The DHT in two dimensions.

(1) Here instead of our initial data being a vector $c_0(k)$, it is a matrix $c_0(n, m)$. The first step of the one-dimensional DHT is applied first to each *row* of the matrix, then to each *column* of the resulting matrix. (Or equivalently, first to each *column* then to each *row*).

This amounts to a *four-way splitting* of the initial matrix c_0 into four submatrices each of size 1/4 of the original matrix containing the *approximation coefficients*, the *horizontal detail coefficients*, the *vertical detail coefficients*, and the *diagonal detail coefficients*.

(2) Let's see how MATLAB does it.

```
>> help dwt2

 DWT2 Single-level discrete 2-D wavelet transform.
    DWT2 performs a single-level 2-D wavelet decomposition
    with respect to either a particular wavelet ('wname',
    see WFILTERS for more information) or particular wavelet filters
    (Lo_D and Hi_D) you specify.

    [CA,CH,CV,CD] = DWT2(X,'wname') computes the approximation
    coefficients matrix CA and details coefficients matrices
    CH, CV, CD, obtained by a wavelet decomposition of the
    input matrix X.
    'wname' is a string containing the wavelet name.
```

(3) Small MATLAB example.

```
>> c0=ones(8,8)
c0 =
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
>> [c1 d11 d12 d13]=dwt2(c0,'haar')
c1 =
    2.0000    2.0000    2.0000    2.0000
    2.0000    2.0000    2.0000    2.0000
    2.0000    2.0000    2.0000    2.0000
    2.0000    2.0000    2.0000    2.0000
d11 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
d12 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
d13 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

```
>> c0=[1 1 1 1 1 1 1 1
      -1 -1 -1 -1 -1 -1 -1 -1
       1 1 1 1 1 1 1 1
      -1 -1 -1 -1 -1 -1 -1 -1
       1 1 1 1 1 1 1 1
      -1 -1 -1 -1 -1 -1 -1 -1
       1 1 1 1 1 1 1 1
      -1 -1 -1 -1 -1 -1 -1 -1];
>> [c1 d11 d12 d13]=dwt2(c0,'haar')
c1 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
d11 =
    2.0000    2.0000    2.0000    2.0000
    2.0000    2.0000    2.0000    2.0000
    2.0000    2.0000    2.0000    2.0000
    2.0000    2.0000    2.0000    2.0000
d12 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
d13 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

The submatrix `d11` contains the horizontal edges of `c0`.

```
>> c0=[1 -1 1 -1 1 -1 1 -1
       1 -1 1 -1 1 -1 1 -1
       1 -1 1 -1 1 -1 1 -1
       1 -1 1 -1 1 -1 1 -1
       1 -1 1 -1 1 -1 1 -1
       1 -1 1 -1 1 -1 1 -1
       1 -1 1 -1 1 -1 1 -1
       1 -1 1 -1 1 -1 1 -1];
>> [c1 d11 d12 d13]=dwt2(c0,'haar')
c1 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
d11 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
d12 =
    2.0000    2.0000    2.0000    2.0000
    2.0000    2.0000    2.0000    2.0000
    2.0000    2.0000    2.0000    2.0000
    2.0000    2.0000    2.0000    2.0000
d13 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

The submatrix `d12` contains the vertical edges of `c0`.

```
>> c0=[1 -1 1 -1 1 -1 1 -1
       -1 1 -1 1 -1 1 -1 1
       1 -1 1 -1 1 -1 1 -1
       -1 1 -1 1 -1 1 -1 1
       1 -1 1 -1 1 -1 1 -1
       -1 1 -1 1 -1 1 -1 1
       1 -1 1 -1 1 -1 1 -1
       -1 1 -1 1 -1 1 -1 1];
>> [c1 d11 d12 d13]=dwt2(c0,'haar')
c1 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
d11 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
d12 =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
d13 =
    2.0000    2.0000    2.0000    2.0000
    2.0000    2.0000    2.0000    2.0000
    2.0000    2.0000    2.0000    2.0000
    2.0000    2.0000    2.0000    2.0000
```

The submatrix `d13` contains the diagonal edges of `c0`.

(4) More MATLAB commands.

