Introduction to Image Processing

Prof. Alexandre Zaghetto
alexandre.zaghetto@mcgill.ca

McGill University
Department of Electrical and Computer Engineering

Topic 07
Image Transforms

1. Introduction

- In some cases, image **processing tasks** are best formulated by **transforming** the input images, carrying the specified task in a **transform** domain, and applying the **inverse transform** to return to the spatial domain.

- A particularly important class of 2-D linear transforms, can be expressed in the general form.

$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) r(x, y, u, v)$$

where *f* is the input image, ***r*** is called the **forward transformation** kernel, and the equation is evaluated for and *u* = 0, 1,…, M–1 and *v* = = 0, 1,..., N-1.

1. Introduction

- *T* is called the **forward transform** of *f*.

- Given *T* we can recover *f* using the **inverse transform**.

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v)s(x, y, u, v)$$

- The equation is evaluated for and *x*=0, 1,…, M–1 and *y*=0, 1,..., N-1.

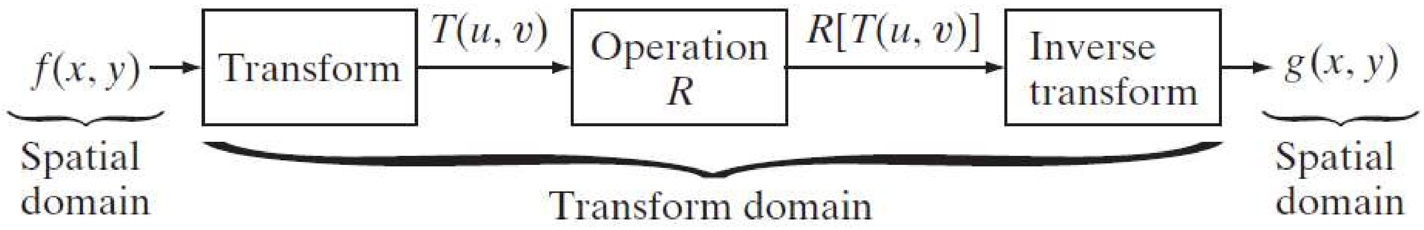
- And ***s*** is called the **inverse transformation** kernel.

1. Introduction

- By using these transforms, it is possible to express an image as a **combination** of a set **of basic signals**, known as the **basis functions**.

- The image output in the **transformed space** may be analyzed, interpreted, and further processed for implementing **diverse** image processing **tasks**.

- General approach for operating in the linear transform domain.



2. Discrete Fourier Transform

• Substituting the kernels below into the previous equations yields the **Discrete Fourier transform** pair

$$r(x, y, u, v) = e^{-j2\pi(ux+vy)/n}$$

$$s(x, y, u, v) = \frac{1}{n^2} e^{j2\pi(ux+vy)/n}$$

$$M = N = n$$

• The Discrete Fourier Transform was discussed in "**Topic 04** - Filtering in the Frequency Domain".

2. Discrete Fourier Transform

- The 2-D Discrete Fourier Transform and its inverse.

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)}$$

$$u = 0, 1, 2, \ldots, M-1 \text{ and } v = 0, 1, 2, \ldots, N-1.$$

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M + vy/N)}$$

$$x = 0, 1, 2, \ldots, M-1 \text{ and } y = 0, 1, 2, \ldots, N-1.$$

- DFT uses a set of complex exponential functions.

- Normally used for general spectral analysis applications.

3. Discrete Cosine Transform

• **Discrete Cosine Transform** (DCT) is the basis for many image and video **compression algorithms**, especially the baseline JPEG and MPEG standards for compression of **still** and **video** images respectively.

• It is obtained by using the following (equal) kernels:

$$r(x, y, u, v) = s(x, y, u, v)$$

$$= \alpha(u)\alpha(v) \cos\left[\frac{(2x + 1)u\pi}{2n}\right] \cos\left[\frac{(2y + 1)v\pi}{2n}\right]$$

$$\alpha(u) = \begin{cases} \sqrt{\dfrac{1}{n}} & \text{for } u = 0 \\ \sqrt{\dfrac{2}{n}} & \text{for } u = 1, 2, \dots, n - 1 \end{cases}$$

3. Discrete Cosine Transform

- DCT uses only (real-valued) cosine functions.
- It translates the correlated data to uncorrelated data.
- The DCT and inverse DCT can be computed using the DFT.
- Example: 8 basis functions of a 4-by-4 matrix.

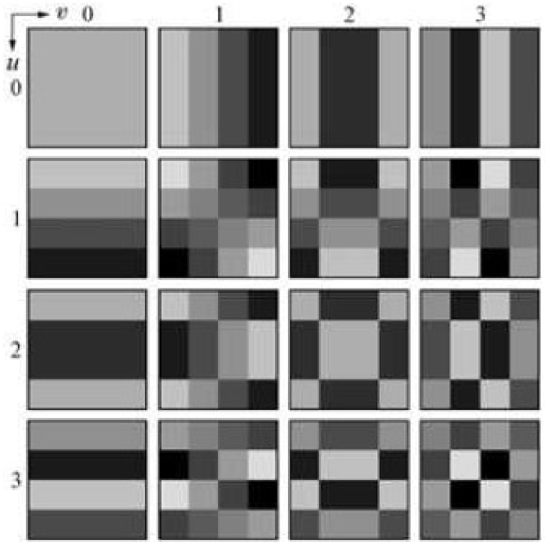
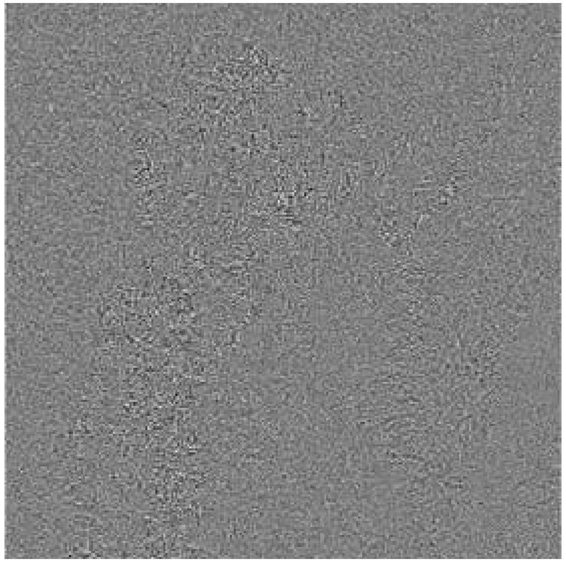


FIGURE 8.23 Discrete-cosine basis functions for $n = 4$. The origin of each block is at its top left.

3. Discrete Cosine Transform

- Example: approximations of the a monochrome image.

  - The result was obtained by **dividing** the original image into subimages of **size 8 x 8** using the DCT, **truncating** 50% of the resulting coefficients, and **taking the inverse transform** of the truncated coefficient arrays.



3. Discrete Cosine Transform

• **Mean-square** reconstruction **error** is related directly to the **energy** or **information** packing properties of the transform employed.

• An image *g(x, y)* can be expressed as a function of its 2-D transform

$$g(x, y) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) s(x, y, u, v)$$

$$x, y = 0, 1, 2, \dots, n - 1$$

• The inverse kernel *s* can be viewed as defining a set of **basis functions** or **basis images**.

3. Discrete Cosine Transform

- This interpretation becomes clearer if we use the notation:

$$\mathbf{G} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v)\, \mathbf{S}_{uv}$$

$$\mathbf{S}_{uv} = \begin{bmatrix} s(0, 0, u, v) & s(0, 1, u, v) & \cdots & s(0, n-1, u, v) \\ s(1, 0, u, v) & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & & & \\ s(n-1, 0, u, v) & s(n-1, 1, u, v) & \cdots & s(n-1, n-1, u, v) \end{bmatrix}$$

- **G** contains the pixels of the image and is defined as a linear combination of n^2 matrices of size $n \times n$ that is, $\boldsymbol{S}_{uv}$, for u, v = 0, 1, 2,…,n-1.

3. Discrete Cosine Transform

- We can **define** a transform coefficient **masking function** χ which is constructed to **eliminate** the **basis images** that make the smallest contribution to the total sum

$$\chi(u, v) = \begin{cases} 0 & \text{if } T(u, v) \text{ satisfies a specified truncation criterion} \\ 1 & \text{otherwise} \end{cases}$$

$$\hat{\mathbf{G}} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \chi(u, v) T(u, v) \mathbf{S}_{uv}$$

- The mean-square error between $\mathbf{G}$ and $\hat{\mathbf{G}}$ approximation is

$$e_{ms} = E\left\{ \|\mathbf{G} - \hat{\mathbf{G}}\|^2 \right\} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \sigma^2_{T(u,v)} \left[1 - \chi(u, v) \right]$$

- The total mean-square approximation error thus is the sum of the variances of the discarded transform coefficients.

3. Discrete Cosine Transform

- Transformations that redistribute or pack the most information into the fewest coefficients provide the smallest reconstruction errors.

4. Discrete Wavelet Transform

• Signal analysts already have at their disposal an impressive arsenal of tools. Perhaps the most well-known of these is **Fourier analysis**.

• Fourier analysis has a **serious drawback**. In transforming to the frequency domain, **time information is lost**.

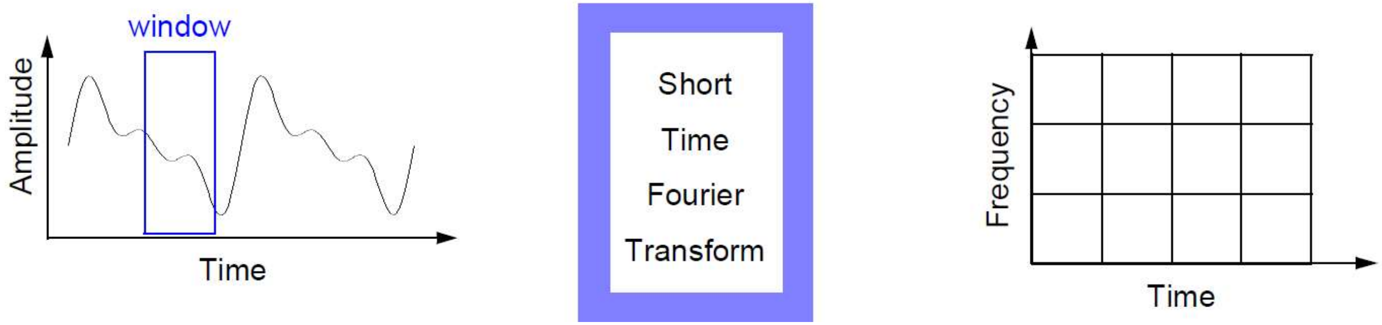
• If a signal does not change much over time this drawback is not very important.

4. Discrete Wavelet Transform

- However, most **interesting signals** contain numerous non-stationary or **transitory characteristics**.

- In an effort to **correct** this deficiency, Dennis Gabor (1946) adapted the Fourier transform to analyze only **a small section** of the signal at a **time**.

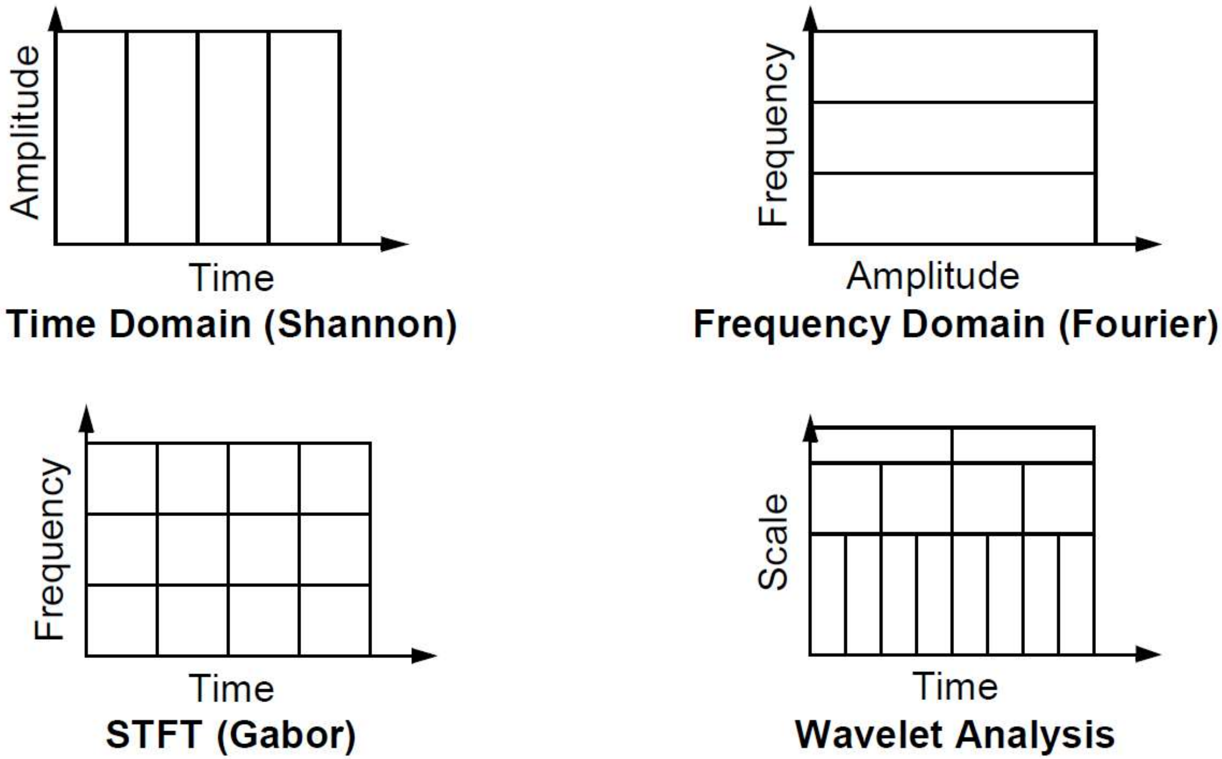
  - Short-Time Fourier Transform (STFT)



- Precision is determined by the size of the window.

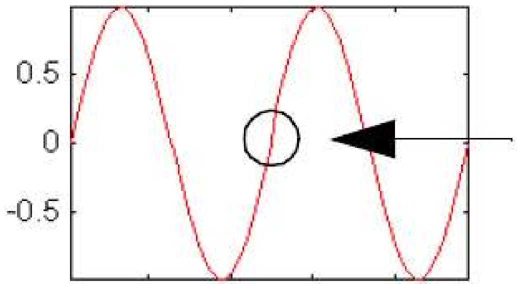
4. Discrete Wavelet Transform

- **Wavelet analysis** represents the next logical step: a windowing technique with variable-sized regions.

