NAG Library Function Document

nag_mlmodwt (c09dcc)

1 Purpose

nag_mlmodwt (c09dcc) computes the one-dimensional multi-level maximal overlap discrete wavelet transform (MODWT). The initialization function nag_wfilt (c09aac) must be called first to set up the MODWT options.

2 Specification

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

```
void nag_mlmodwt (Integer n, const double x[], Nag_WaveletCoefficients keepa,
     Integer lenc, double c[], Integer nwl, Integer *na, Integer icomm[],
     NagError *fail)
```

3 Description

nag_mlmodwt (c09dcc) computes the multi-level MODWT for a data set, x_i, for $i = 1, 2, \ldots, n$, in one dimension. For a chosen number of levels, n_l, with $n_l \leq l_{\text{max}}$, where l_{max} is returned by the initialization function nag_wfilt (c09aac) in **nwlmax**, the transform is returned as a set of coefficients for the different levels stored in a single array. Periodic reflection is currently the only available end extension method to reduce the edge effects caused by finite data sets.

The argument **keepa** can be set to retain both approximation and detail coefficients at each level resulting in $n_l \times (n_a + n_d)$ coefficients being returned in the output array, **c**, where n_a is the number of approximation coefficients and n_d is the number of detail coefficients. Otherwise, only the detail coefficients are stored for each level along with the approximation coefficients for the final level, in which case the length of the output array, **c**, is $n_a + n_l \times n_d$. In the present implementation, for simplicity, n_a and n_d are chosen to be equal by adding zero padding to the wavelet filters where necessary.

4 References

Percival D B and Walden A T (2000) *Wavelet Methods for Time Series Analysis* Cambridge University Press

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 input dataset x_i, for $i = 1, 2, \ldots, n$.

3: **keepa** – Nag_WaveletCoefficients *Input*

On entry: determines whether the approximation coefficients are stored in array **c** for every level of the computed transform or else only for the final level. In both cases, the detail coefficients are stored in **c** for every level computed.

keepa = Nag_StoreAll
Retain approximation coefficients for all levels computed.

keepa = Nag_StoreFinal
Retain approximation coefficients for only the final level computed.

Constraint: **keepa** = Nag_StoreAll or Nag_StoreFinal.

4: **lenc** – Integer *Input*

On entry: the dimension of the array **c**. **c** must be large enough to contain the number of wavelet coefficients.

If **keepa** = Nag_StoreFinal, the total number of coefficients, n_c, is returned in **nwc** by the call to the initialization function nag_wfilt (c09aac) and corresponds to the MODWT being continued for the maximum number of levels possible for the given data set. When the number of levels, n_l, is chosen to be less than the maximum, then the number of stored coefficients is correspondingly smaller and **lenc** can be reduced by noting that n_d detail coefficients are stored at each level, with the storage increased at the final level to contain the n_a approximation coefficients.

If **keepa** = Nag_StoreAll, n_d detail coefficients and n_a approximation coefficients are stored for each level computed, requiring **lenc** $\geq n_l \times (n_a + n_d) = 2 \times n_l \times n_a$, since the numbers of stored approximation and detail coefficients are equal. The number of approximation (or detail) coefficients at each level, n_a, is returned in **na**.

Constraints:

if **keepa** = Nag_StoreFinal, **lenc** $\geq (n_l + 1) \times n_a$;
if **keepa** = Nag_StoreAll, **lenc** $\geq 2 \times n_l \times n_a$.

5: **c**[**lenc**] – double *Output*

On exit: the coefficients of a multi-level wavelet transform of the dataset.

The coefficients are stored in **c** as follows:

If **keepa** = Nag_StoreFinal,

$\mathbf{C}(1 : n_a)$
Contains the level n_l approximation coefficients;

$\mathbf{C}(n_a + (i-1) \times n_d + 1 : n_a + i \times n_d)$
Contains the level $(n_l - i + 1)$ detail coefficients, for $i = 1, 2, \ldots, n_l$;

If **keepa** = Nag_StoreAll,

$\mathbf{C}((i-1) \times n_a + 1 : i \times n_a)$
Contains the level $(n_l - i + 1)$ approximation coefficients, for $i = 1, 2, \ldots, n_l$;

$\mathbf{C}(n_l \times n_a + (i-1) \times n_d + 1 : n_l \times n_a + i \times n_d)$
Contains the level i detail coefficients, for $i = 1, 2, \ldots, n_l$;

The values n_a and n_d denote the numbers of approximation and detail coefficients respectively, which are equal and returned in **na**.

