Lab Report

Course Title : Sessional Based on CSE 4105
Course No. : CSE 4106

<u>Submitted by</u>

Name: Md. Shabir Khan Akash.

Class: 4th year, Odd Semester.

Department : CSE

Roll: 1603108

Section: B

<u>Submitted to</u>

Abu Sayeed

Assistant Professor

Department of CSE, RUET

Discrete Wavelet Transform

The Discrete Wavelet Transform decomposes a digital signal into different sub bands so that the lower frequency sub bands have finer frequency resolution and coarser time resolution compared to the higher frequency sub bands. The wavelet transform is a time-frequency analysis method which selects the appropriate frequency band adaptively based on the characteristics of the signal. Then the frequency band matches the spectrum which improves the time-frequency resolution. Discrete Wavelet Transform (DWT) is the basis of the new JPEG2000 image compression standard. So, basically A discrete wavelet transform (DWT) is a transform that decomposes a given signal into a number of sets, where each set is a time series of coefficients describing the time evolution of the signal in the corresponding frequency band.

The wavelet applications mentioned include numerical analysis, signal analysis, control applications and the analysis and adjustment of audio signals. Discrete Wavelet Transform is a wavelet transform for which the wavelets are sampled at discrete intervals. Discrete Wavelet Transform provides a simultaneous spatial and frequency domain information of the image. In DWT operation, an image can be analyzed by the combination of analysis filter bank and decimation operation. There are two types of DWT -

- Multi-level Discrete Wavelet Transform and Inverse Transform
- Single Level Discrete Wavelet Transform and Inverse Transform

Multi-level Discrete Wavelet Transform and Inverse Transform:

Here, co_effs = pywt.wavedec(x,'wavelet',mode= 'sym',level =n)

Where, [cA(n), cD(c),cD(n-1),.....,cD2,cD1] = co_effs.

When, level = 3 , it means that there will be 3 decompositions of the signal.

x is the input and wavelet is the decomposition that we want to use.

mode is by default symmetric. It can also be periodic etc.

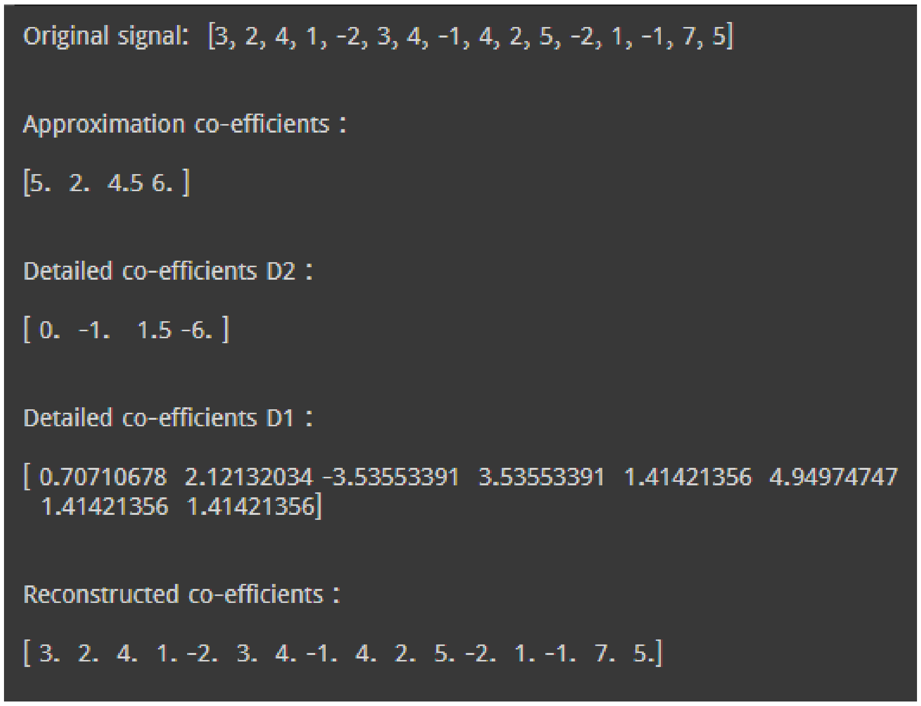
y= pywt.waverec(coeffs,'wavelet',mode = 'sym') , here this is used for inverse wavelength transform.

