Discrete Wavelet Transforms$^{\odot}$

Industrial-Strength, Technology-Enabling Computing (look, listen, read)

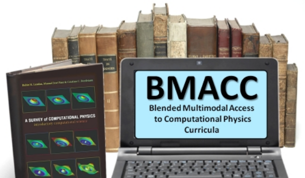
Rubin H Landau

Sally Haerer, Producer-Director

Based on *A Survey of Computational Physics* by Landau, Páez, & Bordeianu

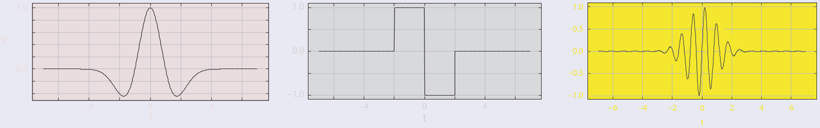
with Support from the National Science Foundation

Course: **Computational Physics II**



Review: Wavelets in a Nutshell

Three Wavelet Examples



- Wavelets = packets
- Nonstationary signals
- Basis functions
- All oscillate
- Varied functional forms
- Vary scale & center
- Finite $\Delta\ \tau\ \Delta\ \omega$
- $\Delta\tau\ \Delta\omega \geq 2\pi$

Problem: Determine $\leq N$ Indep Wavelet TFs $Y_{i,j}$

The Discrete Wavelet Transform (DWT)

$$Y(s,\tau) = \int_{-\infty}^{+\infty} dt\, \psi^*_{s,\tau}(t)\, y(t) \qquad \text{(Wavelet Transform)}$$

- Given: N signal measurements:

$$y(t_m) \equiv y_m, \quad m = 1, \ldots, N$$

- Compute no more DWTs than needed
- ***Hint:*** Lossless: consistent with uncertainty principle
- ***Hint:*** Lossy: consistent with required resolution

How to Discretize DWT?

Auto Scalings, Translations = ♡ Wavelets

Discrete scaling s, discrete time translation τ:

$$s = 2^j, \quad \tau = \frac{k}{2^j}, \quad k, \ j = 0, 1, \ldots \quad \text{(Dyadic Grid)} \tag{1}$$

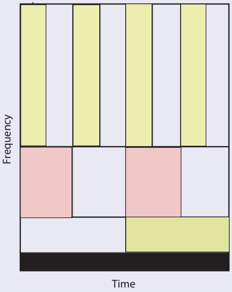
$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}} \Psi \left[\frac{t - k2^j}{2^j} \right] \qquad \text{(Wavelets T = 1)} \tag{2}$$

$$Y_{j,k} = \int_{-\infty}^{+\infty} dt \, \psi_{j,k}(t) \, y(t) \tag{3}$$

$$\simeq \sum_m \psi_{j,k}(t_m) y(t_m) h \quad \text{(DWT)} \tag{4}$$

Time & Frequency Sampling

Sample $y(t)$ in Time & Frequency Ranges

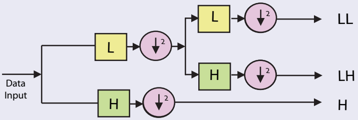


- High ω $\uparrow$ range
- High ω for details
- Few low ω for shape
- Each t, Δ scales

- Uncertainty Prin: $\Delta\omega\,\Delta t \geq 2\pi$
- Don't be wasteful!
- $\Rightarrow$ H $\times$ W = Const

Multi Resolution Analysis (MRA)

Digital Wavelet Transform $\equiv$ Filter



- Filter: Δ relative ω strengths $\equiv$ analyze Δ scale: MRA
- Sample $\longrightarrow$ Filter $\longrightarrow$ Sample $\cdots$

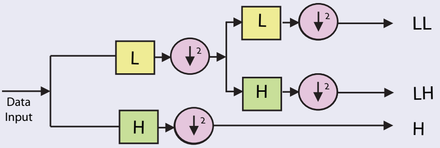
$$g(t) = \int_{-\infty}^{+\infty} d\tau \; h(t-\tau)\, y(\tau) \qquad \text{(Filter)}$$

$$Y(s,\tau) = \int_{-\infty}^{+\infty} dt \; \Psi^{*}\left(\frac{t-\tau}{s}\right) y(t) \simeq \sum w_i \psi_i y(t_i) \quad \text{(Transform)}$$

- w_i = integration weight + wavelet values = “filter coeff”

MRA via Filter Tree (Pyramid Algorithm)

Filtering with Decimation



- H: highpass filters
- L: lowpass filters
- Ea filter: lowers scale
- $\downarrow$2: rm 1/2 signal

- Factor-of-2 “decimation”
- "Subsampling"
- Keeps area constant
- Need little large-s info

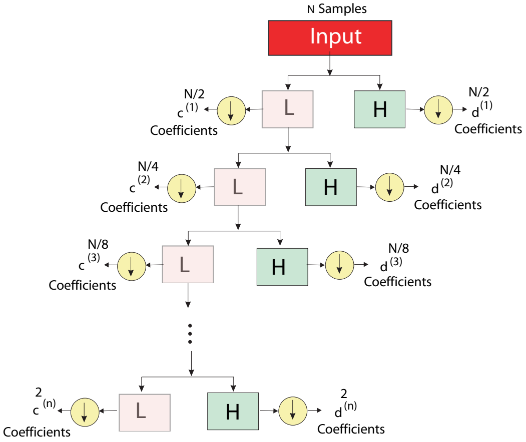
Example from Appendix

High $\longrightarrow$ Medium $\longrightarrow$ Low Resolution

Summary

- Pyramid DWT algorithm compresses data, separates hi res
- Smooth info in low-ω (large s) components
- Detailed info in high-ω (small s) components
- High-res reproduction: more info on details than shape
- Different resolution components = independent
- $\Rightarrow$ Lower data storage
- Rapid reproduction/inversion (JPEG2)

