Fourier and Wavelet Transforms

Shawhin Talebi
The University of Texas at Dallas
Multi-scale Integrated Interactive Intelligent Sensing and Simulation (**MINTS**)

Time Series

A set of values indexed by time

Market Summary > Nasdaq Composite
INDEXNASDAQ: .IXIC

+ Follow

10,664.70 −128.58 (1.19%) ↓
Sep 21, 12:32 PM EDT · Disclaimer

1 day | 5 days | 1 month | 6 months | YTD | 1 year | 5 years | Max

Open	10,610.14	Low	10,519.49
High	10,683.94		

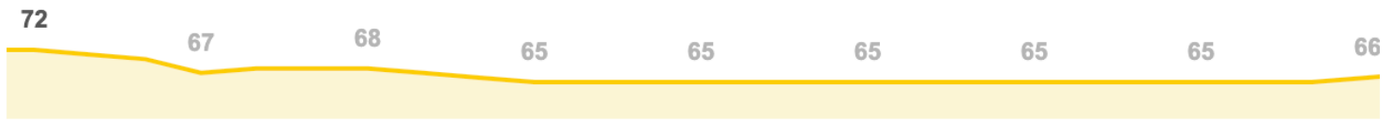
→ More about Nasdaq Composite

Richardson, TX
Monday 11:00 AM
Cloudy

72 °F | °C

Precipitation: 15%
Humidity: 72%
Wind: 12 mph

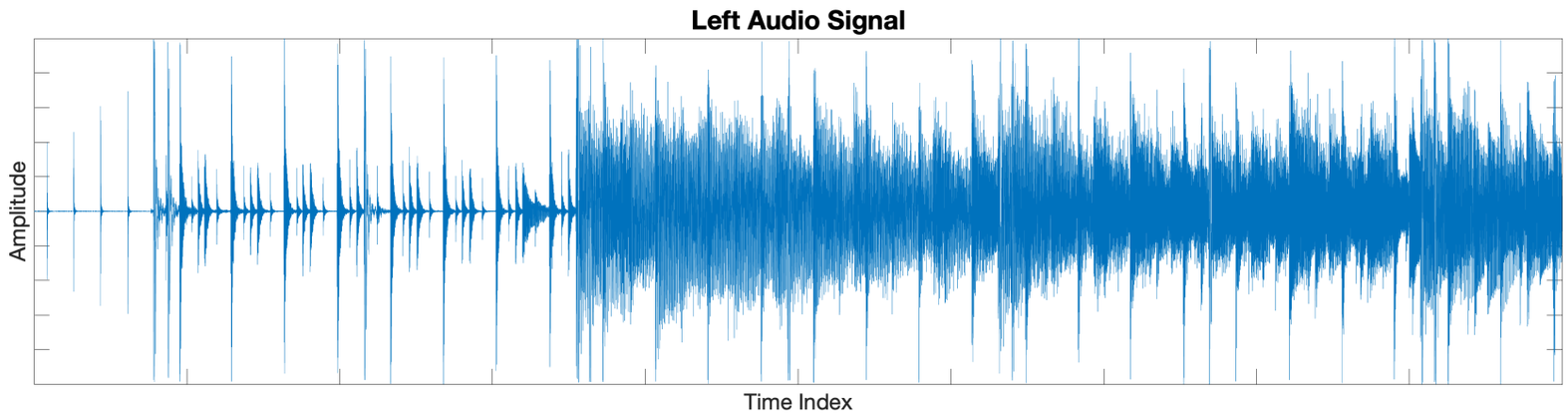
Temperature | Precipitation | Wind

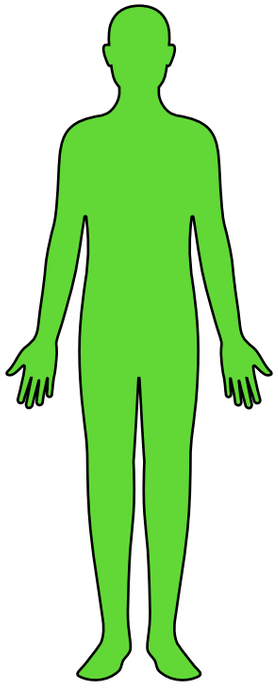


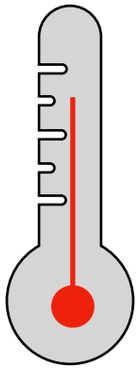
12 PM	3 PM	6 PM	9 PM	12 AM	3 AM	6 AM	9 AM
Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon
72° 64°	68° 62°	74° 60°	80° 62°	84° 64°	87° 67°	86° 64°	83° 61°

