# Data Analysis and Modelling

---

# A Practical Guide to Wavelet Analysis and Noise Techniques

Joseph Pritchard - Adelaide University

April 11, 2023

## Introduction

The Wavelet Transform (WT) process is showcased, in detail, through application of several wavelet functions (Mexican Hat, Morlet, Gaussian) on a given data set. A WT is performed on an associated white noise time-signal and used as a statistical confidence benchmark.

## Determination of Sample Rate Given Driving Data Signal

Note the following technique is designed requiring the input signal be a single frequency signal over the data points provided. It is however sufficient every successive data point equal in magnitude to the first in the set label a half period.

MATLAB software records the magnitude of the first data point in the input file. Data is then scanned for succesive data points of the same magnitude and used to label the periodicity of the data. A desired number of cycles to average over may be specified by the user. In the given sample it is found 16 cycles provides sufficient accuracy to exactly determine the frequency composition of the 145Hz driving signal [2]. The calculated sample rate is $4.4116 \times 10^4 Hz$.

## Wavelet Transform Method

A wavelet function is selected for analysis. The scale values $s$ adjust the width of the wavelet function and as such provide a means of detecting signal frequency. Widths similar in dimension to signal frequency peak width will have high overlap, whilst those far from this value will have minimal overlap. Time offset is determined by the time span of the original signal and is controlled by the parameter $t_0$. This relation can be expressed as

$$\psi(t) = \psi\left(\frac{t - t_0}{s}\right)$$

where $\psi(t)$ represents the time space representation of our chosen wavelet function.

Due to the finite nature of the data sets analyzed by the WT there is a limit on the clarity of signal in the resulting plot. Erroneous signals arrising from this process may be discounted by introducing a cone of influence marker[1]. Throughout, this cone is represeneted as a solid red line on the WT plots. The functional form of the cone is dependent on the chosen wavelet function [3].

## Increased Reliability

Due to their localized nature [1], WT performance can be improved by either decreasing the sampling frequency or decreasing the total time duration of the input signal. In our analysis both methods are utilized, reducing sampling frequency by a factor of 4 to $1.1029 \times 10^4 Hz$ and limiting the number of time data points at $2^{13}$. Restricting time data size to a power of 2 protects against zero padding errors inherent in the MATLAB fft function.

## Noise Generation

Inbuilt MATLAB functionality is used to generate a noise signal. For every data point in the original signal a pseudo-random number is generated between 0 and 1, then multiplied by the standard deviation of the original time signal. The WT is then taken as described above. By virtue of the uniform distribution of the applied MATLAB pseudo-random number generator, the generated noise is white in nature.

# Results

## Mexican Hat Wavelet

The following is the resulting WT utilizing a Mexican Hat wavelet function [1][Figure 1]. For comparison, observe the Fourier Transform (FT) frequency space plot of the output signal shown in figure 2. Observe the periodic peak features within the WT throughout the time domain occuring at a frequency of approximately $50Hz$. Corresponding peaks can be seen in the FT plot at around $38Hz$ and $70Hz$. In conjunction with the FT plot provided we observe the WT exhibits a smoothing property across the underlying dominant frequencies, visible in the delocalized nature of the $50Hz$ WT frequency peak.

Due to the time-localized nature of the mexican hat wavelet, we expect time domain peaks to be clearly visible in the WT plot. This is evidenced in the time-periodic nature of the frequency peaks in the WT, which can be linked to peaks present in the output data signal.

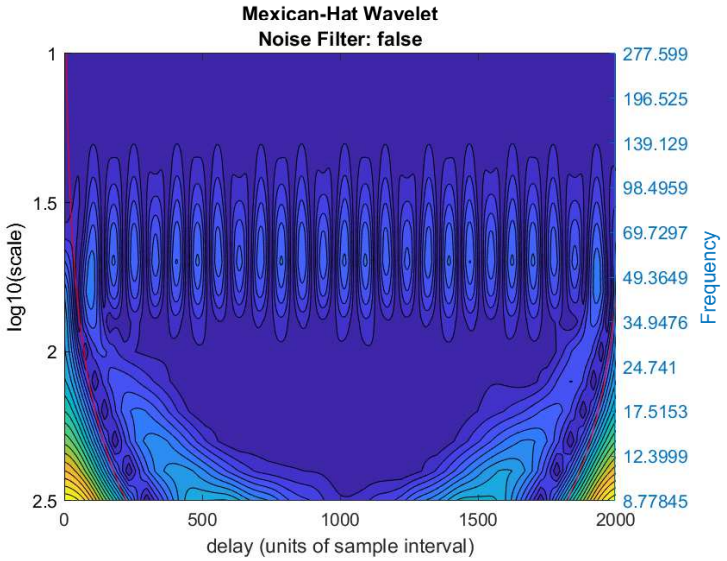


Figure 1:
Time variant frequency peak at $50Hz$ with frequency delocalization. Low frequency signal later attributed to noise - see noise filtering.

