Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Hilbert-Huangova transformace pro detekci evokovaných potenciálů

Ing. Jindřich Ciniburk

disertační práce
k získání akademického titulu doktor
v oboru Inženýrská informatika

Školitel: Prof. Ing. Václav Matoušek, CSc.

Katedra informatiky a výpočetní techniky

Plzeň 2011

University of West Bohemia
Faculty of Applied Sciences

Hilbert-Huang Transform for ERP detection

Ing. Jindřich Ciniburk

dissertation thesis
submitted in partial fulfillment of the requirements
for a degree of Doctor of Philosophy
in Computer Science and Engineering

Supervisor: Prof. Ing. Václav Matoušek, CSc.

Department of Computer Science and Engineering

Pilsen 2011

Declaration

I present a dissertation thesis to be reviewed and defended. The thesis was created at the end of doctoral study on Faculty of Applied sciences, University of West Bohemia. I hereby declare that I have created this work alone if not noted otherwise and that all used references are properly cited in a list at the end of this thesis.

In Pilsen, $29^{th} August, 2011$ .....

Ing. Jindřich Ciniburk

Abstrakt

Při vyhodnocování ERP experimentů je naprosto nezbytné přesně určit amplitudu a latenci jednotlivých ERP komponent. Protože je EEG signál kvazistacionární, je nezbytné pro jeho analýzu použít časově-frekvenční metody, jako je waveletová transformace, krátkodobá Fourierova transformace, nebo matching pursuit. Dalším zástupcem metod časově-frekvenčně analýzy je Hilbert-Huangova transformace, která byla navržena přímo pro zpracování nestacionárních signálů.

V mé práci jsem navrhl několik modifikací Hilbert-Huangovy transformace, které umožní omezit tzv. overshoot efekt, který vzniká v průběhu vytváření obálek. S navrženými vylepšeními jsou přídavné extrémy lépe umístěny, tím je zajištěna vyšší rychlost rozkladu na intrinsic mode funkce a získané intrinsic mode funkce více odpovídají původnímu EEG signálu.

Abstract

While evaluating ERP experiments, it is essential to determine the amplitude and latency of ERP components. Time-frequency domain methods, such as the wavelet transform, short-time discrete Fourier transform, matching pursuit, are usually used for this task, because the EEG signal is quasi-stationary. The Hilbert-Huang transform was designed to process non-stationary signals. Therefore, it should be suitable for processing EEG signals as well.

I have designed several modifications of the Hilbert-Huang transform, which restrain the over/undershoot effect occuring when envelopes are being calculated. My modifications contribute to better estimation of additional extrema and improve the results acquired from processing the EEG signal (even when it is contamined with artifacts). They make the empirical mode decomposition faster and the decomposed IMFs corresponds more with the original EEG signal.

Contents

List of Figures

Acronyms

CCT	Cauchy convergence test.
CWT	continuous wavelet transform.
DWT	discrete wavelet transform.
EEG	electroencephalography.
EKG	electrocardiography.
EMD	empirical mode decomposition.
ERP	event-related potential.
FFT	Fast-Fourier transform.
HAS	Hilbert spectral analysis.
HHT	Hilbert-Huang transform.
HT	Hilbert transform.
IMF	intrinsic mode function.
MP	matching pursuit.
SC	stopage criteria.
SD	standard deviation.
WT	wavelet transform.

Chapter 1

Introduction

The electroencephalography (EEG) still has its place as a diagnostic tool in today's world. It has several advantages over newer methods such as the computerized tomography (CT), magnetic resonance imaging (MRI), functional MRI (fMRI), positron emission tomography (PET). The greatest advantages are excellent temporal resolution (sampling frequency in kHz), low cost of examinations, and very low cost of the equipment. The other advantage is portability; today's EEG devices are small enough to be used as holters. These advantages lead to acquiring long time records (often longer than 24 hours). But there is the greatest disadvantage: the only way to analyze is to conduct a visual analysis of the raw EEG recording. This is still the state of art of the clinical electroencephalography – there hasn't been any significant progress in last 75 years [7]. The situation of the processing and analyzing event-related potentials (ERPs) is similar.

Event Related Potentials (ERPs) play the essential role in the Brain-Computer Interface, in the medicine and attention experiments. We cooperate with the University Hospital in Pilsen, Skoda Auto Inc., and Faculty of Transportation Science of Czech Technical University in Prague on assessment of the level of drivers' attention. Our research group at the Department of Computer Sciences and Engineering, University of West Bohemia is responsible for technical and scientific issues, e.g. EEG/ERP laboratory operation, development of advanced software tools for EEG/ERP research, or analysis and proposal of signal processing methods.

To detect an ERP component means to locate the wave and determine its amplitude and latency. There are suitable time-frequency domain methods to achieve this task, such as Wavelet transform and Matching pursuit. Matching pursuit and Wavelet transform are using predefined functions (wavelets, Gabor atoms) in which the EEG signal is decomposed to. Another alternative is represented by Hilbert Huang transform, which decomposes the signal into Intrinsic Mode Functions (IMFs) defined by the signal itself. The Hilbert Huang transform was specially designed for processing non-stationary signals such as EEG.

The aim of my PhD thesis is to use the Hilbert Huang transform for EEG signal processing, especially for the ERP detection. Hilbert Huang transform is a relatively new method proposed by Huang in [10]. It seems to be very promising for the EEG signal processing because it could calculate instantaneous frequency and amplitude - tasks essential for the ERP detection.

During tests of the Hilbert Huang transform on the data acquired in our laboratory, I have encountered several drawbacks of the Empirical Mode Decomposition method (first part of Hilbert Huang Transform). Therefore, I had to design modifications which adapt the Hilbert Huang transform for EEG signal processing and ERP detection.

Chapter 2

Aims of the PhD Thesis

The PhD thesis is focused on the EEG signal processing with time-frequency methods, especially on using Hilbert Huang transform to accomplish this task.

Following objectives were defined for my PhD thesis:

1. To implement and test HHT for EEG signal processing and ERP detection.
2. To propose necessary modifications of HHT in EEG/ERP analysis (estimation of first and last extrema point, stopping conditions, etc.)
3. To design and implement HHT algorithm improved according to 2.
4. To validate the proposed method in multiple experiments including tests on real world data (acquired in our laboratory).

Chapter 3

Introduction into EEG

3.1 Origin of the EEG Signal

EEG is an abbreviation of the Electroencephalogram and it is a result of the neurophysiologic measurement of an electrical activity emitted by the brain. This method is called the Electroencephalography. The EEG signal is a time variation of potential difference between two electrodes placed on the patient's scalp surface.

The EEG signal is created by weighted summation of signals produced by huge amount of single neurons, located in parts of the brain: cortex and thalamus. The intensity of the electric activity of neuron groups depends on the distance between the electrode and neurons. Neurons located in greater distance from the electrode contribute less to the resulting electric activity than neurons which are closer to the electrode. There is no way to separate contribution of the single neuron.

3.2 Measurement of the EEG Signal

Nowadays is the EEG recorded as a discrete signal and stored on optical discs, flash cards, hard disks and in databases. The capacity of storage media enables to keep recordings from every patient's examination.

Each modern encephalographic recording system consists of [37]

- electrodes,

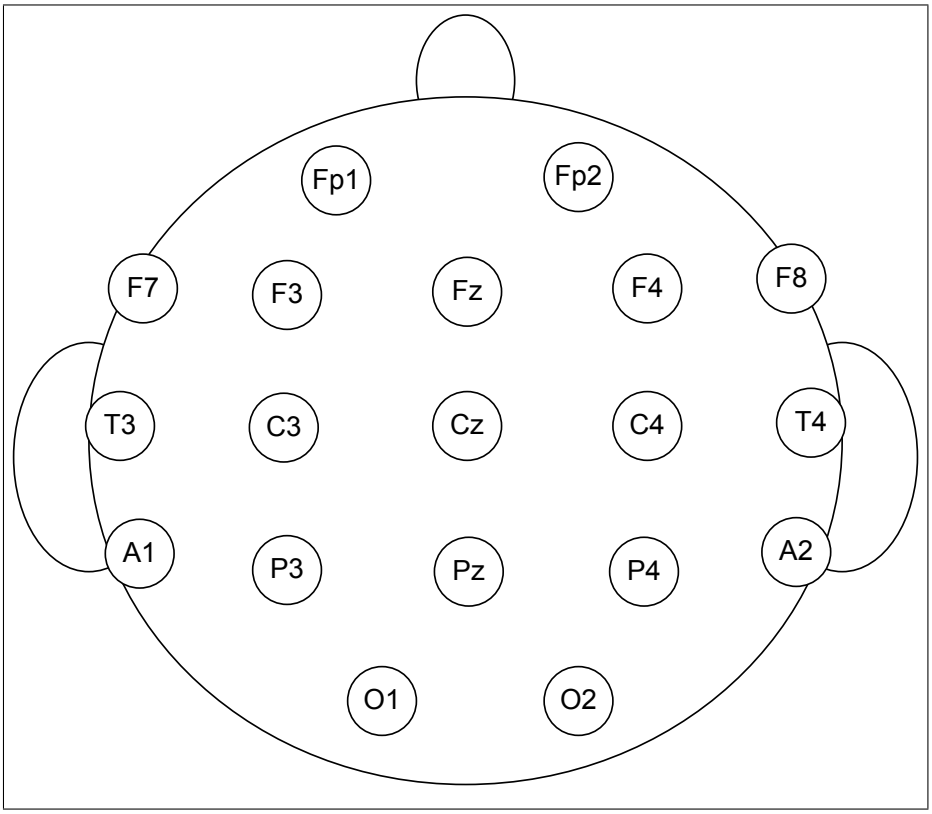

Figure 3.1: 10-20 electrode placement system

- amplifiers with filters,
- an A/D converter
- and a recording device.

