

# NAG Library Function Document

## nag\_mldwt (c09ccc)

### 1 Purpose

nag\_mldwt (c09ccc) computes the one-dimensional multi-level discrete wavelet transform (DWT). The initialization function nag\_wfilt (c09aac) must be called first to set up the DWT options.

### 2 Specification

```
#include <nag.h>
#include <nagc09.h>

void nag_mldwt (Integer n, const double x[], Integer lenc, double c[],
               Integer nwl, Integer dwtlev[], Integer icomm[], NagError *fail)
```

### 3 Description

nag\_mldwt (c09ccc) computes the multi-level DWT of one-dimensional data. For a given wavelet and end extension method, nag\_mldwt (c09ccc) will compute a multi-level transform of a data array,  $x_i$ , for  $i = 1, 2, \dots, n$ , using a specified number,  $n_{\text{fwd}}$ , of levels. The number of levels specified,  $n_{\text{fwd}}$ , must be no more than the value  $l_{\text{max}}$  returned in **nwlmax** by the initialization function nag\_wfilt (c09aac) 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 dataset,  $x$ , with level 1 being the first set of coefficients computed, with the detail coefficients,  $d_1$ , being stored while the approximation coefficients,  $a_1$ , are used as the input to a repeat of the wavelet transform. This process is continued until, at level  $n_{\text{fwd}}$ , both the detail coefficients,  $d_{n_{\text{fwd}}}$ , and the approximation coefficients,  $a_{n_{\text{fwd}}}$  are retained. The output array,  $C$ , stores these sets of coefficients in reverse order, starting with  $a_{n_{\text{fwd}}}$  followed by  $d_{n_{\text{fwd}}}, d_{n_{\text{fwd}}-1}, \dots, d_1$ .

### 4 References

None.

### 5 Arguments

- 1: **n** – Integer *Input*  
*On entry:* the number of elements,  $n$ , in the data array  $x$ .  
*Constraint:* this must be the same as the value **n** passed to the initialization function nag\_wfilt (c09aac).
- 2: **x[n]** – const double *Input*  
*On entry:* **x** contains the one-dimensional input dataset  $x_i$ , for  $i = 1, 2, \dots, n$ .
- 3: **lenc** – Integer *Input*  
*On entry:* the dimension of the array **c**. **c** must be large enough to contain the number,  $n_c$ , of wavelet coefficients. The maximum value of  $n_c$  is returned in **nwc** by the call to the initialization function nag\_wfilt (c09aac) and corresponds to the DWT being continued for the maximum number of levels possible for the given data set. When the number of levels,  $n_{\text{fwd}}$ , is chosen to be less than the maximum, then  $n_c$  is correspondingly smaller and **lenc** can be reduced by noting that the number of coefficients at each level is given by  $\lceil \bar{n}/2 \rceil$  for **mode** = Nag\_Periodic in nag\_wfilt

(c09aac) and  $\lfloor (\bar{n} + n_f - 1)/2 \rfloor$  for **mode** = Nag\_HalfPointSymmetric, Nag\_WholePointSymmetric or Nag\_ZeroPadded, where  $\bar{n}$  is the number of input data at that level and  $n_f$  is the filter length provided by the call to nag\_wfilt (c09aac). At the final level the storage is doubled to contain the set of approximation coefficients.

*Constraint:* **lenc**  $\geq n_c$ , where  $n_c$  is the number of approximation and detail coefficients that correspond to a transform with **nwlmax** levels.