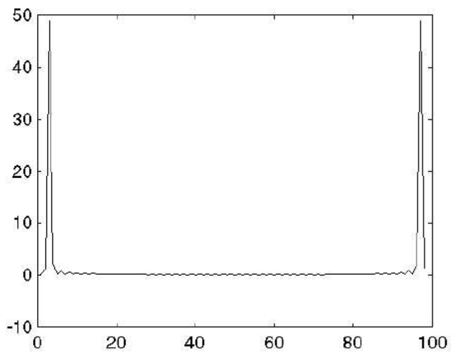


4. Discrete Wavelet Transform

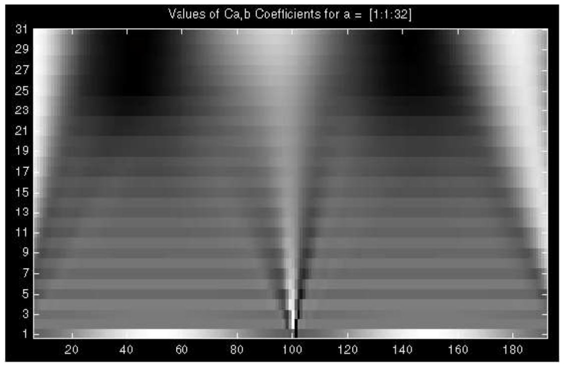
- One major **advantage** afforded by wavelets is the ability to perform **local analysis** — that is, to analyze a localized area of a larger signal.



Sinusoid with a small discontinuity



Fourier Coefficients



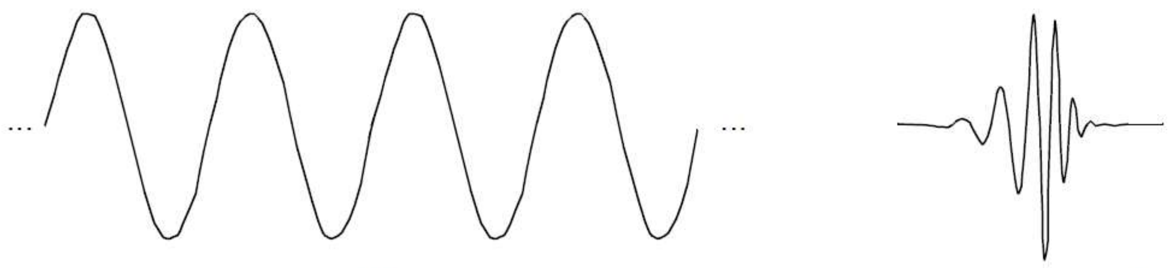
Wavelet Coefficients

4. Discrete Wavelet Transform

• A wavelet is a waveform of effectively **limited duration** that has an **average** value of **zero**.

• Compare wavelets with sine waves, which are the basis of Fourier analysis:

- ➢ Sinusoids do not have limited duration.
- ➢ Sinusoids are smooth and predictable.
- ➢ Wavelets tend to be irregular and asymmetric.

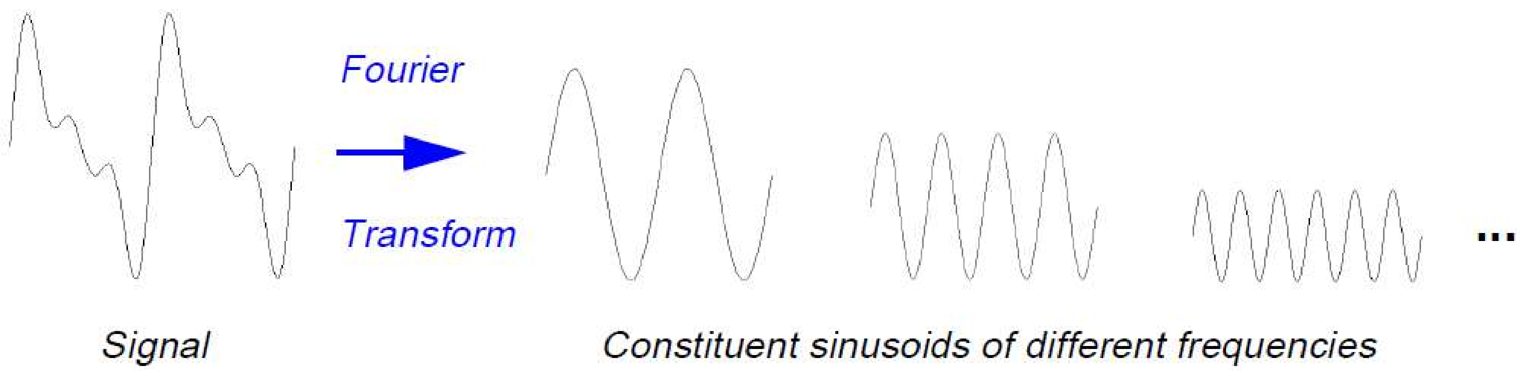


Sine Wave Wavelet (db10)

4. Discrete Wavelet Transform

• Fourier analysis consists of breaking up a signal into sine waves of various frequencies.

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$



4. Discrete Wavelet Transform

- The **Continuous Wavelet Transform** (CWT) of a continuous function *f(x)*, relative to a real-valued wavelet, $\psi(x)$, is defined as

$$W_{\psi}(s, \tau) = \int_{-\infty}^{\infty} f(x)\psi_{s,\tau}(x)\, dx$$

$$\psi_{s,\tau}(x) = \frac{1}{\sqrt{s}}\psi\left(\frac{x - \tau}{s}\right)$$

where s and τ are called **scale** and **translation** parameters, respectively.

4. Discrete Wavelet Transform

- The function *f(x)* can be obtained using the **inverse** Continuous Wavelet Transform,

$$f(x) = \frac{1}{C_\psi} \int_0^\infty \int_{-\infty}^\infty W_\psi(s, \tau) \frac{\psi_{s,\tau}(x)}{s^2} d\tau\, ds$$

$$C_\psi = \int_{-\infty}^\infty \frac{|\Psi(\mu)|^2}{|\mu|} d\mu$$

where $\Psi(\mu)$ is the Fourier transform of $\psi(x)$.

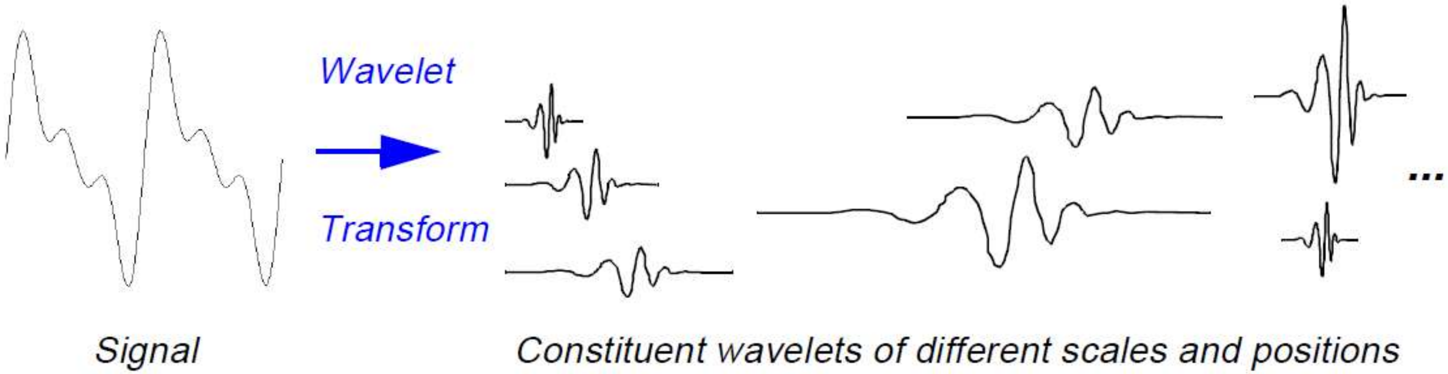
- The previous equations define a reversible transformation as long as the so-called admissibility criterion is satisfied,

$$C_\psi < \infty$$

4. Discrete Wavelet Transform

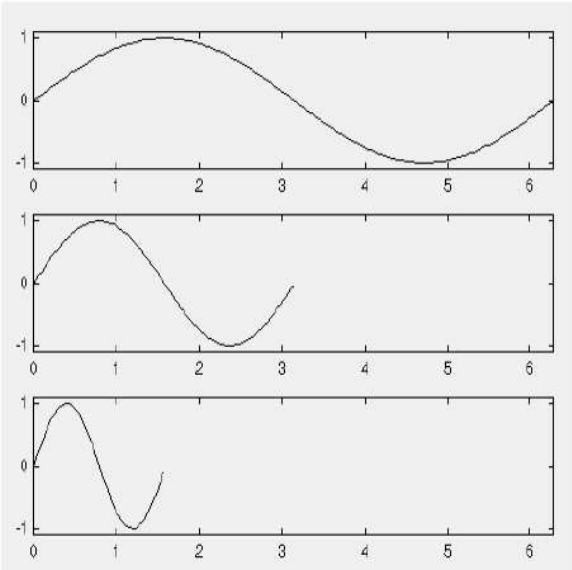
• Similarly, wavelet analysis is the breaking up of a signal into **shifted** and **scaled** versions of the original (or **mother**) wavelet.

$$C(scale, position) = \int_{-\infty}^{\infty} f(t)\psi(scale, position, t)dt$$



4. Discrete Wavelet Transform

- Scaling a wavelet simply means **stretching** (or compressing) it.



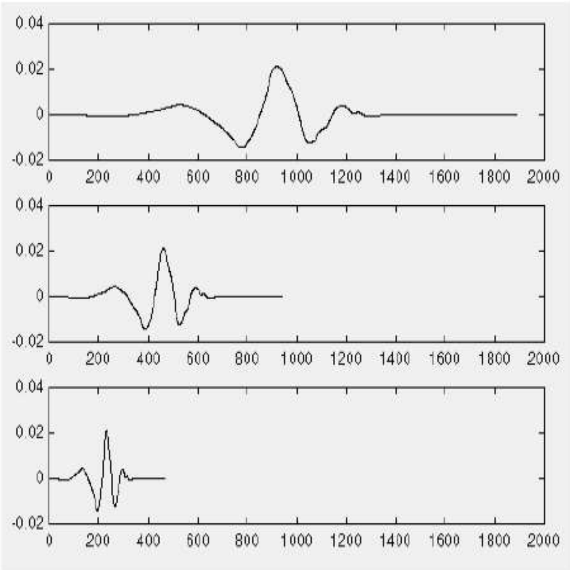
$$f(t) = \sin(t) \quad ; \quad a = 1$$

$$f(t) = \sin(2t) \quad ; \quad a = \frac{1}{2}$$

$$f(t) = \sin(4t) \quad ; \quad a = \frac{1}{4}$$

4. Discrete Wavelet Transform

- The scale factor works exactly the same with wavelets.



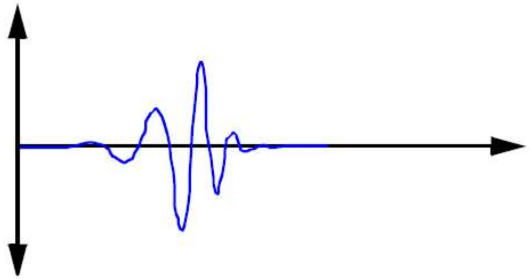
$$f(t) = \psi(t) \quad ; \quad a = 1$$

$$f(t) = \psi(2t) \quad ; \quad a = \frac{1}{2}$$

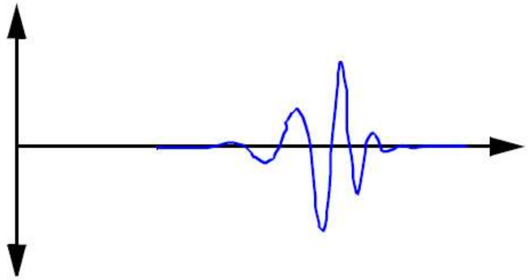
$$f(t) = \psi(4t) \quad ; \quad a = \frac{1}{4}$$

4. Discrete Wavelet Transform

• Shifting a wavelet simply means delaying (or hastening) its onset.



Wavelet function
$\psi(t)$



Shifted wavelet function
$\psi(t-k)$

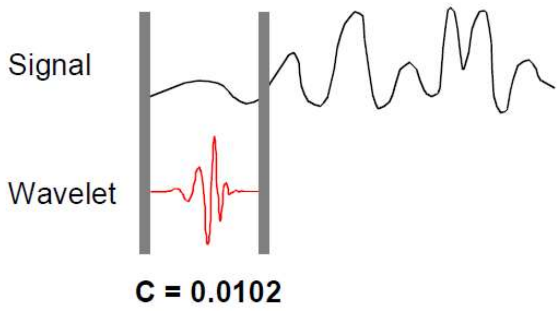
4. Discrete Wavelet Transform

• The CWT is the sum over all time of the signal multiplied by scaled, shifted versions of the wavelet.

• Computing the CWT:

1. Take a wavelet and **compare** it to a section at the start of the original signal.

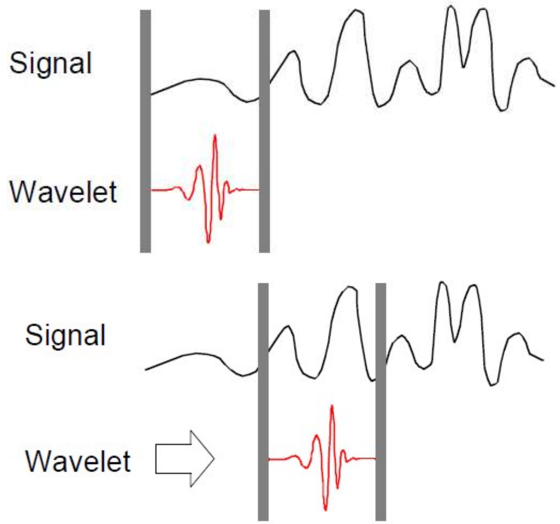
2. Calculate a number, C, that represents **how closely correlated** the wavelet is with this section of the signal.



4. Discrete Wavelet Transform

- Computing the CWT:

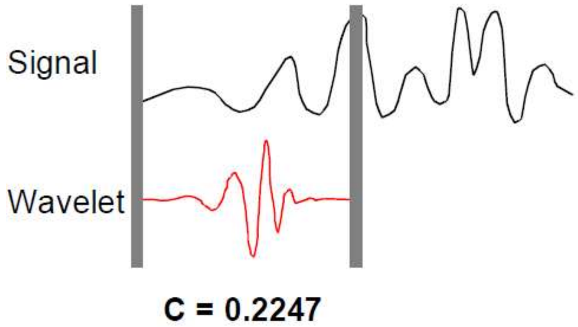
  3. Shift the wavelet to the right and repeat steps 1 and 2 until you've covered the whole signal.



