NAG Library Function Document

nag_wfilt_2d (c09abc)

1 Purpose

nag_wfilt_2d (c09abc) returns the details of the chosen two-dimensional discrete wavelet filter. For a chosen mother wavelet, discrete wavelet transform type (single-level or multi-level DWT) and end extension method, this function returns the maximum number of levels of resolution (appropriate to a multi-level transform), the filter length, the total number of approximation, horizontal, vertical and diagonal coefficients and the number of coefficients in the second dimension for the single-level case. This function must be called before any of the two-dimensional transform functions in this chapter.

2 Specification

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

void nag_wfilt_2d (Nag_Wavelet wavnam, Nag_WaveletTransform wtrans,
                   Nag_WaveletMode mode, Integer m, Integer n, Integer *nwlmax,
                   Integer *nf, Integer *nwct, Integer *nwcn, Integer icomm[],
                   NagError *fail)
```

3 Description

Two-dimensional discrete wavelet transforms (DWT) are characterised by the mother wavelet, the end extension method and whether multiresolution analysis is to be performed. For the selected combination of choices for these three characteristics, and for given dimensions \((m \times n)\) of data matrix \(A\), nag_wfilt_2d (c09abc) returns the dimension details for the transform determined by this combination. The dimension details are: \(l_{\text{max}}\), the maximum number of levels of resolution that would be computed were a multi-level DWT applied; \(n_f\), the filter length; \(n_{ct}\), the total number of approximation, horizontal, vertical and diagonal coefficients (over all levels in the multi-level DWT case); and \(n_{cn}\), the number of coefficients in the second dimension for a single-level DWT. These values are also stored in the communication array \(icomm\), as are the input choices, so that they may be conveniently communicated to the two-dimensional transform functions in this chapter.

4 References

None.

5 Arguments

1: wavnam – Nag_Wavelet

   Input

   On entry: the name of the mother wavelet. See the c09 Chapter Introduction for details.

   wavnam = Nag_Haar
   Haar wavelet.

   wavnam = Nag_Daubechies\(n\), where \(n = 2, 3, \ldots, 10\)
   Daubechies wavelet with \(n\) vanishing moments (2\(n\) coefficients). For example,
   wavnam = Nag_Daubechies4 is the name for the Daubechies wavelet with 4 vanishing
   moments (8 coefficients).
wavnam = Nag_Biorthogonal\(x,y\), where \(x,y\) can be one of \(1_1, 1_3, 1_5, 2_2, 2_4, 2_6, 2_8, 3_1, 3_3, 3_5\) or \(3_7\)

Biorthogonal wavelet of order \(x,y\). For example \(\text{wavnam} = \text{Nag}_x\text{Biorthogonal}_y1\) is the name for the Biorthogonal wavelet of order \(1,1\).

**Constraint:** \(\text{wavnam} = \text{Nag}_x\text{Haar}, \text{Nag}_x\text{Daubechies}_2, \text{Nag}_x\text{Daubechies}_3, \text{Nag}_x\text{Daubechies}_4, \text{Nag}_x\text{Daubechies}_5, \text{Nag}_x\text{Daubechies}_6, \text{Nag}_x\text{Daubechies}_7, \text{Nag}_x\text{Daubechies}_8, \text{Nag}_x\text{Daubechies}_9, \text{Nag}_x\text{Daubechies}_{10}, \text{Nag}_x\text{Biorthogonal}_x1, \text{Nag}_x\text{Biorthogonal}_x3, \text{Nag}_x\text{Biorthogonal}_x5, \text{Nag}_x\text{Biorthogonal}_x2, \text{Nag}_x\text{Biorthogonal}_x4, \text{Nag}_x\text{Biorthogonal}_x6, \text{Nag}_x\text{Biorthogonal}_x8, \text{Nag}_x\text{Biorthogonal}_x1, \text{Nag}_x\text{Biorthogonal}_x3, \text{Nag}_x\text{Biorthogonal}_x5\) or \(\text{Nag}_x\text{Biorthogonal}_x7\).

2: \(wtrans\) – Nag_WaveletTransform

*Input*

On entry: the type of discrete wavelet transform that is to be applied.

\(wtrans = \text{Nag}_\text{SingleLevel}\)

Single-level decomposition or reconstruction by discrete wavelet transform.