Code:

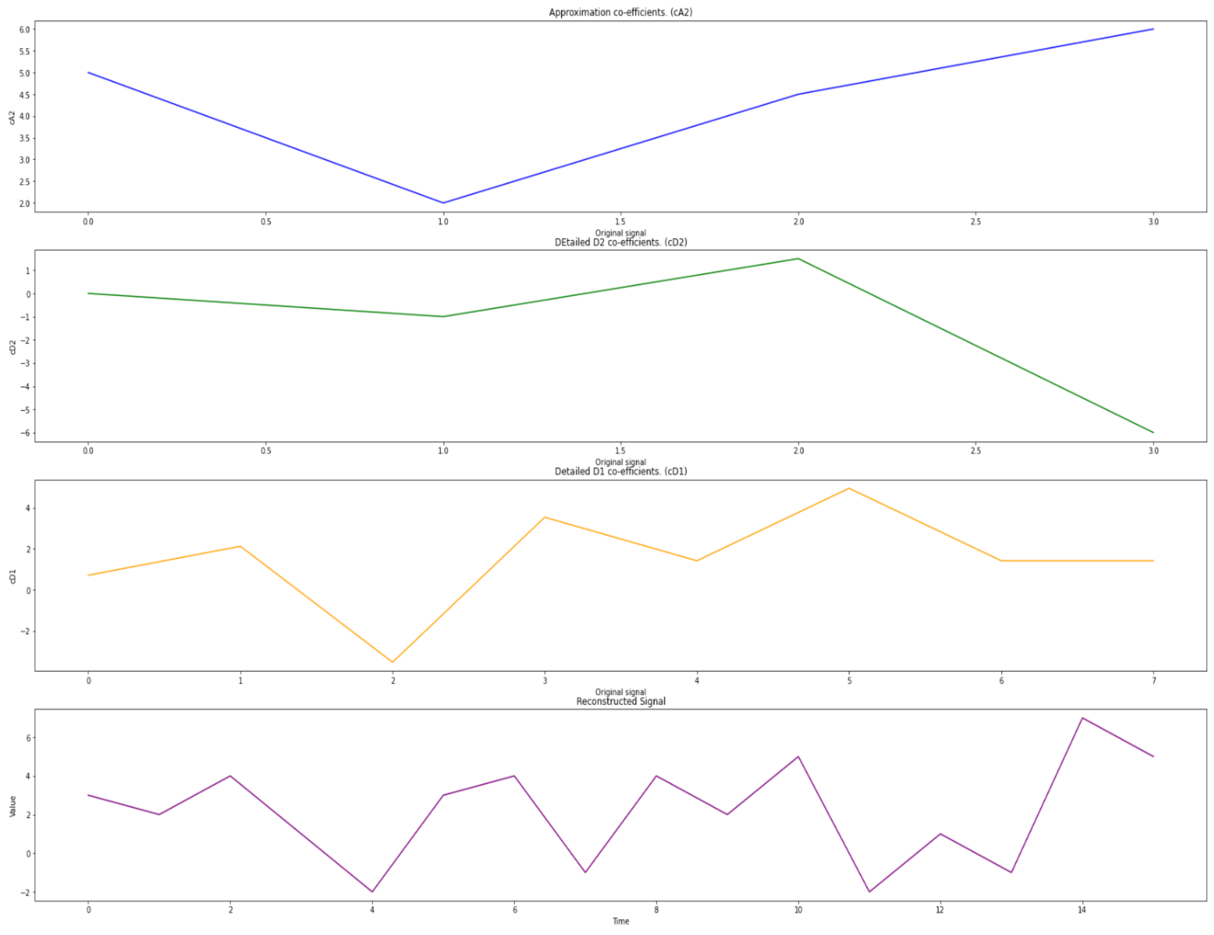
```
import numpy as np
import matplotlib.pyplot as plt
import pywt
import pywt.data

a=[3,2,4,1,-2,3,4,-1,4,2,5,-2,1,-1,7,5];


print("Original signal: ",a);
coeffs = pywt.wavedec(a,'db1', level=2, mode='periodic')
cA2,cD2,cD1 = coeffs;
b = pywt.waverec(coeffs,'db1', mode = 'periodic')

plt.figure(figsize=(30,20));


print("\n");
print("Approximation co-efficients : \n");
print(cA2);
print("\n");


plt.subplot(4,1,1)
plt.plot(cA2,color = 'blue');
plt.xlabel('Original signal');
plt.ylabel('cA2');
plt.title("Approximation co-efficients. (cA2)");


print("Detailed co-efficients D2 : \n");
print(cD2);
print("\n");

plt.subplot(4,1,2)
plt.plot(cD2,color = green);
plt.xlabel('Original signal');
plt.ylabel('cD2');
plt.title("DEtailed D2 co-efficients. (cD2)");


print("Detailed co-efficients D1 : \n");
print(cD1);
print("\n");
```

```
plt.subplot(4,1,3)
plt.plot(cD1,color = 'orange');
plt.xlabel('Original signal');
plt.ylabel('cD1');
plt.title("Detailed D1 co-efficients. (cD1)");


print("Reconstructed co-efficients : \n");
print(b);
print("\n");
plt.subplot(4,1,4)
plt.plot(b,color = 'purple');
plt.xlabel('Time');
plt.ylabel('Value');
plt.title("Reconstructed Signal");

plt.show()
```