Pyramid Algorithm Graphically (see text)



- *L* & *H* via matrix mult (TFs)
- Decimated *H* output saved
- Downsample: ↓ #, Δ scale
- Ends with 2 *H*, *L* points

$N = 8$ Example Matrices

$$
\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{pmatrix}
\xrightarrow{\text{filter}}
\begin{pmatrix} s_1^{(1)} \\ d_1^{(1)} \\ s_2^{(1)} \\ d_2^{(1)} \\ s_3^{(1)} \\ d_3^{(1)} \\ s_4^{(1)} \\ d_4^{(1)} \end{pmatrix}
\xrightarrow{\text{order}}
\begin{pmatrix} s_1^{(1)} \\ s_2^{(1)} \\ s_3^{(1)} \\ s_4^{(1)} \\ \hline d_1^{(1)} \\ d_2^{(1)} \\ d_3^{(1)} \\ d_4^{(1)} \end{pmatrix}
\xrightarrow{\text{filter}}
\begin{pmatrix} s_1^{(2)} \\ d_1^{(2)} \\ s_2^{(2)} \\ d_2^{(2)} \\ \hline d_1^{(1)} \\ d_2^{(1)} \\ d_3^{(1)} \\ d_4^{(1)} \end{pmatrix}
\xrightarrow{\text{order}}
\begin{pmatrix} s_1^{(2)} \\ s_2^{(2)} \\ d_1^{(2)} \\ d_2^{(2)} \\ \hline d_1^{(1)} \\ d_2^{(1)} \\ d_3^{(1)} \\ d_4^{(1)} \end{pmatrix}
$$

Pyramid Algorithm Matrices

Pyramid Algorithm Successive Operations

1. Mult N-D vector of Y by c matrix
2. (See text for c_i derivation)

$$\begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_3 & -c_2 & c_1 & -c_0 \\ c_2 & c_3 & c_0 & c_1 \\ c_1 & -c_0 & c_3 & -c_2 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

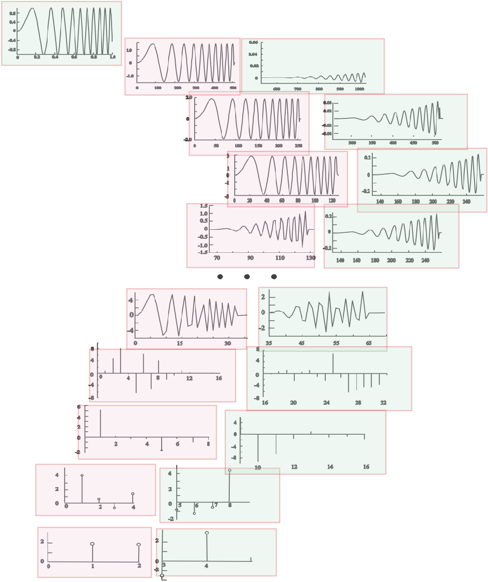
3. Mult $(N/2)$-D smooth vector by c matrix
4. Reorder: new 2 smooth on top, new detailed, older detailed
5. Repeat until only 2 smooth remain

Inversion $Y \rightarrow y$

Using transpose (inverse) of transfer matrix at each stage

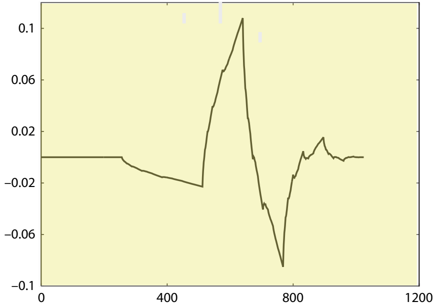
$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} c_0 & c_3 & c_2 & c_1 \\ c_1 & -c_2 & c_3 & -c_0 \\ c_2 & c_1 & c_0 & c_3 \\ c_3 & -c_0 & c_1 & -c_2 \end{pmatrix} \begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}.$$

Chirp Example Graphical



- $1024 \sin(60t^2)$
- 1024 thru H & L
- Downsample
- $\rightarrow 512\ L,\ 512\ H$
- Save details
- Each step $\downarrow\ 2\times$
- Connected dots
- End: $2 \downarrow$ detail

Daubechies Daub4 Wavelet (Derivation in Text)



$$c_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \qquad c_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}$$

$$c_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \qquad c_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$$

Summary: Wavelet Transforms

Continuous → Discrete → Pyramid Algorithm

$$Y(s,\tau) = \int_{-\infty}^{+\infty} dt\ \psi^*_{s,\tau}(t)\ y(t)$$

- Discrete: measurements, $\int \rightarrow \sum_i$
- Transform → digital filter → coefficients
- Multiple scales → series H & L filters
- Compression: N independent components
- Further compression: Variable resolution