NAG Library Function Document

nag_dwt_2d (c09eac)

1 Purpose

nag_dwt_2d (c09eac) computes the two-dimensional discrete wavelet transform (DWT) at a single level. 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_dwt_2d (Integer m, Integer n, const double a[], Integer lda,
     double ca[], Integer ldca, double ch[], Integer ldch, double cv[],
     Integer ldcv, double cd[], Integer ldcd, Integer icomm[],
     NagError *fail)
```

3 Description

nag_dwt_2d (c09eac) computes the two-dimensional DWT of a given input data array, considered as a matrix A, at a single level. For a chosen wavelet filter pair, the output coefficients are obtained by applying convolution and downsampling by two to the input, A, first over columns and then to the result over rows. The matrix of approximation (or smooth) coefficients, C_a, is produced by the low pass filter over columns and rows; the matrix of horizontal coefficients, C_h, is produced by the high pass filter over columns and the low pass filter over rows; the matrix of vertical coefficients, C_v, is produced by the low pass filter over columns and the high pass filter over rows; and the matrix of diagonal coefficients, C_d, is produced by the high pass filter over columns and rows. To reduce distortion effects at the ends of the data array, several end extension methods are commonly used. Those provided are: periodic or circular convolution end extension, half-point symmetric end extension, whole-point symmetric end extension and zero end extension. The total number, n_{ct}, of coefficients computed for C_a, C_h, C_v, and C_d together and the number of columns of each coefficients matrix, n_{cn}, are returned by the initialization function nag_wfilt_2d (c09abc). These values can be used to calculate the number of rows of each coefficients matrix, n_{cm}, using the formula $n_{\mathrm{cm}} = n_{\mathrm{ct}}/(4n_{\mathrm{cn}})$.

4 References

Daubechies I (1992) *Ten Lectures on Wavelets* SIAM, Philadelphia

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: **ca**$[dim]$ – double *Output*

Note: the dimension, *dim*, of the array **ca** must be at least **ldca** $\times n_{\mathrm{cn}}$ where n_{cn} is the argument **nwcn** returned by function nag_wfilt_2d (c09abc).

The (i, j)th element of the matrix is stored in **ca**$[(j - 1) \times$ **ldca** $+ i - 1]$.

On exit: contains the n_{cm} by n_{cn} matrix of approximation coefficients, C_a.

6: **ldca** – Integer *Input*

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

Constraint: **ldca** $\geq n_{\mathrm{cm}}$ where $n_{\mathrm{cm}} = n_{\mathrm{ct}}/(4n_{\mathrm{cn}})$ and n_{cn}, n_{ct} are returned by the initialization function nag_wfilt_2d (c09abc).

7: **ch**$[dim]$ – double *Output*

Note: the dimension, *dim*, of the array **ch** must be at least **ldch** $\times n_{\mathrm{cn}}$ where n_{cn} is the argument **nwcn** returned by function nag_wfilt_2d (c09abc).

The (i, j)th element of the matrix is stored in **ch**$[(j - 1) \times$ **ldch** $+ i - 1]$.

On exit: contains the n_{cm} by n_{cn} matrix of horizontal coefficients, C_h.

8: **ldch** – Integer *Input*

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

Constraint: **ldch** $\geq n_{\mathrm{cm}}$ where $n_{\mathrm{cm}} = n_{\mathrm{ct}}/(4n_{\mathrm{cn}})$ and n_{cn}, n_{ct} are returned by the initialization function nag_wfilt_2d (c09abc).

9: **cv**$[dim]$ – double *Output*

Note: the dimension, *dim*, of the array **cv** must be at least **ldcv** $\times n_{\mathrm{cn}}$ where n_{cn} is the argument **nwcn** returned by function nag_wfilt_2d (c09abc).

The (i, j)th element of the matrix is stored in **cv**$[(j - 1) \times$ **ldcv** $+ i - 1]$.

On exit: contains the n_{cm} by n_{cn} matrix of vertical coefficients, C_v.

10: **ldcv** – Integer *Input*

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

Constraint: **ldcv** $\geq n_{\mathrm{cm}}$ where $n_{\mathrm{cm}} = n_{\mathrm{ct}}/(4n_{\mathrm{cn}})$ and n_{cn}, n_{ct} are returned by the initialization function nag_wfilt_2d (c09abc).

11: **cd**$[dim]$ – double *Output*

Note: the dimension, *dim*, of the array **cd** must be at least **ldcd** $\times n_{\mathrm{cn}}$ where n_{cn} is the argument **nwcn** returned by function nag_wfilt_2d (c09abc).

The (i, j)th element of the matrix is stored in **cd**$[(j - 1) \times$ **ldcd** $+ i - 1]$.

On exit: contains the n_{cm} by n_{cn} matrix of diagonal coefficients, C_d.

12: **ldcd** – Integer *Input*

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

Constraint: **ldcd** $\geq n_{\mathrm{cm}}$ where $n_{\mathrm{cm}} = n_{\mathrm{ct}}/(4n_{\mathrm{cn}})$ and n_{cn}, n_{ct} are returned by the initialization function nag_wfilt_2d (c09abc).

13: **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).

14: **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 3.2.1.2 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_MultiLevel or **icomm** has been corrupted.

NE_INT

On entry, **ldca** = ⟨*value*⟩.
Constraint: **ldca** ≥ ⟨*value*⟩, the number of wavelet coefficients in the first dimension.

On entry, **ldcd** = ⟨*value*⟩.
Constraint: **ldcd** ≥ ⟨*value*⟩, the number of wavelet coefficients in the first dimension.

On entry, **ldch** = ⟨*value*⟩.
Constraint: **ldch** ≥ ⟨*value*⟩, the number of wavelet coefficients in the first dimension.

On entry, **ldcv** = ⟨*value*⟩.
Constraint: **ldcv** ≥ ⟨*value*⟩, the number of wavelet coefficients in the first dimension.

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)).

NE_INT_2

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

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 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 3.6.5 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_dwt_2d (c09eac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Notefor your implementation for any additional implementation-specific information.

9 Further Comments

None.

10 Example

This example computes the two-dimensional discrete wavelet decomposition for a 6×6 input matrix using the Daubechies wavelet, **wavnam** $=$ Nag_Daubechies4, with half point symmetric end extension.

10.1 Program Text

