SIGNAL PROCESSING PROBLEMS, SOLVED IN MATLAB AND IN PYTHON

Udemy Hosted Course: https://www.udemy.com/course/signal-processing/

Instructor: Mike X Cohen: http://sincxpress.com/

Notes and code completion (student): Ron Fredericks: http://BiophysicsLab.com/

Course-work: December 12, 2019 to January 30, 2020 **Notes last updated: Wednesday, May 20, 2020**

TABLE OF CONTENTS

SECTION 1: INTRODUCTION

SIGNAL PROCESSING = DECISION-MAKING + TOOLS

Tools:

- Filters
- Convolution
- Wavelets
- Spectra (FFT)
- Time-frequency
- Cleaning/denoising
- Resampling
- Interpolating
- Feature detection
- SNR/RMS

Improve scientific programming with MATLAB and Python, and signal processing

MATLAB

MATLAB more comfortable for serious signal applications

OCTAVE

https://octave-online.net/

(no online help, problems with .mat loads, not too useful – maybe desktop version is better)

PYTHON / JUPYTER

Running Python online using Jupyter – not too useful, seems unstable. But desktop Anaconda version works well

https://jupyter.org/try

https://notebooks.gesis.org/binder/jupyter/user/ipython-ipython-in-depth-f20pi40o/notebooks/binder/Index.ipynb

HAVING FUN WITH FILTERED GLASS DANCE

D:\Ron\Documents\Ron Fredericks Consulting\Education\Signal processing problems - Udemy\Section 01\sigproc-glassDance

MATLAB

Code: sigprocMXC_filterGlass.m

Data: glassDance.mat

Data reference: Philip Glass, Dance VII (https://www.youtube.com/watch?v=LpewOlR-z_4)

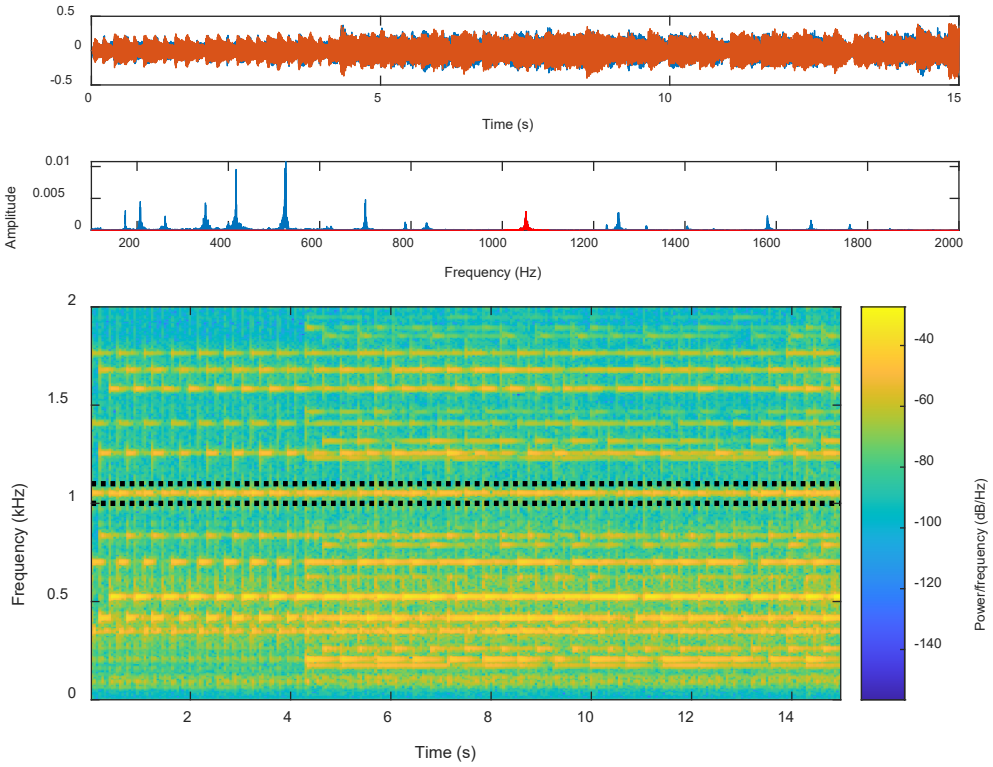
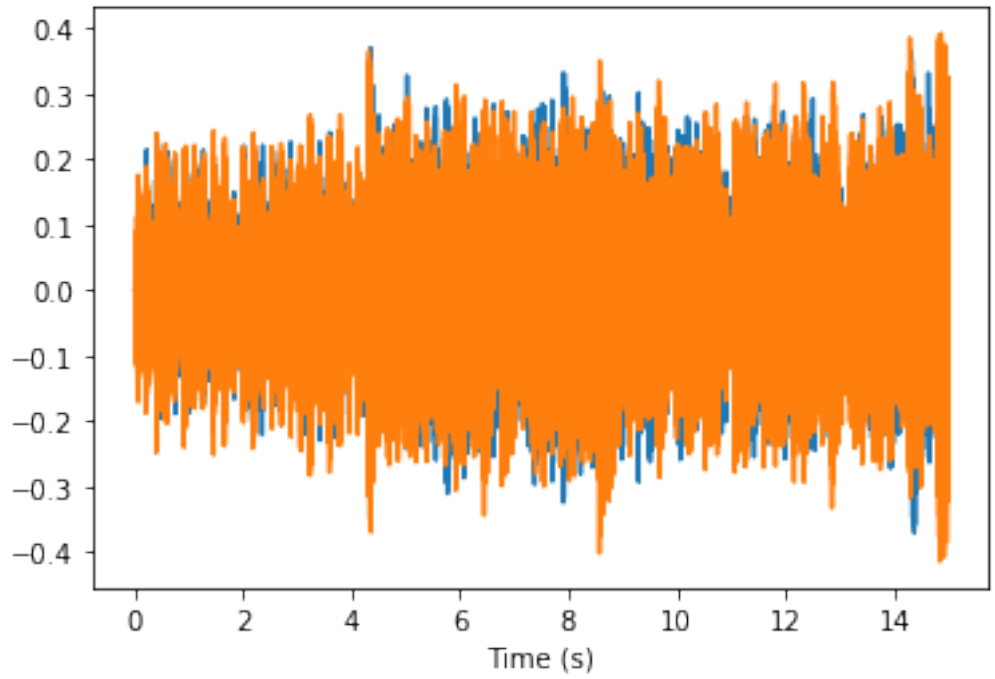
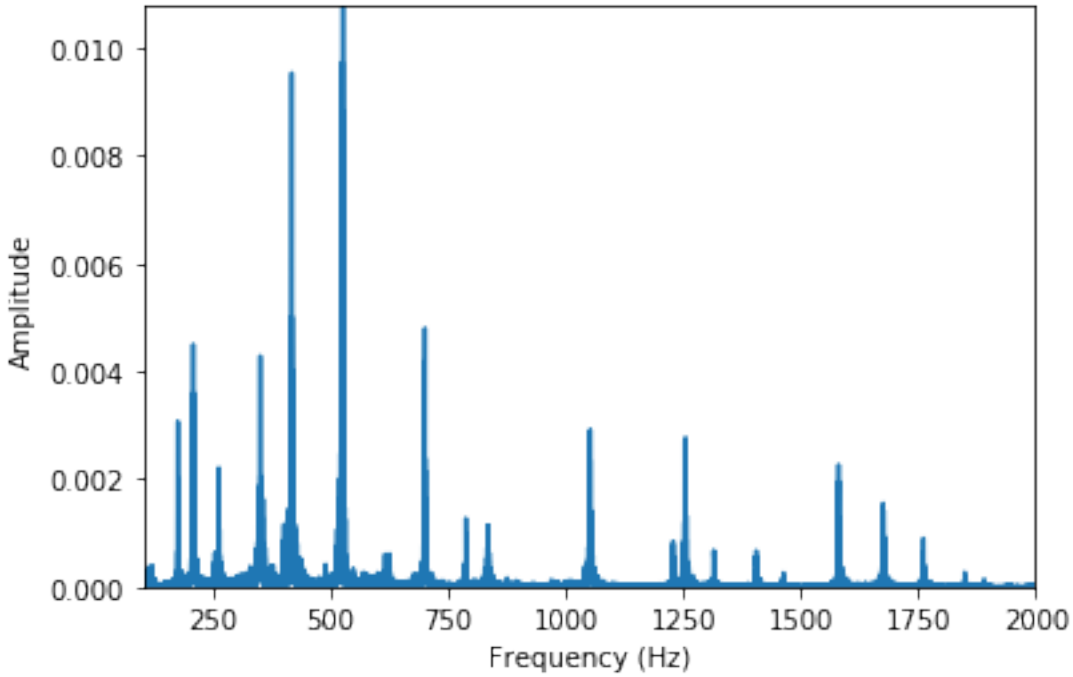


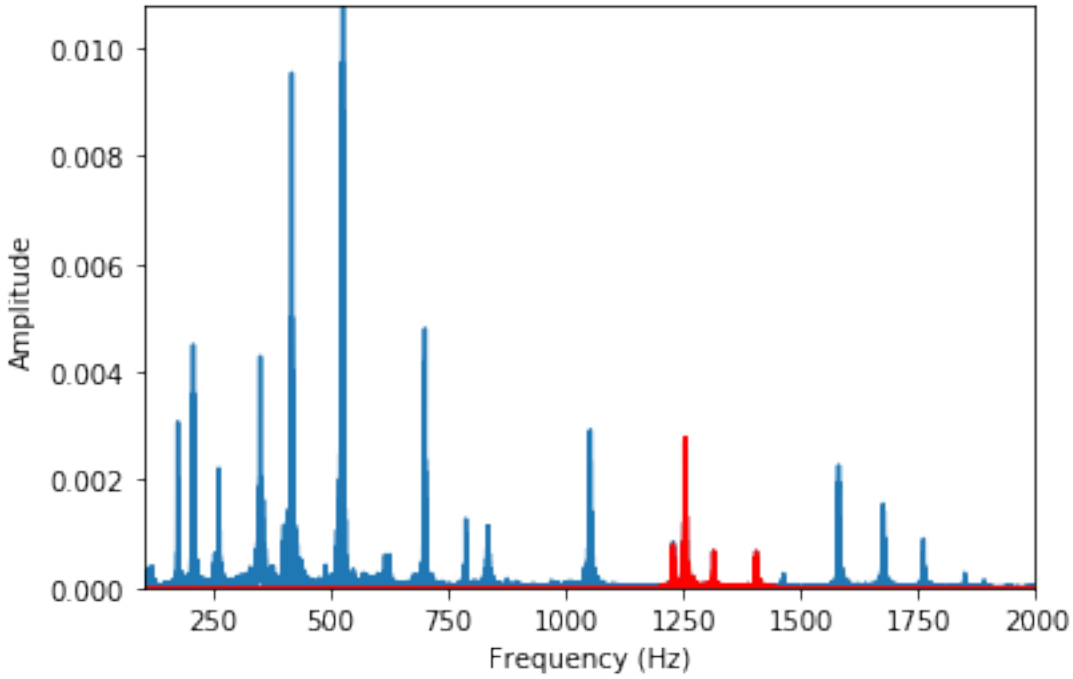
Figure: Time-domain signals (top); static power spectrum, filtered signal power spectrum (middle); time-frequency response (bottom)

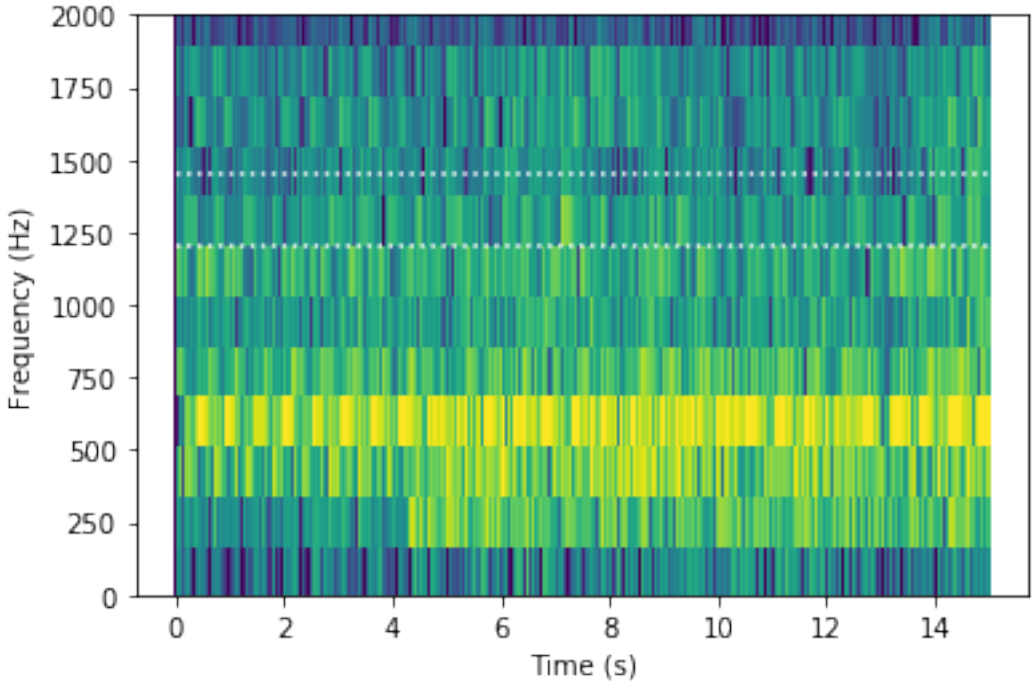
PYTHON NOTEBOOK – JUPYTER

sigprocMXC_filterGlass.ipynb


0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0
2
4
6
8
10
12
14
Time (s)


0.010
0.008
0.006
0.004
0.002
0.000
Amplitude
250
500
750
1000
1250
1500
1750
2000
Frequency (Hz)





WRITING CODE VS USING TOOLBOXES/PROGRAMS

A mix of our own code + a limited number of ToolBoxes seems to be the best answer

SECTION 2: TIME SERIES DENOISING

Code directory: D:\Ron\Documents\Ron Fredericks Consulting\Education\Signal processing problems - Udemy\Section 02\sigprocMXC-TimeSeriesDenoising

9. MEAN-SMOOTH A TIME SERIES

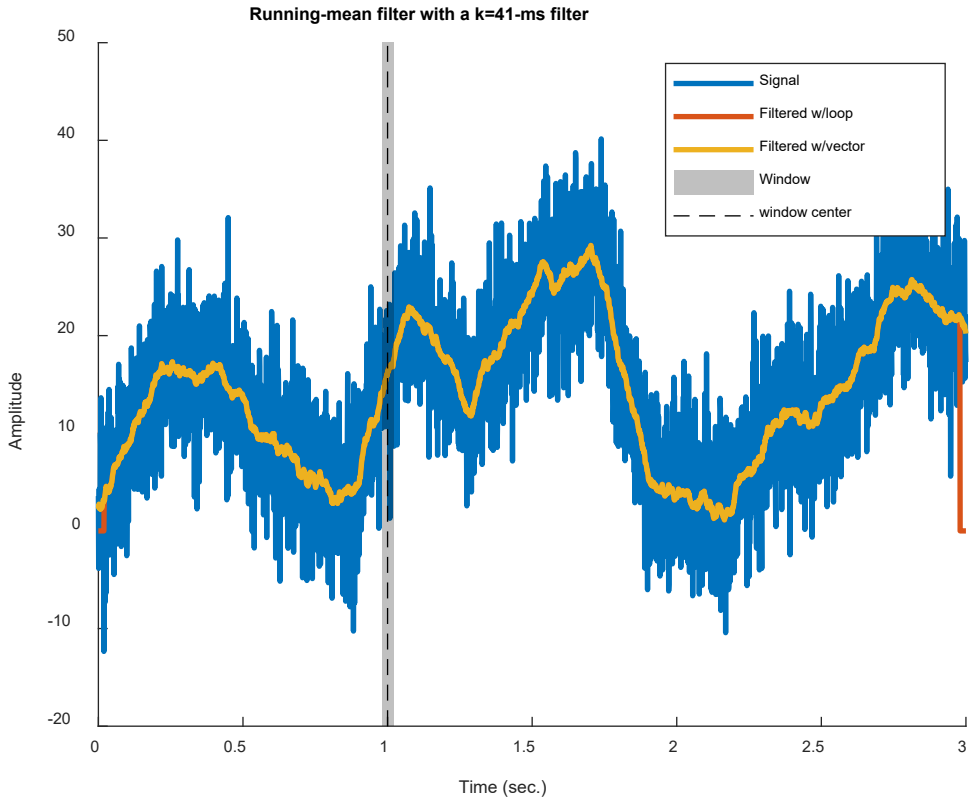
$$y_t = (2k+1)^{-1} \sum_{i=t-k}^{t+k} x_i$$

Edge effects whenever a temporal filter is applied

Filter is useful when noise is distributed positive and negatively with respect to signal of interest

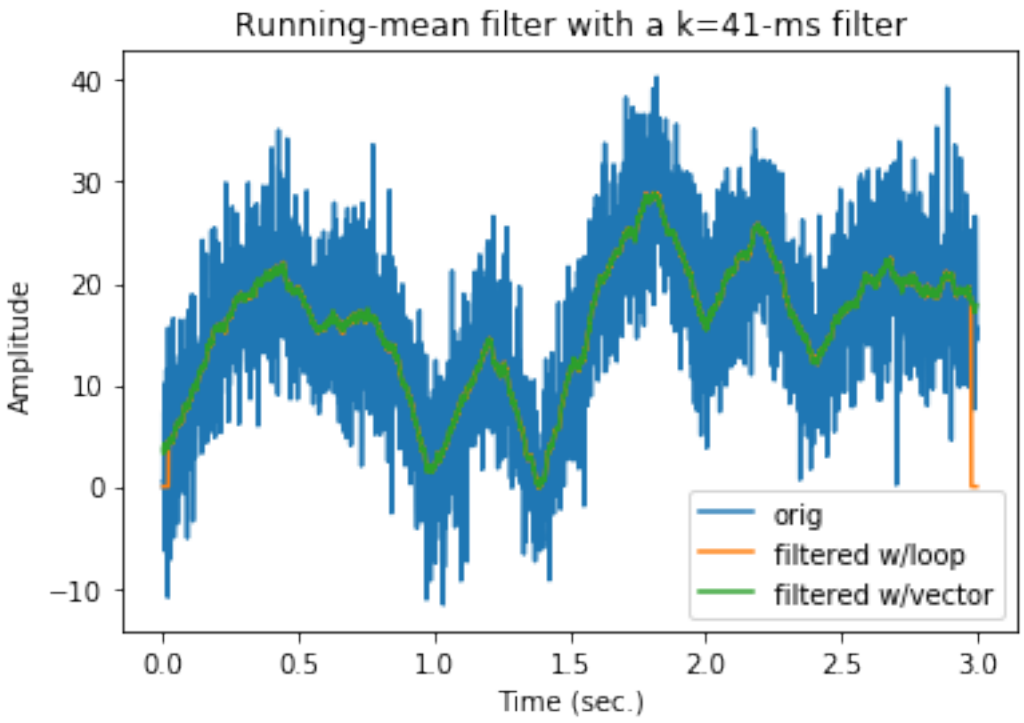
MATLAB

Code: sigprocMXC_mean_smooth.m



PYTHON / JUPYTER

sigprocMXC_timeSeriesDenoising.ipynb



10. GAUSSIAN-SMOOTH A TIME SERIES

$$y_t = \sum_{i=t-k}^{t+k} x_i \, g_i$$

Similar to the means-smoothing filter: each data point was replaced by previous and following data points

Weight previous and following time points according to a Gaussian function

Sum of Gaussian function is 1, starts at 0, ends at 0

$$g = e^{\frac{-4\ln(2)t^2}{w^2}}$$

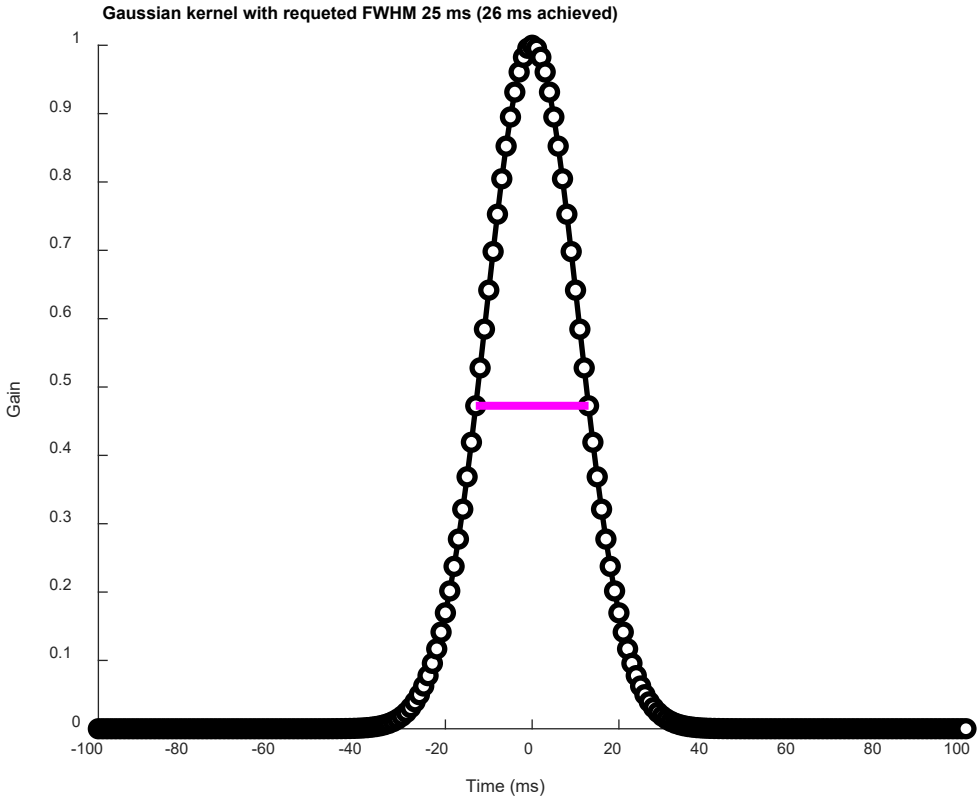
Where w is the full-width at half maximum (FWHM)

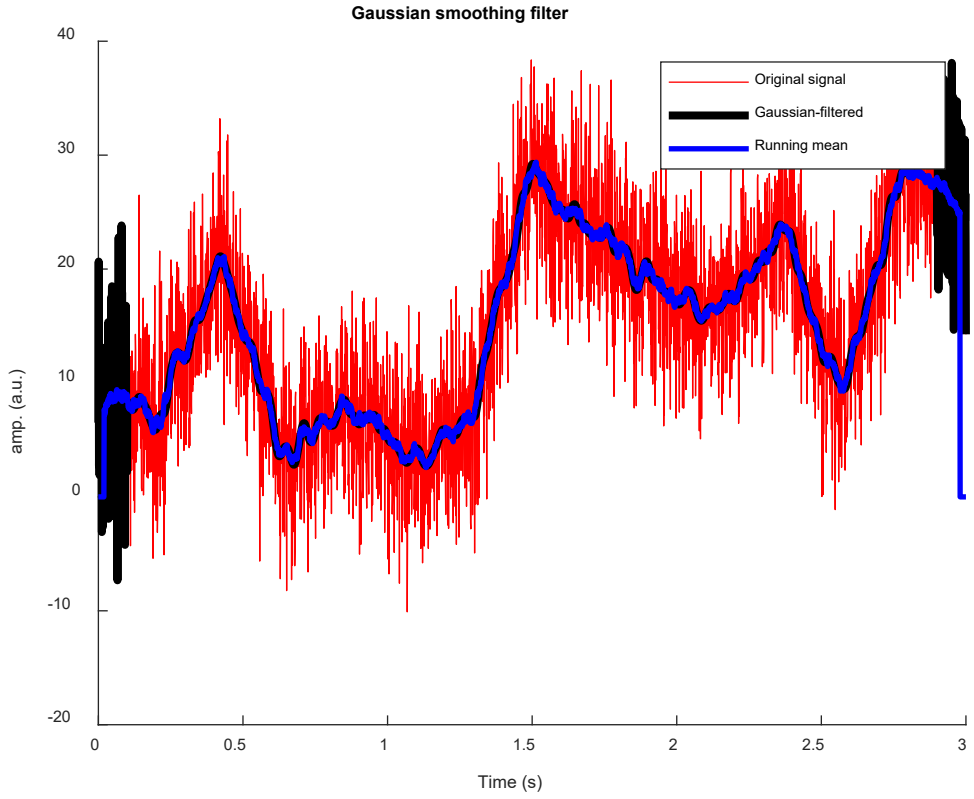
For example, set w to 10 if you want 10 ms of smoothing

Important to normalize the Gaussian window to unit energy

MATLAB

Code: sigprocMXC_Gaussian_smooth.m


Gaussian kernel with requeted FWHM 25 ms (26 ms achieved)
Gain
Time (ms)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-100
-80
-60
-40
-20
0
20
40
60
80
100


Gaussian smoothing filter
amp. (a.u.)
Time (s)
Original signal
Gaussian-filtered
Running mean
40
30
20
10
0
-10
-20
0
0.5
1
1.5
2
2.5
3

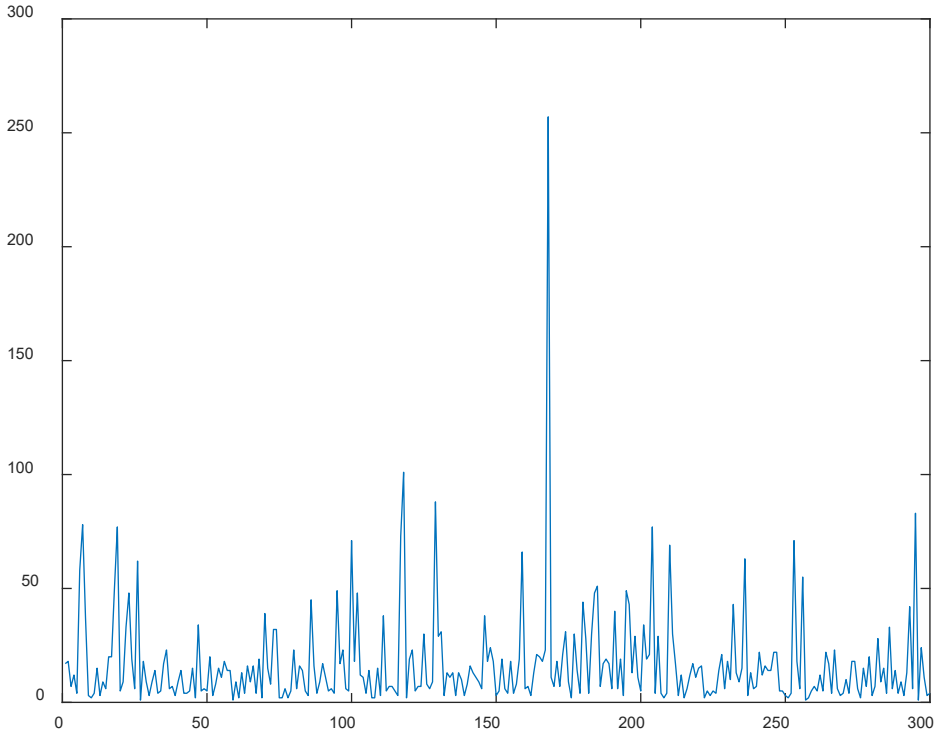
11. GAUSSIAN-SMOOTH A SPIKE TIME SERIES

MATLAB

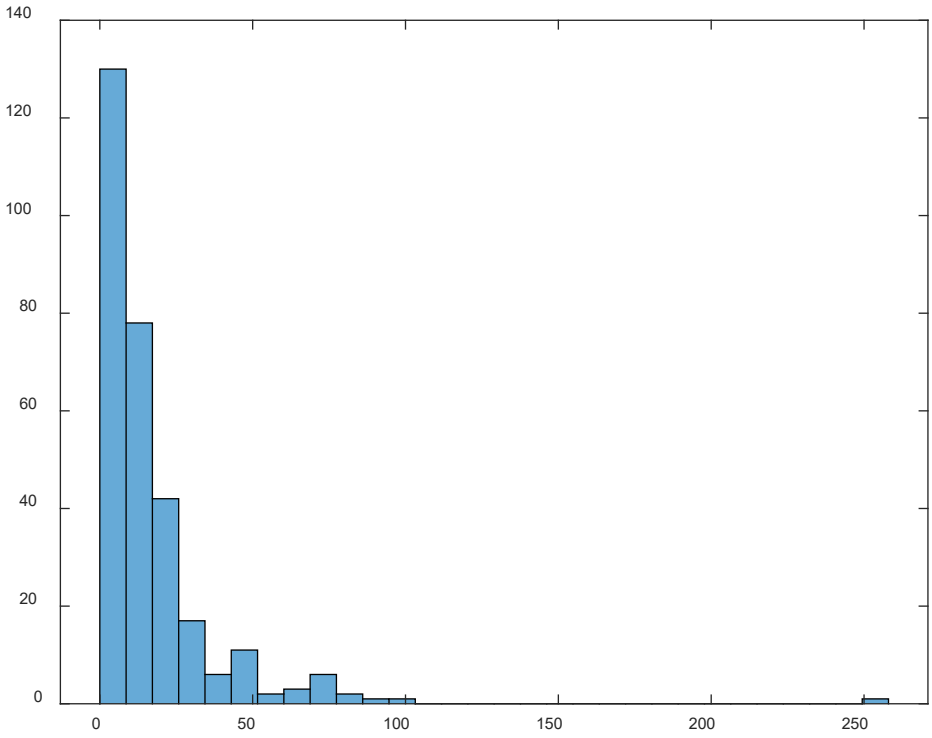
Code: sigprocMXC_GauSmoothSpikes.m

Inter-spike intervals (isi) – exponential distribution for bursts

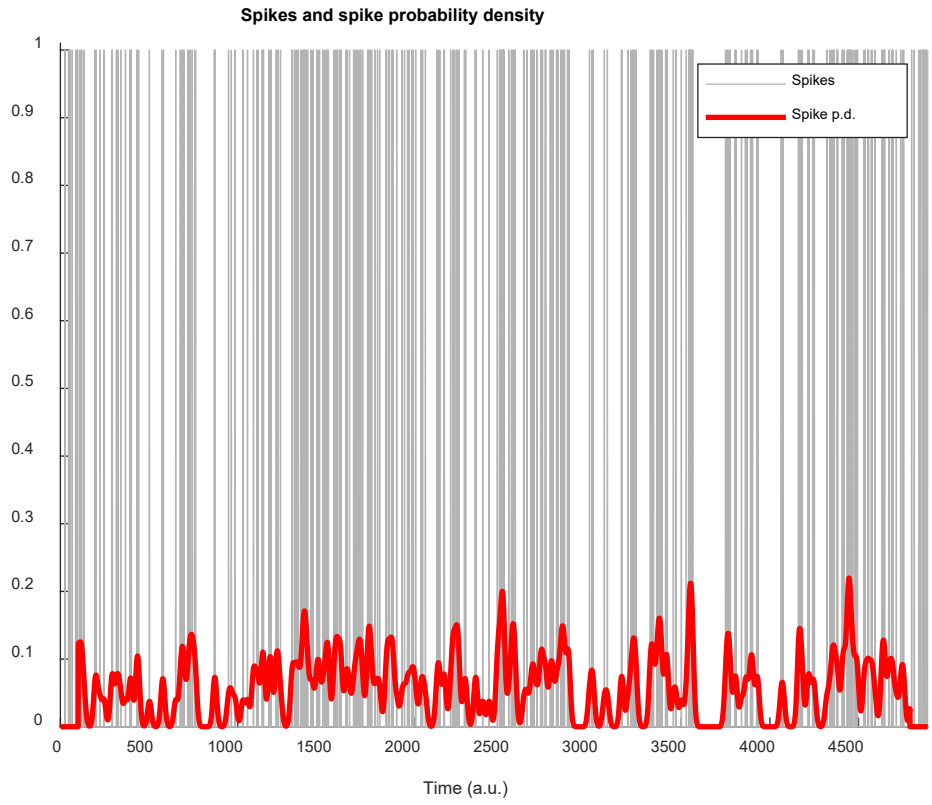
>> plot(isi)



>> histogram(isi, 30) % Note: hist() is now obsolete




```
>> % plot the filtered signal (spike probability density)

>> plot(filtsigG,'r','linew',2)
```



12. DENOISING EMG SIGNALS VIA TKEO

Teager-Kaiser Energy-tracking Operator

$$y_t = x_t^2 - x_{t-1}x_{t+1}$$

Used for Denoising EMG (Electromyogram) signals to determine when a muscle was initiated

TKEO smooths signals for computer analysis by suppressing noise and amplifying signal (improves SNR)

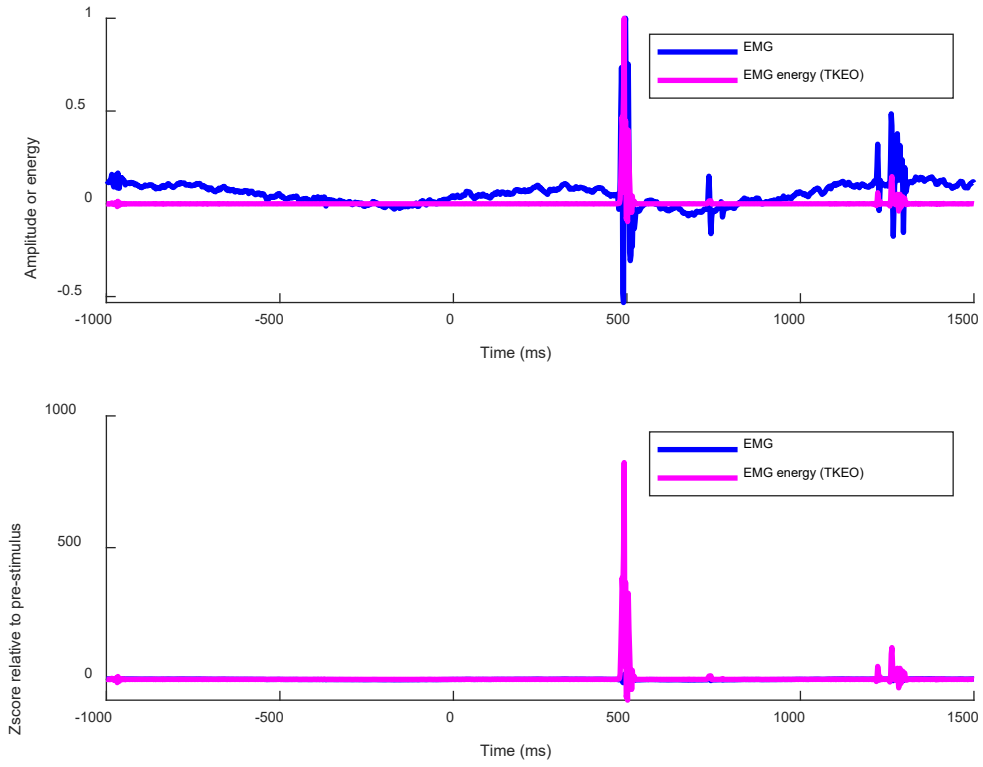
y_t results are in a new unit of energy (microvolts squared) compared to x_t (microvolts). Convert original and filtered EMG results to a z-score to compare them.

$$z_i = \frac{x_i - \bar{x}}{S}$$

Collect mean and standard deviation on raw data before time=0 during the baseline, so signal events are not included.

MATLAB

Code: sigprocMXC_TKEO.m



13. MEDIAN FILTER TO REMOVE SPIKE NOISE

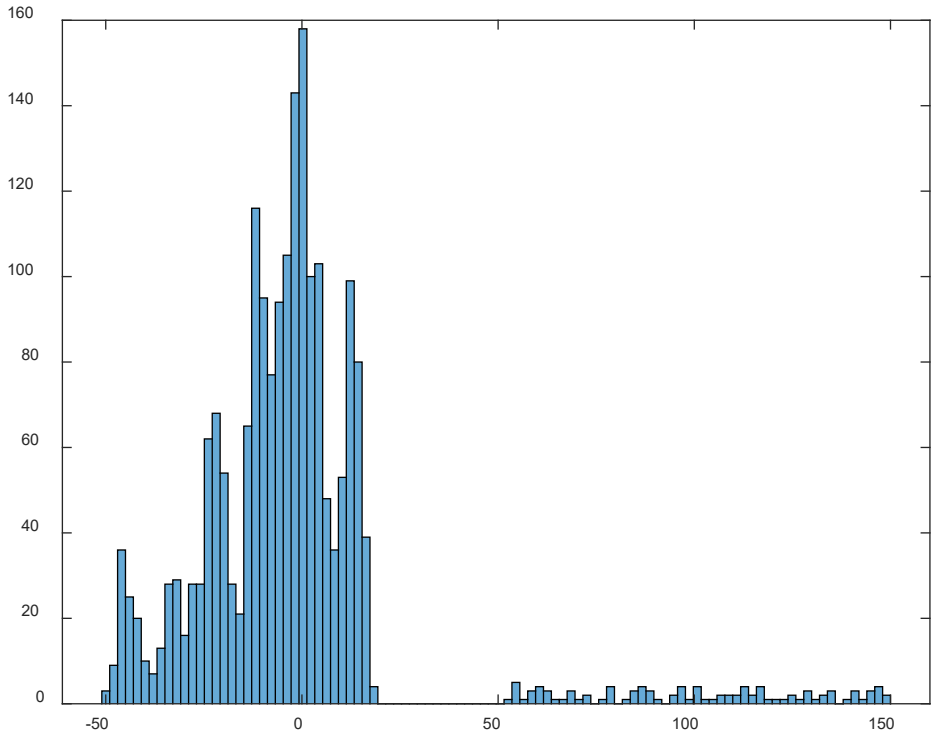
Compute the median to avoid “outliers”:

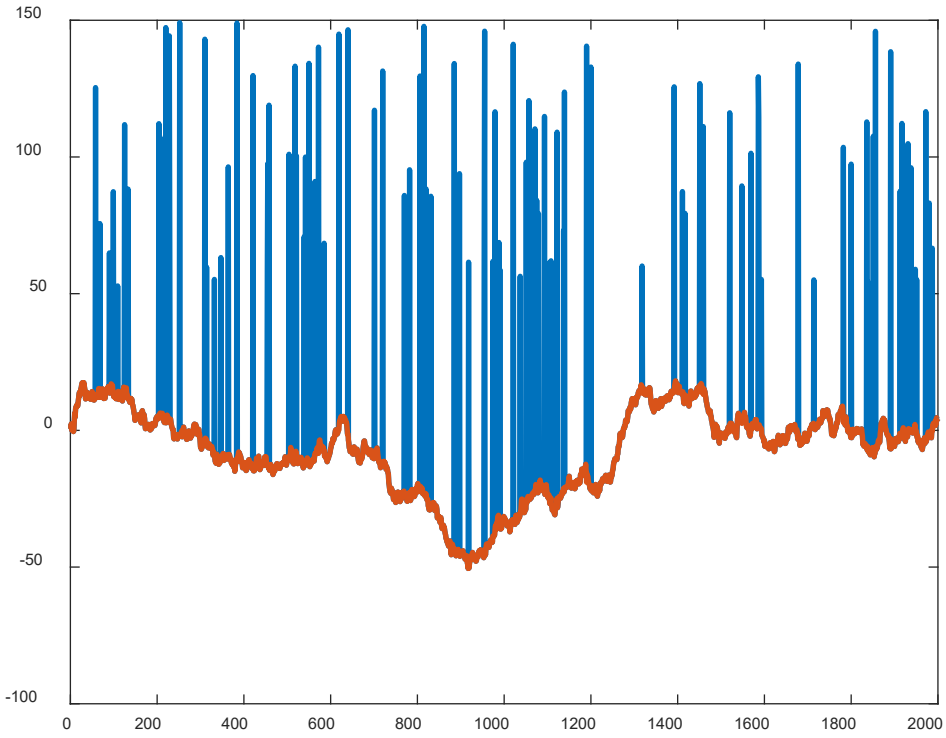
- Sort
- Median = middle value (odd dataset)
- Median = average 2 middle values

Non-linear filter – apply to selected data points, not all data points.

MATLAB

Code: sigprocMXC_median_filter.m

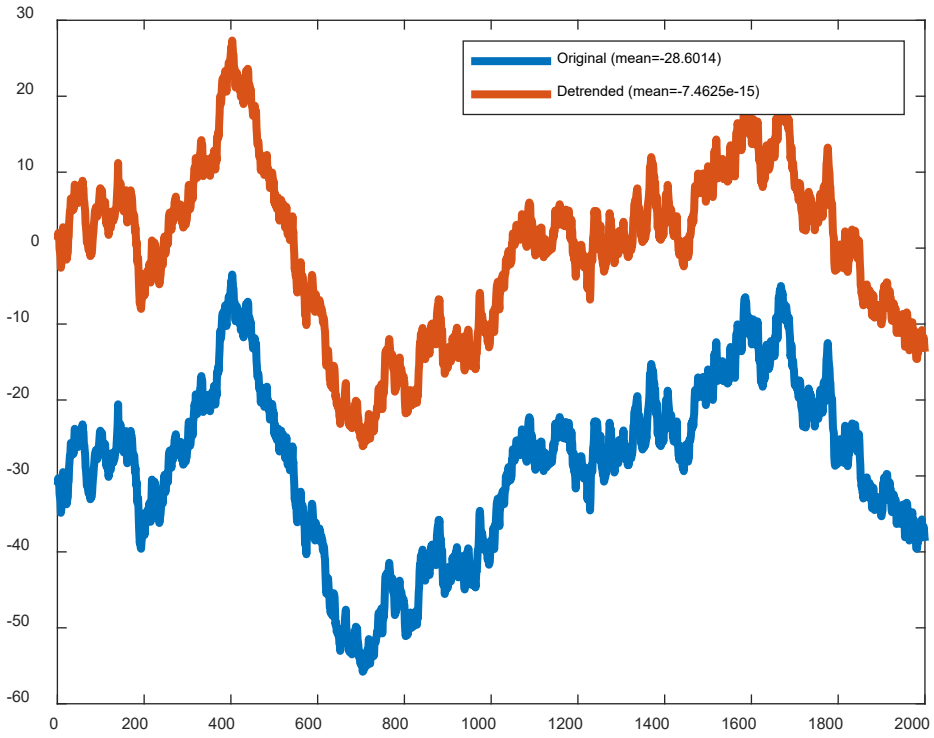

160
140
120
100
80
60
40
20
0
-50
0
50
100
150



14. REMOVE LINEAR TREND (DETRENDING)

MATLAB

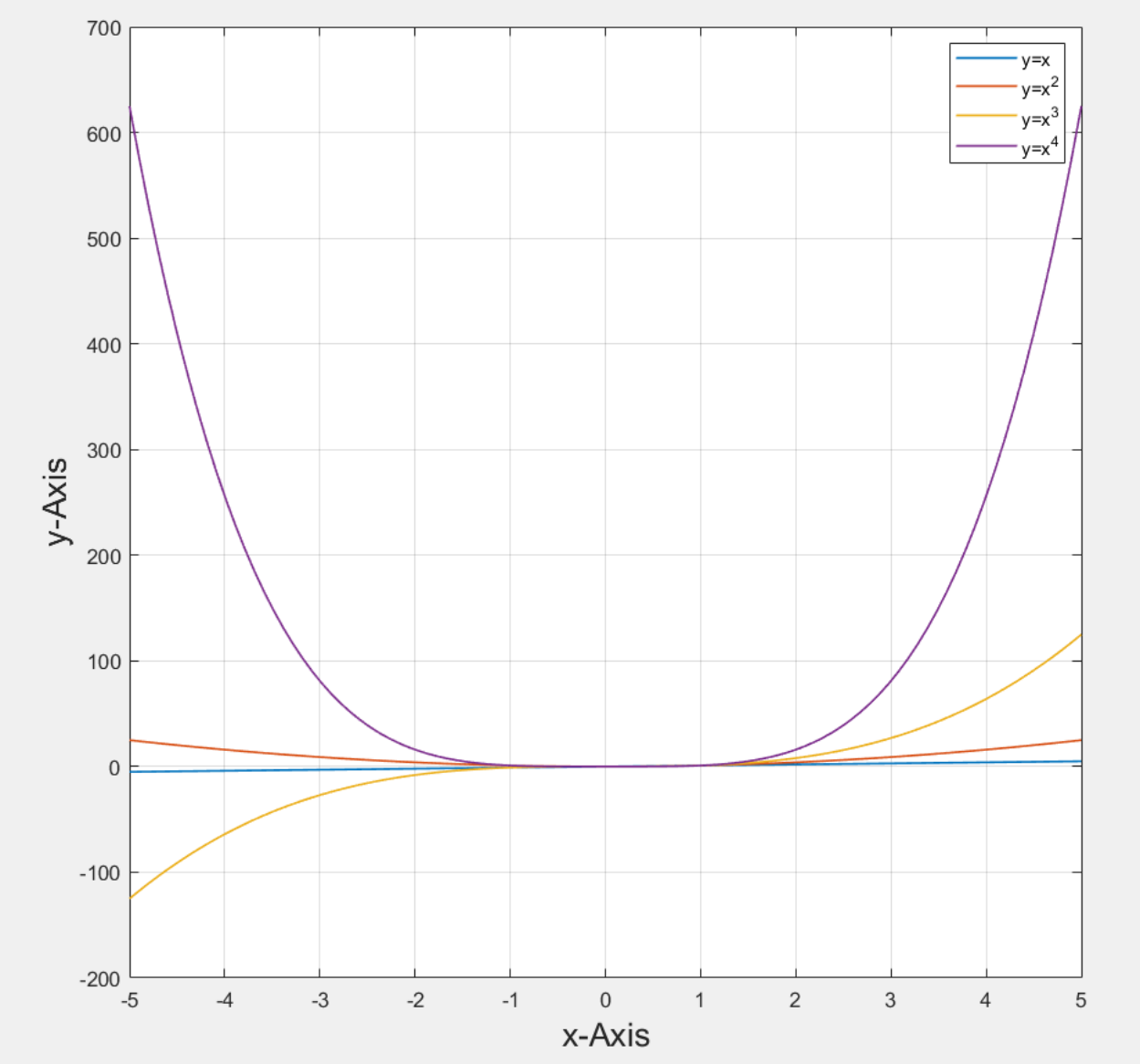
Code: sigprocMXC_detrend.m



15. REMOVE NONLINEAR TREND WITH POLYNOMIALS

Examples of polynomials

$$\beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n$$



Function of x with coefficients with each x a higher power – an n'th order polynomial

