

47

© M. Iskander Fall 2020



EE 421: Digital and Analog Signal  
Processing in Telecommunications

## (Lab 3 & 4) Time Series Denoising Mini-Project

Fall 2020

Due Sunday, November 8, 2020  
At 11:59 pm

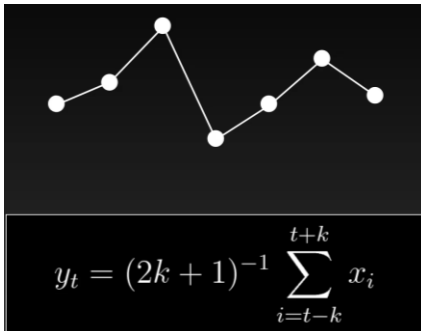
47

48

© M. Iskander Fall 2020



Running-mean time series filter



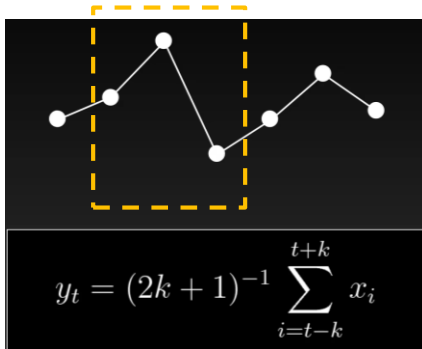
48

49

© M. Iskander Fall 2020



## Running-mean time series filter



- $k$  is the order of the filter

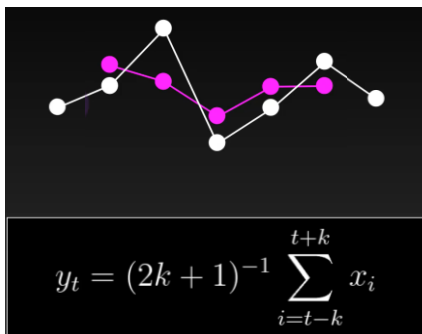
49

50

© M. Iskander Fall 2020



## Running-mean time series filter



- Example of what the smoothed version of this signal might look like.

the white line is  $x$  -- that's the original signal

the pink line here is  $y$  -- that's the filtered signal.

- The larger  $k$  is, the smoother this time series is going to be.

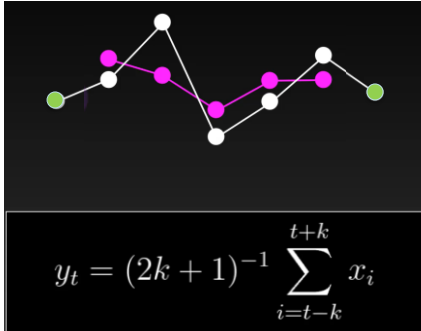
50

51

© M. Iskander Fall 2020



## Running-mean time series filter



- Something funny happening at the edges, it is called “edge effects”
- It happen at the edges of the time series when you apply a temporal filter
- What do you do with those edge points?
  - Option 1: set them to the original signal
  - Option 2: Ignore them