Output:



```
Original signal:  [3, 2, 4, 1, -2, 3, 4, -1, 4, 2, 5, -2, 1, -1, 7, 5]


Approximation co-efficients :

[5.  2.  4.5 6. ]


Detailed co-efficients D2 :

[ 0.  -1.   1.5 -6. ]


Detailed co-efficients D1 :

[ 0.70710678  2.12132034 -3.53553391  3.53553391  1.41421356  4.94974747
  1.41421356  1.41421356]


Reconstructed co-efficients :

[ 3.  2.  4.  1. -2.  3.  4. -1.  4.  2.  5. -2.  1. -1.  7.  5.]
```



<u>Single Level Discrete Wavelet Transform and Inverse Transform :</u>

[cA, cD] = dwt(x, wname) ,

It returns the single-level discrete wavelet transform (DWT) of the vector x using the wavelet specified by wname. The wavelet must be recognized by wavemngr. Dwt function returns two values. These are -

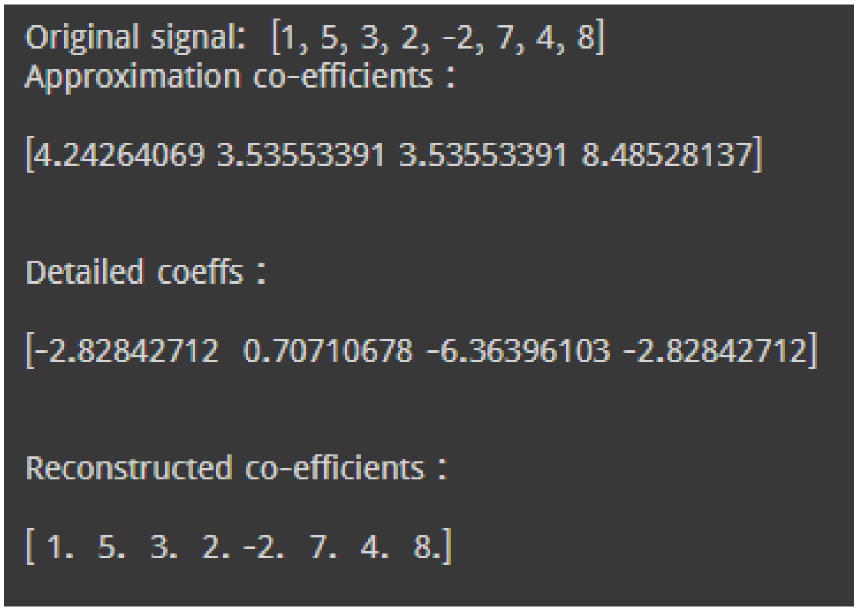
the approximation coefficients vector cA and

detail coefficients vector cD of the Discrete Wavelet Transform.