Bayes Information Criterion

$$b = n\ln(\epsilon) + k\ln(n)$$

$$\epsilon = n^{-1}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Where:

n: Data points

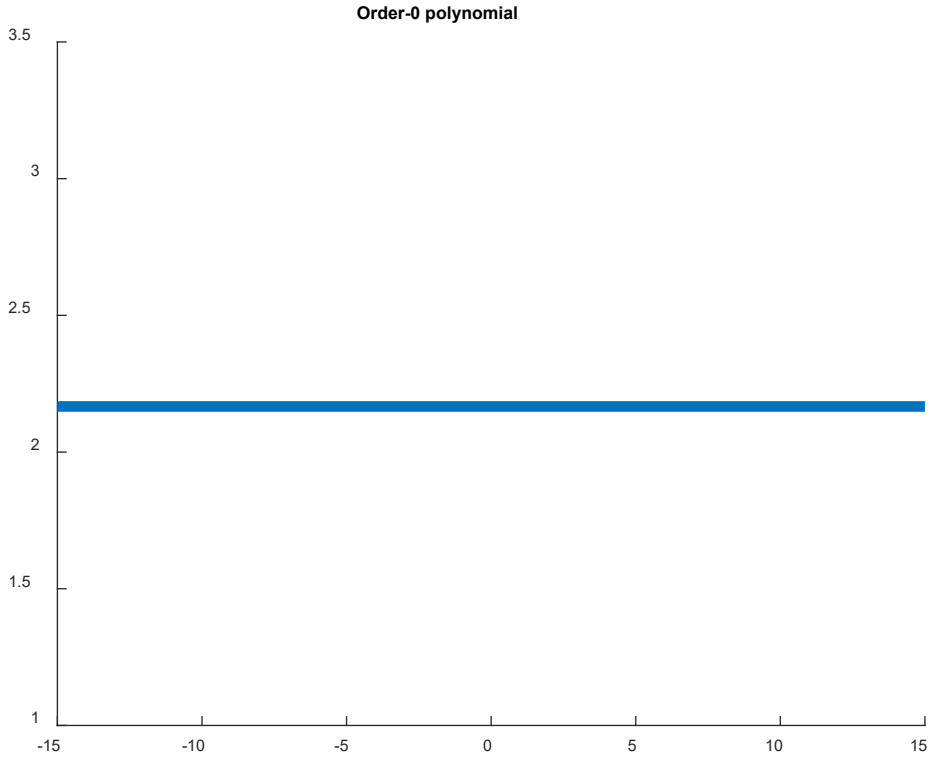
k: Parameters

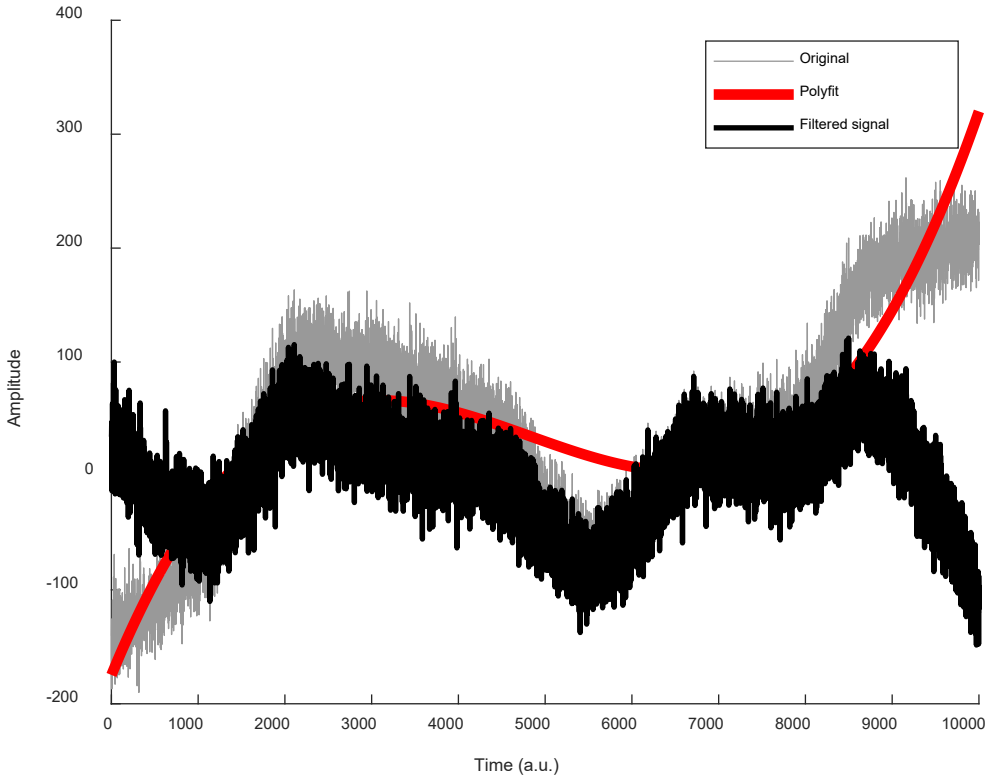
$\hat{y}_i$: Predicted data

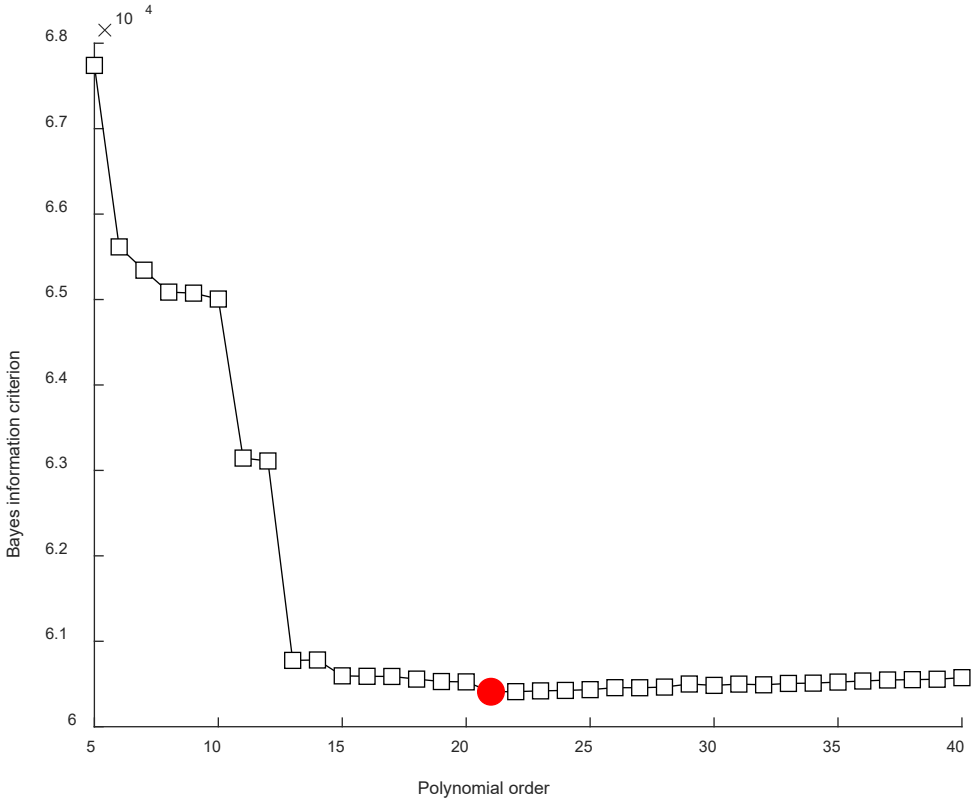
y_i: data

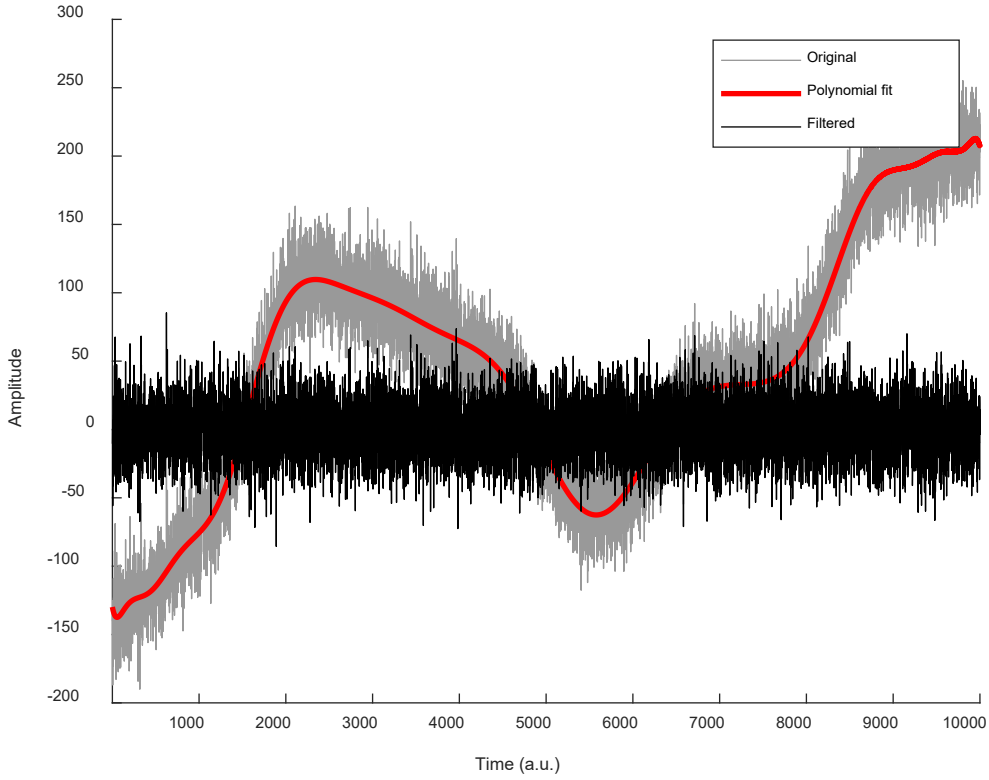
MATLAB

Code: sigprocMXC_polynomialDetrend.m




Original
Polyfit
Filtered signal
Amplitude
Time (a.u.)
400
300
200
100
0
-100
-200
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

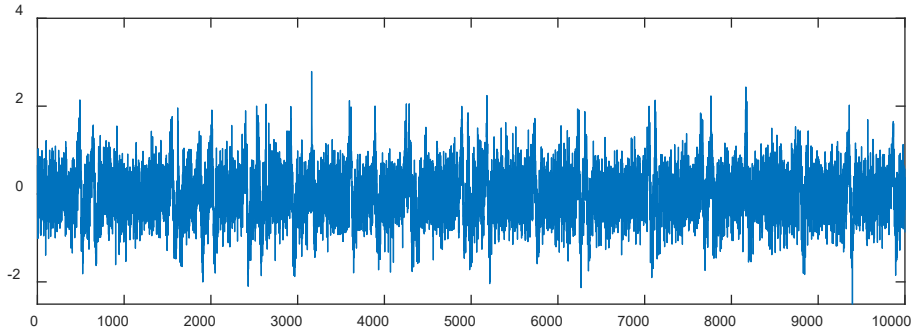

×10^4
Bayes information criterion
Polynomial order
6.8
6.7
6.6
6.5
6.4
6.3
6.2
6.1
6
5
10
15
20
25
30
35
40

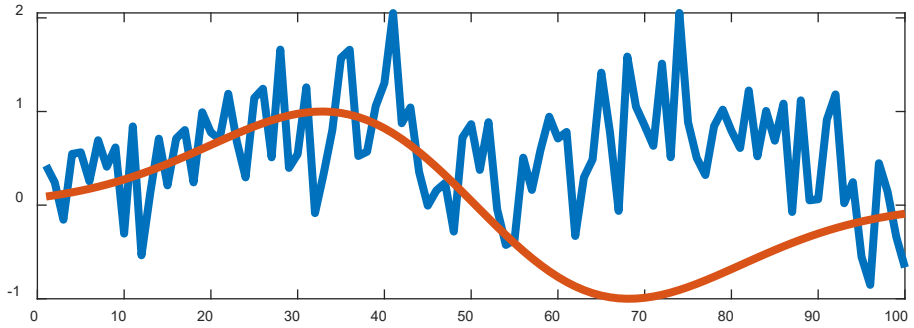


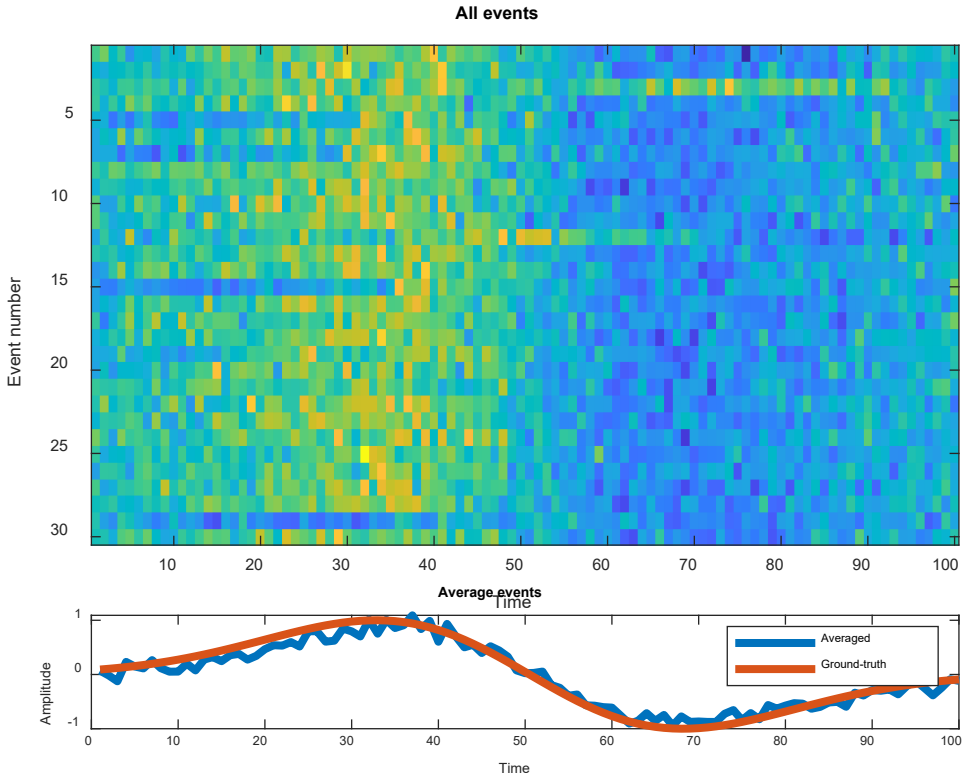
16. AVERAGING MULTIPLE REPETITIONS (TIME-SYNCHRONOUS AVERAGING)

MATLAB

Code: sigprocMXC_averaging.m


4
2
0
-2
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


2
1
0
-1
0
10
20
30
40
50
60
70
80
90
100


All events
Event number
5
10
15
20
25
30
10
20
30
40
50
60
70
80
90
100
Time
Average events
Amplitude
1
0
-1
0
10
20
30
40
50
60
70
80
90
100
Time
Averaged
Ground-truth

$$\beta = (X^T X)^{-1} X^T y$$

$$\gamma = y - X\beta$$

Where:

β: Regression weights

X: Design matrix

y: Data

γ: Residual

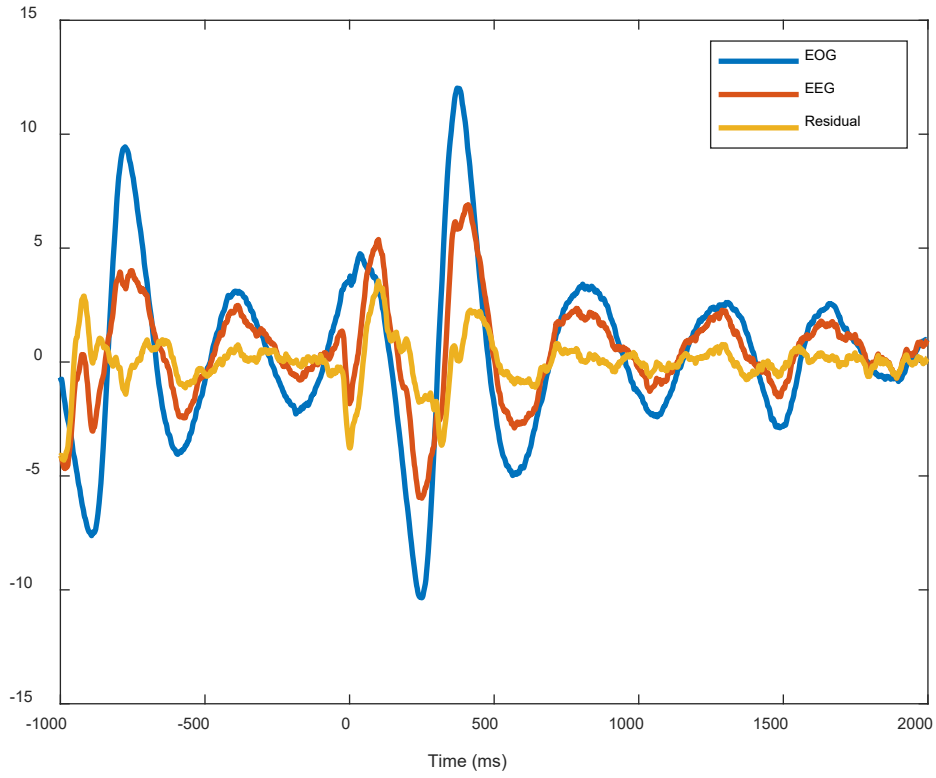
$X\beta$: Predicted data

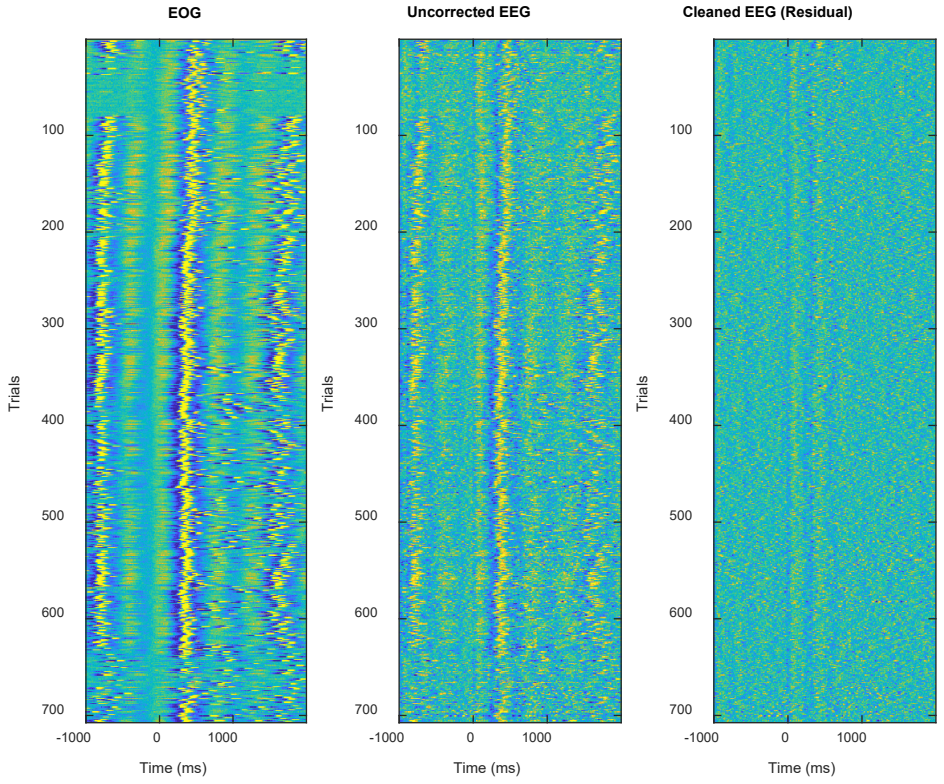
See Udemy sincxpress.com's Linear Algebra course to learn more about these symbols

https://www.udemy.com/course/linear-algebra-theory-and-implementation/

MATLAB

Code: sigprocMXC_template_projection.m

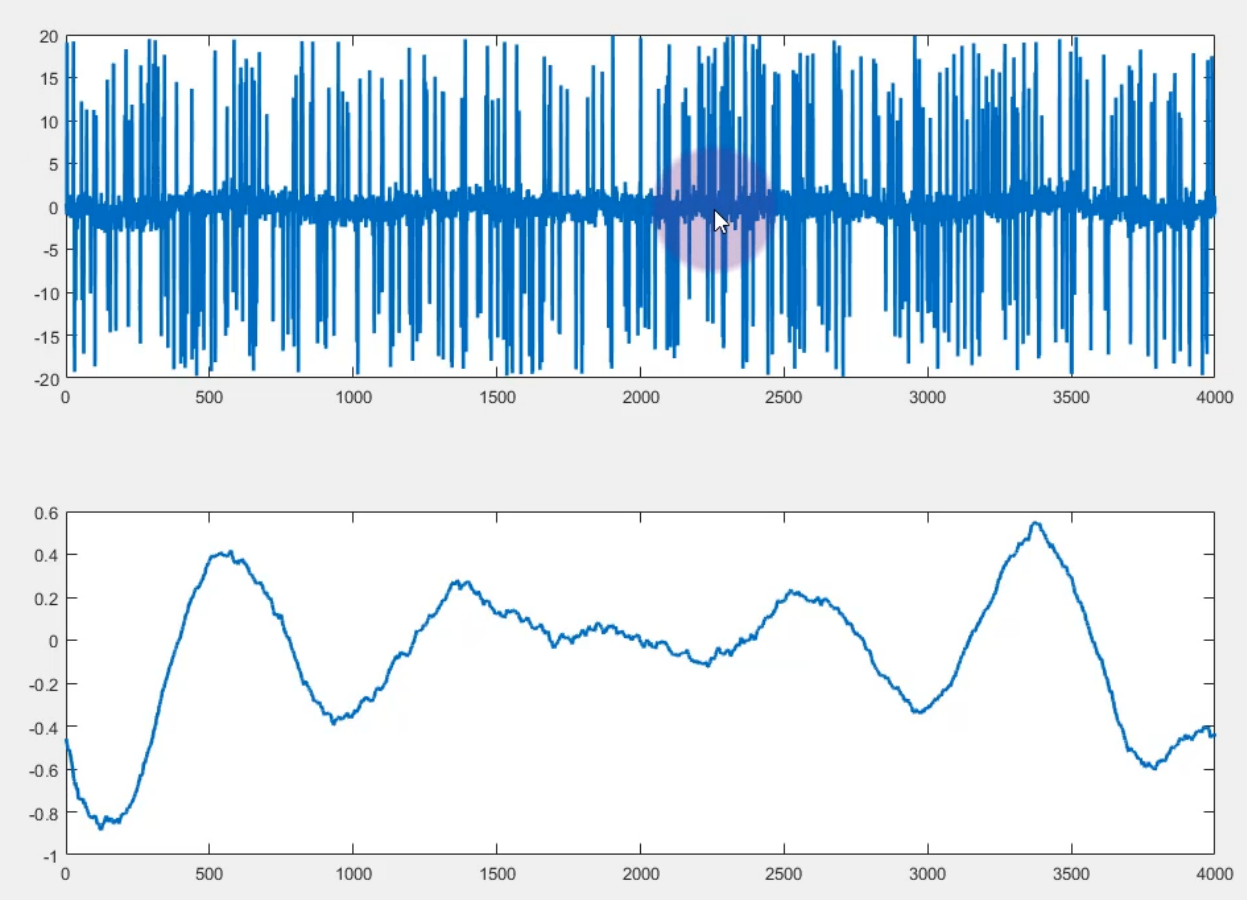

EOG
EEG
Residual
15
10
5
0
-5
-10
-15
-1000
-500
0
500
1000
1500
2000
Time (ms)


EOG
Uncorrected EEG
Cleaned EEG (Residual)
Trials
100
200
300
400
500
600
700
-1000
0
1000
Time (ms)

18. CODE CHALLENGE: DENOISE THESE SIGNALS

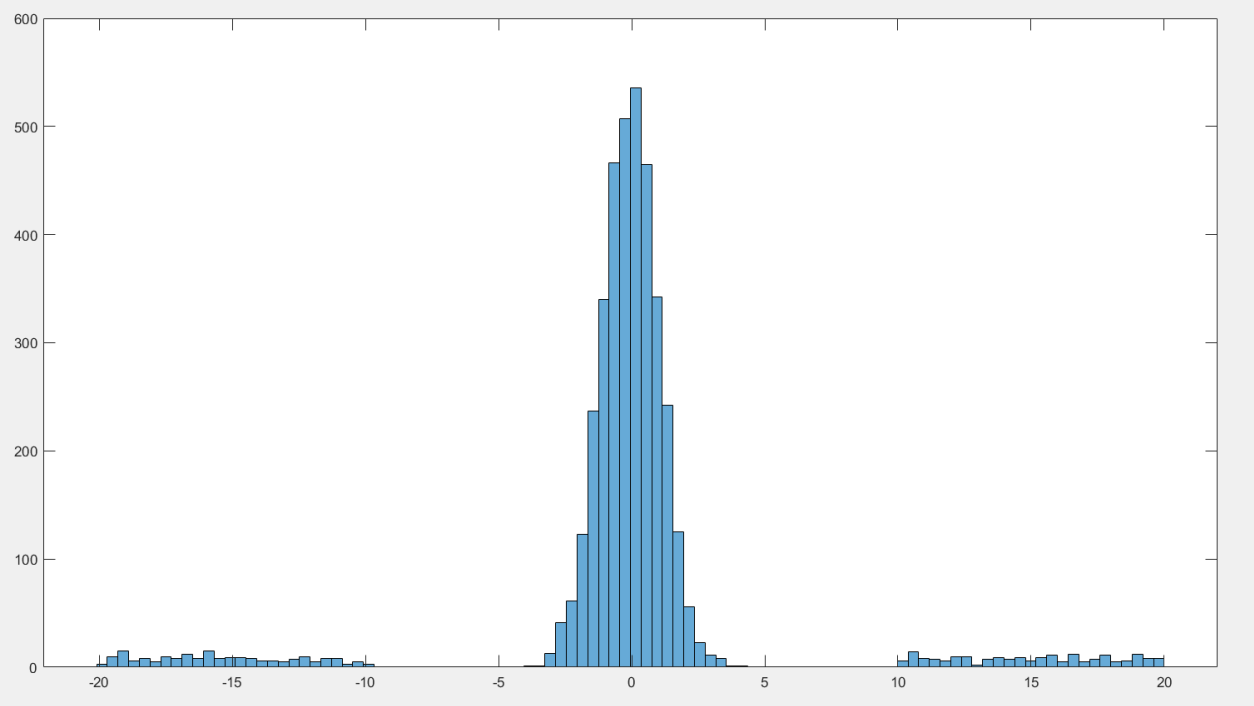
DESIRED RESULTS

Initial data: denoising_codeChallenge.mat



MY RESULTS

My script: sigprocMXC_challenge.m



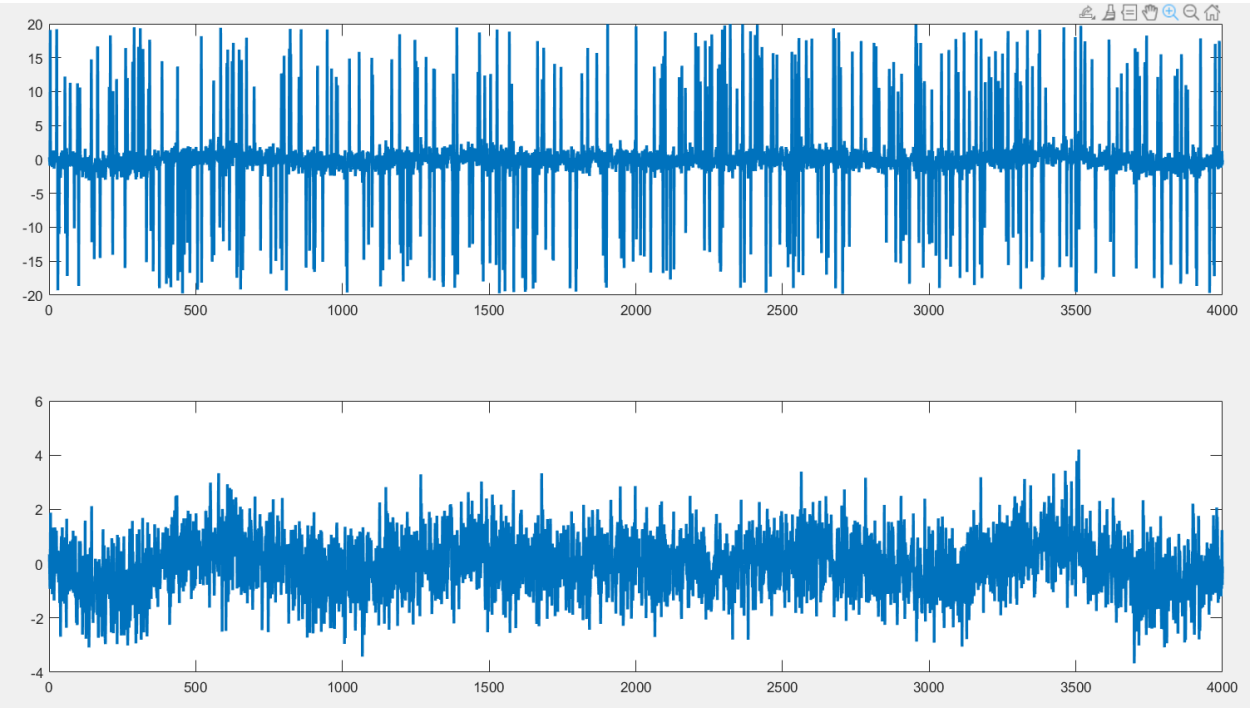


Figure: Step 1 use a median filter to remove spike noise. Project incomplete.

SECTION 3: SPECTRAL AND RHYTHMICITY ANALYSIS

Code directory: D:\Ron\Documents\Ron Fredericks Consulting\Education\Signal processing problems - Udemy\Section 03\ sigprocMXC-spectral

20. CRASH COURSE ON THE FOURIER TRANSFORM

Use Fourier transform to move data from the time domain to the frequency domain

For each pure sine wave frequency bin, calculate how much of that frequency is in the signal to be transformed. Where the height of the bar is related to the amplitude of that sine wave within the signal.

Use the inverse

A bar is placed in the frequency domain to indicate the frequency present in the time series

Height of the bar corresponds to the amount of similarity or some other characteristic that is obtained from the dot product computation at the x-axis location corresponding to the frequency of the sine wave.

Major use #1: Spectral analysis

- Some signals are better understood in the frequency domain.
- Spectral analyses can reveal insights that cannot be seen in the time domain.

Major use #2: A means to an end in signal processing

- Use the convolution theorem to perform operations in the frequency domain, including filtering, autocorrelation, etc.
- Frequency-domain computations are often easier and faster than equivalent time-domain computations.

Frequency resolution = number of time points in signal.

Nyquist is ½ the sampling rate.

Zero padding can be used to improve frequency resolution.

21. FOURIER TRANSFORM FOR SPECTRA ANALYSES

MATLAB

Code: sigprocMXC_FourierTransform.m

Amplitude spectrum via Fourier transform

```
signal = fft(signal);

signalAmp = 2*abs(signal)/npnts;
```

Vector of frequencies in Hz

```
Hz = linspace(0, srate/2, floor(npnts/2)+1);
```

Where

srate/2 is the nyquest frequency or range

floor(npts/2)+1 is the frequency resolution

Inverse Fourier transform with respect to time

```
ifft(signal);
```

Example on how to reduce fft high end frequency without reducing resolution

```
f_scale2 = f_scale(1:end/2)

plot(f_scale2,abs(spectrum(1:length(f_scale2))));
```

Figure 1

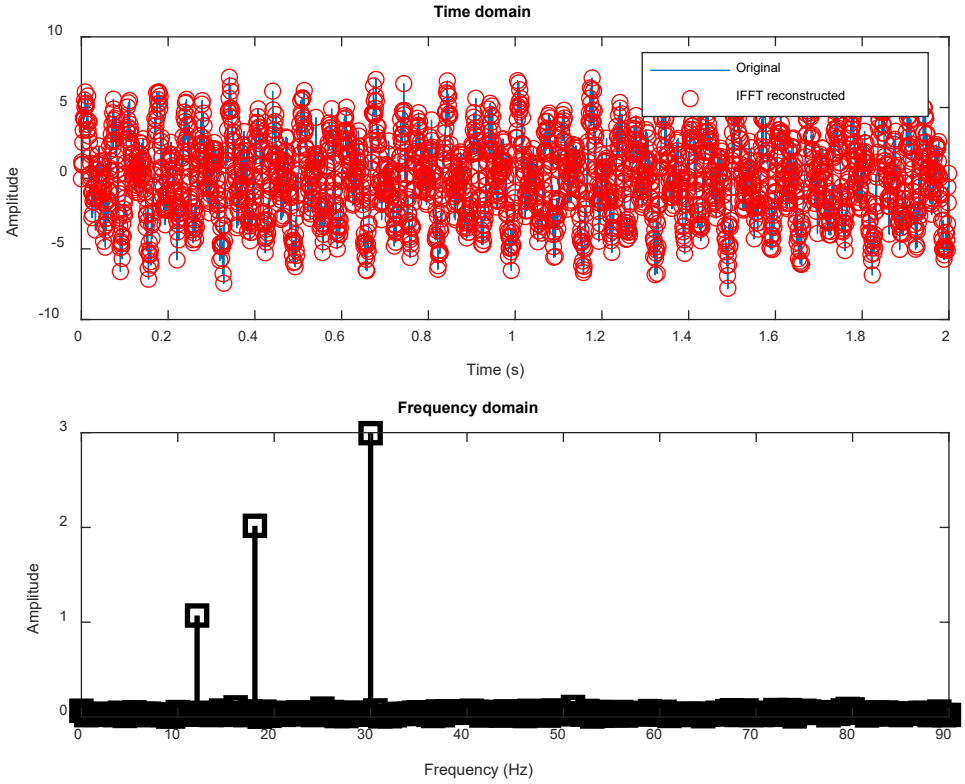
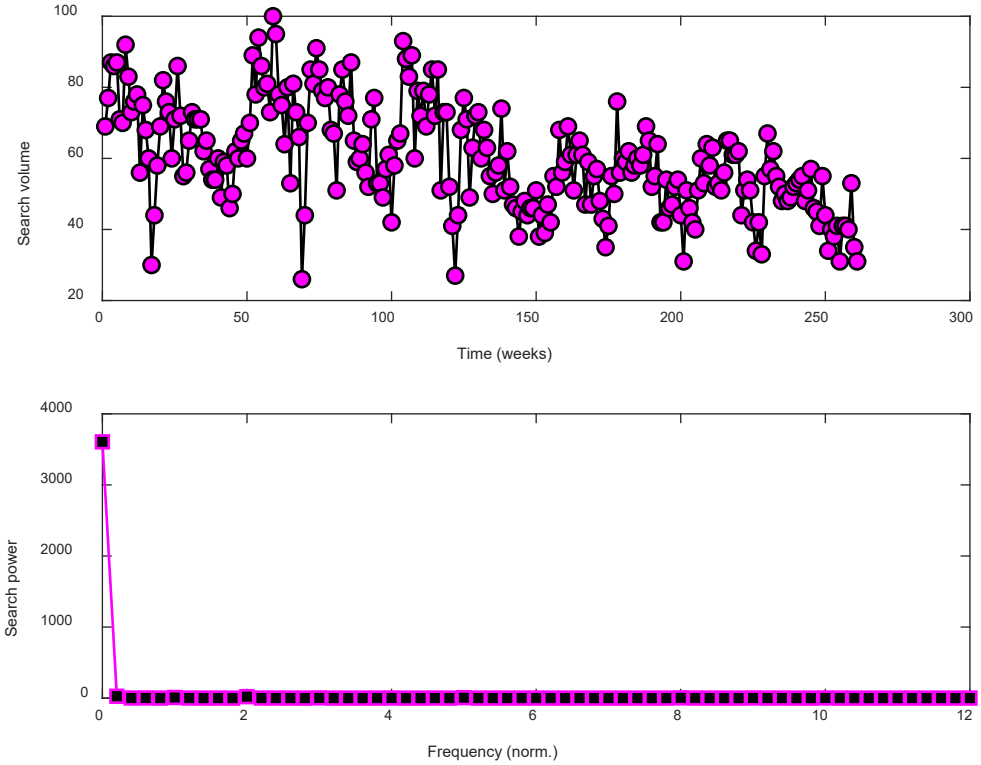

Time domain
Original
IFFT reconstructed
Amplitude
Time (s)
Frequency domain
Amplitude
Frequency (Hz)

Figure 2



22. WELCH'S METHOD AND WINDOWING

MATLAB

Code: sigprocMXC_Welch.m

Figure 1

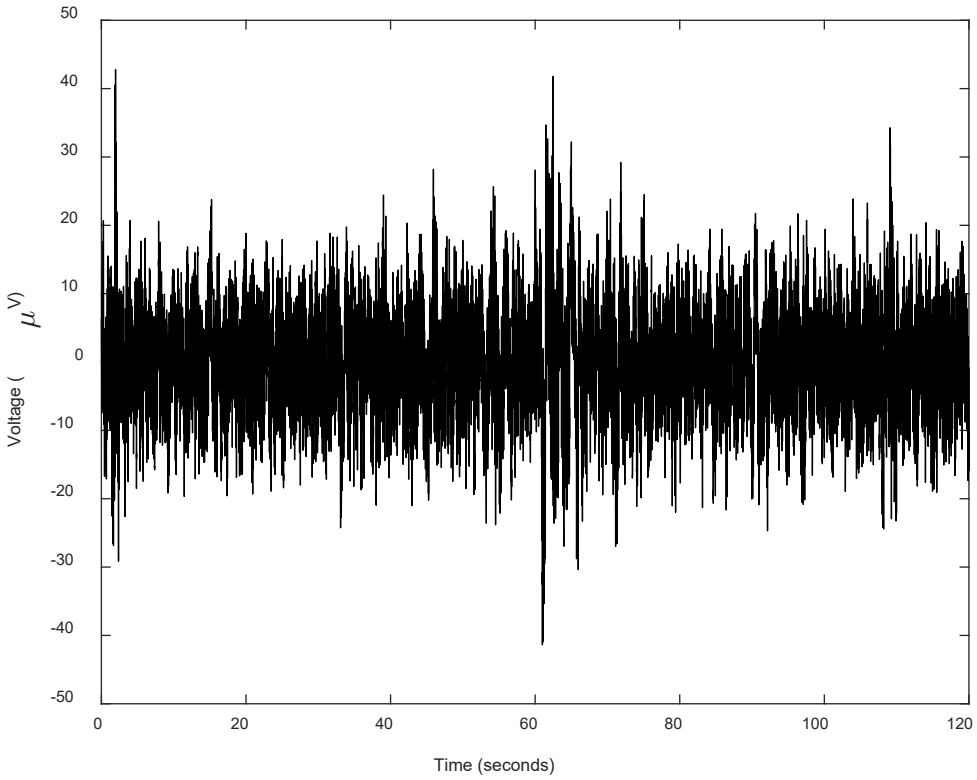

50
40
30
20
10
0
-10
-20
-30
-40
-50
0
20
40
60
80
100
120
Voltage (μV)
Time (seconds)

Figure 2

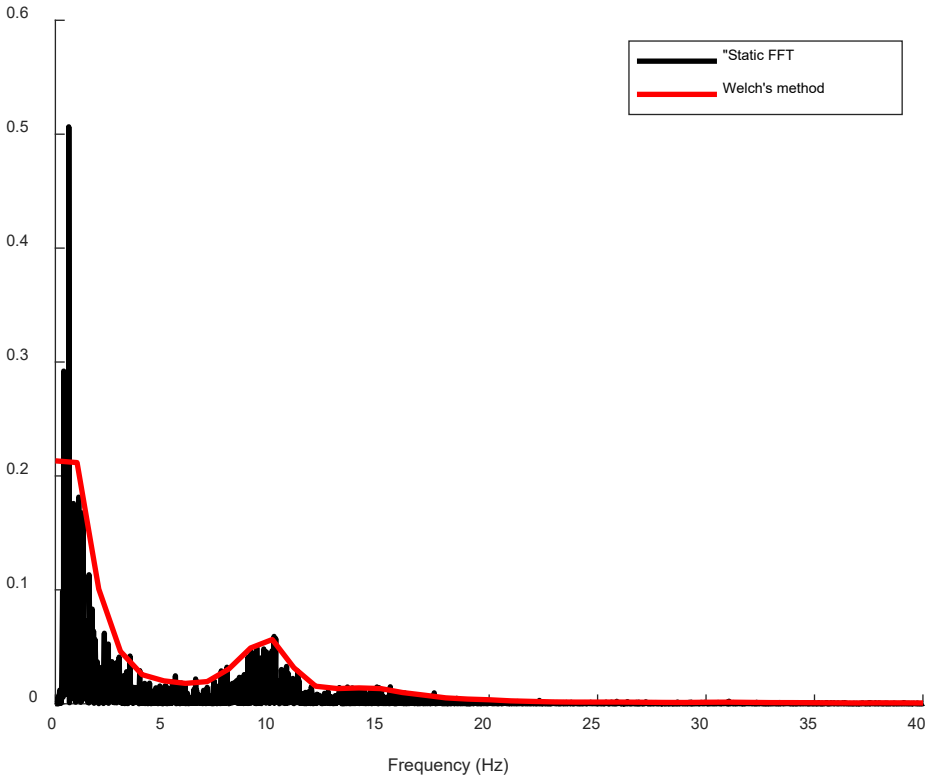
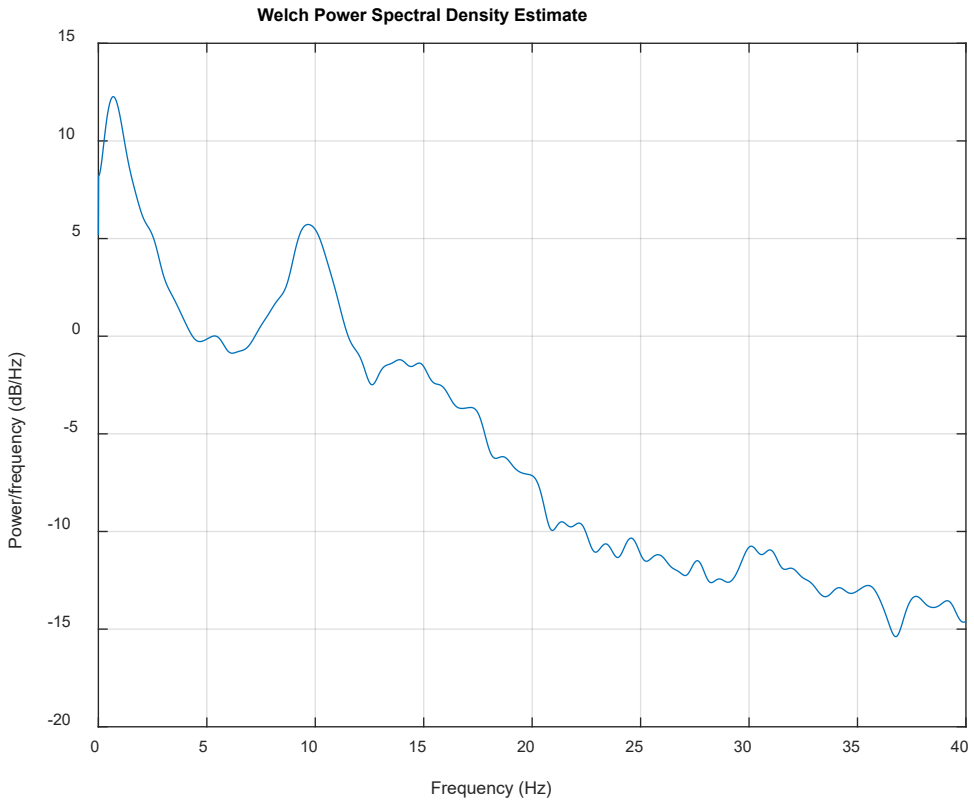

"Static FFT
Welch's method
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40
Frequency (Hz)

Figure 3



23. SPECTROGRAM OF BIRDSONG

MATLAB

Code: sigprocMXC_SpectBirdcall.m

Data: XC403881.mp3

Data reference: Bird call database https://www.xeno-canto.org/403881

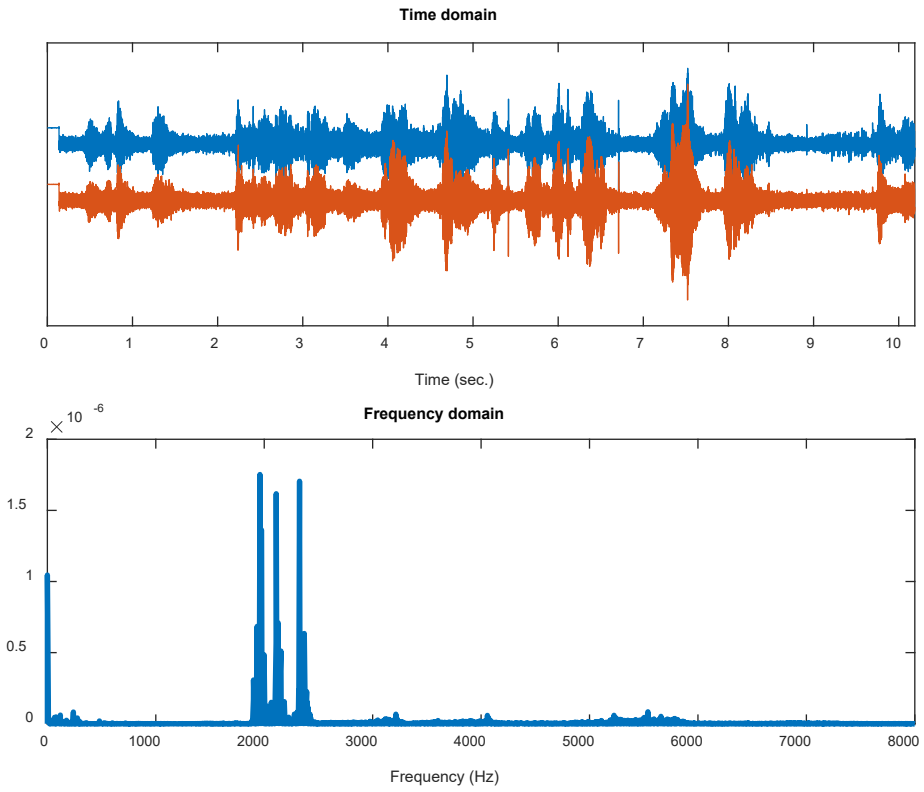

Time domain
0
1
2
3
4
5
6
7
8
9
10
Time (sec.)
Frequency domain
×10 -6
2
1.5
1
0.5
0
0
1000
2000
3000
4000
5000
6000
7000
8000
Frequency (Hz)

