The Continuous Wavelet Transform

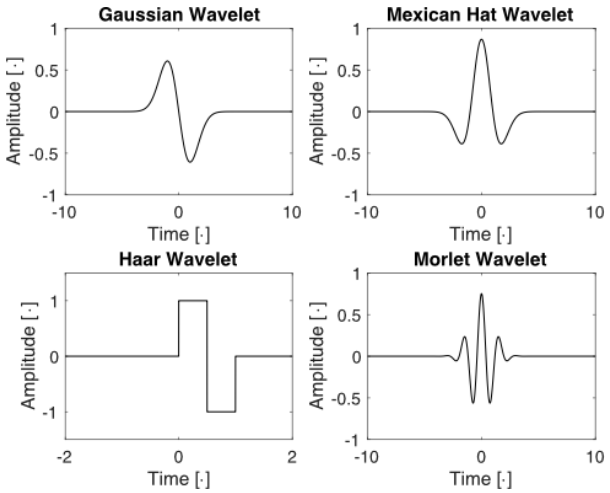
These notes correspond to Slides 6-7.

- The continuous wavelet transform (CWT) overcomes some limitations of the Fourier transform by extending the analysis to more general bases.
- The CWT is a convolution between a wavelet and a signal:

(1) (2)

- A wavelet is a localized waveform that satisfies certain mathematical criteria.

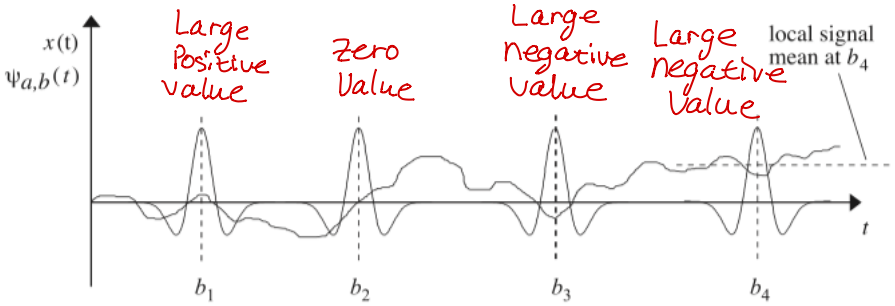
Some example wavelets are shown to the right.

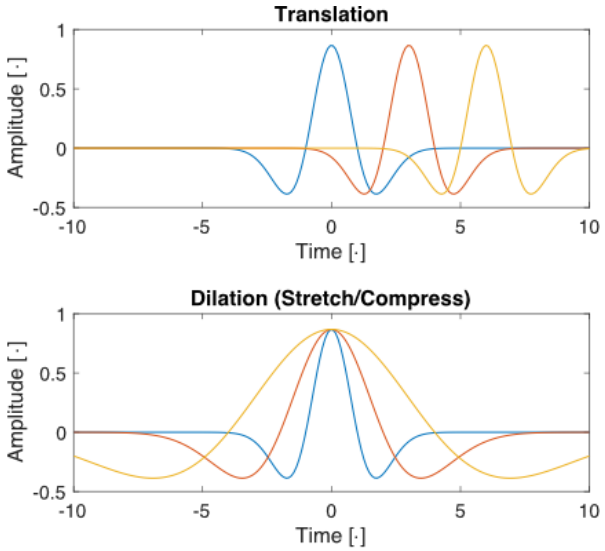


- There are many mother wavelets to choose from and which you use depends on the analysis at hand.

The Morlet wavelet is used throughout this seminar as it allows us to conveniently relate the scale a to frequency ω or f.

- The convolution operation dilates and translates the wavelet and compares the result with the signal $x(t)$.





- The convolution theorem is used to express the CWT as a multiplication of the Fourier transforms of the Signal, $X(\omega)$, and the wavelet, $\Psi(\omega)$.

Further manipulation allows us to express the CWT as an inverse Fourier transform:

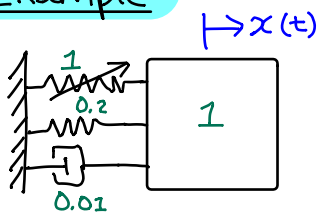
(3) (4)

- For the Morlet wavelet, $\omega = \frac{\omega_0}{a}$, where ω_0 is the central frequency of the wavelet.

- The code used to compute the CWT spectra in this seminar is provided below:

```
function [freq,modulus] = WaveletTransform(time,signal,minFreq, ...
    maxFreq,options)
% Help truncated in this screenshot
% Citation
% --------
% K.J. Moore, M. Kurt, M. Eriten, D.M. McFarland, L.A. Bergman,
% A.F. Vakakis, "Wavelet-Bounded Empirical Mode Decomposition for
% Measured Time Series Analysis," Mechanical Systems and Signal
% Processing, 99:14-29, 2018.
% https://dx.doi.org/10.1016/j.ymssp.2017.06.005

arguments
    time (:,1) double
    signal (:,1) double
    minFreq double
    maxFreq double
    options.numFreq double = 100;
    options.motherWaveletFreq double = 2;
end

% Transform Parameters
dt = time(2)-time(1);
lengthSignal = length(signal);
df = (maxFreq-minFreq)/options.numFreq; % Frequency Resolution
freq = minFreq:df:maxFreq; % Prescribe the frequencies of interest
waveletScale = options.motherWaveletFreq./freq; % Compute scales

% FFT Parameters
nfourier = 2^nextpow2(lengthSignal); % Zero-filling
npt = nfourier/2;
Fourierfreq = 1/dt*(0:npt-1)/nfourier; % Frequency vector

% Compute FFT of xnew
signalFFT = fft(signal,nfourier);
signalFFT(npt+1:end) = [];

% Vectorized Computation of Wavelet Transform of signal
core2 = bsxfun(@times,conj(bsxfun(@times,(2^0.5)*exp(-0.5* ...
    (2*pi*(bsxfun(@times,Fourierfreq',waveletScale)- ...
    options.motherWaveletFreq)).^2),sqrt(waveletScale))), ...
    signalFFT);

% Assert Admissibility Condition
if minFreq == 0
    core2(:,1) = 0;
end
result = ifft(core2,nfourier);
result1 = result(1:lengthSignal,:);
modulus  = abs(result1);
end
```