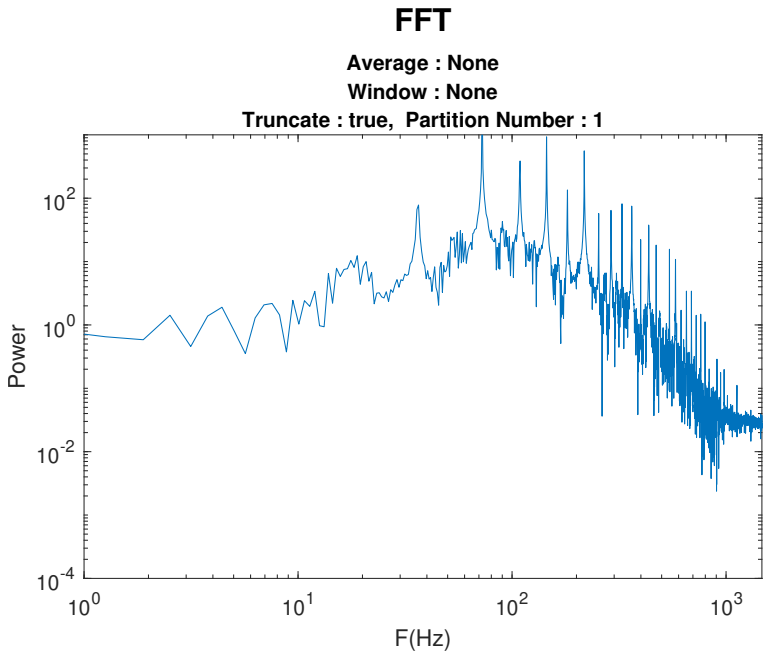


Figure 2:
Fourier Transform of the truncated input data signal. Averaging and windowing are not performed. Note peaks at approximately $38Hz$, $75Hz$, $103Hz$ and higher orders.

# Noise Filter

Applying the same wavelet transformation process to a white noise time signal yields the WT in figure 3. By comparing the WT output data values with their corresponding noise value, we are able to remove data values statiscially insignificant with respect to randomly generated white noise. Thus is done by direct comparison and setting insignificant values to be approximately zero. The resulting plot can be seen in figure 3.

Note that the method of pseudo-random number generation provides a random baseline of noise which, given a different seed, results in an alternate noise WT and thus different filtered output WT. This should be considered when inferring significance of WT points close to high noise regions.

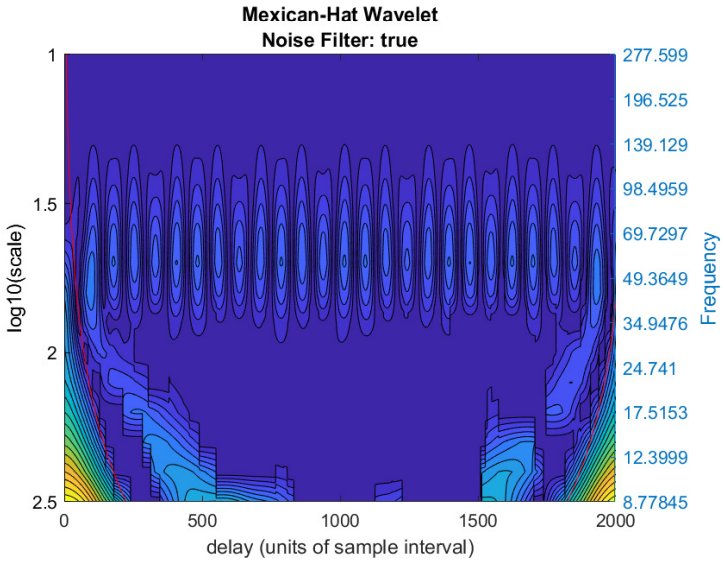


Figure 3:
Wavelet Transformed white noise - note the dominance at low frequencies.