\(wtrans = \text{Nag}_\text{MultiLevel}\)

Multiresolution, by a multi-level DWT or its inverse.

**Constraint:** \(wtrans = \text{Nag}_\text{SingleLevel}\) or \(\text{Nag}_\text{MultiLevel}\).

3: \(mode\) – Nag_WaveletMode

*Input*

On entry: the end extension method.

\(mode = \text{Nag}_\text{Periodic}\)

Periodic end extension.

\(mode = \text{Nag}_\text{HalfPointSymmetric}\)

Half-point symmetric end extension.

\(mode = \text{Nag}_\text{WholePointSymmetric}\)

Whole-point symmetric end extension.

\(mode = \text{Nag}_\text{ZeroPadded}\)

Zero end extension.

**Constraint:** \(mode = \text{Nag}_\text{Periodic}, \text{Nag}_\text{HalfPointSymmetric}, \text{Nag}_\text{WholePointSymmetric}\) or \(\text{Nag}_\text{ZeroPadded}\).

4: \(m\) – Integer

*Input*

On entry: the number of elements, \(m\), in the first dimension (number of rows of data matrix \(A\)) of the input data.

**Constraint:** \(m \geq 2\).

5: \(n\) – Integer

*Input*

On entry: the number of elements, \(n\), in the second dimension (number of columns of data matrix \(A\)) of the input data.

**Constraint:** \(n \geq 2\).

6: \(nwlmax\) – Integer *

*Output*

On exit: the maximum number of levels of resolution, \(l_{\text{max}}\), that can be computed if a multi-level discrete wavelet transform is applied (\(wtrans = \text{Nag}_\text{MultiLevel}\)). It is such that \(2^n_{\text{max}} \leq \min(m,n) < 2^{n_{\text{max}}+1}\), for \(l_{\text{max}}\) an integer.

If \(wtrans = \text{Nag}_\text{SingleLevel}\), \(nwlmax\) is not set.
7: \( nf \) – Integer *  
On exit: the filter length, \( n_f \), for the supplied mother wavelet. This is used to determine the number of coefficients to be generated by the chosen transform.

8: \( nwct \) – Integer *  
On exit: the total number of wavelet coefficients, \( n_c \), that will be generated. When \( \text{wtrans} = \text{Nag\_SingleLevel} \) the number of rows required in each of the output coefficient matrices can be calculated as \( n_{cm} = n_c / (4n_{cn}) \). When \( \text{wtrans} = \text{Nag\_MultiLevel} \) the length of the array used to store all of the coefficient matrices must be at least \( n_{ct} \).

9: \( nwcn \) – Integer *  
On exit: for a single-level transform (\( \text{wtrans} = \text{Nag\_SingleLevel} \)), the number of coefficients that would be generated in the second dimension, \( n_{cn} \), for each coefficient type. For a multi-level transform (\( \text{wtrans} = \text{Nag\_MultiLevel} \)) this is set to 1.

10: \( \text{icomm}[180] \) – Integer  
Communication Array  
On exit: contains details of the wavelet transform and the problem dimension which is to be communicated to the two-dimensional discrete transform functions in this chapter.

11: \( \text{fail} \) – 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 \( \langle \text{value} \rangle \) had an illegal value.

**NE_INT**
On entry, \( m = \langle \text{value} \rangle \).  
Constraint: \( m \geq 2 \).  
On entry, \( n = \langle \text{value} \rangle \).  
Constraint: \( n \geq 2 \).

**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

Not applicable.
8 Parallelism and Performance
Not applicable.

9 Further Comments
None.

10 Example
This example computes the two-dimensional multi-level resolution for a $6 \times 6$ matrix by a discrete wavelet transform using the Haar wavelet with whole-point symmetric end extensions. The number of levels of transformation actually performed is one less than the maximum possible. This number of levels, the length of the wavelet filter, the total number of coefficients and the number of coefficients in each dimension for each level are printed along with the vertical detail coefficients from the first level, before a reconstruction is performed.

