From the EEG recordings to the volum

Report Names: Fanny Grosselin	Verification Name: Yohan Attal Date: Signature:
History of modifications: - *Creation 27/02/2017* : Fanny Grosselin Description of the pipeline used to compute the relaxation index as a Signal to Noise Ratio(SNR) of the alpha peak and to transform it into volum scale.	

Introduction:

This document lists and describes the pipeline to compute the relaxation index from EEG data and to transform it into volum scale in the Melomind app. MeloMind is a new solution based on ElectroEncephaloGraphic (EEG) technology that allow us how to learn to relax efficiently. In the first version of Melomind, we have decided to focus on the Signal over Noise Ratio (SNR) of the maximal value of the spectrum in the alpha range (7-13Hz) as a relaxation index. In fact, a high level of alpha waves is in general related to a low level of arousal and this index could be used to learn people to increase their alpha power in order to learn to decrease their stress and arousal (Brown, B. *New mind, new body*. New York: Harper and Row. 1974). In addition, the studies about brain modification during a relaxation state induced by meditation, mindfulness, tai chi, hypnosis have shown an increase of the alpha waves ("Tai chi/yoga effects on anxiety heart rate", EEG and math computations, Field et al. 2010; "Conducting art therapy research using quantitative EEG measures", Belkofer et al. 2008; "Increased theta and alpha EEG activity during non-directive meditation", Lagopoulos et al. 2009; "Meditation and the EEG", West 1980).

Before describing how we measure the relaxation index, we need to describe the phases inside a Melomind session. When the user start Melomind, there are 2 main recordings:

- We start recording EEG data for a trial period to compute some parameters needed during the neurofeedback session. This is called "calibration".
- After calibration, some processes are done and the neurofeedback session can start. This is called "session". During this recording, neurofeedback based on the relaxation index is applied as audio tracks and it depends on each subject at each particular moment.

During both recordings we compute the relaxation index as a Signal Over Noise Ratio (SNR) of the alpha peak each second on a EEG signal of 4s with a sliding window. So we have 1 relaxation index by second but we need to wait for 3s at the beginning of the session (see figure 1). This relaxation index is transformed in a volum scale to give to the user a feedback of its own alpha activity. In fact, in the audio track, one or several component of the music will have their volum modulated in function to the subject's relaxation index at each specific time.

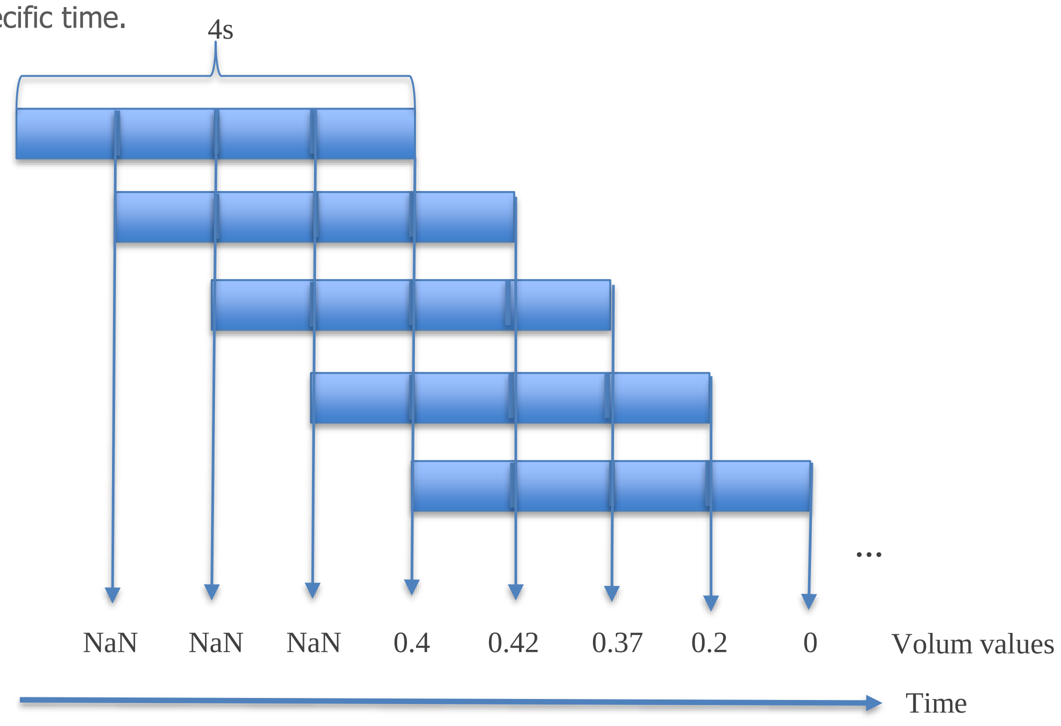


Figure 1: Description of the sliding window used to compute relaxation index.