- 4: **c[lenc]** – double *Output*
- On exit:* let  $q(i)$  denote the number of coefficients (of each type) produced by the wavelet transform at level  $i$ , for  $i = n_{\text{fwd}}, n_{\text{fwd}} - 1, \dots, 1$ . These values are returned in **dwtlev**. Setting  $k_1 = q(n_{\text{fwd}})$  and  $k_{j+1} = k_j + q(n_{\text{fwd}} - j + 1)$ , for  $j = 1, 2, \dots, n_{\text{fwd}}$ , the coefficients are stored as follows:
- c[i - 1]**, for  $i = 1, 2, \dots, k_1$   
 Contains the level  $n_{\text{fwd}}$  approximation coefficients,  $a_{n_{\text{fwd}}}$ .
- c[i - 1]**, for  $i = k_1 + 1, \dots, k_2$   
 Contains the level  $n_{\text{fwd}}$  detail coefficients  $d_{n_{\text{fwd}}}$ .
- c[i - 1]**, for  $i = k_j + 1, \dots, k_{j+1}$   
 Contains the level  $n_{\text{fwd}} - j + 1$  detail coefficients, for  $j = 2, 3, \dots, n_{\text{fwd}}$ .
- 5: **nwl** – Integer *Input*
- On entry:* the number of levels,  $n_{\text{fwd}}$ , in the multi-level resolution to be performed.
- Constraint:*  $1 \leq \mathbf{nwl} \leq l_{\text{max}}$ , where  $l_{\text{max}}$  is the value returned in **nwlmax** (the maximum number of levels) by the call to the initialization function nag\_wfilt (c09aac).
- 6: **dwtlev[nwl + 1]** – Integer *Output*
- On exit:* the number of transform coefficients at each level. **dwtlev[0]** and **dwtlev[1]** contain the number,  $q(n_{\text{fwd}})$ , of approximation and detail coefficients respectively, for the final level of resolution (these are equal); **dwtlev[i - 1]** contains the number of detail coefficients,  $q(n_{\text{fwd}} - i + 2)$ , for the  $(n_{\text{fwd}} - i + 2)$ th level, for  $i = 3, 4, \dots, n_{\text{fwd}} + 1$ .
- 7: **icomm[100]** – 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 (c09aac).
- On exit:* contains additional information on the computed transform.
- 8: **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.

### NE\_ARRAY\_DIM\_LEN

On entry, **lenc** is set too small: **lenc** =  $\langle \text{value} \rangle$ .  
 Constraint: **lenc**  $\geq \langle \text{value} \rangle$ .

### NE\_BAD\_PARAM

On entry, argument  $\langle \text{value} \rangle$  had an illegal value.

**NE\_INITIALIZATION**

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

Either the initialization function was called with **wtrans** = Nag\_SingleLevel or array **icomm** has been corrupted.

On entry, **n** is inconsistent with the value passed to the initialization function: **n** =  $\langle value \rangle$ , **n** should be  $\langle value \rangle$ .

On entry, **nwl** is larger than the maximum number of levels returned by the initialization function: **nwl** =  $\langle value \rangle$ , maximum =  $\langle value \rangle$ .

**NE\_INT**

On entry, **nwl** =  $\langle value \rangle$ .

Constraint: **nwl**  $\geq$  1.

**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.

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

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

taking  $\bar{d}_{ij}$  to be an approximation to the required detail coefficient without noise,  $d_{ij}$ . The resulting coefficients can then be used as input to `nag_imldwt` (c09cdc) in order to reconstruct the denoised signal.

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 of a dataset using the Daubechies wavelet (see **wavnam** = Nag\_Daubechies4 in `nag_wfilt` (c09aac)) using zero end extensions, the number of levels of resolution, the number of coefficients in each level and the coefficients themselves are reused. The original dataset is then reconstructed using `nag_imldwt` (c09cdc).

