Denoising:

* **Running mean time series** - this filter is good for a signal with normal distributed noise:



* **Gaussian smoothing time series** – like mean smoothing but a bit smoother product:



W is the full width at half maximum. Higher w means wider window

* TKEO:



In this way we can **highlight peaks and anomalies in data** which is quite noisy but also has pulses or peaks

* **Find median to remove spike noise – good for spikes only**:

First sort the vector, then find the middle value which is the median.

If we have an even number of numbers in the vector, we average both middle values after sorting.

In both we pick a threshold for what the highest (or lowest depends on the peak) we approve and then replace it with the median. One good way to choose threshold is by making a histogram of all the points and see where the gaussian bell ends

* **Linear detrending:**

Just using detrend to detrend a linear trend in data (after visualizing it)

* **Remove nonlinear trends from data** (remove fluctuations or drifts in data):

It is like linear detrending but now we must use baes criteria to determine what’s the order of the polynom we are fitting to. Bayes information criterion:



It evaluates the fit of a model to the data. Epsilon is the distance between the modeled polynom and the actual data. N is the number of data points k is the number of parameters (how many orders we have in the polynom) y with hat is the predicted data and y without is the real data. Then we plot b against polynomial order and pick the **lowest b. we can also find lowest b by looking for min in b vector**

* Averaging multiple repetitions in order to get better SNR. Simple average but we need to know when it happens or if it is a recuring event in time (מחזורי).
* Remove artifact via least-squares and template matching. Good if we have a second channel that records the artifacts. The least square algorithm:



Where beta is the regression weights X is the design matrix, y is the looked-on data, r is the residual and X times beta is the predicted data. This all is real matrixes. X matrix is two column vector the first column is all 1’s and the second is the times series of the artifact vector

Fourier transform:

* **Nyquist** is half the sampling rate.
* We should always remember to **factor out the DC frequency**. We can do that in two ways:
* One is detrending. But then a real trend might be missed so it doesn’t always fit.
* Subtract the mean value from the data (mean centering). But then the first point is not real but stems from a trend if it is present
* **Welch's method:** after ‘cutting’ the signal into pieces apply it to each part independently. We should always apply Hann window on every piece to minimize edge effects.
* Welch’s method is not always optimal if the frequencies are changing a lot. Then we would like to use wavelets
* **We can see time and frequency domain at the same time on a spectrogram. Its frequency as function of time and it gives a heat map representing the amplitude.**

Digital Filtering:

* **Procedure for filtering data:**

1. Define frequency domain shape and cut-offs.
2. Generate filter kernel
3. Evaluate kernel and its power spectrum
4. Apply filter kernel to data

* **FIR vs IIR:**

|  |  |  |
| --- | --- | --- |
| Name | FIR | IIR |
| Full name | Finite Impulse Response | Infinite Impulse Response |
| Kernel length | Long | Short |
| Speed | Slower | Fast |
| Stability | High | Data-dependent |
| Mechanism | Multiply data with kernel | Multiply data with data |

* **FIR kernels with firls: finite impulse response Least-squares linear-phase kernels:**

With filters we should always stay with simple window on the frequency domain. The kernel goes from zero (DC) to Nyquist they are always normalized by nyquist frequency. the gain goes until one (the amplitude of the window). The window of band pass filter goes to a maximum between the two frequencies we want to pass through it. We don’t want the edges of the kernel on the freq domain because it requires too much energy – this is the transition zone. The order (number of points in the kernel on the time domain) is usually making the kernel better with higher numbers but there is a limit to it where after the limit it will be worse and take longer to compute.

* **FIR1 function –** calling firls without transition width. Fir1 will apply a window instead of adding transition width. **This function is best used when needed maximum attenuation at the boundaries of the filter.**
* **IIR Butterworth filters** – only need to specify two points lower and upper frequencies for band pass or one point for high/low pass. Iir filter orders are much lower than fir filters. For IIR filters there are two coefficients. There are two sets of weights, one for the already filtered signal (A) and B is for the signal we are yet to filter. The right way to estimate IIR filter is by plotting impulse response.
* **Causal and zero phase-shift filters** - what happened in the signal in the past affect the filtered signal in the future (and present) which is causing some phase shift in the filtered signal. To normalize this shift, we first flip the filtered signal. Then we apply the same filter kernel again on the flipped signal and then flip it back again. If I have the MATLAB signal processing package I can just use filtfilt() to apply zero phase shift filtering.
* **Avoid edge effects with “reflection”** – we take the signal and reverse it then add it at the end and the start of the original signal. Its mostly enough to only reflect the signal in the length of the filter kernel. Again filtfilt() already does it. We can also use reflection when using filtfilt() – this function requires the signal to be at least 3 times the kernel length so using reflection we are making the signal much longer than the kernel while not adding new data.
* **Windowed sinc filters** – defined by . This is a low pass filter, and good because it falls very fast (attenuate fast) at the cutoff. we should apply a window (similar to fir1()) to the sinc kernel because it makes a lot of edge effects if not.
* **Improving narrow band filter** – increase its order, should have same transition zones on both sides of the kernel.
* **For wide band filtering we will want to do two separate filters – one high pass filter for the lower boundary and the low pass filter for upper boundary.**
* **Quantifying roll-off characteristics** – the roll off is the decrease in amplitude of the filter kernel with the increase of the frequencies. We quantify it by:

****

where is the frequency where the gain (amplitude of the filter kernel) reaches -3dB.

* **Convolution:**
* The length of the convolution result is the length of the signal plus the length of the convolution kernel minus one.
* When doing a convolution, we should flip the kernel before applying.it is flipped because convolution theorem says so but also, when you flip, then the convolution with an impulse response function of a system gives you the response of that system. If you don't flip, the response comes out backwards.
* **Instead of convolution in the time domain we can transform both signal and kernel to the frequency domain and multiply them. This is the convolution theorem.**
* **Convolution with time domain gaussian gives a smoothing filter. Convolution with a frequency domain gaussian gives a narrowband filter.**
* Gaussian in the frequency domain:



Where h is the frequencies vector, p is the peak frequency and w is the FWHM.

* **Convolution with frequency domain planck taper gives a bandpass filter.**