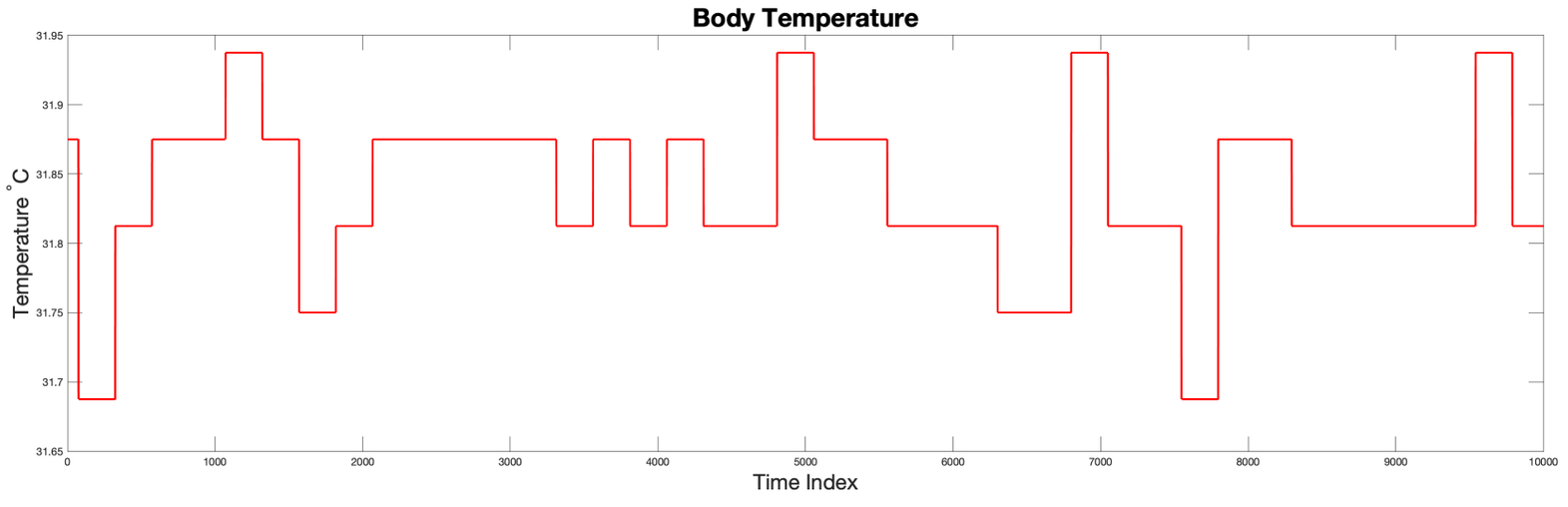
Signals

A set of values ordered by time representing a physical event







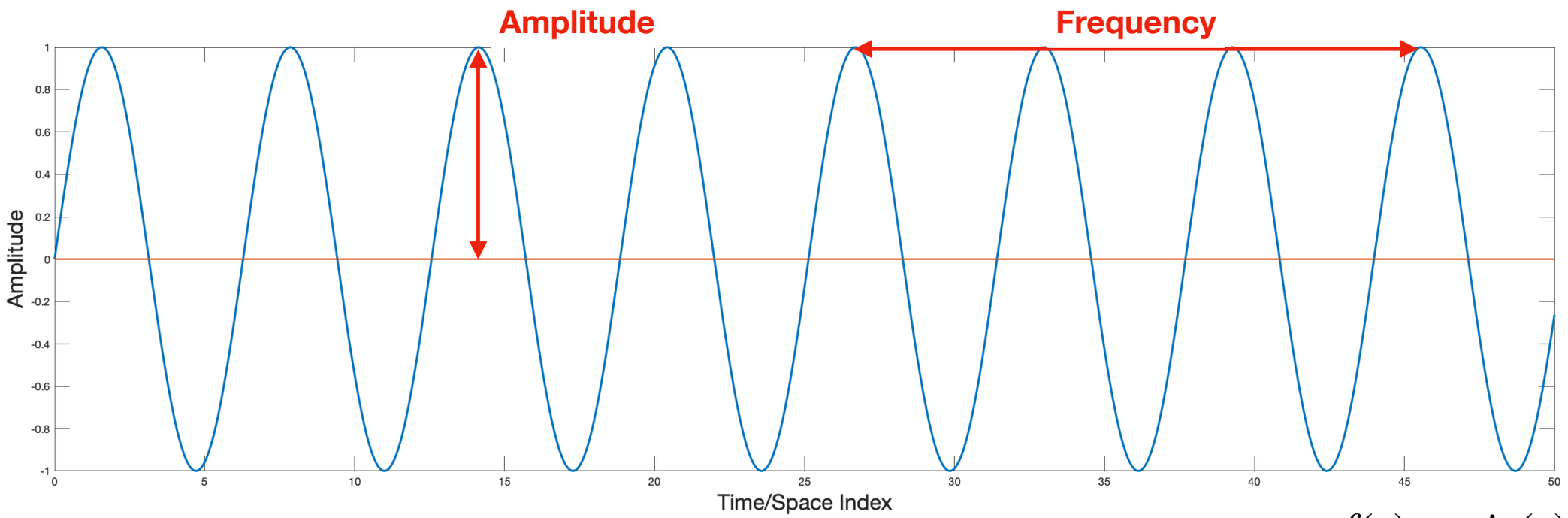


Waves

Oscillating quantity around an equilibrium

Two basic properties

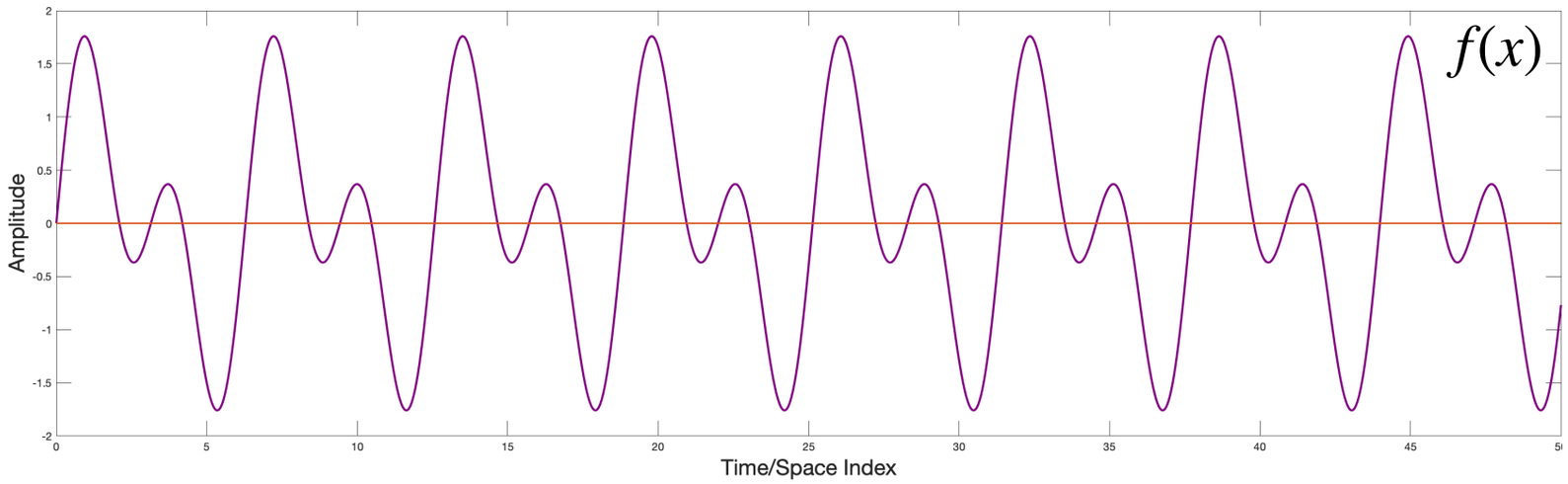
- **Amplitude** - magnitude of quantity
- **Frequency** - characterizes oscillation



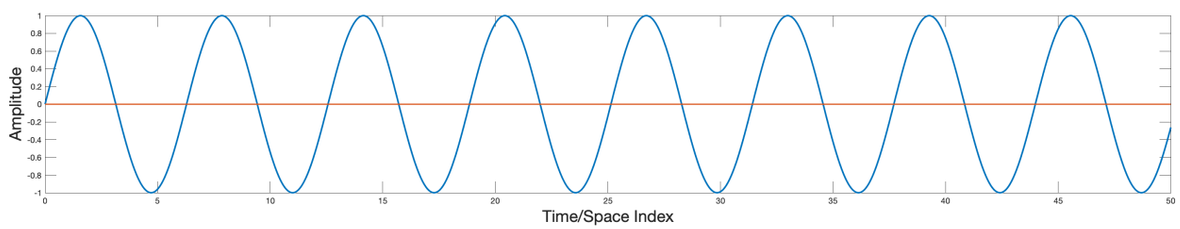
$$f(x) = sin(x)$$

Fourier Transform (FT)

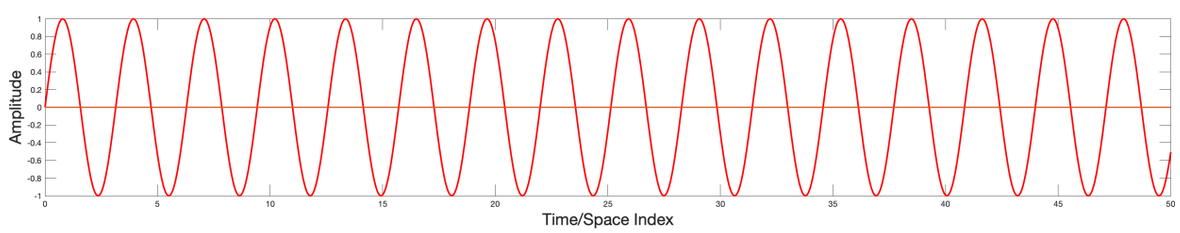
Decomposition of a signal using sinusoids