Figure 1

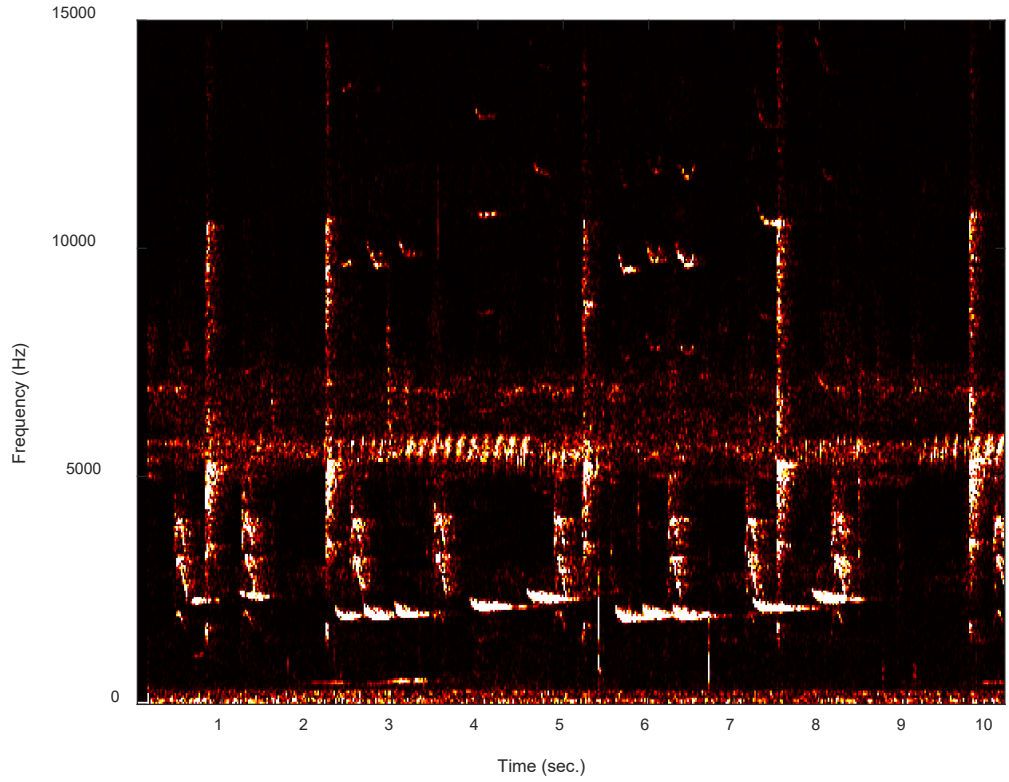


Figure 2

SECTION 4: WORKING WITH COMPLEX NUMBERS

Code directory: D:\Ron\Documents\Ron Fredericks Consulting\Education\Signal processing problems - Udemy\Section 04\

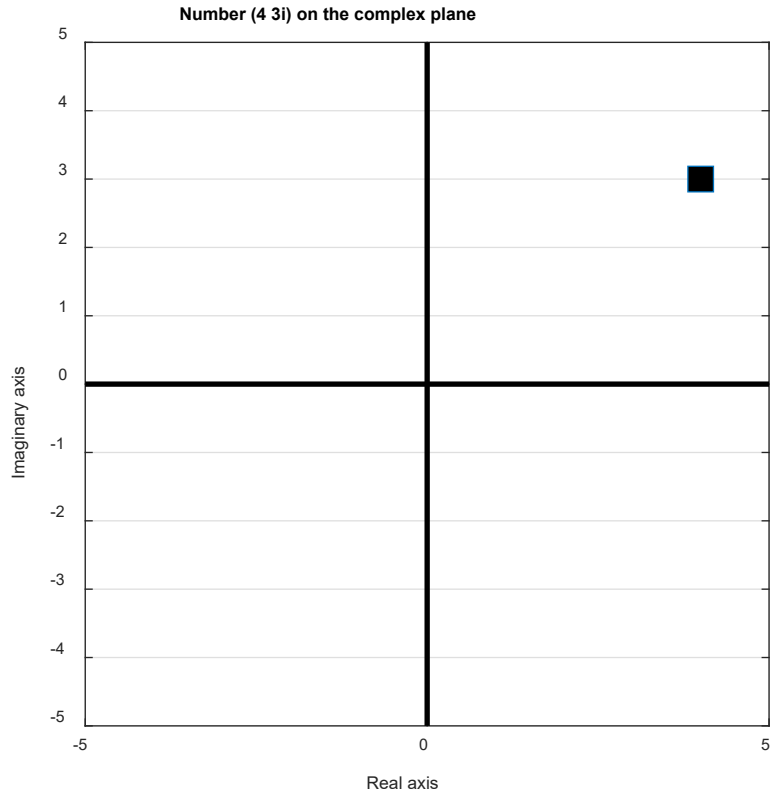
26. FROM THE NUMBER LINE TO THE COMPLEX NUMBER PLANE

MATLAB

Code: sigprocMXC_complexIntro.m

```
% several ways to create a complex number
z = 4 + 3i;
z = 4 + 3*1i;
z = 4 + 3*sqrt(-1);
z = complex(4,3);
```

```
disp([ 'Real part is ' num2str(real(z)) ' and imaginary part is ' num2str(imag(z)) '.' ])
```



27. ADDITION AND SUBTRACTION WITH COMPLEX NUMBERS

MATLAB

Code: sigprocMXC_complexAddSub.m

```
% create two complex numbers
a = complex(4,5);
b = 3+2i;

% let MATLAB do the hard work
z1 = a+b;

% the "manual" way
z2 = complex( real(a)+real(b) , imag(a)+imag(b) );
```

28. MULTIPLICATION WITH COMPLEX NUMBERS

MATLAB

Code: sigprocMXC_complexMult.m

```
% create two complex numbers
a = complex(4,5);
b = 3+2i;

% let MATLAB do the hard work
z1 = a*b;

% the intuitive-but-WRONG way
z2 = complex( real(a)*real(b) , imag(a)*imag(b) );

% the less-intuitive-but-CORRECT way
ar = real(a);
ai = imag(a);
br = real(b);
bi = imag(b);

z3 = (ar + 1i*ai) * (br + 1i*bi);
z4 = (ar*br) + (ar*(1i*bi)) + ((1i*ai)*br) + ((1i*ai)*(1j*bi));
```

29. THE COMPLEX CONJUGATE

MATLAB

Code: sigprocMXC_complexConj.m

```
% create a complex number
a = complex(4,-5);

% let MATLAB do the hard work
ac1 = conj(a);

% the "manual" way
ac2 = complex( real(a) , -imag(a) );

%% magnitude squared of a complex number

amag1 = a*conj(a);

amag2 = real(a)^2 + imag(a)^2;

amag3 = abs(a)^2;
```

30. DIVISION WITH COMPLEX NUMBERS

MATLAB

Code: sigprocMXC_complexDivision.m

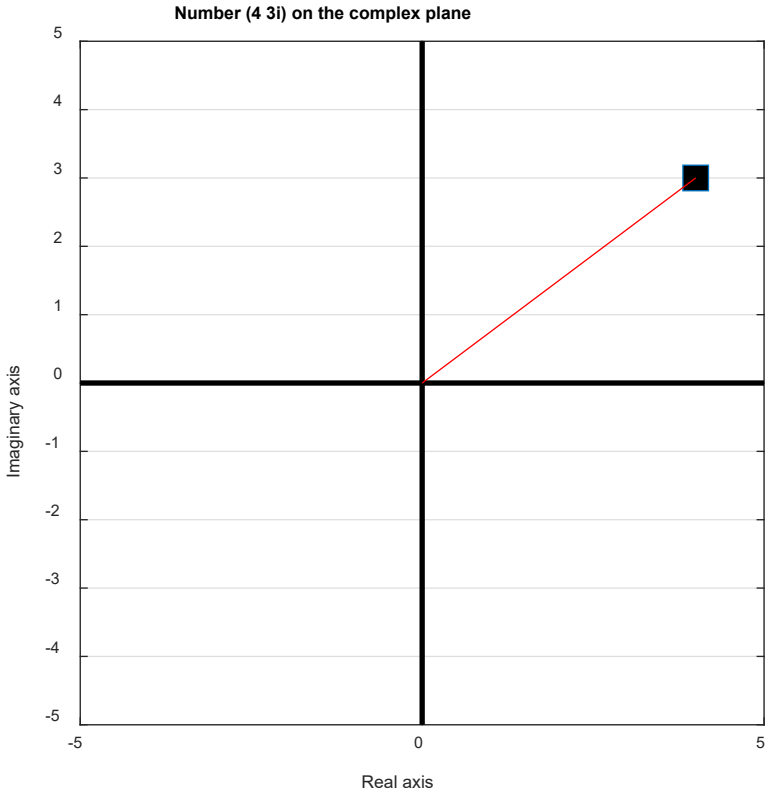
```
% create two complex numbers
a = complex(4,-5);
b = complex(7,8);

% let MATLAB do the hard work
adb1 = a/b;

% the "manual" way
adb2 = (a*conj(b)) / (b*conj(b));
```

31. MAGNITUDE AND PHASE OF COMPLEX NUMBERS

MATLAB

Code: sigprocMXC_complexPolar.m

```
%% compute magnitude and phase of the complex number

% magnitude of the number (distance to origin)
magZ1 = sqrt( real(z)^2 + imag(z)^2 );
magZ2 = abs( z );

% angle of the line relative to positive real axis
angZ1 = atan2( imag(z),real(z) );
angZ2 = angle( z );

% draw a line using polar notation
h = polar([0 angZ1],[0 magZ1],'r');
```



SECTION 5: FILTERING

Code Directory: D:\Ron\Documents\Ron Fredericks Consulting\Education\Signal processing problems - Udemy\Section 05\sigprocMXC-filtering

33. FILTERING: INTUITION, GOALS, AND TYPES

Temporal filtering: separating signal and noise at different frequencies

Filtering in the frequency domain: use fft, zero out some frequencies, pass frequencies we are interested in. Low Pass, High Pass filters.

Filtering in the time domain: define a filter kernel function to assign what frequencies get filtered: FIR (Finite Impulse Filter) example. See convolution theorem.

Procedure for filtering time series data:

1. Define frequency-domain shape and cut-offs (the ideal filter response). Adjust transition width, or bandpass width to correct frequency response of filter.
2. Generate filter kernel (firls – for least square, fir1, butter, other). Adjust order of kernel.
3. Evaluate kernel and its power spectrum, the waveform shape that is applied to the data (inspect in both time and frequency domain, most important step). Take fft of this filter to investigate frequency response. Look for over and underrepresented frequencies, look for attenuation in areas that should be attenuated.
4. Apply filter kernel to data

	FIR (Better quality, more stable)	IIR (better for real-time)
Name	Finite impulse response	Infinite impulse response
Kernel length (filter order)	Long (100's to 1000's of points)	Short (less than 10 points0
Speed	Slower	Fast
Stability	High	Data-dependent
Mechanism	Multiply data with kernel	Multiply data with data

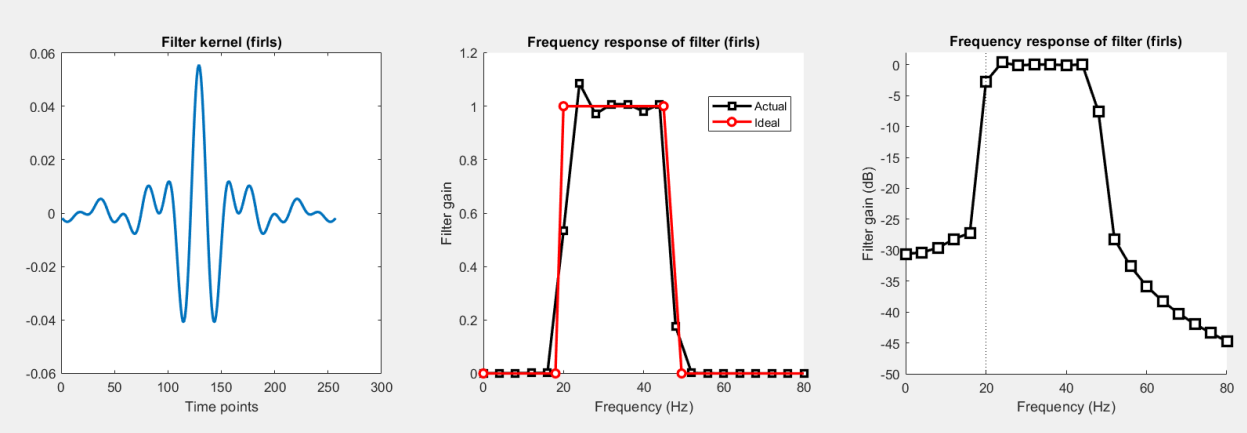
34. FIR FILTERS WITH FIRLS (FINITE IMPULSE RESPONSE LEAST SQUARES)

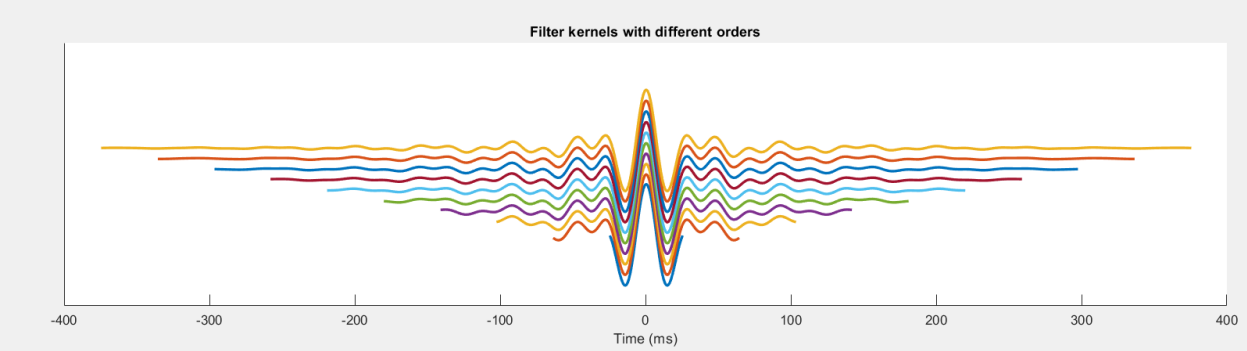
A filter is described using a minimum of 6 data points:

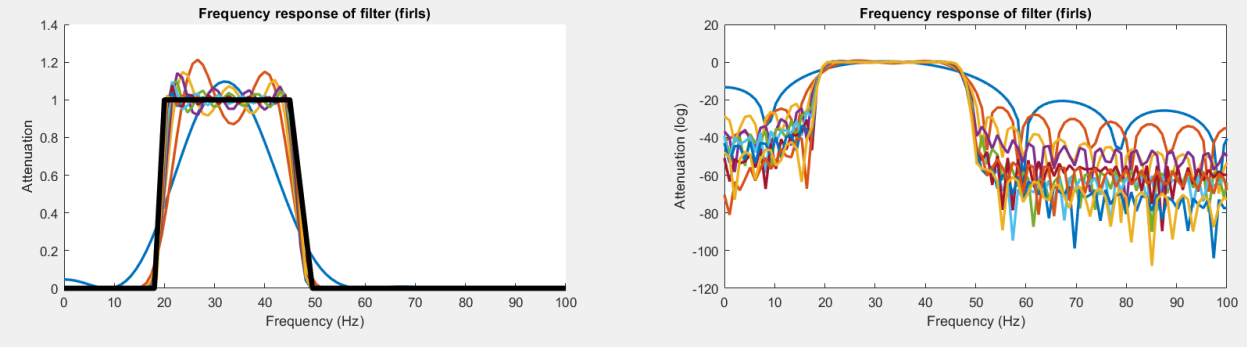
Sharp edges are difficult to represent with a series of sine waves such as when using fft

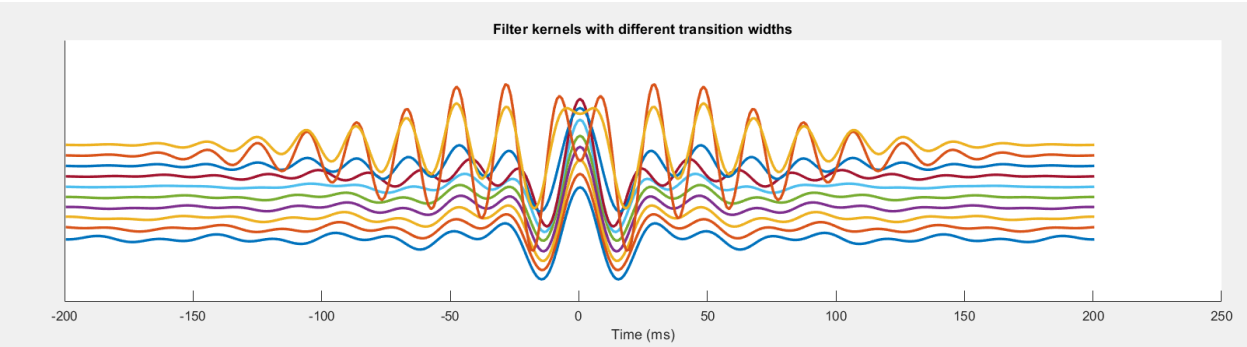
MATLAB

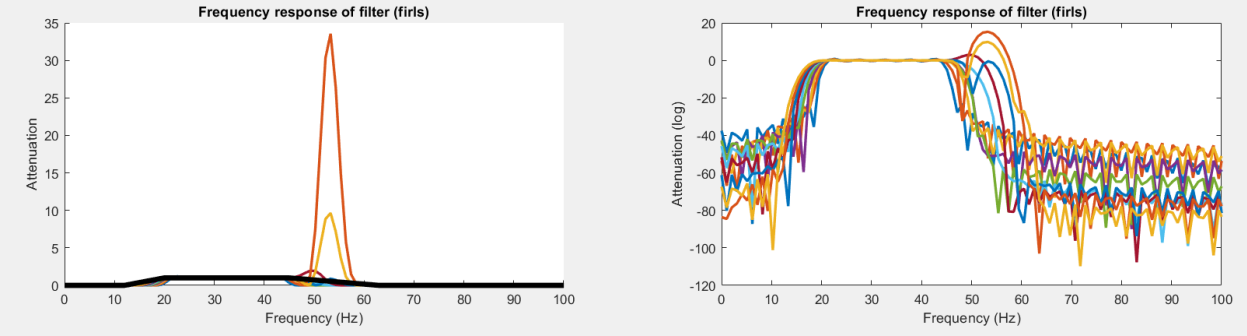
Code: sigprocMXC_firls.m




Filter kernels with different orders
-400
-300
-200
-100
0
100
200
300
400
Time (ms)


Frequency response of filter (firls)
Attenuation
Frequency (Hz)
Frequency response of filter (firls)
Attenuation (log)
Frequency (Hz)


Filter kernels with different transition widths
-200
-150
-100
-50
0
50
100
150
200
250
Time (ms)

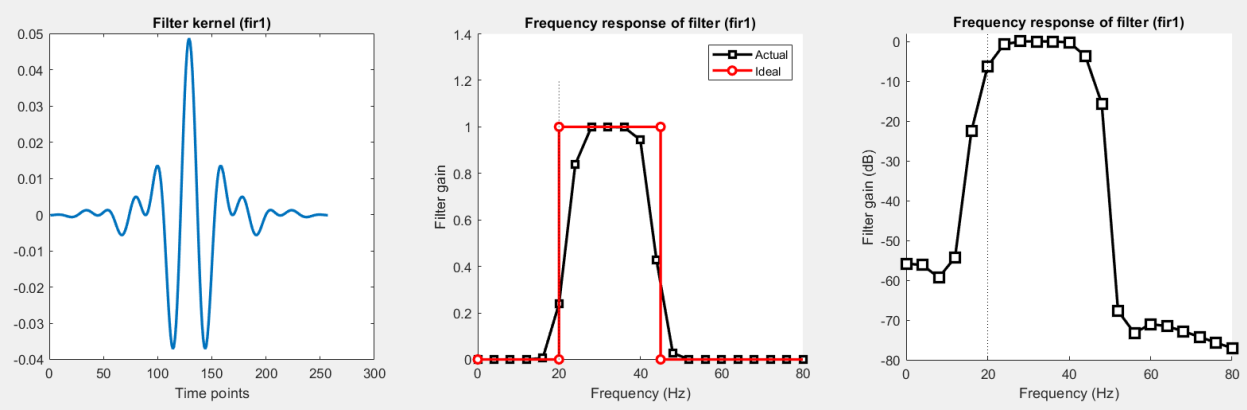

Frequency response of filter (firls)
Attenuation
Frequency (Hz)
Frequency response of filter (firls)
Attenuation (log)
Frequency (Hz)

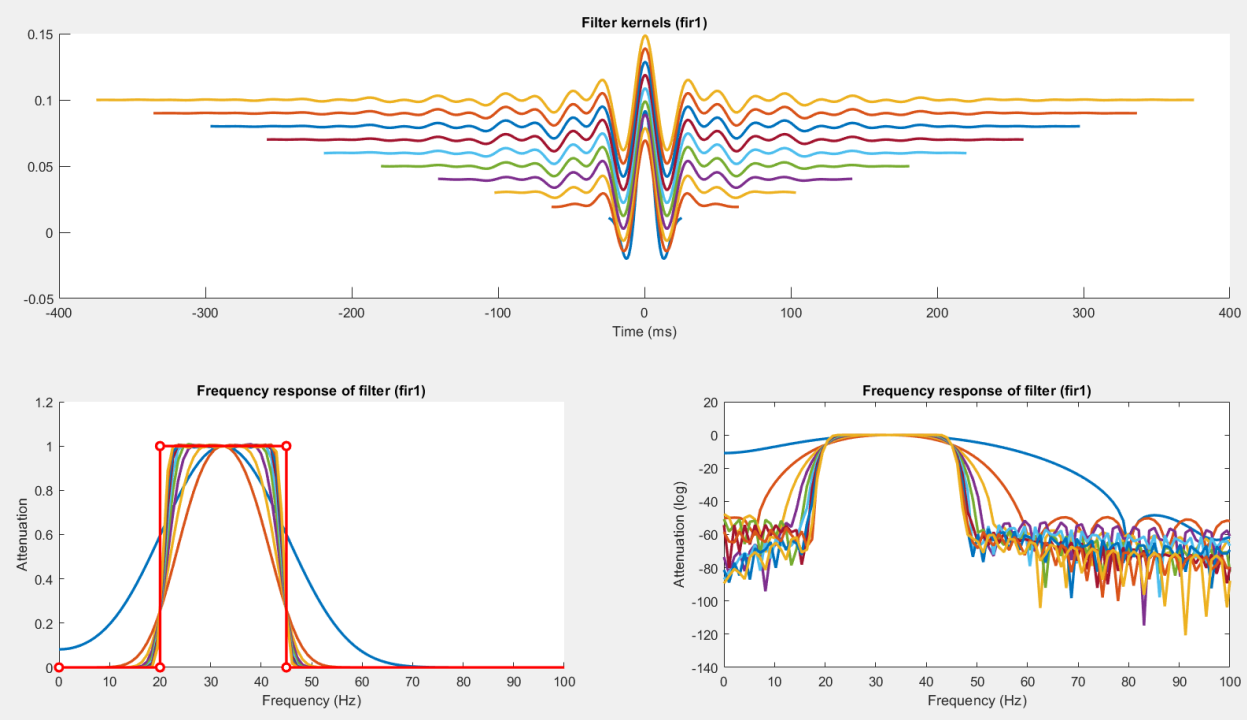
FIR1 calls FIRLS along with no transition zones. Use 2 points for a bandpass filter. A filter kernel will be created with 6 points (by FIRLS) except with zero transition zones. This in turn creates ripple artifacts. The edges are smoothed out using a window in the time domain – which in turn creates the otherwise missing transition zones.

This filter is idea when maximum attenuation is required near the edges.

MATLAB

Code: sigprocMXC_fir1.m



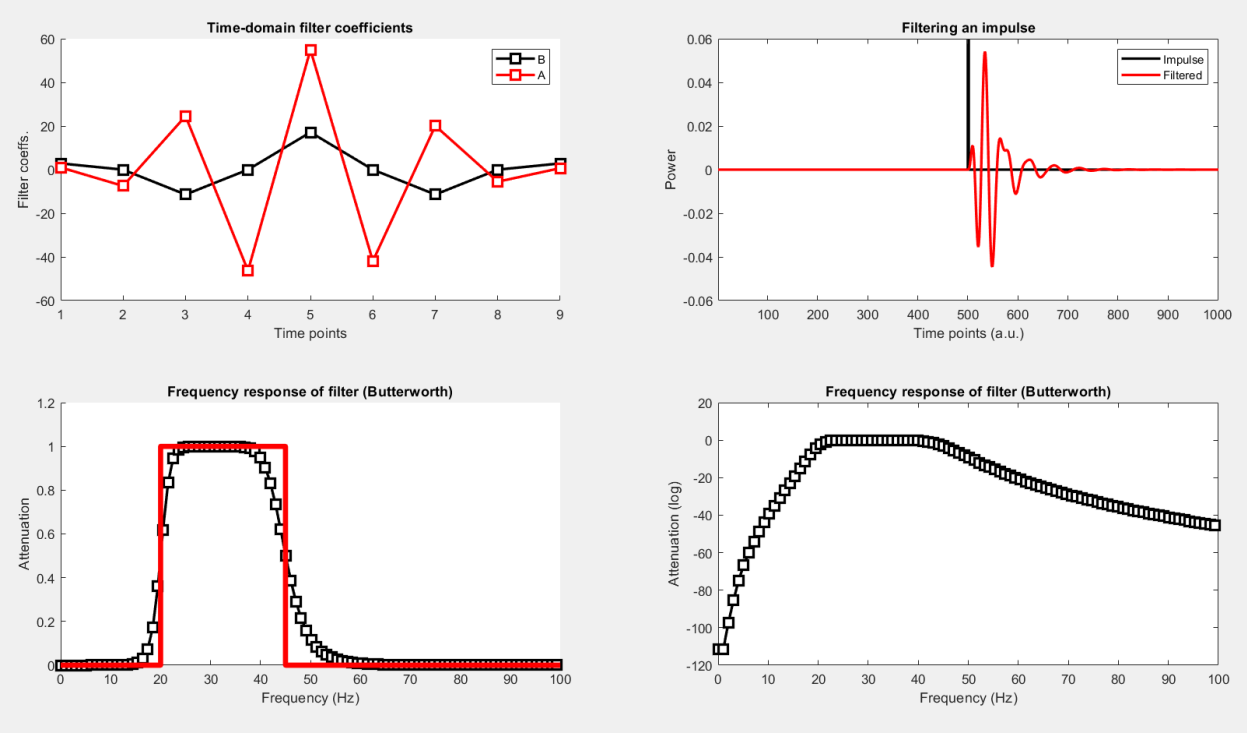


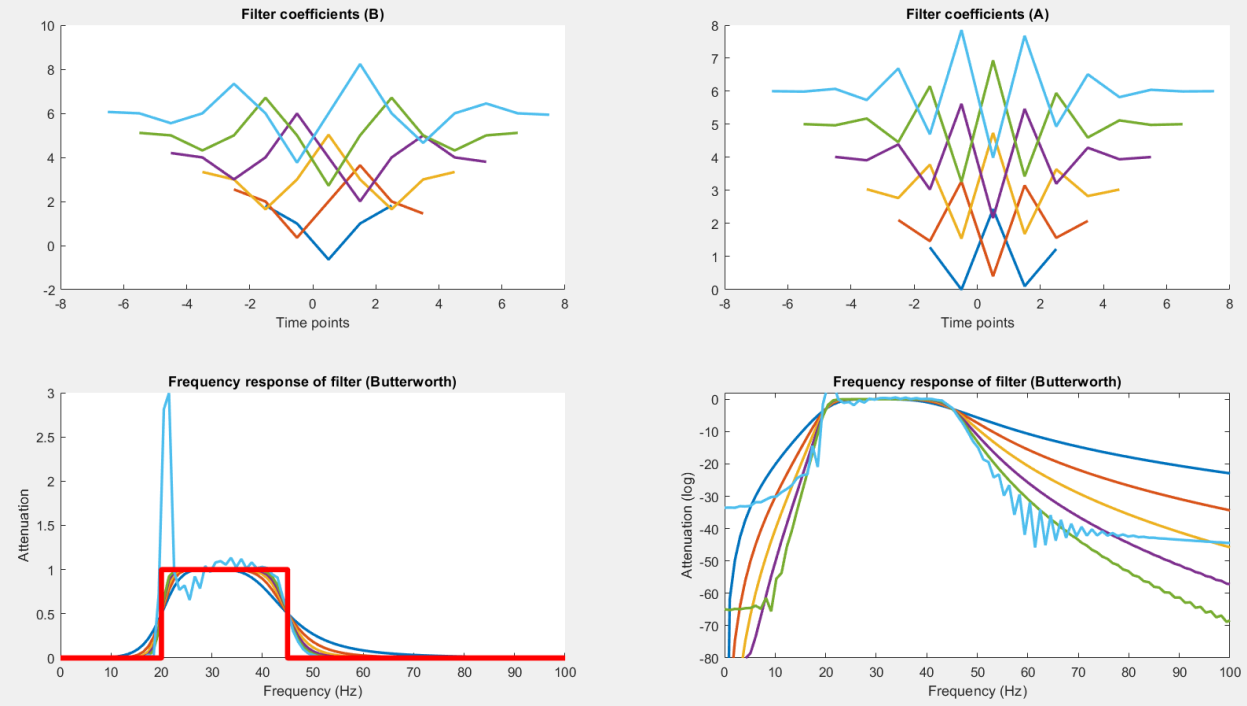
36. IIR BUTTERWORTH FILTERS

The key difference between IIR and FIR is in how you evaluate the filter; parameters are suitable and filtering is good.

MATLAB

Code: sigprocMXC_butter.m

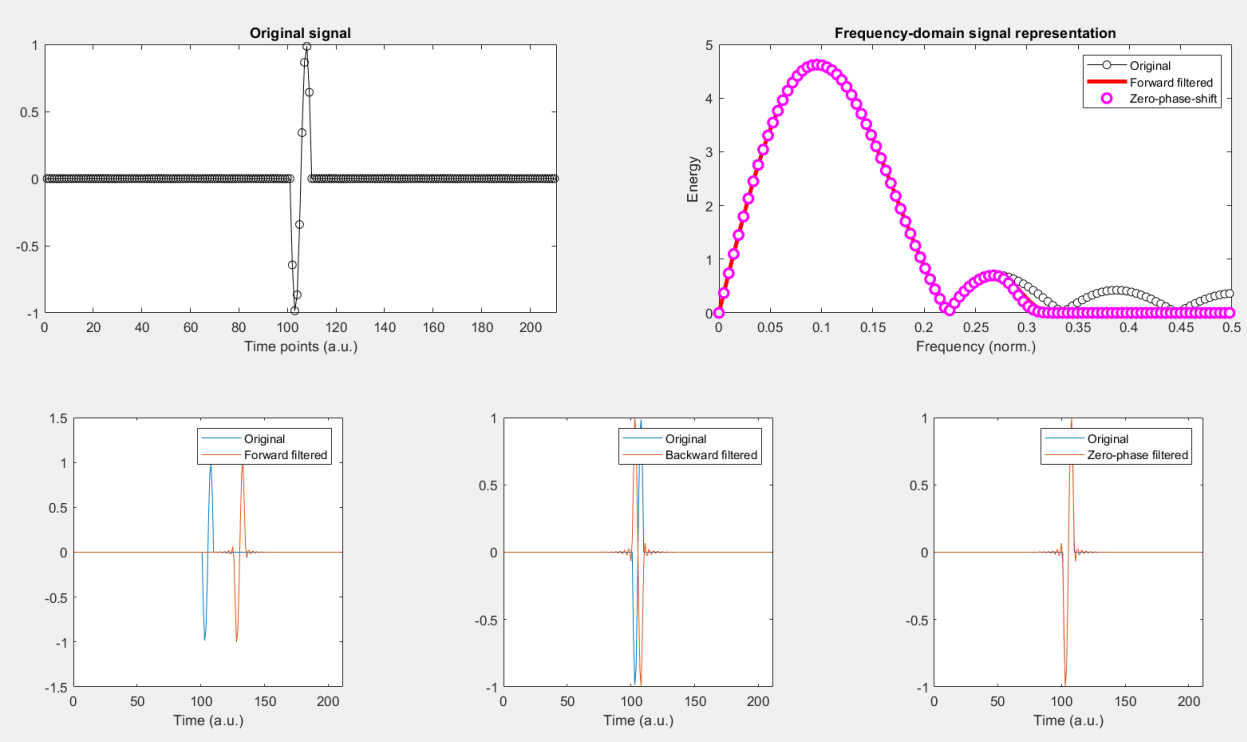

Time-domain filter coefficients
B
A
Filter coeffs.
Time points
Filtering an impulse
Impulse
Filtered
Power
Time points (a.u.)
Frequency response of filter (Butterworth)
Attenuation
Frequency (Hz)
Frequency response of filter (Butterworth)
Attenuation (log)
Frequency (Hz)


Filter coefficients (B)
Time points
Filter coefficients (A)
Time points
Frequency response of filter (Butterworth)
Attenuation
Frequency (Hz)
Frequency response of filter (Butterworth)
Attenuation (log)
Frequency (Hz)

- Causal filters: A filter is called causal if its output is the result of only current and past inputs, i.e., past inputs “cause” the current output. Causal filters are the only option for real-time implementation. All IIR filters behave this way.
- Zero Phase-shift filters: A filter that applies the kernel in both a forward and reverse direction to remove phase error. Some FIR filters behave this way.

MATLAB

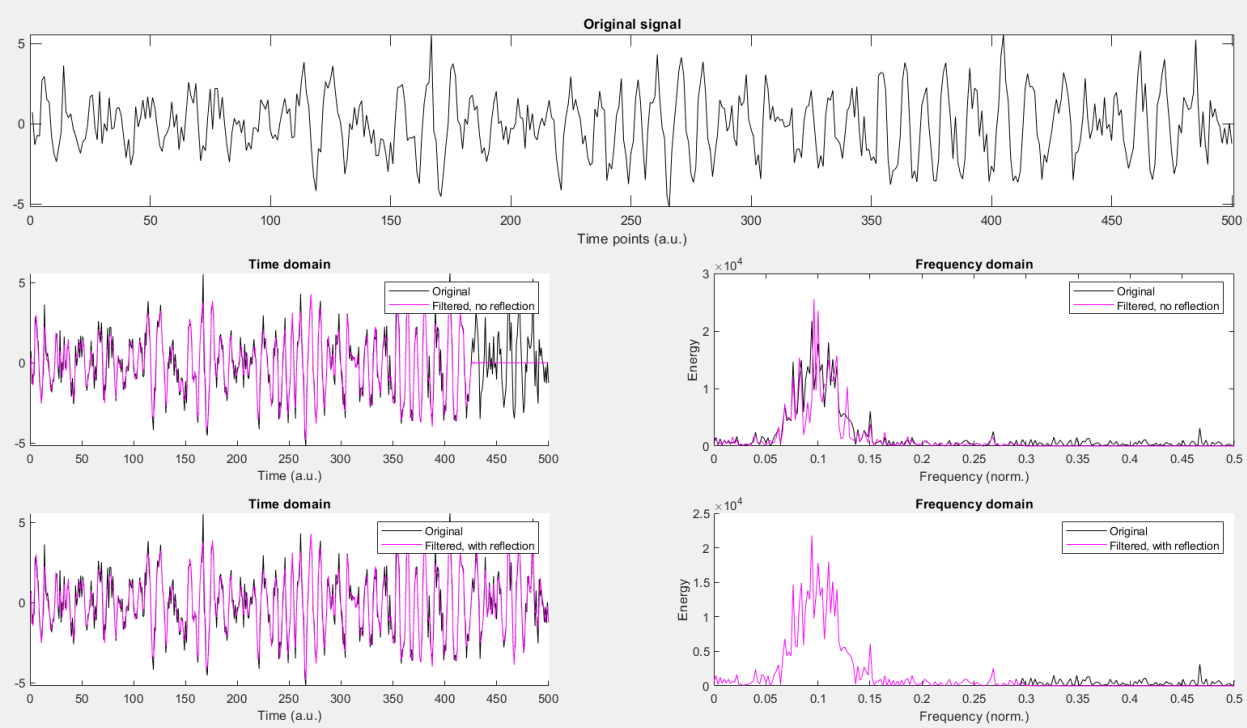
Code: sigprocMXC_causal0phase.m



38. AVOID EDGE EFFECTS WITH REFLECTION

MATLAB

Code: sigprocMXC_reflection.m



39. DATA LENGTH AND FILTER KERNEL LENGTH

Data must be 3 times filter order in order to generate a reliable filter with low edge effects.

If original data cannot be recovered with a longer number of samples, then reflection can be used:

- Use some reflected data before and after original signal to increase the number of datapoints
- Create the filter
- Remove the reflected data points
- Use the filter

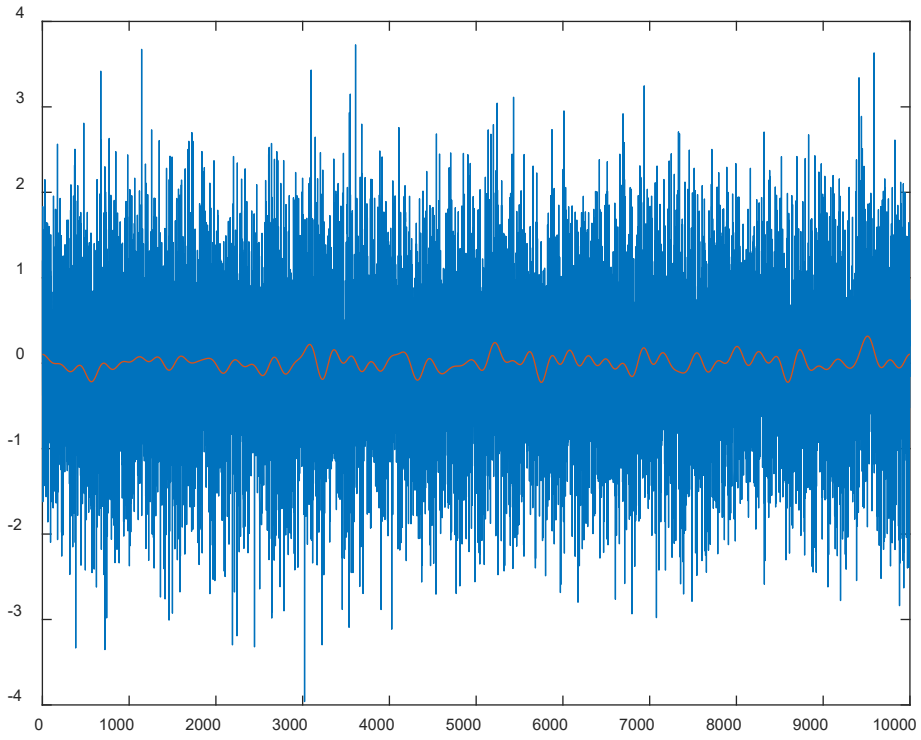
MATLAB

Code: sigprocMXC_signalLength.m

Truncated code:

```
% parameters
dataN = 10000;
filtN = 5001;

% generate data
signal = randn(dataN,1);

% create filter kernel
fkern = fir1(filtN,.01,'low');
```

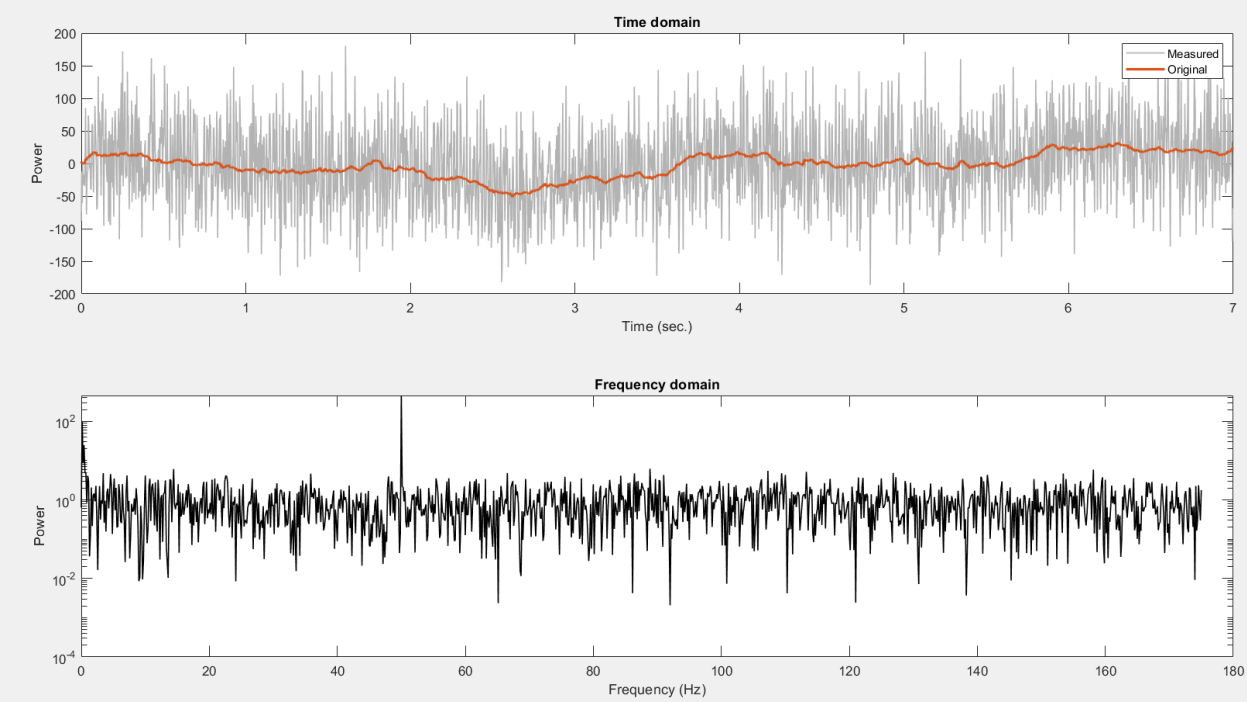
```
% use reflection to increase signal length!
% use semicolons to make this a column-wise vector
signalRefl = [ signal(end:-1:1); signal; signal(end:-1:1) ];


% apply filter kernel to data
fdataR = filtfilt(fkern,1,signalRefl);

% and cut off edges
fdata = fdataR(dataN+1:end-dataN);

plot(signal)
hold on
plot(fdata)
```

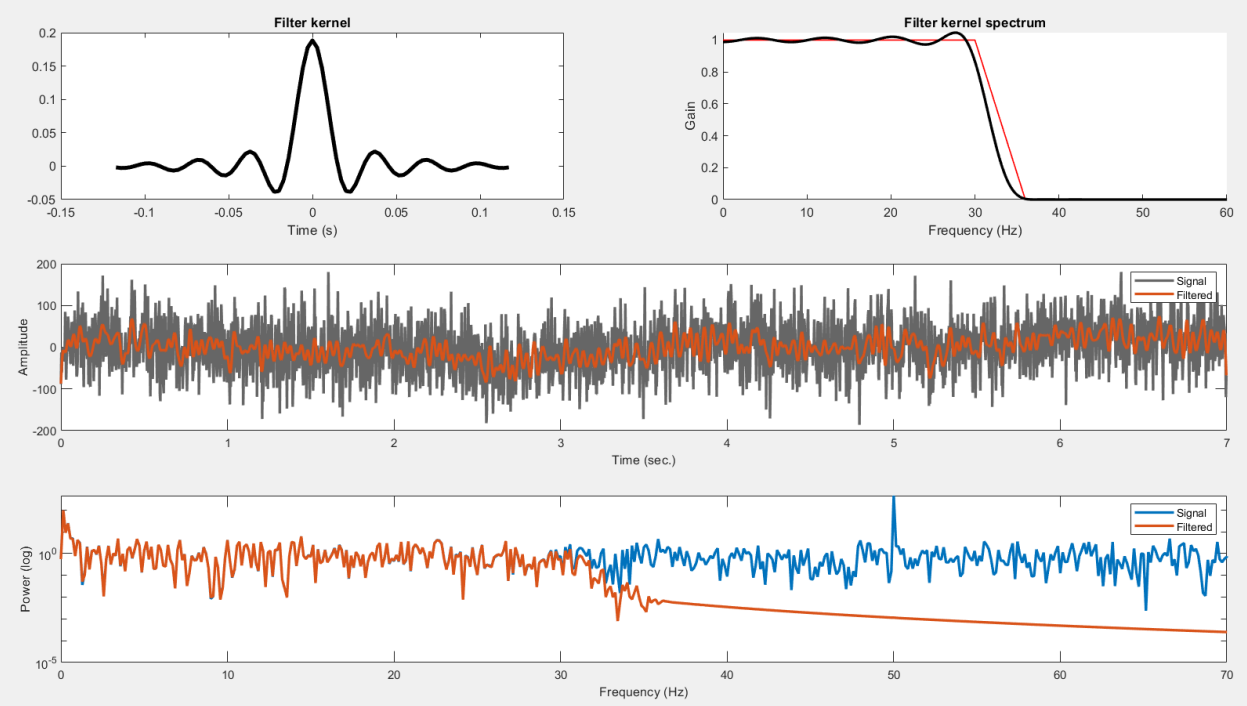


40. LOW-PASS FILTERS

MATLAB

Code: sigprocMXC_lowpass.m

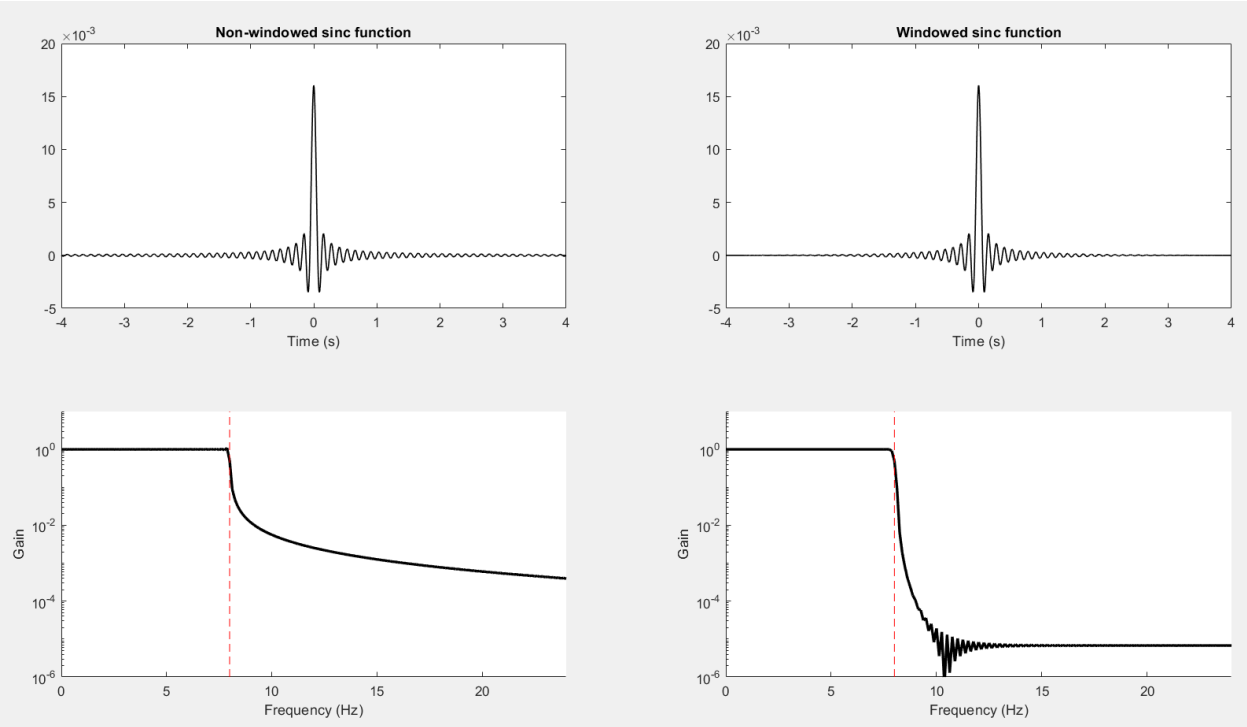

Time domain
Measured
Original
Power
Time (sec.)
Frequency domain
Power
Frequency (Hz)