10.1 Program Text
/* nag_wfilt_2d (c09abc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 24, 2013. */
*/
#include <stdio.h>
#include <nag.h>
#include <string.h>
#include <nag_stdlib.h>
#include <nagc09.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer exit_status = 0;
    Integer i, j, lenc, m, n, nf, nwcm, nwcn, nwct, nwlmax, pda, pdb;
    Integer want_level, want_coeffs;
    /* Arrays */
    char mode[24], wavnam[20], title[50];
    double *a = 0, *b = 0, *c = 0, *d = 0;
    Integer *dwtlevm = 0, *dwtlevn = 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_wfilt_2d (c09abc) Example Program Results\n\n");

    /* Skip heading in data file and read problem parameters */
    #ifdef _WIN32
    scanf_s("%*\n\n%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "\n", &m, &n);
    #else
    scanf("%*\n\n%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "\n", &m, &n);
    #endif
    pda = m;
    pdb = m;
    #ifdef _WIN32
    scanf_s("%19s%23s%*\n", wavnam, _countof(wavnam), mode, _countof(mode));
    #endif
}
#include

scanf("#19s%23s*\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(" MLDWT :: Wavelet : %s\n", wavnam);
printf(" End mode : %s\n", mode);
printf(" m : %" NAG_IFMT "\n", m);
printf(" n : %" NAG_IFMT "\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]
#endif _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

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print_comp (x04cbc).\n\n%s\n", fail.message);
    exit_status = 0;
    goto END;
}

/ * Read data array and write it out*/
#define A(I, J) a[(J-1)*pda + I-1]
#endif _WIN32
for (j = 1; j <= n; j++) scanf_s("%lf", &A(i, j));
#endif

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print_comp (x04cbc).\n\n%s\n", fail.message);
    exit_status = 0;
    goto END;
}

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_wfilt_2d (c09abc).\n\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

lenc = nwct;
if (
    !(c = NAG_ALLOC(lenc, double)) ||
    !(dwtlevm = NAG_ALLOC(nwmax, Integer)) ||
    !(dwtlevn = NAG_ALLOC(nwmax, Integer))
)
{
    printf("Allocation failure\n");
    exit_status = 2;
    goto END;
}

/ * nag_mldwt_2d (c09ecc).
 * Two-dimensional multi-level discrete wavelet transform
 */
nag_mldwt_2d(m, n, a, pda, lenc, c, nwmax, dwtlevm, dwtlevn, icomm, &fail);
if (fail.code != NE_NOERROR)
#include "nag.h"

{ printf("Error from nag_mldwt_2d (c09ecc).\n%s\n", fail.message);
  exit_status = 3;
  goto END;
}

/* Print decomposition */
printf("\n Length of wavelet filter : %12s%NAG_IFMT " \n", ",", nf);
printf("\n Number of Levels : %"NAG_IFMT"\n", nwlmax);
printf("\n Number of coefficients in 1st dimension for each level :\n"");
for (j = 0; j < nwlmax; j++)
  printf("%8"NAG_IFMT"%s", dwtlevm[j], (j+1)%8 ? " " : "\n");
printf("\n Number of coefficients in 2nd dimension for each level :\n"");
for (j = 0; j < nwlmax; j++)
  printf("%8"NAG_IFMT"%s", dwtlevn[j], (j+1)%8 ? " " : "\n");
printf("\n Total number of wavelet coefficients : ");
printf("%10" NAG_IFMT "\n\n", nwct);
printf(" Wavelet coefficients c : 
");
for (j = 0; j < nwct; j++) printf("%8.4f%s", c[j], (j+1)%8 ? " " : "\n");

/* Now select a nominated matrix of coefficients at a nominated level. */
* Remember that level 0 is input data, 1 first coeffs and so on up to nwlmax,
* which is the deepest level and contains approx. coefficients.
*/
want_level = nwlmax - 1;
/* Print only vertical detail coeffs at selected level. */
want_coeffs = 1;
wcm = dwtlevm[nwlmax-want_level];
wcn = dwtlevn[nwlmax-want_level];
if (!((d = NAG_ALLOC(nwcm*nwcn, double))))
  {
    printf("Allocation failure\n");
    exit_status = 4;
    goto END;
  }
/* nag_wav_2d_coeff_ext (c09aec).
* Extract the selected set of coefficients.