=



+



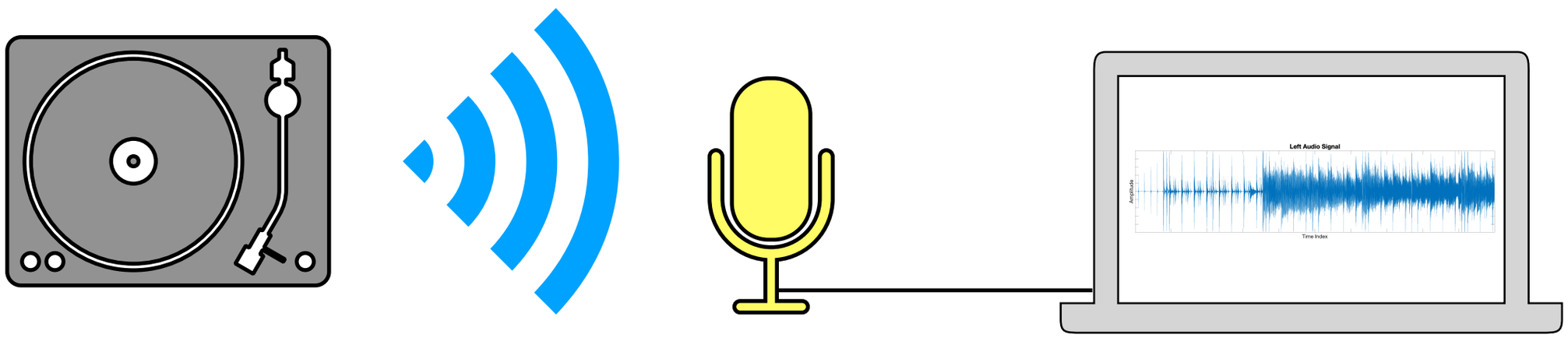
$$f(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$

Euler's Formula:

$$e^{ix} = cos(x) + isin(x)$$

Discrete Fourier Transform (DFT)

FT of discrete sequence



$$\mathbf{x} = (10 \quad 34 \quad 1 \quad \ldots \quad 6)$$

$$f(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$

FT

Discretize

$$f_k = \Sigma_0^{N-1} x_n e^{\frac{-2\pi i k n}{N}}$$

DFT

Discrete Fourier Transform (DFT)

FT of discrete sequence

$$f_k = \Sigma_0^{N-1} x_n e^{\frac{-2\pi i k n}{N}}$$

Let, $R_{kn} = e^{\frac{-2\pi i k n}{N}}$

=> $f_k = \Sigma_0^{N-1} R_{kn} x_n$

$$f_k \longrightarrow \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{K-1} \end{pmatrix} \quad x_n \longrightarrow \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix} \quad R_{kn} \longrightarrow \begin{pmatrix} R_{0,0} & R_{0,1} & \cdots & R_{0,N-1} \\ R_{1,0} & R_{1,1} & & \vdots \\ \vdots & & & \vdots \\ R_{K-1,0} & \cdots & \cdots & R_{K-1,N-1} \end{pmatrix}$$

O(n^2) **=>**

$$\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{K-1} \end{pmatrix} = \begin{pmatrix} R_{0,0} & R_{0,1} & \cdots & R_{0,N-1} \\ R_{1,0} & R_{1,1} & & \vdots \\ \vdots & & & \vdots \\ R_{K-1,0} & \cdots & \cdots & R_{K-1,N-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix}$$

Just matrix multiplication!

So what? How does this help us?

Discrete Fourier Transform (DFT)

FT of discrete sequence

Example: N = K = 4 **Recall,** $R_{kn} = e^{\frac{2\pi i k n}{N}}$

$$R = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-2\pi i/4} & e^{-4\pi i/4} & e^{-6\pi i/4} \\ 1 & e^{-4\pi i/4} & e^{-8\pi i/4} & e^{-12\pi i/4} \\ 1 & e^{-6\pi i/4} & e^{-12\pi i/4} & e^{-18\pi i/4} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}$$

Symmetric:
$R^T = R$

Defines resolution of FT

$$k_0 \equiv e^{-2\pi i/N}$$

$$R = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & k_0 & k_0^2 & k_0^3 \\ 1 & k_0^2 & k_0^4 & k_0^6 \\ 1 & k_0^3 & k_0^6 & k_0^9 \end{pmatrix}$$

4 x 4 case

Discrete Fourier Transform (DFT)

FT of discrete sequence

$$R_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & k_0 & -i & -ik_0 & -1 & -k_0 & i & ik_0 \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & -ik_0 & i & k_0 & -1 & ik_0 & -i & -k_0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -k_0 & -i & ik_0 & -1 & k_0 & i & -ik_0 \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & ik_0 & i & -k_0 & -1 & -ik_0 & -i & k_0 \end{pmatrix}$$

$$k_0 = \frac{1-i}{\sqrt{2}}$$

$$R_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix} \qquad R_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\boxed{k_0 \equiv e^{-2\pi i/N}}$$

Fast Fourier Transform (FFT)

Fast numerical algorithm for computing DFT

$$R_N = \begin{pmatrix} I_{N/2} & D_{N/2} \\ I_{N/2} & -D_{N/2} \end{pmatrix} \begin{pmatrix} R_{N/2} & 0 \\ 0 & R_{N/2} \end{pmatrix} (P) \qquad P\mathbf{x} = \begin{pmatrix} \mathbf{x}_{\text{even}} \\ \mathbf{x}_{\text{odd}} \end{pmatrix}$$

$$R_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix} = \left(\begin{array}{cc|cc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -i \\ \hline 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & i \end{array}\right) \left(\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{array}\right) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad D_2 = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix} \quad R_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

O(n^2)

O(n) + O(2n) + O(1)

↓

O(n log(n))

Just <u>efficient</u> matrix multiplication!

Fast Fourier Transform (FFT)

Fast numerical algorithm for computing DFT

O(n log(n))

$$R_{16} = \begin{pmatrix} I_8 & D_8 \\ I_8 & -D_8 \end{pmatrix} \begin{pmatrix} R_8 & 0 \\ 0 & R_8 \end{pmatrix} \begin{pmatrix} P_{16} \end{pmatrix}$$

$$R_{16} = \begin{pmatrix} I_8 & D_8 \\ I_8 & -D_8 \end{pmatrix} \begin{pmatrix} \begin{pmatrix} I_4 & D_4 \\ I_4 & -D_4 \end{pmatrix} \begin{pmatrix} R_4 & 0 \\ 0 & R_4 \end{pmatrix} \begin{pmatrix} P_8 \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} I_4 & D_4 \\ I_4 & -D_4 \end{pmatrix} \begin{pmatrix} R_4 & 0 \\ 0 & R_4 \end{pmatrix} \begin{pmatrix} P_8 \end{pmatrix} \end{pmatrix} \begin{pmatrix} P_{16} \end{pmatrix}$$