## 10.1 Program Text

```

/* nag_mldwt (c09ccc) Example Program.
 *
 * Copyright 2008, Numerical Algorithms Group.
 *
 * Mark 9, 2009.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagc09.h>

int main(void)
{
    /* Constants */
    Integer    licomm = 100;
    /*Integer scalar and array declarations */
    Integer    exit_status = 0;
    Integer    i, n, nf, nnz, nwc, nwlmax, nwl, nwlinv;
    Integer    *dwtlev = 0, *icomm = 0;
    NagError   fail;
    Nag_Wavelet    wavnamenum;
    Nag_WaveletMode    modenum;
    /*Double scalar and array declarations */
    double     *c = 0, *x = 0, *y = 0;
    /*Character scalar and array declarations */
    char       mode[24], wavnam[20];

    INIT_FAIL(fail);

    printf("nag_mldwt (c09ccc) Example Program Results\n\n");
    fflush(stdout);

    /*      Skip heading in data file*/
    scanf("%*[\n] ");
    /*      Read n - length of input data sequence*/
    scanf("%ld%*[\n] ", &n);
    if (!(x = NAG_ALLOC(n, double)) ||
        !(y = NAG_ALLOC(n, double)) ||
        !(icomm = NAG_ALLOC(licomm, Integer)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    /*      Read Wavelet name (wavnam) and end mode (mode)*/
    scanf("%19s%23s%*[\n] ", wavnam, mode);
    /*
     * 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);
    if (n >= 2)
    {
        printf("MLDWT :: \n");
        printf("      Wavelet   :%16s\n", wavnam);
        printf("      End mode   :%16s\n", mode);
        printf("      N          :%16ld\n\n", n);
        /*      Read data array and write it out*/
        printf("%s\n", " Input Data      X :");
        for (i = 0; i < n; i++)
        {
            scanf("%lf", &x[i]);
            printf("%8.4f%s", x[i], (i+1)%8?" ":"\n");
        }
        printf("\n");
    }
}

```

```

/*
 * nag_wfilt (c09aac)
 * Wavelet filter query
 */
nag_wfilt(wavnamenum, Nag_MultiLevel, modenum, n, &nwlmax, &nf, &nwc,
          icomm, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_wfilt (c09aac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
if (!(c = NAG_ALLOC(nwc, double)) ||
    !(dwtlev = NAG_ALLOC(nwc, Integer)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
nwl = nwlmax;
/*      Perform Discrete Wavelet transform*/
/*
 * nag_mldwt (c09ccc)
 * one-dimensional multi-level discrete wavelet transform (mldwt)
 */
nag_mldwt(n, x, nwc, c, nwl, dwtlev, icomm, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_mldwt (c09ccc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
printf("  Number of Levels : %20ld\n", nwl);
printf("  Number of coefficients in each level : \n");
for (i = 0; i < nwl+1; i++)
    printf("%8ld%s", dwtlev[i], (i+1)%8?" ":"\n");
printf("\n\n");
nnz = 0;
for (i = 0; i < nwl+1; i++)
    nnz = nnz+dwtlev[i];
printf("  Wavelet coefficients C : \n");
for (i = 0; i < nnz; i++)
    printf("%8.4f%s", c[i], (i+1)%8?" ":"\n");
printf("\n\n");
/*      Reconstruct original data*/
nwlinv = nwl;
/*
 * nag_imldwt (c09cdc)
 * one-dimensional inverse multi-level discrete wavelet transform
 * (imldwt)
 */
nag_imldwt(nwlinv, nwc, c, n, y, icomm, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_imldwt (c09cdc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
printf("  Reconstruction                Y : \n");
for (i = 0; i < n; i++)
    printf("%8.4f%s", y[i], (i+1)%8?" ":"\n");
printf("\n");
}

END:
NAG_FREE(c);
NAG_FREE(x);
NAG_FREE(y);
NAG_FREE(dwtlev);
NAG_FREE(icomm);

```

```

    return exit_status;
}

```

## 10.2 Program Data

```

nag_mldwt (c09ccc) Example Program Data
64                                     : n
Nag_Daubechies4 Nag_ZeroPadded      : wavnam, mode
6.5271 6.512 6.5016 6.5237 6.4625
6.3496 6.4025 6.4035 6.4407 6.4746
6.5095 6.6551 6.61 6.5969 6.6083
6.652 6.7113 6.7227 6.7196 6.7649
6.7794 6.8037 6.8308 6.7712 6.7067
6.769 6.7068 6.7024 6.6463 6.6098
6.59 6.596 6.5457 6.547 6.5797
6.5895 6.6275 6.6795 6.6598 6.6925
6.6873 6.7223 6.7205 6.6843 6.703
6.647 6.6008 6.6061 6.6097 6.6485
6.6394 6.6571 6.6357 6.6224 6.6073
6.6075 6.6379 6.6294 6.5906 6.6258
6.6369 6.6515 6.6826 6.7042      : X(1:n)