6: **nwl** – Integer *Input*

On entry: the number of levels, n_l, 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 (c09aac).

7: **na** – Integer * *Output*

On exit: **na** contains the number of approximation coefficients, n_a, at each level which is equal to the number of detail coefficients, n_d. With periodic end extension (**mode** $=$ Nag_Periodic in nag_wfilt (c09aac)) this is the same as the length, **n**, of the data array, **x**.

8: **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.

9: **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_ARRAY_DIM_LEN

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

NE_BAD_PARAM

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

NE_INITIALIZATION

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 $= \langle value \rangle$.

On entry, the initialization function nag_wfilt (c09aac) has not been called first or it has not been called with **wtrans** $=$ Nag_MODWTMulti, or the communication array **icomm** has become corrupted.

NE_INT

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

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 wavelet coefficients at each level can be extracted from the output array **c** using the information contained in **na** on exit.

10 Example

A set of data values (**n** = 64) is decomposed using the MODWT over two levels and then the inverse (nag_imlmodwt (c09ddc)) is applied to restore the original data set.

10.1 Program Text

```
/* nag_mlmodwt (c09dcc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */
/* 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, na, nf, nwc, nwcmax, nwlmax, nwl, nwlinv;
  Integer         *icomm = 0;
  NagError        fail;
  Nag_Wavelet     wavnamenum;
  Nag_WaveletCoefficients keepnum;
  Nag_WaveletMode modenum;
  /*Double scalar and array declarations */
  double          *c = 0, *x = 0, *y = 0;
  /*Character scalar and array declarations */
  char            keep[15], mode[24], wavnam[20];

  INIT_FAIL(fail);

  printf("nag_mlmodwt (c09dcc) Example Program Results\n\n");
  fflush(stdout);

  /*     Skip heading in data file*/
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  /*     Read n - length of input data sequence*/
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n] ", &n);
```

```
#else
  scanf("%"NAG_IFMT"%*[^\n] ", &n);
#endif
  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)*/
#ifdef _WIN32
  scanf_s("%19s%23s%14s%*[^\n] ", wavnam, _countof(wavnam), mode,
          _countof(mode), keep, _countof(keep));
#else
  scanf("%19s%23s%14s%*[^\n] ", wavnam, mode, keep);
#endif
/*
 * 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);
keepnum = (Nag_WaveletCoefficients) nag_enum_name_to_value(keep);
if (n >= 2)
  {
    printf("MLMODWT :: \n");
    printf("     Wavelet             :%16s\n", wavnam);
    printf("     End mode            :%16s\n", mode);
    printf("     Store coefficients :%16s\n", keep);
    printf("     N                   :%16"NAG_IFMT"\n\n", n);
    /*        Read data array and write it out*/
    printf("%s\n", " Input Data      X :");
    for (i = 0; i < n; i++)
      {
#ifdef _WIN32
 scanf_s("%lf", &x[i]);
#else
 scanf("%lf", &x[i]);
#endif
 printf("%8.4f%s", x[i], (i+1)%8?" ":"\n");
      }
    printf("\n");
    /*
     * nag_wfilt (c09aac)
     * Wavelet filter query
     */
    nag_wfilt(wavnamenum, Nag_MODWTMulti, modenum, n, &nwlmax, &nf, &nwcmax,
       icomm, &fail);
    if (fail.code != NE_NOERROR)
      {
 printf("Error from nag_wfilt (c09aac).\n%s\n", fail.message);
 exit_status = 1;
 goto END;
      }
    /* Choose to decompose over two levels */
    if (nwlmax >= 2) nwl = 2;
    /* Set size of array c according to number of coefficients stored */
    if (keepnum==Nag_StoreFinal)
      nwc = nwcmax - (nwlmax-nwl)*n;
    else
      nwc = nwcmax + (nwlmax-1)*n - (nwlmax-nwl)*2*n;

    if (!(c = NAG_ALLOC(nwc, double)))
      {
 printf("Allocation failure\n");
 exit_status = -1;
 goto END;
      }
    /*        Perform Maximal Overlap Discrete Wavelet transform*/
```

```
    /*
     * nag_mlmodwt (c09dcc)
     * one-dimensional multi-level maximal overlap discrete wavelet
     * transform (mlmodwt)
     */
    nag_mlmodwt(n, x, keepnum, nwc, c, nwl, &na, icomm, &fail);
    if (fail.code != NE_NOERROR)
      {
 printf("Error from nag_mlmodwt (c09dcc).\n%s\n", fail.message);
 exit_status = 1;
 goto END;
      }
    printf("   Number of Levels : %20"NAG_IFMT"\n", nwl);
    printf("   Number of coefficients in each level : %20"NAG_IFMT"\n", na);
    printf("   Wavelet coefficients C : \n");
    for (i = 0; i < nwc; i++)
      printf("%8.4f%s", c[i], (i+1)%8?" ":"\n");
    printf("\n\n");
    /*           Reconstruct original data*/
    nwlinv = nwl;
    /*
     * nag_imlmodwt (c09ddc)
     * one-dimensional inverse multi-level discrete wavelet transform
     * (imlmodwt)
     */
    nag_imlmodwt(nwlinv, keepnum, nwc, c, n, y, icomm, &fail);
    if (fail.code != NE_NOERROR)
      {
 printf("Error from nag_imlmodwt (c09ddc).\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(icomm);

return exit_status;
}
```