Electrodes are attached to the patient's scalp and read the signal from the surface. To decrease the impedance between the patient's scalp and the electrode, a conductive gel is applied. When the impedance is lower, the EEG signal is more suitable for the processing. The usual layout of electrodes for the standard EEG examination was established in 195. It is called 10-20 electrode placement system [37] (see figure 3.1. However the count of the used electrodes could be greater or lower (e.g. five or even three electrodes are sufficient for some event related potentials experiments).

The EEG signal acquired from the patient's scalp is too weak (μV) to activate the differential amplifier. It has to be pre-amplified before the amplification or processing (A/D converter). There are pre-amplifiers (usually the Common-Emitter amplifiers (see more int [36])) designed special for this task.

Before the analogue signal is converted into the discrete signal, the signal is amplified using the difference amplifier. It amplifies the difference between two electrodes. Therefore three kinds of electrodes are usually used: the grounding, reference electrode and active electrodes. The difference amplifiers reduce the outside interference and artifacts because they amplifies just the potential difference between two electrode distorted by the same artifact.

To store the signal on the computer, it is necessary to convert it to the discrete form. The A/D converter does this task. The signal is sampled repeatedly with the fixed time interval and each sample is converted into the digital form. a sufficient sampling frequency (at least two times higher than the highest measured frequency) is the requirement for a suitable A/D converter. So is the resolution - the smallest amplitude which could be sampled (it is recommended to be at least $0.5\mu V$) [37].

3.3 Brain Rhythms

Brain waves have been categorized into five main groups according to their frequency range:

- α The frequency of alpha waves lies within the range 8-13Hz. The amplitude is higher over the occipital areas and normally is less than $50\mu V$. Best seen is with eyes closed and under condition of physical relaxation and relaxed awareness without any attention or concentration [23; 32].

- β The frequency of beta waves lies within the range 14-26Hz (in some literature no upper bound given). Higher frequencies usually called fast beta and corresponds with gamma range. Amplitude is normally up to $30\mu V$. Beta can be found, when the patient is actively thinking or solving concrete problem [32].

γ The usual frequency range is above 30Hz, usually up to 45Hz (sometimes called fast beta). Its occurrence is very rare, it is used for diagnoses of certain brain diseases [32].

δ Delta waves are very slow 0.5-4Hz. Delta waves are associated with the deep sleep, but they could be rarely present in the waking state. They could be easily confused with artifact (caused by neck and jaw muscles). [23].

θ Waves with frequencies between 4-7.5Hz are called theta waves. Theta waves play an important role in the infancy and childhood. Normal adults have their theta activity only during drowsiness and sleep [23; 32].

3.4 Interference

The acquired signal conists of signals originating in the neural activity, and signals that originate from other sources than the neural activity. These recorded non-cerebral signals are termed as artifacts. Artifacts could be divided into two categories:

- physiologic
- extraphysiologic

. Physiological artifacts are generated by the patient's body but not by the brain. Extraphysiologic artifacts come from the equipment and the environment (outside of the patient's body)[3].

3.4.1 Physiologic Artifacts

Muscle Activity

Muscle artifacts are most common artifacts. Most often they are caused by clenching of jaw muscles (figure 3.2). Duration of the muscle artifacts is shorter than the duration of potentials generated in the brain. The frequency of muscle artifacts is in the range 50-100Hz [3].

Figure 3.2: Muscles and EKG artifact [3]

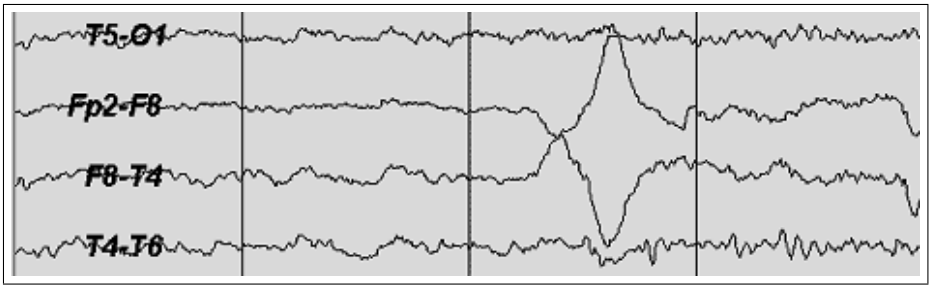

Figure 3.3: Example of eye movement artifacts [3].

Eye Movements

The eyeball acts as a dipole where the positive pole represents the cornea and the negative pole represents the retina. When the eye moves, it generates the alternate current field measurable with all electrodes near the eye (figure 3.3. The muscle artifacts, produced by muscles of the eye, influence the measured signal as well.

Glossokinetic Artifact

The tongue acts also as a dipole in the same manner as eyes. When the tip of the tongue moves, it produces broad potential field which drops from frontal to occipital areas. Frequency is variable but usually lies in the delta range [3].

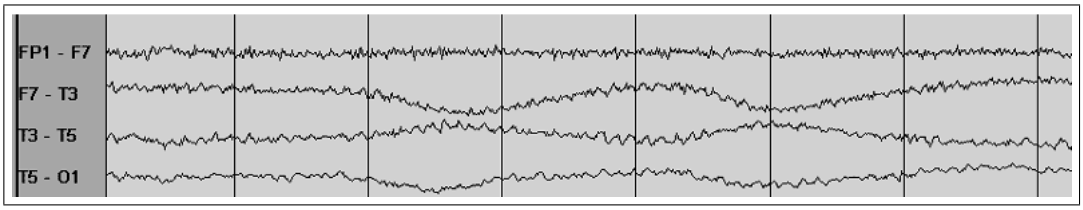

Figure 3.4: Skin artifact [3]

EKG Artifact

electrocardiography (EKG) artifacts are related to the field of the heart potentials. ECG artifacts are synchronized with the ECG tracing (figure 3.2) and could be recognized through their rhythmical repetition [3].

Pulse

Pulse artifacts are caused by the pulsating vessel when the EEG electrode is placed near such vessel. Then the EEG signal contains pulse artifact with low frequency similar to the EEG activity. This artifacts correspond the ECG artifacts (delayed 200-300ms past the ECG) [3].

Respiration Artifact

One of the respiration artifacts is slow and rhythmic. It corresponds with body movements, which affects the impedance of electrodes. The second type corresponds with the exhalation or inhalation and it is presented as slow or sharp waves. The second type affects the electrodes, on which the patient is lying [3].

Skin Artifact

Skin artifacts are caused by the biological artifacts which alter the impedance of the electrode. Sodium chloride and lactic acid from sweating react with the metal of electrodes and produce large and very slow waves ($\tilde{0}$.5Hz, figure 3.4) [3].

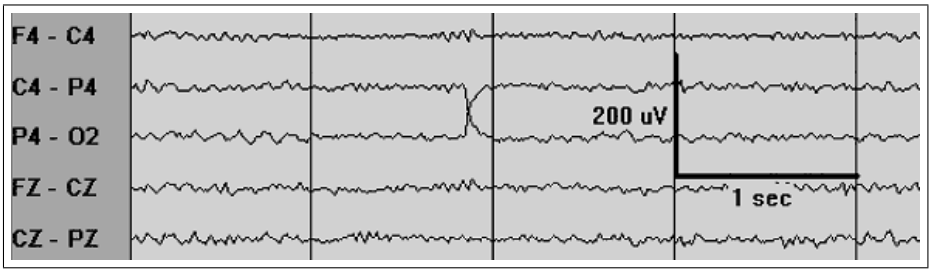

Figure 3.5: Electrode artifact. Sudden change of the impedance. [3]

3.4.2 Extraphysiologic Artifacts

Electrode Artifacts

When the impedance of the electrode changes suddenly, "pop" artifact (figure 3.5) appears. It is usually limited to the single electrode. The impedance of the electrode could change slowly. One of the causes is the drying-out of the conductive gel applied on the electrode [3].