4. Discrete Wavelet Transform

- Computing the CWT:

4. Scale (stretch) the wavelet and repeat steps 1 through 3.

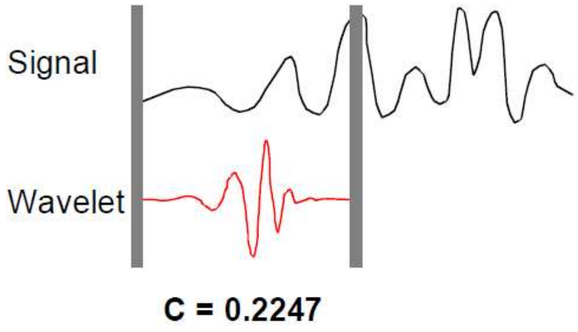


5. Repeat steps 1 through 4 for all scales.

4. Discrete Wavelet Transform

- Computing the CWT:

4. Scale (stretch) the wavelet and repeat steps 1 through 3.

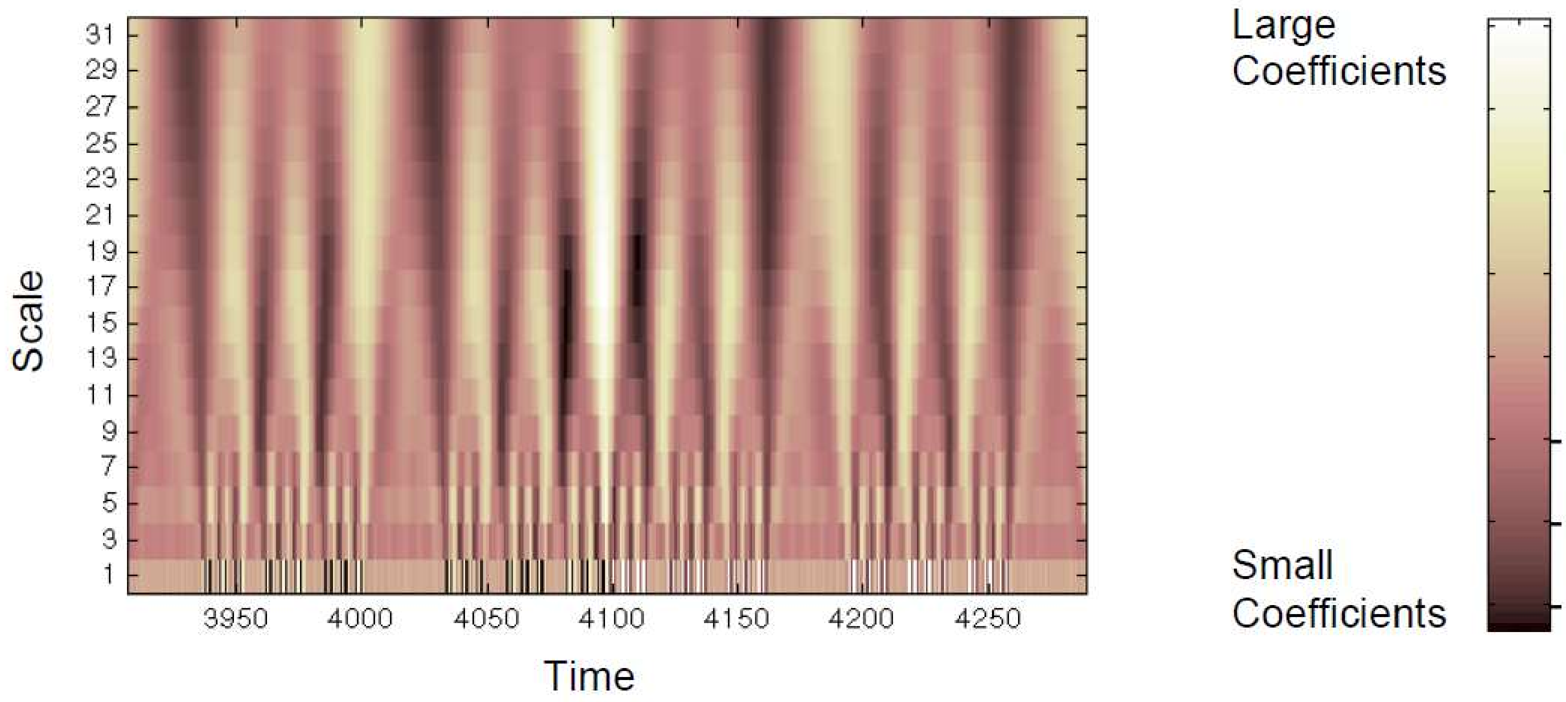


5. Repeat steps 1 through 4 for all scales.

- When you're done, you'll have the coefficients produced at different scales by different sections of the signal.

4. Discrete Wavelet Transform

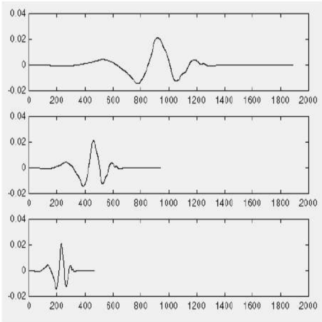
- How to make sense of all these coefficients?



4. Discrete Wavelet Transform

- There is a correspondence between wavelet scales and frequency:

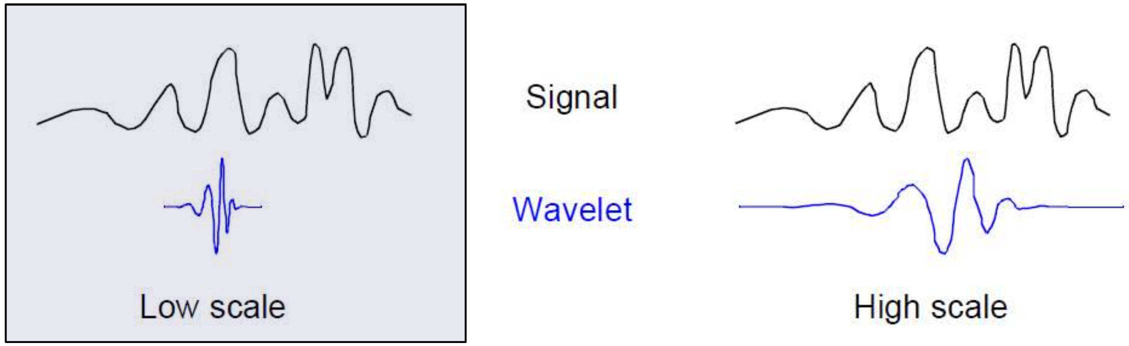
  - ➢ Low scale ➔ Compressed wavelet ➔ Rapidly changing details ➔ High frequency



$$f(t) = \psi(t) \quad ; \quad a = 1$$

$$f(t) = \psi(2t) \quad ; \quad a = \frac{1}{2}$$

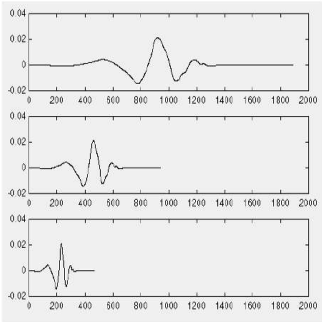
$$f(t) = \psi(4t) \quad ; \quad a = \frac{1}{4}$$



4. Discrete Wavelet Transform

- There is a correspondence between wavelet scales and frequency:

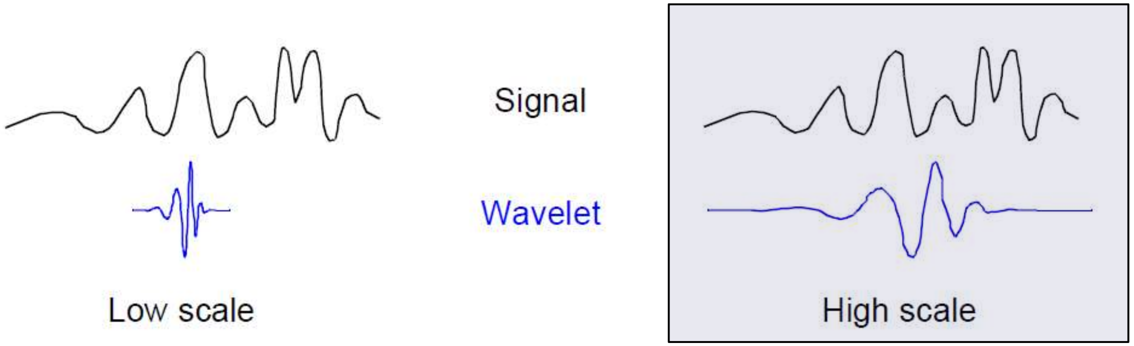
  - ➢ High scale a ➔ Stretched wavelet ➔ Slowly changing, coarse features ➔ Low frequency



$$f(t) = \psi(t) \quad ; \quad a = 1$$

$$f(t) = \psi(2t) \quad ; \quad a = \frac{1}{2}$$

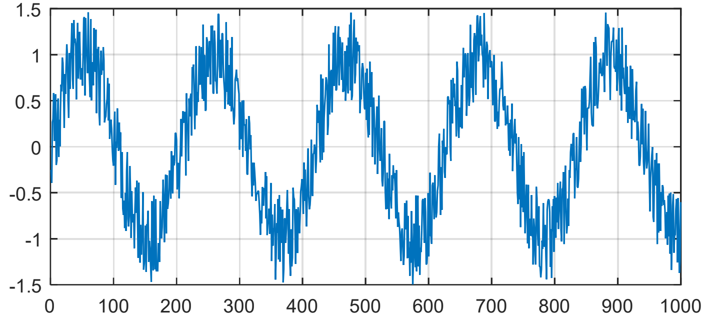
$$f(t) = \psi(4t) \quad ; \quad a = \frac{1}{4}$$

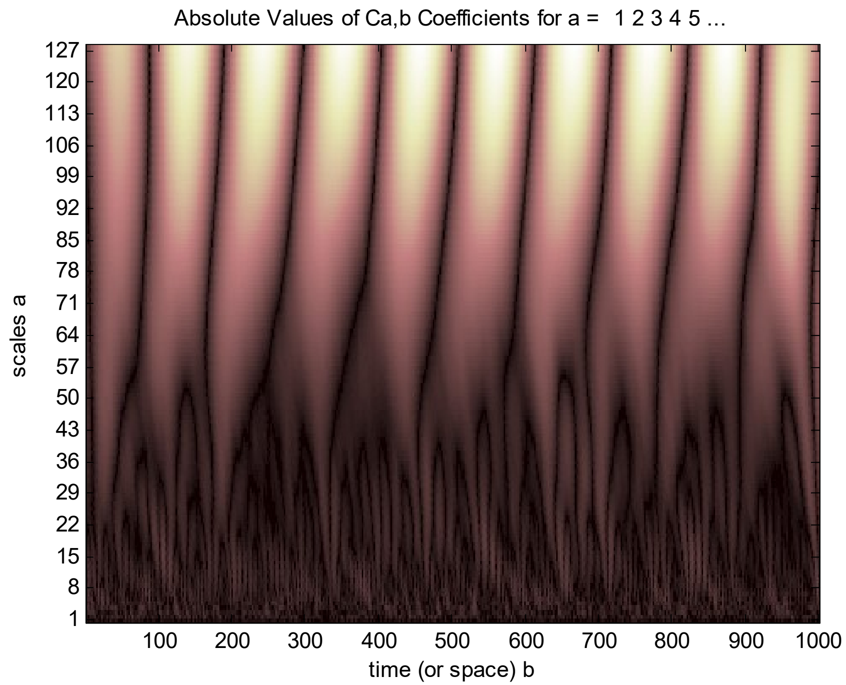


4. Discrete Wavelet Transform

- There is a correspondence between wavelet scales and frequency:

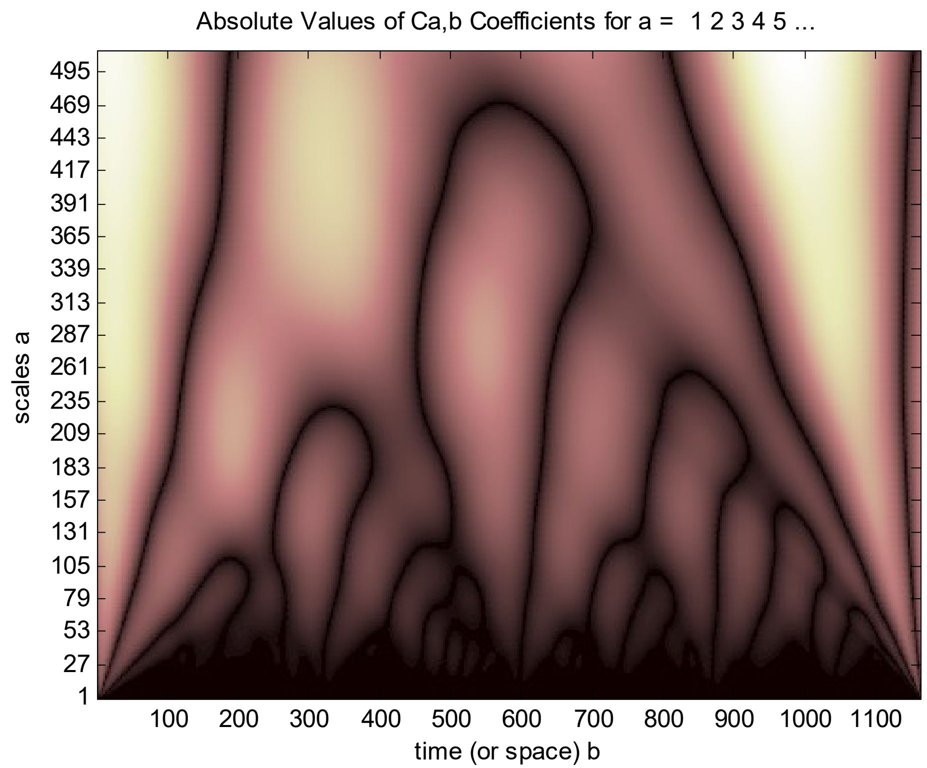
```
load noissin;
c = cwt(noissin,1:128,'db4','plot');
```





4. Discrete Wavelet Transform

- MATLAB: s32Lunar.m



4. Discrete Wavelet Transform

- What is "**continuous**" about the Continuous Wavelet Transform (CWT) are the **scales** at which it operates.

- CWT can operate at **every scale**, from that of the original signal up to some maximum scale which you determine.

- The CWT is also **continuous** in terms of **shifting**.

- What if we choose only a **subset** of **scales** and **positions** at which to make our calculations?

    - It turns out, that if we choose scales and **positions based** on **powers of two** then our analysis will just as **accurate**.

- We obtain such an analysis from the **Discrete Wavelet Transform** (DWT).

4. Discrete Wavelet Transform

- An efficient way to implement this scheme **using filters** was developed in 1988 by **Mallat**.

- For many signals, the **low-frequency** content is the most important part. It is what gives the signal its **identity**.

- The **high-frequency** content, on the other hand, imparts **nuance**.

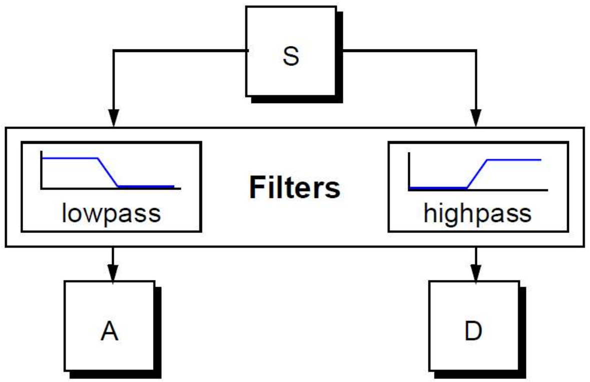
- It is for this reason that, in wavelet analysis, we often speak of **approximations** and **details**.

