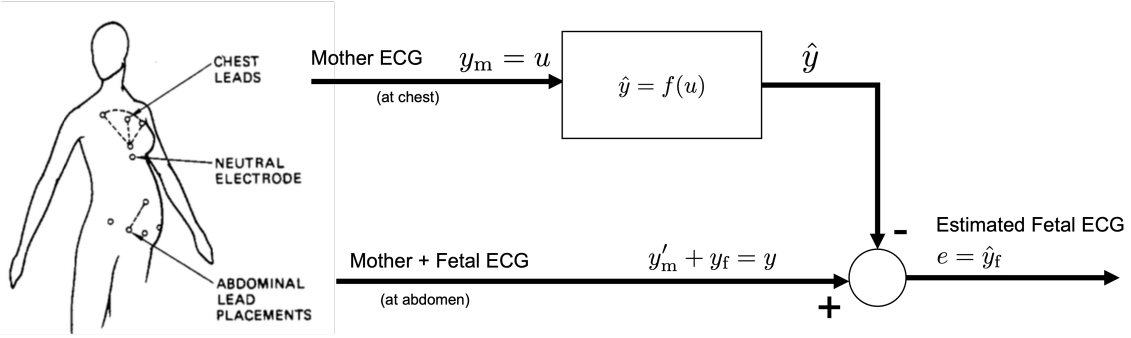
Exercise Set 10 - Linear Filter Estimation

2024/05/02

Model-Based Signal Processing and Medical Diagnostics

R. Wildhaber
F. Waldmann

Exercise 10-1: Wiener Filter for Fetal ECG Reconstruction `[to be submitted]`



In this exercise, we isolate the fetal ECG signal of a mother's ECG recording using a Wiener filter. For that, we have a two-channel recording with a first channel signal `y_maternal` and a second channel signal `y_fetal_maternal`, as shown in the figure above. Signal `y_maternal` is measured on the mother's chest and is therefore sensing the mother ECG only. Signal `y_fetal_maternal` is recorded on the mother's abdomen, sensing a superposition of the mother's and the fetal's ECG.

Tasks:

(a) Start with the provided template `wiener_filter_template.py`. Complete the marked lines of code of the function `wiener_filter()` to implement a Wiener filter according to the lecture notes.

(b) Continue with template `ex-10.1-Wiener-Filter-fetal-ECG-reconstruction-template.py`. Apply the `wiener_filter()` on `y_maternal` and `y_fetal_maternal` to get `y_fetal` (Fetal ECG).

(c) `[optional]` Rerun the code with the clean test signals `y_maternal_clean` and `y_fetal_maternal_clean`. Consider and interpret the estimated filter coefficients h (printed to the terminal).

Provided Signals and Corresponding Python Variables:

```
y_maternal = data[:, 0] # Mother ECG (chest)
y_maternal_fetal = data[:, 1] # Mother and fetal ECG (abdomen)
y_fetal_true= data[:, 2] # True fetal ECG at abdomen
y_maternal_clean = data[:, 3] # Noise-free mother ECG (chest)
y_maternal_fetal_clean = data[:, 4] # Noise-free mother with fetal ECG (abdomen)
```

Exercise 10-2: LMS Filter for a Fetal EGG Reconstruction [to be submitted]

Use a LMS filter to again isolate the fetal ECG from the mother ECG as in the previous exercise, based on the same two-channel recording `y_maternal` and `y_fetal_maternal`.

Tasks:

(a) Start with the provided template `lms_filter_template.py`. Complete the marked code lines of the function `lms_filter()` to implement an LMS filter according to the lecture notes.

(b) Continue with template `ex-10.2-LMS-Filter-fetal-ECG-reconstruction-template.py`. Apply the `lms_filter()` on `y_maternal` and `y_fetal_maternal` to get `y_fetal` (Fetal ECG).

(c) Change the step-size `L` of the LMS filter. How does it influence the result?

Exercise 10-3: Adaptive Echo Cancellation [advanced, optional]

Introduction Echo cancellation is the suppression of echos in a signal. A common method for this are adaptive filters which is implemented in most mobile phones and communication applications such as Zoom, Teams, etc.

For this exercise, we consider a scenario where two participants (p1 and p2) are in an online call. The microphone input of p1 is processed with echo cancellation (filtered) before being transmitted to p2, and vice versa. This is to avoid positive feedback loops since the microphone of p1 not only records the voice of p1, but also the speaker output with the voice of p2.

Implementation Isolate the voice of p2 from the recording by canceling all correlated information from p1. The audio file `audio.wav` contains two recordings (left and right channel); the left channel `y_clean` is the received sound (p2), the right channel `y_mix` the recording of voice p1 and the echo of p2 (injected via loudspeakers).

Note: Experimenting with audio signals can cause errors such as jitter and sample peaks, leading to high, unpredicted sound pressures. Make sure to set the volume to the minimum before listening to a new sound output.

Tasks:

(a) Complete the template `ex-10.3-Echo-Cancelling-template.py` to apply the `winer_filter()` on `y_mixedl` and `y_cleanl` to get `y_out` (solely voice of p1). Listen to the file `audio_processed.wav`.

(b) Change the filter by setting `USE_Filter="WIENER"`. What has changed?

Exercise 10-4: Adaptive Filter to Remove Powerline Interferences [optional]

Work with signal `secg-poweline-50Hz-fs-512Hz.csv`. Implement an adaptive filter to improve the ECG quality. This signal also provides a reference signal as a "ground truth" to verify your result (column y in the data table).

Tasks:

a) Extract from the powerline signal a sparse reference signal, which is set to 1 at each sine wave peak and 0 otherwise, i.e.,

```
ind, _ = find_peaks(y_powerline, distance=10)
y_dirac_comb_50hz = np.zeros_like(t)
y_dirac_comb_50hz[ind] = 1  # generate a 50 Hz pulses
```

b) Call the `lms_filter()` function with the correct input and reference signals

```
y_hat, h_hat, error = lms_filter(u, y, L, mu)
```

Hint: Use a filter length of $L = 50$ and an update rate of `beta` $= 0.02$ as a starting point.

Provided Signals and Corresponding Python Variables:

```
t = ecg_data_table[:, 0] # time
y_ecg_powerline = ecg_data_table[:, 1] # observations
y_ecg = ecg_data_table[:, 2] # "true" signal
y_powerline = ecg_data_table[:, 3] # powerline
```