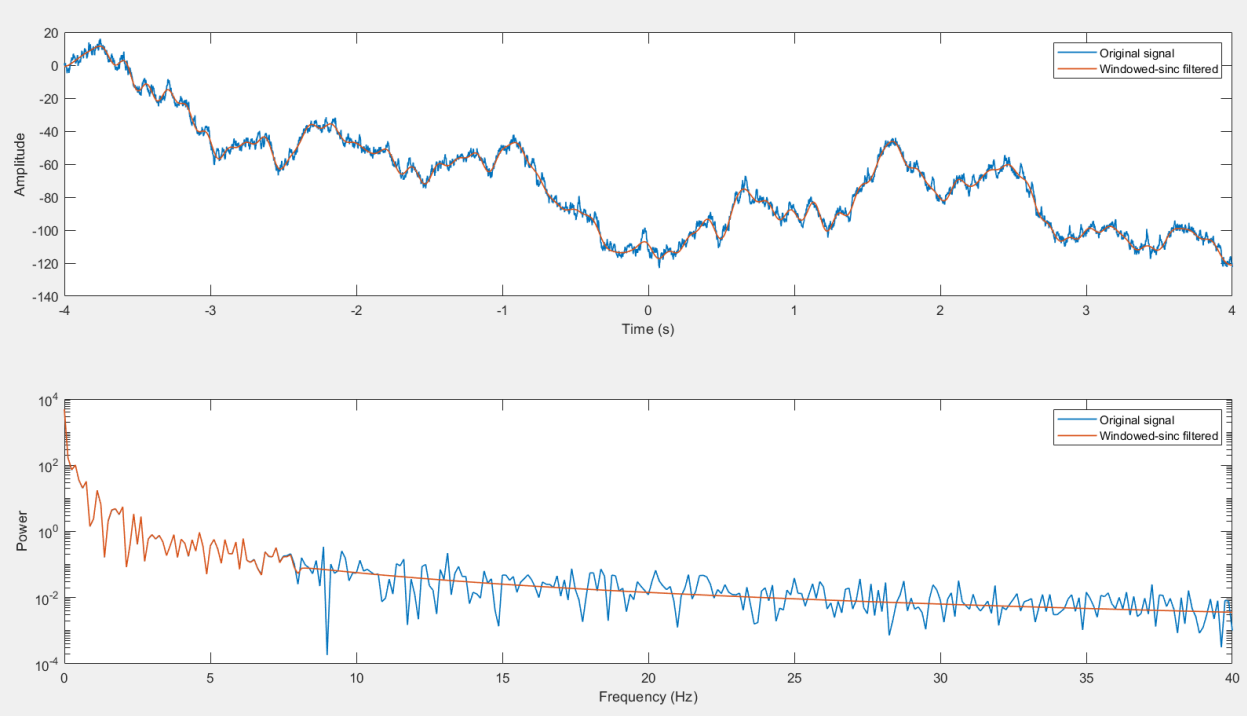

Filter kernel
Time (s)
Filter kernel spectrum
Gain
Frequency (Hz)
Signal
Filtered
Amplitude
Time (sec.)
Signal
Filtered
Power (log)
Frequency (Hz)

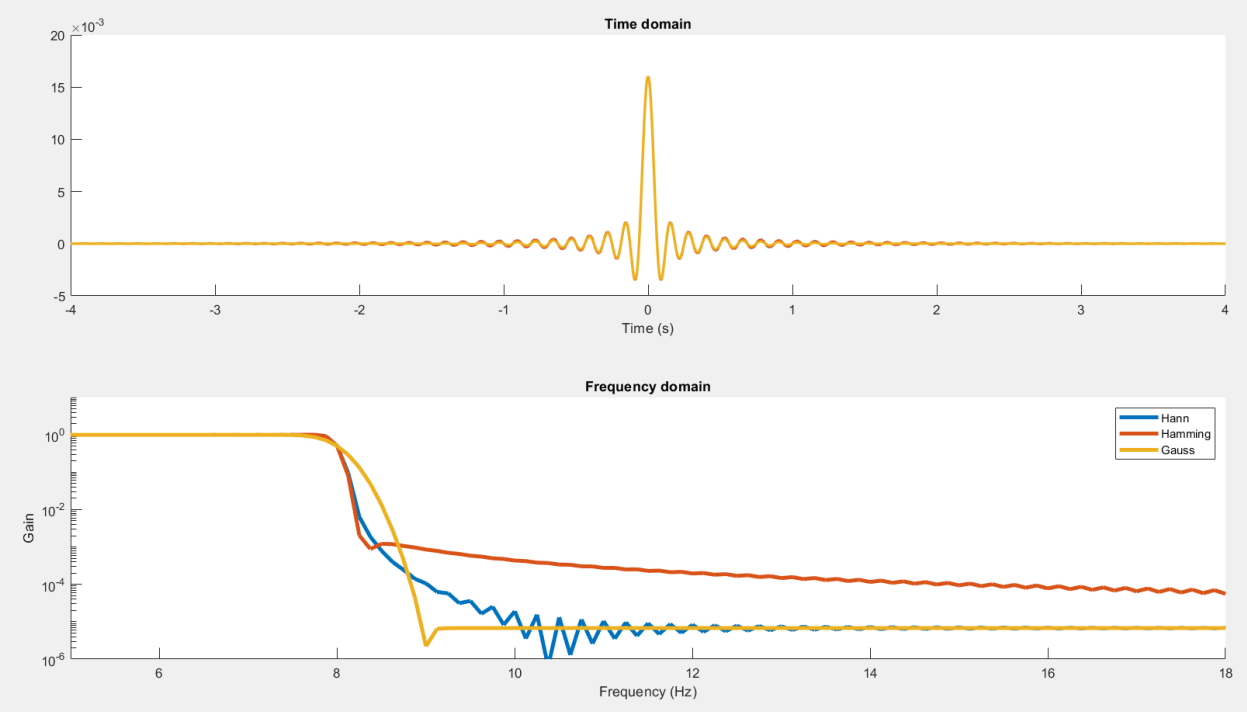
$$y = \frac{\sin(2\pi f t)}{t}$$

MATLAB

Code: sigprocMXC_windowSinc.m



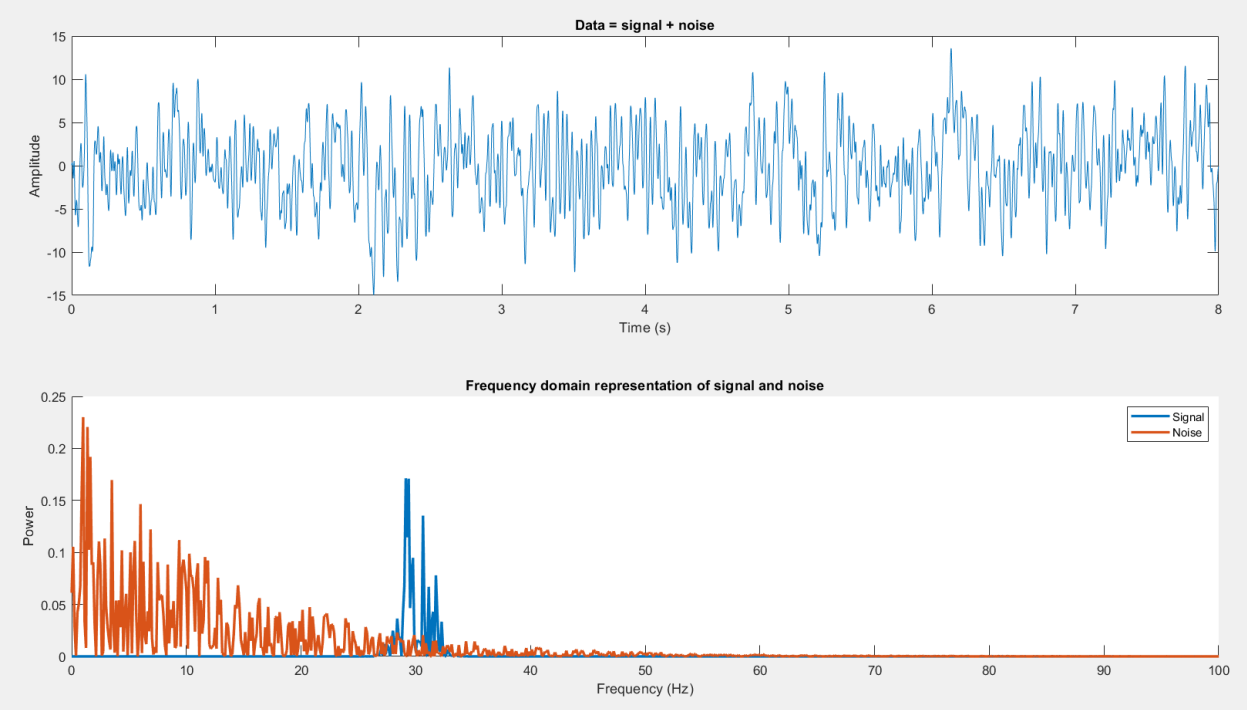


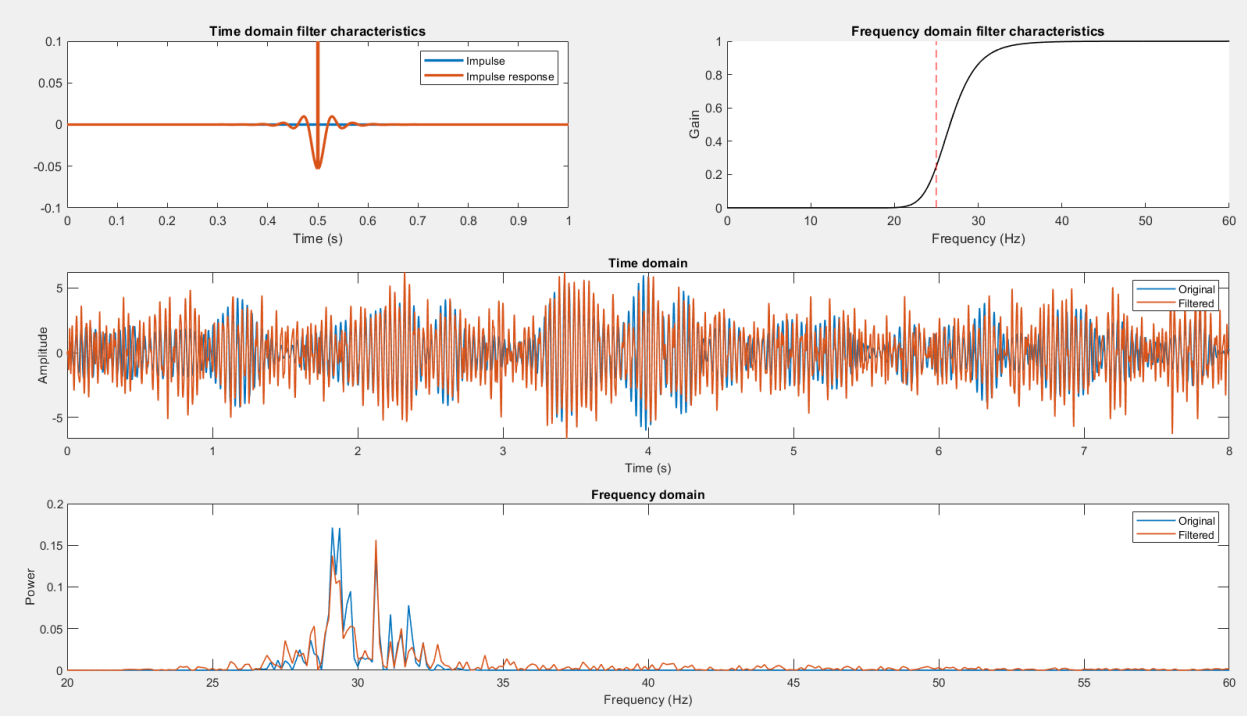


42. HIGH-PASS FILTERS (BUTTERWORTH IIR FILTER)

MATLAB

Code: sigprocMXC_highpass.m

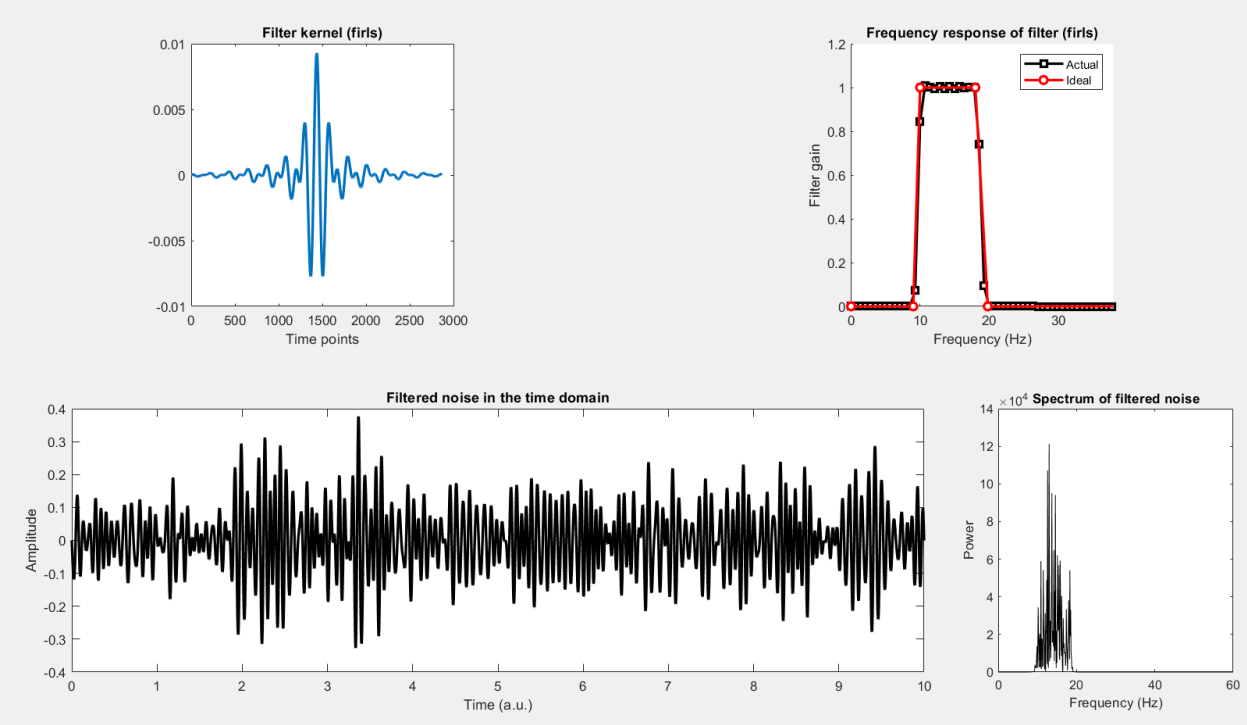




43. NARROW-BAND FILTERS (FIRLS)

MATLAB

Code: sigprocMXC_narrowband.m

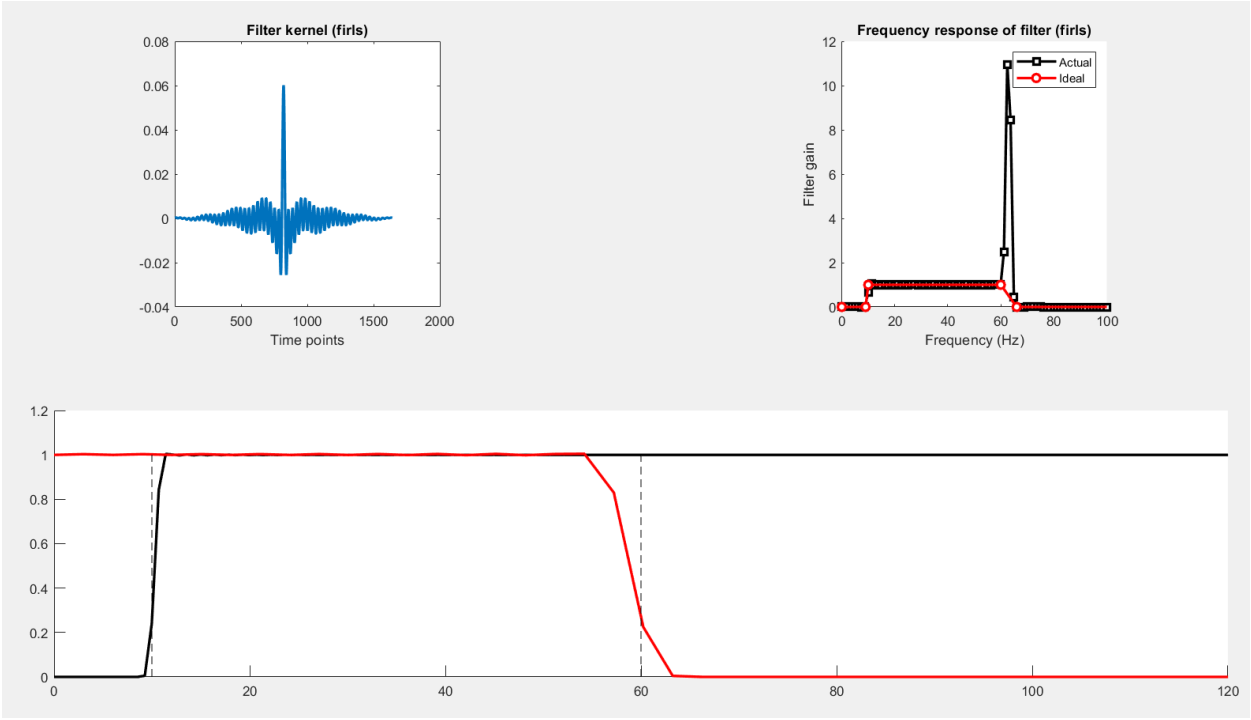


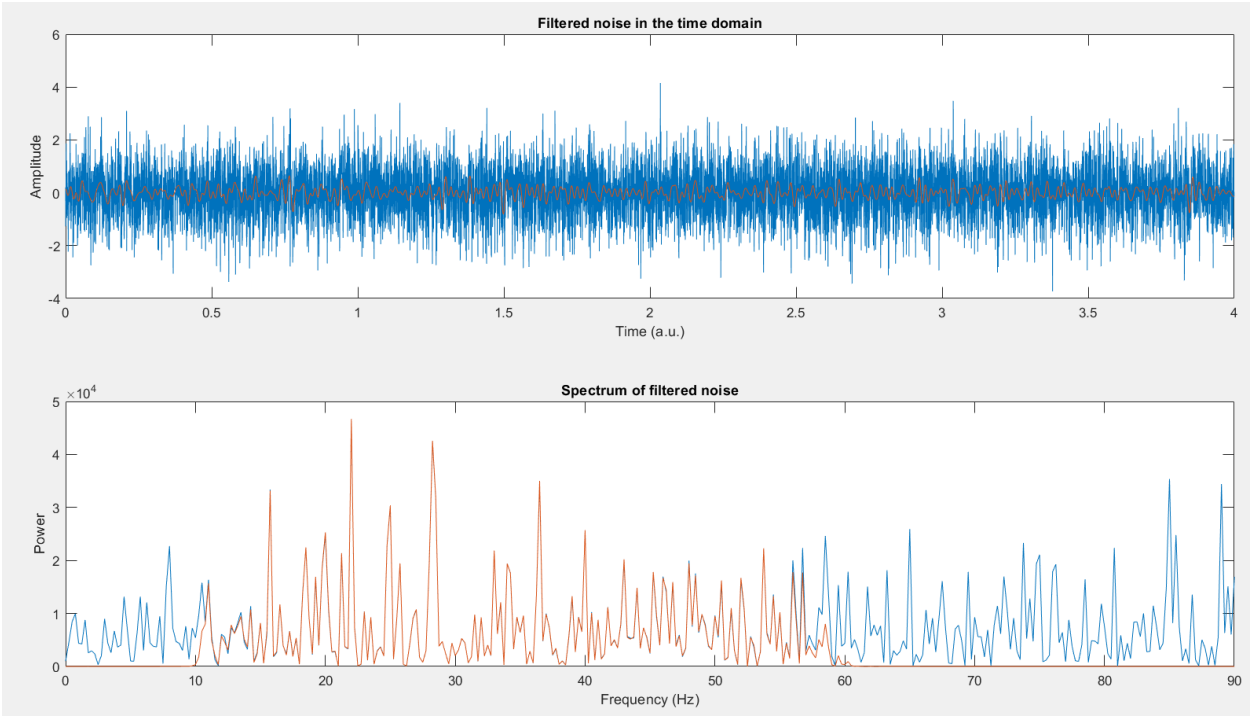
44. TWO-STAGE WIDE-BAND FILTER

Apply a high-pass filter first, then apply a low-pass filter to the filtered data at a higher order

MATLAB

Code: sigprocMXC_2stageWide.m





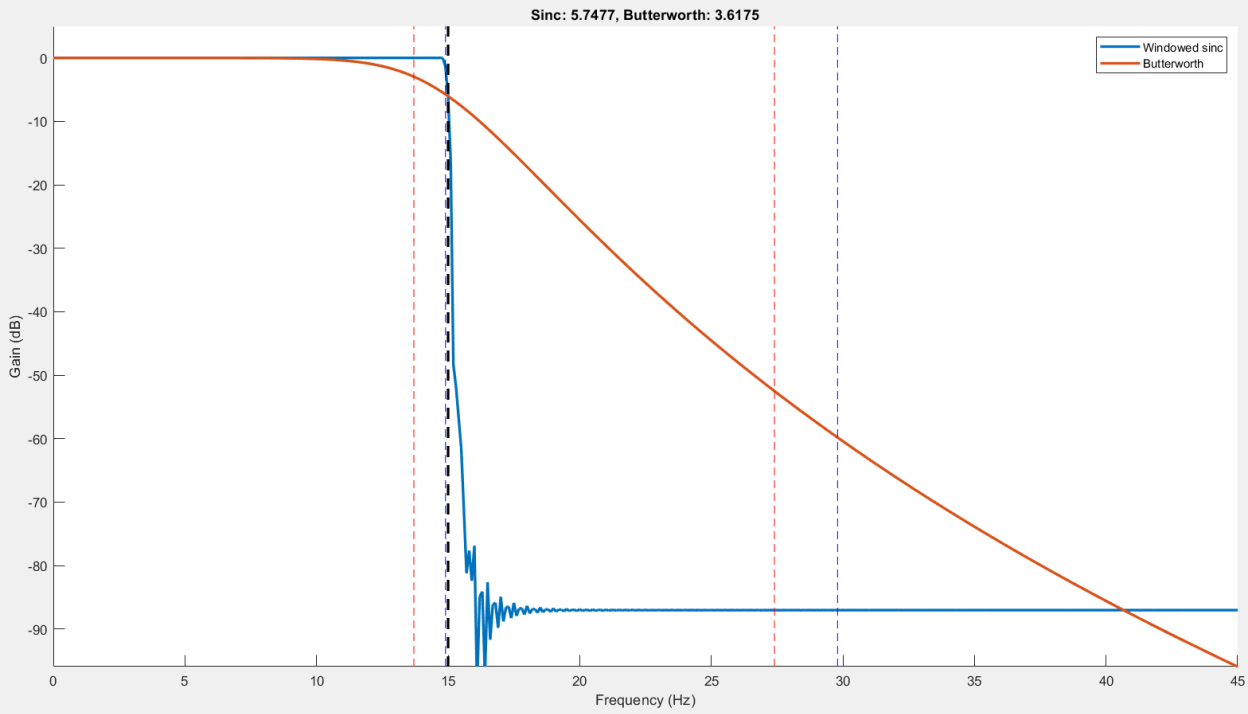
45. QUANTIFYING ROLL-OFF CHARACTERISTICS (WINDOWED SINC VS BUTTERWORTH)

Roll-off is not always supposed to be as steep as possible. A sharp roll-off can introduce ripples into the filtered data. Not all filters have a sensible roll-off, for example the roll-off could be too steep.

$$\text{Roll-off} = \frac{g_{2f} - g_{-3}}{Hz_{2f} - Hz_f}$$

MATLAB

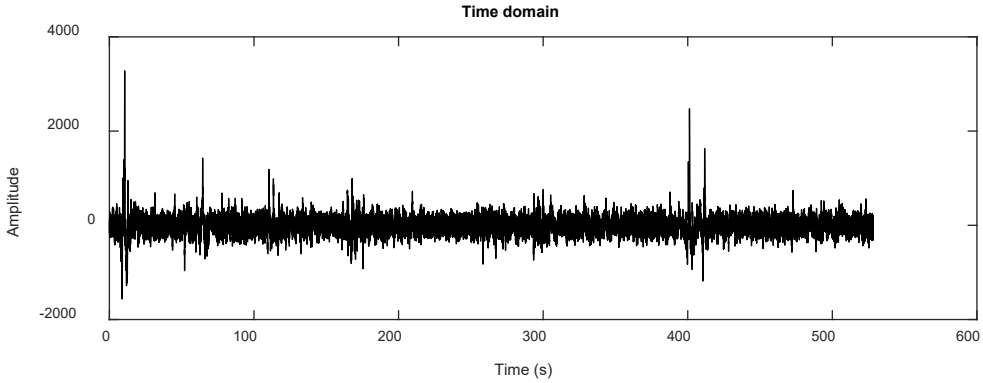
Code: sigprocMXC_rolloff.m

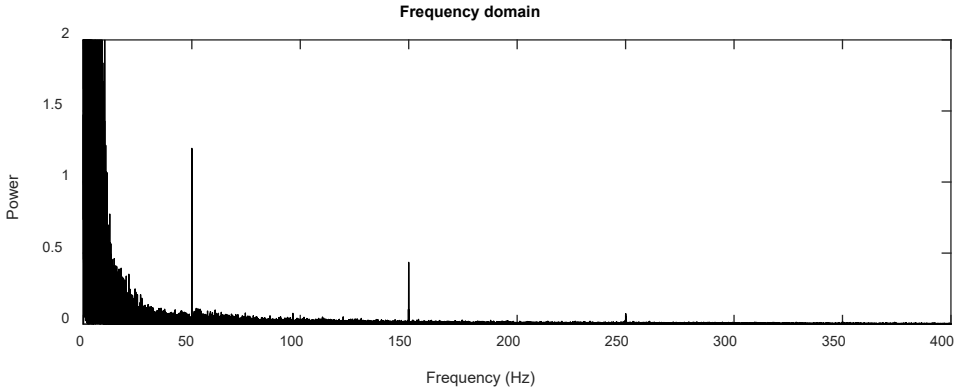


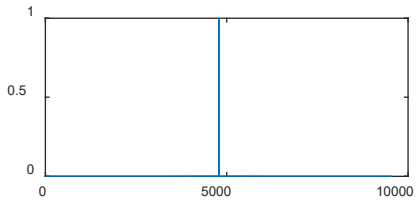
46. REMOVE ELECTRICAL LINE NOISE AND ITS HARMONICS

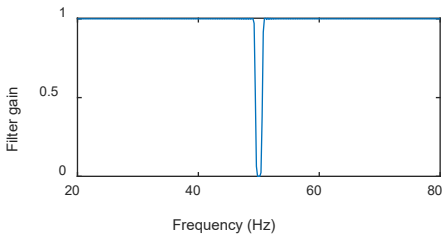
MATLAB

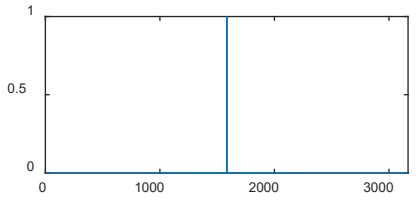
Code: sigprocMXC_linenoise.m

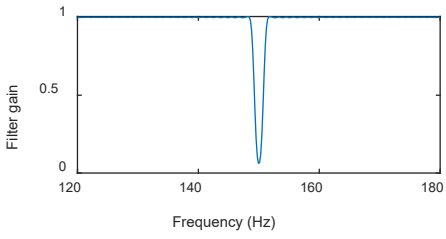

Time domain
Amplitude
Time (s)

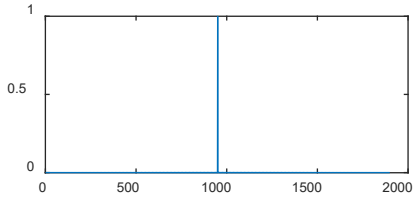

Frequency domain
Power
Frequency (Hz)

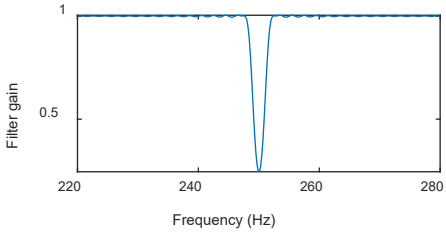


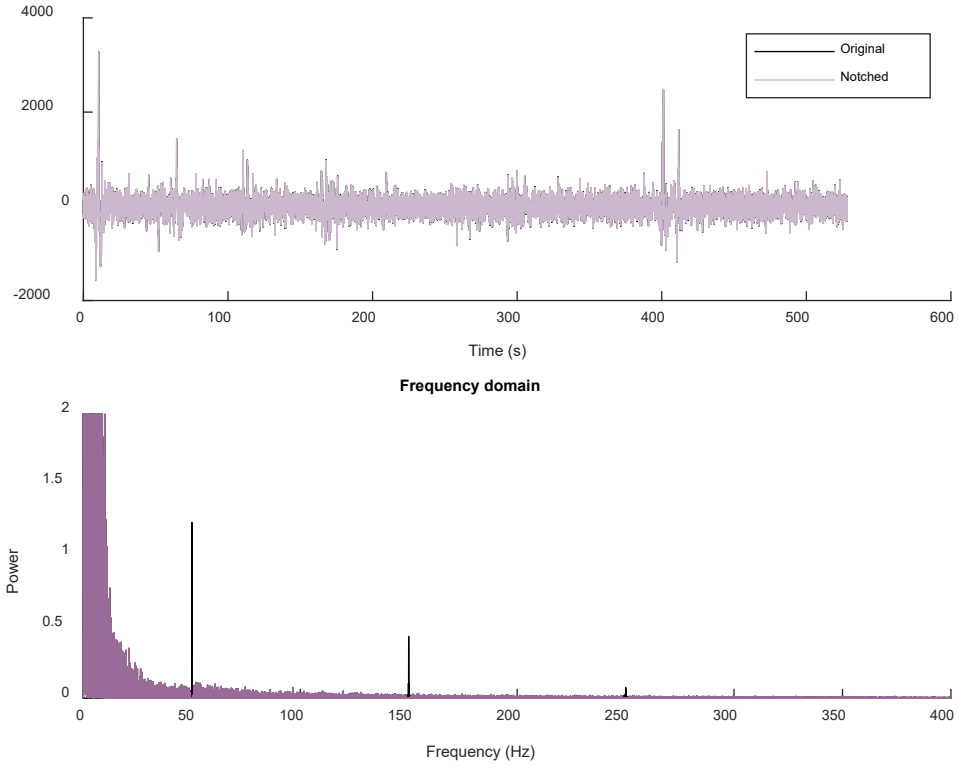

Filter gain
Frequency (Hz)




Filter gain
Frequency (Hz)




Filter gain
Frequency (Hz)



47. USE FILTERING TO SEPARATE BIRDS IN A RECORDING

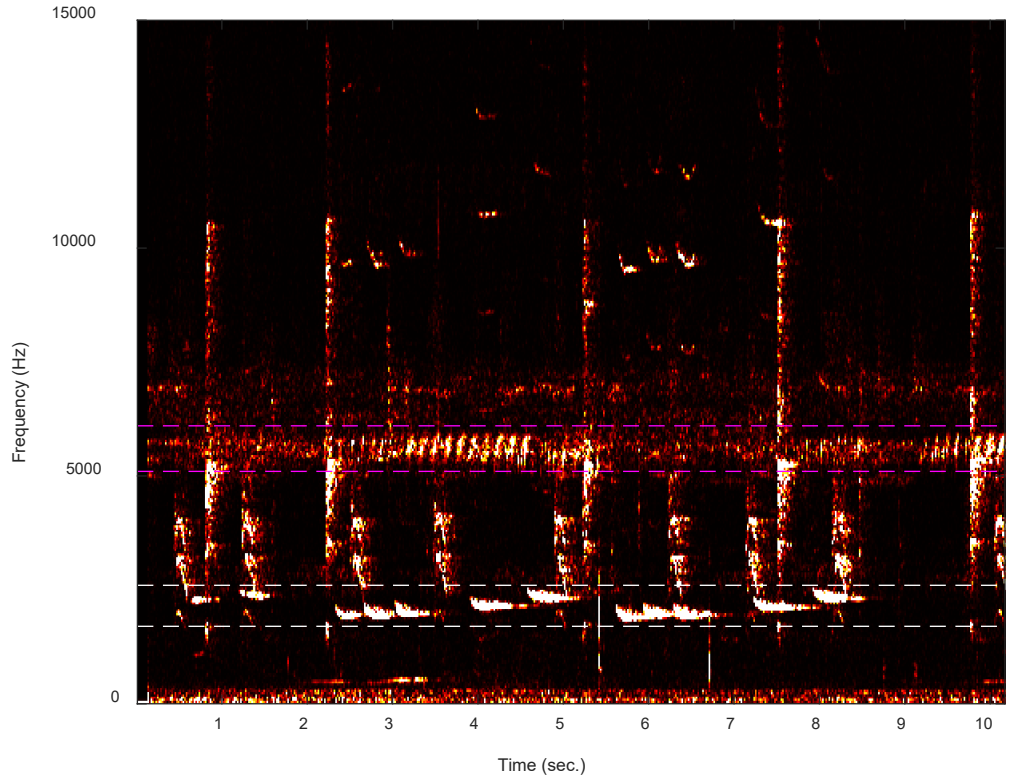
MATLAB

Code: sigprocMXC_filterTheBirds.m

Refer back to section 3:

For the following code:

- sigprocMXC_SpectBirdcall.m
- XC403881.mp3



48. CODE CHALLENGE: FILTER THESE SIGNALS!

Find a set of filters to match the time and frequency domain black lines

MATLAB

Data: filtering_codeChallenge.mat

SECTION 6: CONVOLUTION

Code Directory: D:\Ron\Documents\Ron Fredericks Consulting\Education\Signal processing problems - Udemy\Section 06\sigprocMXC-convolution

50. TIME-DOMAIN CONVOLUTION

Convolution: a way to combine two time series (or images) to create a third time series

Signal: The “interesting” time series

Kernel: The filter

Convolution result: A mixture of the features of the signal and the kernel

Example in time domain:

Dot product:

- The most important computation in linear algebra and signal processing
- a single number, scaler
- something about the relationship or mapping of two different time series
- a transpose b, where a and b are column vectors with same number of elements
- element-wise summing of individual elements
-

Two ways to show Dot Product:

$$\begin{pmatrix}1\\3\\0\\5\end{pmatrix}\cdot\begin{pmatrix}0\\8\\1\\6\end{pmatrix}=1*0+3*8+0*1*5*6=54$$

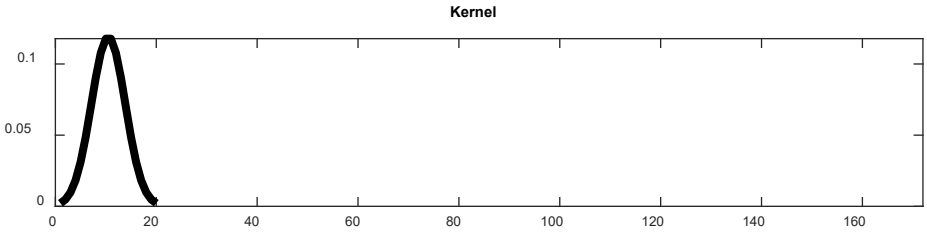
Or

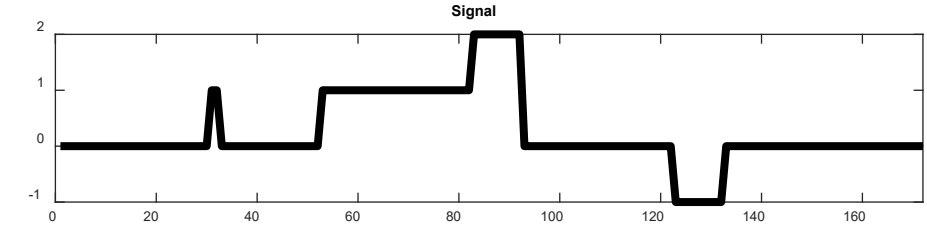
$$\alpha=a\cdot b=<a,b>=\ a^Tb=\sum_{i=1}^{n}a_ib_i$$

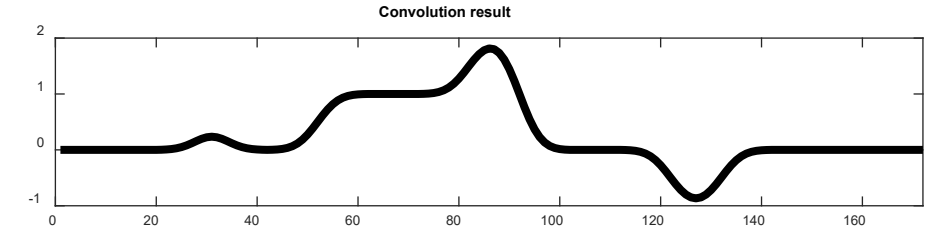
51. CONVOLUTION IN MATLAB

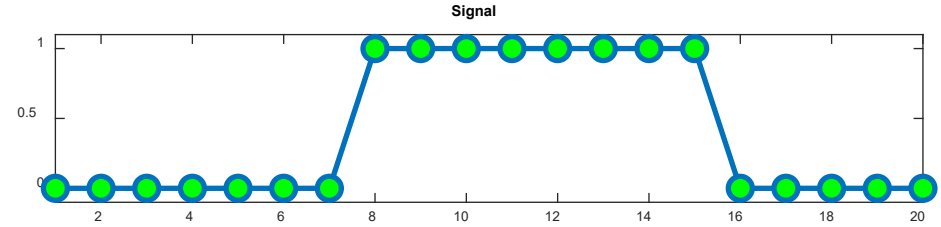
MATLAB

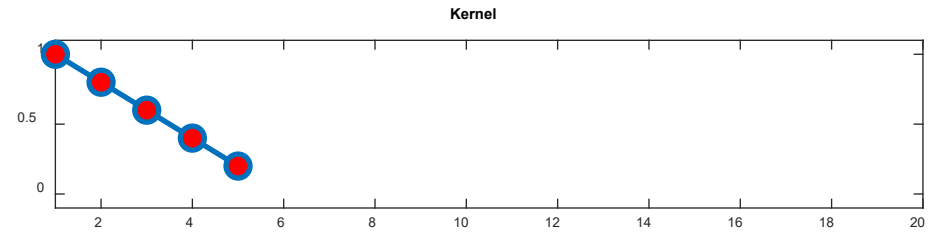
Code: sigprocMXC_timeConvolution.m

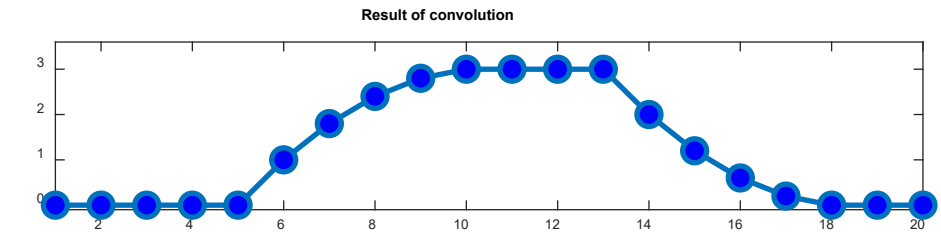

Kernel
0.1
0.05
0
0
20
40
60
80
100
120
140
160

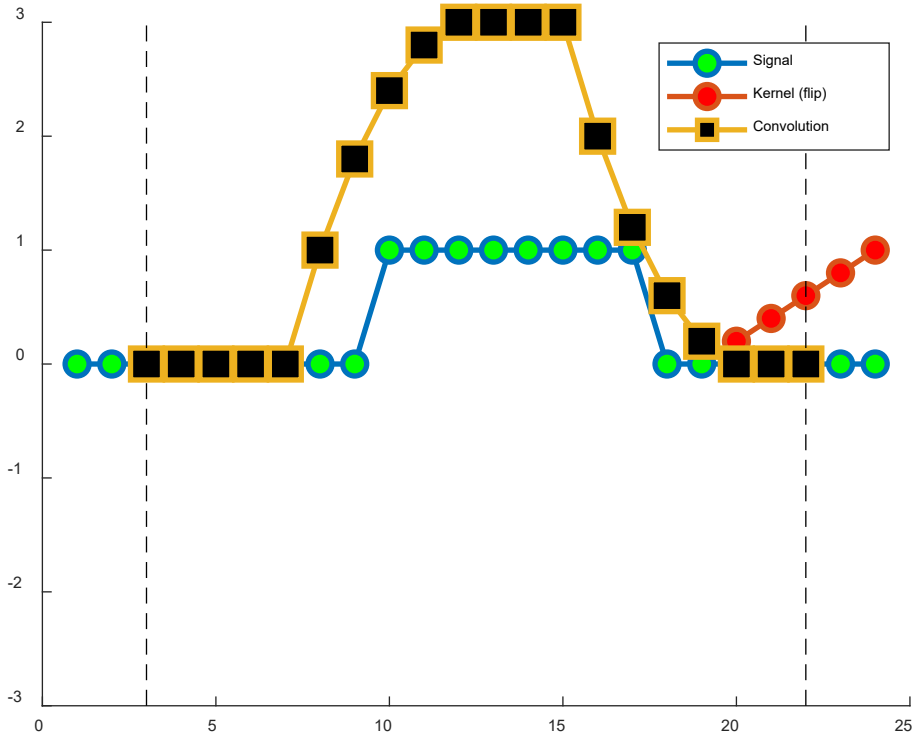

Signal
2
1
0
-1
0
20
40
60
80
100
120
140
160

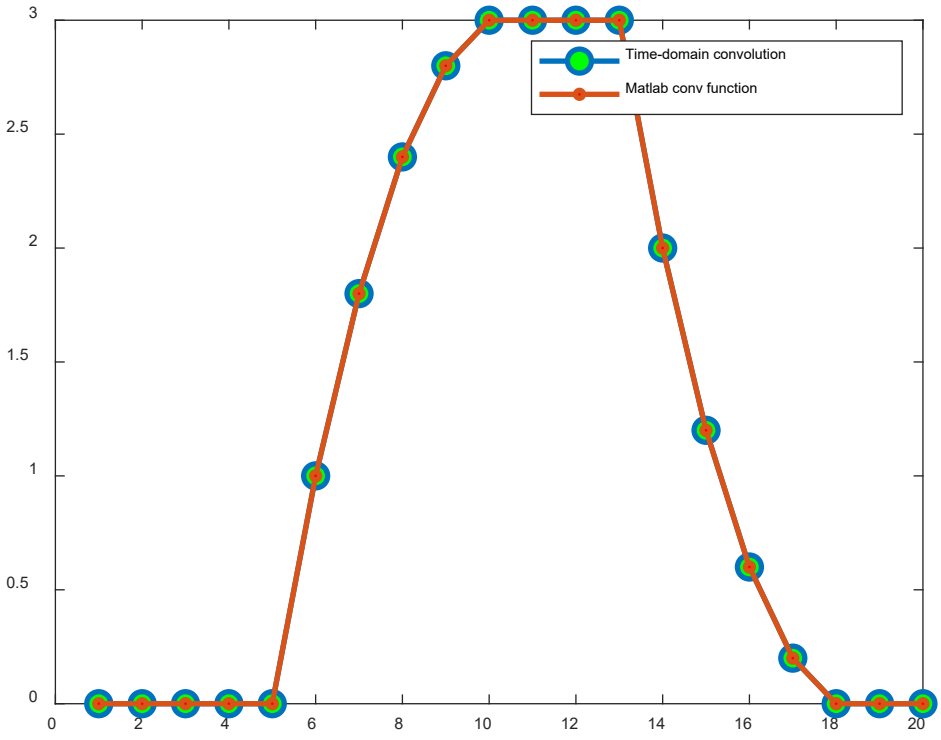

Convolution result
2
1
0
-1
0
20
40
60
80
100
120
140
160


Signal
1
0.5
0
2
4
6
8
10
12
14
16
18
20


Kernel
0.5
0
2
4
6
8
10
12
14
16
18
20


Result of convolution
3
2
1
0
2
4
6
8
10
12
14
16
18
20


Signal
Kernel (flip)
Convolution
3
2
1
0
-1
-2
-3
0
5
10
15
20
25


Time-domain convolution
Matlab conv function
3
2.5
2
1.5
1
0.5
0
0
2
4
6
8
10
12
14
16
18
20

52. WHY IS THE KERNEL FLIPPED BACKWARDS??!!

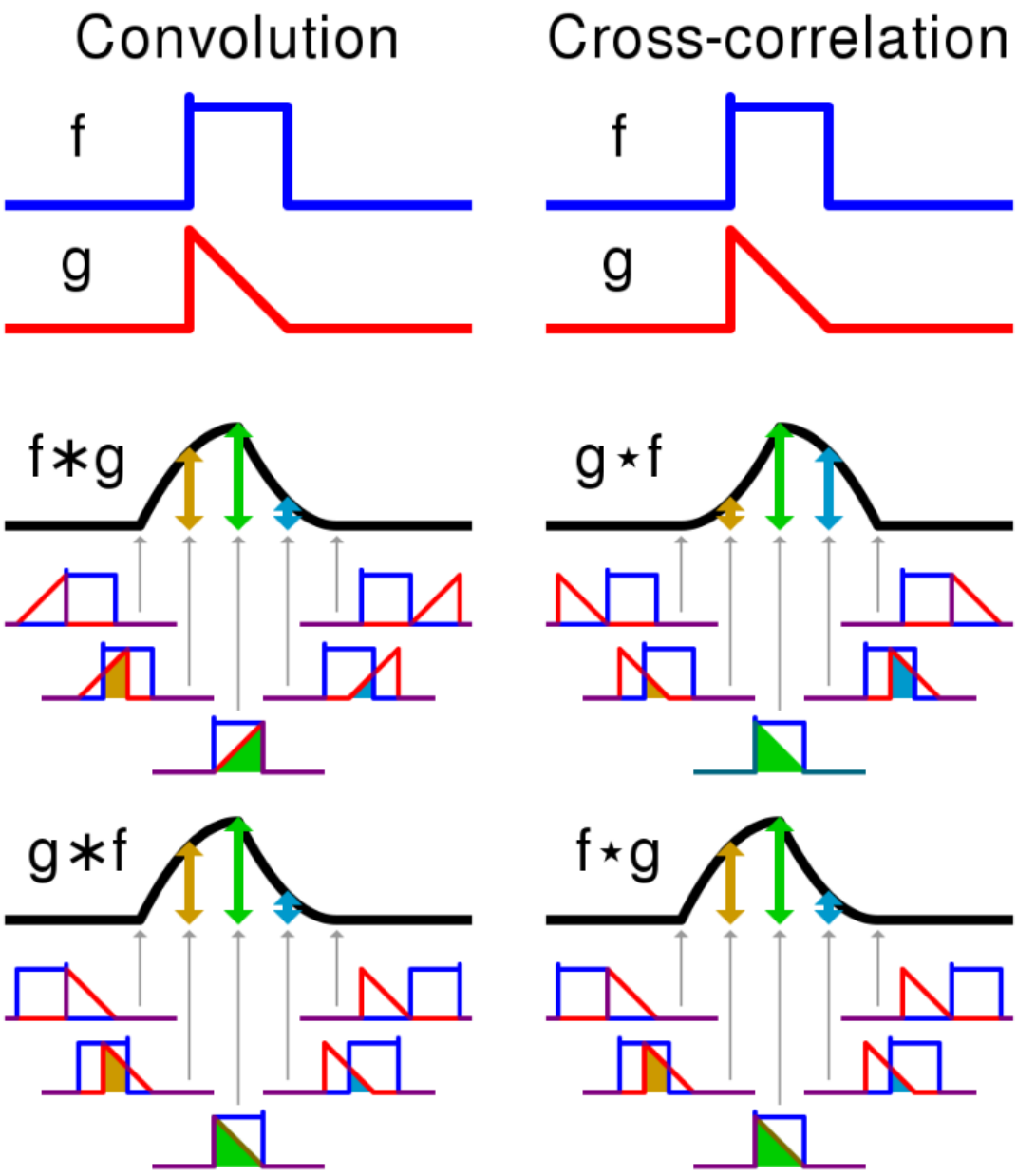
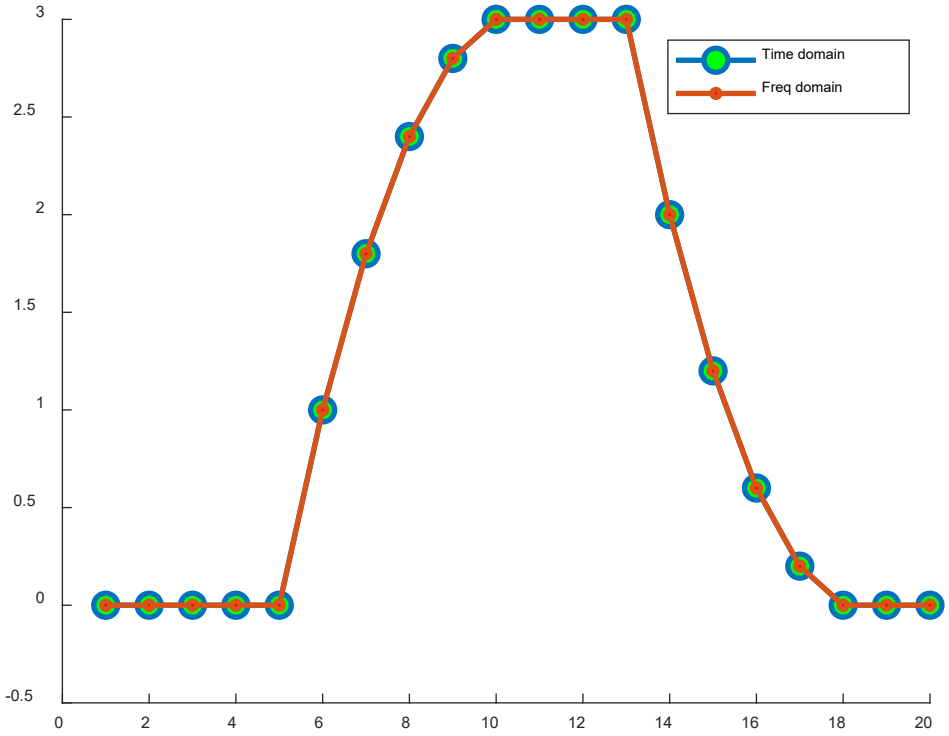


Figure reference: https://en.wikipedia.org/wiki/Convolution

53. THE CONVOLUTION THEOREM

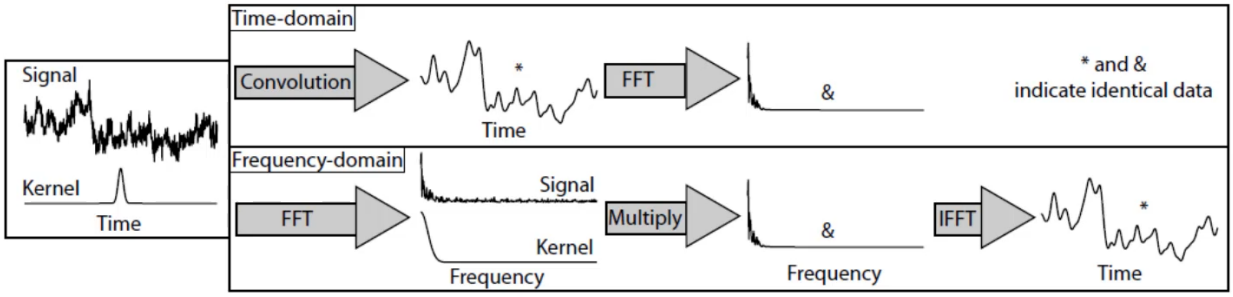
Code: sigprocMXC_convolutionTheorem.m



54. THINKING ABOUT CONVOLUTION AS SPECTRAL MULTIPLICATION

Convolution in the time domain is the equivalent to multiplication in the frequency domain. Useful for:

- Signal process
- Spectral analysis
- Time Frequency analysis
- Narrow band filtering
- Wavelet analyses



Reference: “Convolution Theorem” https://dartbrains.org/features/notebooks/6_Signal_Processing.html

Time-domain convolution is a time series of dot products by repeatedly computing dot product between kernel and signal. Followed by FFT to get the power spectrum.