4. Discrete Wavelet Transform

• The **approximations** are the high-scale, low-frequency components of the signal.

• The **details** are the low-scale, high-frequency components.

• The filtering process, at its most basic level, looks like this:

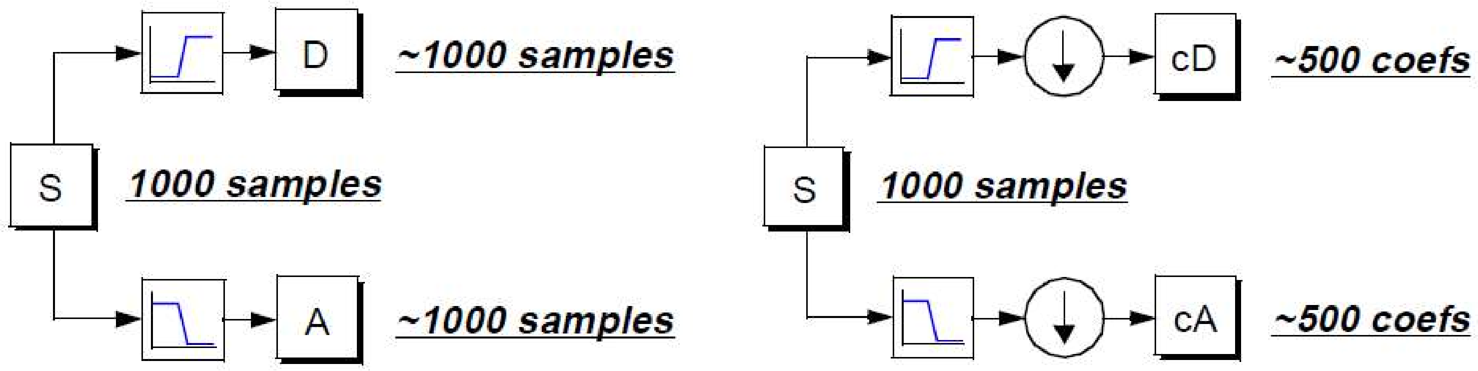


• If we actually perform this operation on a real digital signal, we wind up with **twice as much data** as we started with.

4. Discrete Wavelet Transform

- To correct this problem, we introduce the notion of **downsampling**.

- This simply means **throwing away** every second data point.

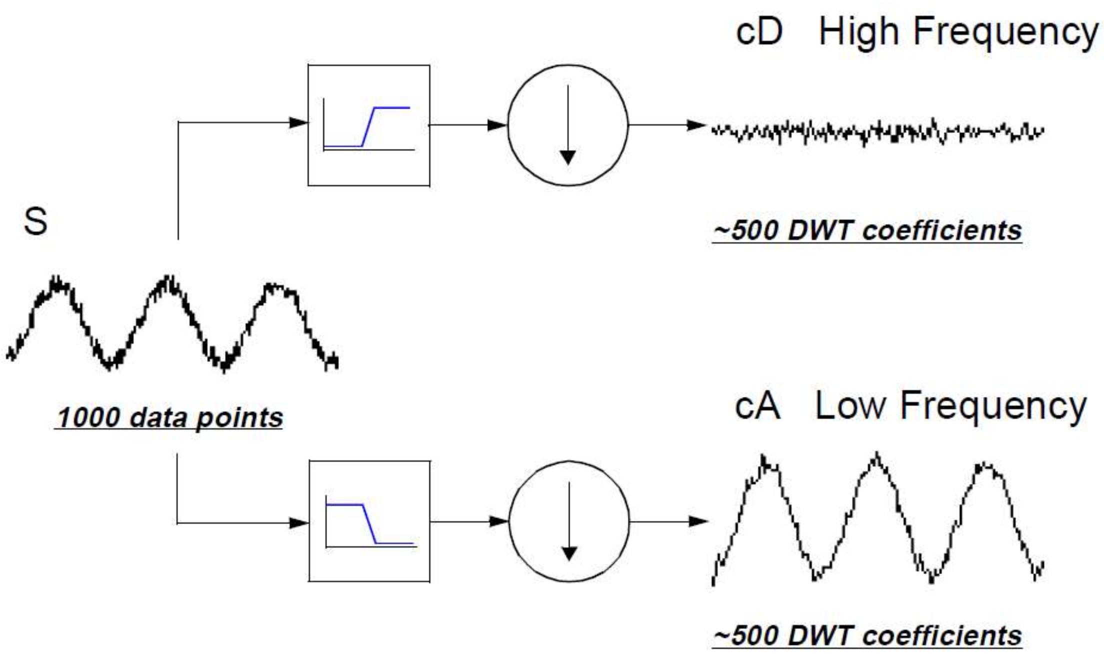


- The process on the right, which includes downsampling, produces **DWT coefficients**.

4. Discrete Wavelet Transform

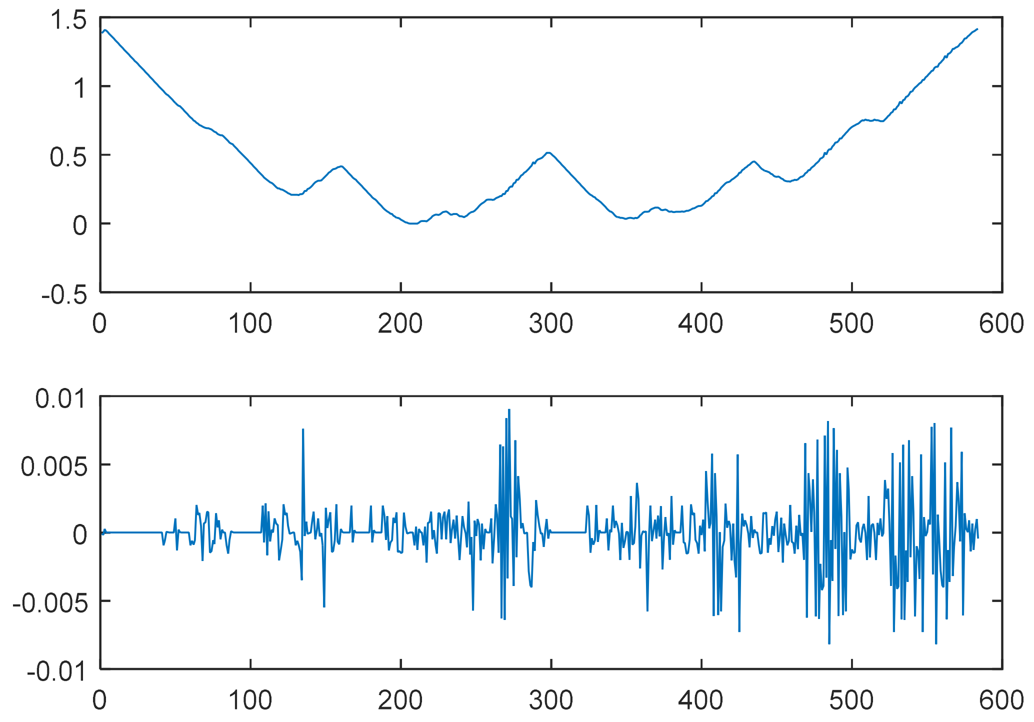
- To correct this problem, we introduce the notion of **downsampling**.

```
s = sin(20.*linspace(0,pi,1000)) + 0.5.*rand(1,1000);
[cA,cD] = dwt(s,'db2');
```



4. Discrete Wavelet Transform

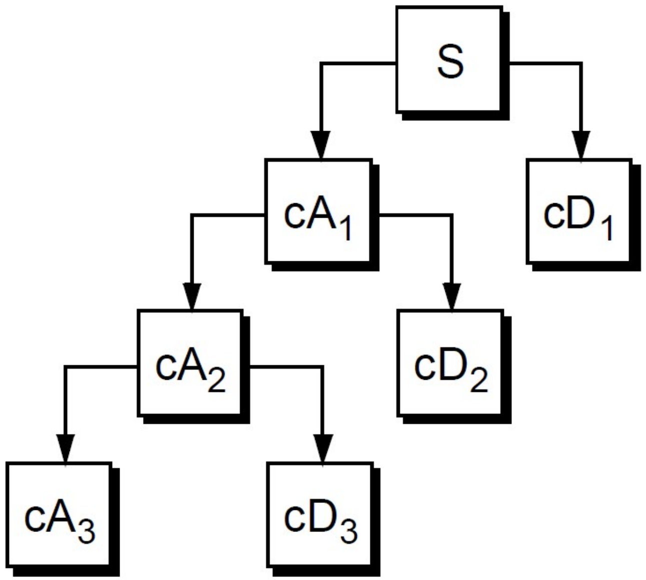
- MATLAB: s39ApproxDet.m



4. Discrete Wavelet Transform

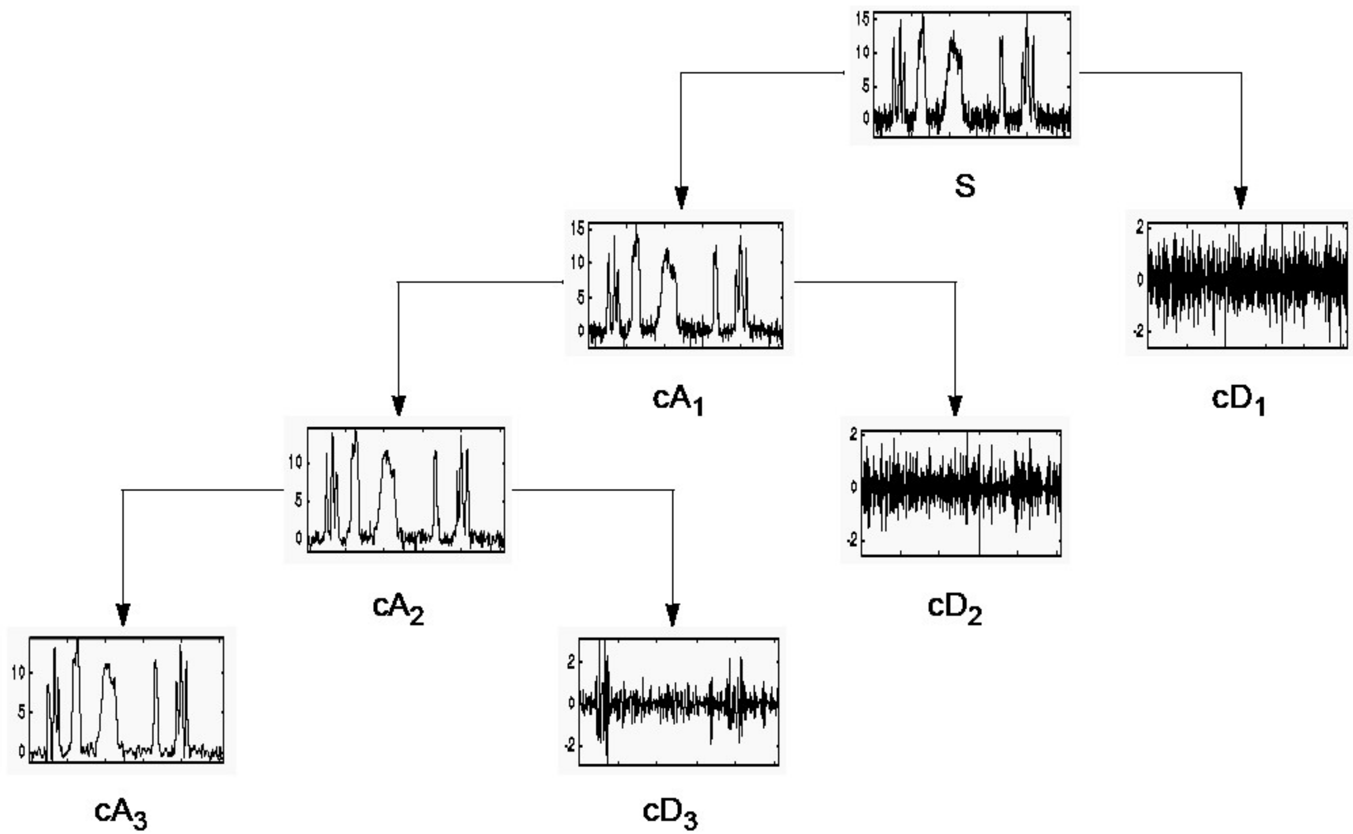
- The decomposition process can be iterated, with successive approximations being decomposed in turn, so that one signal is broken down into many lower-resolution components.

- This is called the wavelet decomposition tree.



4. Discrete Wavelet Transform

- The decomposition process can be iterated.



4. Discrete Wavelet Transform

- We've learned how the discrete wavelet transform can be used to analyze, or decompose, signals and images.

- The other half of the story is how those components can be assembled back into the original signal with no loss of information.

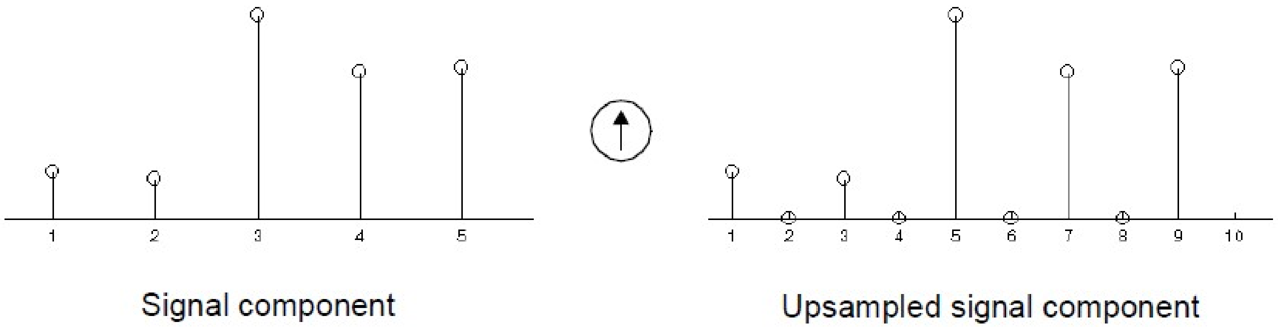
- This process is called reconstruction, or synthesis.

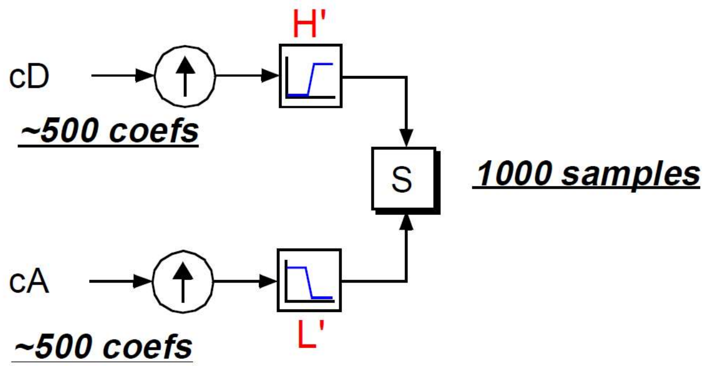
- The mathematical manipulation that effects synthesis is called the inverse discrete wavelet transform (IDWT).

