Single Level Discrete 1-D Wavelet Transform

MatDeck's function, discretewavet(), performs a single, level 1-D wavelet decomposition. The decomposition is done using certain wavelet decomposition filters which are defined by user. The wavelet filter coefficients can be determined independently by using the wavefilter() function. There are more than 100 built-in wavelet filters in MatDeck. discretewavet() is very easy to use and get started with. An input signal can be defined as a column vector but also as row vector.

The first test signal is generated as a white Gaussian noise with a mean value equal to zero and the variance equal to unity. The length of the signal is 10. The function, normrandvec(), returns the result as a column vector.

```
xin := normrandvec(0 , 1 , 10)      Test signal generated as white Gaussian noise
yinp := [1  0  0  0  0  0  0]       Test signal as row vector
Xin := discretewavet(xin , "db2")   Discerete wavelet transform using Daubechies wavlet filter db2
wname := "db3"                      Daubechies wavlet filter db3
Yinp := discretewavet(yinp , wname) Discrete wavelet transform using wavlet wname
```

The function, discretewavet(), returns a two-column matrix. The first column, with the column index equal to zero, represents the approximation coefficients, while the second column represents the detail coefficient of the wavelet transform.

```
Cap := col2vec(Yinp , 0)
Cde := col2vec(Yinp , 1)
```

Inverse single level one-dimensional wavelet transform

The digital signal is processed in the wavelet domain in order to remove the noise and perform compression. After the signal processing is done, the wavelet domain signal must be reconstructed and converted back into the time domain. The signal reconstruction is done using a inverse single level discrete wavelet transform. MatDeck's function, discretewaveti(), is used for this purpose. The function, discretewaveti(), takes the approximation and detail coefficients as an input, and the appropriate indication of the wavelet is used. It returns the reconstructed signal which should be the same size and value as the original input signal.

```
yr := discretewaveti(Cap , Cde , wname)
yr^T = [1  0  0  -3.469e-18  -3.469e-18  0  0]
```

We can perform the inverse transform of the decomposed noise signal as well:

```
xr := discretewaveti(col2vec(Xin , 0) , col2vec(Xin , 1) , "db2")
```