Frequency domain convolution is an element wise multiplication between FFT of signal and FFT of kernel to get power spectrum. Followed by IFFT to bring convolution into the time domain. This method is faster than time-domain convolution and offers a new way to conceptualize convolution.

Difference between narrow and wider Gaussian Kernel.

Edge effects are caused by original 20 Hz sine wave not begin an integer number fit within the time window.

Result of convolution is a low pass filter.

Morlet wavelet. A sine wave multiplied by a Gaussian.

Example of a narrow band filter

TIME-FREQUENCY ANALYSIS WITH COMPLEX MORLEY WAVELETS

Video: TFtheory.mp4

Complex sine wave. Use Euler's formula/notation to describe the complex sine wave.

$$e^{ik} = \cos(k) + i\sin(k)$$

$$k = 2\pi ft + \theta$$

$$e^{2\pi ft}$$

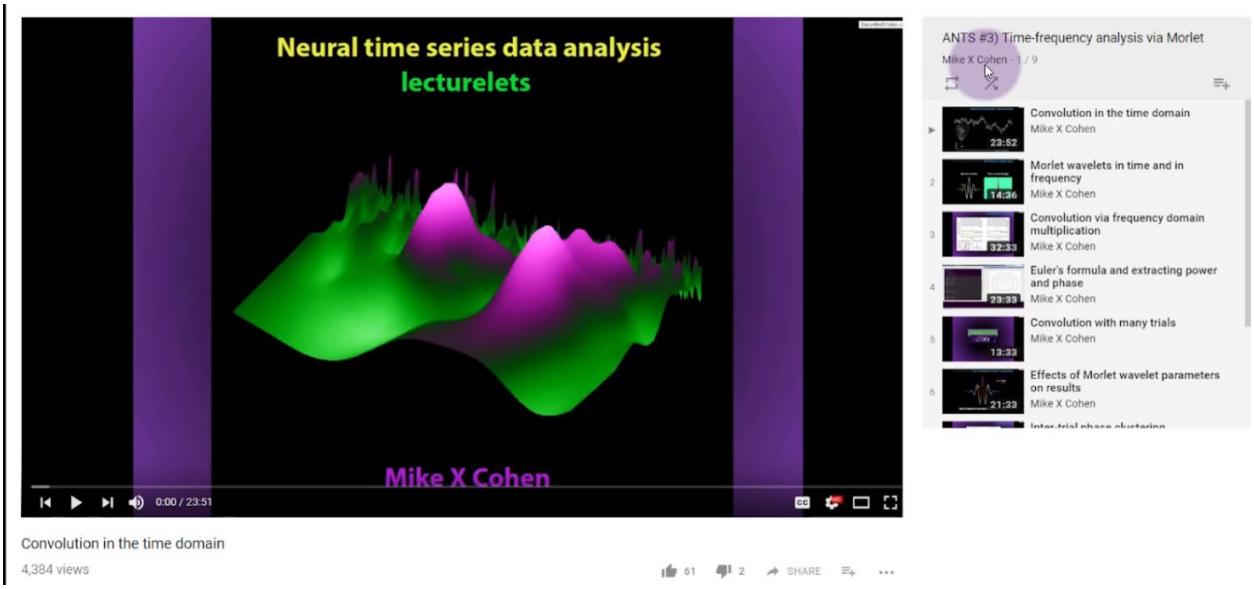
A complex sine wave modulated by a gaussian

The dot product has real and imaginary parts

D1) Real Part: Projection onto the real axis – a bandpass filtered signal – see lecture on wavelet narrow band-pass filtering

D2) Magnitude: produces time-frequency power (spectrogram)

D3) Phase Angle: Angle of the vector



https://mitpress.mit.edu/books/analyzing-neural-time-series-data

http://mikexcohen.com/lectures.html

https://sccn.ucsd.edu/eeglab/index.php

http://www.fieldtriptoolbox.org/documentation/

55. CONVOLUTION WITH TIME-DOMAIN GAUSSIAN (SMOOTHING FILTER)

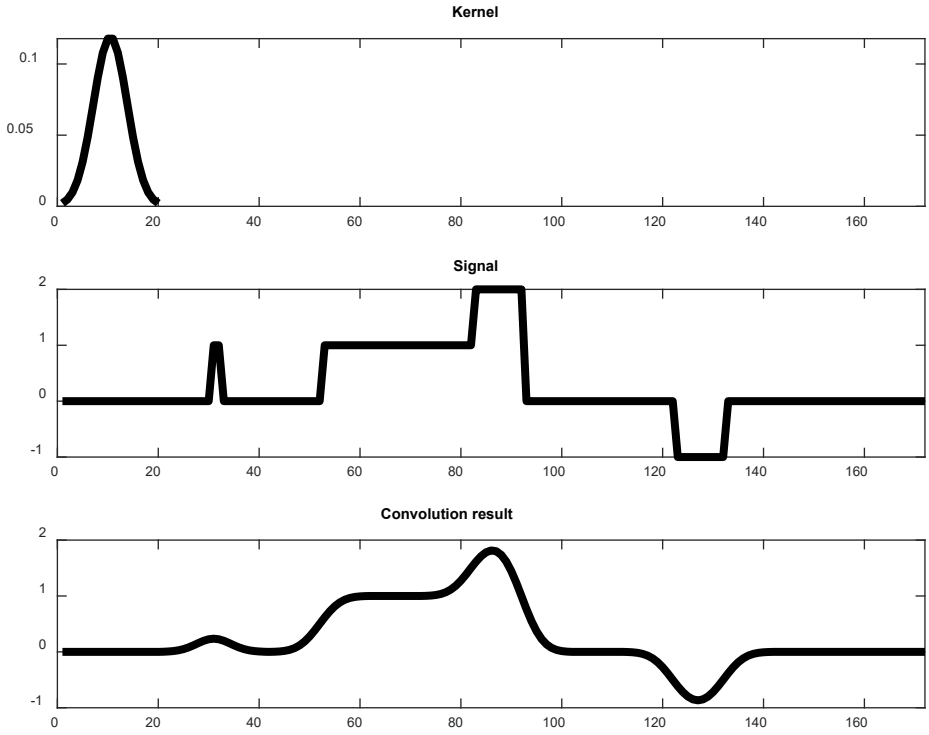
Reference: Lecture# 10. GAUSSIAN-SMOOTH A TIME SERIES

Create Gaussian Kernel defining full-width half-maximum: fwhm set to 25%

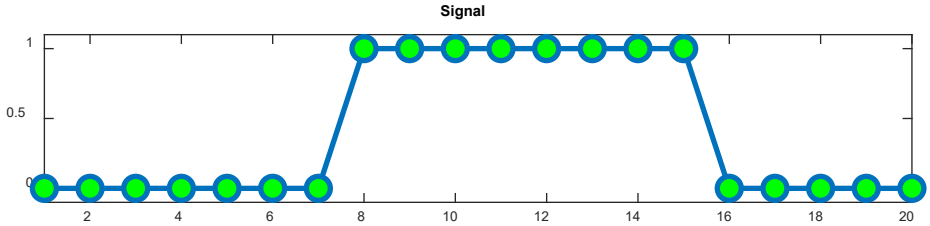
Reduce edge effects with this method

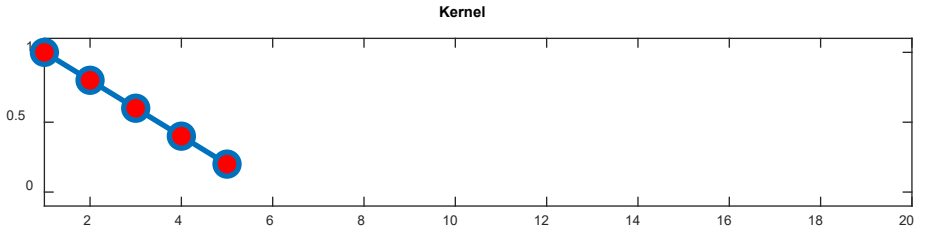
MATLAB

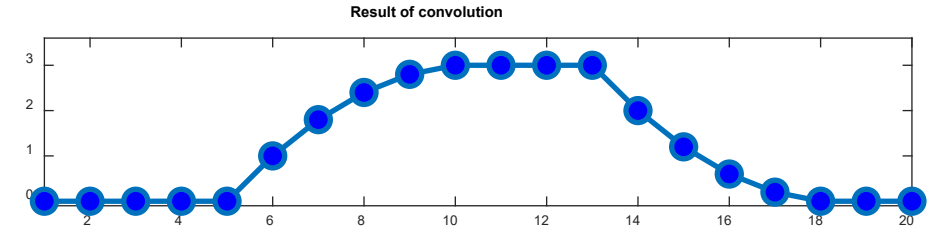
Code: sigprocMXC_timeConvolution.m

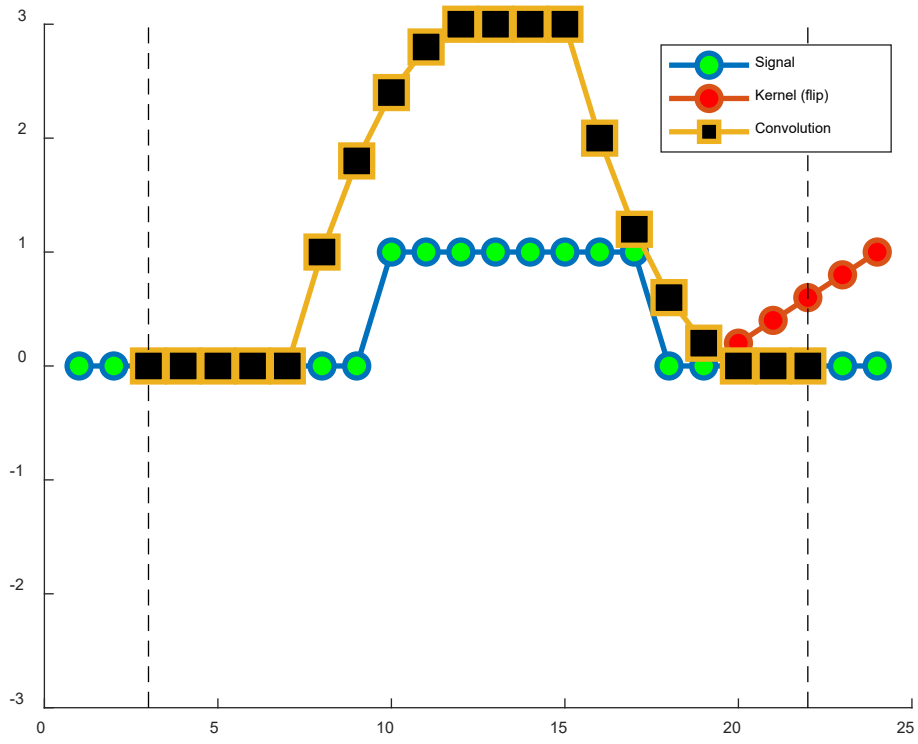


The wider the kernel, the more “smearing” of the convolution result. When the kernel is too narrow, the signal and convolution result look nearly the same.

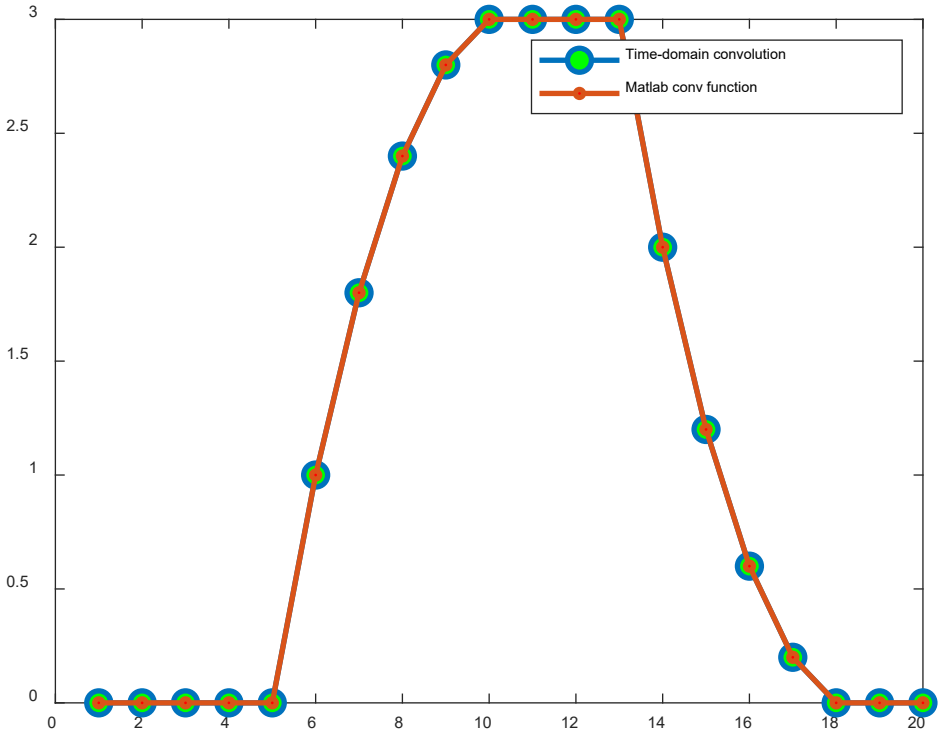

Signal
1
0.5
0
2
4
6
8
10
12
14
16
18
20


Kernel
0.5
0
2
4
6
8
10
12
14
16
18
20


Result of convolution
3
2
1
0
2
4
6
8
10
12
14
16
18
20


Signal
Kernel (flip)
Convolution
3
2
1
0
-1
-2
-3
0
5
10
15
20
25

Nice Animation



56. CONVOLUTION WITH FREQUENCY-DOMAIN GAUSSIAN (NARROWBAND FILTER)

A time domain gaussian is always produces a low-pass filter after convolution

A frequency-domain gaussian produces a narrowband filter after convolution back to the time domain

$$g = e^{-.5(\frac{(h-p)}{s})^2}$$

$$s = \frac{w(2\pi - 1)}{4\pi}$$

h: frequencies (Hz)

p: peak frequency (Hz)

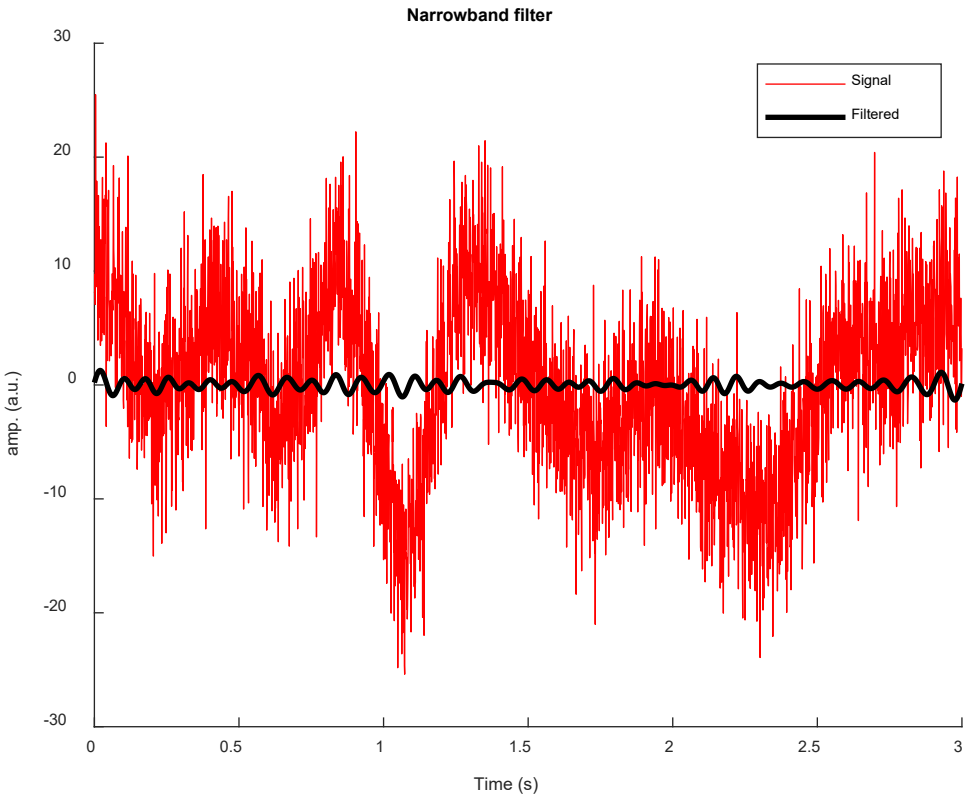
w: FWHM (full width half maximum) (Hz)

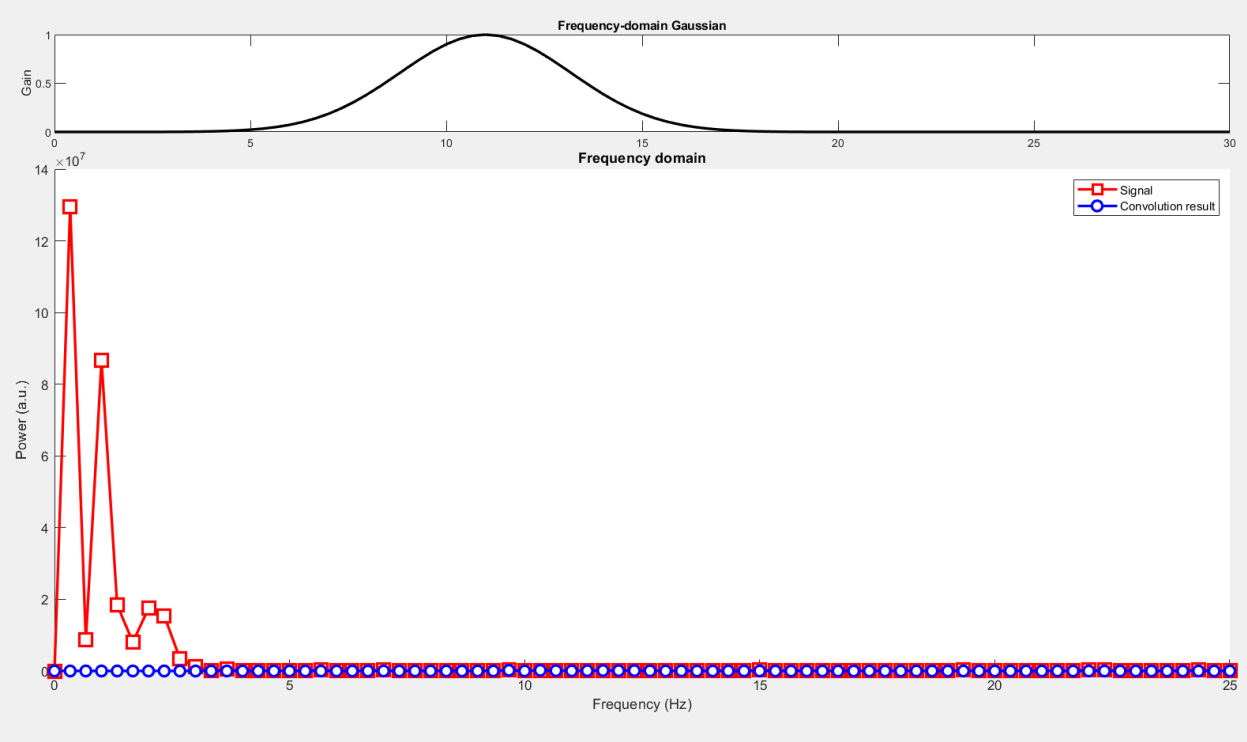
MATLAB

Code: sigprocMXC_FreqDomainGaus.m

Code snippet showing frequency gaussian:

```
plot(hz,fx)
set(gca,'xlim',[0 30])
```





Zoom in on the narrowband filter frequencies to see that the filter is useful and works.

57. CONVOLUTION WITH FREQUENCY-DOMAIN PLANK TAPER (BANDPASS FILTER)

$$\begin{cases} \dfrac{1}{e^{z_l+1}}, & 0 < t < N-1 \\ 1, & \epsilon(N-1) \le t \le (1-\epsilon)(N-1) \\ \dfrac{1}{e^{z_r+1}}, & (1-\epsilon)(N-1) < t < N-1 \end{cases}$$

$$z_l = \epsilon(N-1)\left(\frac{1}{t} + \frac{1}{t-\epsilon(N-1)}\right)$$

$$z_r = \epsilon(N-1)\left(\frac{1}{N-1-t} + \frac{1}{(1-\epsilon)(N-1)-t}\right)$$

$$0 < \epsilon \le .5$$

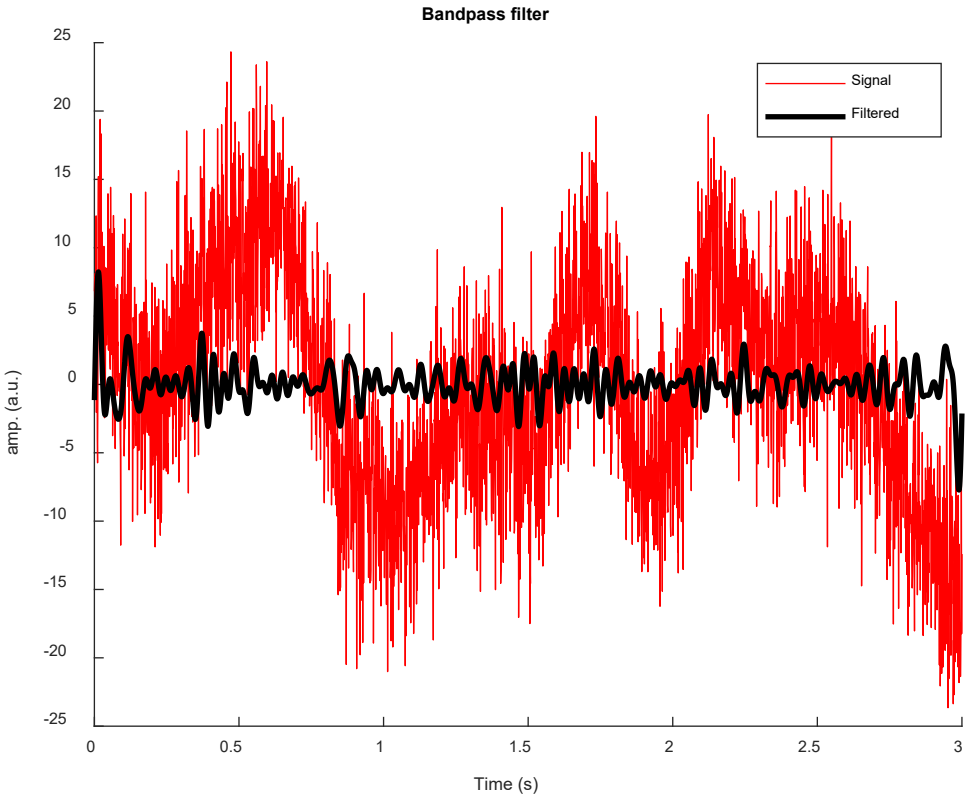
Left side of the Plank taper is a sigmoid function

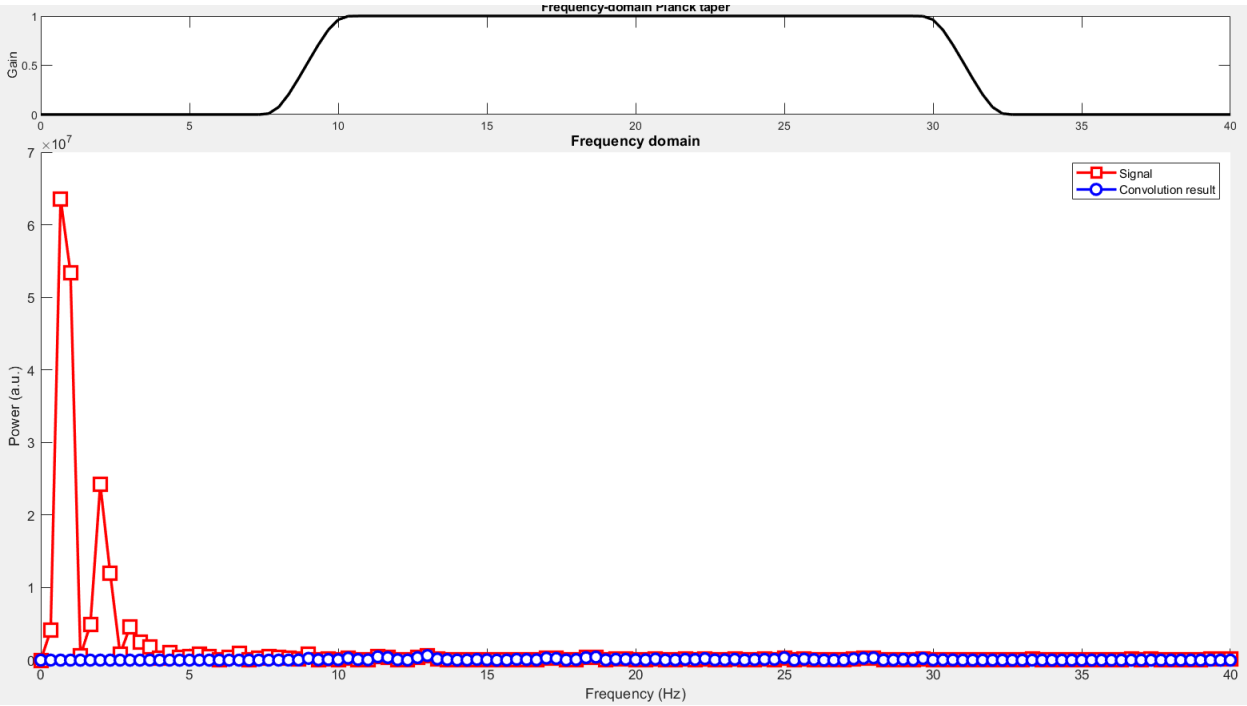
Right side of the Plank taper is an inverse sigmoid function

Middle piece of the piece-wise function is just 1, for a flat top

MATLAB

Code: sigprocMXC_planckBandPass.m





Zoom in on the bandpass filter frequencies to see that the filter is useful and works.

58. CODE CHALLENGE: CREATE A FREQUENCY-DOMAIN MEAN-SMOOTHING FILTER

Start with the Time Series Denoising mean smooth MATLAB code, replace the code snippet “implement the funning mean filter” loop in the frequency domain.

Come up with the kernel, take the fft of the kernel times the fft of the signal, then take ifft of that result. The final filtered signal should look nearly the same as the running-mean filter result. Edge effects will be different.

MATLAB

Code to start with:

Section 02\sigprocMXC-TimeSeriesDenoising\sigprocMXC_mean_smooth.m

SECTION 7: WAVELET ANALYSIS

60. WHAT ARE WAVELETS?

Wavelets:

- Morlet – sine wave multiplied by a Gaussian: time frequency analysis
- Haar
- Difference of Gaussian (DoG)

- Daubechies 4 4 tap
- Mexican Hat
- Coilet – like an FIR filter

Properties:

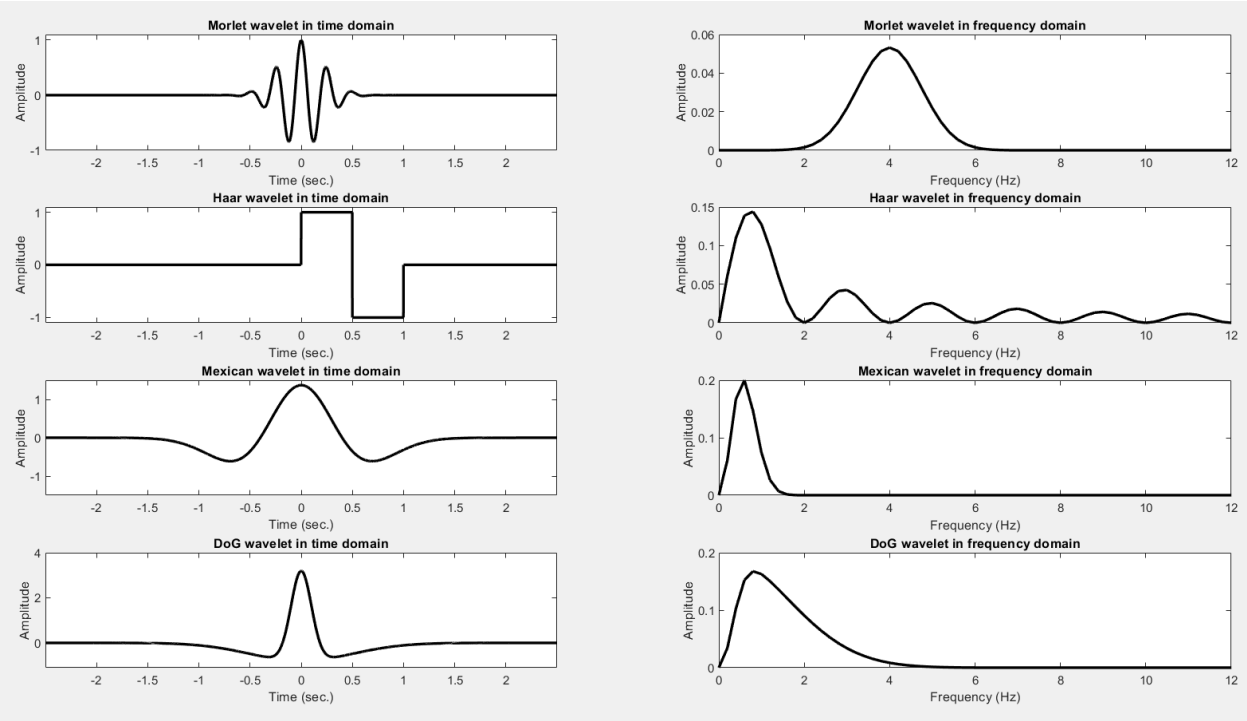
- Taper down to zero at beginning and end
- Integrate to one

Applications

- Filtering (time-frequency analysis
- Feature detection (pattern matching)

MATLAB

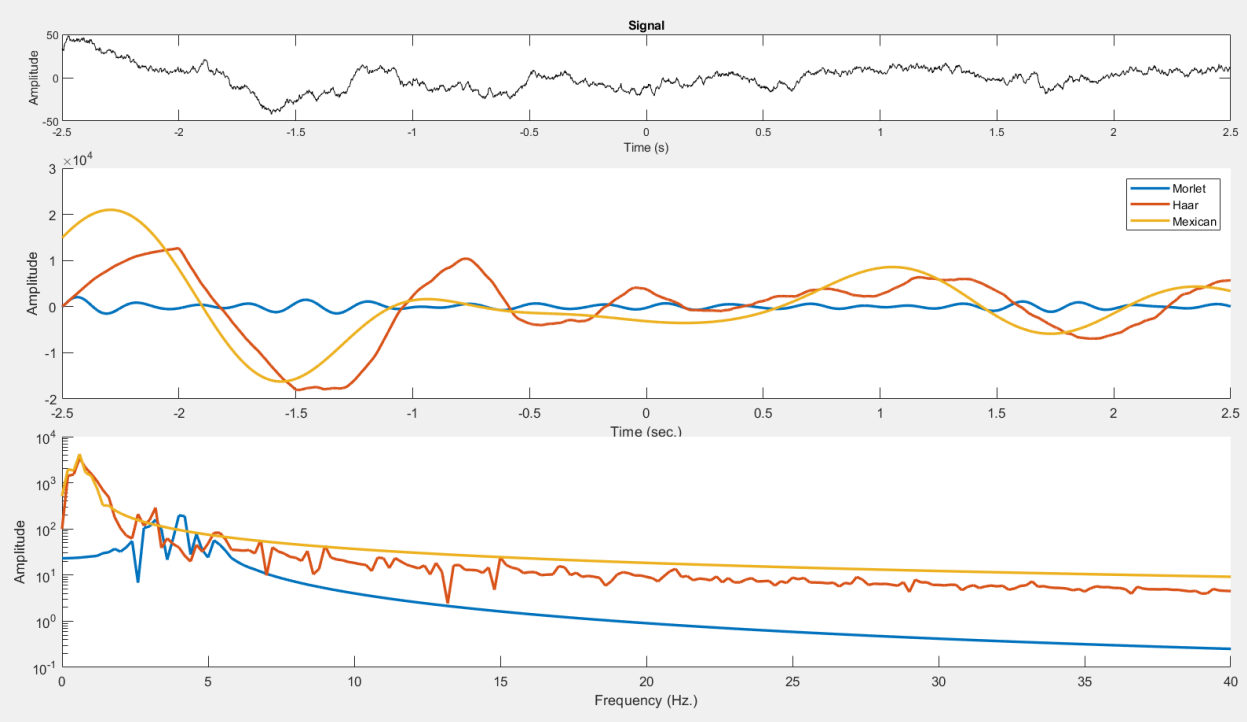
Code: sigprocMXC_wavelets.m



61. CONVOLUTION WITH WAVELETS

MATLAB

Code: sigprocMXC_waveletConv.m



62. SCIENTIFIC PUBLICATION ABOUT DEFINING MORLET WAVELETS

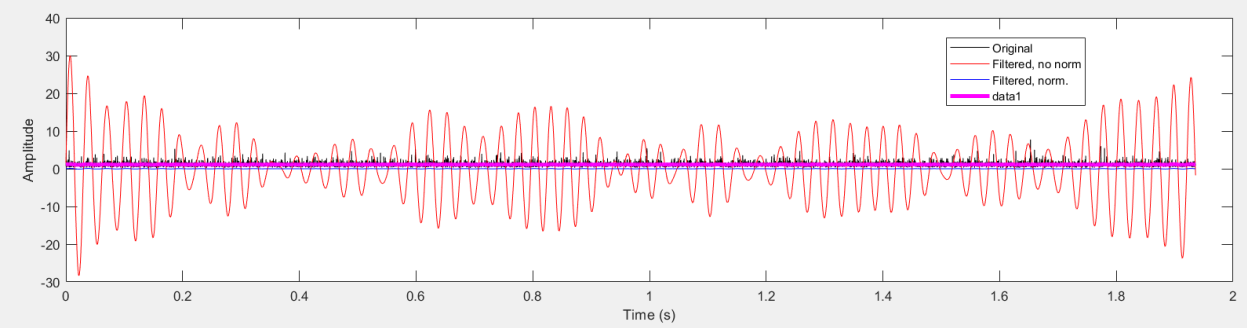
PUBLICATION: A BETTER WAY TO DEFINE AND DESCRIBE MORLET WAVELETS FOR TIME-FREQUENCY ANALYSIS

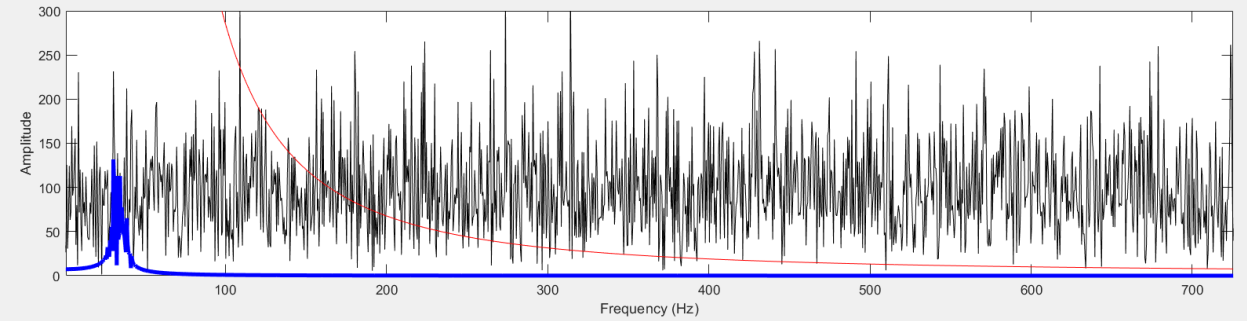
https://www.biorxiv.org/node/120256.full

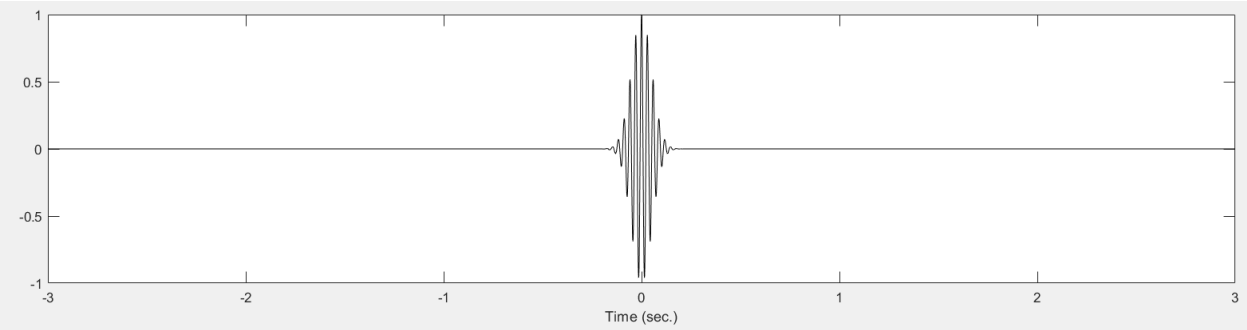
63. WAVELET CONVOLUTION FOR NARROWBAND FILTERING

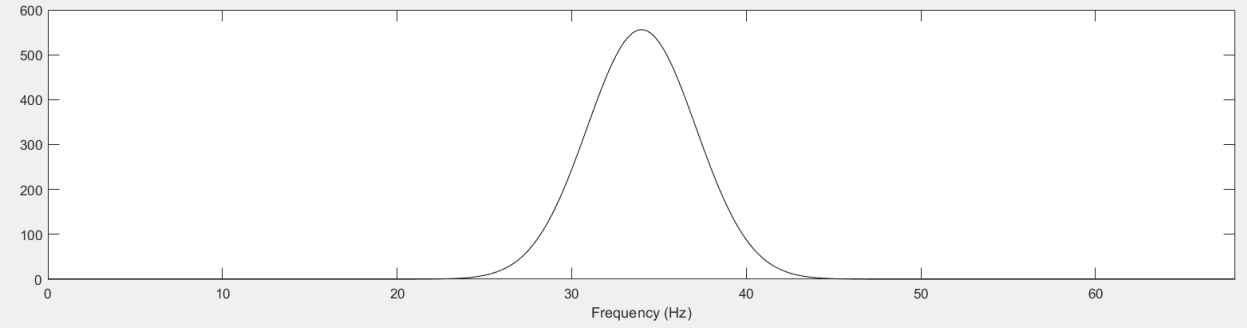
MATLAB

Code: sigprocMXC_wavelets4narrowband.m









Create a curve from the envelop of signal using Hilbert transform

64. OVERVIEW: TIME-FREQUENCY ANALYSIS WITH COMPLEX MORLET WAVELETS

Same exact material as section 6 lecture 50: Time domain convolution

65 LINK TO YOUTUBE CHANNEL WITH 3 HOURS OF RELEVANT MATERIAL

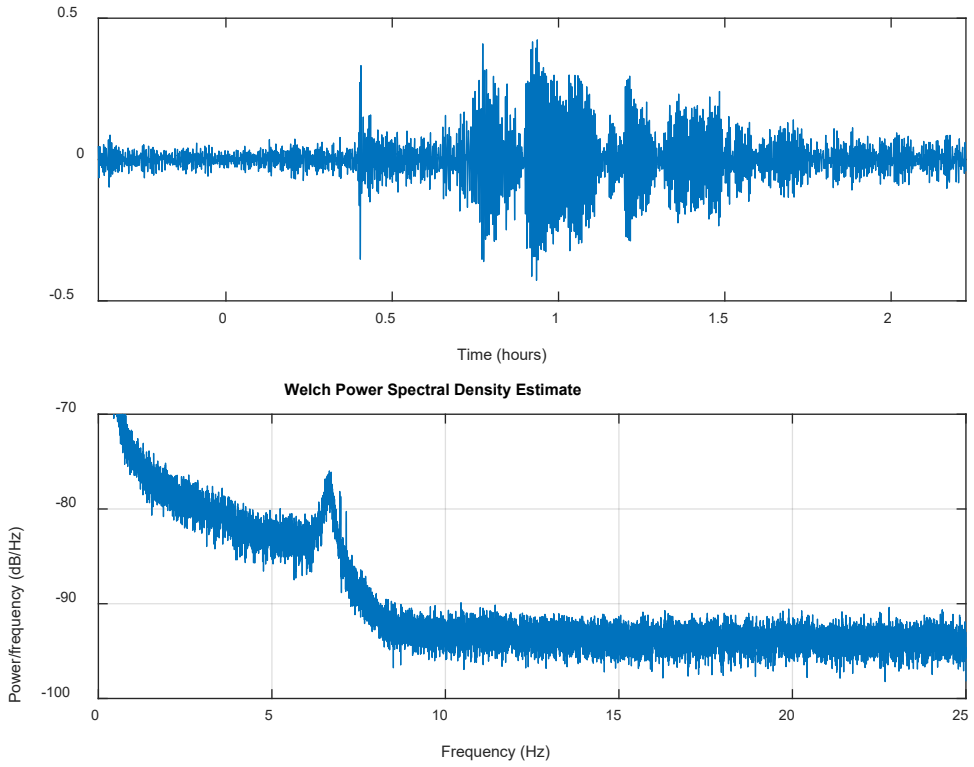
https://www.youtube.com/watch?v=HSMwxBg7iq4&list=PLn0OLiymPak2G__qvavn3T8k7R8ssKxVr