```
/* nag_dwt_2d (c09eac) 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 i, j, m, n, nf, nwcm, nwcn, nwct, nwl, pda, pdb, pdc;
  /* Arrays */
  char mode[24], wavnam[20], title[50];
  double *a = 0, *b = 0, *ca = 0, *cd = 0, *ch = 0, *cv = 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);

  printf("nag_dwt_2d (c09eac) 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
  pda = m;
  pdb = m;
#ifdef _WIN32
  scanf_s("%19s%23s%*[^\n]\n", wavnam, (unsigned)_countof(wavnam), mode,
          (unsigned)_countof(mode));
#else
  scanf("%19s%23s%*[^\n]\n", wavnam, mode);
#endif
  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(" DWT :: Wavelet : %s\n", wavnam);
  printf("       End mode: %s\n", mode);
  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 */
#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
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  printf("\n");
  fflush(stdout);
  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;
  }
  printf("\n");

  /* nag_wfilt_2d (c09abc).
   * Two-dimensional wavelet filter initialization
   */
  nag_wfilt_2d(wavnamenum, Nag_SingleLevel, 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 = 2;
    goto END;
  }
  nwcm = nwct / (4 * nwcn);
  if (!(ca = NAG_ALLOC((nwcm) * (nwcn), double)) ||
      !(cd = NAG_ALLOC((nwcm) * (nwcn), double)) ||
      !(cv = NAG_ALLOC((nwcm) * (nwcn), double)) ||
      !(ch = NAG_ALLOC((nwcm) * (nwcn), double))
         )
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  pdc = nwcm;
  /* nag_dwt_2d (c09eac).
   * Two-dimensional discrete wavelet transform
   */
  nag_dwt_2d(m, n, a, pda, ca, pdc, ch, pdc, cv, pdc, cd, pdc, icomm, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dwt_2d (c09eac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  fflush(stdout);

  /* Print decomposition */
#ifdef _WIN32
  strcpy_s(title, (unsigned)_countof(title),
           "Approximation coefficients   CA :");
#else
  strcpy(title, "Approximation coefficients   CA :");
#endif
  nag_gen_real_mat_print_comp(order, matrix, diag, nwcm, nwcn, ca, pdc,
                              "%8.4f",
                              title, Nag_NoLabels, 0, Nag_NoLabels, 0, 80, 0,
                              0, &fail);
  printf("\n");
  fflush(stdout);
#ifdef _WIN32
  strcpy_s(title, (unsigned)_countof(title),
           "Diagonal coefficients        CD :");
#else
  strcpy(title, "Diagonal coefficients        CD :");
#endif
  nag_gen_real_mat_print_comp(order, matrix, diag, nwcm, nwcn, cd, pdc,
                              "%8.4f",
                              title, Nag_NoLabels, 0, Nag_NoLabels, 0, 80, 0,
                              0, &fail);
  printf("\n");
  fflush(stdout);
#ifdef _WIN32
  strcpy_s(title, (unsigned)_countof(title),
           "Horizontal coefficients      CH :");
#else
  strcpy(title, "Horizontal coefficients      CH :");
#endif
  nag_gen_real_mat_print_comp(order, matrix, diag, nwcm, nwcn, ch, pdc,
                              "%8.4f",
                              title, Nag_NoLabels, 0, Nag_NoLabels, 0, 80, 0,
                              0, &fail);
  printf("\n");
  fflush(stdout);
#ifdef _WIN32
  strcpy_s(title, (unsigned)_countof(title),
           "Vertical coefficients        CV :");
#else
```