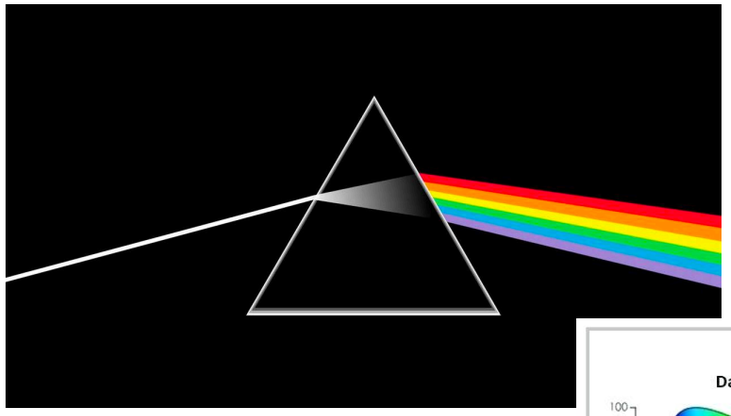
Recursive program

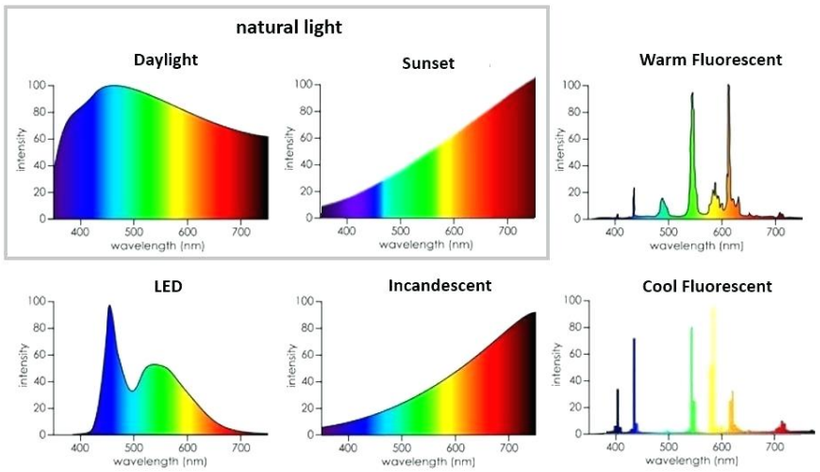
$$R_{16} = \begin{pmatrix} I_8 & D_8 \\ I_8 & -D_8 \end{pmatrix} \begin{pmatrix} \begin{pmatrix} I_4 & D_4 \\ I_4 & -D_4 \end{pmatrix} \begin{pmatrix} \begin{pmatrix} I_2 & D_2 \\ I_2 & -D_2 \end{pmatrix} \begin{pmatrix} R_2 & 0 \\ 0 & R_2 \end{pmatrix} \begin{pmatrix} P_4 \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} I_2 & D_2 \\ I_2 & -D_2 \end{pmatrix} \begin{pmatrix} R_2 & 0 \\ 0 & R_2 \end{pmatrix} \begin{pmatrix} P_4 \end{pmatrix} \end{pmatrix} \begin{pmatrix} P_8 \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} I_4 & D_4 \\ I_4 & -D_4 \end{pmatrix} \begin{pmatrix} \begin{pmatrix} I_2 & D_2 \\ I_2 & -D_2 \end{pmatrix} \begin{pmatrix} R_2 & 0 \\ 0 & R_2 \end{pmatrix} \begin{pmatrix} P_4 \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} I_2 & D_2 \\ I_2 & -D_2 \end{pmatrix} \begin{pmatrix} R_2 & 0 \\ 0 & R_2 \end{pmatrix} \begin{pmatrix} P_4 \end{pmatrix} \end{pmatrix} \begin{pmatrix} P_8 \end{pmatrix} \end{pmatrix} \begin{pmatrix} P_{16} \end{pmatrix}$$

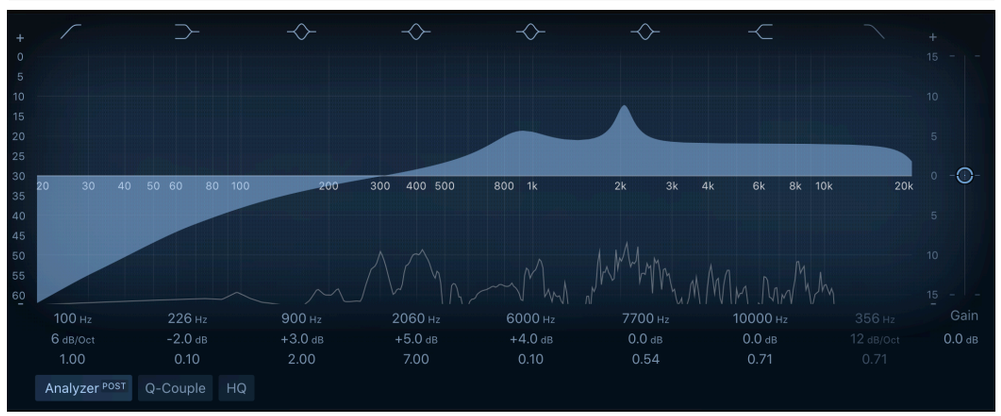
Matrix sparsity enables distributed computation

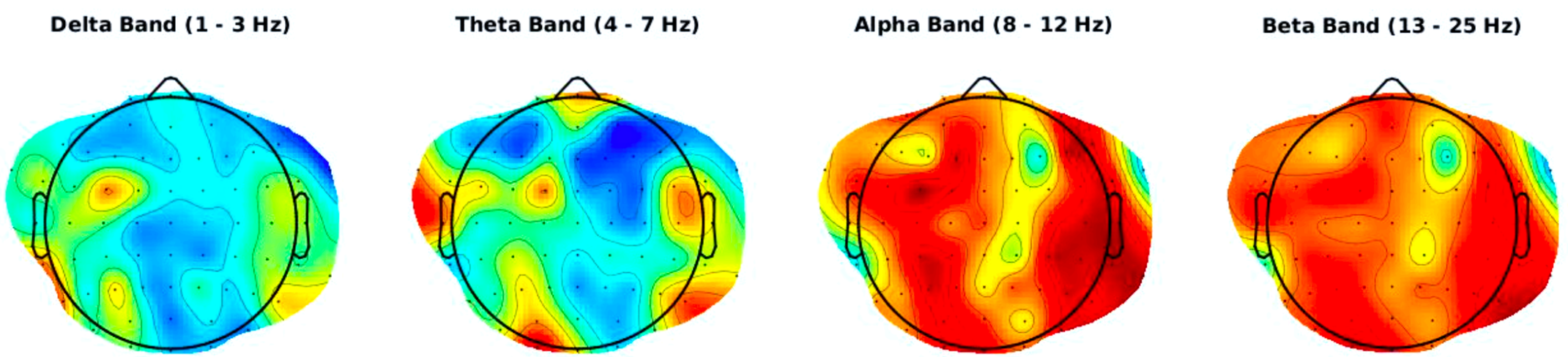
Spectral Analysis

Examination of signal via constituent frequency energies









Example

```
clear; clc; close all
```

Spectral Analysis of Electric Guitar Playing E2

Code Authored By: Shawhin Talebi

The University of Texas at Dallas

Mutli-scale Integrated Remote Sensing and Simulation (MINTS)