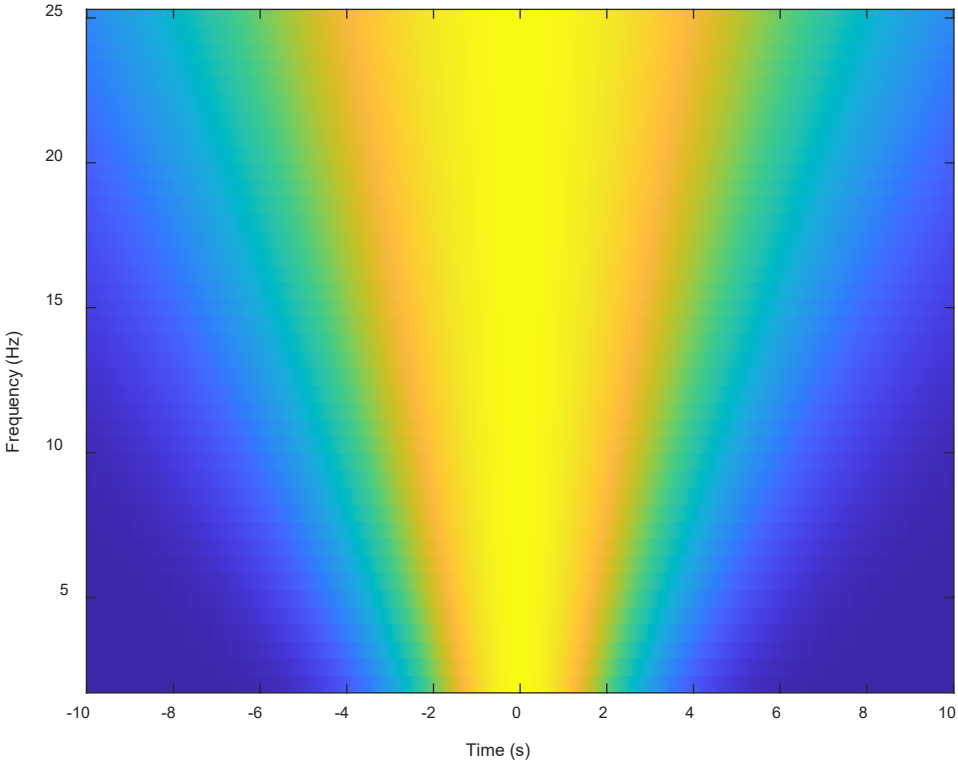
66 MATLAB: TIME-FREQUENCY ANALYSIS WITH COMPLEX WAVELETS

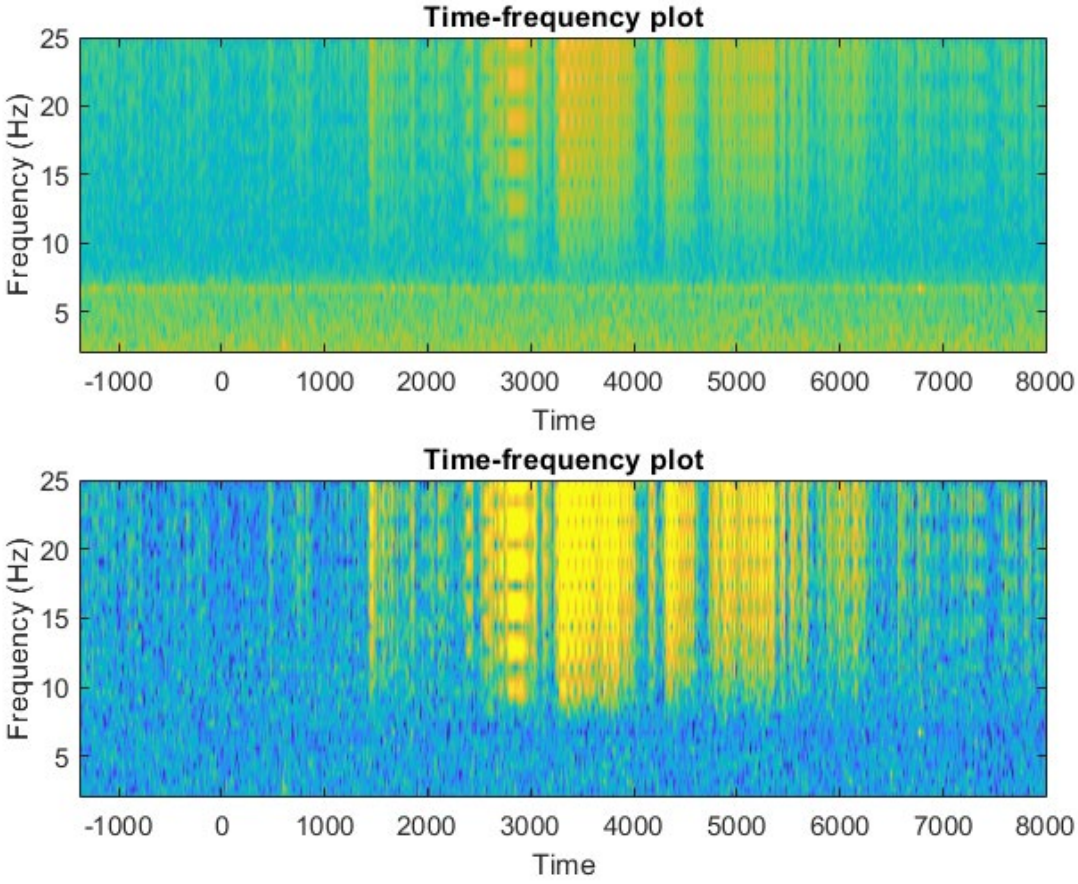
MATLAB

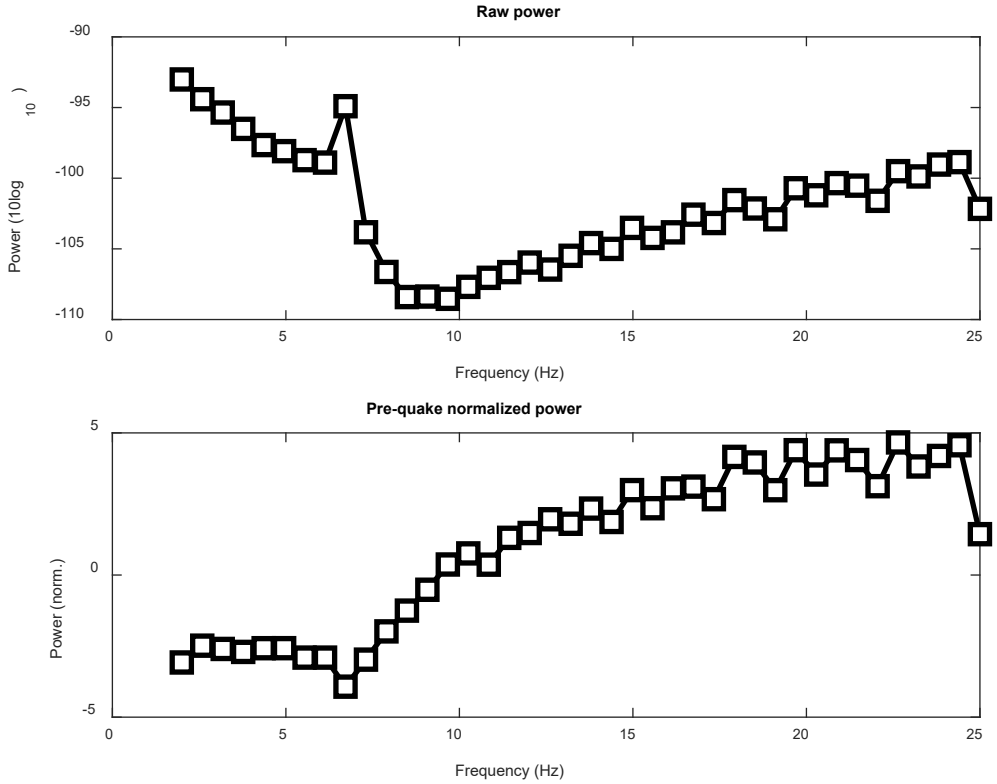
Code: sigprocMXC_waveletTF.m

Data: http://www.vibrationdata.com/Solomon_Time_History.zip




25
20
15
10
5
Frequency (Hz)
-10
-8
-6
-4
-2
0
2
4
6
8
10
Time (s)

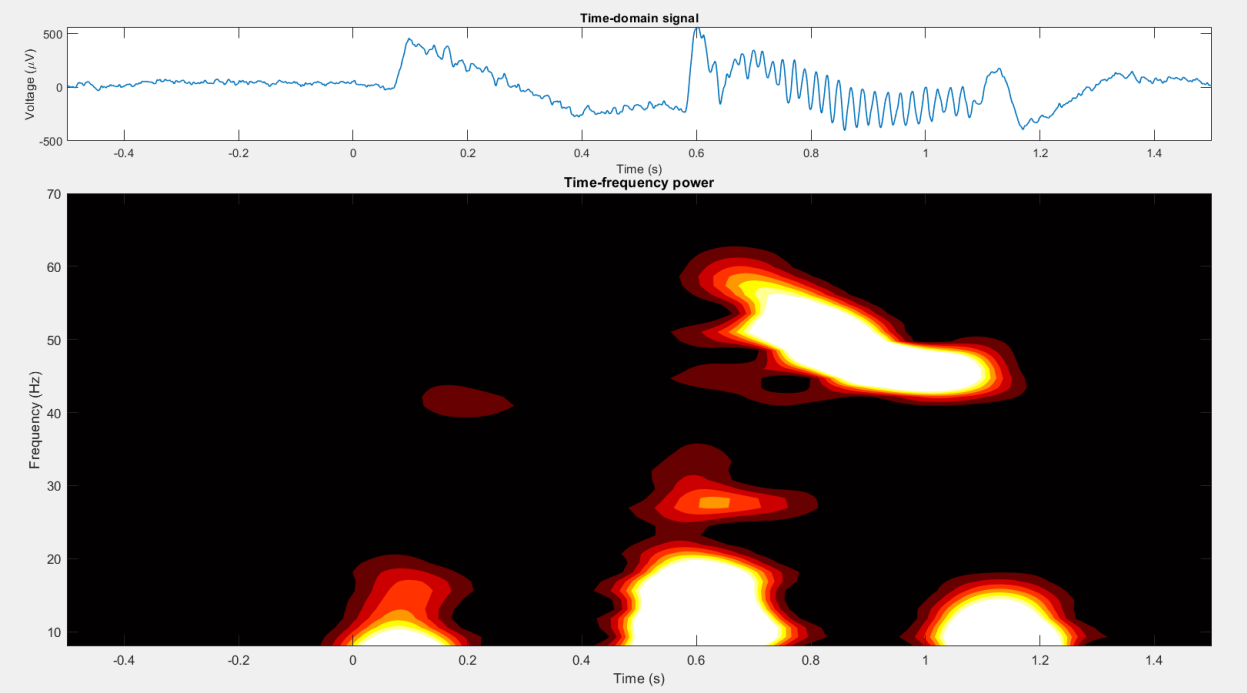

Time-frequency plot
Frequency (Hz)
25
20
15
10
5
-1000
0
1000
2000
3000
4000
5000
6000
7000
8000
Time
Time-frequency plot
Frequency (Hz)
25
20
15
10
5
-1000
0
1000
2000
3000
4000
5000
6000
7000
8000
Time

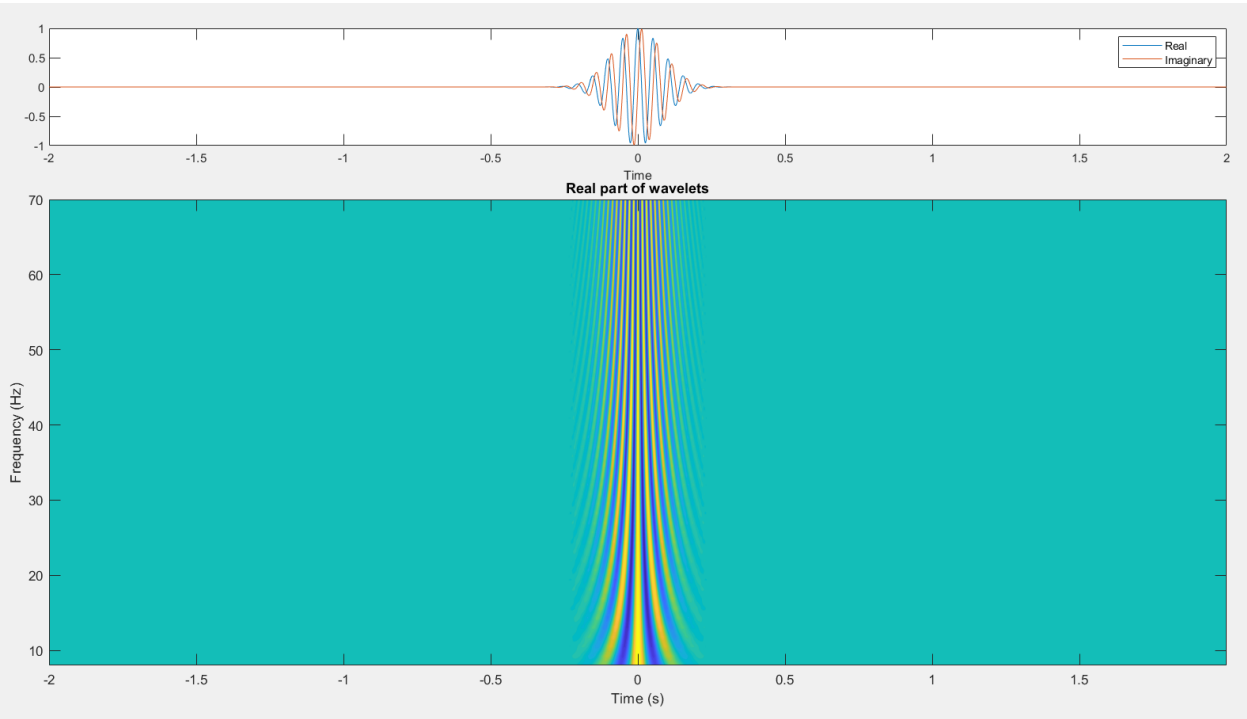


67. TIME-FREQUENCY ANALYSIS OF BRAIN SIGNALS

MATLAB

Code: sigprocMXC_timefreqBrain.m





68. CODE CHALLENGE: COMPARE WAVELET CONVOLUTION AND FIR FILTER!

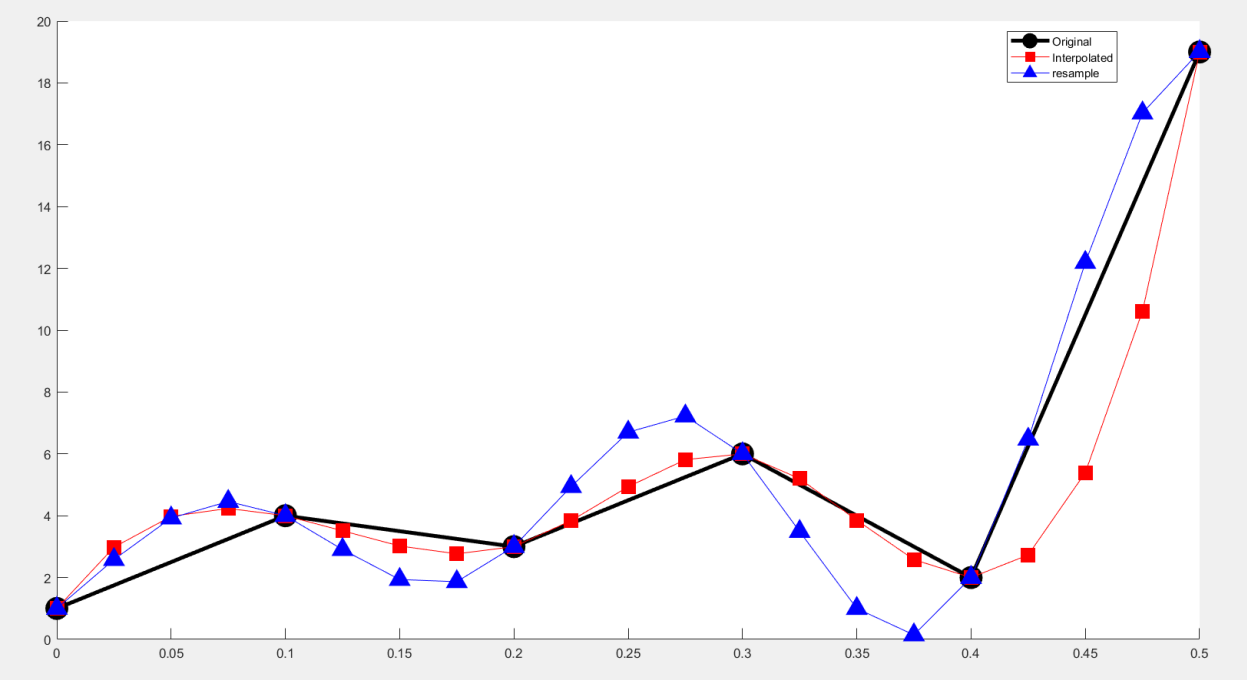
SECTION 8: RESAMPLING, INTERPOLATING, EXTRAPOLATING

70. UPSAMPLING

Starting from a low sampling rate -> get a higher sampling rate

MATLAB

Code: sigprocMXC_upsample.m

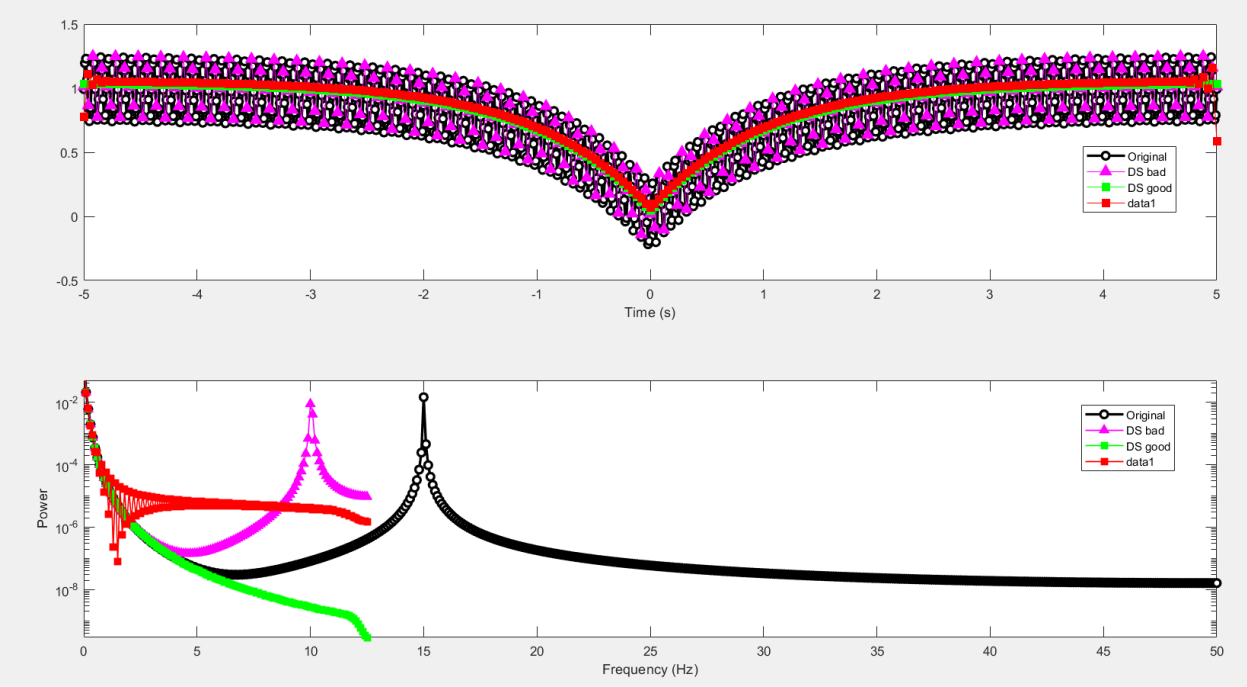


71. DOWNSAMPLING

Anti-aliasing filter – Low-pass filter at new Nyquist frequency

MATLAB

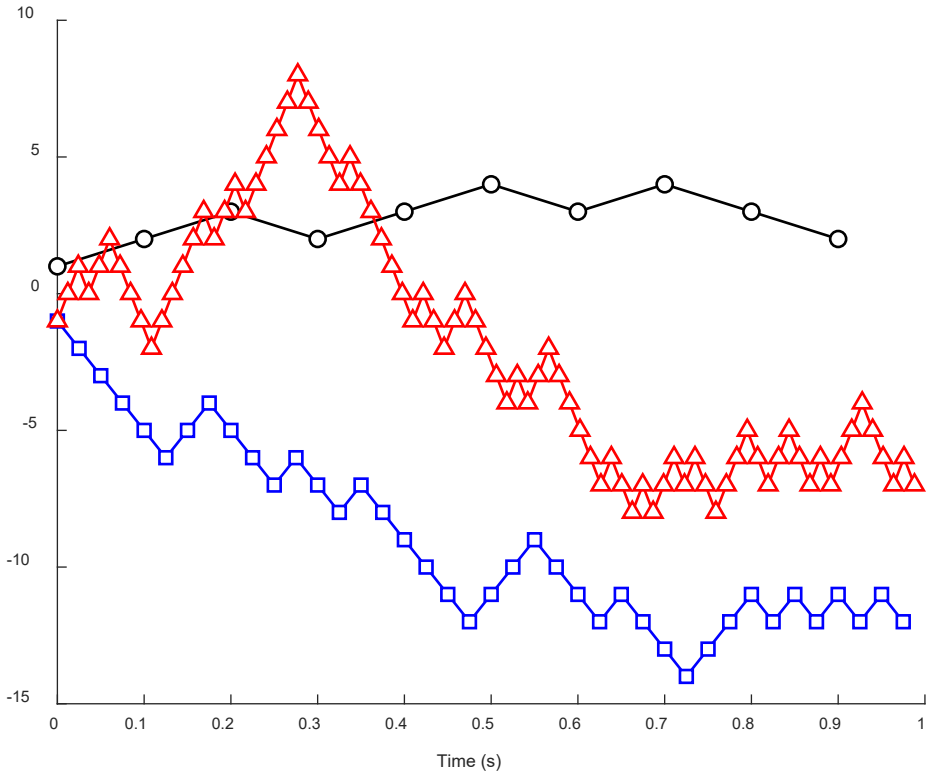
Code: sigprocMXC_downsample.m

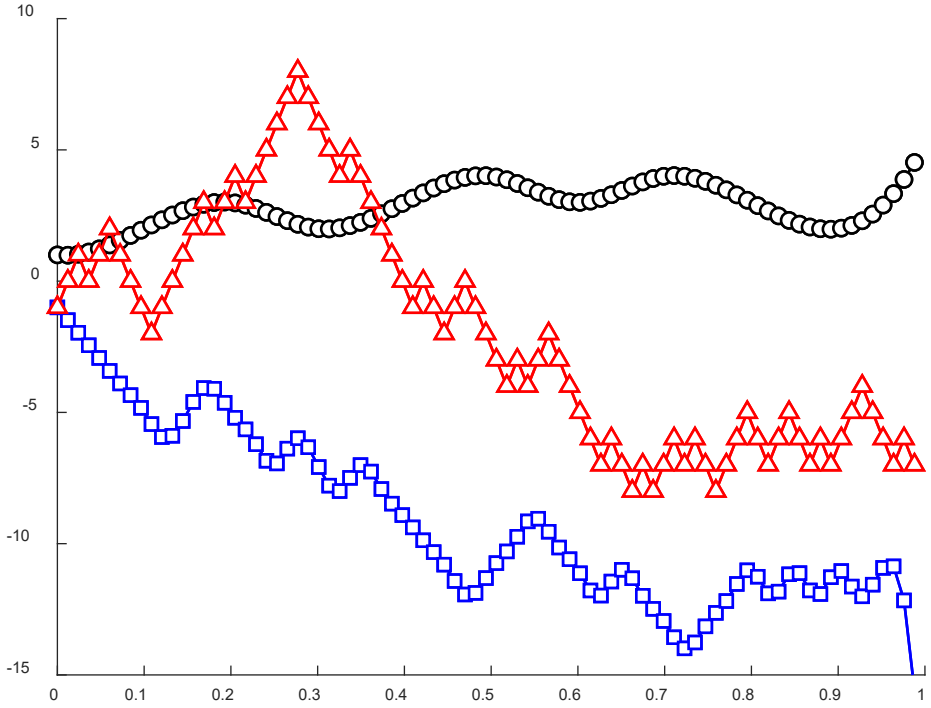


72. STRATEGIES FOR MULTIRATE SIGNALS

MATLAB

Code: sigprocMXC_multirate.m


10
5
0
-5
-10
-15
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time (s)

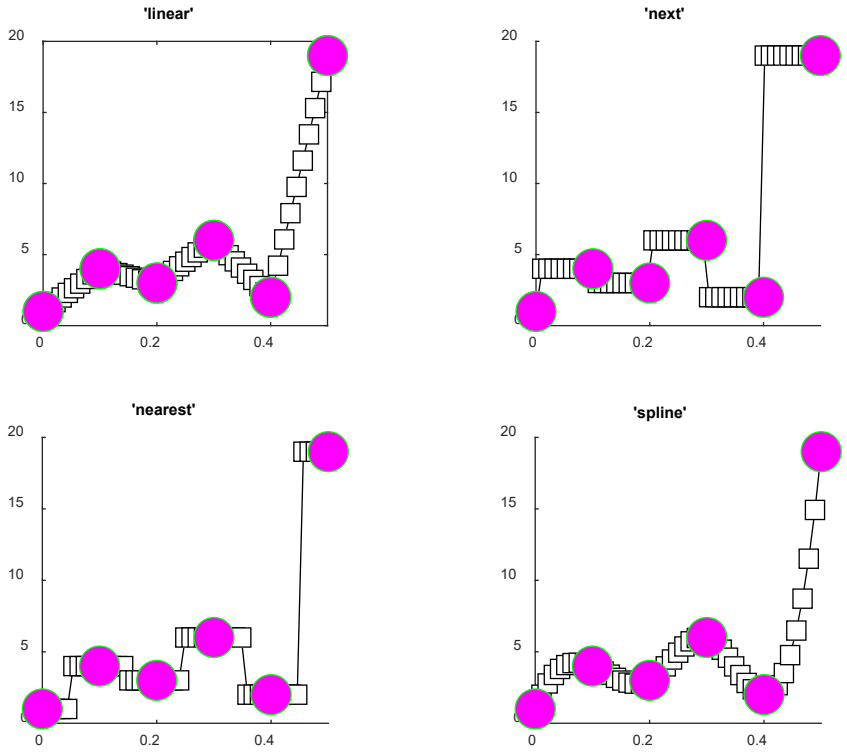

10
5
0
-5
-10
-15
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

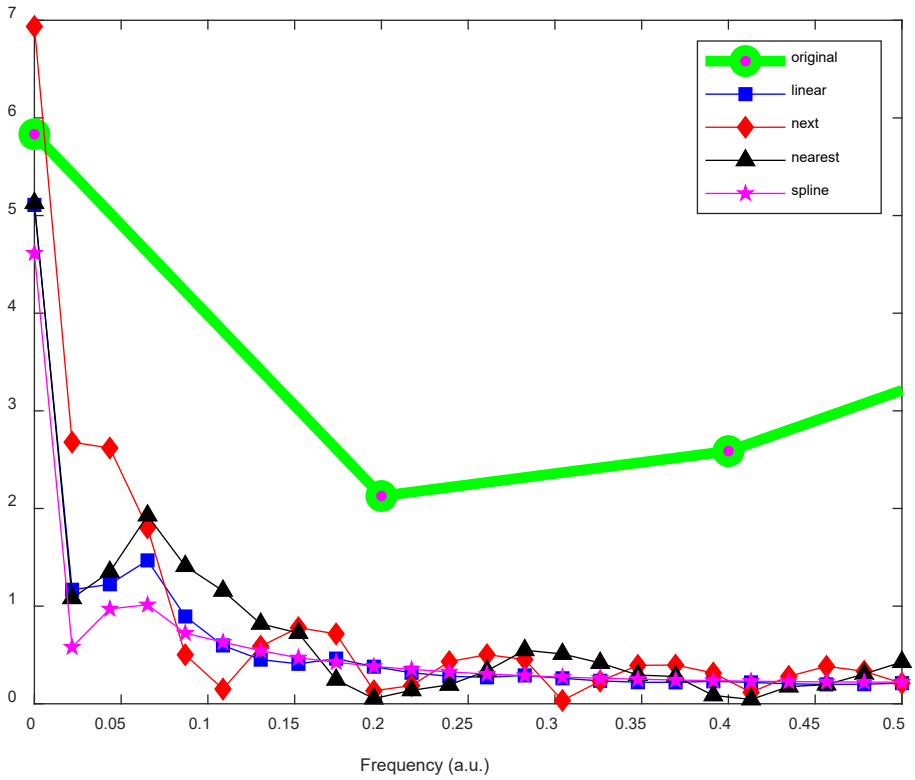
73. INTERPOLATION

Interpolation is just upsampling

MATLAB

Code: sigprocMXC_interp.m



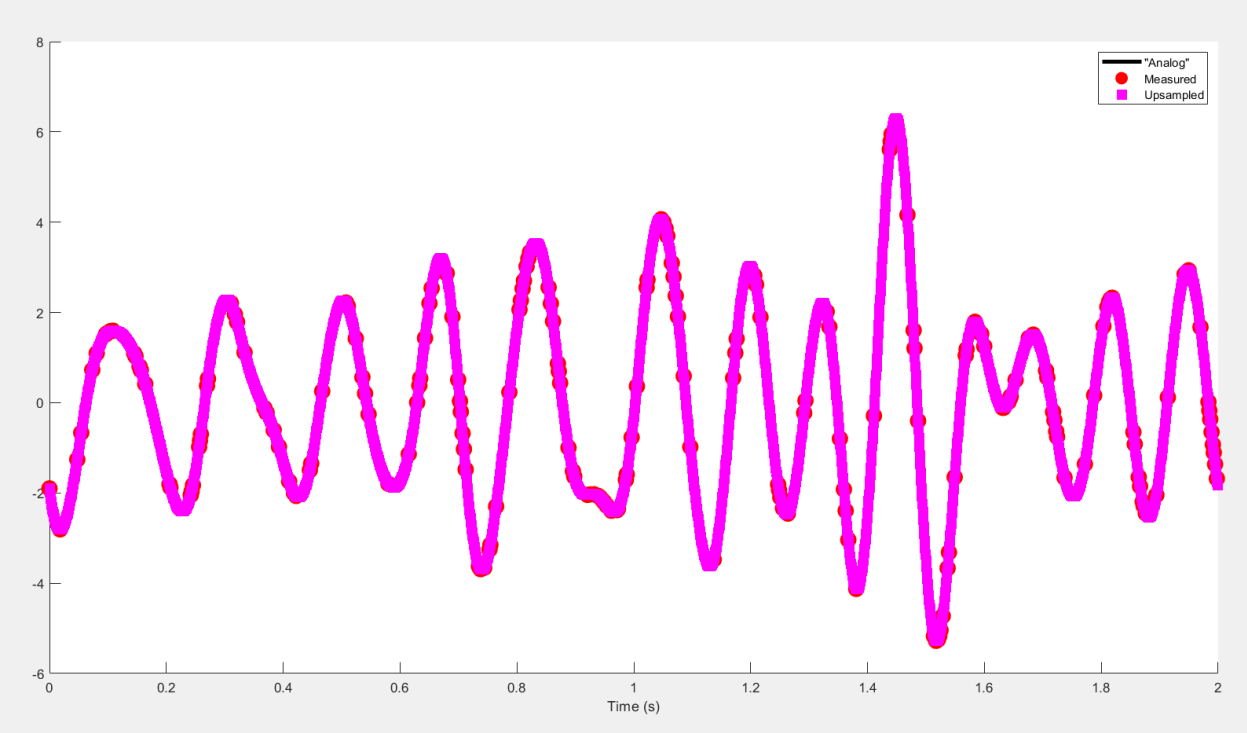


Note: spline interpolation has no edges in time domain, so the frequency domain does not have bouncing effects in plot.

74. RESAMPLE IRREGULARLY SAMPLED DATA

MATLAB

Code: sigprocMXC_irregular.m



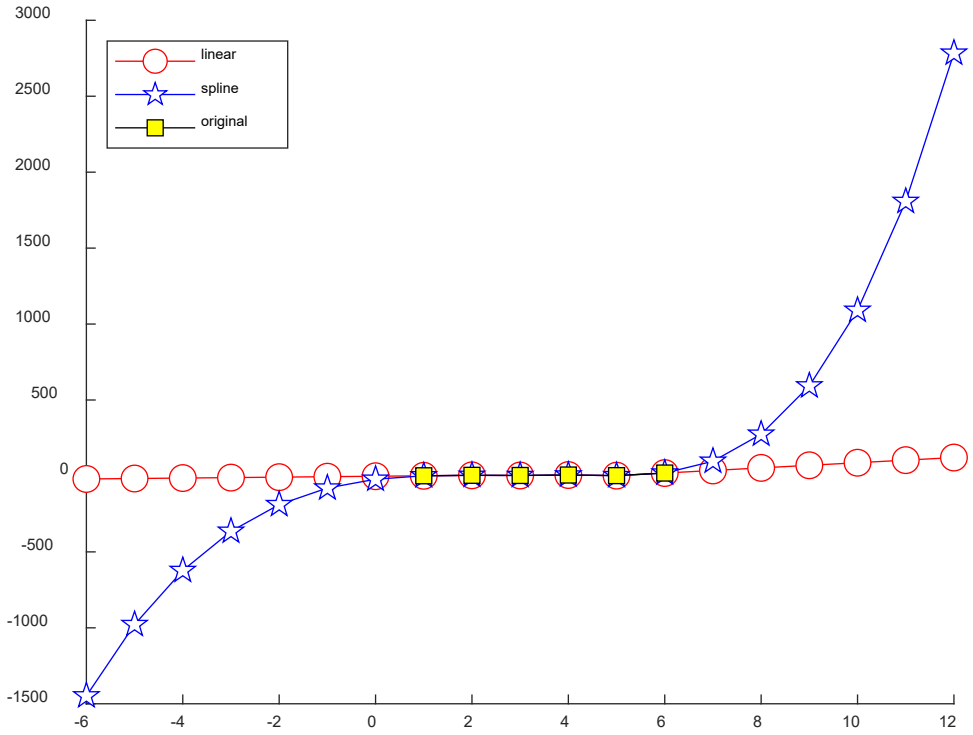
Signal changing relatively slowly compared to the average sampling rate of the time signal, 7 Hz in this code. If peak frequency goes up to 100 Hz, then the interpolation does not match the original analog signal. When you want to bring irregularly sampled data up to a higher sampling rate, must be concerned about the spectral concentration of the signal relative to the lowest Nyquist frequency in the original signal.

75. EXTRAPOLATION

Guessing what happens outside to the boundaries of measured signal – no anchor points.

MATLAB

Code: sigprocMXC_extrap.m



76. SPECTRAL INTERPOLATION

Assume power spectrum smoothly changes between gap.

Take size of window (in time) represented by the gap

Take the same size window before and after the gap using known signal at the two boundaries

Take fft of both known time windows: before and after

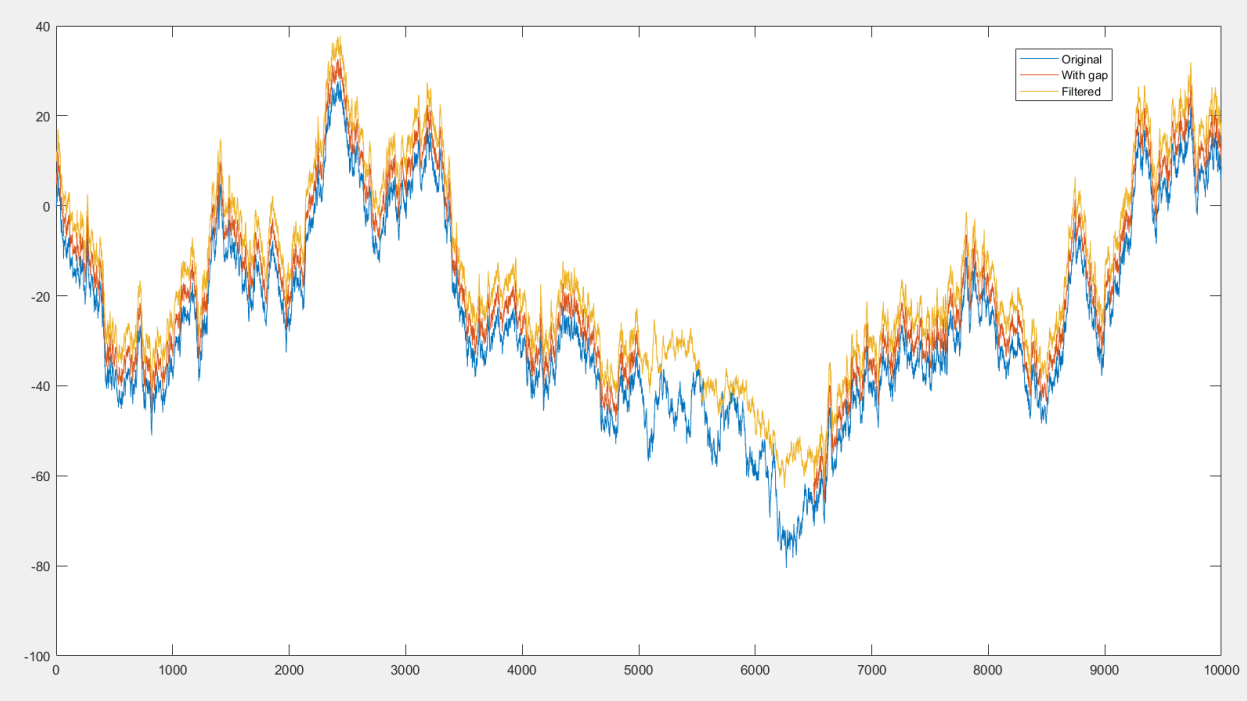
Average these two spectra together to produce a third spectra

Take the ifft of this average fft

Add a linear trend to fit between both boundaries to connect the ifft result into the gap

MATLAB

Code: sigprocMXC_spectralInterp.m



77. DYNAMIC TIME WARPING

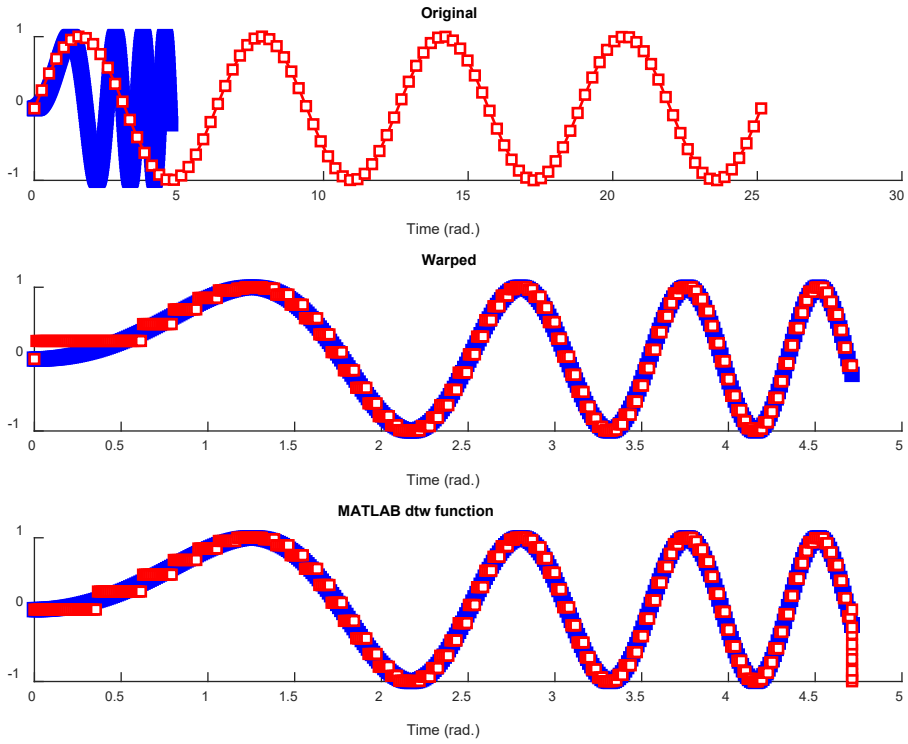
Other methods:

Constrained DTW

Multiresolution DTW – reduce the total number of blocks that need to be searched

MATLAB

Code: sigprocMXC_dtw.m



78. CODE CHALLENGE: DENOISE AND DOWNSAMPLE THIS SIGNAL!

Fix:

- Find all NaN's in the data (0/0) and remove
- Excessive noise bursts – interpolate across and remove
- Resample time series to be regularly spaced

MATLAB

Data: resample_codeChallenge.mat

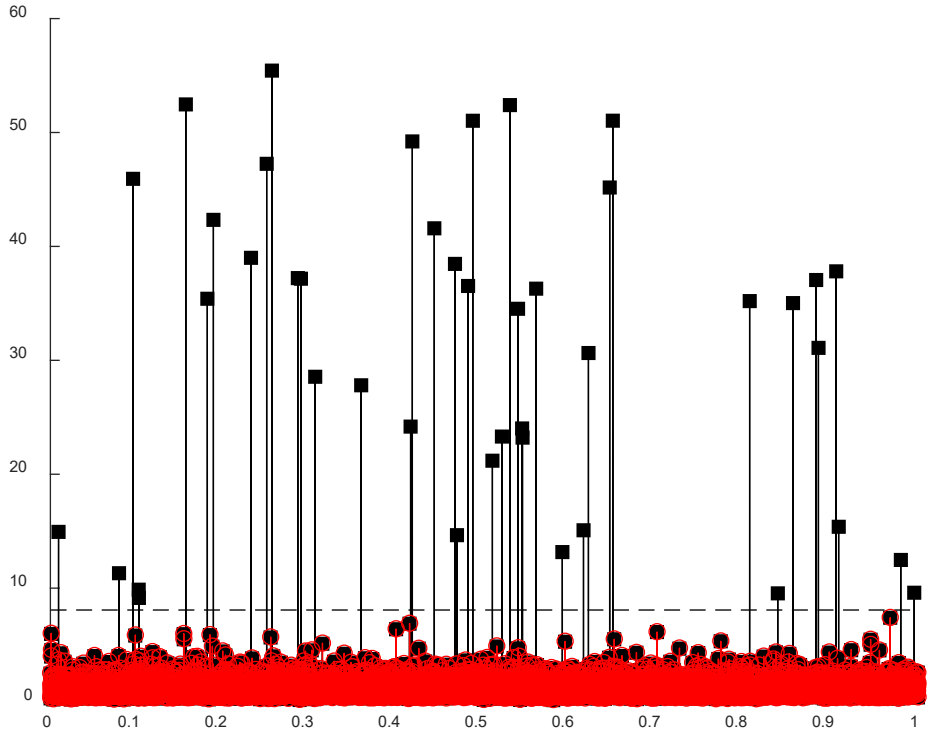
SECTION 9: OUTLIER DETECTION

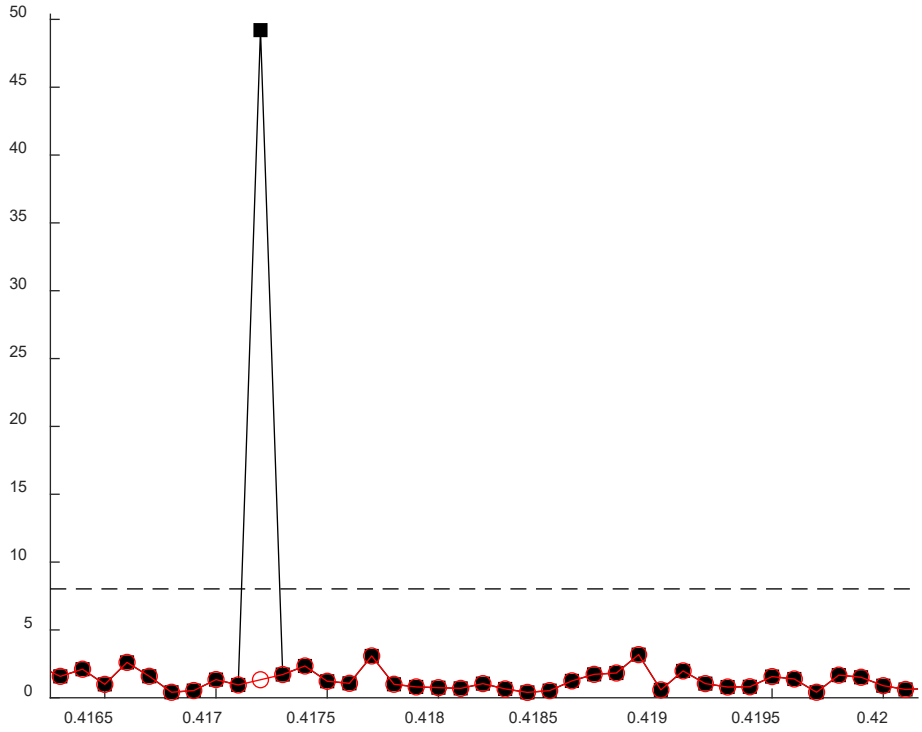
Smoothing (using the Curve Fitting Toolbox) a dataset does not solve the problem with outliers

80. OUTLIERS VIA STANDARD DEVIATION THRESHOLD

MATLAB

Code: sigprocMXC_outZ.m



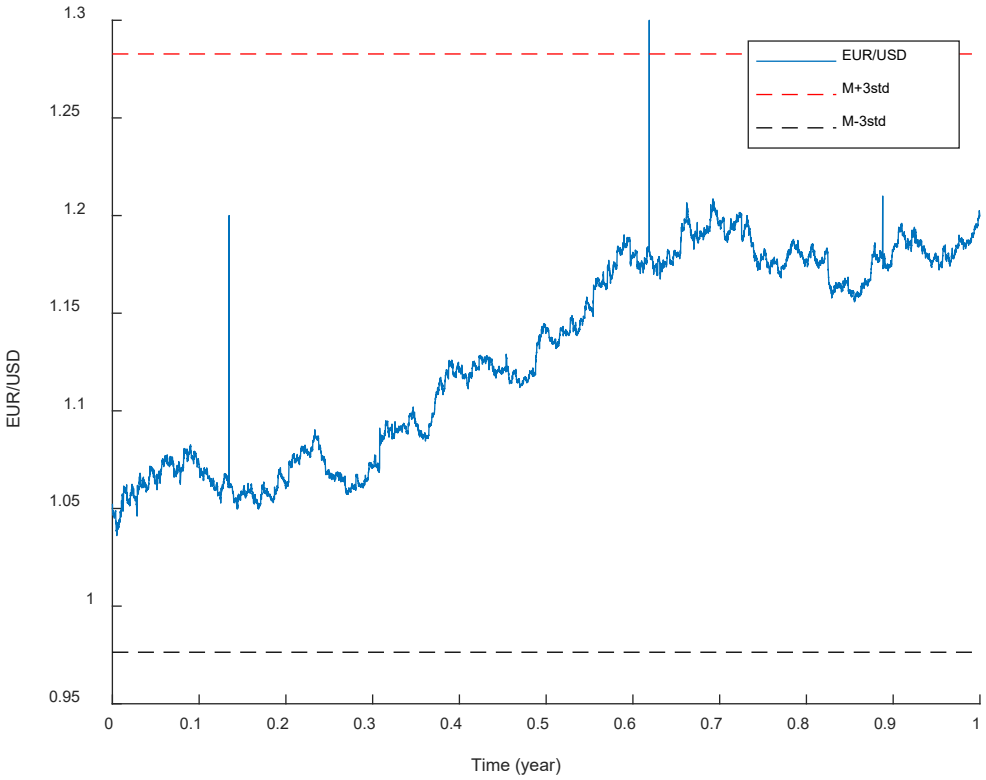


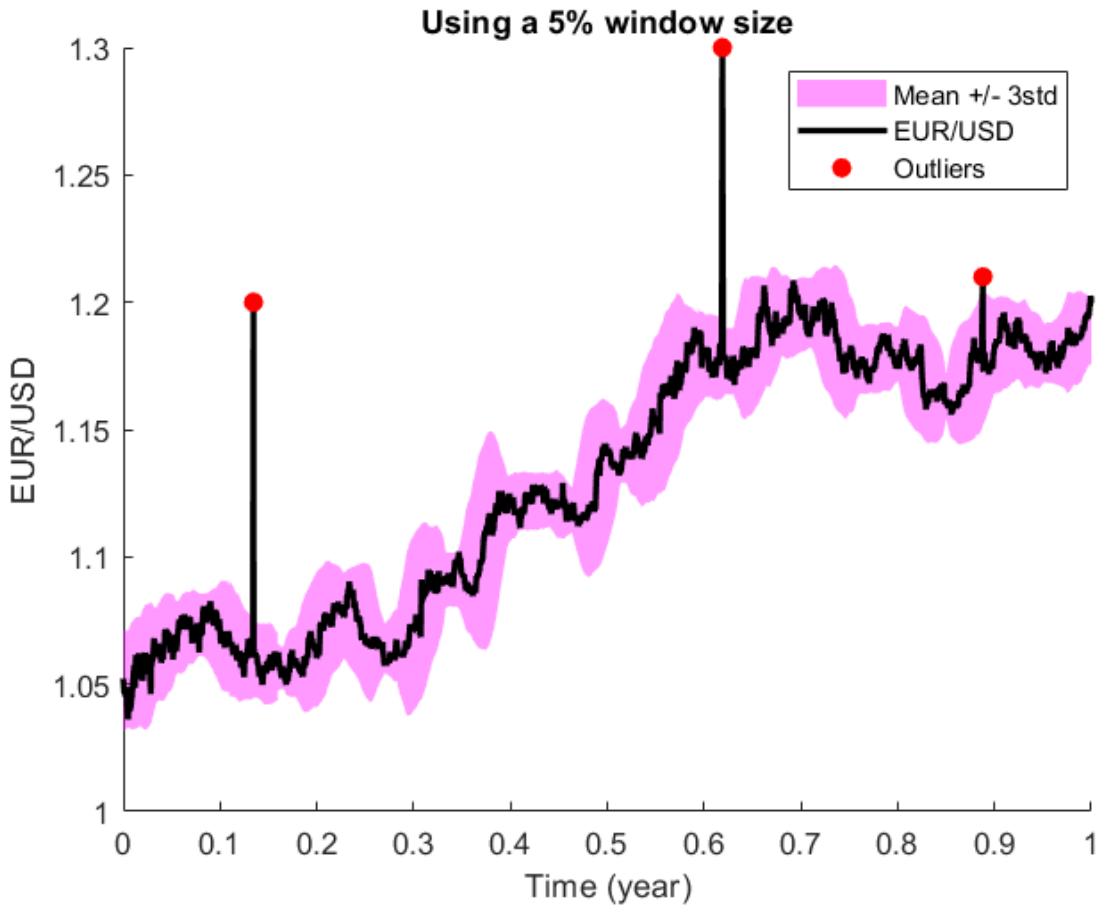
In the zoomed-in plot above we see an example of a point that was treated as an outlier. The data point is converted to an interpolated value using the +1 and -1 data points in time.