```
  strcpy(title, "Vertical coefficients        CV :");
#endif
  nag_gen_real_mat_print_comp(order, matrix, diag, nwcm, nwcn, cv, pdc,
                              "%8.4f",
                              title, Nag_NoLabels, 0, Nag_NoLabels, 0, 80, 0,
                              0, &fail);
  printf("\n");

  /* nag_idwt_2d (c09ebc).
   * Two-dimensional inverse discrete wavelet transform
   */
  nag_idwt_2d(m, n, ca, pdc, ch, pdc, cv, pdc, cd, pdc, b, pdb, icomm, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_idwt_2d (c09ebc).\n%s\n", fail.message);
    exit_status = 3;
    goto END;
  }
  fflush(stdout);

  /* Print reconstruction */
#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);
END:
  NAG_FREE(a);
  NAG_FREE(b);
  NAG_FREE(ca);
  NAG_FREE(cd);
  NAG_FREE(ch);
  NAG_FREE(cv);
  return exit_status;
}
```

10.2 Program Data

```
nag_dwt_2d (c09eac) Example Program Data
  6                 6                                 : m,n
  Nag_Daubechies4   Nag_HalfPointSymmetric            : wavnam, mode
  8.0000   7.0000   3.0000   3.0000   1.0000   1.0000
  4.0000   6.0000   1.0000   5.0000   2.0000   9.0000
  8.0000   1.0000   4.0000   9.0000   3.0000   7.0000
  9.0000   3.0000   8.0000   2.0000   4.0000   3.0000
  1.0000   3.0000   7.0000   1.0000   5.0000   2.0000
  4.0000   3.0000   7.0000   7.0000   6.0000   1.0000 :a
```

10.3 Program Results

```
nag_dwt_2d (c09eac) Example Program Results

 Parameters read from file ::
 DWT :: Wavelet : Nag_Daubechies4
        End mode: Nag_HalfPointSymmetric

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

 Approximation coefficients   CA :
    6.3591  10.3477   8.0995  10.3210   8.7587   3.5783
```

```
  11.5754   6.3762  12.1704   7.4521   8.6977  14.8535
   2.0630   8.4499  15.4726  12.1764   3.8920   2.7112
  10.2143   6.2445  13.8571   8.1060   7.7701  13.2127
   6.3353   8.7805  10.2727  10.0472   6.8614   7.5814
  11.7141  11.1018   5.2923   8.1272  14.5540   2.5729

Diagonal coefficients       CD :
   0.4777   1.0230  -0.3147   0.0625   0.0831  -1.3316
   1.0689   1.5671  -2.1422   0.5565   1.7593  -2.8097
  -0.9555  -1.9276   0.9195  -0.2228  -0.5125   2.6989
   0.2899   0.4453  -0.5695   0.1541   0.4749  -0.7946
   0.4944   1.4145   0.3488  -0.1187  -0.6212  -1.5177
  -1.3753  -2.5224   1.7581  -0.4316  -1.1835   3.7547

Horizontal coefficients     CH :
   0.4100  -0.1827   1.5354   0.0784   0.8101  -1.3594
   2.3496  -0.9422   2.3780  -1.0540   2.7743  -2.2648
  -1.2690   0.0152  -6.9338  -1.7435  -1.6917   1.2388
   0.6317  -0.0969   2.3300   0.4637   0.6365  -0.1162
  -0.2343   0.3923   5.5457   2.1818   0.2103  -0.8573
  -1.8880   0.8142  -4.8552   0.0736  -2.7395   3.3590

Vertical coefficients       CV :
   1.5365   5.9678   3.4309  -1.0585  -5.0275  -4.8492
   0.6779  -0.0294  -5.3274   1.6483   4.8689  -1.8383
  -1.1065  -2.8791   0.1535   0.0982   0.8417   2.8923
  -0.1359  -2.6633  -5.8549   1.8440   6.2403   0.5697
   1.4244   5.2140   1.6410  -0.4669  -3.2369  -4.5757
   1.0288   2.2521   0.0574  -0.1359  -0.5170  -2.6854

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