Power grid 50Hz(60Hz)

This artifact is induced from external power sources like the power grid (50Hz Europe, 60Hz North America) see figure 5.2. Similar problems are caused by switched-mode power supplies. They spread interference with higher frequencies across the power grid. When the impedance between grounding and active electrode is significantly higher then this artifact affects the EEG signal more intensively. But it could be easily removed by using filters and lowering the grounding impedance. The common practice is to use batteries for the EEG measuring unit. The best solution would be a room shielded with Faraday cage.

3.5 Properties of the EEG Signal

If we calculate statistics of the EEG signal in different time points, we realize that these statistics vary significantly. If the statistics were nearly equal in the whole signal, the signal would be deemed as stationary. Therefore, the EEG signal is

deemed as non-stationary because its statistic differs. This is caused with the nature of the signal as it consits of different brain rhythms, artifacts, etc.

The EEG signal can be divided into the short intervals with different length but the same statistics within the interval. Such intervals are called segments and the process of acquiring them is called segmentation [32].

Nowadays the EEG is stored in its digital form only. Therefore one of the significant properties of the EEG signal is the sampling frequency. It represents the time resolution. For continuous EEG the sampling frequency up to 256Hz is sufficient usually. Higher sampling frequency around 1 or 2 kHz is more conventient for the ERP detection.

Chapter 4

Introduction into ERP

4.1 What is ERP?

Event-related potentials (ERPs) are EEG signals, which represent the response of the cortex to sensory, affective or cognitive events. The ERP are created as large sum of action potentials, which follow sensory or cognitive events.

The amplitude of ERPs is quite small, up to $30\mu V$(the background EEG activity has amplitudes even $100\mu V$). Therefore, it is necessary to use averaging technique to highlight them and suppress the background EEG [32; 39].

4.2 Properties of ERP Wave

Three parameters could be used to describe ERP waves (figure 4.1):

- the amplitude,
- the latency,
- the scalp distribution.

The amplitude represents the rate of neural activity as the response to the stimulus. The latency (delay after stimulus) reveals the timing of the neural activity. The scalp distribution provides us with the information which part of

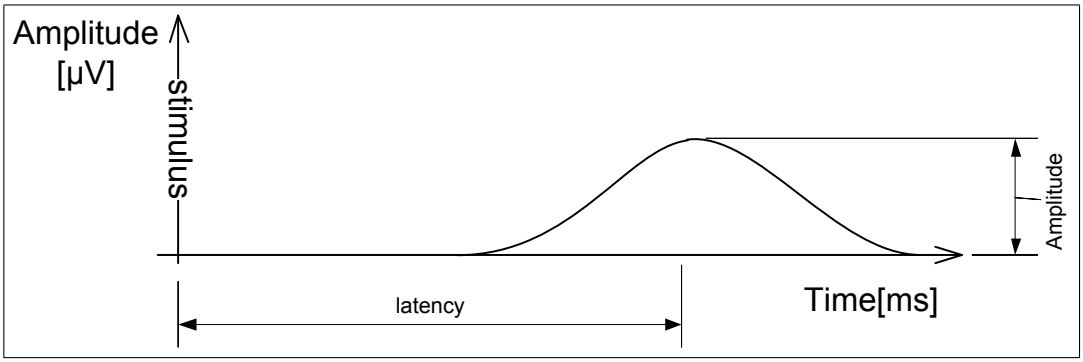

Figure 4.1: Properties of ERP wave

a brain was involved in the response (visual components are higher on the top of the head).

In the ideal case the amplitude will peak at the point of the ERP component. But the interference and averaging distort the ERP wave, so the maximum amplitude isn't suitable for describing the ERP component. The average amplitude or the area under the curve would be a better criterion. The area under the curve isn't so much affected by the distortion caused by averaging. When the ERP component is wider because of averaging, then the amplitude is lower, but the area under the curve remains almost same. The latency is affected in the same manner. When the averaging is performed on the ERP waves, the resulting waveform will be "wider" (has lower frequency) and the maximum amplitude doesn't have to be exactly in the middle of the wave (interference, artifacts). Then the better way how to establish latency, would be to divide the area below the curve (ERP component) into the two equal parts. The latency would be measured as the distance from this point (see more in [20]).

4.3 Sorts of ERPs

In this section some of the ERP components (parts of the ERP waveform) are described briefly.

ERP components are usually labeled as P1, N2, P3. The labeling refers their polarity; P for the positive and N for the negative orientation (be aware that

the positive orientation is plotted downward in medical practice). The number represents the position within the waveform. The labeling is not linked to the nature of the underlying brain activity. Therefore the the auditory component P1 and N1 are not related with visual components P1 and N1. Some of the later components such as P3 could be modality-independent, but they could contains specific sub-components.[19].

4.3.1 Visual Sensory Response

C1

C1 is the first major visual ERP component. It isn't labeled as P or N, because its amplitude polarity can vary. Usually it is positive for stimuli in the lower visual field and negative for stimuli in the upper visual field. When it is positive it merges with P1 component usually. It starts 40-60ms after stimuli and the peak is in range 80-100ms post-stimuli. Its generation is located in primary visual cortex [19].

P1

The P1 component follows the C1 wave. P1 achieves its highest amplitude at lateral occipital electrodes. P1 onsets 60-90ms post-stimuli and its peak is located between 100-130ms. The latency of P1 varies according to the contrast of the stimulus. The P1 component is also affected by parameters of te stimulus and spatial attention (see more in [19]).

N1

After P1 wave, the N1 wave comes. The N1 component is usually divided into several sub-components. The peak of the earliest component appears after 100-150 ms at frontal electrodes. At least two posterior N1 components follow the first N1 component. They peak between 150 and 200ms after stimulus (one from parietal cortex and second form occipital cortex) [19].

P2

The N1 wave is followed by the P2 wave on frontal and central electrodes. When the stimuli contain target features and come relatively infrequently, the component is larger. In this way the P2 is similar to the P3, but the P3 can occur by far more complex target categories. The P2 wave is often overlapped with N1, N2 and P3 waves [19].

N170

The component N170 has its peak between 150 and 200ms. The N170 is later and/or larger for inverted faces used as stimuli. This effect is also observed for non-face stimuli when the subject has extensive experience of viewing these stimuli in the upright orientation. The similar effect is observed when the subject is stimulated with familiar stimuli such as words [19].

4.3.2 Auditory Sensory Response

Very Early Components

It is possible to observe a sequence of ERP peaks within the first 10 ms after auditory stimuli. These peaks are generated by various stages along the brainstem auditory pathways. Therefore these peaks are called brainstem evoked potentials (BERs) or auditory brainstem responses (ABRs). BERs are useful for testing auditory pathology, especially for infants. The BERs are followed by mid-latency components. Their latency is in range 10-50 ms. These waves arise from the medial geniculate nucleus (auditory thalamus) and the primary auditory cortex. After these waves comes the auditory P1 wave (ca. 50ms) which achieve its largest amplitudes at fronto-central electrodes [19].

N1

The auditory N1 has several sub-components like the visual N1 wave. The first fronto-central has its maximum amplitude about 75ms after stimuli and its origin is in the auditor cortex. The second wave peaks around 100 ms and the third peaks about 150ms. The latency of N1 component is affected by the attention [19].

Mismatch Negativity (MMN)

When the subject is exposed to identical stimuli and occasionally to other different stimuli called mismatching stimuli, the mismatching stimuli elicit the negative wave. This wave achieves its highest amplitude in the area of central scalp. Its latency lies between 160 and 220ms. The MMN is observed even when the subject isn't focused on the stimuli (stimuli are not task-relevant) [19].

The N2 Family

The N2 family contains several different components. Their latency corresponds to the time range of the N2. The first one could be called the basic N2 which is elicited by repetitive non-target stimuli. If the subject is exposed to other stimuli (usually called deviants), the amplitude will be larger. If the deviants are task-relevant (differs from the MMN), then the later N2 component called N2b will be observed. This component could occur at both visual and auditory stimuli [19].

The P3 Family

In the time range of P3 wave, several different ERP components could be found. The first two of them are the P3a and P3b. The P3a and P3b are elicited by the infrequent shift in the tone or intensity (deviant stimulus), but the P3b is only present when the shift is task-relevant. The P3b component is almost always meant as the P3/P300 component (I follow this trend). The amplitude of the P3(P3b) wave is larger when the target probability is lower. The amplitude is also higher when the target-stimulus comes "unexpected". The third factor is the attention of the subject. More is the subject focused on the task, the higher amplitude of the P3 wave is.

