NAG Library Function Document

nag_mldwt_3d (c09fcc)

1 Purpose

nag_mldwt_3d (c09fcc) computes the three-dimensional multi-level discrete wavelet transform (DWT). The initialization function nag_wfilt_3d (c09acc) must be called first to set up the DWT options.

2 Specification

```
#include <nag.h>
#include <nagc09.h>
void nag_mldwt_3d (Integer m, Integer n, Integer fr, const double a[],
     Integer lda, Integer sda, Integer lenc, double c[], Integer nwl,
     Integer dwtlvm[], Integer dwtlvn[], Integer dwtlvfr[], Integer icomm[],
     NagError *fail)
```

3 Description

nag_mldwt_3d (c09fcc) computes the multi-level DWT of three-dimensional data. For a given wavelet and end extension method, nag_mldwt_3d (c09fcc) will compute a multi-level transform of a three-dimensional array 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_3d (c09acc) 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 data, A. Level 1 consists of the first set of coefficients computed: the seven sets of detail coefficients are stored at this level while the approximation 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 eight types of coefficients are stored. All coefficients are packed into a single array.

4 References

Wang Y, Che X and Ma S (2012) Nonlinear filtering based on 3D wavelet transform for MRI denoising *URASIP Journal on Advances in Signal Processing* **2012:40**

5 Arguments

1: **m** – Integer *Input*

On entry: the number of rows of each two-dimensional frame.

Constraint: this must be the same as the value **m** passed to the initialization function nag_wfilt_3d (c09acc).

2: **n** – Integer *Input*

On entry: the number of columns of each two-dimensional frame.

Constraint: this must be the same as the value **n** passed to the initialization function nag_wfilt_3d (c09acc).

3: **fr** – Integer *Input*

On entry: the number of two-dimensional frames.

Constraint: this must be the same as the value **fr** passed to the initialization function nag_wfilt_3d (c09acc).

4: **a**[dim] – const double *Input*

Note: the dimension, *dim*, of the array **a** must be at least **lda** × **sda** × **fr**.

On entry: the m by n by fr three-dimensional input data A, where with A_{ijk} stored in **a**[$(k-1) \times$ **lda** × **sda** $+ (j-1) \times$ **lda** $+ i - 1$].

5: **lda** – Integer *Input*

On entry: the stride separating row elements of each of the sets of frame coefficients in the three-dimensional data stored in **a**.

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

6: **sda** – Integer *Input*

On entry: the stride separating corresponding coefficients of consecutive frames in the three-dimensional data stored in **a**.

Constraint: **sda** $\geq$ **n**.

7: **lenc** – Integer *Input*

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

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

8: **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_3d_coeff_ext (c09fyc) or nag_wav_3d_coeff_ins (c09fzc) is recommended. For completeness the following description provides details of precisely how the coefficients 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 $i = 1, 2, \ldots, n_{\text{fwd}}$, such that $q(i) =$ **dwtlvm**$[n_{\text{fwd}} - i] \times$ **dwtlvn**$[n_{\text{fwd}} - i] \times$ **dwtlvfr**$[n_{\text{fwd}} - i]$. Then, letting $k_1 = q(n_{\text{fwd}})$ and $k_{j+1} = k_j + q(n_{\text{fwd}} - \lceil j/7 \rceil + 1)$, for $j = 1, 2, \ldots, 7n_{\text{fwd}}$, 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, $a_{n_{\text{fwd}}}$. Note that for computational efficiency reasons these coefficients are stored as **dwtlvm**$[0] \times$ **dwtlvn**$[0] \times$ **dwtlvfr**$[0]$ in **c**.

c$[i-1]$, for $i = k_j + 1, \ldots, k_{j+1}$
Contains the level $n_{\text{fwd}} - \lceil j/7 \rceil + 1$ detail coefficients. These are:

LLH coefficients if $j \bmod 7 = 1$;

LHL coefficients if $j \bmod 7 = 2$;

LHH coefficients if $j \bmod 7 = 3$;

HLL coefficients if $j \bmod 7 = 4$;

HLH coefficients if $j \bmod 7 = 5$;

HHL coefficients if $j \bmod 7 = 6$;

HHH coefficients if $j \bmod 7 = 0$,

for $j = 1, \ldots, 7n_{\text{fwd}}$. See Section 2.1 in the c09 Chapter Introduction for a description of how these coefficients are produced.