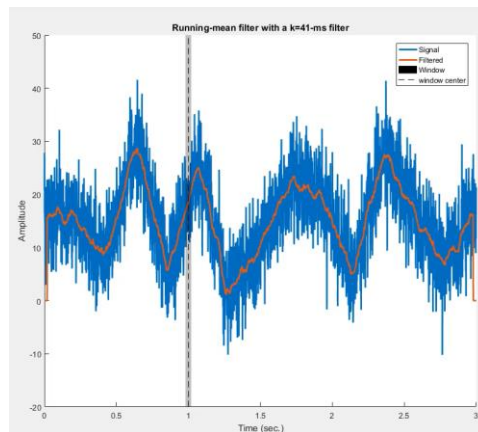
51

52

© M. Iskander Fall 2020



## Running-mean time series filter



\*Edges are set to 0

52

53

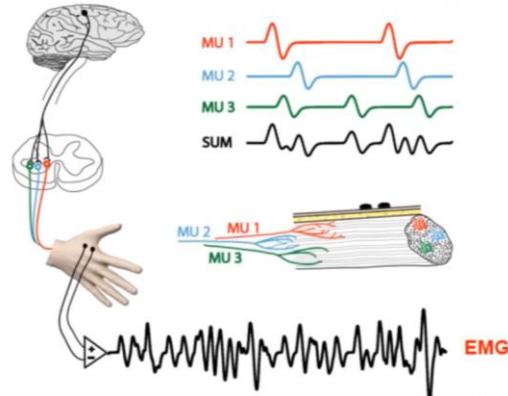
© M. Iskander Fall 2020



# Denoising EMG signals

## EMG: Electromyogram

EMG as a measure of motor unit activity



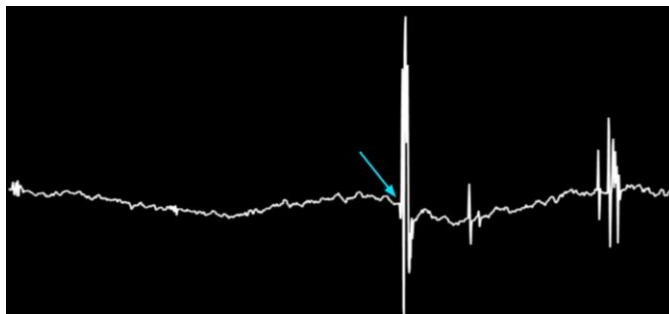
53

54

© M. Iskander Fall 2020



# Denoising EMG signals



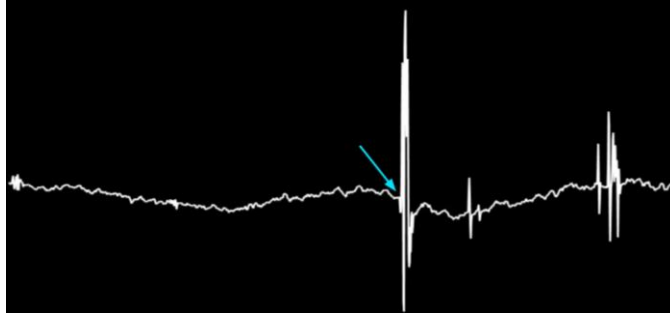
54

55

© M. Iskander Fall 2020



## Denoising EMG signals via TKEO algorithm



- Teager-Kaiser energy-tracking operator

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

55

56

© M. Iskander Fall 2020



## Denoising EMG signals via TKEO algorithm

```
% import data
load emg4TKEO.mat

% initialize filtered signal
emgf = emg;

% the loop version for interpretability
for i=2:length(emgf)-1
    emgf(i) = emg(i)^2 - emg(i-1)*emg(i+1);
end
```

Name	Size	Bytes	Class	Attributes
emg	1x1281	5124	single	
emgtime	1x1281	10248	double	
fs	1x1	8	double	

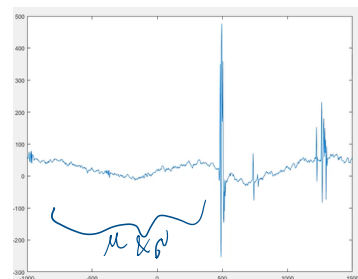
$$y_t = x_t^2 - x_{t-1}x_{t+1}$$

Comparing EMG<sub>filtered</sub> & EMG

```
% convert both signals to z-score
% find timepoint zero
time0 = dsearchn(emgtime',0);

% convert original EMG to z-score from time-zero
emgZ = (emg - mean(emg(1:time0))) / std(emg(1:time0));

% same for filtered EMG energy
emgZf = (emgf - mean(emgf(1:time0))) / std(emgf(1:time0));
```