The P3 wave is generated after the stimulus has been processed and categorized according to the task (depends on the probability of the task-relevant stimulus), therefore the P3 represents the early cognitive function of the brain (the earlier waves are just sensorical) [19].

Language related ERP components

The best known language-related component is the N400. The N400 is usually elicited as the response to violations of semantic expectancies. The N400 component could be elicited by non-linguistic stimuli [19].

4.4 Simple ERP Experiment

Let me introduce the simple ERP experiment, which I have used for acquiring the testing data. This description could clarify terminology used for the ERP experiments.

The most simple ERP experiment focused on continuous performance task. Is classic odd-ball paradigm. We have performed same experiment in our laboratory. Subject is stimulated with target stimulus letter "Q" (15%) and the non-target stimuli the letter "O" (85%). The letters are presented on the screen of the computer. The delay between two successive stimuli is 1500ms. The proband's task was to count the occurrences of the "Q" letter (The proband doesn't know the count before the experiment).This experiment is designed to elicit the P3b component [13; 19].

The ongoing EEG signal was recorded with our BrainAmp device directly into another computer than the one presenting the stimuli. We have recorded signals from only five active electrodes P3, P4, Fz, Pz and Cz. Also the BrainAmp recorded marks for each stimuli. The stimuli marks are different for each group of stimuli (target and non-target). Therefore, each mark is described with the type of the stimulus and the time index of the sample when the stimulus presented. It is essential for ERP experiments to store these marks; without those marks it would not be possible to locate epochs in the EEG signal and to process them. We have been using sampling frequency of 1 kHz and the resolution 0.1 μV.

Chapter 5

ERP Detection Techniques

5.1 Signal to Noise Ratio

When we are processing the EEG signal to detect the ERP wave, we have to keep in mind that the amplitude of the strongest ERP wave (called P3) peaks up to 20μV, but amplitudes of common EEG rhythms are higher ($\alpha < 50\mu V$, $\lambda < 90\mu V$ [32]). The signal-to-noise ratio (SNR) expresses this property of the signal. In our case the SNR is 20μV/90μV, it could be expressed as 0.2 as well. Such SNR is very low for simple ERP detection, but there is a technique which could increase the SNR to more suitable level. Such level of SNR could be achieved by averaging. In order to use the averaging method, there has to be enough epochs acquired during the examination process. [19].

5.2 Averaging as Basic Method for ERP Detection

It is required to increase value of SNR. The averaging technique could accomplish such task. Short epochs extracted from the continuous EEG signal are aligned with respect to the stimulus (time locking event) and mutually averaged. The i-th sample of the calculated wave is the average of N (count of averaged epochs) samples at the i-th position. This approach is based on two premises:

- The ERP waves are assumed to be almost identical in each trial.
- The rest of the EEG signal is completely unrelated to the time-locked event (the stimuli).

When we include sufficient number of epochs in the average, the remaining noise (background EEG) will be close to the zero at every point, but the ERP wave will stay almost unchanged. The relationship between the noise R and the number of averaged trials could be expressed as $(1/\sqrt{N}) \cdot R$ (see more in [19]). There are three ways how to calculate averages:

- Stimulus-Locked (described above),
- Response-Locked
- and Time-Locked.

The stimulus-locked averaging is described above, other methods are described in the following sections.

5.2.1 Response-Locked Averages

When the latency of particular trials varies significantly (typically during experiments focused on reaction time), it is necessary to calculate the average by some other way than the stimulus-locked approach which could distort the wave (see more in section 5.2.3). In such cases it is better to use response-locked averages. In response-locked averaging, the epochs are aligned according to the response (ERP wave) at each single-trial.

5.2.2 Time-Locked Spectral Averaging

Another way how to calculate averages, is the Time-Locked Spectral averaging. For every epoch the time-frequency representation (time-frequency map) is calculated. Discrete and Continuous wavelet transforms, Short-time Fourier transform, Matching pursuit and Hilbert-Huang transform are suitable methods for the task. The epochs are aligned to the selected time (could be stimulus) and then averages of time-frequency maps of all epochs are calculated. This approach is very useful in cases when the phase shift of the ERP waves in trials could vary.

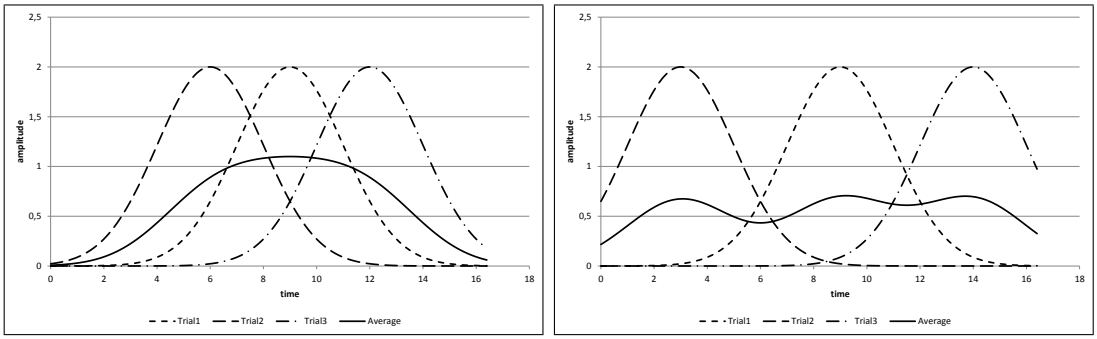

Figure 5.1: Latency variability could cause deformation of ERP wave, when the trials are averaged.

5.2.3 Latency Variability

During the averaging, we have to be very careful about the latency variability. This latency variability could lower the maximum amplitude of the ERP wave. But when the latency significantly varies, resulting averaged ERP wave could be distorted in such way that the ERP wave couldn't be recognized at all. For the illustration, see figure 5.1.

5.3 Interference and Artifacts

There are many glitches ocurring during the averaging. We have to avoid them or minimize their impact on the resulting averaged ERP wave. Such glitches could be caused by properties of the ERP wave itself, for example by the latency variability (section 5.2.3). But distortion of the averaged ERP wave could be caused also by external influences (interference) or by signals of non-EEG origin.

5.3.1 Noise From the Power Grid

Interference coming from the power grid is always present. The noise is induced into conductors connecting the electrodes. The frequency of the noise is 50Hz (Europe) and 60Hz (North America). The interference can be easily removed with a notch-filter (see figure 5.2). The filtering has to be done before the averaging,

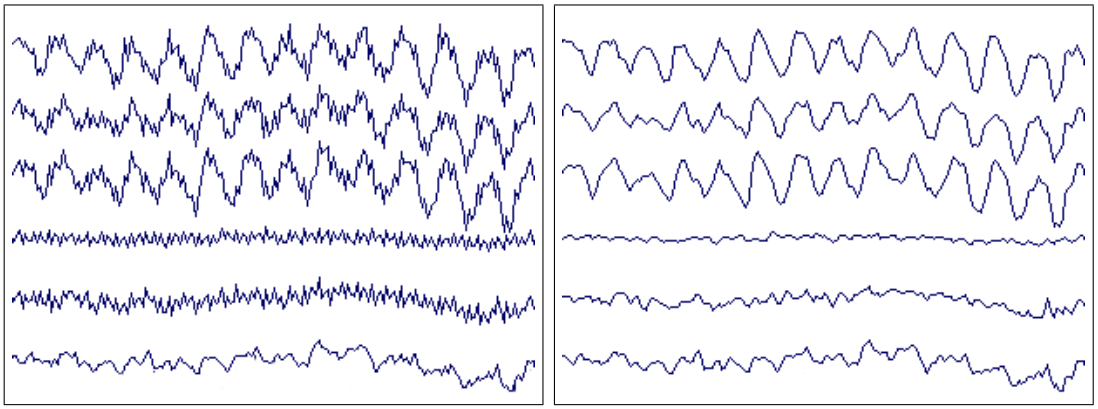

Figure 5.2: 50Hz noise in the EEG signal and the same signal after processing with a notch-filter.

because there is a significant chance, that the noise doesn't disappear during the process of averaging. Such inconvenient cases might occur, when the 50Hz sinusoid has the same phase shift in each trial. Therefore, the averaging process is not able to eleminate the interference.