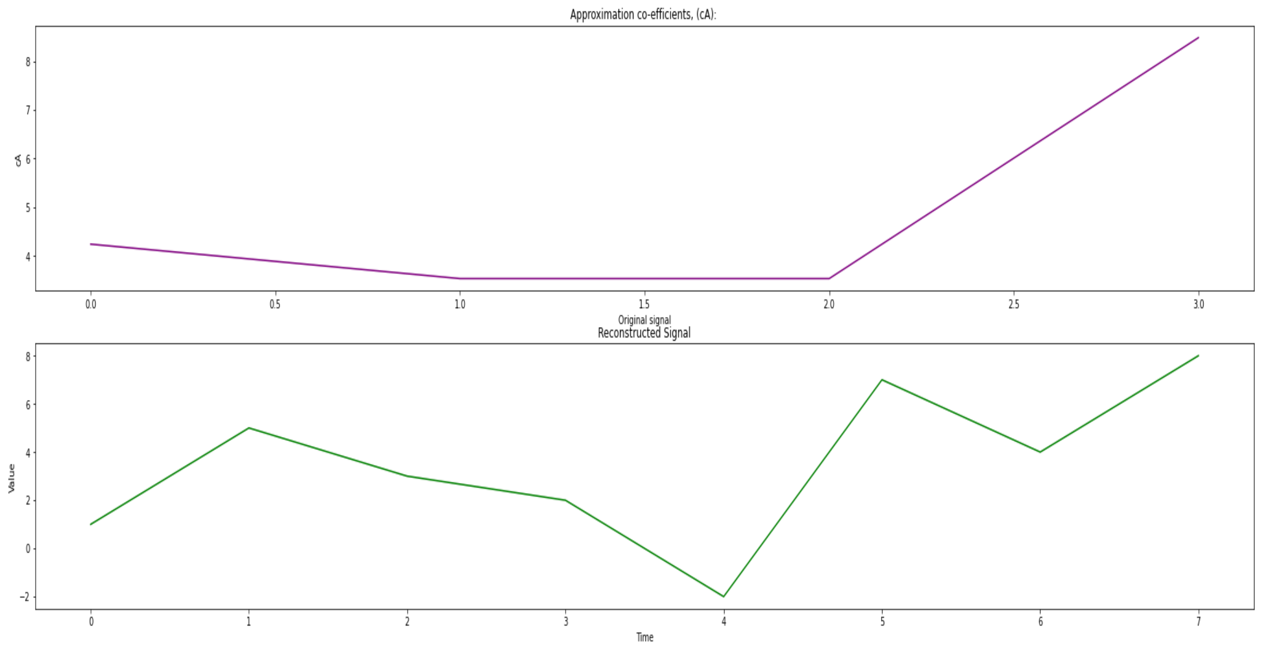
Code:

```
import numpy as np
import matplotlib.pyplot as plt
import pywt

sig = [1,5,3,2, -2, 7,4 ,8]
print( "Original signal: ", sig);

cA,cD = pywt.dwt(sig,'haar')
sigInv = pywt.idwt(cA,cD, 'haar')

plt.figure(figsize=(30,20));

print("Approximation co-efficients : \n");
print(cA);
print("\n");

print("Detailed coeffs : \n");
print(cD);
print("\n");

print("Reconstructed co-efficients : \n");
print(sigInv);
print("\n");


plt.subplot(4,1,1)
plt.plot(cA,color = 'purple');
plt.xlabel('Original signal');
plt.ylabel('cA');
plt.title("Approximation co-efficients, (cA): ");
print("\n");

plt.subplot(4,1,2)
plt.plot(sigInv,color = 'green');
plt.xlabel('Time');
plt.ylabel('Value');
plt.title("Reconstructed Signal");

plt.show()
```

Output:



```
Original signal:  [1, 5, 3, 2, -2, 7, 4, 8]
Approximation co-efficients :

[4.24264069 3.53553391 3.53553391 8.48528137]


Detailed coeffs :

[-2.82842712  0.70710678 -6.36396103 -2.82842712]


Reconstructed co-efficients :

[ 1.  5.  3.  2. -2.  7.  4.  8.]
```


Approximation co-efficients, (cA):
cA
Original signal
Reconstructed Signal
Value
Time