Now, we will describe in more details what are the steps to compute the relaxation index (called SNR) and how is it translated into volum at each second.

1. *Processes during calibration:*

During calibration, we receive 2 EEG signals of 250 datapoints. These data are sent (locally on the smartphone) in an array named initCalibrationData which is the array which will be send to the server as a JsonFile at the end of calibration. These data are also sent (locally on the smartphone) to another array named rawCalibrationData which contains all the data already recorded but with Not a Number (NaN) values linearly interpolated. These NaN values can appear instead of EEG values if we lose data packet(s) during the Bluetooth transmission. For each new 1-second packet, if we detect NaN values, we apply a linear interpolation of possible NaN values thanks to the current and one previous second data from rawCalibrationData (or just the current data if we are at the first second of recording). However, in spite of the linear interpolation, this is again possible to have NaN values. In fact, if NaN values are at the beginning of the first 1s-packet or at the end of the final 1s-packet of the recording, we are not able to interpolate them. But if we have NaN values at the beginning or the end of 1s-packet during the recording (not the first or the final packet), we can interpolate NaN values because each time, we have the previous 1s-packet in rawCalibrationData.

Then, we check if we have NaN values inside the 1s-packet.
→If yes, add 250 data points with NaN values in an array ppCalibrationData which is the array (locally on the smartphone) which contains no-preprocessed data and will be used after to do other processing and compute relaxation index. In addition to that, we add 0 in qualityCalibration which is an array (locally on the

smartphone) holding one quality value by second and will be send to the server in the same Json file than data.

→If no NaN values are inside the 1s-packet, we do the following 3 steps:

- We substract the mean amplitude of the segment from each sample in order to remove the DC offset of the segment.
- We filter the powerline noise (50 and 100Hz) with a notch filter.
- We compute quality with the quality Checker and put it on qualityCalibration array. Then, we fill in ppCalibrationData with the 1s-packet. If the quality is bad (=0), we replace the 250 datapoints in the ppCalibrationData by NaN values.

2. *Processes at the end of the calibration:*

At the end of calibration, we have 2 EEG signals of 250xY with Y the duration of the calibration in seconds.

a) Compute the thresholds for the outliers

For each of the **whole EEG signal from the calibration (ppCalibrationData) without the NaN parts (bad data)**:

- We substract the mean amplitude of all calibration data from each sample in order to remove the DC offset of the calibration data.
- We filter the powerline noise (50 and 100Hz) with a notch filter.
- We apply a bandpass between 2 and 30Hz to remove artifacts of low frequencies and to focus on this range of frequencies for the analysis.
- We set the thresholds for the outliers by applying an interquartile (IQR) formula, as a measure of finding where the bulk of the values lie. The thresholds are determined by 1.5xIQR. We keep these thresholds locally on the smartphone until the end of the neurofeedback session.

b) Preprocessed calibration data

For each second, we preprocess a 4-second packet of EEG signal from calibration (ppCalibrationData) (sliding window) (see figure 1):

- We remove NaN values from the 4s-packet.
- We substract the mean amplitude of the 4-second packet of EEG data from each sample in order to remove the DC offset of the 4-second packet.
- We filter the powerline noise (50 and 100Hz) with a notch filter.
- We apply a bandpass between 2 and 30Hz to remove artifacts of low frequencies and to focus on this range of frequencies for the analysis.
- We detect outliers (below and above 1.5xIQR) based on the whole calibration data and apply a linear interpolation on it.

c) Get the best channel in terms of quality and compute smoothed SNR

We get the whole EEG signal for each channel. Each whole EEG signal was preprocessed **4s by 4s with an overlap of 3s (sliding window)**. We apply a function named MBT_ComputeCalibration with the preprocessed signals in order to get the best channel in terms of quality and to recover the smoothed SNR. Each second we have a 1 smoothed SNR computed with a 4-seconds EEG packet from the best channel during the calibration.