Note that for computational efficiency reasons these coefficients are stored as **dwtlvfr**$[\lceil j/7 \rceil - 1] \times$ **dwtlvm**$[\lceil j/7 \rceil - 1] \times$ **dwtlvn**$[\lceil j/7 \rceil - 1]$ in **c**.

9: **nwl** – Integer *Input*

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

Constraint: $1 \leq$ **nwl** $\leq l_{\text{max}}$, where l_{max} is the value returned in **nwlmax** (the maximum number of levels) by the call to the initialization function nag_wfilt_3d (c09acc).

10: **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 $(n_{\text{fwd}} - i + 1)$th level of resolution, for $i = 1, 2, \ldots, n_{\text{fwd}}$.

11: **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 $i = 1, 2, \ldots, n_{\text{fwd}}$.

12: **dwtlvfr**[**nwl**] – Integer *Output*

On exit: the number of coefficients in the third dimension for each coefficient type at each level. **dwtlvfr**$[i-1]$ contains the number of coefficients in the third dimension (for each coefficient type computed) at the $(n_{\text{fwd}} - i + 1)$th level of resolution, for $i = 1, 2, \ldots, n_{\text{fwd}}$.

13: **icomm**[**260**] – 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_3d (c09acc).

On exit: contains additional information on the computed transform.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

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

NE_INITIALIZATION

Either the communication array **icomm** has been corrupted or there has not been a prior call to the initialization function nag_wfilt_3d (c09acc).

The initialization function was called with **wtrans** = Nag_SingleLevel.

NE_INT

On entry, **fr** = ⟨*value*⟩.
Constraint: **fr** = ⟨*value*⟩, the value of **fr** on initialization (see nag_wfilt_3d (c09acc)).

On entry, **m** = ⟨*value*⟩.
Constraint: **m** = ⟨*value*⟩, the value of **m** on initialization (see nag_wfilt_3d (c09acc)).

On entry, **n** = ⟨*value*⟩.
Constraint: **n** = ⟨*value*⟩, the value of **n** on initialization (see nag_wfilt_3d (c09acc)).

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

NE_INT_2

On entry, **lda** = ⟨*value*⟩ and **m** = ⟨*value*⟩.
Constraint: **lda** ≥ **m**.

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

On entry, **nwl** = ⟨*value*⟩ and **nwlmax** = ⟨*value*⟩ in nag_wfilt_3d (c09acc).
Constraint: **nwl** ≤ **nwlmax** in nag_wfilt_3d (c09acc).

On entry, **sda** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **sda** ≥ **n**.

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 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction 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

Not applicable.

9 Further Comments

The example program shows how the wavelet coefficients at each level can be extracted from the output array **c**. Denoising can be carried out by applying a thresholding operation to the detail coefficients at every level. If c_{ij} is a detail coefficient then $\hat{c}_{ij} = c_{ij} + \sigma\epsilon_{ij}$ 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_3d (c09fdc) in order to reconstruct the denoised signal. See Section 10 in nag_wav_3d_coeff_ins (c09fzc) 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 computes the three-dimensional multi-level discrete wavelet decomposition for $7 \times 6 \times 5$ input data using the biorthogonal wavelet of order 1.1 (set **wavnam** = Nag_Biorthogonal1_1 in nag_wfilt_3d (c09acc)) with periodic end extension, prints a selected set of wavelet coefficients and then reconstructs and verifies that the reconstruction matches the original data.