81. OUTLIERS VIA LOCAL THRESHOLD EXCEEDANCE

MATLAB

Code: sigprocMXC_localOutliers.m

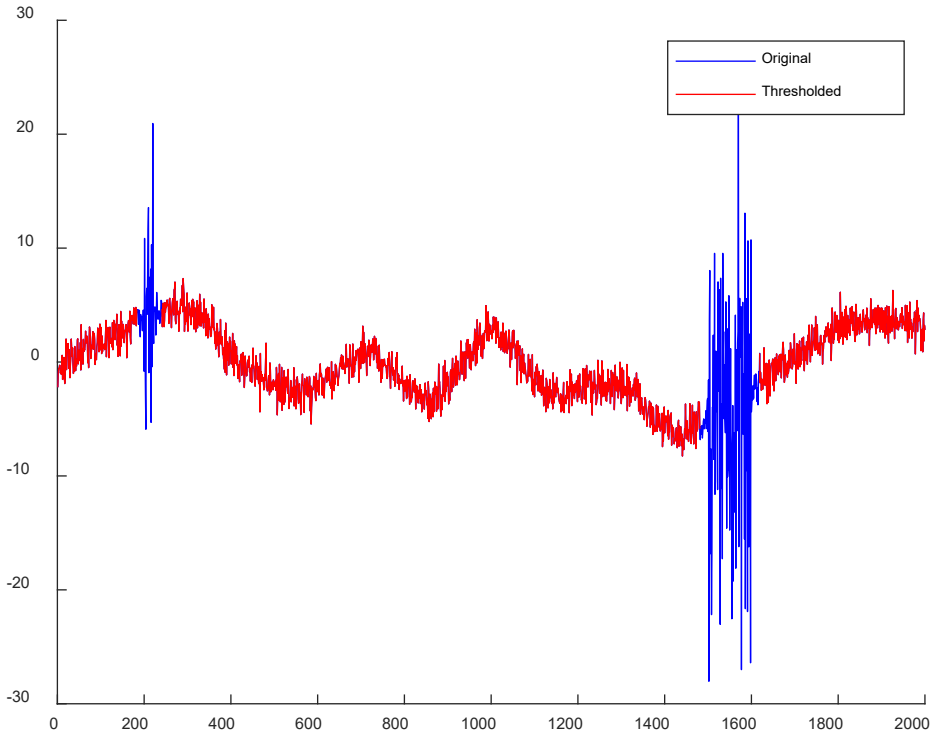

EUR/USD
M+3std
M-3std
EUR/USD
Time (year)
1.3
1.25
1.2
1.15
1.1
1.05
1
0.95
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

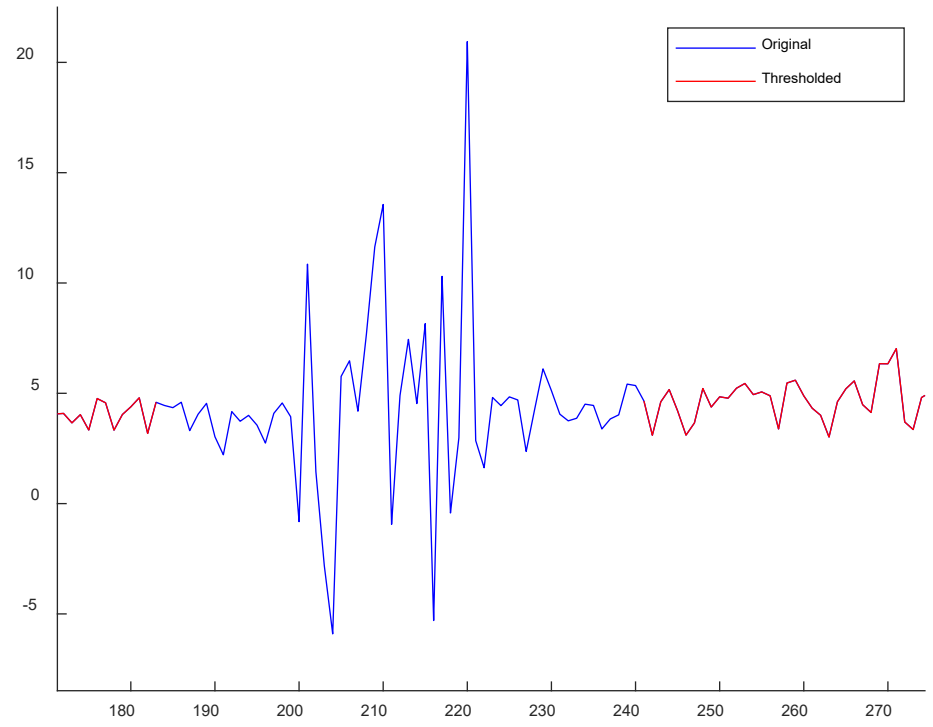


82. OUTLIER TIME WINDOWS VIA SLIDING RMS

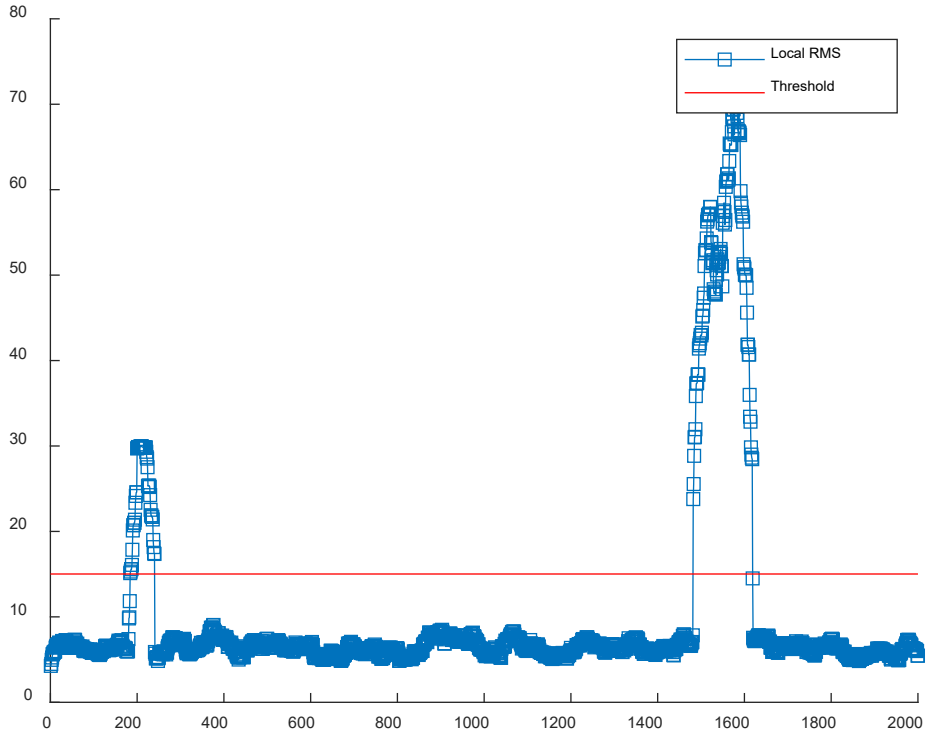
MATLAB

Code: sigprocMXC_RMSoutlierWindows.m


Original
Thresholded
30
20
10
0
-10
-20
-30
0
200
400
600
800
1000
1200
1400
1600
1800
2000


Original
Thresholded
20
15
10
5
0
-5
180
190
200
210
220
230
240
250
260
270

Close up of first outlier window showing some useful data on right and left being tagged as outlier. Lower threshold to include more of the shoulder data.



83. CODE CHALLENGE

Use a loop to find optimum pct_win variable (window size as percent of total signal length as a percent) instead of threshold as used in code and lecture 82 above.

Code challenge is to reproduce the results shown above.

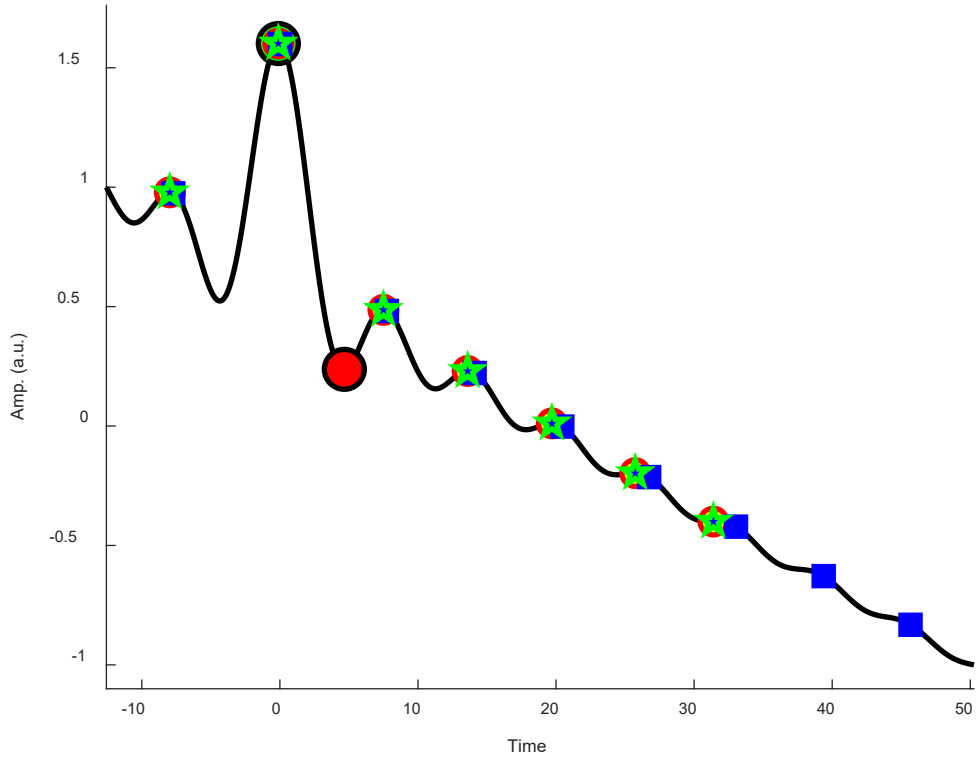
SECTION 10: FEATURE DETECTION

85. LOCAL MAXIMA AND MINIMA

One (usually) global maxima vs multiple local maxima (and minima). The global maxima is also a local maxima in a given region. It is acceptable for there to be more than one global maxima when their peaks are nearly the same.

MATLAB

Code: sigprocMXC_localMinMax.m

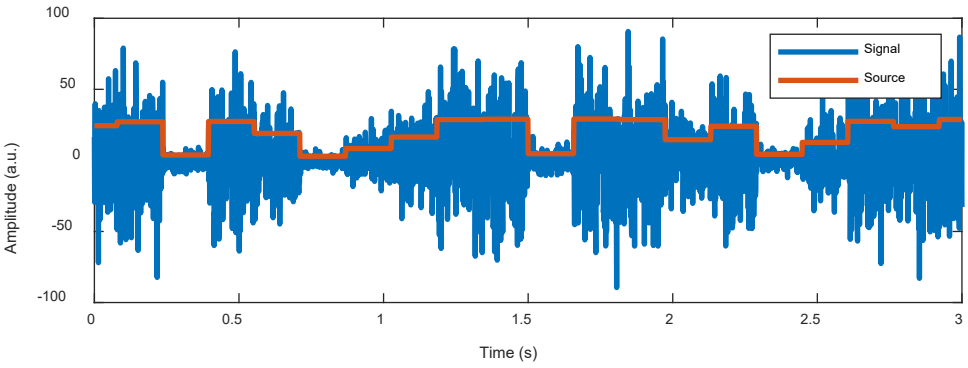


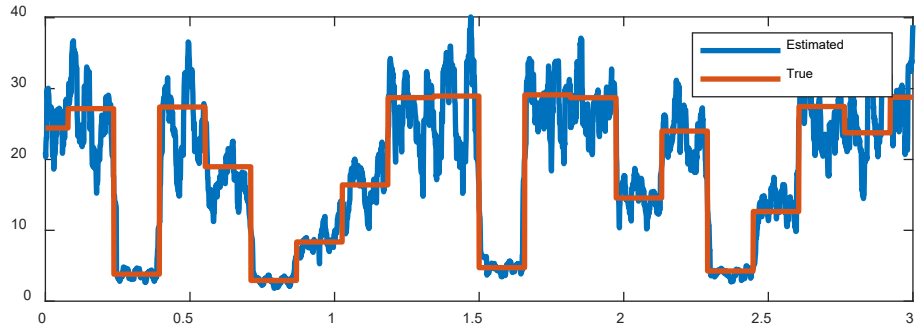
86 RECOVER SIGNAL FROM NOISE AMPLITUDE

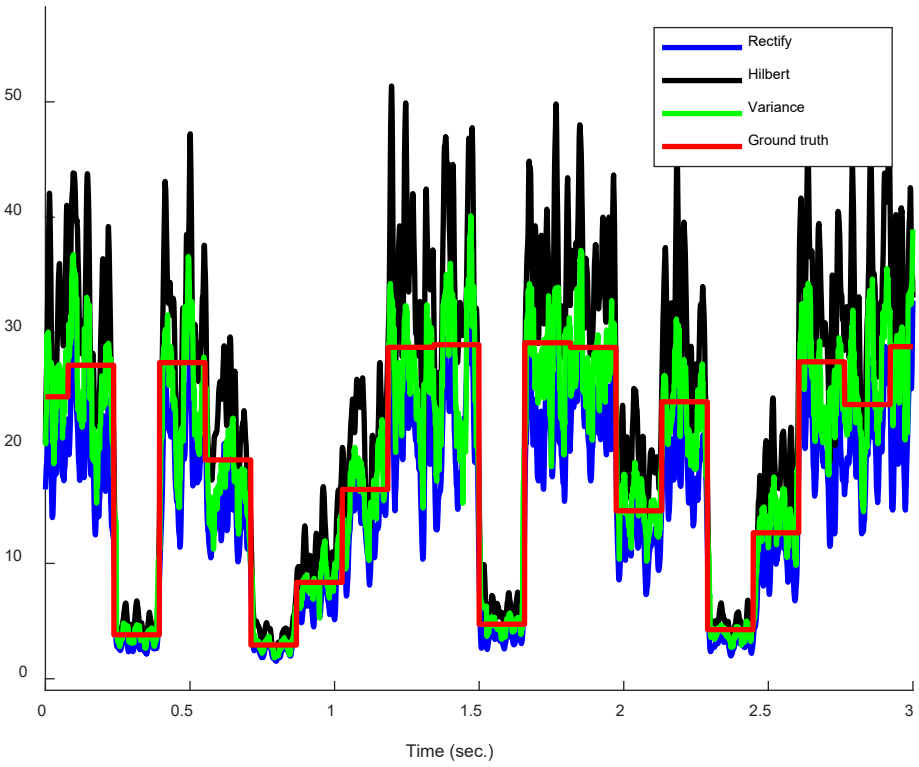
Show a few options to extract modulation amplitude from a signal

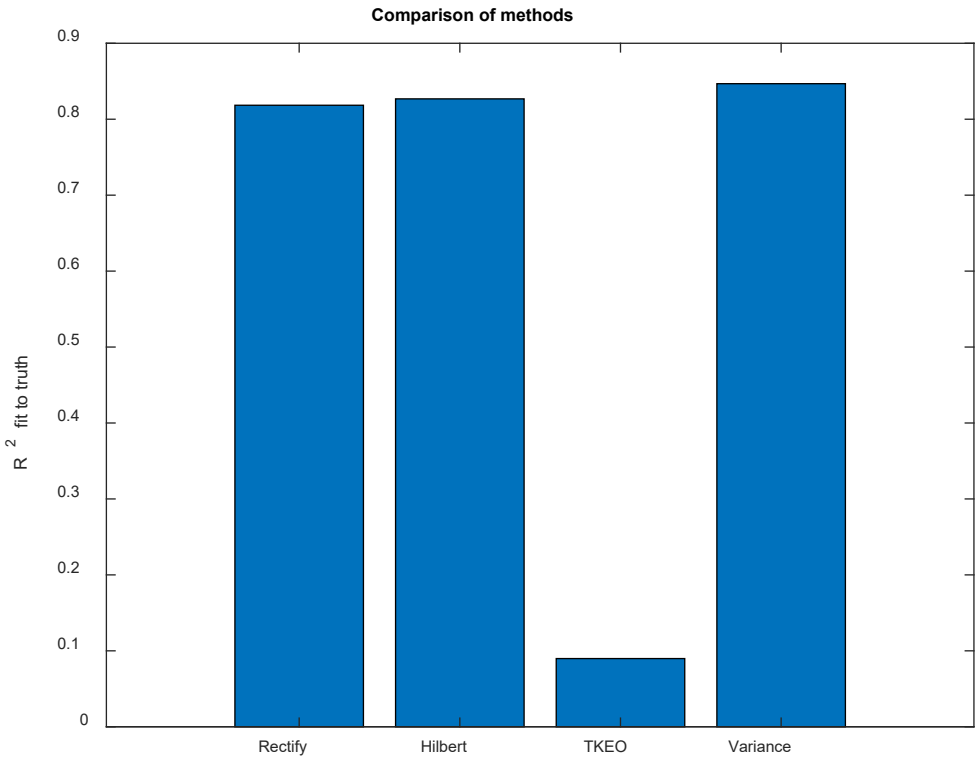
MATLAB

Code: sigprocMXC_signalFromNoise.m


Signal
Source
Amplitude (a.u.)
Time (s)


Estimated
True


Rectify
Hilbert
Variance
Ground truth
Time (sec.)



87. WAVELET CONVOLUTION FOR FEATURE EXTRACTION

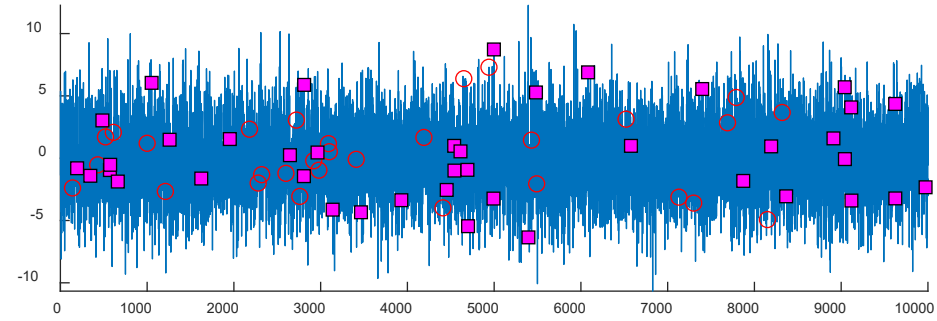
See lecture: 16. Averaging Multiple Repetitions (Time-Synchronous Averaging)

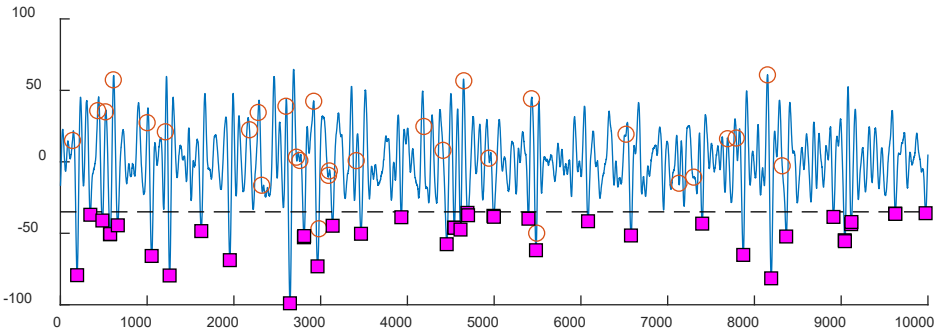
Wavelets have two properties:

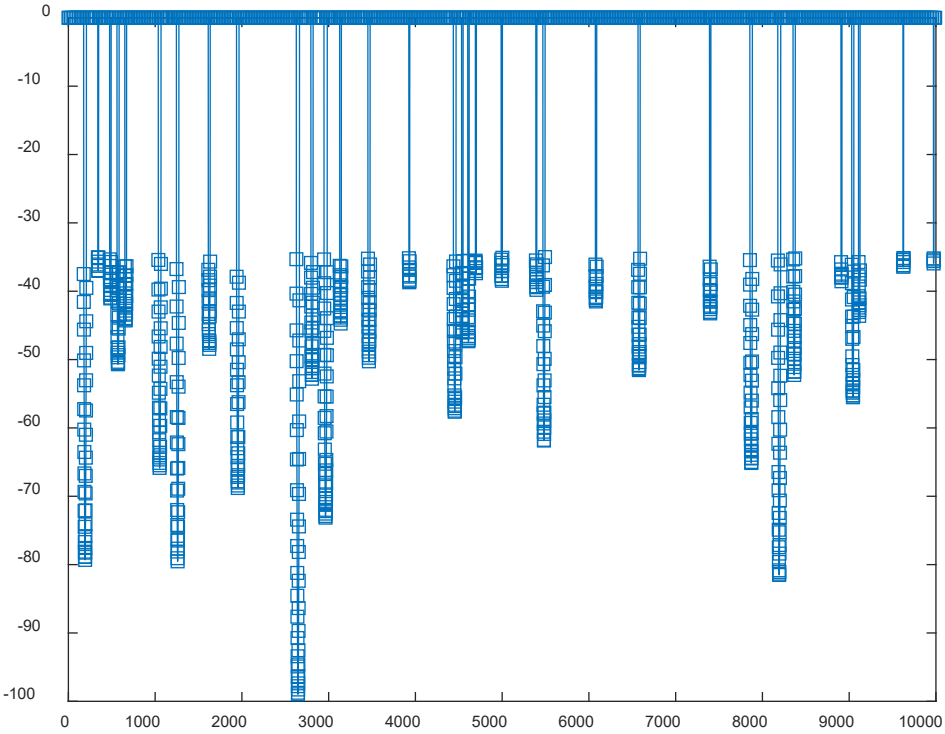
- Both ends taper to zero
- Sums / Integrates to zero

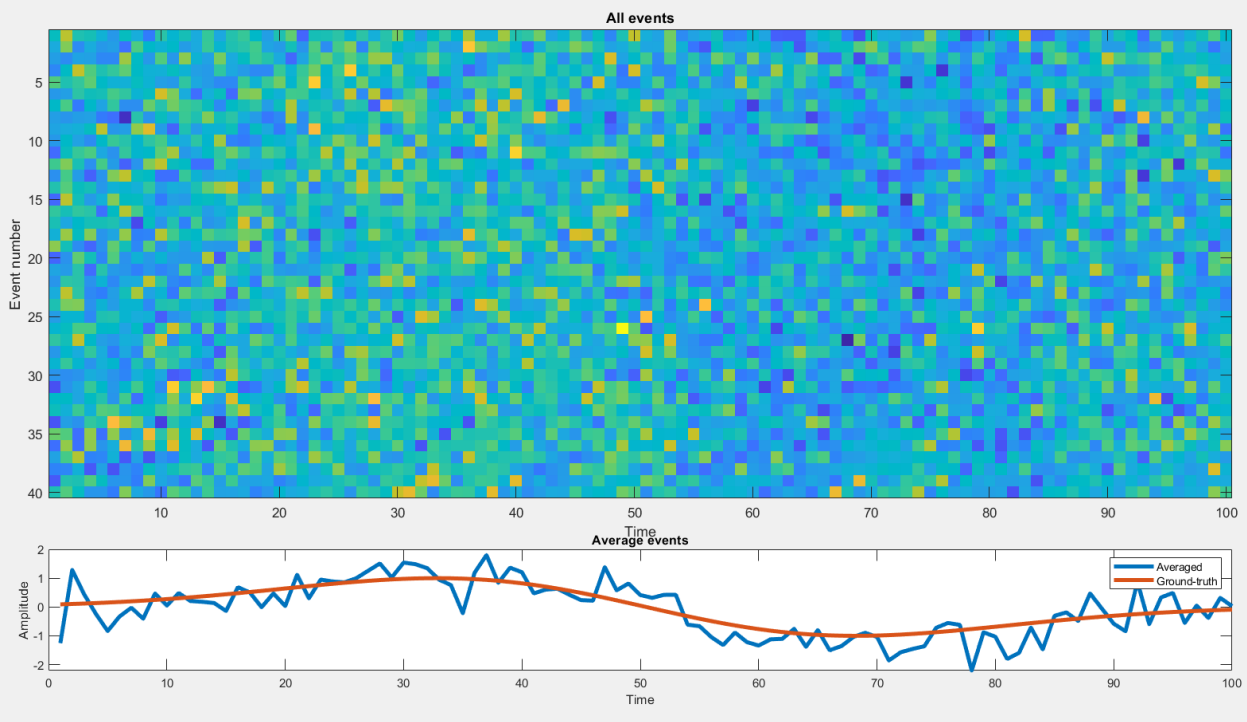
MATLAB

Code: sigprocMXC_waveletFeatureEx.m


10
5
0
-5
-10
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


100
50
0
-50
-100
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

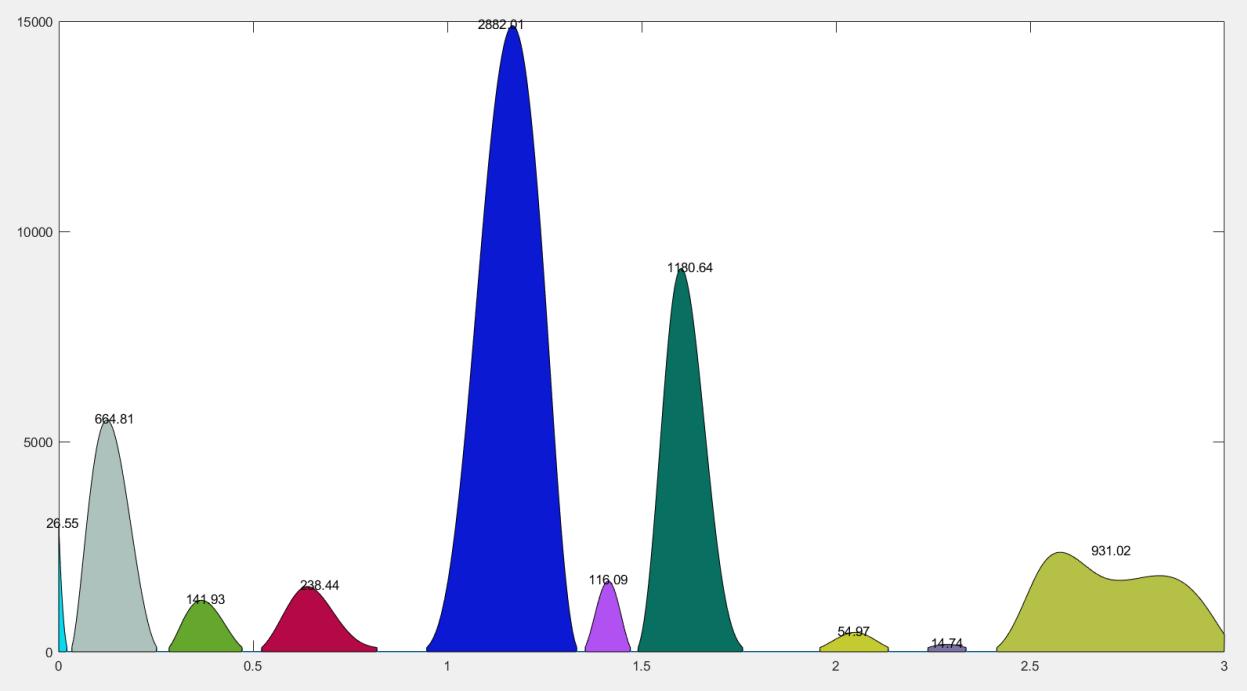


88. AREA UNDER THE CURVE

In mathematics, a Riemann sum is a certain kind of approximation of an integral by a finite sum.

MATLAB

Code: sigprocMXC_AUC.m

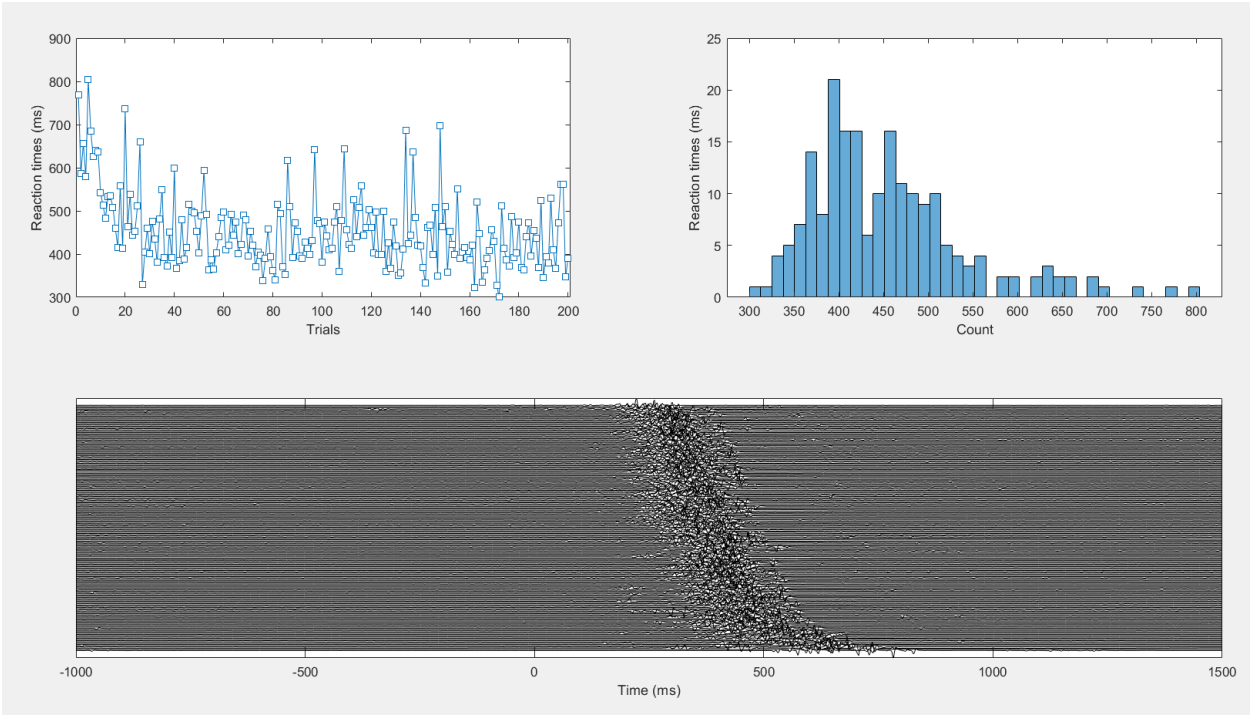


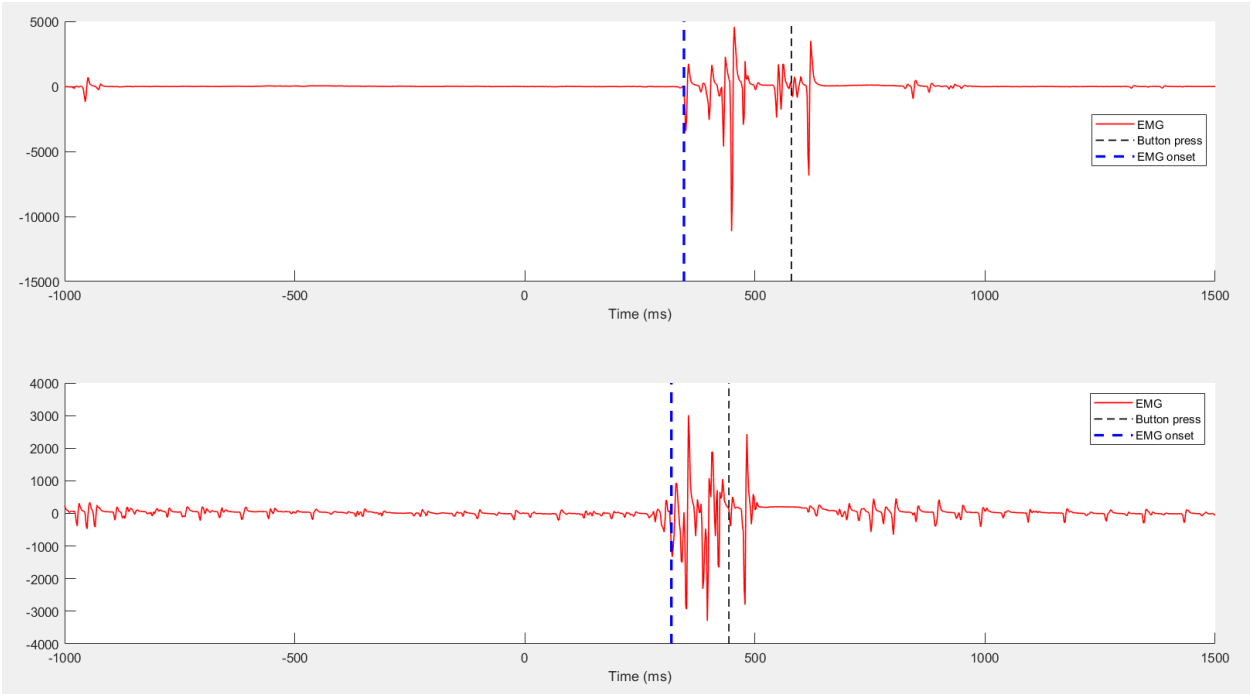
89. APPLICATION: DETECT MUSCLE MOVEMENTS FROM EMG RECORDINGS

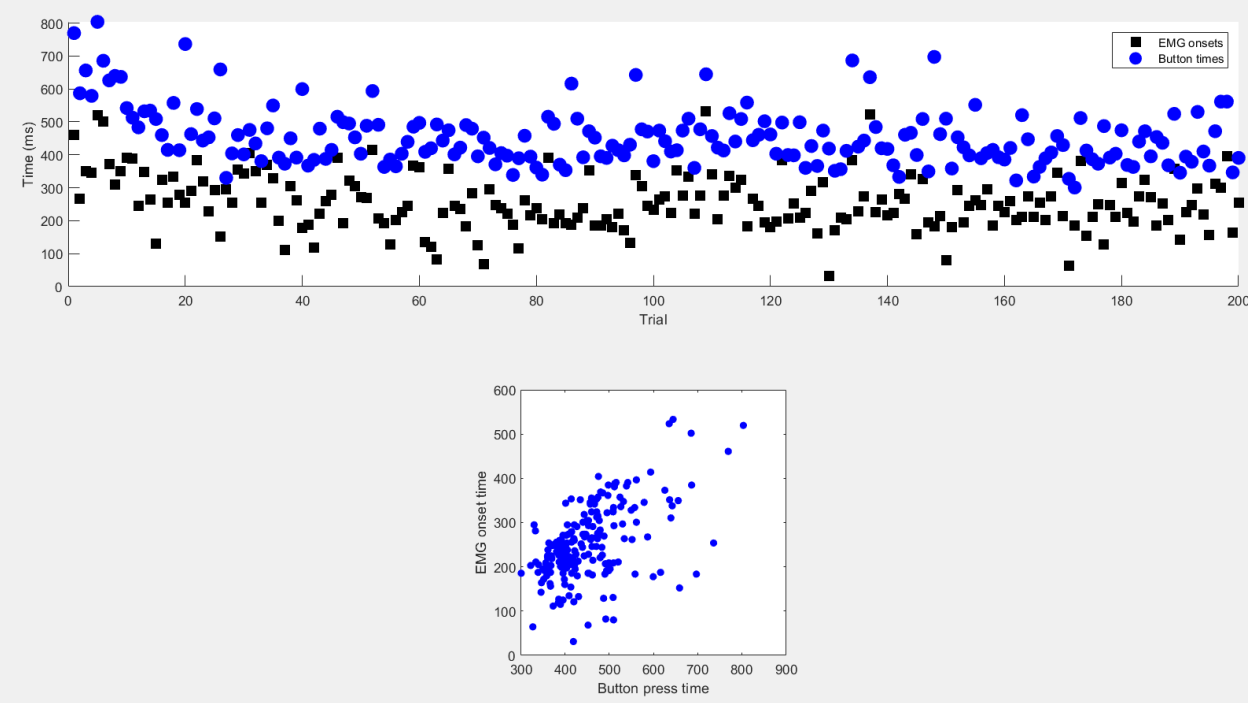
MATLAB

Code: sigprocMXC_EMGonsets.m

Data: EMGRT.mat


Reaction times (ms)
Trials
Reaction times (ms)
Count
Time (ms)


EMG
Button press
EMG onset
Time (ms)
EMG
Button press
EMG onset
Time (ms)



90. FULL WIDTH AT HALF-MAXIMUM

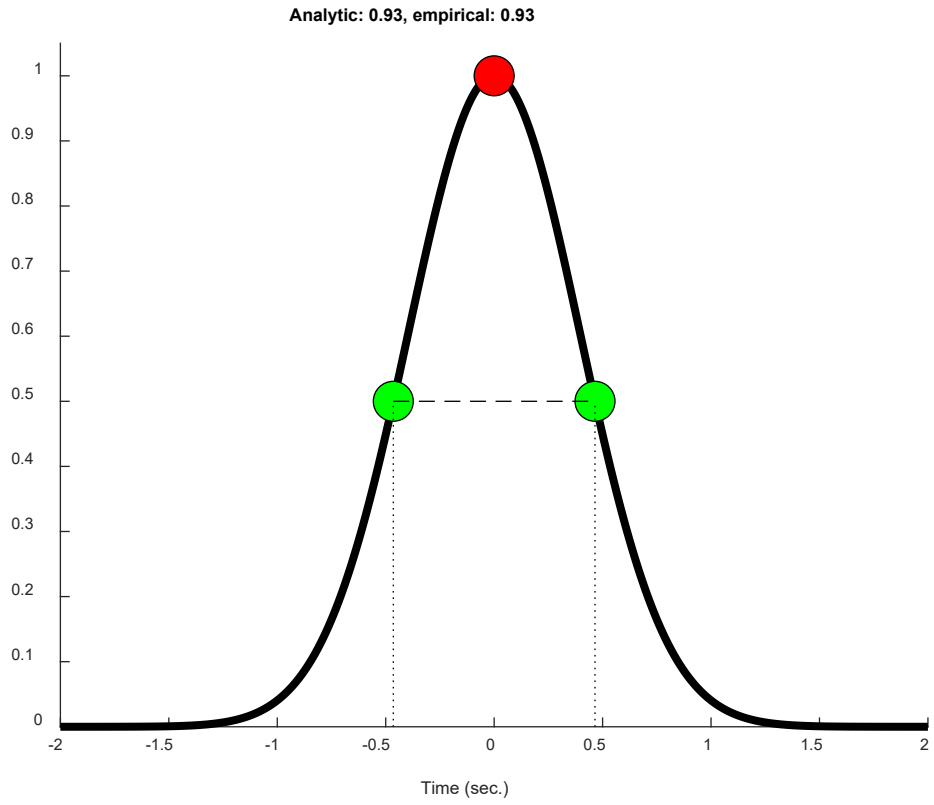
FWHM is a measure of time

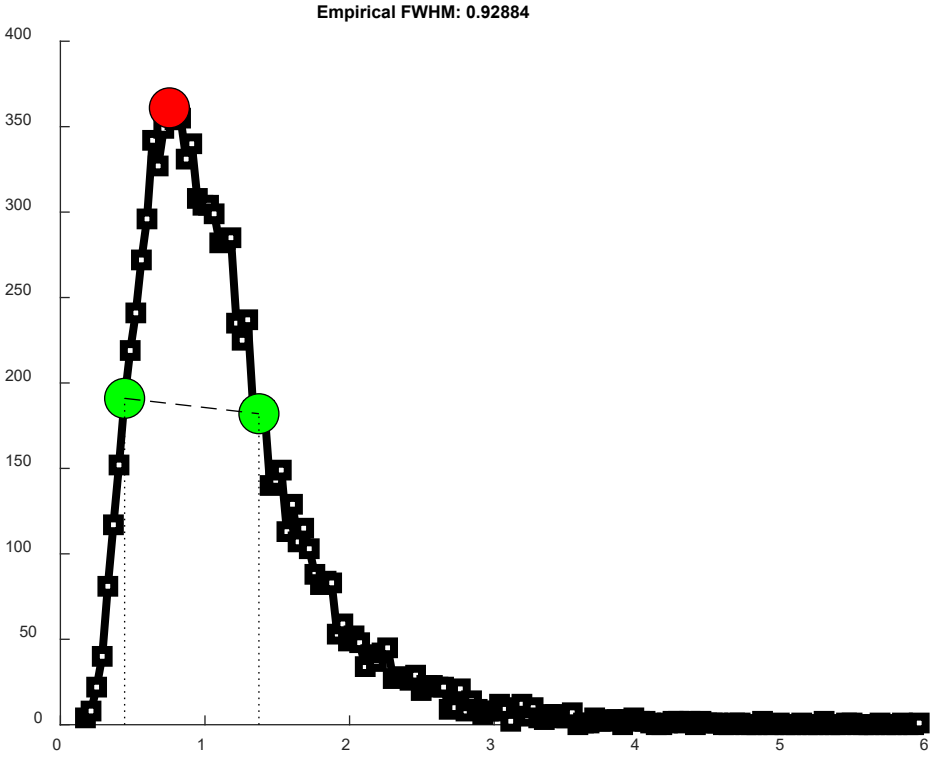
An estimate of how much smoothing to be introduced into the time series, aka temporal uncertainty.

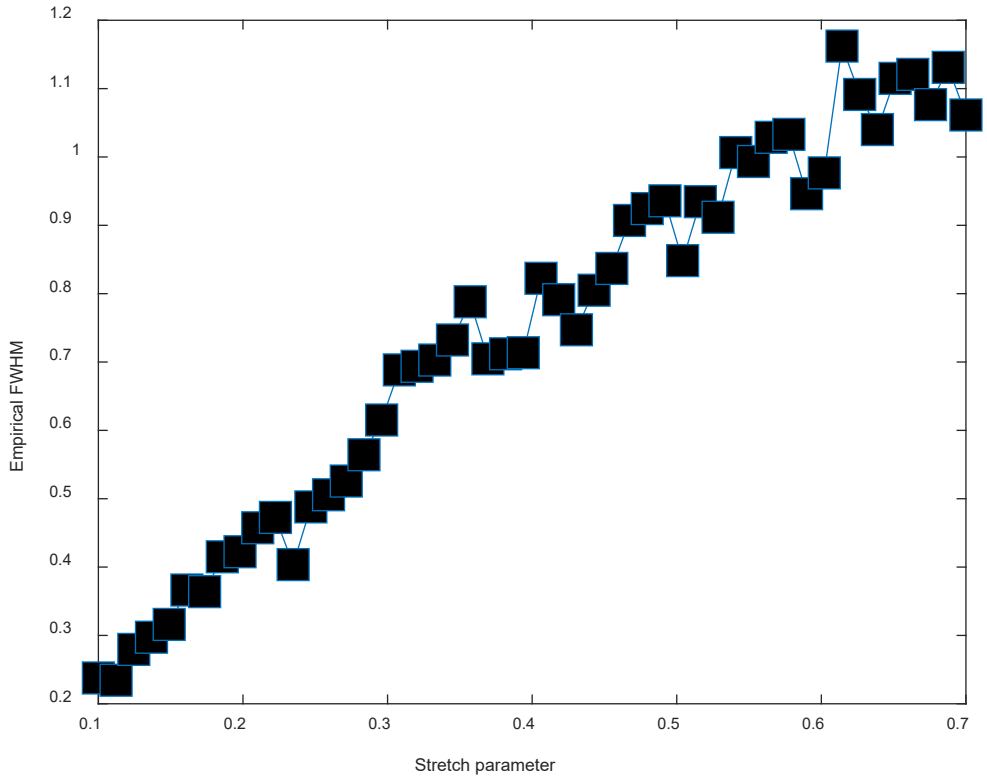
Analytic value is based on theoretical perfect gaussian curve, while empirical value is based actual gaussian which is affected by sample rate.

MATLAB

Code: sigprocMXC_FWHM.m


Analytic: 0.93, empirical: 0.93
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Time (sec.)


