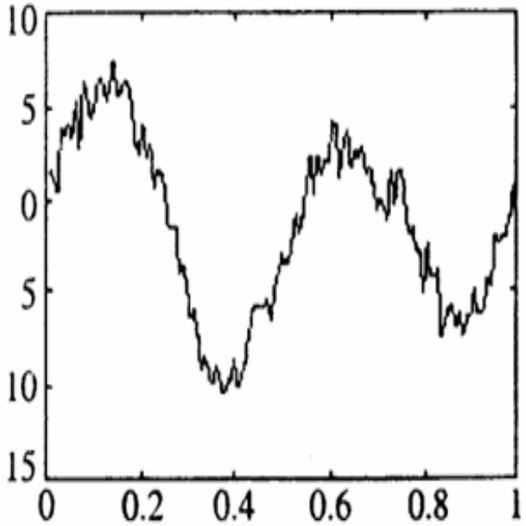
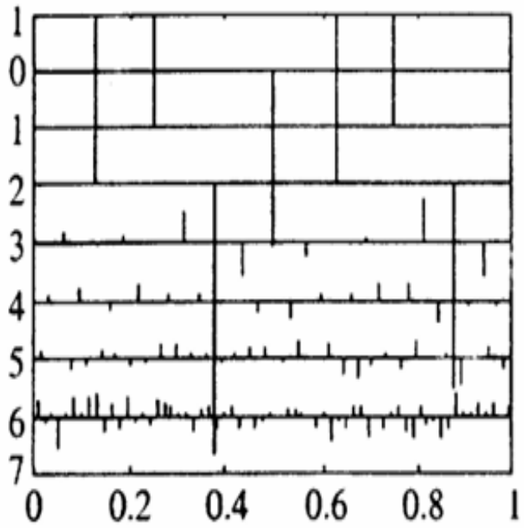
WAVELET DENOISING : EXTRACTING SIGNAL FROM NOISE

Wavelet denoising is a powerful technique for removing noise from signals. This presentation will explore in detail about the process of denoising using wavelet transform as well as algorithms and other estimation methods associated with it.

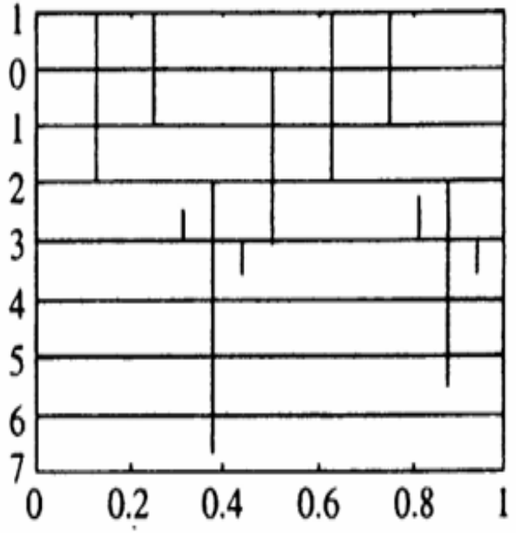
ABBURU SAHASRANSHU – CB.EN.U4CCE21001
DHIVYADARSHAN G – CB.EN.U4CCE21016
HARI KRISHNA N – CB.EN.U4CCE21036



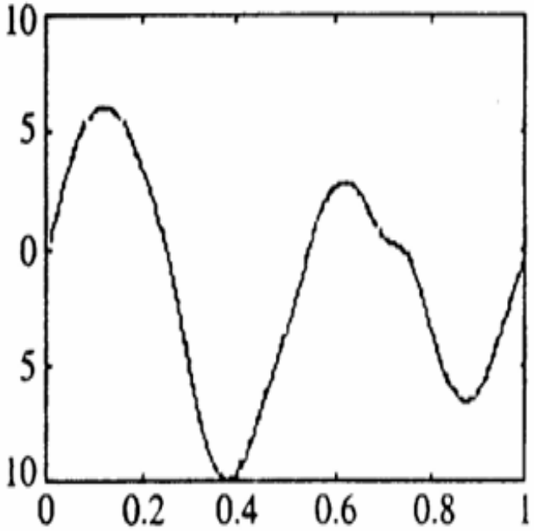
(a) Noisy signal



(b) Wavelet transform coefficients



(c) Thresholded coefficients



(d) Denoised signal

INTRODUCTION TO WAVELET DENOISING

Wavelet Denoising

The discrete wavelet transform (DWT) is applied to a signal to identify its wavelet coefficients.

A threshold is introduced to manipulate these coefficients, which reduces noise in the signal while preserving its essential structure.