10.1 Program Text

```
/* nag_mldwt_3d (c09fcc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagc09.h>
#include <nagx02.h>

#define A(I,J,K) a[I-1 + (J-1)* lda + (K-1)* lda * sda]
#define B(I,J,K) b[I-1 + (J-1)* ldb + (K-1)* ldb * sdb]
#define E(I,J,K) e[I-1 + (J-1)* m + (K-1)* m * n]
#define D(I,J,K) d[I-1 + (J-1)* ldd + (K-1)* ldd * sdd]

int main(void)
{

  /* Scalars */
  Integer         exit_status = 0;
  Integer         lda, ldb, ldd, sda, sdb, sdd, lenc, i, j, k;
  Integer         m, n, fr, nwcfr, nwcm, nwcn, nwct, nwlmax, nwl, nwlinv, nf;
  Integer         want_coeffs, want_level;
  double          eps, esq, frob;
  /* Arrays */
  char            mode[25], wavnam[25];
  double          *a = 0,  *b = 0,  *c = 0,  *d = 0,  *e = 0;
  Integer         *dwtlvfr = 0,  *dwtlvm = 0,  *dwtlvn = 0;
  Integer         icomm[260];
  /* 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_mldwt_3d (c09fcc) Example Program Results\n\n");
  fflush(stdout);

  /* Skip heading in data file and read problem parameters */
#ifdef _WIN32
  scanf_s("%*[^\n] %"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[^\n]", &m, &n, &fr);
#else
  scanf("%*[^\n] %"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[^\n]", &m, &n, &fr);
#endif
  lda = m;
  ldb = m;
  sda = n;
  sdb = n;
#ifdef _WIN32
  scanf_s("%24s%24s%*[^\n]\n", wavnam, _countof(wavnam), mode, _countof(mode));
#else
  scanf("%24s%24s%*[^\n]\n", wavnam, mode);
#endif
```

```
  if (!(a = NAG_ALLOC((lda)*(sda)*(fr), double)) ||
      !(b = NAG_ALLOC((ldb)*(sdb)*(fr), double))||
      !(e = NAG_ALLOC((m)*(n)*(fr), 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  : %4"NAG_IFMT"\n", m);
  printf("         n  : %4"NAG_IFMT"\n", n);
  printf("         fr : %4"NAG_IFMT"\n\n", fr);

  /* 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 */
  for (k=1; k<=fr; k++)
  {
    for (i=1; i<=m; i++)
    {
#ifdef _WIN32
      for (j=1; j<=n; j++) scanf_s("%lf", &A(i, j, k));
#else
      for (j=1; j<=n; j++) scanf("%lf", &A(i, j, k));
#endif
#ifdef _WIN32
      scanf_s("%*[^\n] ");
#else
      scanf("%*[^\n] ");
#endif
    }
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif
  }

  /* Print out the input data */
  printf("Input Data :\n");
  fflush(stdout);
  for (k=1; k<=fr; k++)
  {
    /* nag_gen_real_mat_print_comp (x04cbc).
     * Prints out a matrix.
     */
    nag_gen_real_mat_print_comp(order, matrix, diag, m, n, &A(1, 1, k), lda,
                                "%8.4f"," ", 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);
  }

  /* nag_wfilt_3d (c09acc).
   * Three-dimensional wavelet filter initialization
   */
```

```
  nag_wfilt_3d(wavnamenum, Nag_MultiLevel, modenum, m, n, fr, &nwlmax, &nf,
               &nwct, &nwcn, &nwcfr, icomm, &fail);
  if (fail.code != NE_NOERROR)
  {
    printf("Error from nag_wfilt_3d (c09acc).\n%s\n", fail.message);
    exit_status = 3;
    goto END;
  }

  lenc = nwct;
  if (!(c = NAG_ALLOC((lenc), double)) ||
      !(dwtlvm = NAG_ALLOC((nwlmax), Integer)) ||
      !(dwtlvn = NAG_ALLOC((nwlmax), Integer)) ||
      !(dwtlvfr = NAG_ALLOC((nwlmax), Integer)))
  {
    printf("Allocation failure\n");
    exit_status = 4;
    goto END;
  }

  nwl = nwlmax;

  /* nag_mldwt_3d (c09fcc).
   * Three-dimensional multi-level discrete wavelet transform
   */
  nag_mldwt_3d(m, n, fr, a, lda, sda, lenc, c,
               nwl, dwtlvm, dwtlvn, dwtlvfr, icomm, &fail);
  if (fail.code != NE_NOERROR)
  {
    printf("Error from nag_mldwt_3d (c09fcc).\n%s\n", fail.message);
    exit_status = 5;
    goto END;
  }

  printf("Number of Levels : %4"NAG_IFMT"\n", nwl);
  printf("Number of coefficients in 1st dimension for each level:\n");