If MBT_ComputeCalibration is called without input, we set an index of -1 for the best channel and Inf values for SNR from calibration. An error message is also displayed on the console: "ERROR: MBT_COMPUTECALIBRATION CANNOT PROCESS WITHOUT GOOD INPUTS".

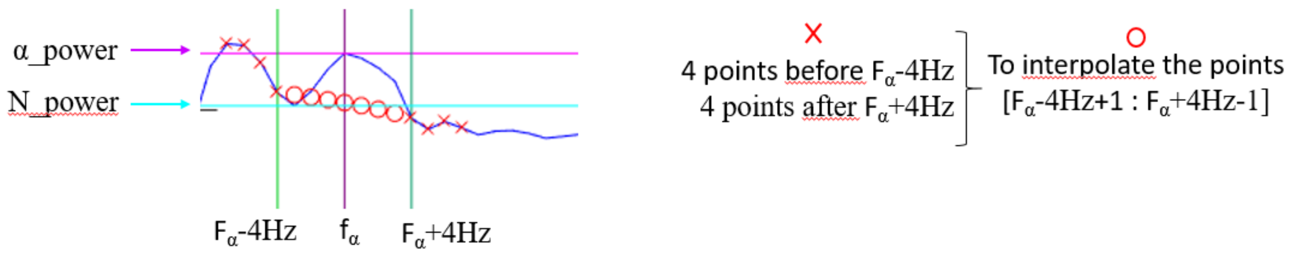
<u>How MBT_ComputeCalibration processes?</u>

➔ *To find the best channel:*

For each channel, we **keep only the 1-second EEG packets data which have a quality higher or equal to 0.5**. Then, for each channel we compute the averaged quality over the kept packets. The best channel is the channel which has the higher averaged quality. If they have both the same averaged quality, we consider the left channel as the best channel. If this best quality is lower than 0.5, we consider that we have too bad EEG signal in calibration. Having quality lower than 0.5 is possible because this is initialized to 0. The best channel is kept in memory of the smartphone until the end of the neurofeedback session.
In this case, we set an index of -2 for the best channel and Inf values for SNR from calibration. An error message is also displayed on the console: "ERROR: MBT_COMPUTECALIBRATION CANNOT PROCESS WITH ONLY BAD QUALITY SIGNALS".

➔ *To compute relaxation index as SNR*

Each second, for each **4-seconds packet** (from those which were kept in the previous step), we apply MBT_ComputeSNR (**sliding window**). This function does a spectral normalization with a differenciator function to remove the 1/f trend ("*Distinguishing low frequency oscillations within the 1/f spectral behavior of electromagnetic brain signals*", Demanuele et al., 2007). Then, it finds the maximale value of the power (ɑ_power) between 7 and 13Hz. It estimates the power of the noise (N_power) at the corresponding frequency by a linear interpolation:


α_power
N_power
F_α-4Hz
f_α
F_α+4Hz
×
4 points before F_α-4Hz
4 points after F_α+4Hz
○
To interpolate the points
[F_α-4Hz+1 : F_α+4Hz-1]

Finally, SNR = $\frac{\text{ɑ\_power}}{\text{N\_power}}$. If SNR is lower than 1, we set SNR to 1.

If MBT_ComputeSNR is called without input, SNR value is set to Inf and an error message is displayed on the console: "ERROR: MBT_COMPUTESNR CANNOT PROCESS WITHOUT SIGNALS IN INPUT".

➔ *To smooth relaxation index (SNR)*

In order to translate SNR into volum during the session, we need to smooth the SNR values because SNR values change very fastly and this is not pleasant to hear. To rescale the SNR values during the session we need to normalize them with the mean and the standard deviation of the smoothed SNR values from the calibration which will be locally saved in the smartphone until the end of the neurofeedback session. That is why we need to smooth SNR values from the calibration. To do that, for each SNR value computed from the calibration (**second by second on 4-seconds signals each time**), we apply a function named MBT_SmoothRelaxIndex which averages the SNR value with the previous values:

- At the first packet of the calibration, we have not previous value, so we do not average it. *If the first smoothed SNR value is equal to NaN an error message is displayed on the console: "Four consecutive EEG packet of 1s have bad quality.". We say four consecutive EEG packet because we compute SNR on a signal of 4s. For having NaN SNR, each EEG data of 1s is also a NaN value.*
- At the second packet, we have one previous value, so we average the SNR value with the previous one. *If the second smoothed SNR value is equal to NaN, it means that the current SNR value (4s) and the*

previous SNR value (4s but 3s in common) are NaN. So in this case we have 5 consecutive NaN values and an error message is displayed on the console: "Five consecutive EEG packet of 1s have bad quality.".

