RSA SYNCHRONY
GENERATING INPUT DATA

Goal

We want to generate input data for the RSA synchrony algorithm by Drew Abbney that leads to an output that indicates a high level of synchrony. This data can then be used as a baseline for testing, validating and comparing the algorithm.

Input data

- Two 1D-arrays of **inter-beat intervals** in milliseconds for mother and child
- The length of the arrays is variable but both **add up to the same number**, the recording length. There are min/max values for the recording length.
- There are min/max values for **mother and infant IBIs based on physiological data**

Algorithm

- calculate continuous RSA of mother and infants
    - **resample** both IBI signals to 5 hz
    - **split** resampled IBI into BPM and raw RSA using polynomial filter: BPM is a trend (low frequency) and RSA is filtered data (high frequency)
    - apply specific **filters** to raw RSA of child and mother (see below)
    - Using a sliding window of `window_size=15s,` calculate **`log(var())`** of the filtered raw RSA (logarithmically scaled changes of variability). This is the **continuous RSA.**
- d**etrending:** remove linear dependencies by subtracting first derivative from continuous RSA
- calculate **cross-correlation** between detrended signals
- select **zero-lag coefficient as measure of synchrony** (correlation value for `phase_shift=0`)

Output data

The main feature of interest is the **zero-lag coefficient**. Large positive values indicate strong synchrony, large negative values indicate strong inverse synchrony. Small values indicate little or no synchrony. Therefore, **the target is a zero-lag coefficient that lies within a range of large values.** For validation, also the full cross-correlation function (ccf) should have similar characteristics as the ones resulting from real physiological data. These are yet to be determined but generally, the ccf should include smooth changes rather than crazy jumps.

Side-note about the algorithm

It might make sense to normalise the detrended RSA signals of mother and infant before calculating the cross-correlation. XCorr is both dependant of the shapes and the absolute magnitudes of the input signals. As we are more interested in dynamics, we can minimise the influence of absolute magnitudes by applying normalisation to the signals.

Approach A: Machine learning for input data generation

It is possible to use machine learning for automatically understanding the relationship between input data and output data and therefore enabling input data generation that yields desired outputs. Using for example `sicket.learn`, we can optimise the parameters of a function `generateDyadIBI` based on an error function `evaluateZeroLagCoefficient`.

`generateDyadIBI`

The exact inner workings of this functions are yet to be determined.

- Constraints:
  - s*um constraint:* Both IBI series must add up to the same sum, the recurring length. This can be a fixed constant value
  - *amplitude constraint*: the magnitude of each sample must fit into the min/max range for adults/children based on physiological data
- Parameters (idea):
  IBI signals can be expressed as the sum of scaled frequency components using the Fourier Transform. The parameters to be learned by the ML algorithm could be the vector of scalars for a set of frequency bands. Possibly, the number of bands and their associated frequencies could be variable and subject to the ML algorithm.

`evaluateZeroLagCoefficient`

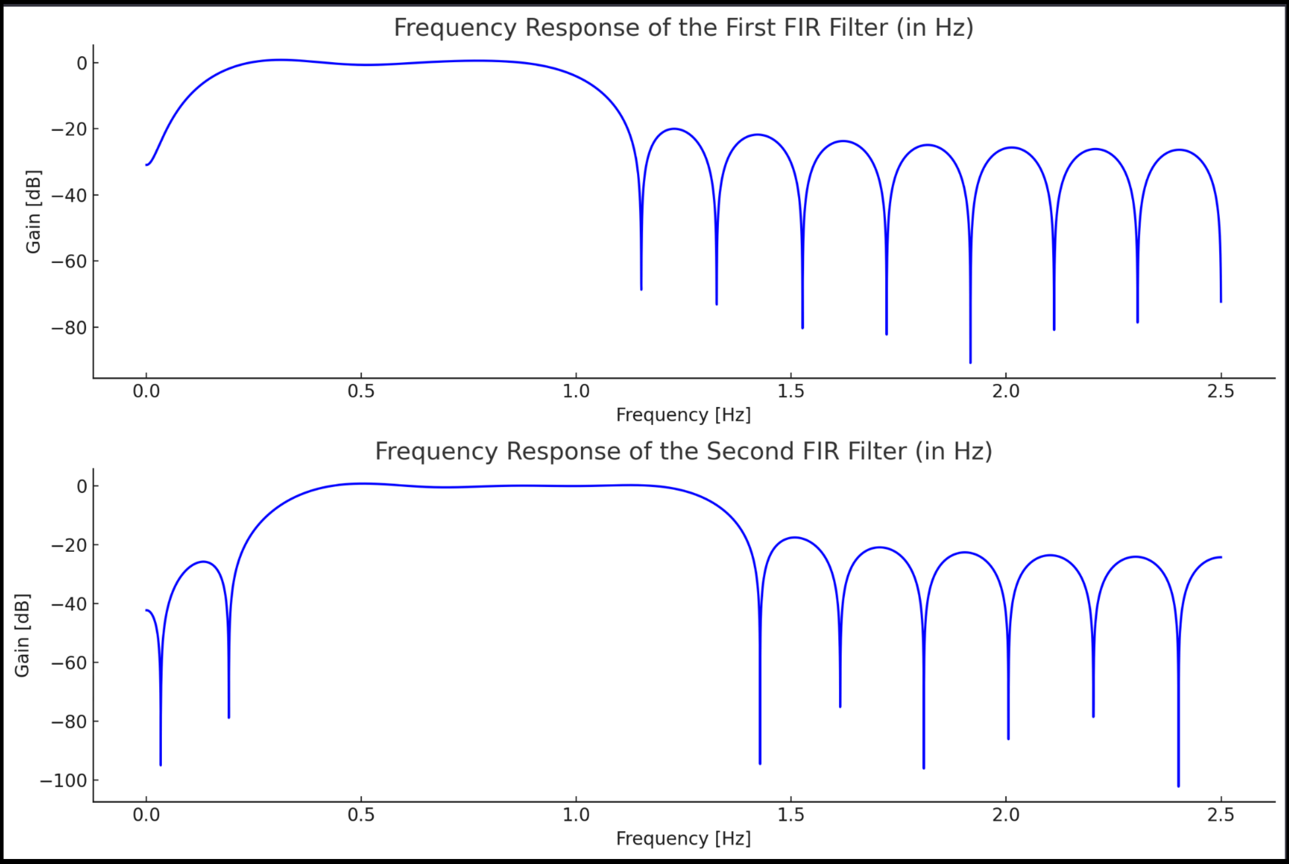
The error function will first run the data through the RSA synchrony algorithm in order to obtain a value for the zero-lag coefficient. This value is then either compared to an optimal value or maximised. The error value is then either the difference to the optimum or the difference to the current maximum.

Approach B: Base pattern, synchronised variation

In general, we should obtain high RSA synchrony values if there is synchronised variation in the two IBI series. To simulate this, we can start with generating base patterns for adult and child IBI based on average heart rates like 100bpm for the child and 70bpm for the adult. Then, a random variation pattern is generated that is used to scale the samples of both patterns synchronously. This approach is much faster to implement so it should probably be the starting point.

Addition: Frequency responses of filters

For signals sampled at 5hz, these are the frequency responses of the filters used in the Algorithm. The first filter is the one used for adults (`adult_rsa_5Hz_cLSq.mat`), the second one for children (`child_RSA.mat`).


Frequency Response of the First FIR Filter (in Hz)
Gain [dB]
Frequency [Hz]
Frequency Response of the Second FIR Filter (in Hz)
Gain [dB]
Frequency [Hz]