4. Discrete Wavelet Transform

- Wavelet analysis involves filtering and downsampling.

- The wavelet reconstruction process consists of upsampling and filtering.



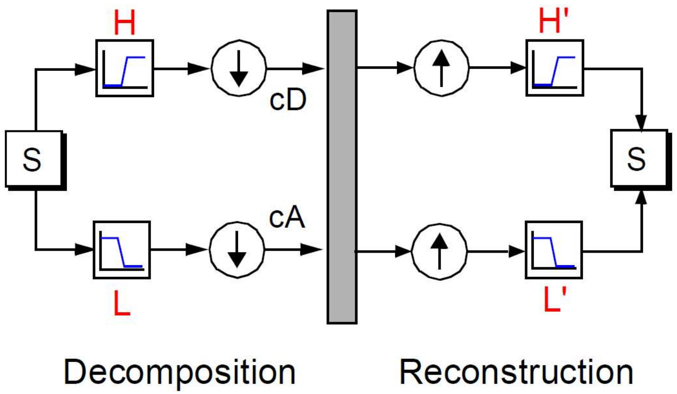


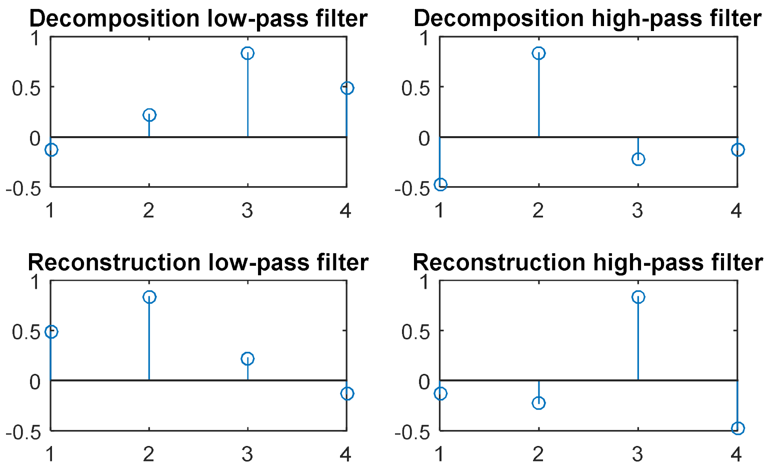
4. Discrete Wavelet Transform

- Wavelet analysis involves filtering and downsampling.

- By carefully choosing the decomposition and reconstruction filters we can “cancel out” the effects of aliasing caused by the subsampling.

- **Quadrature mirror filters**:



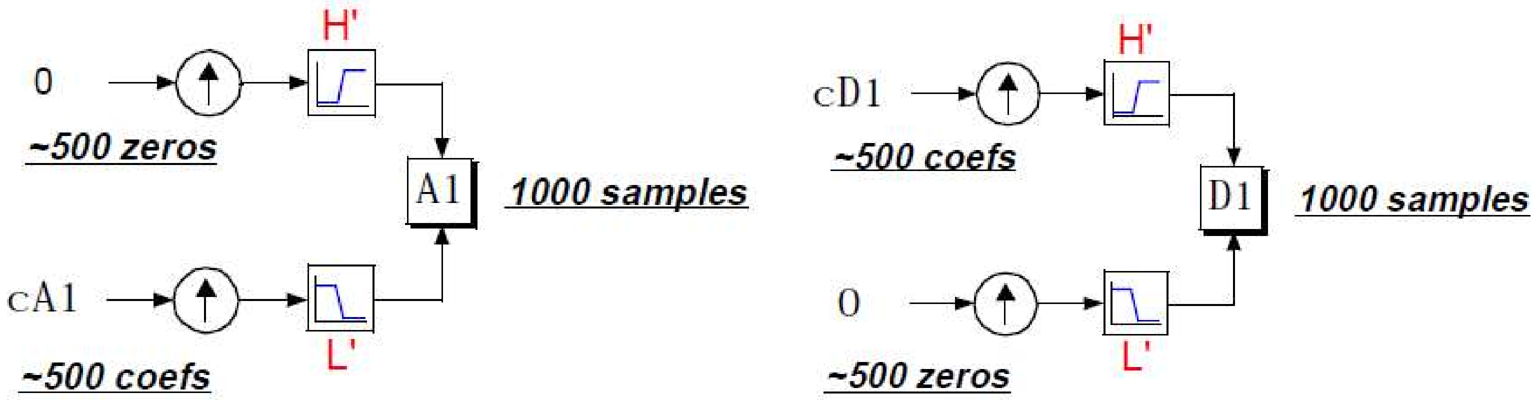


The four filters for db2

- MATLAB: s44Filters.m

4. Discrete Wavelet Transform

• It is also possible to reconstruct the approximation A and detail D themselves **from their coefficient vectors** cA and cD.

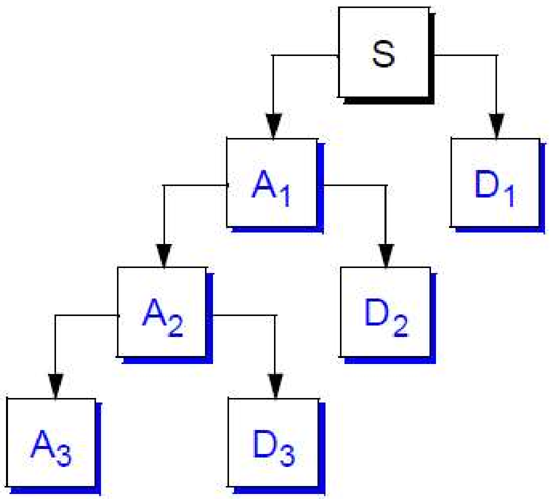


• And S = A1+ D1.

4. Discrete Wavelet Transform

- Extending this technique to a multi-level analysis.

Reconstructed Signal Components



$$S = A_1 + D_1$$
$$= A_2 + D_2 + D_1$$
$$= A_3 + D_3 + D_2 + D_1$$

4. Discrete Wavelet Transform

- The choice of filters not only determines whether perfect reconstruction is possible, it also determines the shape of the wavelet we use to perform the analysis.

- To construct a wavelet of some practical utility, you seldom start by drawing a waveform.

- Instead, it usually makes more sense to design the appropriate filters and then use them to create the waveform.

4. Discrete Wavelet Transform

- Example:

  - Starting from the reconstruction low-pass filter L'

```
[L, H, Lprime, Hprime] = wfilters('db2')

Lprime = [0.4830    0.8365    0.2241   -0.1294]
```

  - If we reverse the order Lmult and then multiply every second sample by –1, we obtain the highpass filter H’:

```
Hprime = [-0.1294   -0.2242   -0.8365   0.4830]
```

  - Next, upsample Hprime by two, inserting zeros in alternate positions:

```
HU = [-0.1294 0 -0.2241 0  0.8365 0 -0.4830 0]
```

4. Discrete Wavelet Transform

- Example:

  - Convolve the upsampled vector with the original lowpass filter:

```
H2 = conv(HU,Lprime);
```

  - If we iterate this process several more times, repeatedly upsampling and convolving the resultant vector with the four-element filter vector Lprime, a pattern begins to emerge.

  - Scale the final result by $(\sqrt{2})^i$, where *i* is the number of iterations.

- This result shows that the wavelet's shape is determined entirely by the coefficients of the reconstruction filters.

4. Discrete Wavelet Transform

• This relationship has profound implications.

• It means that you cannot choose just any shape, call it a wavelet, and perform an analysis.

• At least, you can’t choose an arbitrary wavelet waveform if you want to be able to reconstruct the original signal accurately.

• You are compelled to choose a shape determined by
•quadrature mirror decomposition filters.

4. Discrete Wavelet Transform

- The **mother wavelet** function ψ is determined by the highpass filter, which also produces the **details** of the wavelet decomposition.

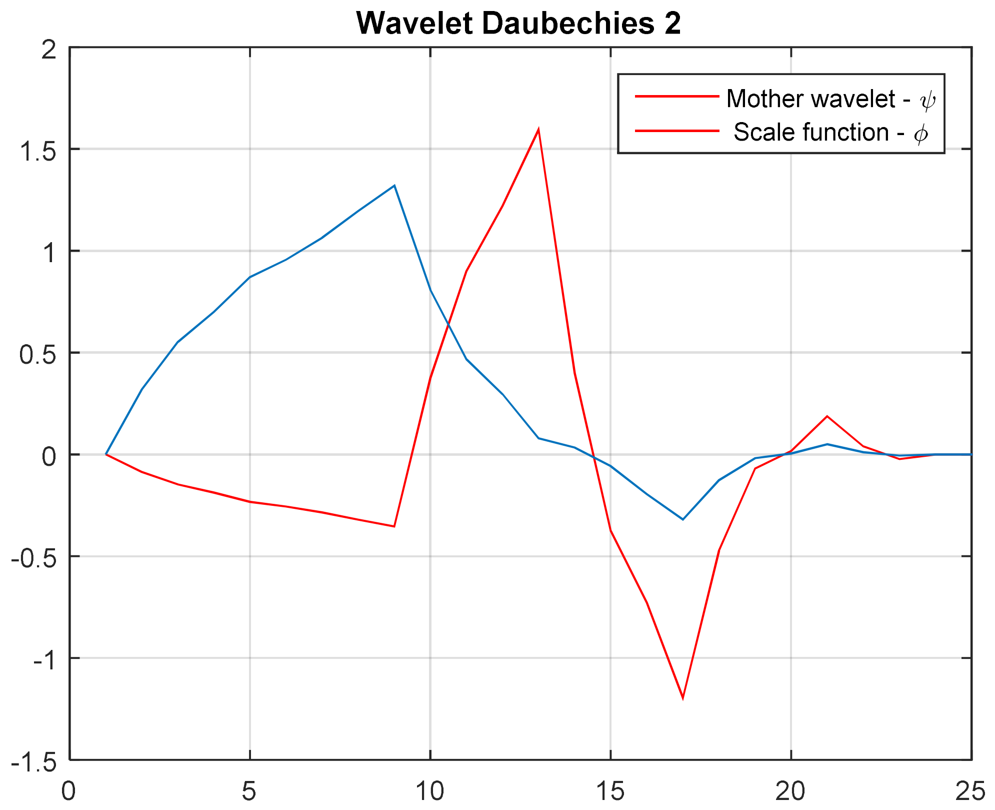
- There is an additional function associated with some but not all wavelets. This is the so-called **scaling** function, ϕ.

- It is determined by the lowpass quadrature mirror filters, and thus is associated with the **approximations** of the wavelet decomposition.

- Iteratively upsampling and convolving the lowpass filter produces a shape approximating the **scaling function**.

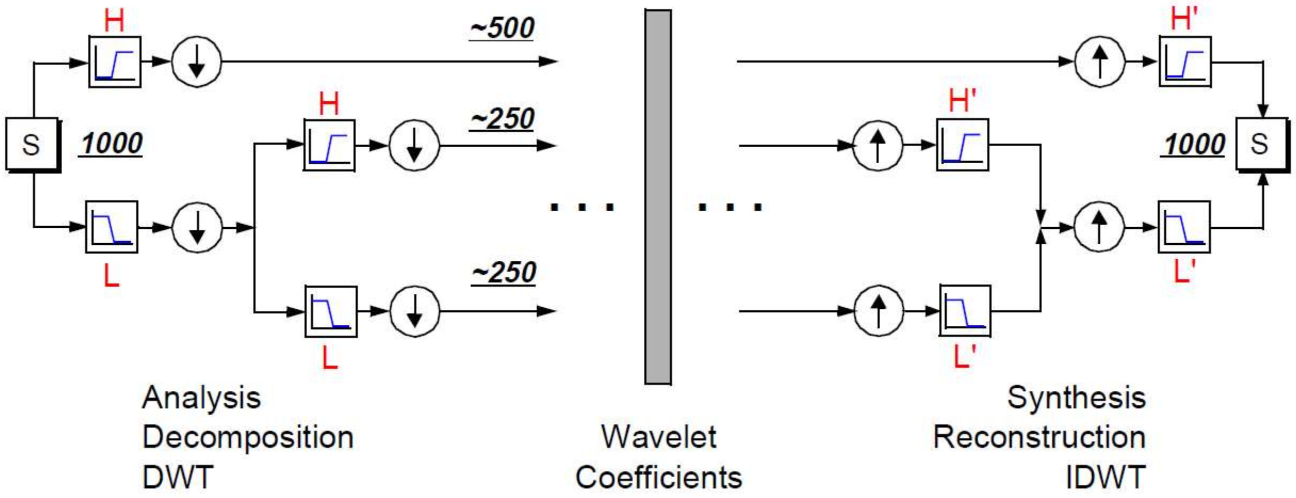
4. Discrete Wavelet Transform

- MATLAB: s53WaveFilters.m



4. Discrete Wavelet Transform

• Multistep analysis-synthesis:

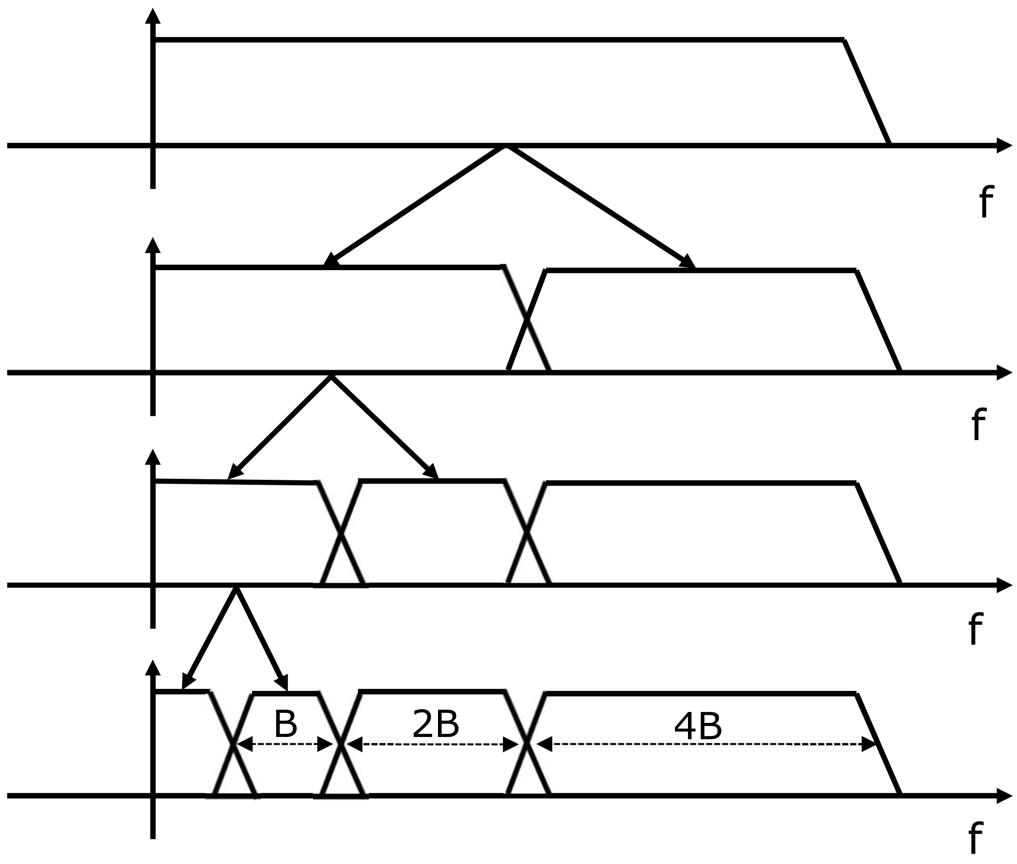


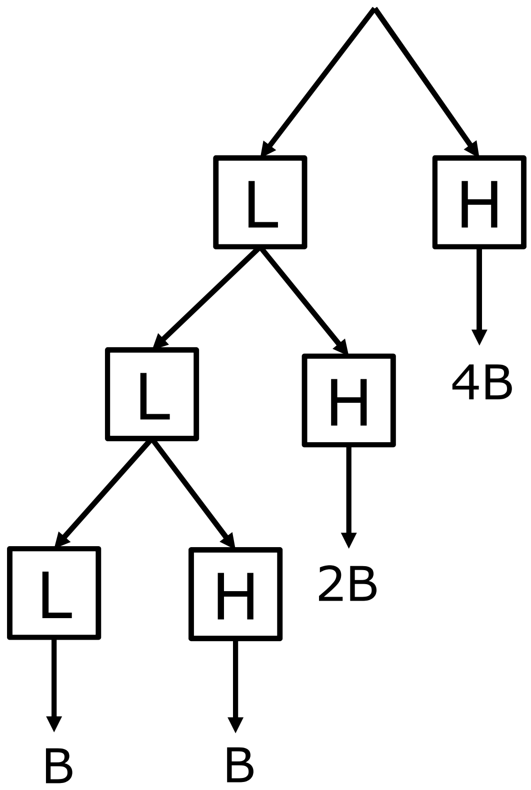
• Of course, there is no point breaking up a signal merely to have the satisfaction of immediately reconstructing it.