10.2 Program Data

```
nag_mlmodwt (c09dcc) Example Program Data
64                                      : n
Nag_Daubechies4  Nag_Periodic Nag_StoreFinal : wavnam, mode, keepa
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_mlmodwt (c09dcc) Example Program Results

MLMODWT ::
     Wavelet            : Nag_Daubechies4
     End mode           :    Nag_Periodic
     Store coefficients :  Nag_StoreFinal
     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 :                       2
   Number of coefficients in each level :                   64
   Wavelet coefficients C :
  6.6448   6.6505   6.6415   6.6090   6.5631   6.5119   6.4657   6.4371
  6.4162   6.4041   6.4062   6.4235   6.4652   6.5191   6.5744   6.6170
  6.6375   6.6496   6.6575   6.6741   6.7038   6.7335   6.7633   6.7849
  6.7939   6.7970   6.7868   6.7649   6.7407   6.7102   6.6814   6.6571
  6.6269   6.5993   6.5773   6.5598   6.5574   6.5688   6.5881   6.6173
  6.6492   6.6741   6.6941   6.7052   6.7078   6.7083   6.7001   6.6842
  6.6616   6.6338   6.6146   6.6072   6.6139   6.6306   6.6428   6.6459
  6.6384   6.6252   6.6147   6.6113   6.6143   6.6189   6.6264   6.6361
  0.0107   0.0084   0.0003  -0.0065  -0.0000   0.0196   0.0191  -0.0152
 -0.0369  -0.0291  -0.0131   0.0227   0.0461   0.0005  -0.0488  -0.0145
  0.0518   0.0503  -0.0038  -0.0243  -0.0087  -0.0111  -0.0316  -0.0191
  0.0323   0.0461  -0.0001  -0.0300  -0.0107   0.0164   0.0112  -0.0156
 -0.0225  -0.0091   0.0090   0.0244   0.0050  -0.0281  -0.0150   0.0146
  0.0145   0.0034  -0.0019   0.0058   0.0188   0.0074  -0.0133  -0.0127
 -0.0062  -0.0008   0.0077   0.0022  -0.0151  -0.0192  -0.0041   0.0091
  0.0136   0.0230   0.0203  -0.0081  -0.0274  -0.0179  -0.0013   0.0074
 -0.0150   0.0126   0.0048  -0.0276  -0.0227   0.0639  -0.0184  -0.0048
 -0.0303   0.0180   0.0327  -0.0343   0.0119  -0.0046   0.0167   0.0025
 -0.0524   0.0369   0.0029   0.0055  -0.0070  -0.0134   0.0099   0.0088
 -0.0095   0.0103  -0.0114  -0.0181   0.0269   0.0132  -0.0371   0.0250
 -0.0186   0.0138   0.0022  -0.0058  -0.0112   0.0207  -0.0058  -0.0054
  0.0115  -0.0089  -0.0106   0.0180  -0.0096   0.0107  -0.0156   0.0068
  0.0074  -0.0242   0.0169   0.0075  -0.0045   0.0031  -0.0108   0.0092
 -0.0115   0.0061  -0.0002   0.0078  -0.0012  -0.0168   0.0074   0.0157


   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
```