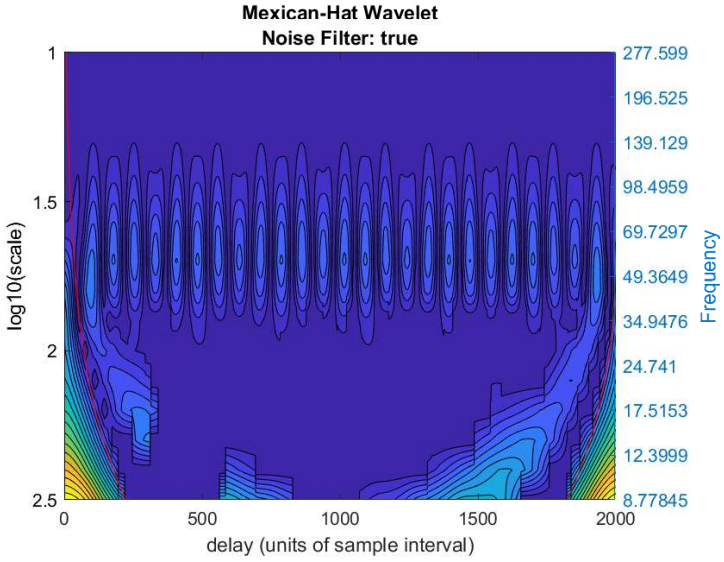


Figure 4:
WT with noise filter applied. Note the time variant frequency peak close to $50Hz$ with frequency delocalization.

## Morlet Wavelet

Redefining the wavelet function as the Morlet Wavelet function [1] the analysis yields the WT in figure 5.

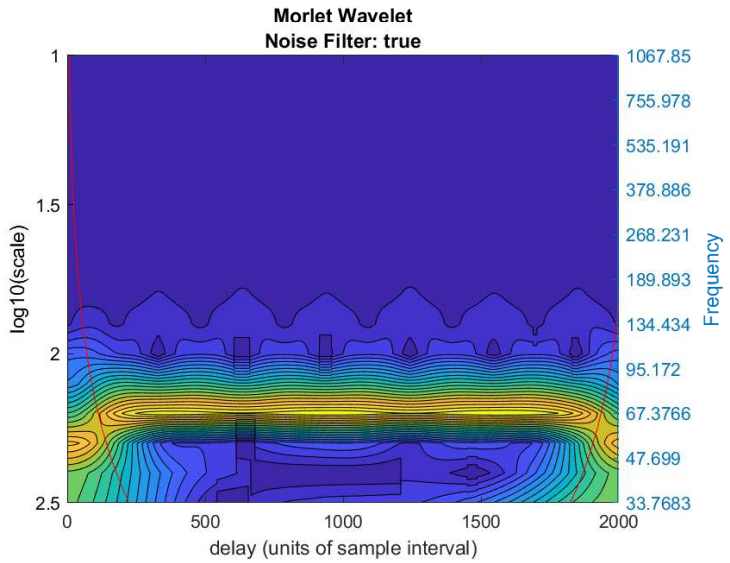


Figure 5:
Morlet Transformed output signal with noise filter. Note prominent time-invariance of the frequency peak at approximately $67Hz$.

Contrary to the Mexican Hat Wavelet case we observe a frequency peak consistent throughout the time domain. This occurs at a value close to $67Hz$. The Morlet wavelet function features a damped time oscillation of the form

$$e^{i\omega t}e^{-bt^2}$$

Due to the presence of the highly oscillatory $e^{iwt}$ term, we expect high overlap with a time oscillating test signal, and thus increased clarity in the WT frequency domain. This is in contrast to the Mexican Hat case[1] where, due to the time localized nature of the wavelet function, frequency peaks are not as sharply defined.

At the expense of increased frequency resolution we witness a decrease in time-domain clarity. Notable by the constant nature of the signal across all time values, again contrasting the Mexican Hat case.

## Gaussian Wavelet

Analysis using a Gaussian Wavelet function yields the WT observed in figure 6. Here we witness a reduction in frequency peak amplitude in exchange for increased time resolution. Analogous to the Mexican Hat being less oscillatory than the Morlet Wavelet, we again have a reduction in wavelet function time-oscillation. The pure Gaussian function features a single localized peak in the time domain. Peaks in the test signal are picked out with greater clarity, leading to increased spacing between features along the $44Hz$ frequency line. Oscillatory behavior is not as clearly resolved, leading to the reduced ampitude of frequency peaks in comparison to the Mexicna Hat and Morlet cases.