Read Audio File

```
[y, Fs] = audioread('Etest.WAV');
```

Perform FFT

```
x = fft(y);
```

Get length of audio signal

```
L = length(y);
```

Convert indicies to frequency values

```
f = Fs*(0:(L/2))/L;
```

Calculate 2-sided Power Spectrum

```
P2 = abs(x/L);
```

Calculate 1-sided Power Spectrum

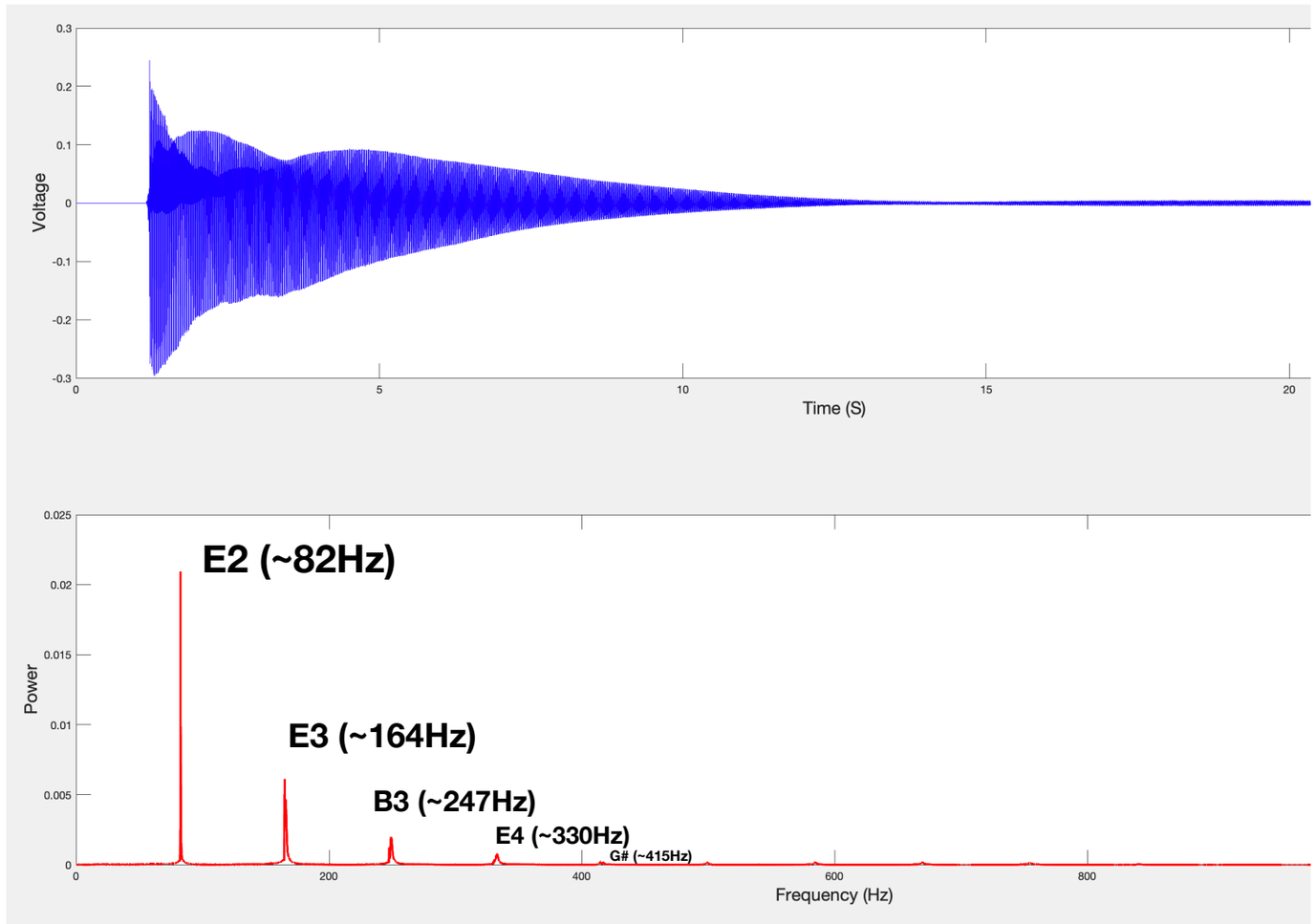
```
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
```

Plot Audio Signal

```
subplot(2,1,1)
plot((1:length(y))/Fs, y, 'b-')

% format time domain plot
ax=gca;
ax.XLabel.String = 'Time (S)';
ax.XLabel.FontSize = 16;
ax.YLabel.String = 'Voltage';
ax.YLabel.FontSize = 16;
```

Plot Power Spectrum of Audio Signal

```
subplot(2,1,2)
plot(f(1:25000),P1(1:25000),'r-', 'LineWidth', 2);

% format frequency domain plot
ax=gca;
ax.XLabel.String = 'Frequency (Hz)';
ax.XLabel.FontSize = 16;
ax.YLabel.String = 'Power';
ax.YLabel.FontSize = 16;
```

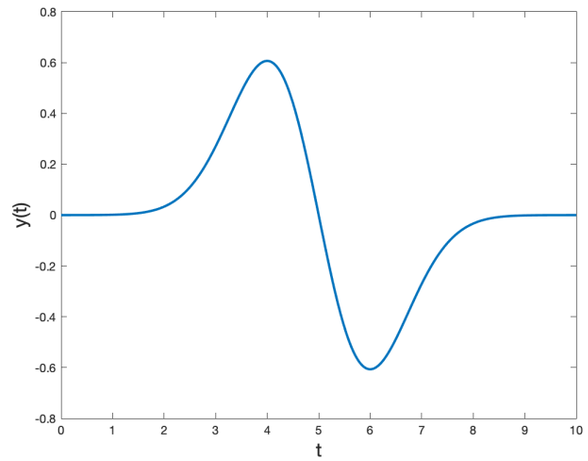


https://github.com/ShawhinT/F20_PHYS5315_GuestLectures

Wavelets

Wave-like oscillation *localized* in space (time)

Example:



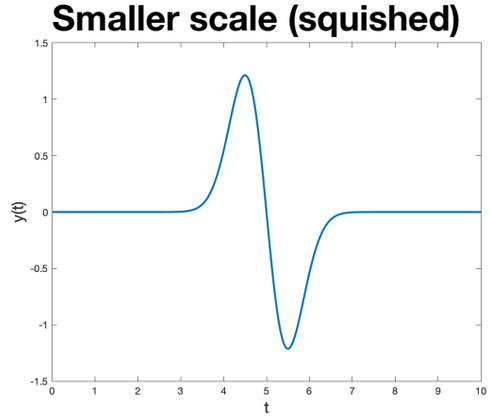
$$-(x-b)e^{\frac{-(x-b)^2/(2a^2)}{\sqrt{2\pi}a^3}}$$

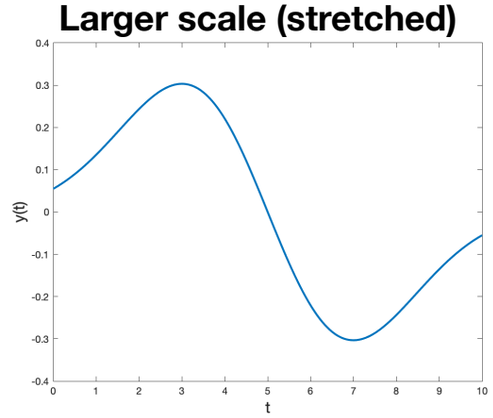
First derivative of Gaussian Function

[1] Two basic properties:

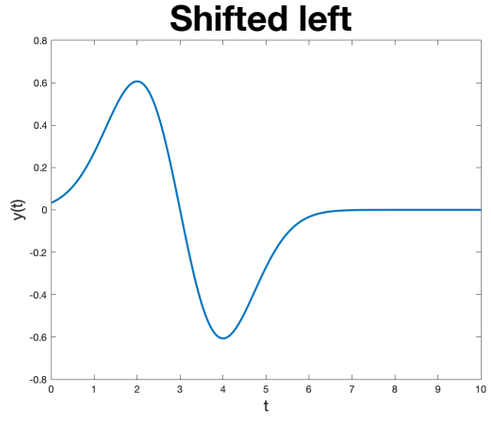
- **Scale (dilation)** - how “stretched” or “squished” wavelet is (related to frequency)

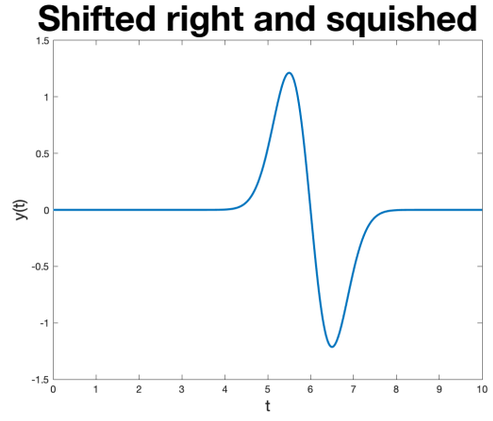
- **Location** - where wavelet is positioned in time





a



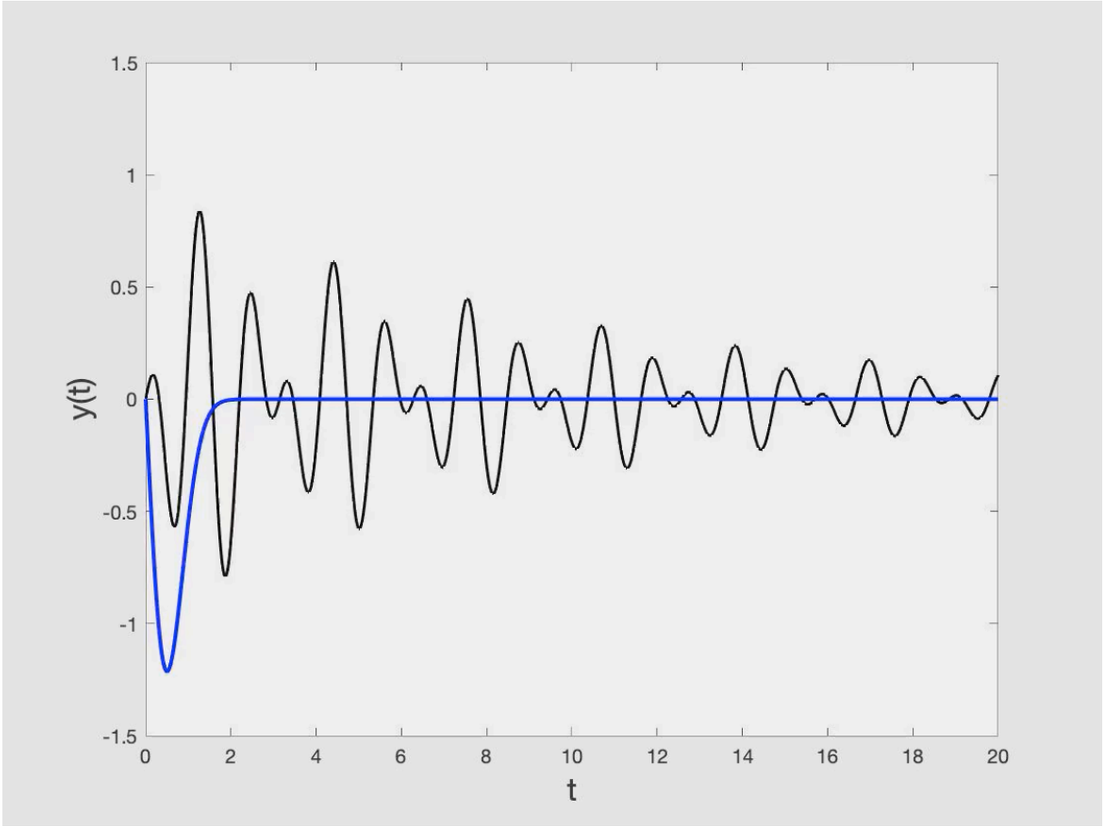


b

[1] Addison, P.S. (2005). Wavelet transforms and the ecg: A review. Physiological Measurement, 26 (5), p. R155

Wavelet Transform

Decomposition of a signal using wavelets of varying scale and location



Basic idea: compute *how much* of wavelet is *in* the signal for a particular scale and location

i.e. evaluate convolution of signal and wavelet at varying *scales*

Why wavelets?

- Traditional **Fourier transform** (decomposition using sine and cosine) gives global average over entire signal, thus may obscure local information
- **Wavelet transforms** can extract local spectral **and** temporal information simultaneously
- **Variety of wavelets** to choose from

Addison, P.S. (2005). Wavelet transforms and the ecg: A review. Physiological Measurement, 26 (5), p. R155

Wavelet Transform

Two methods

Continuous Wavelet Transform (CWT)

"Mother" (basis) wavelets are defined everywhere

$$\psi = \psi(t)$$

Transform must be discretized for implementation

$$T(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\, \psi^* \frac{(t-b)}{a} dt$$

$T(a,b)$ may have redundant information since same feature may be *captured* by multiple scales

Discrete Wavelet Transform (DWT)

"Mother" (basis) wavelets are **only defined** on discrete grid

$$\psi = \psi_{m,n}(t)$$

Transform is already discrete!

$$T_{m,n} = \int_{-\infty}^{\infty} x(t)\, \psi_{m,n}(t)\, dt$$

Coefficients ($T_{m,n}$) do not have redundant information since discretized wavelets can be defined to be orthonormal

$$\int_{-\infty}^{\infty} \psi_{m,n}(t) \psi_{m',n'}(t)\, \mathrm{d}t = \begin{cases} 1 & \text{if} \quad m = m' \text{ and } n = n' \\ 0 & \text{otherwise.} \end{cases}$$

Addison, P.S. (2005). Wavelet transforms and the ecg: A review. Physiological Measurement, 26 (5), p. R155

Example

```
clear; close all; clc
```

Code to use Maximal Overlap Discrete Wavelet Transform (MODWT) to Analyze ECG Data

Code authored by: Shawhin Talebi

The University of Texas at Dallas

Mutli-scale Integrated Remote Sensing and Simulation (MINTS)

Load Data

```
load('ECG.mat');
```

Perform Wavelet Transform

Use the maximal overlap discrete wavelet transform (MODWT) to enhance the R peaks in the ECG waveform. The MODWT is an undecimated wavelet transform, which handles arbitrary sample sizes.