The process involves the following steps:

1. Decomposition: Compute the wavelet coefficients.

2. Thresholding: Apply a thresholding function to the wavelet coefficients.

3. Reconstruction: Reconstruct the denoised signal using the fast inverse wavelet transform.

WAVELET SHRINKAGE – STATISTICAL MODELLING

Statistical Modeling in Wavelet Shrinkage

- Wavelet shrinkage uses **statistical modeling** to identify signal components from noisy data. The idea is based on the assumption that:

1. **Signal coefficients** (wavelet coefficients representing the actual signal) will have large magnitudes.
2. **Noise coefficients** (wavelet coefficients due to noise) will tend to be small in magnitude.

- Using statistical methods, noise is estimated, and thresholding rules are applied to distinguish signal from noise.
- The two main statistical models used are:

1. **Gaussian noise models**: Noise is assumed to follow a Gaussian distribution.
2. **Bayesian inference and Empirical Bayes shrinkage**: Probabilistic modeling is used to derive estimates of signal/noise levels and shrink coefficients accordingly.

NOISE ESTIMATION IN WAVELET SHRINKAGE

- Thresholding methods can be grouped into two categories: global thresholds and level-dependent thresholds. The former means that we choose a single value for threshold A to be applied globally to all empirical wavelet coefficients while the latter means that a possibly different threshold value A_j is chosen for each resolution level j.

SHRINKAGE FUNCTIONS

- Shrinkage functions are mathematical functions used in wavelet shrinkage to suppress noise while retaining the important signal information in the wavelet domain. These functions define how the wavelet coefficients are shrunk based on certain thresholds during the denoising process.
- Soft and Hard threshold are two of the shrinkage functions used in this project for denoising of the signal.

HARD THRESHOLD

$$F_h(x) = \begin{cases} x & \text{if } x \geq \lambda \text{ or } x \leq -\lambda, \\ 0 & \text{if } -\lambda < x < \lambda. \end{cases}$$

Sharp Transition

- Hard thresholding sets coefficients below the threshold to zero, resulting in a more aggressive noise reduction.

Risk for Abrupt Change or Distortions

- While effective in high noise scenarios, it can sometimes introduce artifacts or distort the original signal, especially in low noise situations.

SOFT THRESHOLD

$$F_s(x) = \begin{cases} x - \lambda & \text{if } x \geq \lambda, \\ 0 & \text{if } -\lambda < x < \lambda, \\ x + \lambda & \text{if } x \leq -\lambda. \end{cases}$$

Shrinking Toward Zero

- Soft thresholding shrinks coefficients towards zero, effectively reducing noise while preserving important signal features.

Smooth Transition

- It provides a smooth transition between shrinking and preserving coefficients, resulting in a more natural signal reconstruction.

APPLICATIONS AND CURRENT USAGE

- Hard and Soft thresholding are still relevant in the field of wavelet-based denoising due to their simplicity, computational efficiency.
- Despite advanced shrinkage methods replacing them in complex denoising tasks, hard and soft thresholding are still used in:
- Audio Denoising
- Compression schemes like JPEG 2000
- EEG/ECG Denoising: Replacement by Bayesian Inference Models.
- When artifacts and physiological noise are non-stationary, Bayesian shrinkage methods adapt better to varying noise patterns in EEG or ECG signals.
- Deep learning (DL)-based methods have replaced traditional wavelet-based approaches in several areas due to their ability to automatically learn complex patterns and adapt to non-linear data especially in areas like speech enhancement, Medical Imaging, Fault detection in Engineering and Bio metric recognition etc.

SHRINKAGE RULES

- In wavelet shrinkage-based denoising methods (both soft thresholding and hard thresholding), specific shrinkage rules are applied to wavelet coefficients. These rules determine how much to suppress or retain the coefficients during denoising. The shrinkage rules are used to distinguish between signal (which should be preserved) and noise (which should be suppressed).

UNIVERSAL THRESHOLD RULE

- The universal threshold is a popular heuristic for noise removal. It is defined based on the **noise variance** in the signal.

$$\lambda = \sigma\sqrt{2\log(N)}$$

- σ: Noise standard deviation.
- N: Length of the signal.