```
>> help wavedec2

 WAVEDEC2 Multilevel 2-D wavelet decomposition.
    [C,S] = WAVEDEC2(X,N,'wname') returns the wavelet
    decomposition of the matrix X at level N, using the
    wavelet named in string 'wname' (see WFILTERS).
    Outputs are the decomposition vector C and the
    corresponding bookkeeping matrix S.
    N must be a strictly positive integer (see WMAXLEV).

    The output wavelet 2-D decomposition structure [C,S]
    contains the wavelet decomposition vector C and the
    corresponding bookeeping matrix S.
    Vector C is organized as:
      C = [ A(N)   | H(N)   | V(N)   | D(N) | ...
    H(N-1) | V(N-1) | D(N-1) | ...  | H(1) | V(1) | D(1) ].
      where A, H, V, D, are row vectors such that:
    A = approximation coefficients,
    H = hori. detail coefficients,
    V = vert. detail coefficients,
    D = diag. detail coefficients,
    each vector is the vector column-wise storage of a matrix.
    Matrix S is such that:
      S(1,:) = size of app. coef.(N)
      S(i,:) = size of det. coef.(N-i+2) for i = 2,...,N+1
      and S(N+2,:) = size(X).
```

```
>> help wrcoef2

 WRCOEF2 Reconstruct single branch from 2-D wavelet coefficients.
    WRCOEF2 reconstructs the coefficients of an image.

    X = WRCOEF2('type',C,S,'wname',N) computes the matrix of
    reconstructed coefficients of level N, based on the
    wavelet decomposition structure [C,S] (see WAVEDEC2 for
    more information).
    'wname' is a string containing the name of the wavelet.
    If 'type' = 'a', approximation coefficients are reconstructed
    otherwise if 'type' = 'h' ('v' or 'd', respectively),
    horizontal (vertical or diagonal, respectively) detail
    coefficients are reconstructed.

    Level N must be an integer such that:
    0 <= N <= size(S,1)-2 if 'type' = 'a' and such that
    1 <= N <= size(S,1)-2 if 'type' = 'h', 'v'or 'd'.

    X = WRCOEF2('type',C,S,'wname') or
    X = WRCOEF2('type',C,S,Lo_R,Hi_R) reconstruct
    coefficients of maximum level N = size(S,1)-2.
```

```
>> help appcoef2

APPCOEF2 Extract 2-D approximation coefficients.
   APPCOEF2 computes the approximation coefficients of a
   two-dimensional signal.

   A = APPCOEF2(C,S,'wname',N) computes the approximation
   coefficients at level N using the wavelet decomposition
   structure [C,S] (see WAVEDEC2).
   'wname' is a string containing the wavelet name.
   Level N must be an integer such that 0 <= N <= size(S,1)-2.

   A = APPCOEF2(C,S,'wname') extracts the approximation
   coefficients at the last level size(S,1)-2.

>> help detcoef2

DETCOEF2 Extract 2-D detail coefficients.
   D = DETCOEF2(O,C,S,N) extracts from the wavelet
   decomposition structure [C,S], the horizontal, vertical
   or diagonal detail coefficients for O = 'h'
   (or 'v' or 'd',respectively), at level N. N must
   be an integer such that 1 <= N <= size(S,1)-2.
   See WAVEDEC2 for more information on C and S.

   [H,V,D] = DETCOEF2('all',C,S,N) returns the horizontal H,
   vertical V, and diagonal D detail coefficients at level N.

   D = DETCOEF2('compact',C,S,N) returns the detail
   coefficients at level N, stored row-wise.

   DETCOEF2('a',C,S,N) is equivalent to DETCOEF2('all',C,S,N).
   DETCOEF2('c',C,S,N) is equivalent to DETCOEF2('compact',C,S,N).
```