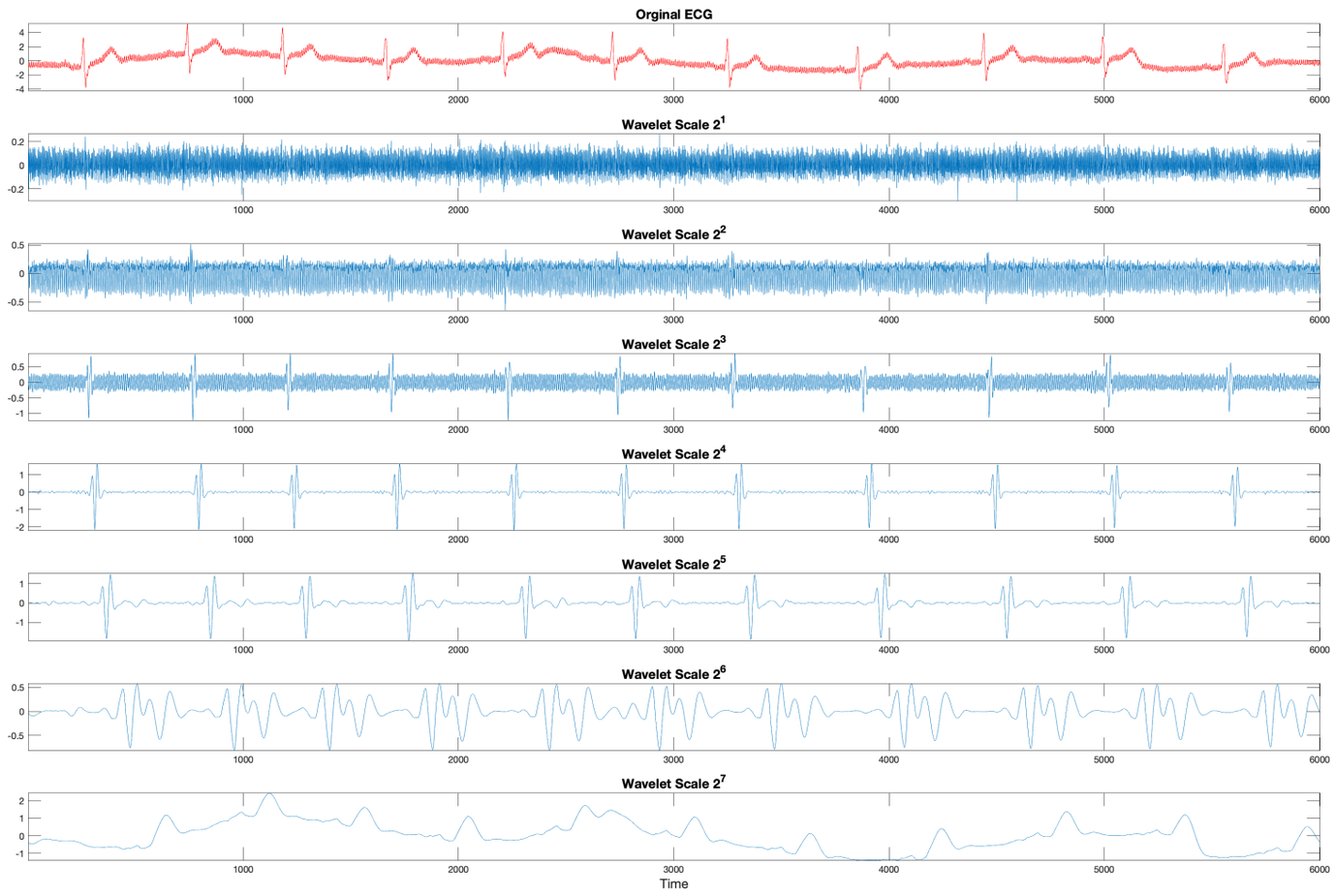
Perform MODWT for 6 different scales. The rows of wt give the wavelet (detail) coefficents for each scale. Here we are using Symlets wavelet with 4 vanishing moments

Other wavelet families can be used. use waveletfamilies('f') to view all

```
% define number of levels to use
numLevels = 6;

% perform maximal overlap discrete wavelet transform (MODWT)
wt = modwt(ECG, 'sym4', numLevels);
```

Compare Coefficents for all MODWT Scales

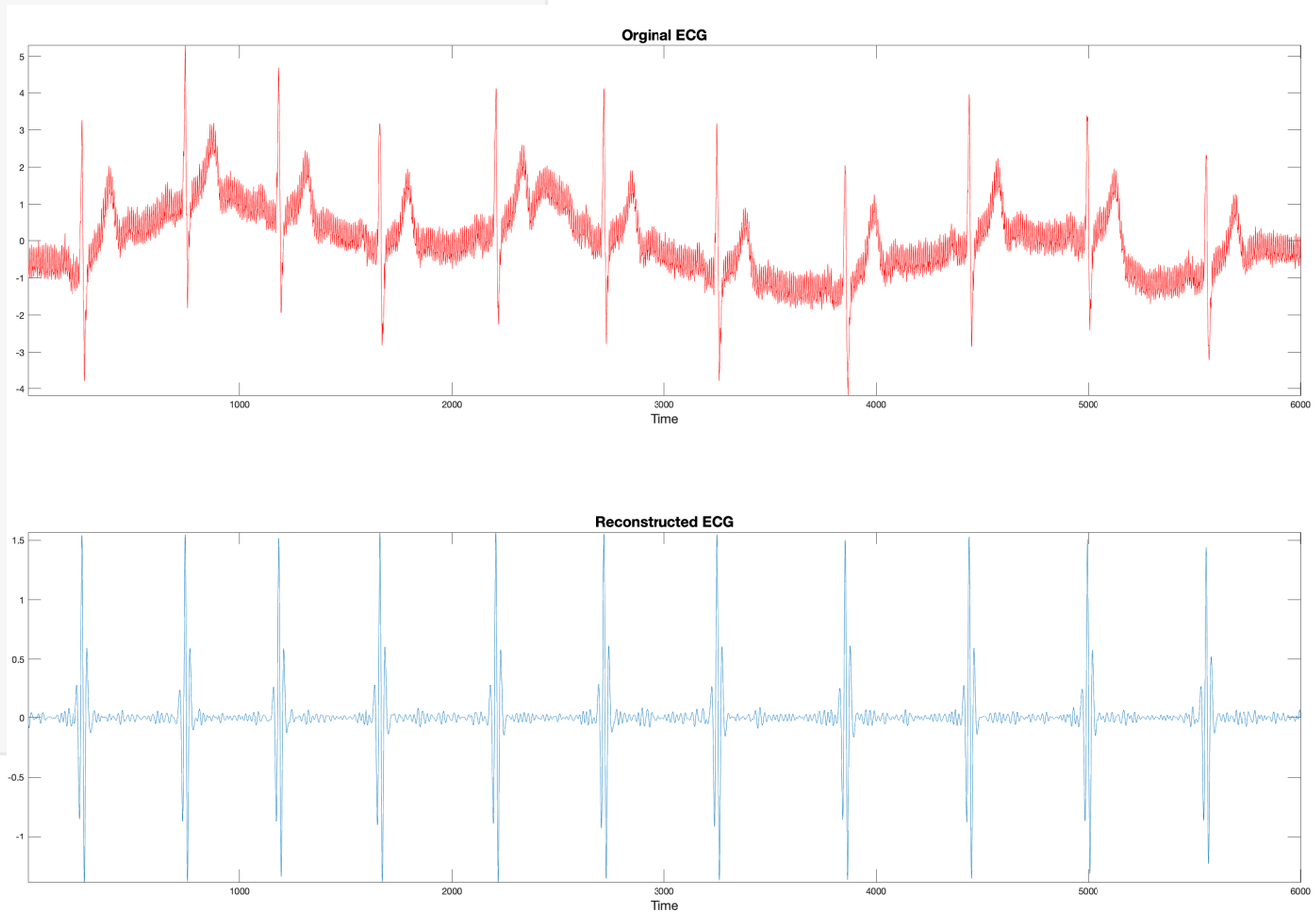
```
% get number of Wavelet Scales
[numScales, ~] = size(wt);

% create figure
fig = figure(1);
fig.Units = 'normalized';
fig.Position = [0 0 1 1];

% plot original signal
subplot(numScales+1,1,1)
plot(ECG, 'r-')
title('Orginal ECG', 'FontSize', 14)
axis tight

% plot wavelet scale coefficents
for i=2:numScales+1

    subplot(numScales+1,1,i)
    plot(wt(i-1,:))
    title(strcat('Wavelet Scale 2^', string(i-1)), 'FontSize', 14)
    axis tight
end

xlabel('Time', 'FontSize', 14)

% print figure to file
print('waveletTransform_signal_decomposition', '-dpng');
```