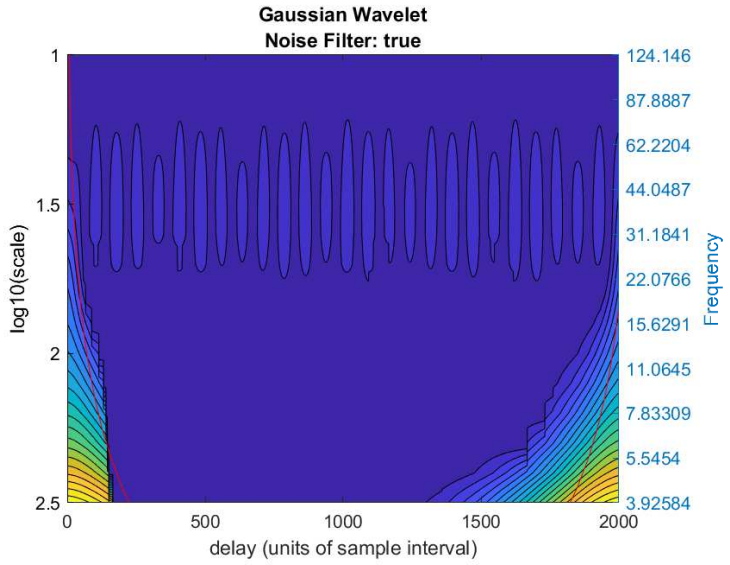


Figure 6:
Gaussian WT of output signal with noise filter. Note the high time resolution of frequency space peaks situated around $44Hz$ and corresponding low amplitude.

## Conclusion

Detection of time and frequency localization of a test signal are displayed with clarity by the WT process. Through analysis of three wavelets, tradeoff between frequency-space and time-space resolution is linked to time-oscillation within the chosen wavelet function. Increased oscillation leads to a greater frequency resolution at the expense of time resolution.

Noise filtering provides a method of statistical significance testing and may be used to disgard features lying below that expected from random generation of a chosen noise color. Cone of Influence comparison further allows for the removal of erroneous results inherent in the WT process.

# References

[1] Rowell, G 2020, *Data Analysis and Modelling Course Notes*, Data Analysis and Modelling, University of Adelaide

[2] Rowell, G 2020, *Data Analysis and Modelling Assignment 3 Guidelines*, Data Analysis and Modelling, University of Adelaide

[3] C. Torrence, G. Compo, 1998, *A Practical Guide to Wavelet Analysis*, Program in Atmospheric and Oceanic Sciences, University of Colorado, Boulder, Colorado

[4] Pritchard, J 2020, *A Practical Guide to Fourier Data Techniques*, Data Analysis and Modelling Assignment Submission, University of Adelaide, Adelaide, Australia

# Appendix

Following is the MATLAB software used in the generation of all plots included in this report. FT generation code is included as a reference[4].