- At the third packet, we have two previous values, so we average the SNR value with the two previous values. *If the third smoothed SNR value is equal to NaN, it means that the current SNR value (4s) and the 2 previous SNR values (4s with 3s in common with the current studied packet and 4s with 2s in common with the current studied packet) are NaN. So in this case we have 6 consecutive NaN values and an error message is displayed on the console: "Six consecutive EEG packet of 1s have bad quality.".*
- For the fourth packet and after, we have at least three previous values, so we average the SNR value with the three previous values. *If the smoothed SNR value is equal to NaN, it means that the current SNR value (4s) and the 3 previous SNR value are NaN. So in this case we have at least 7 consecutive NaN values and an error message is displayed on the console: "At least seven consecutive EEG packet of 1s have bad quality.".*

If we have NaN or Inf values, we do not average SNR values with these values.
For example, if we have [1 1.5 1 2.2 2 1.8 1.4 NaN Inf 2.1] for previous SNR values and 1.8 for the current SNR value, the smoothed SNR value = (1.8 + 2.1) /2 = 1.95.

If MBT_SmoothRelaxIndex is called without input, smoothed SNR value is set to Inf and an error message is displayed on the console: "ERROR: MBT_SMOOTHRELAXINDEX CANNOT PROCESS WITHOUT GOOD INPUT".

If the current SNR value and the previous SNR values are equal to Inf, smoothed SNR value is set to Inf and an error message is displayed on the console: "ERROR: MBT_SMOOTHRELAXINDEX CANNOT PROCESS WITHOUT GOOD PASTRELAXINDEX IN INPUT".
For example, if we have [1 1.5 1 2.2 2 1.8 1.4 Inf Inf Inf] for previous SNR values and Inf for the current SNR value, the smoothed SNR value = Inf.
If we have [1 1.5 1 2.2 2 1.8 1.4 Inf 1.8 2.1] for previous SNR values and Inf for the current SNR value, the smoothed SNR value = (1.8 + 2.1) /2 = 1.95.

If the current SNR value and the previous SNR values are equal to NaN, smoothed SNR value is set to NaN.
For example, if we have [1 1.5 1 2.2 2 1.8 1.4 NaN NaN NaN] for previous SNR values and NaN for the current SNR value, the smoothed SNR value = NaN.
If we have [1 1.5 1 2.2 2 1.8 1.4 NaN 1.8 2.1] for previous SNR values and NaN for the current SNR value, the smoothed SNR value = (1.8 + 2.1) /2 = 1.95.

3. Processes online during the session:

We need a vector of float (pastRelaxIndex) to hold the relaxation indexes of the session.
During session, we receive 2 EEG signals of 250 datapoints. These data are sent (locally on the smartphone) in an array named initSessionData which is the array which will be send to the server as a JsonFile at the end of session. These data are also sent (locally on the smartphone) to another array named rawSessionData which contains all the data already recorded during the session but with NaN values linearly interpolated. Like during calibration, for each new 1-second packet, if we detect NaN values, we apply a linear interpolation of possible NaN values (which are possible Bluetooth missing values) thanks to the current and one previous second data from rawSessionData (or just the current data if we are at the first second of the session). However, in spite of the linear interpolation, this is again possible to have NaN values.