5.3.2 Artifacts

Dealing with artifacts is more important task by the ERP detection than by processing of the ongoing EEG signal. Artifacts are caused usually by the muscle proband's activity (see more in section 3.4 and [3] [16] [14]). Usually the amplitude of artifacts is higher than $100\mu V$; it is no exception, that artifact could have the amplitude above 200μV. Such high amplitude drives the SNR very low, making the number of trials required to be included into the average grow rapidly. Therefore, it is the common practice to exclude trials with artifacts before the averaging. Even better solution of this problem is to anticipate the artifacts during the experiment, instruct the proband to try not to blink (as much as possible), seat him comfortably or adapt the experiment to eliminate the artifacts as much as possible. Good data (without artifacts) are irreplaceable.

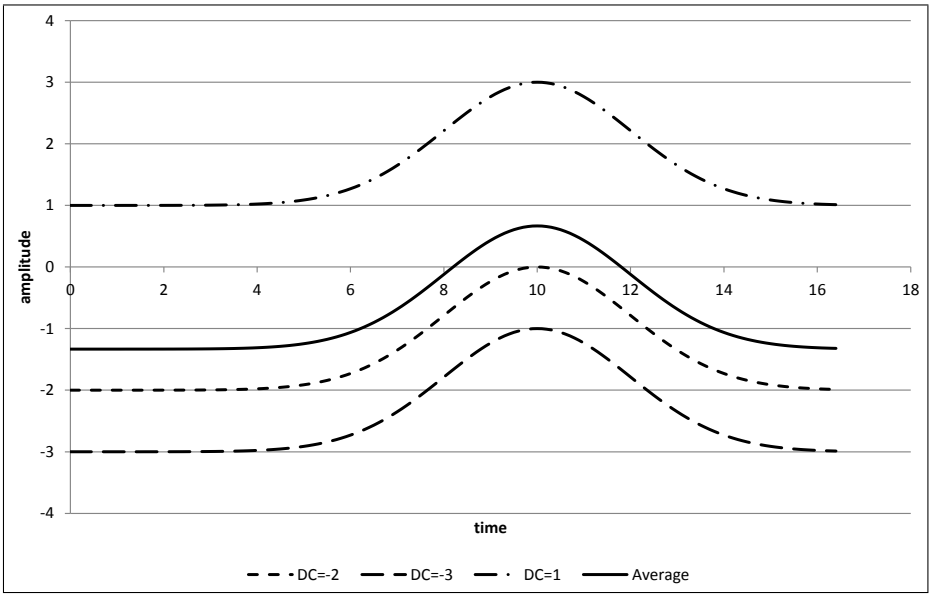

Figure 5.3: Created averaged ERP wave without baseline correction.

5.3.3 Baseline Correction

During the running ERP experiment, proband might sweat or the conductive gel on electrodes might get dry. Both circumstances cause the change of the impedance of the electrode. As a result of both phenomena, we get the different value of DC component for each trial. In the worst cases, changes of the impedance of the electrodes can demonstrate themselves as slow waves, that don't belong to components of the EEG signal.

When the averaged is calculated from trials, where the DC component changes along with trials, the resulting amplitude of the averaged ERP depends on the values of DC component of each trial, not on the amplitudes of ERP waves (see figure 5.3).

This issue could be easily solved by a simple method called base-line correction. Before including trials into the averaged ERP wave, the average value of first samples of the signal is calculated. The count of first samples can be derived from the latency of the first ERP component. If we are expecting that the first component comes after 100ms, we can use the first 100ms of the signal to calcu-

late the average. Than the average is simply subtracted from the whole signal of the trial, sample by sample. This will ensure that the DC component doesn't have the influence on the amplitude of the averaged ERP wave.

Chapter 6

Time-frequency Domain Methods for ERP detection

For the time-frequency representation, wavelet transform (WT) and matching pursuit (MP) are often used. These two methods are suitable also for the processing of the EEG signal and ERP detection. In our team we use the WT and MP for ERP detection. Therefore, I'm going to shortly introduce the MP and WT and their application. The comparison of MP, WT and HHT will be presented in chapter 11.

The content of this chapter is taken from our common paper [53].

6.1 Wavelet Transform

Wavelet Transform (WT) is a suitable method for analyzing and processing non-stationary signals such as EEG. The WT has good ability of time and frequency localization, which is necessary for ERP detection. For the EEG signal processing it, is possible to use continuous wavelet transform (CWT) or discrete wavelet transform (DWT).

6.1.1 Principles of Continuous Wavelet Transform

Let me demonstrate the principle of the CWT using the Mexican hat wavelet. The Mexican hat is defined as:

$$\Psi(\frac{t-b}{a}) = [1 - (\frac{t-b}{a})^2] \cdot e^{-\frac{1}{2}[\frac{(t-b)}{a}]^2} \tag{6.1}$$

where a (dilatation) corresponds with the frequency, and b (translation) describes shifting the wavelet over the signal (Figure 6.1, Figure 6.2). The translation parameter was set to 1 when we performed the CWT.

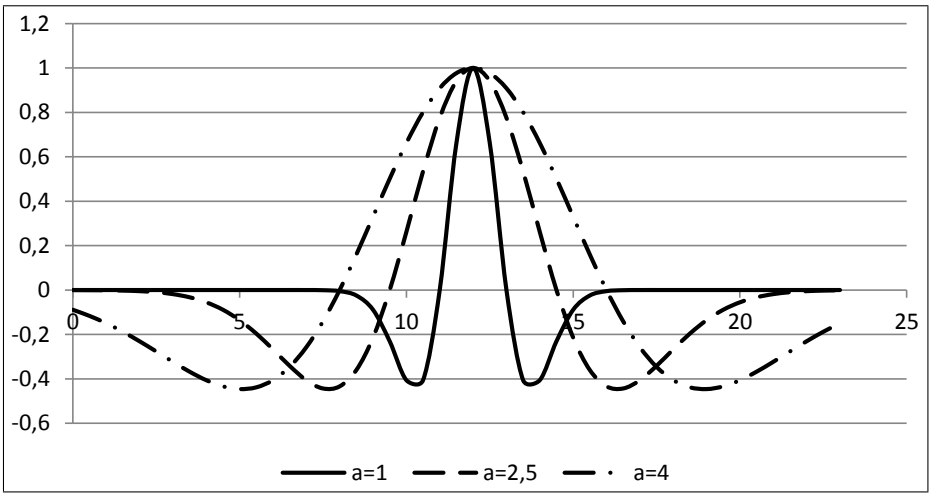

Figure 6.1: Dilatation of Mexican hat wavelet

We have used the following algorithm of CWT for the ERP detection:

1. We have chosen the mother wavelet, set the starting and ending value of dilatation; the translation step has been set to 1.

2. We have calculated the sum of correlation for the current dilatation and repeated for every translation step in order to cover the whole signal.

3. According to the chosen step, we have changed the dilatation and continued with the 2.

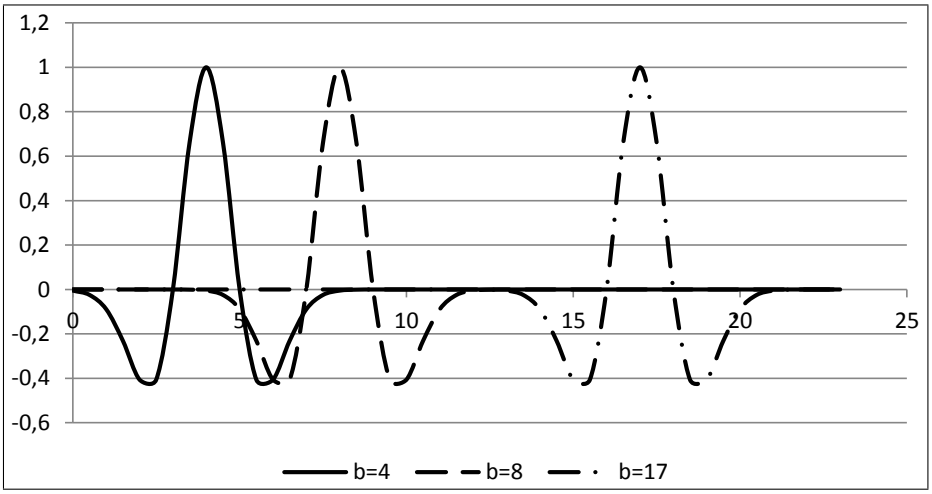

Figure 6.2: Translation of Mexican hat wavelet

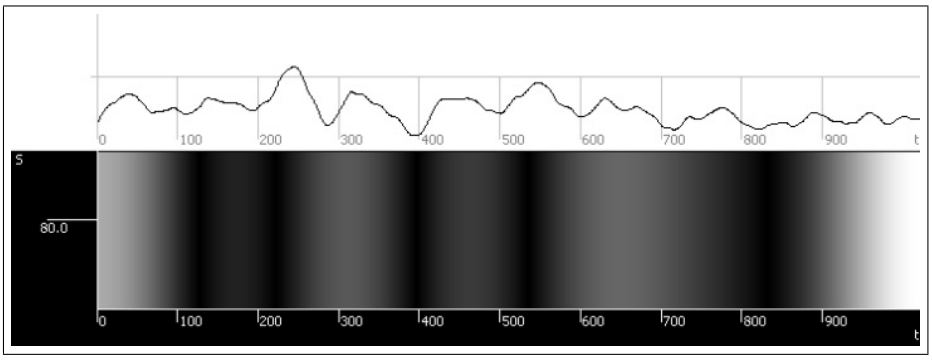

Figure 6.3: Input signal and its scalogram.