• We perform wavelet analysis because the coefficients thus obtained have many known uses, de-noising and compression being foremost among them.

4. Discrete Wavelet Transform

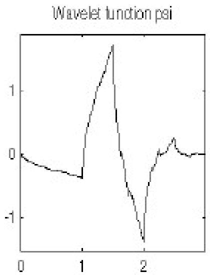
- Multistep analysis-synthesis:



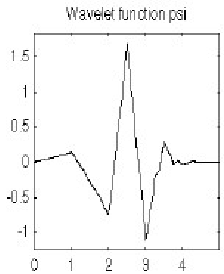


4. Discrete Wavelet Transform

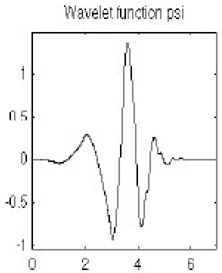
- Examples of wavelets: nine members of the Daubechies family.



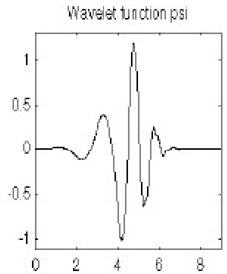
db2



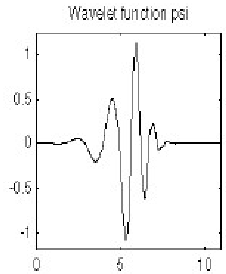
db3



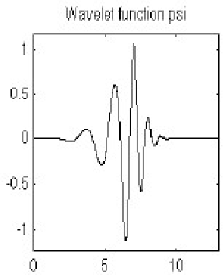
db4



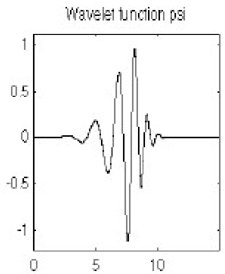
db5



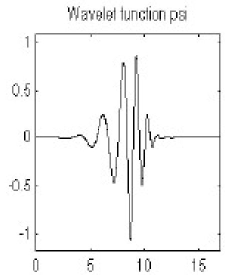
db6



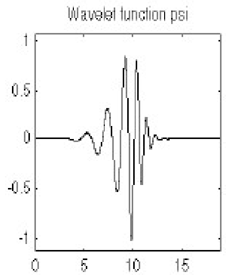
db7



db8



db9



db10

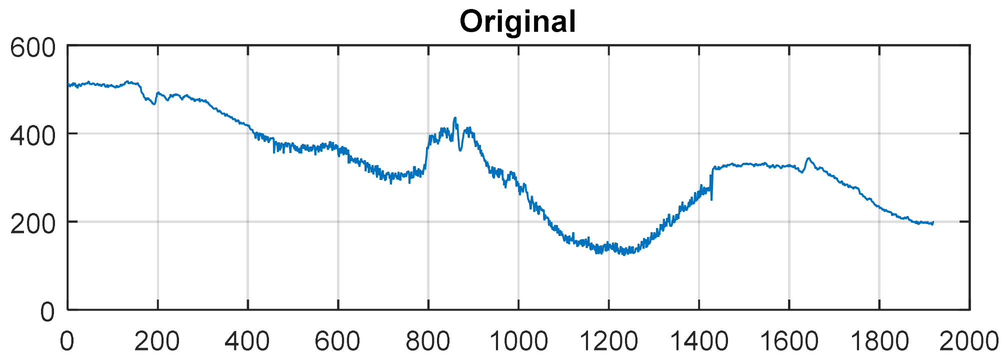
4. Discrete Wavelet Transform

- Example of One-Dimensional Analysis:

  - This example involves a real-world signal — electrical consumption measured over the course of three days.

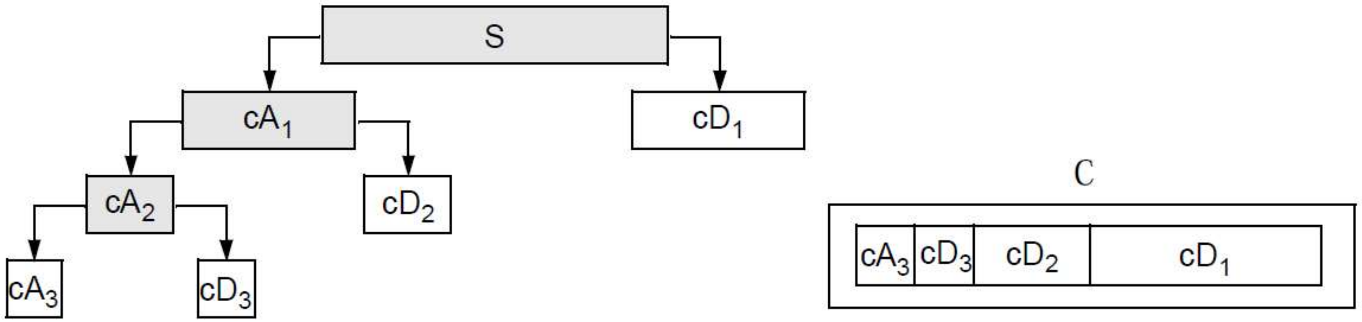
  - This signal is particularly interesting because of noise introduced when a defect developed in the monitoring equipment as the measurements were being made.

  - Wavelet analysis effectively removes the noise.



4. Discrete Wavelet Transform

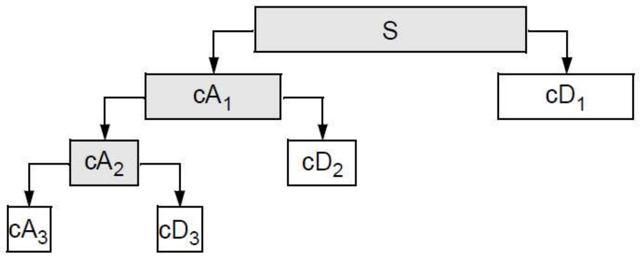
- Example of One-Dimensional Analysis:
  - Level 3 decomposition of the signal (again using the 'db3' wavelet.

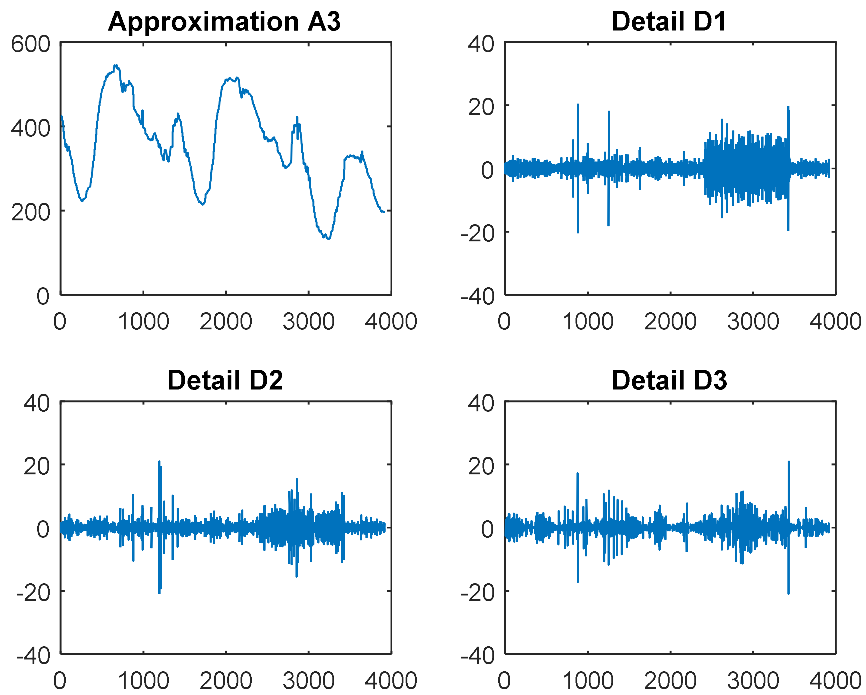


4. Discrete Wavelet Transform

- Example of One-Dimensional Analysis:

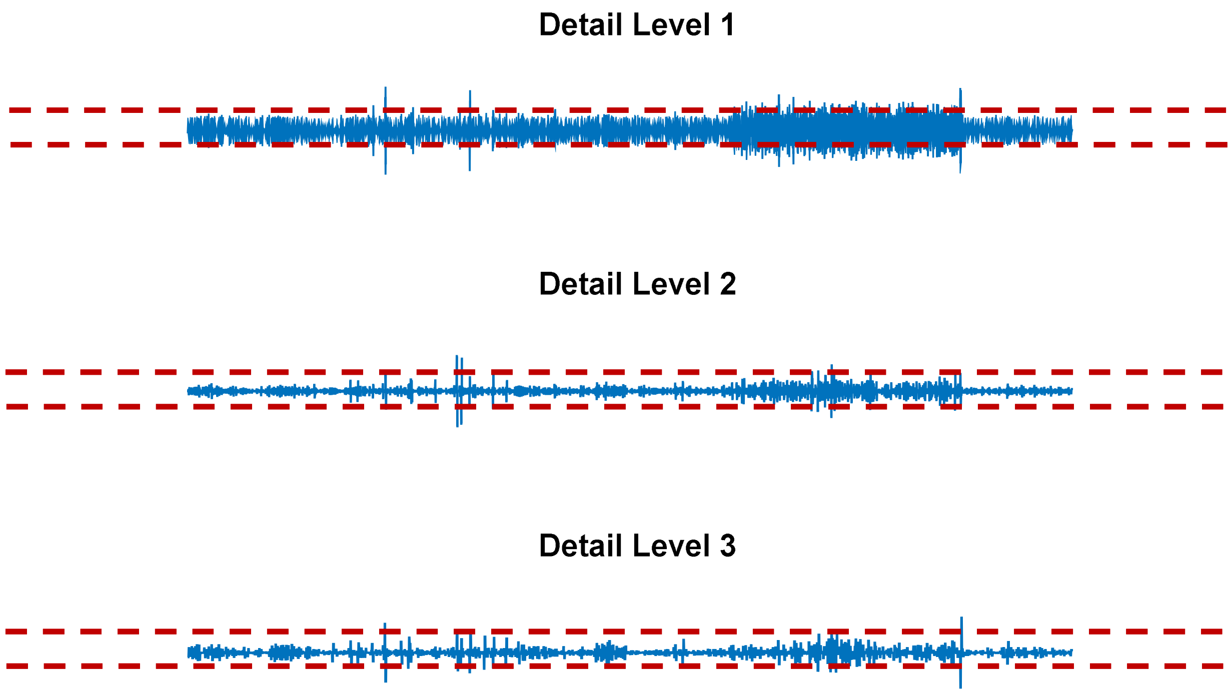
  ➢ Level 3 decomposition of the signal (again using the 'db3' wavelet.