Empirical FWHM: 0.92884
400
350
300
250
200
150
100
50
0
0
1
2
3
4
5
6



91. CODE CHALLENGE: FIND THE FEATURES!

Inaccurate because of sparse sampling, peak may not be the best estimate without smoothing

Smooth time series so there is less noise, but not too much to reduce sensitivity

Upsample, interpolate, so more data points to better estimate FWHM

Which to perform first: smooth or upsample.

Use code from last lecture

SECTION 11: VARIABILITY

93. TOTAL AND WINDOWED VARIANCE AND RMS

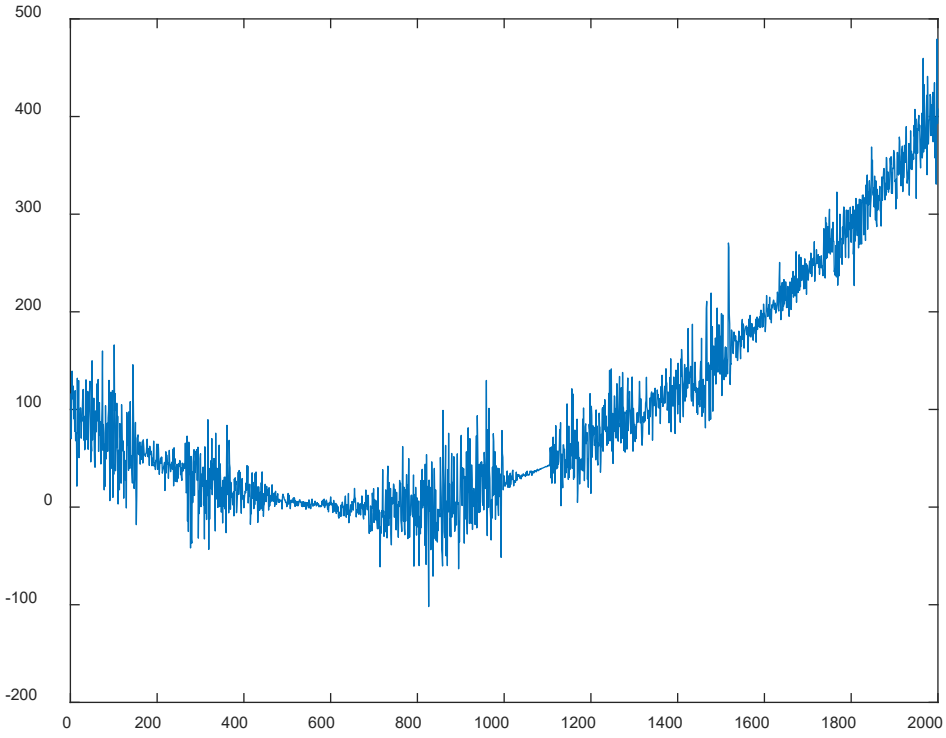
Three ways to measure dispersion, aka the energy in a signal:

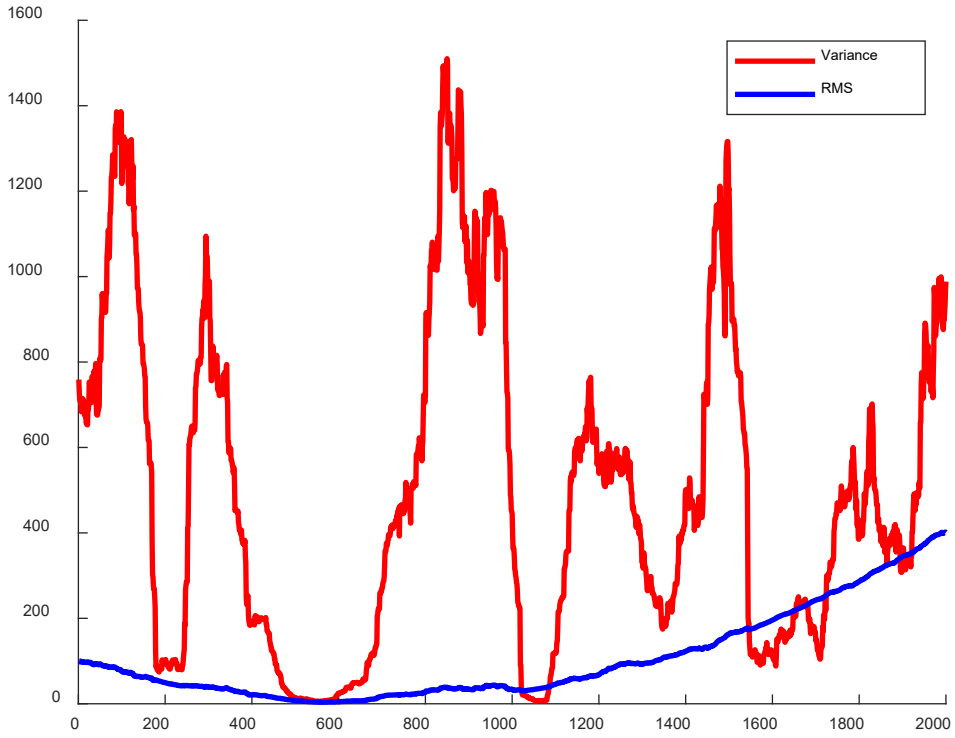
- rms = total energy in the signal $$rms = \sqrt{n^{-1} \sum_{i=1}^{n} y_i^2}$$

- var = dispersion around the mean energy in the signal $var = \sum_{i=1}^{n}(y_i - \bar{y})^2$
- std = dispersion $std = \sqrt{(n-1)^{-1}\sum_{i=1}^{n}(y_i - \bar{y})^2}$

MATLAB

Code: sigprocMXC_windowedVar.m



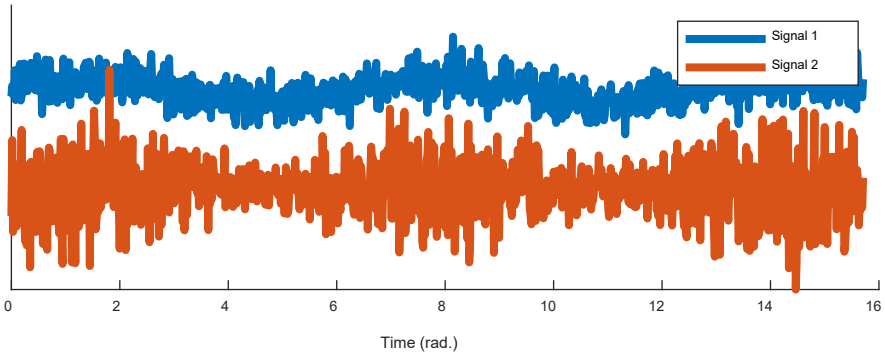


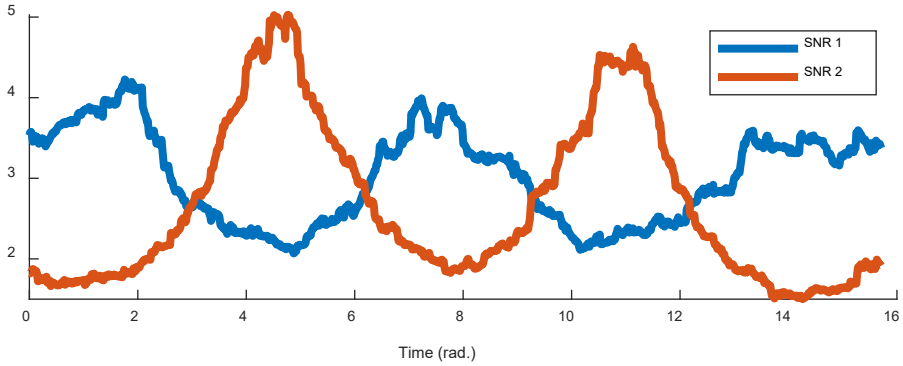
94. SIGNAL-TO-NOISE (SNR)

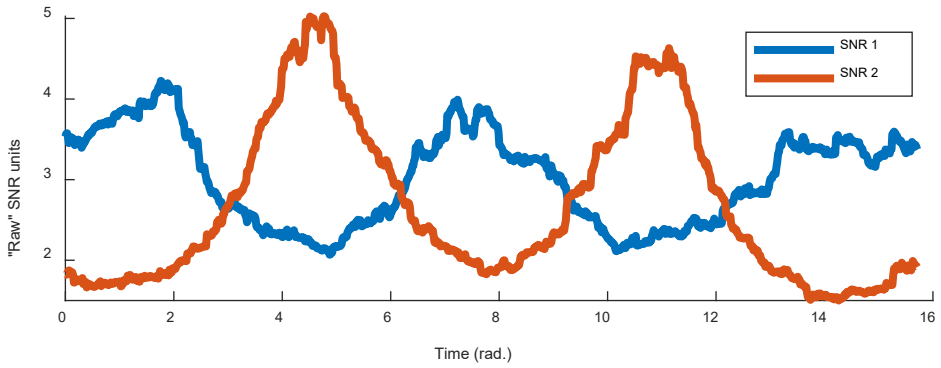
MATLAB

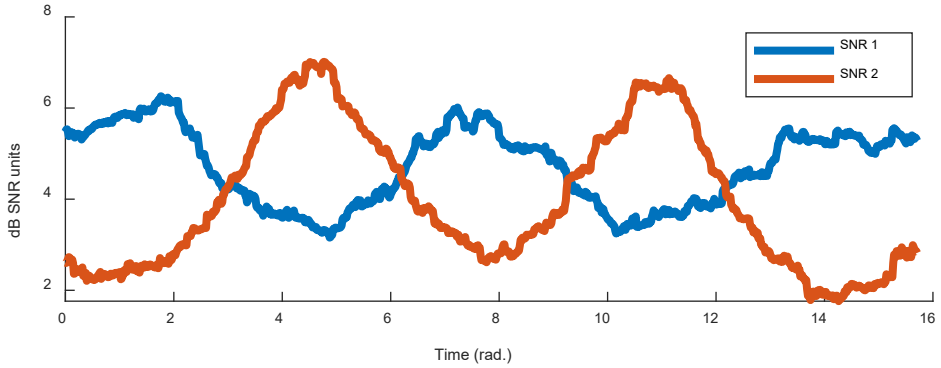
Code: sigprocMXC_SNR.m

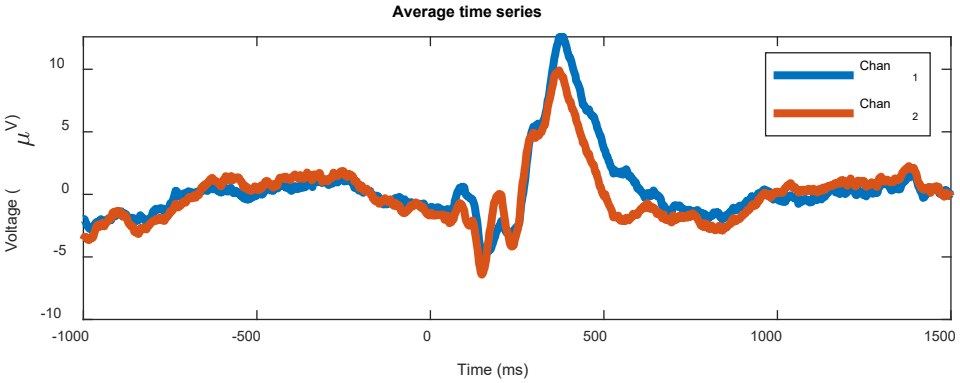
Data: SNRdata.mat

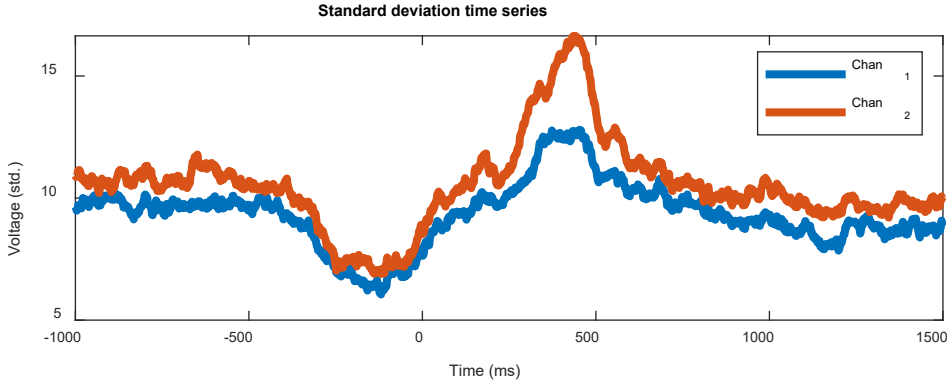

Signal 1
Signal 2
0
2
4
6
8
10
12
14
16
Time (rad.)

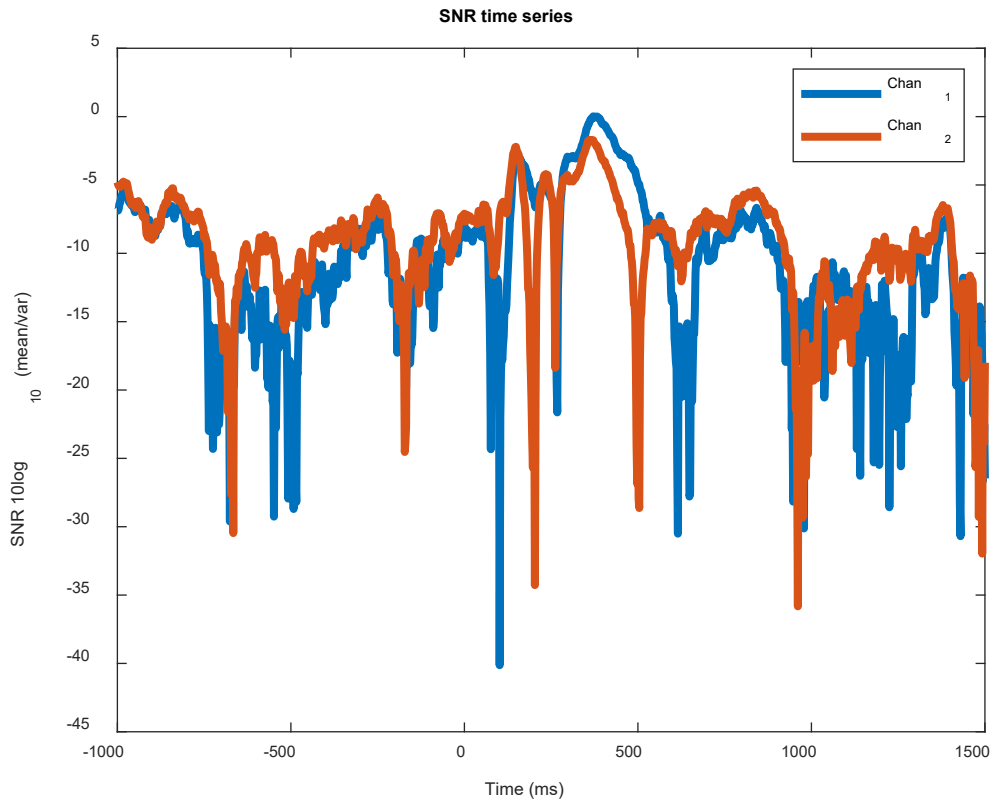

SNR 1
SNR 2
5
4
3
2
0
2
4
6
8
10
12
14
16
Time (rad.)


"Raw" SNR units
Time (rad.)
SNR 1
SNR 2


dB SNR units
Time (rad.)
SNR 1
SNR 2


Average time series
Voltage (μV)
Time (ms)
Chan 1
Chan 2


Standard deviation time series
Voltage (std.)
Time (ms)
Chan 1
Chan 2



Warning: Imaginary parts of complex X and/or Y argument ignored In sigprocMXC_SNR (line 110)

SNR at 375ms in channel 1 = 17.7849

SNR at 375ms in channel 2 = 9.1782

95. COEFFICIENT OF VARIATION (CV)

Looks line the inverse of a SNR

Valid for negative data

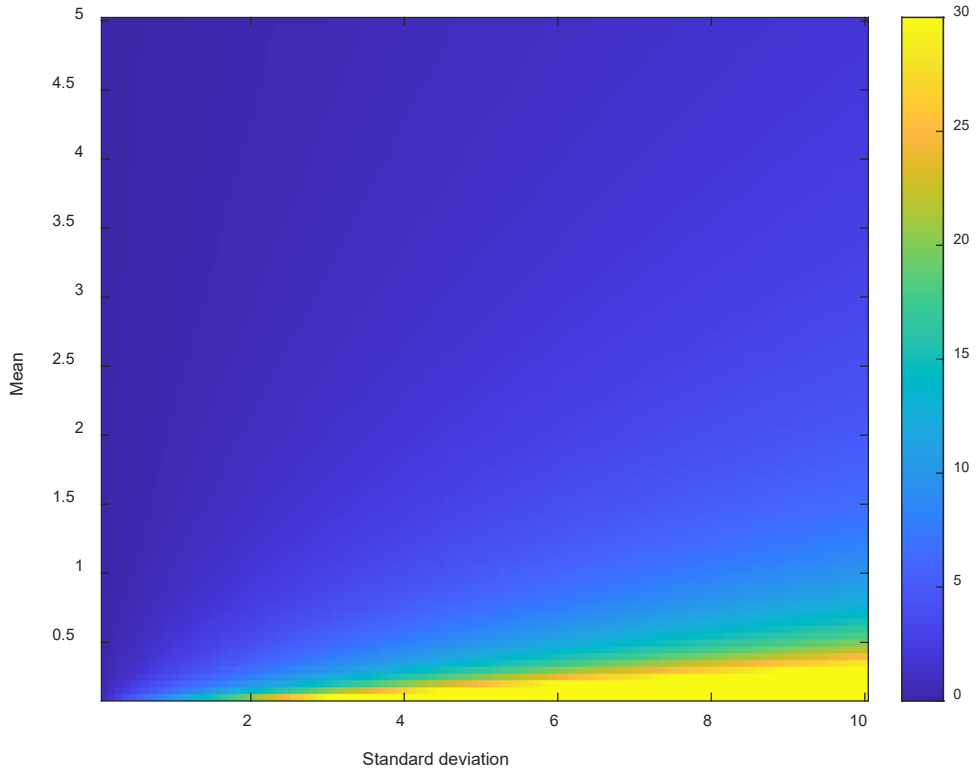
Noise is standard deviation (unitless)

$$cv = \frac{\sigma}{\mu}$$

The more robust a signal, the cv is closer to zero

MATLAB

Code: sigprocMXC_CV.m



96. ENTROPY

The use of entropy in communications and signal processing, aka Shannon Entropy

$$H = -\sum_{i=1}^{n} p_i log_2(p_i)$$

$$H \; is \; entropy$$

$$p_i \; is \; probablity \; data \; in \; bin \; i$$

% MATLAB code example

H = -sum(p.*log2(p+eps));

H units are in “bits”. We know this because log2 produces units of bits.

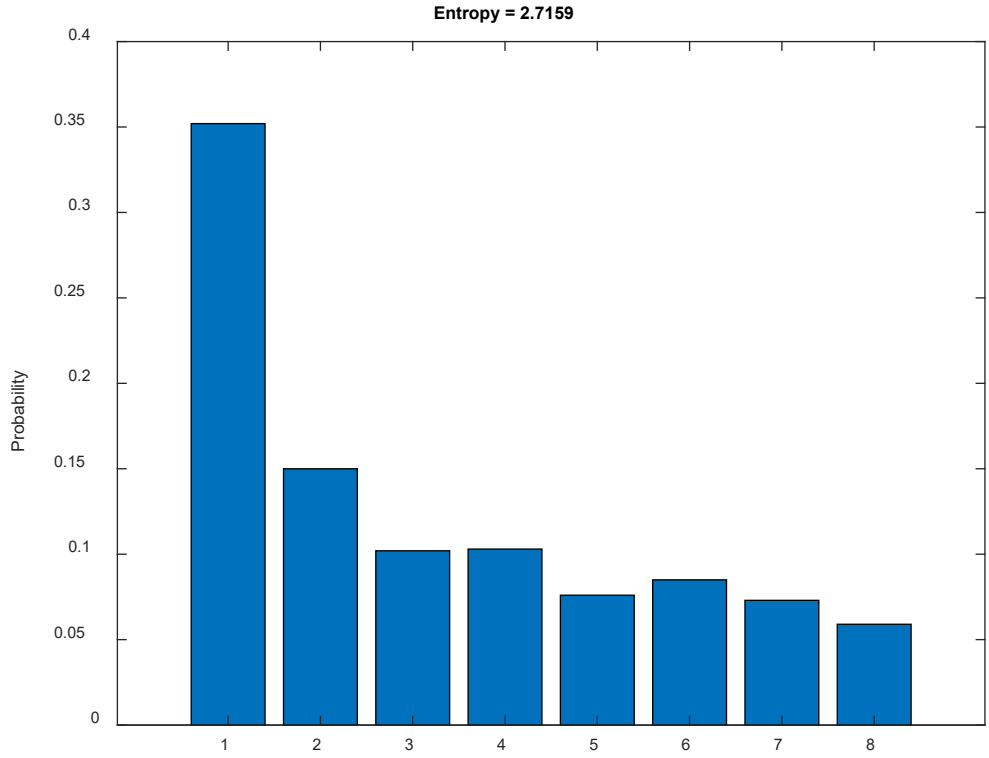
For a binary (1,0) dataset, an entropy of .5 bits indicates a random dataset

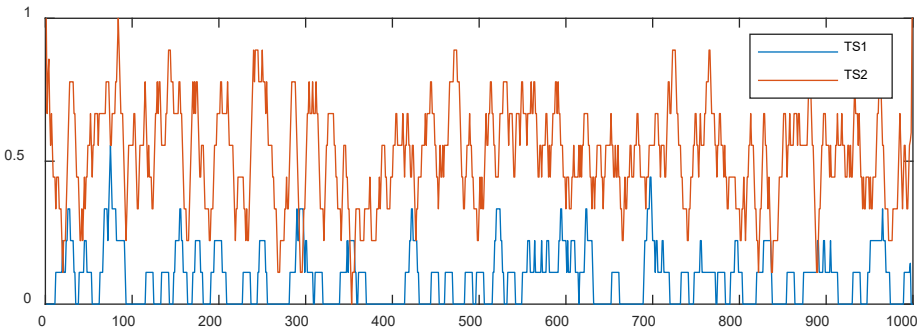
The entropy of a theoretical random string of 0’s and 1’s is .5 – such as the entropy of a bunch of coin flip trials.

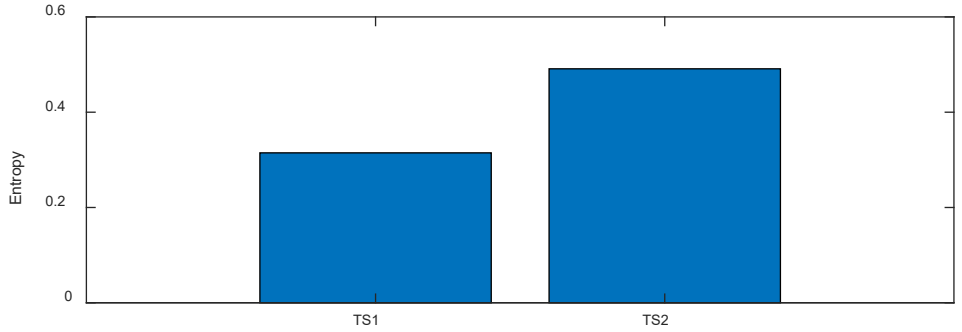
MATLAB

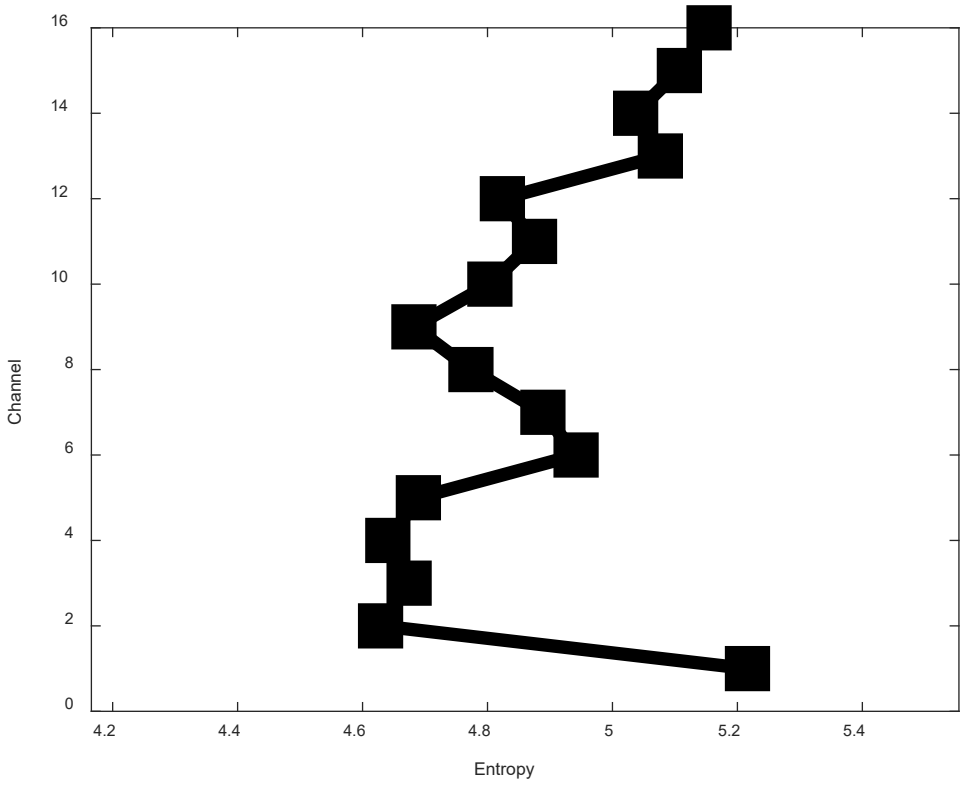
Code: sigprocMXC_entropy.m

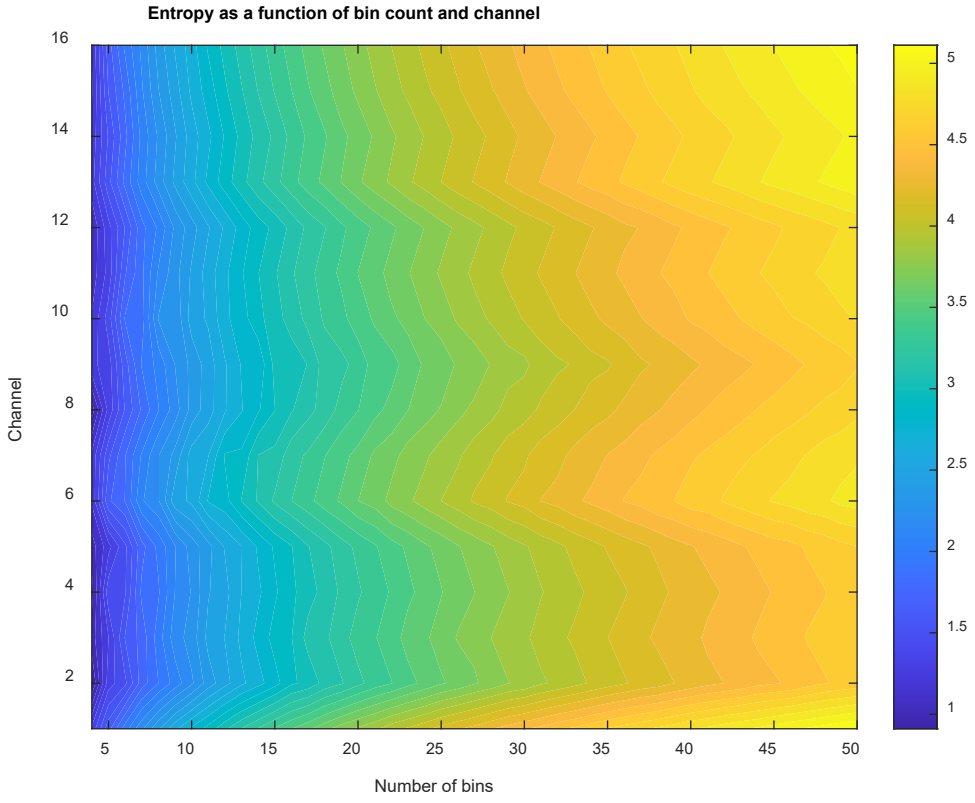
Data: v1_laminar.mat

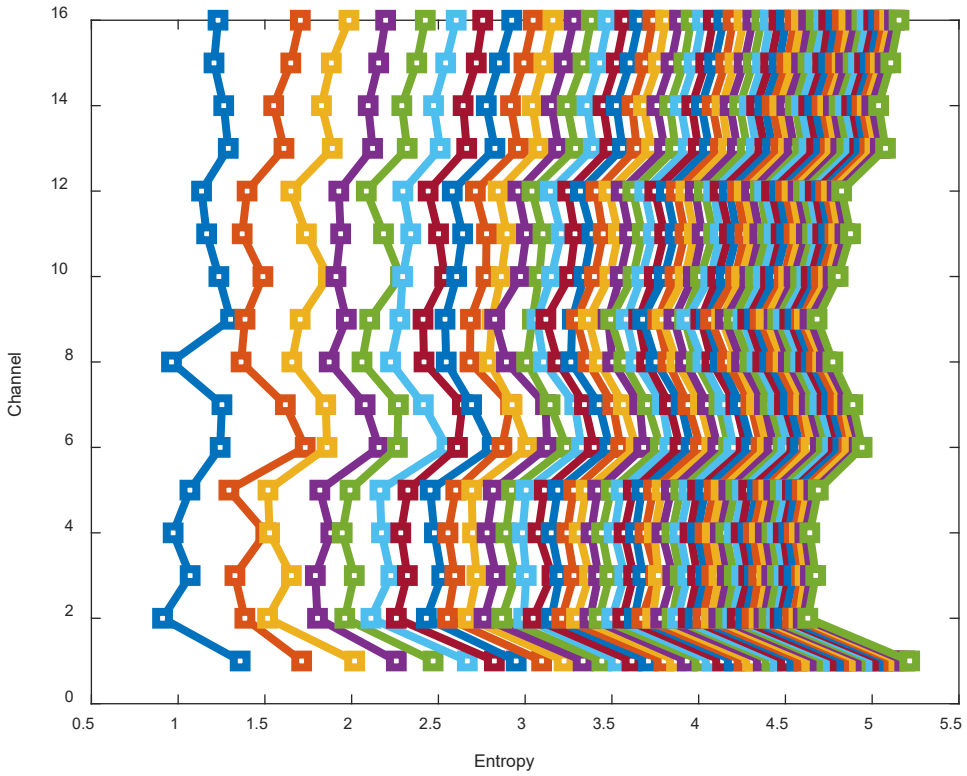



TS1
TS2
1
0.5
0
0
100
200
300
400
500
600
700
800
900
1000


Entropy
0.6
0.4
0.2
0
TS1
TS2


Channel
16
14
12
10
8
6
4
2
0
4.2
4.4
4.6
4.8
5
5.2
5.4
Entropy


Entropy as a function of bin count and channel
Channel
Number of bins
16
14
12
10
8
6
4
2
5
10
15
20
25
30
35
40
45
50
5
4.5
4
3.5
3
2.5
2
1.5
1


Channel
Entropy
16
14
12
10
8
6
4
2
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5

97. CODE CHALLENGE

Start with script: sigprocMXC_SNR.m

What is the effect of temporal filtering on the SNR at the timepoint = 375 (line 118 in code)

Apply low-pass filtering on the two channels of the erp variable (line 122 in code)

Repeat over a range of frequencies

SECTION 12. BONUS SECTION

Course discounts and discussion board

NOTES

TOOLBOXES USED IN THIS COURSE

CURVE FITTING TOOLBOX

smooth

IMAGE PROCESSING TOOLBOX

bwconncomp

SIGNAL PROCESSING TOOLBOX

STATISTICS AND MACHINE LEARNING TOOLBOX

corr (use alternate function corrcoef to avoid using this toolbox)

ZSCORE

36. IIR Butterworth filters: zscore requires Statistics and Machine Learning Toolbox

I wrote this function to replace zscore in the butterworth script. I did get the trial version of Statistics and Machine Learning Toolbox and verified that my function seems to produce the same results as the toolbox.

```
% test code
temp = myzscore(fkernB);
mean(temp)
std(temp)
% function to replace zscore in script
function zs = myzscore(X)
    zs = (X - mean(X)) / std(X);
end
```

Test code results...

mean(temp) = -8.3267e-18

std(temp) = 1

WHAT IS THE RELATION BETWEEN FFT LENGTH AND FREQUENCY RESOLUTION?

82

The frequency resolution is dependent on the relationship between the FFT length and the sampling rate of the input signal.

If we collect 8192 samples for the FFT then we will have:

$$\frac{8192 \text{ samples}}{2} = 4096 \text{ FFT bins}$$

If our sampling rate is 10 kHz, then the Nyquist-Shannon sampling theorem says that our signal can contain frequency content up to 5 kHz. Then, our frequency bin resolution is:

$$\frac{5 \text{ kHz}}{4096 \text{ FFT bins}} \simeq \frac{1.22 \text{ Hz}}{\text{bin}}$$

This is may be the easier way to explain it conceptually but simplified: your bin resolution is just $\frac{f_{samp}}{N}$, where f_{samp} is the input signal's sampling rate and N is the number of FFT points used (sample length).

We can see from the above that to get smaller FFT bins we can either run a longer FFT (that is, take more samples *at the same rate* before running the FFT) or decrease our sampling rate.

PLOTTING NOTES

PLOT

Link: https://www.mathworks.com/help/matlab/ref/plot.html

plot(x,y): creates a 2-D line plot of the data in Y versus the corresponding values in X.

stem(x,y): Plot using stems that extend from a baseline along the x-axis. The data values are indicated by circles terminating each stem.

Example: stem(x,y,'ks') where k stands for color black, and s stands for shape square

pwelch(x): Welch's power spectral density estimate

patch(X, Y, C): plot one or more filled polygonal regions.

Example: h = patch([time; time(end:-1:1)], [mean_ts+std3_ts; mean_ts(end:-1:1) – std3_ts(end:-1:1)],'m');

SUBPLOTS

subplot(211)

…

subplot(212)

…

LABELS

xlabel('Frequency (Hz)')

ylabel('Amplitude')

title('Frequency domain')

LINE STYLE, COLOR, AND MARKER

Examples:

- 'ro-', red circles connected with short lines
- 'bp-', blue stars connected with short lines
- 'g', green solid line
- 'c*', cyan stars with no connecting line

CONTROL X-AXIS

set(gca,'xlim',[0 max(frex)*3]) % example how to define range of x axis

COMMAND WINDOW CONTROLS

Datacursormode – turn on data cursor mode in the plotted figures

zoom – turn on zooming with “on”, or turn “off”

figure – create a new figure in which to plot

MATLAB OPERATORS AND SPECIAL CHARACTERS

https://www.mathworks.com/help/matlab/matlab_prog/matlab-operators-and-special-characters.html

EXTERNAL REFERENCES

MATLAB Signal Analyses Videos

- https://www.mathworks.com/videos/signal-analysis-made-easy-100811.html

Master the Fourier transform and its applications

- https://www.udemy.com/course/fourier-transform-mxc/learn/lecture/8606308?start=0#overview

Google: Explore what the world is searching

- https://trends.google.com/trends/?geo=US
- Use download to csv file to get data for time plot and fft
- See lecture 21. Fourier transform for spectra analyses

xeno-canto is a website dedicated to sharing bird sounds from all over the world.

- https://www.xeno-canto.org/
- Good source for fft and spectrogram analyses starting from mp3 birdsongs
- See lecture 23. Spectrogram of Birdsong for more details

Histdata is a website dedicated to sharing historical foreign exchange data

- http://www.histdata.com/download-free-forex-historical-data/?/ascii/1-minute-bar-quotes/eurusd/2017
- See lecture 81. Outliers via local threshold exceedance for more details

Discounts on other coursed my mike

The links below offer direct links and discounts, or use the code 202002 to get the best price for any of my courses

MATLAB COMMANDS AND FUNCTIONS

.: beginning of element-wise arithmetic on vectors or matrix - such as .+ or .^.

audioread: [bc, fs] = audioread('xc403881.mp3'); Read audio sound file with sound channels in bc, sample rate in fs.

bsxfun: Apply element-wise operation to two arrays with implicit expansion enabled.

cat(): Concatenate arrays.

ceil(): Round toward positive infinity. IE round up.

cell: Cell array – a data type with indexed data containers.

cell2mat(): Convert cell data to matrix data.

clear: Clear workspace

clf: deletes from the current figure all graphics objects whose handles are not hidden.

close(figure(1)): completely close figure 1.

corr: Statistics Toolbox, similar to corrcoef – linear or rank correlation

corrcoef: Correlation coefficient

cumsum(A): returns the cumulative sum of A starting at the beginning of the first array dimension in A whose size does not equal 1.

conv(data, event, 'same'): w = conv(u,v,shape) returns a subsection of the convolution, as specified by shape. For example, conv(u,v,'same') returns only the central part of the convolution, the same size as u, and conv(u,v,'valid') returns only the part of the convolution computed without the zero-padded edges.

deal(): Distribute inputs to outputs.

detrend(x): removes the best straight-fit line from the data in x, including slope adjustment, leaving flat line average.

diff(): differences and approximate derivatives

display(anything): display the thing in the command window. Like ending line with ";" except without the "ans ="

dsearchn(time',1): nearest point search – nearest point to the value of 1 in array time.

eps: smallest value that operating system is capable of within MATLAB, The size of computer rounding errors.

fft(X): computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.

figure(2), clf: commands to clear figure 2 plot (also see close()).

filtfilt(b,a,x): Signal processing toolbox, zero-phase digital filtering. Transfer function defined by b,a. Input signal defined by x.

Example without requiring a designfilt() function to generate b,a) from lecture 86:

- % rectify and lowpass filter
- rectsig = abs(signal);
- k = 9;
- rectsig = filtfilt(ones(1,k)/k,1,rectsig);

find(X): returns a vector containing the linear indices of each nonzero element in array X.

floor(): Round toward negative infinity. IE round down.

griddedInterpolant(time, data, 'spline'): Use griddedInterpolant to perform interpolation on a 1-D, 2-D, 3-D, or N-D gridded data set. griddedInterpolant returns the interpolant F for the given dataset. You can evaluate F at a set of query points, such as (xq,yq) in 2-D, to produce interpolated values vq = F(xq,yq). See resample() in signal processing toolbox for an alternative method.

hilbert(): Signal processing, Discrete-time analytic signal using Hilbert transform.

hist(): replace this function with historgram().

- For details why: https://www.mathworks.com/help/matlab/creating_plots/replace-discouraged-instances-of-hist-and-histc.html

histogram(X,nbins): plot X using a number of bins specified by the scalar, nbins.

imagesc: Display image with scaled colors.

interp1 (v,xq,method): returns interpolated values of a 1-D function, Vector x contains the sample points, and v contains the corresponding values, v(x). method specifies any of the alternative interpolation methods and uses the default sample points, 'pchip' method produces a smooth curve, 'linear' produces a jagged line.

isempty(A): returns logical 1 (true) if A is empty, and logical 0 (false) otherwise.

length(X): number of values in array X.

linspace(x1,x2,n): generates n points. Range is from x1 to x2. The spacing between the points is (x2-x1)/(n-1).

load data.mat: load a file containing data called data.mat.

logical(X): Convert values to 1 or 0: any non-zero value is converted to 1, otherwise 0.

mean(A): Average or mean value of array

median(A): returns the median value of A.

min example: [M,I] = min(A): returns the minimums of matrix A in M, and the indices for M in I.

polyfit(): polynomial fit (returns coefficients).

polyval(): predicted data, evaluation of polynomial.

pwelch: [pxx,f] = pwelch(x,window,noverlap,f,fs); Welch's power spectral density estimate.

rand(n,1): returns an n-by-1 column vector of uniformly distributed random numbers between 0 and 1.

randn(n,1): Generate an n(rows)-by-1(column) matrix of normally distributed random numbers (mean=0 and std=1).

Another example:

- % optional: add noise to a signal
- signal = signal + randn(size(time));

randperm(n): returns a row vector containing a random permutation of the integers from 1 to n without repeating elements.

resample(data, p, q): resample(x,p,q) resamples the input sequence, x, at p/q times the original sample rate. If x is a matrix, then resample treats each column of x as an independent channel. resample applies an antialiasing FIR lowpass filter to x and compensates for the delay introduced by the filter. This function assumes that the data goes to zero at both sides of the time series.

rat(newSrate/srate): rat(X) returns the rational fraction approximation of X to within the default tolerance, 1e-6*norm(X(:),1). [N,D] = rat(___) returns two arrays, N and D, such that N./D approximates X, using any of the above syntaxes.

sign(): sign (signum) function:

Y = sign(x) returns an array Y the same size as x, where each element of Y is:

- 1 if the corresponding element of x is greater than 0.
- 0 if the corresponding element of x equals 0.
- -1 if the corresponding element of x is less than 0.
- x./abs(x) if x is complex.

soundsc(bc, fs): sends audio signal y to the speaker at sample rate fs.

spectrogram: Spectrogram using short-time Fourier transform.

std(A): Standard deviation.

unique(A): Unique values in array, with values sorted by default.

whos: Display data contained in a mat file after loading (ie the Workspace).

zeros(sz1, sz2): create an array of zeros sz1 x sz2.

TOGGLETOOLBOX

- Utility to switch MATLAB toolboxes on or off
- https://stackoverflow.com/questions/40731035/how-can-i-temporarily-disable-a-matlab-toolbox

MATLAB TO C++

- https://github.com/jonathf/matlab2cpp
- http://arma.sourceforge.net/faq.html#related

NOTES ON MATLAB SCRIPT/CODE TOOLBOX DOCUMENTATION

http://citebay.com/how-to-cite/matlab-symbolic-toolbox/

Dependency Reporting

On the Current Folder pane, click, and then select Reports > Dependency Report.

MATLAB File List

sigprocMXC_filterGlass

toolbox : \signal\signal\filtfilt.m

toolbox : \signal\signal\fir1.m

toolbox : \signal\signal\hann.m

toolbox :? Multiple class methods match spectrogram.m

Unable to determine which of the following files will run : (Learn More)

\signal\signal\spectrogram.m

C:\Program Files\MATLAB\R2019b\toolbox\signal\signal\@tall\spectrogram.m

Where are we in software documentation?

Donald E. Knuth, Literate Programming, 1992

Robert C. Martin, Clean Code, 2009

Kent Beck, Four Rules of Simple Design, 2009

https://en.wikipedia.org/wiki/Code_smell

VECTORIZING CODE IN MATLAB

https://www.mathworks.com/videos/vectorizing-code-in-matlab-97131.html

Doug Hull, MathWorks

(Originally posted on Doug's MATLAB Video Tutorials blog.)

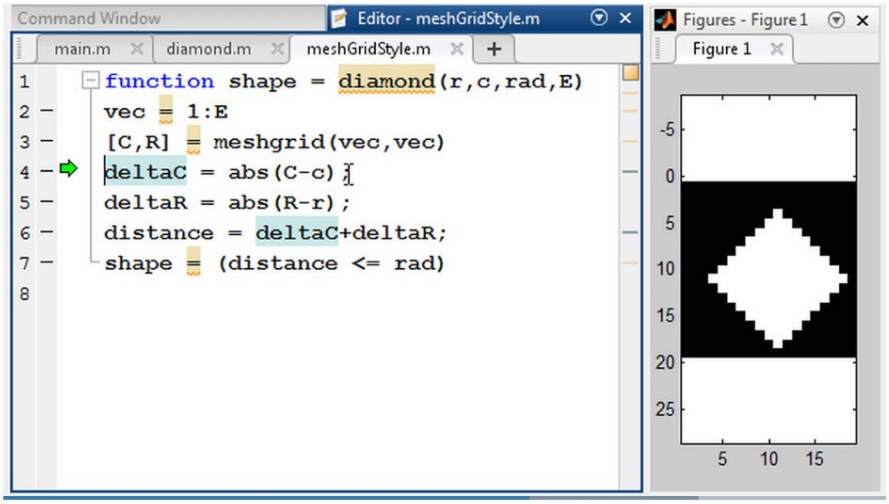
When you vectorize code, you avoid looping through an array and instead do operations on the matrix level. This avoids for loops in MATLAB. The reason for vectorizing are often stylistic. Vectorizing is preferred because it can make your code shorter, easier to read and maintain. This simple example contrasts for loop and vectorized implementations of the same algorithm.

Product Focus

MATLAB

Recorded: 4 Jun 2014

VECTORIZED FUNCTION

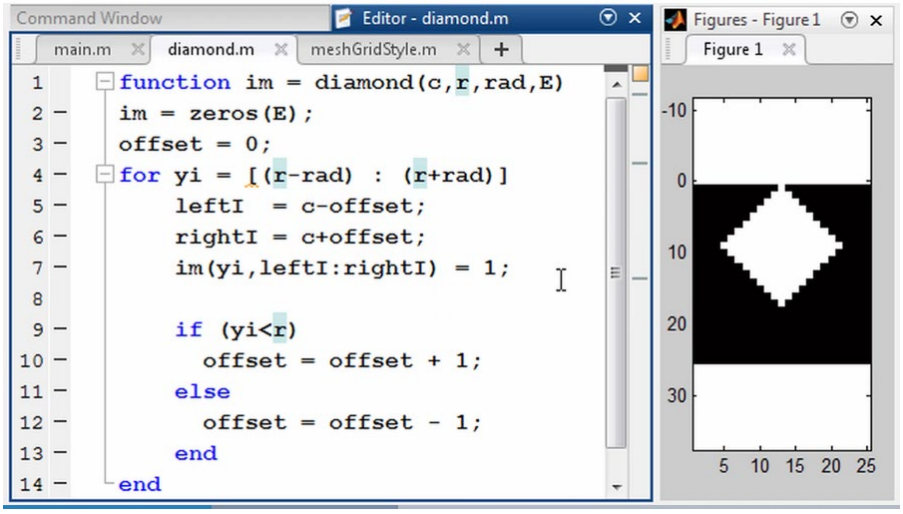


Feedback

Vectorizing Code in MATLAB

Doug Hull, MathWorks

ORIGINAL FUNCTION



NOTES

DRAQ POLYNOMIAL FUNCTIONS

```
%% Create a 4-quadrant graph with polynomials

figure(1), clf

% grid
% h1 = plot(zeros(2,1),gridSize,'k--'); hold on
% h1.Color(4) = gridAlpha; % set alpha value
% h2 = plot(gridSize,zeros(2,1),'k--');
% h2.Color(4) = gridAlpha; % set alpha value
x = linspace(-5,5);

y1 = x;
% line
plot(x,y1,'linew',1)
hold on
% parabola
y2 = x.^2;
plot(x,y2,'linew',1)
% s
y3 = x.^3;
plot(x,y3,'linew',1)
%
y4 = x.^4;
plot(x,y4,'linew',1)
```

```
xticks([-5 -4 -3 -2 -1 0 1 2 3 4 5]);
xticklabels({'-5', '-4', '-3', '-2', '-1', '0', '1', '2', '3', '4', '5'})
legend('y=x', 'y=x^2', 'y=x^3', 'y=x^4')

% details
xlabel('x-Axis', 'FontSize', 16)
ylabel('y-Axis', 'FontSize', 16)
grid on
axis square
```