56

57

© M. Iskander Fall 2020



## Denoising EMG signals via TKEO algorithm

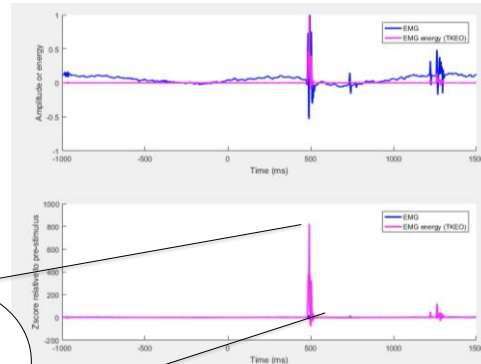
```
% plot
figure(1), clf

% plot "raw" (normalized to max-1)
subplot(211), hold on
plot(emgtime, emg./max(emg), 'b', 'linewidth', 2)
plot(emgtime, emgf./max(emgf), 'm', 'linewidth', 2)

xlabel('Time (ms)'), ylabel('Amplitude or energy')
legend(['EMG'; 'EMG energy (TKEO)'])

% plot zscored
subplot(212), hold on
plot(emgtime, emgz, 'b', 'linewidth', 2)
plot(emgtime, emgzf, 'm', 'linewidth', 2)
```

Filtered signal, goes up to a z-score of 800. So that means that it's 800 standard deviations greater than the pre-stimulus mean, or this baseline period here. In contrast, the original signal goes up to somewhere in the order of maybe ~20.



You can imagine if you are writing an algorithm to detect a sharp change in the signal, it's going to be much easier to find a change in the pink signal compared to the blue signal.

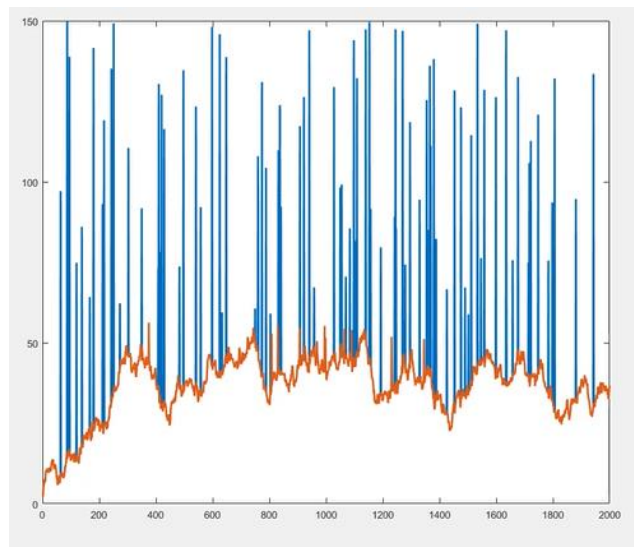
57

58

© M. Iskander Fall 2020



## Removing spike noise using median filter



58

59
© M. Iskander Fall 2020
ASHESI

## Removing spike noise using median filter

```

% use hist to pick threshold
figure(1), clf
histogram(signal,100)
zoom on

% visual-picked threshold
threshold = 57;

% find data values above the threshold
suprathresh = find( signal>threshold );

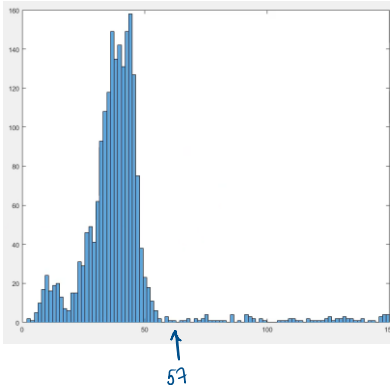
% initialize filtered signal
filtsig = signal;

% loop through suprathreshold points and set to median of k
k = 20; % actual window is k*2+1
for ti=1:length(suprathresh)

    % lower bound
    lowbnd = max(1,suprathresh(ti)-k);
    uppbnd = min(suprathresh(ti)+k,n);

    % compute median of surrounding points
    filterig(suprathresh(ti)) = median(signal(lowbnd:uppbnd));
end

% plot
figure(2), clf
plot(1:n,signal, 'm',filtsig, 'b');
zoom on
```



