# NAG Toolbox

# nag_wav_3d_mxolap_multi_inv (c09fd)

## 1 Purpose

nag_wav_3d_mxolap_multi_inv (c09fd) computes the inverse three-dimensional multi-level discrete wavelet transform (IDWT). This function reconstructs data from (possibly filtered or otherwise manipulated) wavelet transform coefficients calculated by nag_wav_3d_multi_fwd (c09fc) from an original input array. The initialization function nag_wav_3d_init (c09ac) must be called first to set up the IDWT options.

## 2 Syntax

```
[b, ifail] = nag_wav_3d_mxolap_multi_inv(nwlinv, c, m, n, fr, icomm, 'lenc', lenc)
[b, ifail] = c09fd(nwlinv, c, m, n, fr, icomm, 'lenc', lenc)
```

## 3 Description

nag_wav_3d_mxolap_multi_inv (c09fd) performs the inverse operation of nag_wav_3d_multi_fwd (c09fc). That is, given a set of wavelet coefficients, computed up to level $n_{\mathrm{fwd}}$ by nag_wav_3d_multi_fwd (c09fc) using a DWT as set up by the initialization function nag_wav_3d_init (c09ac), on a real three-dimensional array, $A$, nag_wav_3d_mxolap_multi_inv (c09fd) will reconstruct $A$. The reconstructed array is referred to as $B$ in the following since it will not be identical to $A$ when the DWT coefficients have been filtered or otherwise manipulated prior to reconstruction. If the original input array is level 0, then it is possible to terminate reconstruction at a higher level by specifying fewer than the number of levels used in the call to nag_wav_3d_multi_fwd (c09fc). This results in a partial reconstruction.

## 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 Parameters

### 5.1 Compulsory Input Parameters

1: **nwlinv** – INTEGER

The number of levels to be used in the inverse multi-level transform. The number of levels must be less than or equal to $n_{\mathrm{fwd}}$, which has the value of argument **nwl** as used in the computation of the wavelet coefficients using nag_wav_3d_multi_fwd (c09fc). The data will be reconstructed to level (**nwl** – **nwlinv**), where level 0 is the original input dataset provided to nag_wav_3d_multi_fwd (c09fc).

*Constraint*: $1 \leq$ **nwlinv** $\leq$ **nwl**, where **nwl** is the value used in a preceding call to nag_wav_3d_multi_fwd (c09fc).

2: **c**(**lenc**) – REAL (KIND=nag_wp) array

The coefficients of the multi-level discrete wavelet transform. This will normally be the result of some transformation on the coefficients computed by function nag_wav_3d_multi_fwd (c09fc).

Note that the coefficients in **c** may be extracted according to level and type into three-dimensional arrays using nag_wav_3d_coeff_ext (c09fy), and inserted using nag_wav_3d_coeff_ins (c09fz).

3: **m** – INTEGER

The number of elements, $m$, in the first dimension of the reconstructed array $B$. For a full reconstruction of **nwl** levels, where **nwl** is as supplied to nag_wav_3d_multi_fwd (c09fc), this must be the same as argument **m** used in a preceding call to nag_wav_3d_multi_fwd (c09fc). For a partial reconstruction of **nwlinv** $<$ **nwl** levels, this must be equal to **dwtlvm**(**nwlinv** $+ 1$), as returned from nag_wav_3d_multi_fwd (c09fc)

4: **n** – INTEGER

The number of elements, $n$, in the second dimension of the reconstructed array $B$. For a full reconstruction of **nwl**, levels, where **nwl** is as supplied to nag_wav_3d_multi_fwd (c09fc), this must be the same as argument **n** used in a preceding call to nag_wav_3d_multi_fwd (c09fc). For a partial reconstruction of **nwlinv** $<$ **nwl** levels, this must be equal to **dwtlvn**(**nwlinv** $+ 1$), as returned from nag_wav_3d_multi_fwd (c09fc).