4. Discrete Wavelet Transform

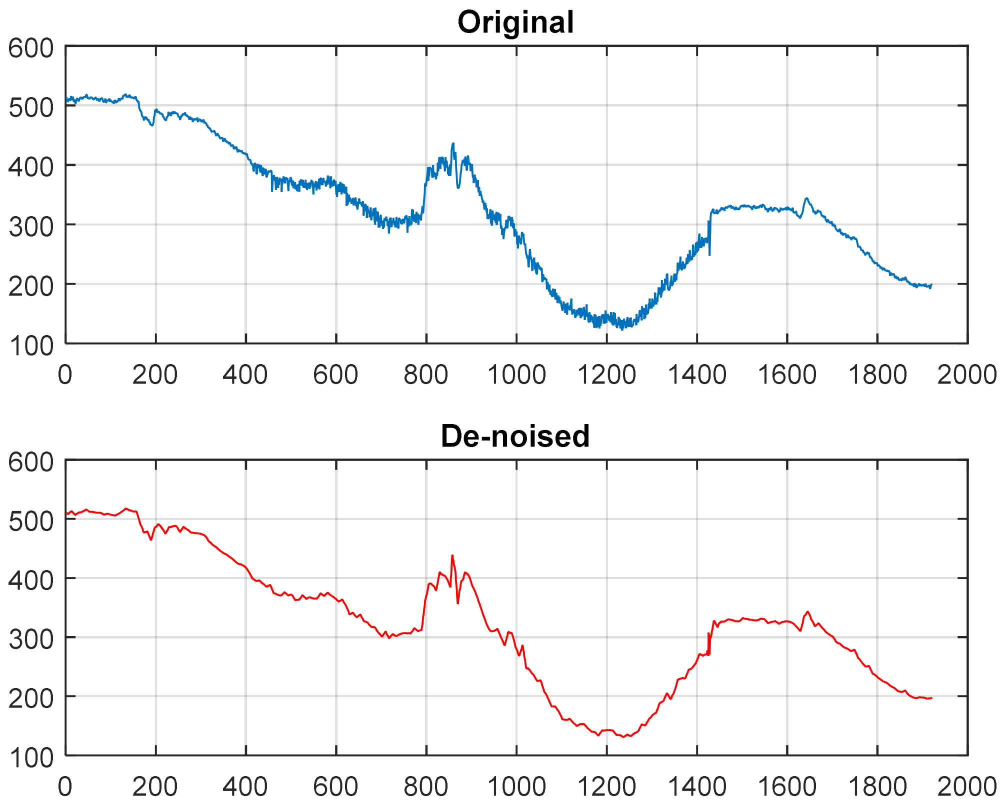
- Example of One-Dimensional Analysis:
  - ➢ Setting a threshold



4. Discrete Wavelet Transform

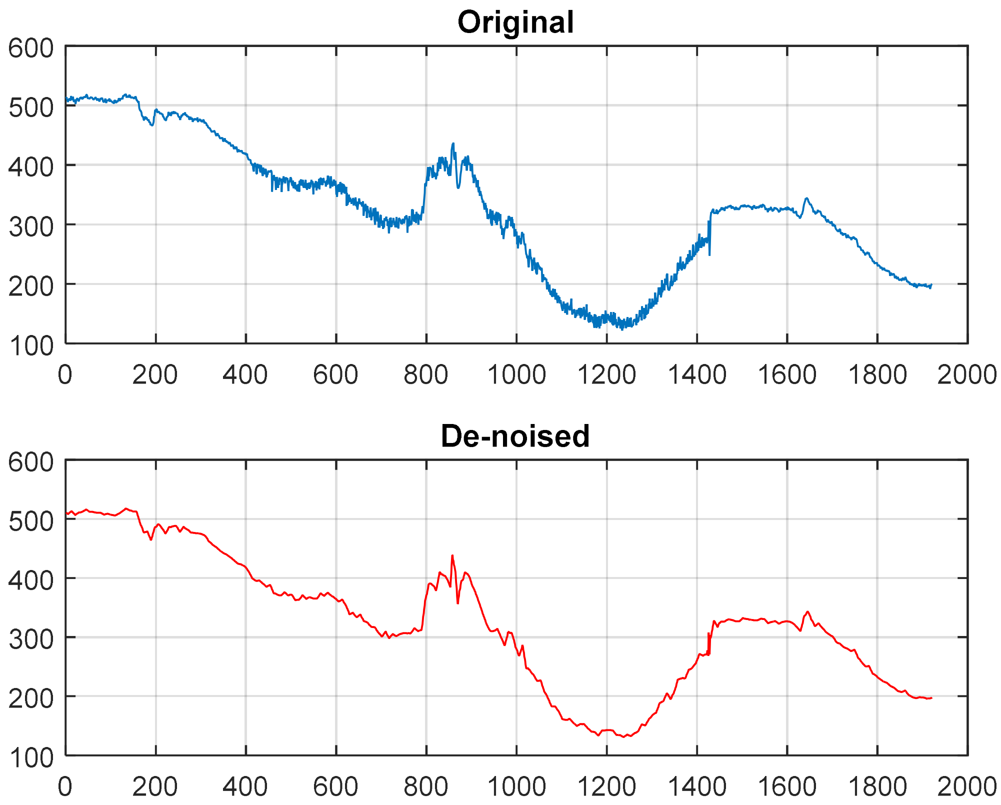
- Example of One-Dimensional Analysis:

    - Reconstruction



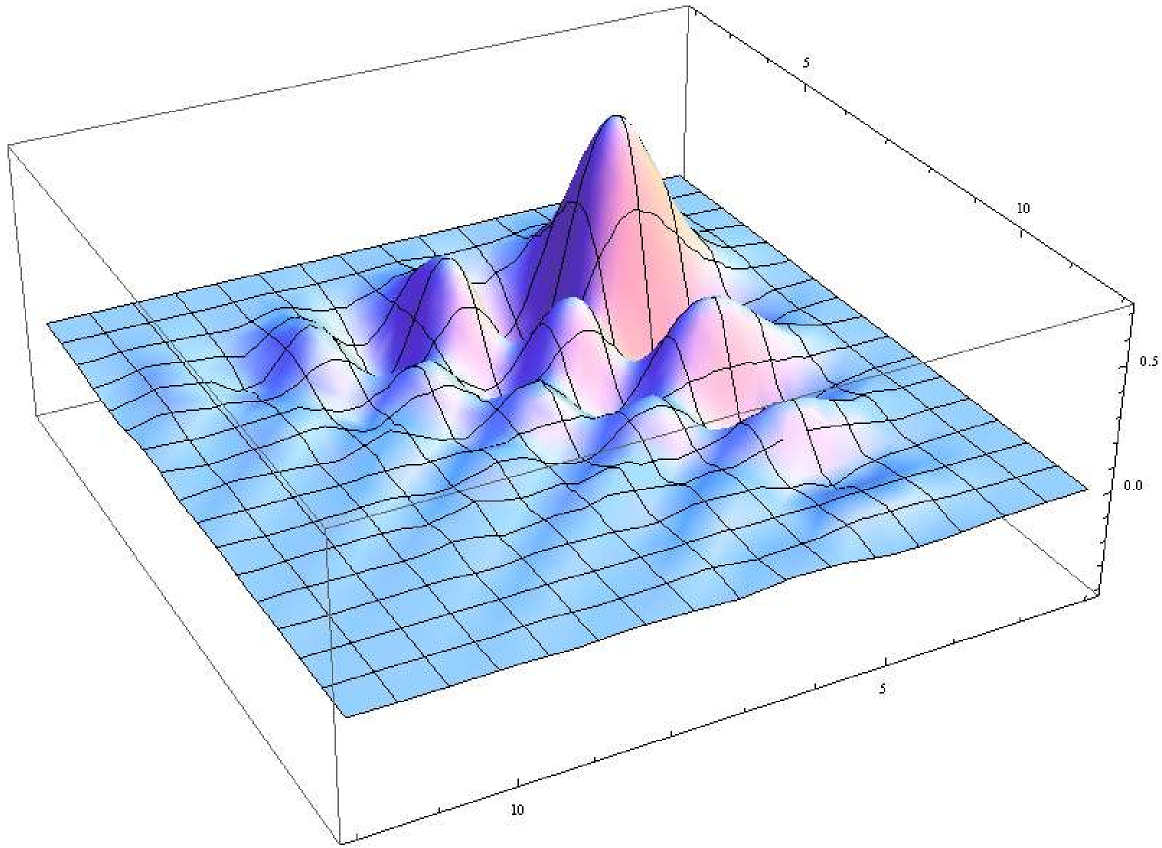
4. Discrete Wavelet Transform

- MATLAB: s61Denoise1D.m



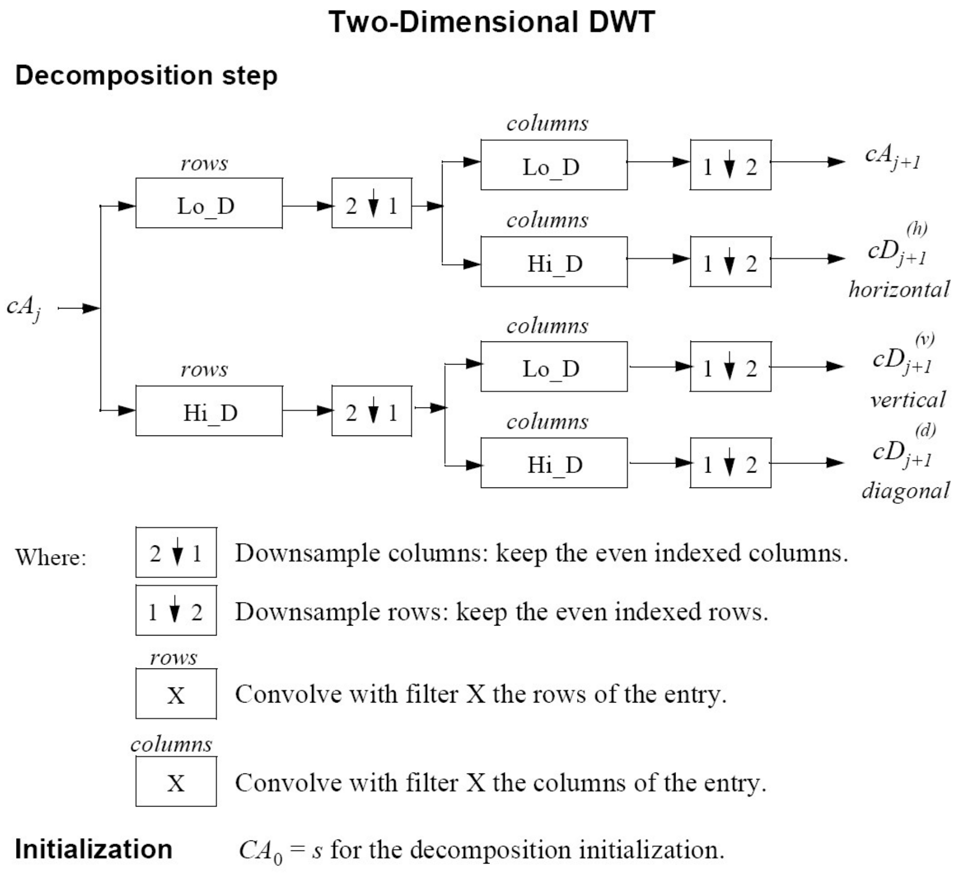
4. Discrete Wavelet Transform

- Two-Dimensional Analysis



4. Discrete Wavelet Transform

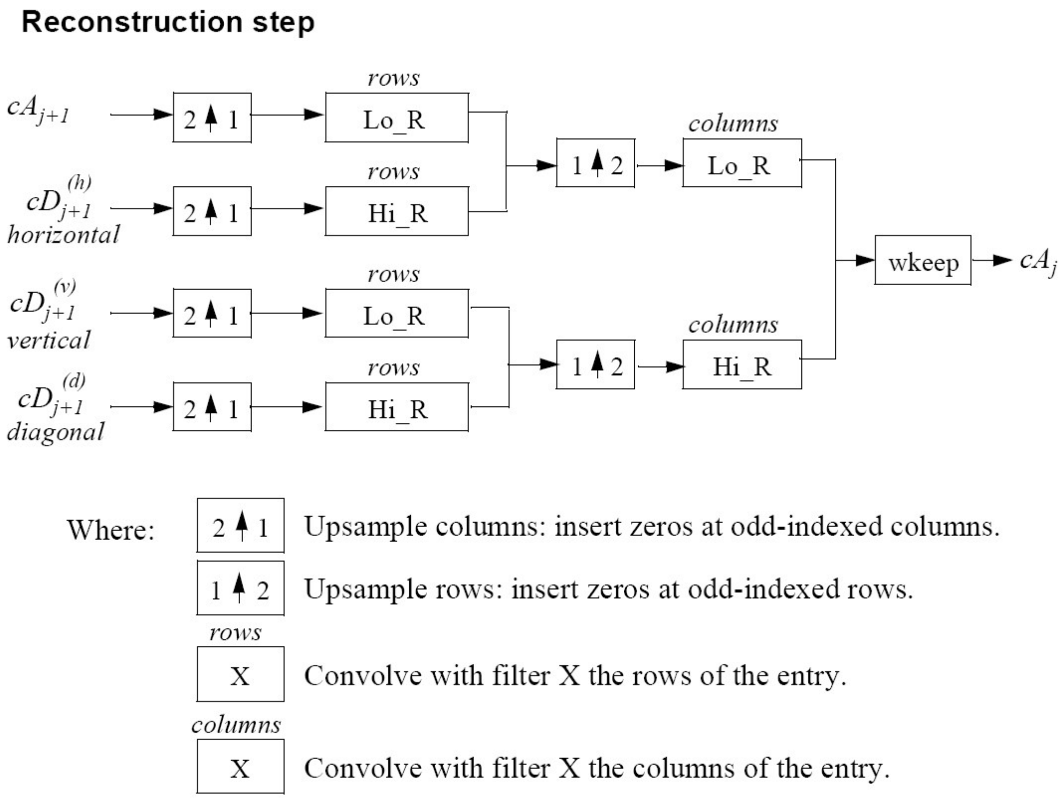
- Two-Dimensional Analysis



4. Discrete Wavelet Transform

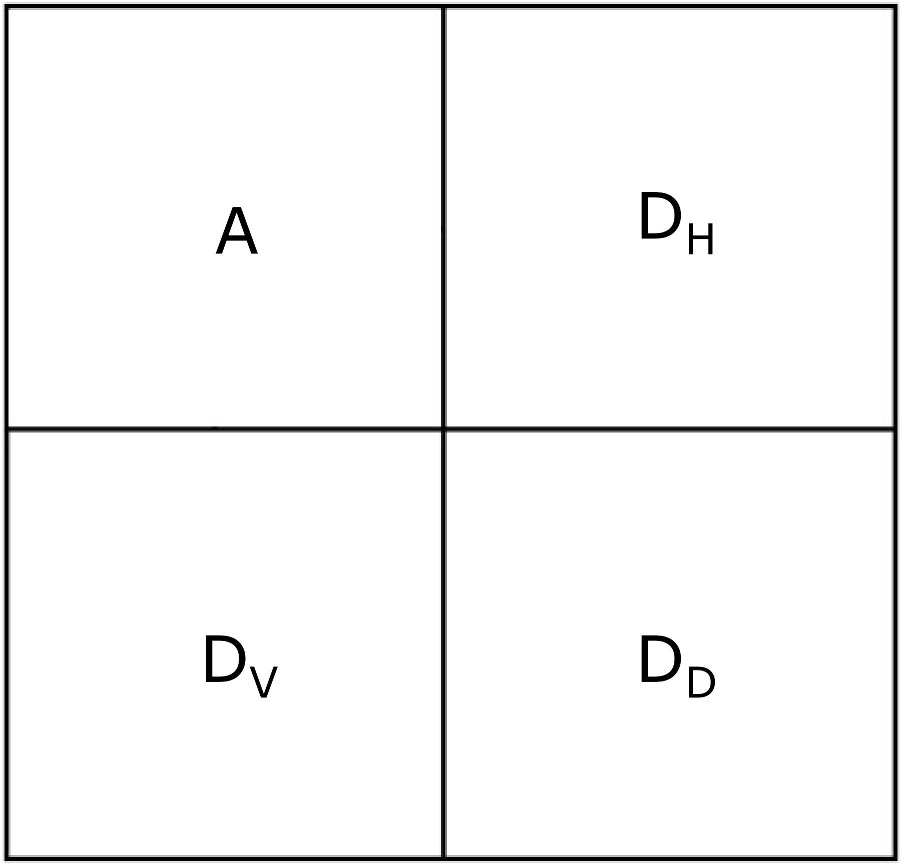
- Two-Dimensional Analysis

Two-Dimensional IDWT



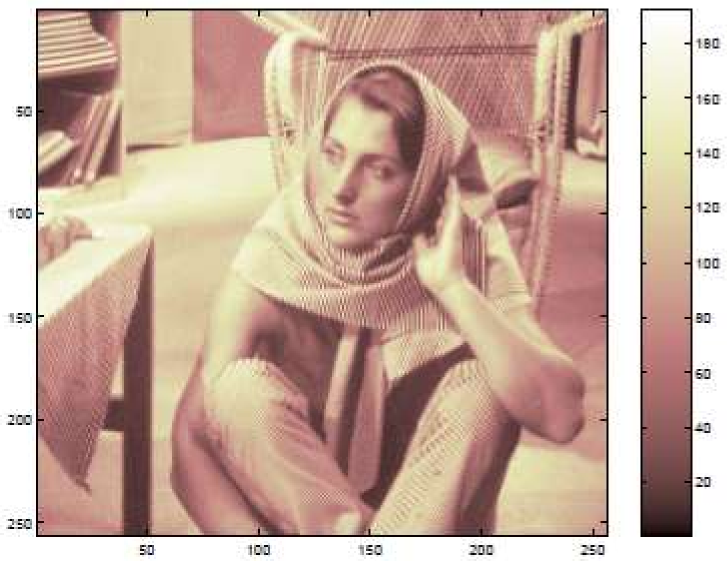
4. Discrete Wavelet Transform

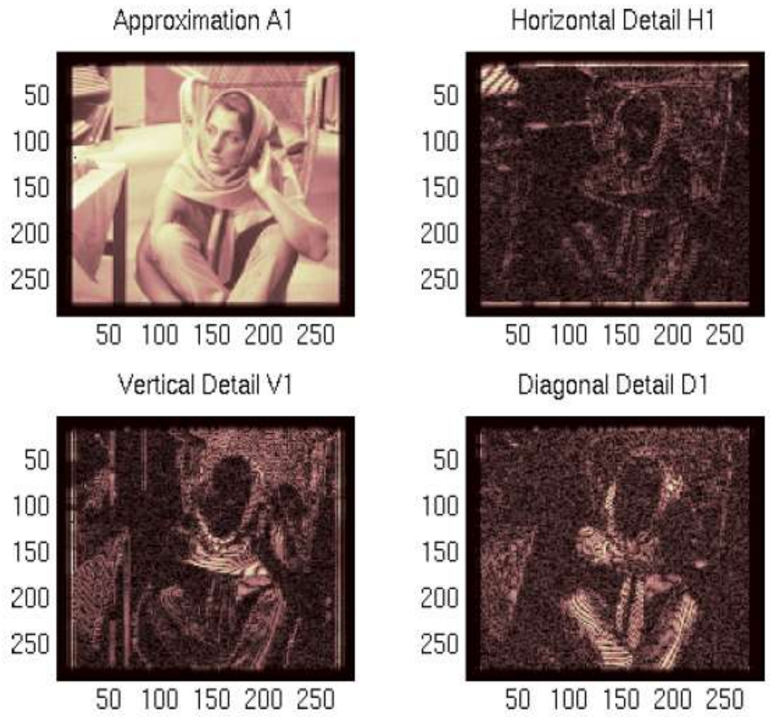
- Two-Dimensional Analysis (one-step decomposition)



4. Discrete Wavelet Transform

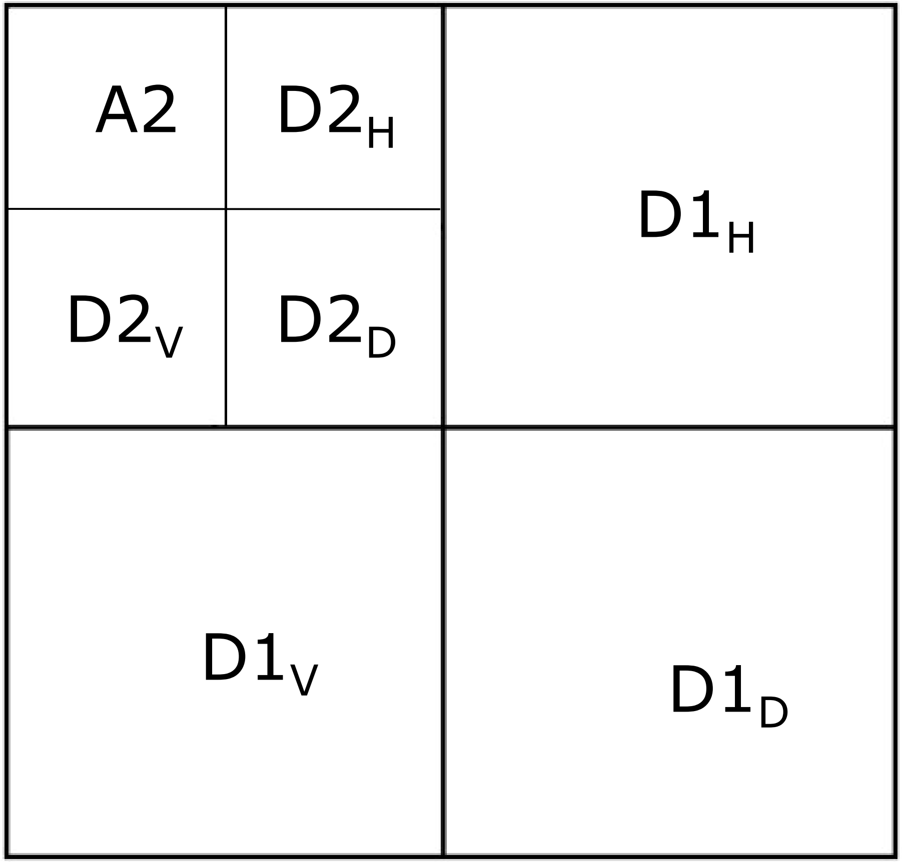
- Two-Dimensional Analysis (one-step decomposition)





4. Discrete Wavelet Transform

- Two-Dimensional Analysis (multiple-level decomposition)

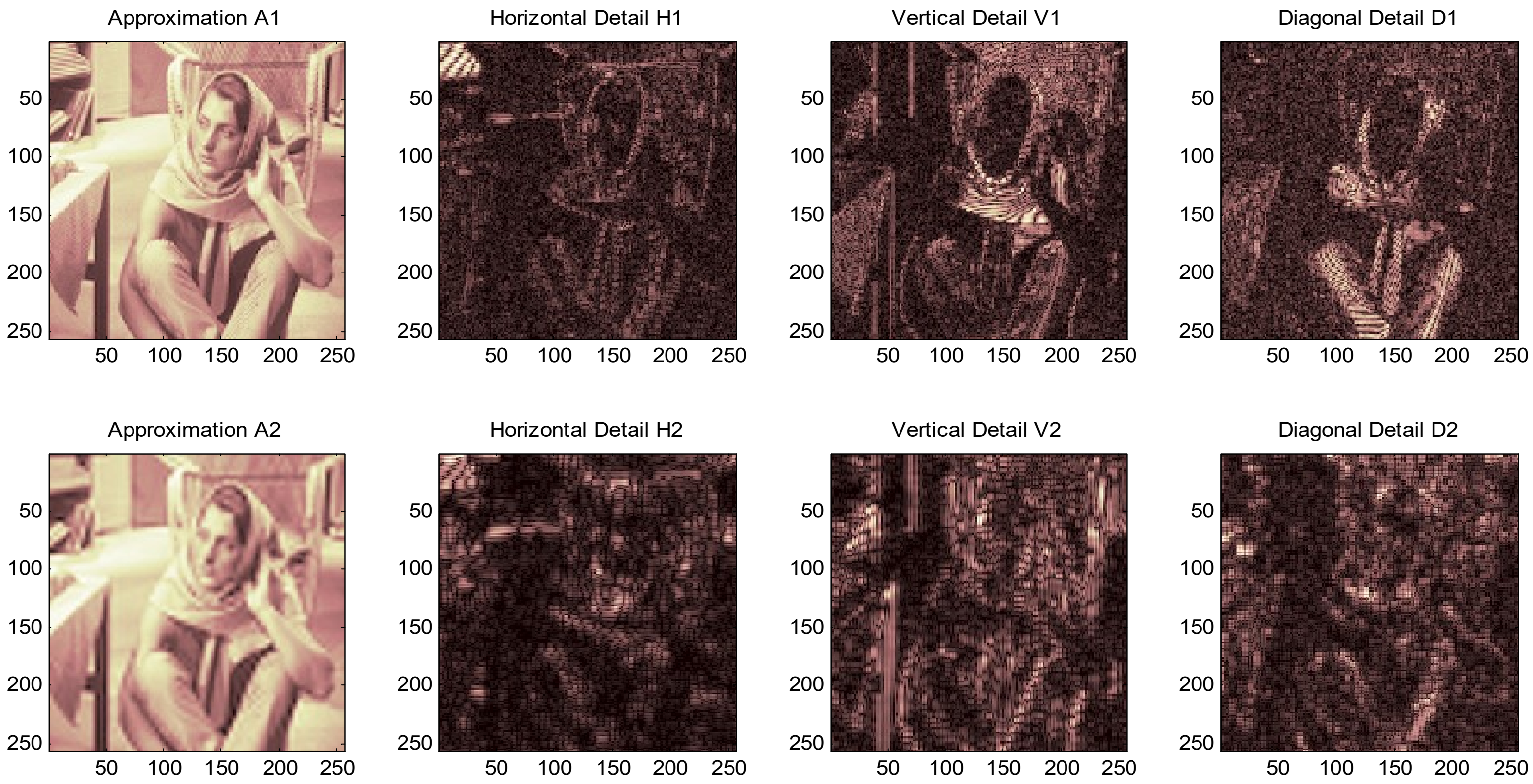


4. Discrete Wavelet Transform

- Two-Dimensional Analysis (multiple-level decomposition)

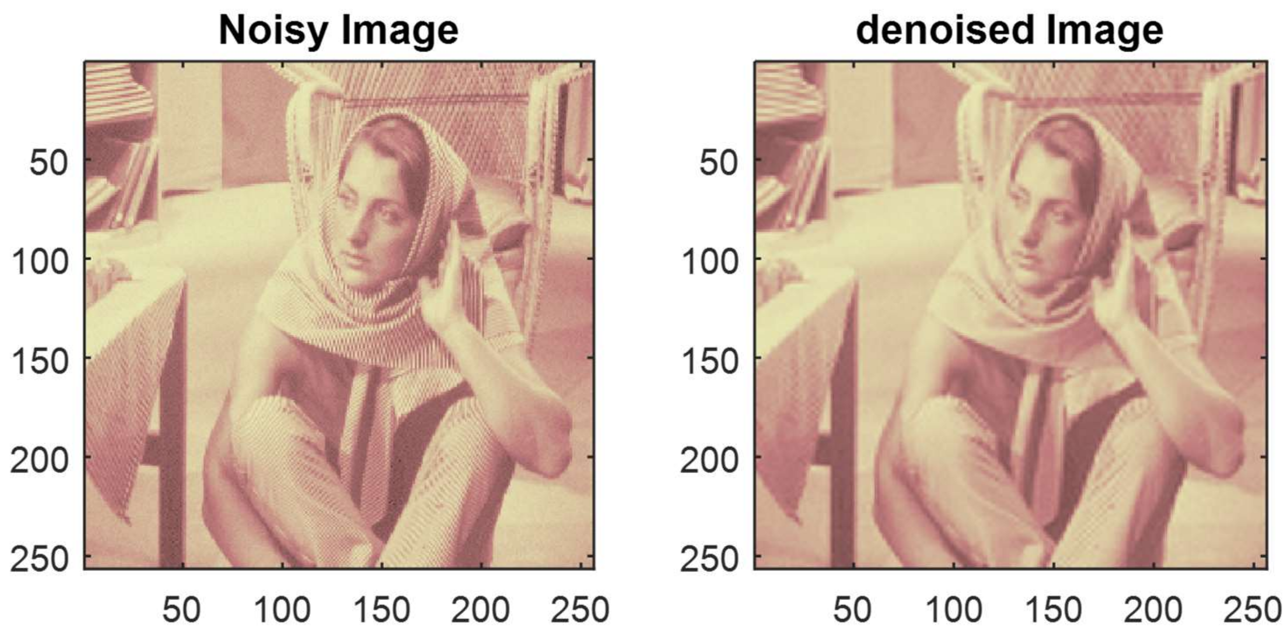
4. Discrete Wavelet Transform

- Example of Two-Dimensional Analysis (Extension to Image Denoising, Two-step Decomposition)



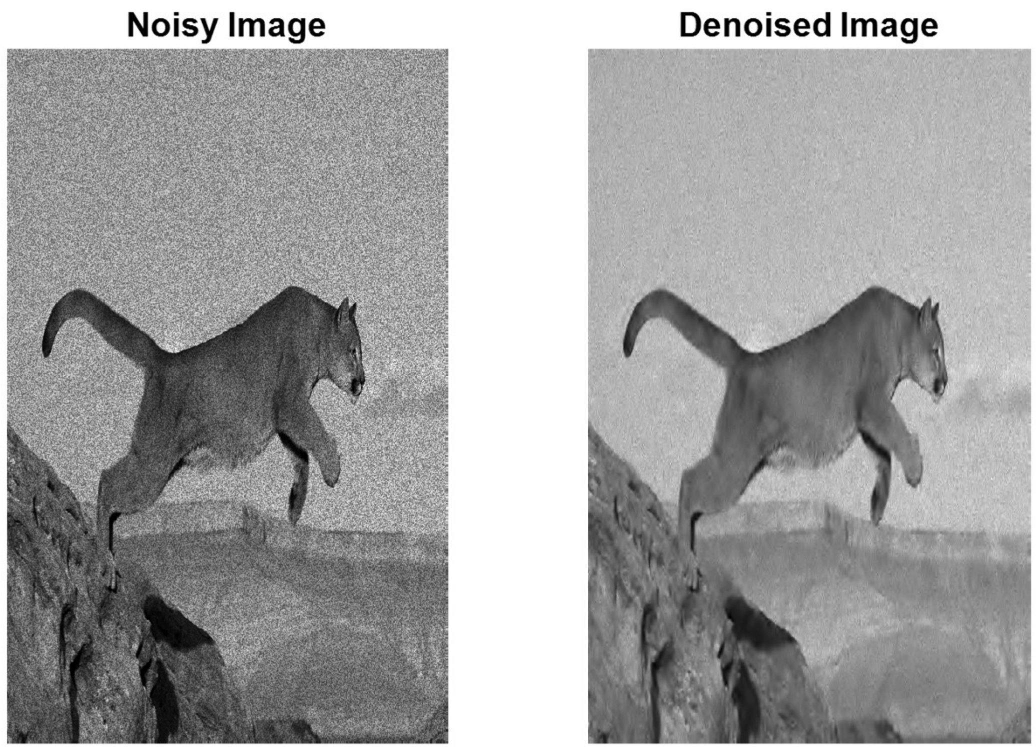
4. Discrete Wavelet Transform

• Example of Two-Dimensional Analysis (Extension to Image Denoising, Two-step Decomposition): s72Denoise2Da.m



4. Discrete Wavelet Transform

• Example of Two-Dimensional Analysis (Extension to Image Denoising, Two-step Decomposition): s73Denoise2Db.m



4. Discrete Wavelet Transform

• Example of Two-Dimensional Analysis (Extension to Image Denoising, Two-step Decomposition): s74Denoise2Dc.m