*/
nag_wav_2d_coeff_ext(want_level, want_coeffs, lenc, c, d, nwcm, icomm, &fail);
if (fail.code != NE_NOERROR)
  {
    printf("Error from nag_wav_2d_coeff_ext (c09aec).\n%s\n", fail.message);
    exit_status = 5;
    goto END;
  }
/* Print out the selected coefficients */
printf("\n");
fflush(stdout);
#ifdef _WIN32
  sprintf_s(title, _countof(title),
    "Type %" NAG_IFMT " coefficients at selected wavelet level 
"%s" NAG_IFMT " :", want_coeffs, want_level);
#else
  sprintf(title, "Type %" NAG_IFMT " coefficients at selected wavelet level 
"%s" NAG_IFMT ":", want_coeffs, want_level);
#endif
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);

/* nag_imldwt_2d (c09edc).
* Two-dimensional inverse multi-level discrete wavelet transform
*/
nag_imldwt_2d(nwlmax, 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 = 6;
goto END;
}

/* Print reconstruction */
printf("\n");
fflush(stdout);

#ifdef _WIN32
strcpy_s(title, _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 = 7;
goto END;
}

END:
NAG_FREE(a);
NAG_FREE(b);
NAG_FREE(c);
NAG_FREE(d);
NAG_FREE(dwtlevm);
NAG_FREE(dwtlevn);
return exit_status;

10.2 Program Data

nag_wfilt_2d (c09abc) Example Program Data
6
6
m, n
Nag_Haar Nag_Haar Nag_Haar Nag_Haar Nag_Haar Nag_Haar Nag_Haar Nag_Haar Nag_Haar Nag_Haar
6.0000 7.0000 8.0000 0.0000 1.0000 9.0000
9.0000 1.0000 9.0000 9.0000 2.0000 8.0000
3.0000 0.0000 4.0000 1.0000 3.0000 1.0000
2.0000 5.0000 9.0000 4.0000 4.0000 2.0000
1.0000 8.0000 3.0000 3.0000 5.0000 3.0000
8.0000 1.0000 6.0000 4.0000 6.0000 1.0000
: data matrix A

10.3 Program Results

nag_wfilt_2d (c09abc) Example Program Results

Parameters read from file ::
MLDWT :: Wavelet : Nag_Haar
    End mode : Nag_Haar Nag_Haar Nag_Haar Nag_Haar Nag_Haar Nag_Haar Nag_Haar Nag_Haar Nag_Haar Nag_Haar
m : 6
n : 6
Input Data A :
6.0000 7.0000 8.0000 0.0000 1.0000 9.0000
9.0000 1.0000 9.0000 9.0000 2.0000 8.0000
3.0000 0.0000 4.0000 1.0000 3.0000 1.0000
2.0000 5.0000 9.0000 4.0000 4.0000 2.0000
1.0000 8.0000 3.0000 3.0000 5.0000 3.0000
8.0000 1.0000 6.0000 4.0000 6.0000 1.0000
: data matrix A

Length of wavelet filter : 2
Number of Levels : 2
Number of coefficients in 1st dimension for each level : 2
Number of coefficients in 2nd dimension for each level: 2 3
Total number of wavelet coefficients: 43

Wavelet coefficients c:

| 5.2500  | 1.5000  | 4.5000  | 0.7500  | 1.2500  | 2.5000  | 0.5000  | 1.7500  |
| 3.5000  | 0.0000  | 0.0000  | 4.0000  | 4.0000  | 1.0000  | -7.0000 | 2.0000  |
| 3.5000  | 1.5000  | -2.0000 | 0.0000  | -5.0000 | -4.0000 | -2.0000 | 0.0000  |
| -1.0000 | 0.5000  | -4.5000 | 3.0000  | -7.0000 | 4.0000  | -1.0000 | -1.0000 |
| -1.0000 | 0.0000  | -1.5000 |

Type 1 coefficients at selected wavelet level 1:

| 3.5000 | 4.0000 | -7.0000 |
| 0.0000 | 4.0000 | 2.0000  |
| 0.0000 | 1.0000 | 3.5000  |

Reconstruction B:

| 6.0000 | 7.0000 | 8.0000 | 0.0000 | 1.0000 | 9.0000 |
| 9.0000 | 1.0000 | 9.0000 | 9.0000 | 2.0000 | 8.0000 |
| 3.0000 | 0.0000 | 4.0000 | 1.0000 | 3.0000 | 1.0000 |
| 2.0000 | 5.0000 | 9.0000 | 4.0000 | 4.0000 | 2.0000 |
| 1.0000 | 8.0000 | 3.0000 | 3.0000 | 5.0000 | 3.0000 |
| 8.0000 | 1.0000 | 6.0000 | 4.0000 | 6.0000 | 1.0000 |