4. We stopped the run of the algorithm when the maximum dilatation value was reached.

The result of the wavelet transform is visualized in a scalogram where each coefficient represents a degree of correlation between the transformed wavelet and the signal. The scalogram is gray scaled and the highest values are white (Figure 6.3).

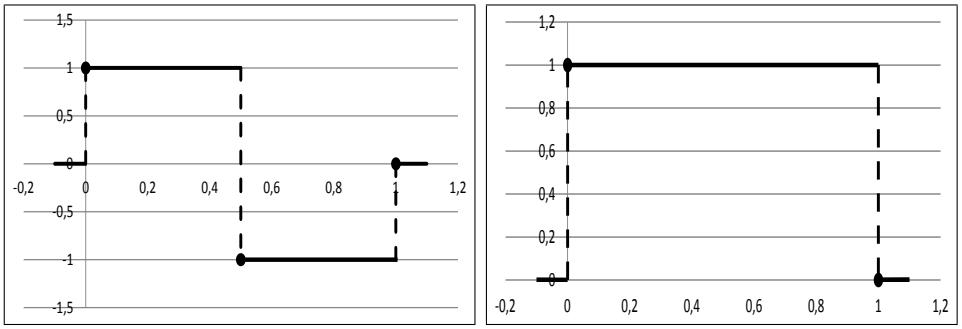

Figure 6.4: Haar wavelet (scaling function on the right, wavelet function on the left)

6.1.2 Principles of Discrete Wavelet Transform

The continuous wavelet function, known from the CWT, is replaced by two discrete signals - a wavelet function and a scaling function (see Figure 6.4 for Haar wavelet example).

Given the limited spectrum band of wavelet functions, the convolution process with this function can be interpreted as a limited band-pass filter [38]. In terms of digital signal processing, wavelet transform can be considered as a bank of filters with signal decomposition into sub-frequency bands. The slowest fundamental frequency components are detected using a scale function. Wavelet function is then documented by a high pass filter and the scale function is a complementary low pass filter. Relevant coefficients are determined taking the convolution of signal and the corresponding analyzing function [34] [27]. The scale is inversely proportional to the frequency; the low frequencies correspond to large scales and to the dilated wavelet function. Using the wavelet analysis at large scales, we obtain global information from the signal (an approximate component). At small scales we obtain detailed information (a detailed component) representing rapid changes in the signal [27].

Calculation of DWT coefficients is implemented by a gradual application of wavelet function (high frequency filter) and scale function (low frequency filter) to the given signal using Mallatov decomposer scheme [1] (see Figure 6). For each level of decomposition p so-called detailed component $D_p(n)$ of the input signal

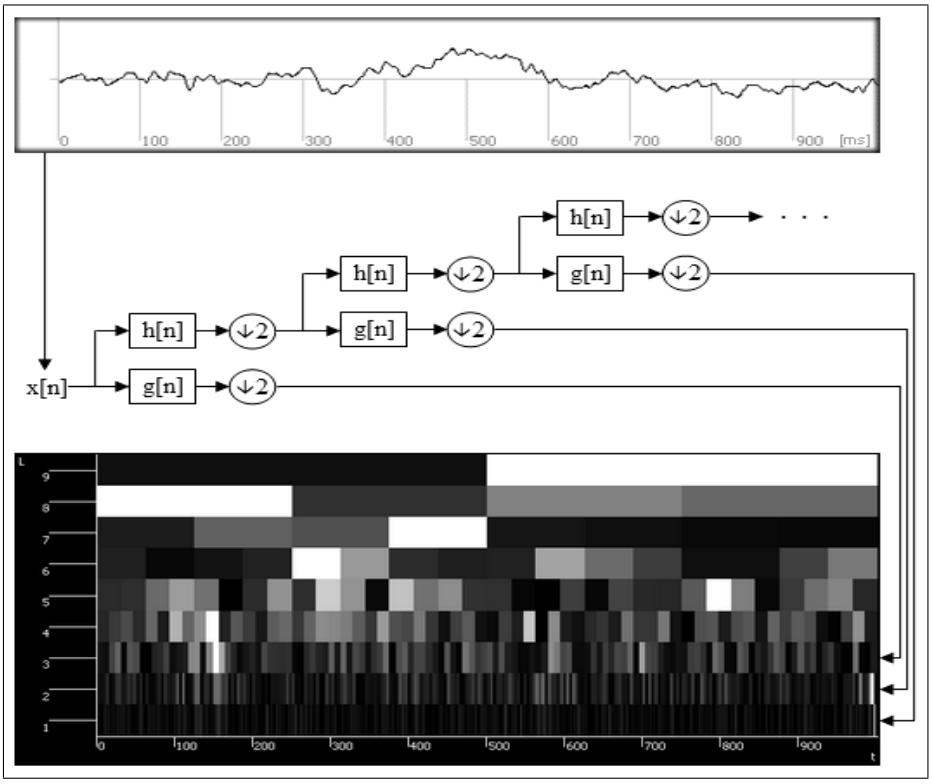

Figure 6.5: Principle of Discrete Wavelet transform [34]

is the output of high pass filter $h_d(k)$. The approximation component $A_p(n)$ is the output of low frequency filter $h_d(k)$ Using the convolution and the subsequent subsampling the following relationships are valid [27]:

6.1.3 ERP Detection with WT

During our experiments we detected the P3 component in the EEG signal (this ERP component usually follows a stimulus with delay starting at 300 ms). When we look for the P3 component, we compute the correlation between a wavelet (which is scaled to correspond to the P3 component) and the EEG signal only in the corresponding part of the signal, where the P3 component could be situated. This approach avoids a false ERP detection in the signal parts, which couldn't contain the P3 component. Wavelet coefficients are affected by the match of

Wavelet	threshold value
Mexican hat	25
Gaussian	-25
Haar	10
Daubechies6	10
Symmlet8	10

Table 6.1: Threshold values for wavelet coefficients detecting ERP component.

scaled wavelet and the signal and also by the signal amplitude. Because the degree of correlation is different for each type of wavelet we had to establish corresponding threshold values empirically (table 6.1). To determine a threshold for ERP detection we have to preprocess EEG signal rejecting the epochs with artifacts and to correct the baseline of each epoch [34]. When the degree of correlation is higher than the established threshold, the ERP component is considered to be detected. Based on the threshold values given in table 6.1, Mexican hat, Gaussian wavelet, Haar and Symmlet8 were selected for the next elaboration.

The disadvantage of CWT is its computational complexity which is linearly growing according to the number of signal samples. An increase in the number of input signal samples doesn't have so big impact if we use DWT. To detect ERP components, we use 2 kHz sampling frequency. Then the epoch, which has to be at least one second long, has 2048 samples. CWT computation on 2048 samples takes approximately 1.3 second. Therefore CWT is not suitable for BCI application [1]. We can say that the time of DWT computation on the same sample is insignificant.

6.2 Matching pursuit

The matching pursuit (MP) algorithm decomposes any signal into the sum of so-called atoms, which are selected from a dictionary. The atom that best approximates the input signal, is chosen in each iteration. This atom is subtracted from the input signal and the residue enters the next iteration of the algorithm.

Averaged Epochs	Wavelet	correctly detected		false positive detection	false negative detection
		count	[%]		
10	Mexican hat	32	80.0	6	2
	Gaussian	33	82.0	6	1
20	Mexican hat	34	85.0	5	1
	Gaussian	34	85.0	3	3
30	Mexican hat	36	90.0	3	1
	Gaussian	37	92.5	2	1

Table 6.2: P3 component detection using CWT

Averaged Epochs	Wavelet	correctly detected		false positive detection	false negative detection
		count	[%]		
10	Simmlet8	28	70.0	8	4
	Haar	25	62.5	9	6
20	Simmlet8	33	82.5	3	4
	Haar	29	72.5	6	5
30	Simmlet8	34	85.0	4	2
	Haar	31	77.5	5	4

Table 6.3: P3 component detection using DWT

The total sum of atoms selected successively in algorithm iterations is an approximation of the original signal - more iterations we do, more accurate approximation we get.