Example

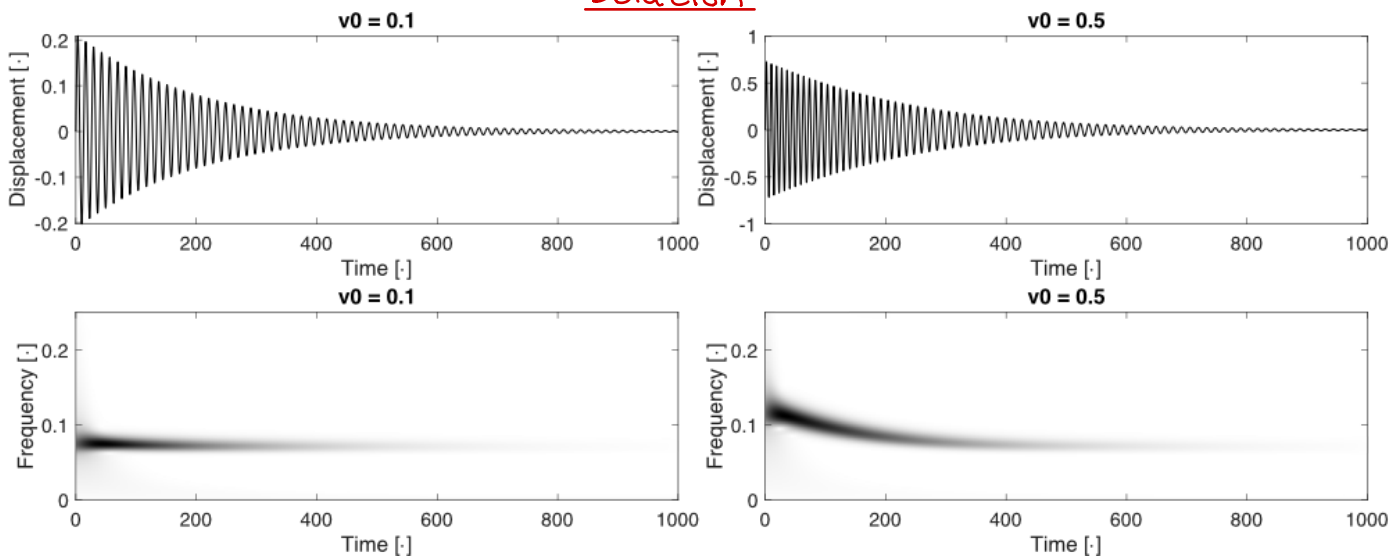


Consider a Duffing oscillator with the following equation of motion:

$$\ddot{x} + c\dot{x} + kx + \alpha x^3 = 0 \quad (5)$$

Simulate the response for $v_0 = 0.1$ and $v_0 = 0.5$, then plot and compare the wavelet spectra.

Solution




```
clc,clear,close all
% Parameters
m = 1; % kg
c = 0.01; % Ns/m
k = 0.2; % N/m
alpha = 1; % N/m^3

% Time
dt = 1e-1;
t = 0:dt:1000;

options = odeset;
options.RelTol = 1e-8;
options.AbsTol = 1e-9;

v0 = 0.1;
[~,y1] = ode45(@(t,y) sys(t,y,m,c,k,alpha),t,[0;v0],options);
v0 = 0.5;
[~,y2] = ode45(@(t,y) sys(t,y,m,c,k,alpha),t,[0;v0],options);

% Compute Wavelet Transforms
[freq1,mods1] = WaveletTransform(t,y1(:,1),0,0.25,"numFreq",200);
[freq2,mods2] = WaveletTransform(t,y2(:,1),0,0.25,"numFreq",200);

figure
subplot(2,1,1)
plot(t,y1(:,1),'k')
xlabel('Time [\cdot]'); ylabel('Displacement [\cdot]')
title('v0 = 0.1'); set(gca,'fontsize',14)

subplot(2,1,2)
imagesc(t,freq1,mods1'); set(gca,'ydir','normal')
title('v0 = 0.1'); set(gca,'fontsize',14)
colormap(flipud(gray))

figure
subplot(2,1,1)
plot(t,y2(:,1),'k')
ylabel('Displacement [\cdot]'); xlabel('Time [\cdot]')
title('v0 = 0.5'); set(gca,'fontsize',14)

subplot(2,1,2)
imagesc(t,freq2,mods2'); set(gca,'ydir','normal')
xlabel('Time [\cdot]'); ylabel('Frequency [\cdot]')
title('v0 = 0.5'); set(gca,'fontsize',14)
colormap(flipud(gray))

function dy = sys(t,y,m,c,k,alpha)
dy(1,1) = y(2);
dy(2,1) = -1/m*(c*y(2)+k*y(1)+alpha*y(1).^3);
end
```

Download Codes:

https://github.com/KeeganJMoore/Wavelet-Transform-Demonstration