  for (i=0; i<nwl; i++)
  {
    printf("%4"NAG_IFMT"%s", dwtlvm[i], i+1%8 ? "" : "\n");
  }
  printf("\n");
  printf("Number of coefficients in 2nd dimension for each level:\n");
  for (i=0; i<nwl; i++)
  {
    printf("%4"NAG_IFMT"%s", dwtlvn[i], i+1%8 ? "" : "\n");
  }
  printf("\n");
  printf("Number of coefficients in 3rd dimension for each level:\n");
  for (i=0; i<nwl; i++)
  {
    printf("%4"NAG_IFMT"%s", dwtlvfr[i], i+1%8 ? "" : "\n");
  }
  printf("\n\n");
  fflush(stdout);

  /* Print the first level HLL coefficients*/
  want_level = 1;
  want_coeffs = 4;

  /* Use the extraction routine c09fyc to retrieve the required
   * coefficients.
   */
  nwcm = dwtlvm[nwl - want_level];
  nwcn = dwtlvn[nwl - want_level];
  nwcfr = dwtlvfr[nwl - want_level];
  ldd = nwcm;
  sdd = nwcn;
  if (!(d = NAG_ALLOC((ldd)*(sdd)*(nwcfr), double)))
  {
    printf("Allocation failure\n");
    exit_status = 6;
```

```
  goto END;
}

/* nag_wav_3d_coeff_ext (c09fyc).
 * Extract coefficients into a 3D array D.
 */
nag_wav_3d_coeff_ext(want_level,want_coeffs,lenc,c,d,ldd,sdd,icomm,&fail);
if (fail.code != NE_NOERROR)
{
  printf("Error from nag_wav_3d_coeff_ext (c09fyc).\n%s\n", fail.message);
  exit_status = 7;
  goto END;
}

/* Print the details of the level */
printf("-----------------------------------------------------\n");
printf("Level : %4"NAG_IFMT"", want_level);
printf("; output is %4"NAG_IFMT"", nwcm);
printf(" by %4"NAG_IFMT"", nwcn);
printf(" by %4"NAG_IFMT"\n", nwcfr);
printf("-----------------------------------------------------\n\n");

/* Print out the selected set of coefficients*/
switch (want_coeffs)
{
  case 0:
    printf("Approximation coefficients (LLL)\n");
    break;
  case 1:
    printf("Detail coefficients (LLH)\n");
    break;
  case 2:
    printf("Detail coefficients (LHL)\n");
    break;
  case 3:
    printf("Detail coefficients (LHH)\n");
    break;
  case 4:
    printf("Detail coefficients (HLL)\n");
    break;
  case 5:
    printf("Detail coefficients (HLH)\n");
    break;
  case 6:
    printf("Detail coefficients (HHL)\n");
    break;
  case 7:
    printf("Detail coefficients (HHH)\n");
    break;
}

printf("Level %4"NAG_IFMT"", want_level);
printf(", Coefficients %4"NAG_IFMT":\n", want_coeffs);
for (k=1; k<=nwcfr; k++)
{
  printf(" Frame %4"NAG_IFMT" :\n", k);
  for (i=1; i<=nwcm; i++)
  {
    for (j=1; j<=nwcn; j++)
    {
      printf("%8.4f%s", D(i, j, k), j%8 ? "" : "\n");
    }
    printf("\n");
  }
}
fflush(stdout);

nwlinv = nwl;

/* nag_imldwt_3d (c09fdc).
 * Three-dimensional inverse multi-level discrete wavelet transform
```

```
   */
  nag_imldwt_3d(nwlinv, lenc, c, m, n, fr, b, ldb, sdb, icomm, &fail);
  if (fail.code != NE_NOERROR)
  {
    printf("Error from nag_imldwt_3d (c09fdc).\n%s\n", fail.message);
    exit_status = 8;
    goto END;
  }

  /* Check reconstruction matches original*/
  eps = 10.0 * (double)(m) * (double)(n) * (double)(fr) *
        nag_machine_precision;

  for (k=1; k<=fr; k++)
    for (j=1; j<=n; j++)
      for (i=1; i<=m; i++)
        E(i, j, k) =  B(i, j, k) - A(i, j, k);

  frob = 0.0;
  for (k=1; k<=fr; k++)
  {
    esq = 0.0;
    for (j=1; j<=n; j++)
    {
      for (i=1; i<=m; i++)
      {
        esq = esq + pow(E(i, j, k), 2);
      }
      frob = MAX(frob, sqrt(esq));
    }
  }

  if (frob>eps)
  {
    printf("\nFail: Frobenius norm of B-A, where A is the original \n"
           "data and B is the reconstrucion, is too large.\n");
  }
  else
  {
    printf("\nSuccess: the reconstruction matches the original.\n");
  }

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

10.2 Program Data

