3.0000   6.0000   6.0000   8.0000
  7.0000   9.0000   3.0000   1.0000   3.0000   4.0000   7.0000   2.0000
                                                                : a
```

10.3 Program Results

```
nag_mldwt_2d (c09ecc) Example Program Results

 Parameters read from file ::
 MLDWT :: Wavelet  : Nag_Daubechies2
          End mode : Nag_HalfPointSymmetric
          m        : 7
          n        : 8

 Input Data     A :
    3.0000   7.0000   9.0000   1.0000   9.0000   9.0000   1.0000   0.0000
    9.0000   9.0000   3.0000   3.0000   4.0000   1.0000   2.0000   4.0000
    7.0000   8.0000   1.0000   3.0000   8.0000   9.0000   3.0000   3.0000
    1.0000   1.0000   1.0000   1.0000   2.0000   8.0000   4.0000   0.0000
    1.0000   2.0000   4.0000   6.0000   5.0000   6.0000   5.0000   4.0000
```

```
    2.0000   2.0000   5.0000   7.0000   3.0000   6.0000   6.0000   8.0000
    7.0000   9.0000   3.0000   1.0000   3.0000   4.0000   7.0000   2.0000

 Number of Levels : 2
 Number of coefficients in 1st dimension for each level :
       4        5
 Number of coefficients in 2nd dimension for each level :
       4        5

Wavelet coefficients C :
-------------------------------------------------------
 Level : 2; output is 4 by 4
-------------------------------------------------------
 Approximation coefficients
   24.9724  25.6017  20.8900   7.9280
   27.6100  27.0955  18.7941   8.2804
   11.2663  11.0273  19.6410  18.6651
   27.6050  26.6443  14.5913  18.0835
 Vertical coefficients
   -2.5552  -6.1078  -4.0629   8.2136
   -1.6061  -7.2355  -3.3633   7.6075
   -0.2225  -1.6283  -0.5301   3.7415
   -0.9052  -6.5810   0.8023   1.8591
 Horizontal coefficients
   -3.8069  -3.0730   2.1121  -1.8525
   -2.7548  -4.5949  -0.8321  -4.8155
    4.8398   4.5104  -1.5308  -0.6456
   -6.4332  -4.5381   2.4753   6.8224
 Diagonal coefficients
   -0.8978  -0.2326  -1.2515   2.6346
    0.5708  -4.9783  -1.5309   6.4569
   -0.1854  -1.8430   0.2426  -0.0754
    0.0345   7.1864   1.5938  -5.9745
-------------------------------------------------------
 Level : 1; output is 5 by 5
-------------------------------------------------------
 Vertical coefficients
   -2.5981   4.6471   2.5392  -2.8415  -0.2165
   -1.3203  -0.0592   3.0490  -2.5837   1.0458
   -0.4330  -1.6405  -1.1752   0.2533  -2.3448
   -0.4118  -0.0682  -2.4608  -0.0167   0.4387
   -1.5368  -1.1450  -0.5547   4.5936  -3.6863
 Horizontal coefficients
   -4.3301  -1.8170   0.8023   5.7566  -2.8146
    4.3089   3.6908   0.8349   3.4653   1.7108
   -1.5311  -1.0736   1.5257   0.0212  -0.9608
    2.8873   3.1148  -1.9118  -0.4007  -1.5302
   -2.2377  -2.7611   2.4453  -0.3705   4.3448
 Diagonal coefficients
   -1.5000   4.4151  -0.0057  -0.8236  -1.1250
   -0.1953  -2.9530   1.8840  -1.7635   0.9877
   -0.4330   0.2745   1.1450   0.4632  -0.5547
   -0.3538  -0.3215   0.6462   1.3705  -1.2778
    0.7288   0.4587  -1.8873  -1.8828   2.4028

 Reconstruction           B :
    3.0000   7.0000   9.0000   1.0000   9.0000   9.0000   1.0000   0.0000
    9.0000   9.0000   3.0000   3.0000   4.0000   1.0000   2.0000   4.0000
    7.0000   8.0000   1.0000   3.0000   8.0000   9.0000   3.0000   3.0000
    1.0000   1.0000   1.0000   1.0000   2.0000   8.0000   4.0000   0.0000
    1.0000   2.0000   4.0000   6.0000   5.0000   6.0000   5.0000   4.0000
    2.0000   2.0000   5.0000   7.0000   3.0000   6.0000   6.0000   8.0000
    7.0000   9.0000   3.0000   1.0000   3.0000   4.0000   7.0000   2.0000
```