The matching pursuit algorithm is most often associated with so-called Gabor atoms dictionary. Gabor atoms are defined as the Gaussian window

$$g(t) = e^{-\pi \cdot t^2} \tag{6.2}$$

modulated using cosine function as follows

$$g = g_{(s,u,v,w)}(t) = g(\frac{t-u}{s}) \cdot \cos(vt + w) \tag{6.3}$$

Each atom is uniquely defined by the ordered quadruple (s, u, v, w), where s denotes the scale, u is the shift, v is the frequency and w denotes the phase shift.

6.2.1 Classic ERP detection with MP

The principle of matching pursuit algorithm is to decompose the input signal into individual atoms; initially the signal trend is approximated by the atoms, secondly signal details are approximated. During recordings of the brain activity ERPs appear just like the signal trends, which are disturbed by EEG signal. After several iterations the input signal is approximated by the atoms in such a way that the signal trend is highlighted [10].

According to equation 6.3 each atom is uniquely defined by four values *(s,u,v,w)*. In addition, after running of the algorithm a modulus is available for each atom. The modulus is a degree of correlation between the atom and the input signal in the iteration. The trend of the atom in time can be determined from these values. The accuracy of approximation of the original signal can be determined according to the value of the module (higher value means better approximation). ERP reflects the signal trend and the value of the modulus is high in the case of its occurrence. At the same time, the value of the shift corresponds to the location of its anticipated occurrence [25]. The idea of ERP components detection was introduced in [22].

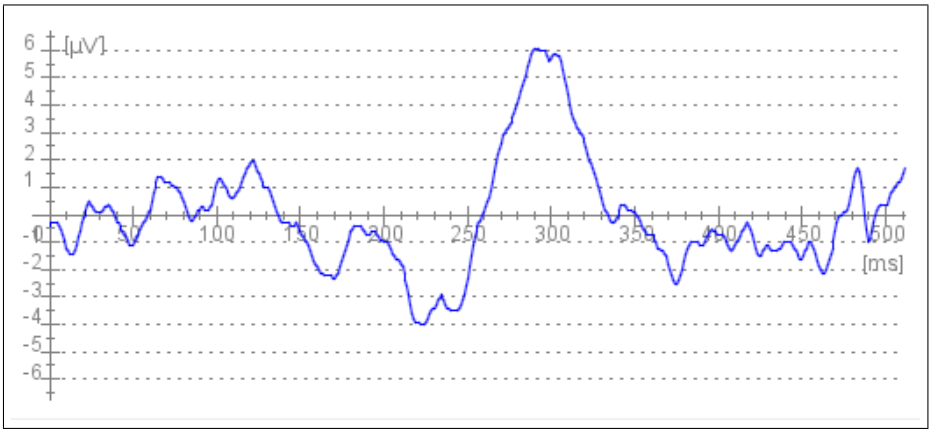

Figure 6.6: Input signal with P3 component [25]

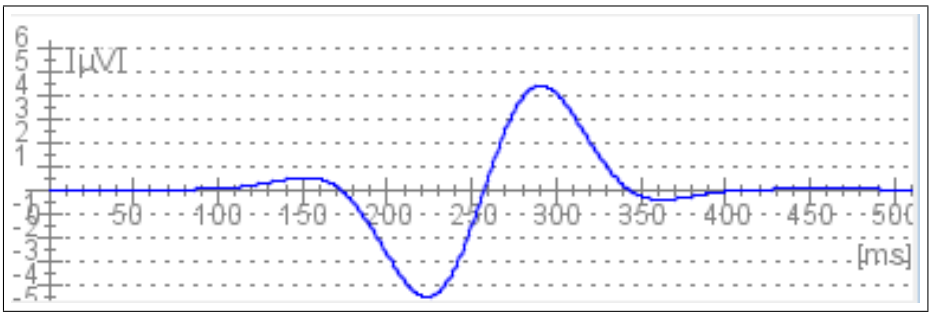

Figure 6.7: Gabor atom which best approximates P3 component [25]

An example of P3 components detection is shown in Figures 6.6, 6.7 and 6.8. Wigner-Villa's transformation (see [42] and [7]) was used to display the output of the matching pursuit algorithm. This transformation shows the energy density of the signal in time frequency spectrum.

6.2.2 Principles of Modification of ERP Detection with MP

The basic idea of the MP algorithm modification is not to base an ERP component detection on classification of feature vectors (feature vectors are parameters of

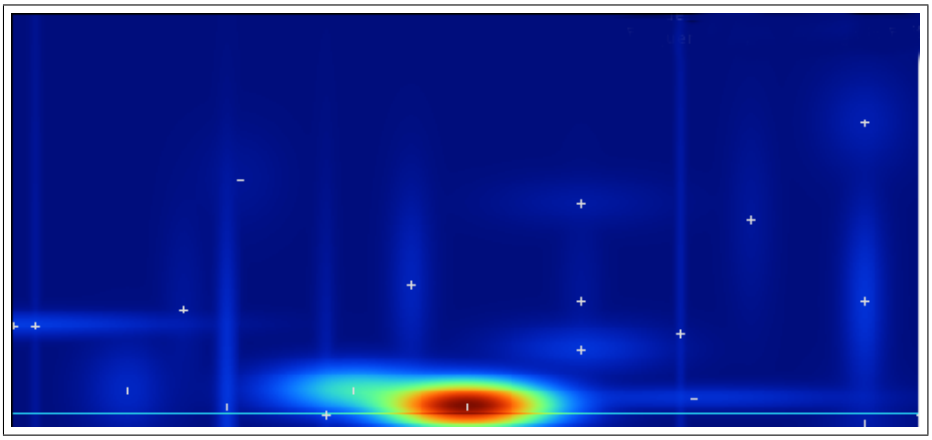

Figure 6.8: Wigner-Ville transform of MP algorithm output [25]

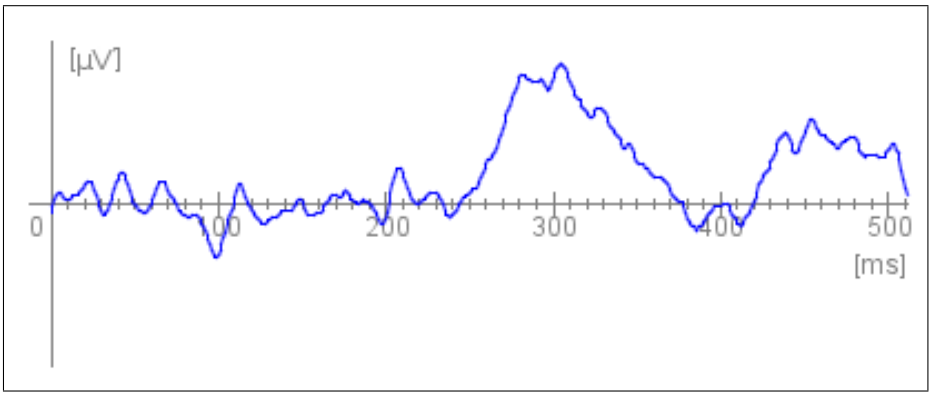

Figure 6.9: Input signal

Gabor atoms), but to use the MP algorithm as the method of input signal filtering and then to compute correlation between filtered (reconstructed) signal and an ERP component model.

First we approximate an input signal (figure 6.9) using several Gabor atoms and then we reconstruct the input signal from them. Loss of information caused by approximations is considered as filtering of the input signal (figure 6.10).