Example

Reconstruct Signal via Inverse MODWT

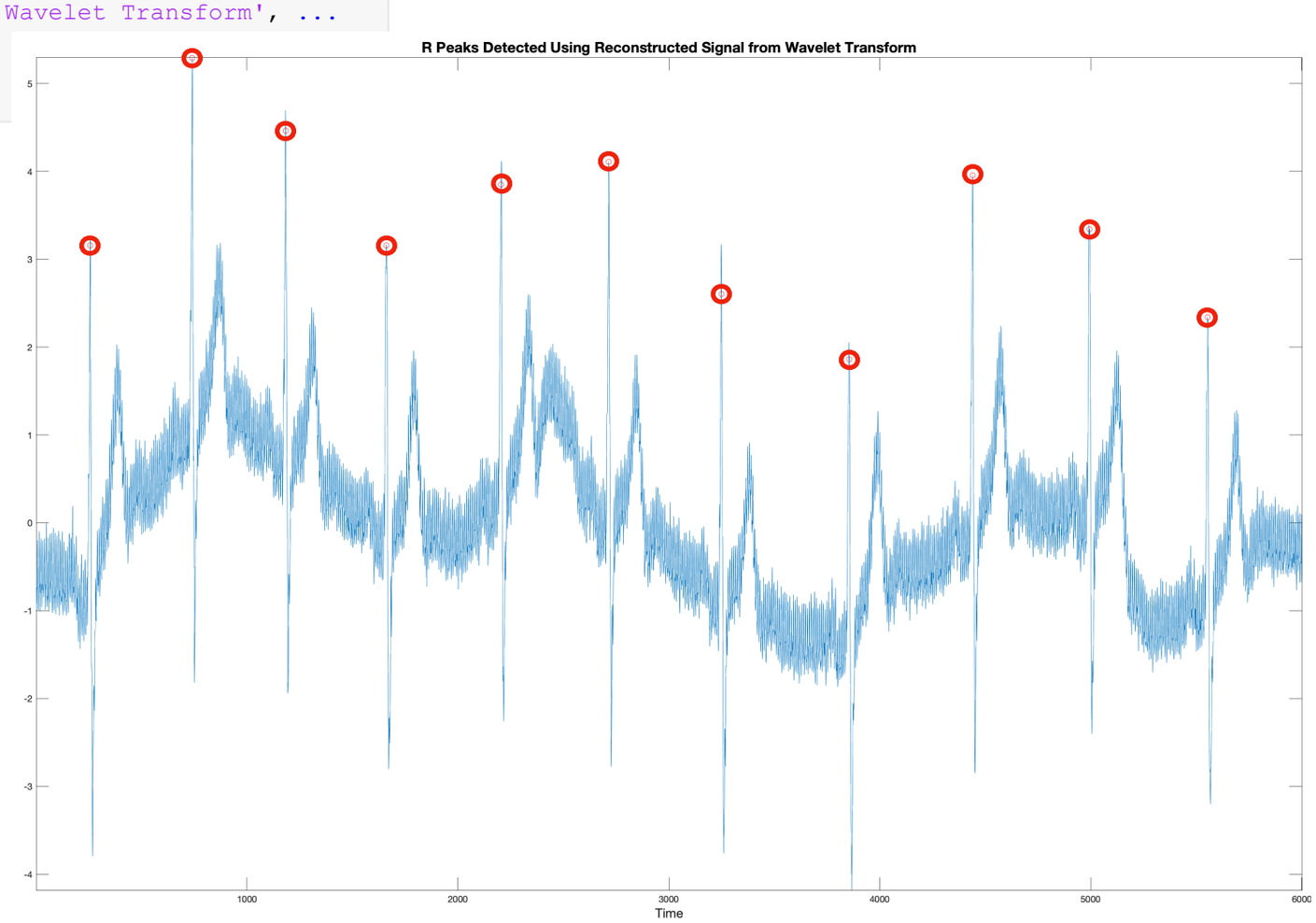
```
% create figure
fig = figure(2);
fig.Units = 'normalized';
fig.Position = [0 0 1 1];

% initialize an array to store the coefficents corresponding to scale 2^4
recwt = zeros(size(wt));

% store coefficents corresponding to scale 4 in array
recwt(4,:) = wt(4,:);

% reconstruct signal using only scale 4 with inverse transform
recECG = imodwt(recwt,'sym4');


% compare original signal and reconstructed
% plot original signal
subplot(2,1,1)
plot(ECG, 'r-')
title('Orginal ECG', 'FontSize', 16)
xlabel('Time', 'FontSize', 14)
axis tight
% plot reconstructed signal
subplot(2,1,2)
plot(recECG)
title('Reconstructed ECG', 'FontSize', 16)
xlabel('Time', 'FontSize', 14)
axis tight

% print figure to file
print('waveletTransform_reconstructed_signal', '-dpng');
```



Example

Find R Peaks Using Reconstructed Signal

```
% define array of time indicies
t = 1:length(ECG);

% square reconstructed signal
recECG = abs(recECG).^2;
% find peaks in squared reconstructed signal
[qrspeaks,locs] = findpeaks(recECG, t, 'MinPeakHeight', 0.35, 'MinPeakDistance',100);

% create figure
fig = figure(3);
fig.Units = 'normalized';
fig.Position = [0 0 1 1];

% plot signal with detected r peaks using reconstructed signal from DWT
plot(t,ECG)
hold on
plot(locs,ECG(locs),'ro')
xlabel('Time', 'FontSize', 14)
title('R Peaks Detected Using Reconstructed Signal from Wavelet Transform', ...
    'FontSize', 16)
axis tight
print('waveletTransform_rpeak_annotation', '-dpng');
```



https://github.com/ShawhinT/F20_PHYS5315_GuestLectures