Then, like during calibration, we check if we have NaN values inside the 1s-packet.
→If yes, add 250 data points with NaN values in an array ppSessionData which is the array (locally on the smartphone) which contains no-preprocessed data and will be used after to do other processings and compute relaxation index. In addition to that, we add 0 in qualitysession which is an array (locally on the smartphone) holding one quality value by second and will be send to the server in the same Json file than data.
→If no NaN values are inside the 1s-packet, we compute the quality of the 1s-packet:

a) Compute the quality of the signals

- We substract the mean amplitude of each second data from each sample in order to remove the DC offset of the 1-second EEG data.
- We filter the powerline noise (50 and 100Hz) by a notch filter.
- We compute quality with the quality Checker and put it on qualitySesison array. Fill in ppSessionData with the 1s-packet. If the quality is bad (=0), we replace the 250 datapoints in the ppSessionData by NaN values.

These signals inside ppSessionDatawith NaN values will be used to compute some statistics about the relaxation state at the end of the session (see WorkSummary_2017_02_21_KAP).

To compute the relaxation index, we will work 4s by 4s with a sliding window of 1s like during for the calibration.
For each second, we preprocess a 4-second packet of EEG signal from session (ppSessionData) (sliding window):

b) Preprocessed session data

- We remove NaN values from the 4s-packet.
- We sbstract the mean amplitude of each 4-second packet data from each sample in order to remove the DC offset of the 4-second EEG packet.
- We filter the powerline noise (50 and 100Hz) with a notch filter.
- We apply a bandpass between 2 and 30Hz to remove artifacts of low frequencies and to focus on this range of frequencies for the analysis.
- We detect outliers (below and above 1.5xIQR) based on the whole calibration data and apply a linear interpolation on it.

c) Get the SNR value of the session data

We apply a function named MBT_ComputeRelaxIndex with the preprocessed signal of the session and the index of the best channel. This function allows to get the SNR value of the best channel (defined in the calibration) thanks to the call of MBT_ComputeSNR on the EEG values from the best channel. The SNR value is computed like in the calibration. So we have two EEG signals in input but only one SNR value (the one from the best channel). If all the EEG values from the session packet is equal to NaN (because of the quality checker which set the EEG values to NaN when the quality is bad), the SNR value is set to NaN.

If MBT_ComputeRelaxIndex is called without EEG packet in input or an index of -1 or -2 for the best channel, the SNR value from the session is set to Inf and an error message is displayed on the console: "ERROR: MBT_COMPUTERELAXINDEX CANNOT PROCESS WITHOUT GOOD INPUTS".

d) Smooth the SNR value of the session data

We push each new computed SNR value in pastRelaxIndex.
The, we apply the function called MBT_SmoothRelaxIndex to smooth the last SNR value with the SNR values of the past. To smooth the SNR value, we do the same process as it is described for the calibration.

See the corresponding section to see the possible error messages and values to handle a problem.

e) Normalize the SNR value of the session data

We apply MBT_NormalizeRelaxIndex to normalize the smoothed SNR of the session with the smoothed SNR values from the calibration which are different from Inf and NaN:

Normalized smoothed SNR value = $\frac{Smoothed\ SNR\ value - mean(Smoothed\ SNR\ values\ from\ calibration)}{std(Smoothed\ SNR\ values\ from\ calibration)}$

If MBT_NormalizeRelaxIndex is called without Smoothed SNR values from calibration in input, the normalized SNR value is set to Inf and an error message is displayed on the console: "ERROR: MBT_NORMALIZERELAXINDEX CANNOT PROCESS WITHOUT GOOD INPUTS".

f) Convert the smoothed normalized SNR value of the session data into volum

We apply the function named MBT_RelaxIndexToVolum to transform the smoothed normalized SNR into volum. This function applies the hyperbolic tangent to put the SNR value in a range of [-1:1]. Then, we add +1 in order to have SNR value in a range of [0:2]. Then, we divide by 2 for having a SNR value in a range of [0:1]. Because we want no volum when the person is relaxed, we apply this formula: volum = 1 – SNR value.

If SNR value = Inf, we set the volum to -1 and we have an error message which is displayed on the console: "ERROR: MBT_RELAXINDEXTOVOLUM CANNOT PROCESS WITHOUT SMOOTHEDRELAXINDEX IN INPUT".

If SNR value = NaN, we set the volum to 0.5. This can appear when for example, we have at least 4 consecutive packets with bad signal. Indeed, the quality checker set to NaN values when the quality is bad.
When SNR value = NaN, a message is displayed on the console: "At least four consecutive EEG packets have bad quality."

See SchemaComputeSNR_v6.0_2017_02_27_FAG.pdf to see the detail schemas.