5: **fr** – INTEGER

The number of elements, $fr$, in the third dimension of the reconstructed array $B$. For a full reconstruction of **nwl** levels, where **nwl** is as supplied to nag_wav_3d_multi_fwd (c09fc), this must be the same as argument **fr** used in a preceding call to nag_wav_3d_multi_fwd (c09fc). For a partial reconstruction of **nwlinv** $<$ **nwl** levels, this must be equal to **dwtlvfr**(**nwlinv** $+ 1$), as returned from nag_wav_3d_multi_fwd (c09fc).

6: **icomm**(**260**) – INTEGER array

Contains details of the discrete wavelet transform and the problem dimension as setup in the call to the initialization function nag_wav_3d_init (c09ac).

## 5.2 Optional Input Parameters

1: **lenc** – INTEGER

*Default*: the dimension of the array **c**.

The dimension of the array **c**.

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

## 5.3 Output Parameters

1: **b**($ldb, sdb,$ **fr**) – REAL (KIND=nag_wp) array

$sdb =$ **n**.

The $m$ by $n$ by $fr$ reconstructed array, $B$, with $B_{ijk}$ stored in $\mathbf{b}(i,j,k)$. The reconstruction is based on the input multi-level wavelet transform coefficients and the transform options supplied to the initialization function nag_wav_3d_init (c09ac).

2: **ifail** – INTEGER

**ifail** $= 0$ unless the function detects an error (see Section 5).

## 6 Error Indicators and Warnings

Errors or warnings detected by the function:

**ifail** $= 1$

Constraint: **nwlinv** $\leq$ **nwl** as used in the call to nag_wav_3d_multi_fwd (c09fc).

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

**ifail** $= 2$

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

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

**ifail** $= 3$

**lenc** is too small, the number of wavelet coefficients required for a transform operating on **nwlinv** levels. If **nwlinv** $=$ **nwlmax**, the maximum number of levels as returned by the initial call to nag_wav_3d_init (c09ac), then **lenc** must be at least $n_{\mathrm{ct}}$, the value returned in **nwct** by the same call to nag_wav_3d_init (c09ac).

**ifail** $= 4$

**fr** is too small, the number of coefficients in the third dimension at the required level of reconstruction.

**m** is too small, the number of coefficients in the first dimension at the required level of reconstruction.

**n** is too small, the number of coefficients in the second dimension at the required level of reconstruction.

**ifail** $= 6$

Either the communication array **icomm** has been corrupted or there has not been a prior call to the initialization function nag_wav_3d_init (c09ac).

The initialization function was called with **wtrans** $=$ 'S'.

**ifail** $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

**ifail** $= -399$

Your licence key may have expired or may not have been installed correctly.

**ifail** $= -999$

Dynamic memory allocation failed.

## 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 Further Comments

None.

## 9 Example

See Section 10 in nag_wav_3d_multi_fwd (c09fc).