```

## 10.3 Program Results

nag\_mldwt (c09ccc) Example Program Results

MLDWT ::

```

Wavelet : Nag_Daubechies4
End mode : Nag_ZeroPadded
N       : 64

```

```

Input Data      X :
6.5271 6.5120 6.5016 6.5237 6.4625 6.3496 6.4025 6.4035
6.4407 6.4746 6.5095 6.6551 6.6100 6.5969 6.6083 6.6520
6.7113 6.7227 6.7196 6.7649 6.7794 6.8037 6.8308 6.7712
6.7067 6.7690 6.7068 6.7024 6.6463 6.6098 6.5900 6.5960
6.5457 6.5470 6.5797 6.5895 6.6275 6.6795 6.6598 6.6925
6.6873 6.7223 6.7205 6.6843 6.7030 6.6470 6.6008 6.6061
6.6097 6.6485 6.6394 6.6571 6.6357 6.6224 6.6073 6.6075
6.6379 6.6294 6.5906 6.6258 6.6369 6.6515 6.6826 6.7042

```

```

Number of Levels : 6
Number of coefficients in each level :
7       7       8       10      14      21      35

```

```

Wavelet coefficients C :
0.0000 -0.0227 -0.3446 2.7574 -10.1970 44.8800 15.9443 0.0010
-0.4881 -10.2673 11.3258 -1.7469 2.0785 -0.7334 -0.0054 -0.1402
-5.8980 -1.1527 5.5613 2.1352 0.3203 -0.4004 0.0010 0.5229
0.5055 -2.7274 -0.0911 -0.2806 -0.3669 2.9467 -0.3799 -0.1552
0.0218 0.0922 5.4626 -2.1620 0.5196 -0.0287 -0.0199 0.0920
-0.0134 -0.1298 -5.5168 2.3105 -0.5383 -0.0155 0.3057 0.6186
-1.5542 0.2682 0.1566 0.0030 -0.0152 -0.0589 0.0126 0.0063
0.0171 -0.0268 0.0077 -0.0189 0.0207 0.0104 -0.3207 -0.6062
1.6288 -0.2414 -0.0671 3.1657 -1.1462 0.2785 0.0523 -0.0030
-0.0270 -0.0442 0.0090 0.0171 -0.0230 -0.0015 0.0213 -0.0402
-0.0263 -0.0099 0.0021 -0.0250 0.0210 -0.0028 -0.0298 -0.0095
0.0034 0.0281 -0.0188 -0.0002 -0.0173 -0.0076 -0.0014 0.0184
-0.0318 0.0048 0.0047 -3.2555 1.1710 -0.2913

```

```

Reconstruction      Y :
6.5271 6.5120 6.5016 6.5237 6.4625 6.3496 6.4025 6.4035
6.4407 6.4746 6.5095 6.6551 6.6100 6.5969 6.6083 6.6520
6.7113 6.7227 6.7196 6.7649 6.7794 6.8037 6.8308 6.7712
6.7067 6.7690 6.7068 6.7024 6.6463 6.6098 6.5900 6.5960
6.5457 6.5470 6.5797 6.5895 6.6275 6.6795 6.6598 6.6925
6.6873 6.7223 6.7205 6.6843 6.7030 6.6470 6.6008 6.6061
6.6097 6.6485 6.6394 6.6571 6.6357 6.6224 6.6073 6.6075

```

6.6379 6.6294 6.5906 6.6258 6.6369 6.6515 6.6826 6.7042

---