# Table of Contents



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

# USER SETTINGS

```
%Suppress figure rendering
set(0,'DefaultFigureVisible','off')

%Import data
data = importdata('BBChaos_145.txt');
y = data(:,2);

%Set f_s
input_Hz = 145;
input_period = 1/(input_Hz);
fsamp = Calculate_sample_rate(data(:,1), input_period, 16);

%Set decimation factor
d = 4;

%Define Wavelets

%Mexican Hat
MH.Func = @(t, t0, s) (1 - ((t-t0)./s).^2).*exp(-0.5*((t-t0)./s).^2);
MH.Norm = @(s, Y) (2*pi*s)*Y;
MH.S_to_f = @(fs, s) (fs) .* ( ( 2*pi / sqrt(2.5) ) .* s ).^(-1);
MH.name = "Mexican-Hat";
MH.coi = @(scale) sqrt(2)*scale;

%Gaussin Wavelet
GW.Func = @(t, t0, s) pi^(-0.5) * exp(-0.5*((t-t0)./s).^2);
GW.Norm = @(s, Y) (2*pi*s)*Y;
GW.S_to_f = @(fs, s) (fs) .* ( ( 2*pi / sqrt(0.5) ) .* s ).^(-1);
GW.name = "Gaussian";
GW.coi = @(scale) sqrt(2)*scale;

%Morlet Wavelet - w0 must satisfy admissability critereon
w0 = 6;
C = (1 + exp(-w0^2) - 2*exp((-3/4)*w0^2))^(-1/2);
MW.Func = @(t, t0, s) C *(1/(pi^0.25)) .* exp(1i*w0.* ((t-t0)./s) ) .*
 exp(- ((t-t0)./s).^2 / 2);
MW.Norm = @(s, Y) (2*pi*s)*Y;
MW.S_to_f = @(fs, s) (fs) .* ( (4*pi / (w0 + sqrt(2 + w0^2))) .*
 s ).^(-1);
```

```
MW.name = "Morlet";
MW.coi = @(scale) sqrt(2)*scale;

%Paul Wavelet
PW.Func = @(t, t0, s) ((-8)/(sqrt(pi*24))) .* (1 - 1i* ((t-t0)./
s) ).^3;
PW.Norm = @(s, Y) (2*pi*s)*Y;
PW.S_to_f = @(fs, s) (fs) .* ( ( 4*pi/5 ) .* s ).^(-1);
PW.name = "Paul";
PW.coi = @(scale) scale*sqrt(2);

%Set Wavelet Function
WF = GW;

%Choose if noise filter applied
filter = true;
```

# SOFTWARE ANALYSIS

```
%Decimate and truncate data
%Improves quality of WT
%WT is best for short time data sets
y = y(1:d:2^13);

fSamp = fsamp/d;

Pyy = (abs(fft(y))).^2;
Y = fft(y);
stop = length(Pyy);

%Set time axis
t = 1:stop;
t0 = stop/2;

%Set scale array as log base 10
%Max at 2.5 for MW and MH
x = -1:0.1:2.5;
scale = 10.^x;

%Perform Transform
WT = W_T(scale, t, t0, Y, stop, WF);

%Apply a Noise Filter if filter==true
if(filter)
    r = std(y)*randn(1,length(y));
    R = fft(r);
    WTN = W_T(scale, t, t0, R', stop, WF);
    %Set signal above noise to be zero
    WT = Filter(WT, WTN, scale, stop);
end

%Plot
Plot_WT(t, x, scale, WT, fSamp, WF, filter);
```

# DEPENDENCIES

```
function Plot_WT(t, x, scale, WT, fSamp, WF, filter)
%Plot a Wavelet data set WT

    FIG = figure;

    %Final integer specifies the number of contours to render
    contourf(t, x, abs(WT),20);
    axis ij;
    hold on;

    %Define and plot Cone of Influence
    coi = WF.coi(scale);
    t0 = t(1:length(coi));
    plot(scale,log10(coi),'r');
    plot(max(t) - scale, log10(coi), 'r');

    %Set labels and axis format
    xlabel('delay (units of sample interval)')
    ylabel('log10(scale)')
    axis([0 2000 1 2.5])
    yyaxis right

    u = 1: 1.5/10 : 2.5;
    scale = 10 .^ u;
    x = WF.S_to_f(fSamp, scale);
    yticklabels(x);
    ylabel('Frequency');
    axis ij;

    title({WF.name + ' Wavelet', "Noise Filter: " + filter});

    %hold off

    %Save result
    saveas(FIG, WF.name + filter, 'epsc');
end

function result = W_T(scale, t, t0, Y, stop, WF)
%Produce the WT of Y using Wavelet Function WF
    WT = zeros(length(scale),stop);

    for ii = 1:length(scale)
        %Pick out scale scalar value
        s = scale(ii);

        %Set wavelet function
        w = WF.Func(t, t0, s);

        %Swap 1st and 2nd half of wavelet function in prep. for ifft
        %Only select 1:stop # of points
        w = [w(stop/2:stop) w(1:(stop/2-1))];
```

```
        %Normalize wavelet function wrt s
        W = WF.Norm(s, fft(w));

        %Calculate time domain for current scale value
        WT(ii,:) = ifft(conj(W').*Y);
    end
    result = WT;
end


function result = Filter(WT, WTN, scale, stop)
%Set all WT value below WTN to ~zero
    for ii = 1:length(scale)
        for jj = 1:stop
            if abs(WT(ii,jj)) < abs(WTN(ii,jj))
                WT(ii,jj) = eps;
            end
        end
    end
    result = WT;
end
```

*Published with MATLAB® R2018b*