## 9.1 Program Text

```
     function c09fd_example

fprintf('c09fd example results\n\n');

m  = nag_int(7);
n  = nag_int(6);
fr = nag_int(5);
wavnam = 'Bior1.1';
mode = 'period';
wtrans = 'Multilevel';
a = zeros(m, n, fr);
a(:, :, 1) = [3, 2, 2, 2, 1, 1;
              2, 9, 1, 2, 1, 3;
              2, 5, 1, 2, 1, 1;
              1, 6, 2, 2, 7, 2;
              5, 3, 2, 2, 4, 7;
              2, 2, 1, 1, 2, 1;
              6, 2, 1, 3, 6, 9];
a(:, :, 2) = [2, 1, 5, 1, 2, 3;
              2, 9, 5, 2, 1, 2;
              2, 3, 2, 7, 1, 1;
              2, 1, 1, 2, 3, 1;
              2, 1, 2, 8, 3, 3;
              1, 4, 5, 1, 2, 7;
              8, 1, 3, 9, 1, 2];
a(:, :, 3) = [3, 1, 4, 1, 1, 1;
              1, 1, 2, 1, 2, 6;
              4, 1, 7, 2, 5, 6;
              3, 2, 1, 5, 9, 5;
              1, 1, 2, 2, 2, 1;
              2, 6, 3, 9, 5, 1;
              1, 1, 8, 2, 1, 3];
a(:, :, 4) = [5, 8, 1, 2, 2, 1;
              1, 2, 2, 9, 2, 9;
              2, 2, 2, 1, 1, 3;
              1, 1, 1, 5, 1, 2;
              3, 2, 8, 1, 9, 2;
              2, 1, 9, 1, 2, 2;
              3, 6, 5, 3, 2, 2];
a(:, :, 5) = [5, 2, 1, 2, 1, 1;
              3, 1, 9, 1, 2, 1;
              2, 3, 1, 1, 7, 2;
              7, 2, 2, 6, 1, 1;
              5, 1, 7, 2, 1, 1;
              2, 1, 3, 2, 2, 1;
              5, 3, 9, 1, 4, 1];

% Query wavelet filter dimensions
[nwl, nf, nwct, nwcn, nwcfr, icomm, ifail] = ...
      c09ac(wavnam, wtrans, mode, m, n, fr);

% Perform Discrete Wavelet transform
[c, dwtlvm, dwtlvn, dwtlvfr, icomm, ifail] = c09fc(n, fr, a, nwct, nwl, icomm);

fprintf(' Number of Levels : %d\n\n', nwl);
fprintf(' Number of coefficients in 1st dimension for each level:\n');
fprintf(' %8d', dwtlvm(1:nwl));
fprintf('\n');
fprintf(' Number of coefficients in 2nd dimension for each level:\n');
fprintf(' %8d', dwtlvn(1:nwl));
fprintf('\n');
fprintf(' Number of coefficients in 3rd dimension for each level:\n');
fprintf(' %8d', dwtlvfr(1:nwl));
fprintf('\n');

% Print the first level HLL coefficients
want_level = 1;

% Select the approximation coefficients.
```

```
want_coeffs = 4;

% Identify each set of coefficients in c
for ilevel = nwl:-1:1

  if ilevel ~= want_level
    continue
  end

  nwcm = dwtlvm(nwl-ilevel+1);
  nwcn = dwtlvn(nwl-ilevel+1);
  nwcfr = dwtlvfr(nwl-ilevel+1);

  fprintf('\n---------------------------------\n');
  fprintf(' Level %d output is %d by %d by %d.\n', ilevel, nwcm, nwcn, nwcfr);
  fprintf('---------------------------------\n\n');

  for itype_coeffs = 0:7

    if itype_coeffs ~= want_coeffs
      continue
    end

    % Unless we're looking at the deepest level of nesting, which contains
    % approximation coefficients, advance the pointer on past the preceding
    % levels
    if ilevel == nwl
      locc = 0;
    else
      locc = 8*dwtlvm(1)*dwtlvn(1)*dwtlvfr(1);
      for i = ilevel + 1 : nwl - 1
        locc = locc + 7*dwtlvm(nwl-i+1)*dwtlvn(nwl-i+1)*dwtlvfr(nwl-i+1);
      end
    end

    % Now decide which coefficient type we are considering
    switch (itype_coeffs)
      case {0}
        if (ilevel==nwl)
          fprintf('Approximation coefficients (LLL)\n');
          locc = locc + 1;
        end
      case {1}
        fprintf('Detail coefficients (LLH)\n');
        if (ilevel==nwl)
          % Advance pointer past approximation coefficients
          locc = locc + nwcm*nwcn*nwcfr + 1;
        else
          locc = locc + 1;
        end
      case {2}
        fprintf('Detail coefficients (LHL)\n');
        if (ilevel==nwl)
          % Advance pointer past approximation coefficients and 1 set of