DONOHO UNIVERSAL THRESHOLD

- This is an implementation of the universal thresholding rule for wavelet denoising. It assumes that the noise follows a Gaussian distribution.
- The universal threshold is given by:

$$\lambda_{\text{universal}} = \sigma\sqrt{2\log(n)}$$

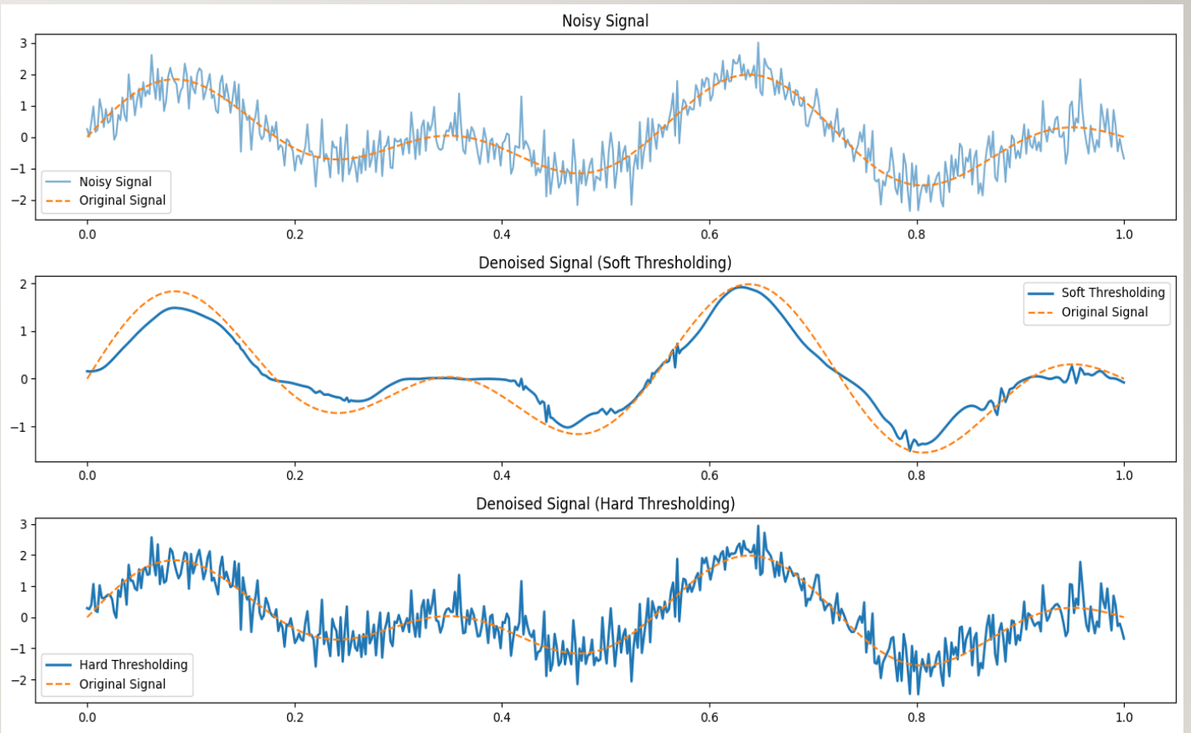
- σ: Estimated noise standard deviation.
- n: Number of samples.

IMPLEMENTATION IN PYTHON

- **Import Libraries**
  Import necessary libraries like ‘PyWavelets’ and ‘numpy’ and ‘matplotlib’.
- **Load Signal and Apply Wavelet Transform**
  Load the noisy signal and apply the wavelet transform using ‘pywt.wavedec’.
- **Apply Thresholding and reconstruct signal**
  Apply either soft or hard thresholding to the wavelet coefficients and reconstruct the denoised signal using ‘pywt.waverec’.

PYTHON IMPLEMENTATION AND RESULTS

```python
import pywt
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error

# Simulated signal with Gaussian white noise
np.random.seed(42)
x = np.linspace(0, 1, 500)
signal = np.sin(4 * np.pi * x) + np.sin(7 * np.pi * x)  # Original signal
noise = np.random.normal(0, 0.5, len(x))  # Gaussian white noise
noisy_signal = signal + noise

# Wavelet parameters
wavelet = 'db6'  # Daubechies wavelet
level = 4        # Decomposition level

# Perform Wavelet Transform
coeffs = pywt.wavedec(noisy_signal, wavelet, level=level)

# Calculate threshold based on noise variance (universal threshold)
threshold = np.sqrt(2 * np.log(len(noisy_signal))) * 0.1
threshold1 = np.sqrt(2 * np.log(len(noisy_signal))) * 0.4

# Apply Soft and Hard Thresholding
soft_coeffs = [pywt.threshold(c, threshold1, mode='soft') for c in coeffs]
hard_coeffs = [pywt.threshold(c, threshold, mode='hard') for c in coeffs]

# Reconstruct the signal for both methods
soft_denoised_signal = pywt.waverec(soft_coeffs, wavelet)
hard_denoised_signal = pywt.waverec(hard_coeffs, wavelet)

# Calculate Mean Squared Error (MSE)
mse_soft = mean_squared_error(signal, soft_denoised_signal[:len(signal)])
mse_hard = mean_squared_error(signal, hard_denoised_signal[:len(signal)])
```



THANK YOU