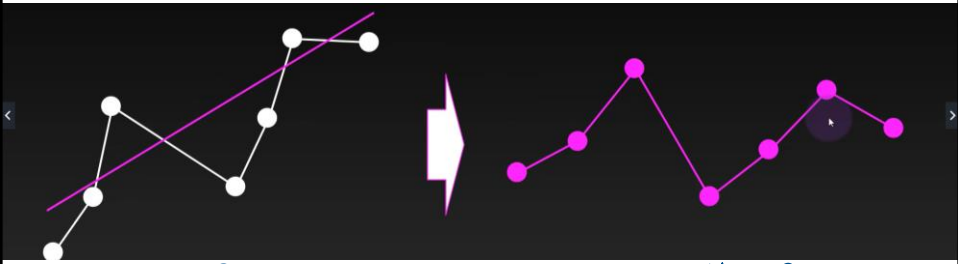
↑  
57

to prevent errors  
e.g. if spike happened a  
index 3  
you can't get median  
from -17 to 23

59

60
© M. Iskander Fall 2020
ASHESI

## Linear detrending



$\mu \neq 0$

$\mu \approx 0$

```

% linear detrending
detsignal = detrend(signal);
```

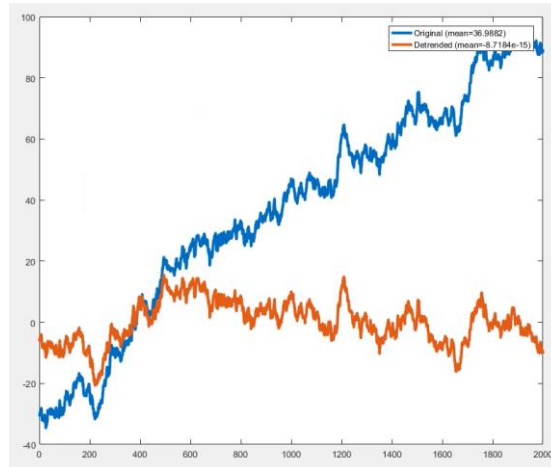
60

61

© M. Iskander Fall 2020



## Linear detrending



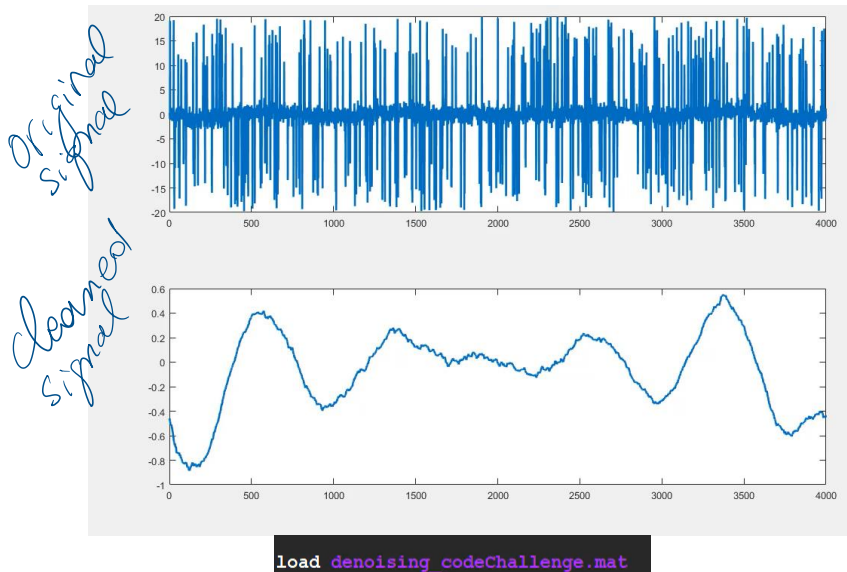
61

62

© M. Iskander Fall 2020



## Code Challenge



62



## Code Challenge



Load **denoising\_codeChallenge.mat**



Figure out how to clean the noise signal to get a signal that looks like my clean signal



You might need to use more than one denoising strategy



Your result might not look exactly like the clean signal in every tiny little detail.... The point is to get your clean signal more or less like my clean version