          % detail coefficients
          locc = locc + 2*nwcm*nwcn*nwcfr + 1;
        else
          % Advance pointer past 1 set of detail coefficients
          locc = locc + nwcm*nwcn*nwcfr + 1;
        end
      case {3}
        fprintf('Detail coefficients (LHH)\n');
        if (ilevel==nwl)
          % Advance pointer past approximation coefficients and 2 sets of
          % detail coefficients
          locc = locc + 3*nwcm*nwcn*nwcfr + 1;
        else
          % Advance pointer past 2 sets of detail coefficients
          locc = locc + 2*nwcm*nwcn*nwcfr + 1;
        end
      case {4}
```

```
        fprintf('Detail coefficients (HLL)\n');
        if (ilevel==nwl)
          % Advance pointer past approximation coefficients and 3 sets of
          % detail coefficients
          locc = locc + 4*nwcm*nwcn*nwcfr + 1;
        else
          % Advance pointer past 3 sets of detail coefficients
          locc = locc + 3*nwcm*nwcn*nwcfr + 1;
        end
      case {5}
        fprintf('Detail coefficients (HLH)\n');
        if (ilevel==nwl)
          % Advance pointer past approximation coefficients and 4 sets of
          % detail coefficients
          locc = locc + 5*nwcm*nwcn*nwcfr + 1;
        else
          % Advance pointer past 4 sets of detail coefficients
          locc = locc + 4*nwcm*nwcn*nwcfr + 1;
        end
      case {6}
        fprintf('Detail coefficients (HHL)\n');
        if (ilevel==nwl)
          % Advance pointer past approximation coefficients and 5 sets of
          % detail coefficients
          locc = locc + 6*nwcm*nwcn*nwcfr + 1;
        else
          % Advance pointer past 4 sets of detail coefficients
          locc = locc + 5*nwcm*nwcn*nwcfr + 1;
        end
      case {7}
        fprintf('Detail coefficients (HHH)\n');
        if (ilevel==nwl)
          % Advance pointer past approximation coefficients and 6 sets of
          % detail coefficients
          locc = locc + 7*nwcm*nwcn*nwcfr + 1;
        else
          % Advance pointer past 5 sets of detail coefficients
          locc = locc + 6*nwcm*nwcn*nwcfr + 1;
        end
      end

  if itype_coeffs > 0 || ilevel == nwl

    if (itype_coeffs==0)
      % For a multi level transform approx coeffs stored as
      % nwcm x nwcn x nwcfr
      i1 = locc;
      for k = 1:nwcfr
        for j = 1:nwcn
          for i = 1:nwcm
            d(i,j,k) = c(i1);
            i1 = i1 + 1;
          end
        end
      end
    else
      % ... but detail coefficients are stored as ncwfr x nwcm x nwcn
      for k = 1:nwcfr
        for j = 1:nwcn
          for i = 1:nwcm
            i1 = locc - 1 + (j-1)*nwcfr*nwcm + (i-1)*nwcfr + k;
            d(i,j,k) = c(i1);
          end
        end
      end
    end

    % Print out the selected set of coefficients
    fprintf('Level %d, Coefficients %d:\n', ilevel, itype_coeffs);
    for k = 1:nwcfr
      fprintf('Frame %d:\n', k);
```

```
      for i = 1:nwcm
        for j=1:nwcn
          fprintf('%8.4f ', d(i, j, k));
        end
        fprintf('\n');
      end
    end

  end

  end
end

% Reconstruct original data
[b, ifail] = c09fd(nwl, c, m, n, fr, icomm);

% Check reconstruction matches original
eps = 10*double(m*n*fr)*x02aj;
err = a-b;
frob = 0;
for i=1:fr
  fnew = sqrt(sum(sum(err(:,:,i).^2)));
  frob = max(frob,fnew);
end

if frob < eps
  fprintf('\nSuccess: the reconstruction matches the original.\n');
else
  fprintf('\nFail: Frobenius norm of b-a is too large.\n');
end
```

## 9.2 Program Results

```
     c09fd example results

 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.
```