```
nag_mldwt_3d (c09fcc) Example Program Data
 7 6 5        : m, n, fr
 Nag_Biorthogonal1_1 Nag_Periodic : wavnam, mode
 3.0000  2.0000  2.0000  2.0000  1.0000  1.0000
 2.0000  9.0000  1.0000  2.0000  1.0000  3.0000
 2.0000  5.0000  1.0000  2.0000  1.0000  1.0000
 1.0000  6.0000  2.0000  2.0000  7.0000  2.0000
 5.0000  3.0000  2.0000  2.0000  4.0000  7.0000
 2.0000  2.0000  1.0000  1.0000  2.0000  1.0000
 6.0000  2.0000  1.0000  3.0000  6.0000  9.0000

 2.0000  1.0000  5.0000  1.0000  2.0000  3.0000
 2.0000  9.0000  5.0000  2.0000  1.0000  2.0000
 2.0000  3.0000  2.0000  7.0000  1.0000  1.0000
```

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

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

 5.0000  8.0000  1.0000  2.0000  2.0000  1.0000
 1.0000  2.0000  2.0000  9.0000  2.0000  9.0000
 2.0000  2.0000  2.0000  1.0000  1.0000  3.0000
 1.0000  1.0000  1.0000  5.0000  1.0000  2.0000
 3.0000  2.0000  8.0000  1.0000  9.0000  2.0000
 2.0000  1.0000  9.0000  1.0000  2.0000  2.0000
 3.0000  6.0000  5.0000  3.0000  2.0000  2.0000

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

10.3 Program Results

```
nag_mldwt_3d (c09fcc) Example Program Results

Parameters read from file ::
MLDWT :: Wavelet  : Nag_Biorthogonal1_1
         End mode : Nag_Periodic
         m  :    7
         n  :    6
         fr :    5

Input Data :
    3.0000   2.0000   2.0000   2.0000   1.0000   1.0000
    2.0000   9.0000   1.0000   2.0000   1.0000   3.0000
    2.0000   5.0000   1.0000   2.0000   1.0000   1.0000
    1.0000   6.0000   2.0000   2.0000   7.0000   2.0000
    5.0000   3.0000   2.0000   2.0000   4.0000   7.0000
    2.0000   2.0000   1.0000   1.0000   2.0000   1.0000
    6.0000   2.0000   1.0000   3.0000   6.0000   9.0000

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

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

    5.0000   8.0000   1.0000   2.0000   2.0000   1.0000
    1.0000   2.0000   2.0000   9.0000   2.0000   9.0000
    2.0000   2.0000   2.0000   1.0000   1.0000   3.0000
    1.0000   1.0000   1.0000   5.0000   1.0000   2.0000
    3.0000   2.0000   8.0000   1.0000   9.0000   2.0000
```

```
    2.0000   1.0000   9.0000   1.0000   2.0000   2.0000
    3.0000   6.0000   5.0000   3.0000   2.0000   2.0000

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

Number of Levels :    2
Number of coefficients in 1st dimension for each level:
   2   4
Number of coefficients in 2nd dimension for each level:
   2   3
Number of coefficients in 3rd dimension for each level:
   2   3

-------------------------------------------------------
Level :    1; output is    4 by    3 by    3
-------------------------------------------------------

Detail coefficients (HLL)
Level    1, Coefficients    4:
 Frame    1 :
 -4.9497  0.0000  0.0000
  0.7071  1.7678 -3.1820
  0.7071  2.1213  1.7678
  0.0000  0.0000  0.0000
 Frame    2 :
  4.2426 -2.1213 -4.9497
  0.7071 -0.0000 -0.7071
 -1.4142 -3.1820  1.4142
  0.0000  0.0000  0.0000
 Frame    3 :
  2.1213 -4.9497 -0.7071
 -2.8284 -4.2426  4.9497
  2.1213  2.8284 -0.7071
  0.0000  0.0000  0.0000

Success: the reconstruction matches the original.
```