The nature of the MP algorithm is to suppress noise. Then the reconstructed signal corresponds to the trend of the input signal. This can be suitably used

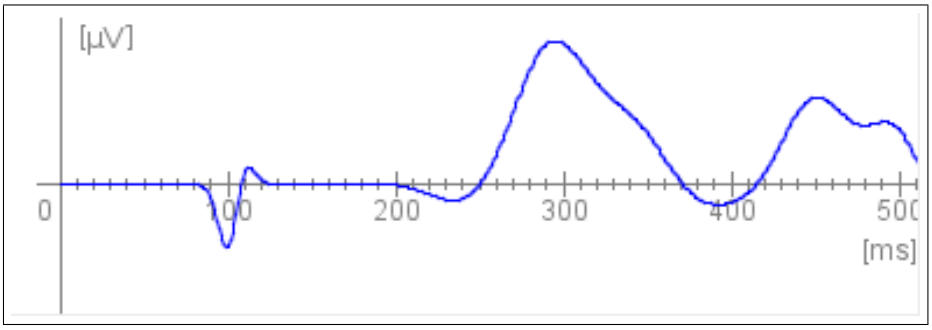

Figure 6.10: Reconstruction of input signal from five Gabor atoms

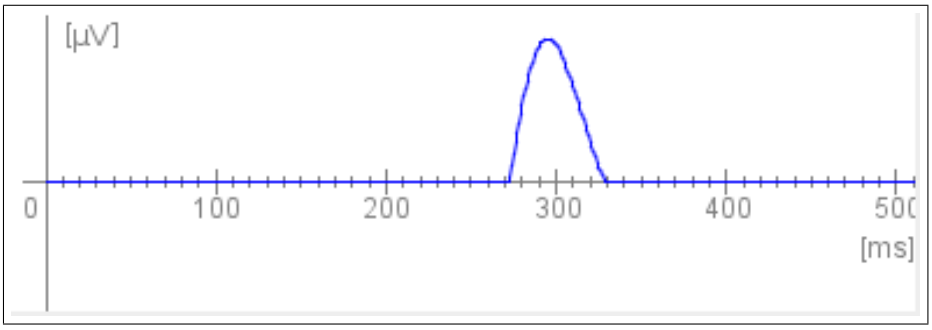

Figure 6.11: ERP component model in the corresponding location

because ERP components are (if not discarded by artifacts) the part of the signal trend. The following phase includes the detection itself when an ERP component model (figure 6.11) is used. This model is obtained e.g. by averaging a sufficient number of epochs containing raw ERP signal or by filtering of ERP component from one epoch. The ERP component model is shifted on the restored signal in the expected range of ERP component. Correlation between the ERP component model and the reconstructed signal is computed for each shift. The maximum value of the correlation and the attaching shift value are stored. After calculating all possible correlations the stored maximum value is compared to the threshold. If the maximum value is equal to or greater than the threshold, the ERP component is detected in the corresponding location.

Averaged Epochs	Matching pursuit	correctly detected		false positive detection	false negative detection
		count	[%]		
10	Modified	31	77.5	7	2
	Classic	26	65.0	5	9
20	Modified	34	85.0	3	3
	Classic	24	60.0	9	7
30	Modified	36	90.0	2	2
	Classic	31	77.5	6	3

Table 6.4: P3 component detection using MP algorithm

The results of ERP detection using modified and classic MP algorithms are available in table 6.4.

Chapter 7

Hilbert-Huang Transform

The Hilbert-Huang transform was designed to analyze nonlinear and non-stationary. The method was proposed by Huang in [11]. It consists of empirical mode decomposition (EMD) and the Hilbert spectral analysis (HAS) methods, both of these methods were introduce by Huang et al [11].

7.1 Intrinsic Mode Functions

An intrinsic mode function (IMF) is a function which has to fulfill following two conditions:

1. In the whole data set, the number of extrema and the number of zero crossings must be either equal or differ by one at most.

2. The mean value of the envelope defined by the local maxima and the local minima is zero at any point [10; 11; 18].

An IMF represents simple oscillatory mode as counterpart to a simple harmonic function, but it is much more general by its definition. The conditions which IMF fulfills, are necessary for defining instantaneous frequency.

7.2 Empirical Mode Decomposition

The goal of the empirical mode decomposition is to decompose the original data (signal) to the IMFs and the residue. The EMD is a data driven method and IMFs are derived directly from the signal itself [15].The most of the data are not IMFs. At any time the data may involve more than one oscillatory mode. That is why the simple Hilbert transform cannot provide the full description of the frequency. The process of acquiring the IMFs is called sifting and it's described below [9; 14; 17; 28]:

1. Initialize the residue to the original signal $r_0(t) = x(t)$ and IMF counter $i = 1$
2. Extract the i-th IMF:
   (a) Initialize $h_0(t) = r_{i-1}(t)$ and initialize step counter $k = 1$
   (b) Locate local maxima and minima in $h_{k-1}(t)$
   (c) Create upper envelope by connecting detected maxima with cubic spline
   (d) Create lower envelope by connecting detected minima with cubic spline
   (e) Calculate the mean $m_{k-1}(t)$ by averaging the upper and lower envelopes
   (f) Calculate $h_k(t) = h_{k-1}(t) - m_{k-1}(t)$
   (g) Check stopping criteria (see chapter 7.2.1)
      i. If stopping criteria are satisfied, then $IMF_i(t) = h_k(t)$
      ii. Else $k = k + 1$ and continue with 2b.
3. New residue is $r_i(t) = r_{i-1}(t) - IMF_i(t)$
4. Check stopping criteria of EMD
   (a) If $r_i(t)$ has at least 2 extrema then $i = i + 1$ and continue with 2.
   (b) Else the decomposition is finished and $r_i(t)$ is the residue after the decomposition.

7.2.1 Stopping Criteria

During the EMD we want to retrieve IMFs described in chapter 7.1. These functions have to fulfill two conditions. The second condition (mean of the envelopes is meant to be zero) is very difficult to fulfill. As the points 2b to 2g of the EMD (from chapter 7.2) are repeated, the mean approaches to zero. But this makes amplitude variations of the individual waves more even. When we want to achieve strictly zero mean, we can assume that the amplitudes are constant and we lose very important information of the signal. So there were proposed two stopage criteria (SC). The first one is standard deviation (SD) proposed in [5; 10; 41]:

$$SC = SD = \sum_{t=0}^{T} \frac{|h_{k-1}(t) - h_k(t)|^2}{h_{k-1}^2(t)} \tag{7.1}$$

The alternative to the SD is similar to Cauchy convergence test (CCT) [9]:

$$SC = CCT = \frac{\sum_{t=0}^{T} |h_{k-1}(t) - h_k(t)|^2}{\sum_{t=0}^{T} h_{k-1}^2(t)} \tag{7.2}$$

The sifting process will stop when the SC is smaller than the selected threshold. The second stoppage criterion is based on the S-number defined as the number of consecutive sifting when the number of zero-crossings and extrema are equal or differs by one at most.

7.3 Hilbert Transform

Hilbert transform (HT) [21; 31] returns the analytic signal from real data sequence. The analytic signal $x = x_r + i \cdot x_i$ has its real part, x_r which represents the original data, and its imaginary partx_i, which contains the Hilbert transform. The imaginary part is a version of the original real sequence with a 90° phase shift. Sines are therefore transformed to cosines and vice versa. The Hilbert transformed series has the same amplitude and frequency content as the original real data and includes phase information that depends on the phase of the original data.

The Hilbert transform is useful for calculating instantaneous attributes of time series, especially the amplitude and frequency. The instantaneous amplitude is the amplitude of the complex Hilbert transform; the instantaneous frequency expresses the rate of change of the instantaneous phase angle. In case of a pure sinusoid, the instantaneous amplitude and frequency are constant.

7.3.1 Computing Standard Discrete-Time Analytic Signal of Same Sample Rate

The analytic signal for a sequence x has a one-sided Fourier transform (with 0 negative frequencies). To approximate the analytic signal, the Hilbert method calculates a FFT of the input sequence, replaces those FFT coefficients corresponding to negative frequencies with zeros, and calculates an inverse FFT of the result. In detail, Hilbert uses a four-step algorithm [31]:

1. It calculates the FFT of the input sequence, storing the result in a vector x.
2. It creates a vector h with following values:
   - 1 for i = 1, (n/2)+1
   - 2 for i = 2, 3, ... , (n/2)
   - 0 for i = (n/2)+2, ... , n
3. It calculates the element-wise product of x and h.
4. It calculates the inverse Fast-Fourier transform (FFT) of the sequence obtained in step 3 and returns the first n elements of the result.

7.3.2 Representing the Result of Hilbert Transform

When we have all IMFs from EMD, we can calculate the analytic signal by using the algorithm described in section 7.3.1. Calculated analytic signal $Z(t)$ is defined as [5; 10]:

$$Z(t) = X(t) + iY(t) = a(t)e^{i\theta(t)}, \tag{7.3}$$

where $X(t)$ is the original signal, $Y(t)$ the Hilbert transform of $X(t)$, so the instantaneous attributes of $Z(t)$ are defined:

$$a(t) = \sqrt{X(t)^2 + Y(t)^2} \tag{7.4}$$

$$\theta(t) = arctan(\frac{Y(t)}{X(t)}) \tag{7.5}$$

$$\omega(t) = \frac{d\theta(t)}{dt} \tag{7.6}$$

where $a(t)$ is the instantaneous amplitude, $\theta(t)$ is the instantaneous phase and $\omega(t)$ is the desired instantaneous frequency. If you want to know more about visualization see [2; 17].

7.4 Application of HHT

The Hilbert-Huang transform was designed recently, in 1998 [11]. So far it has been used in various domains where it is necessary to process non-linear and non-stationary data. The method has particular properties and advantages described in table 7.1. Basis functions used for decomposition are derived from data itself when the EMD is performed (other methods like Fourier, Wavelet transforms and Matching pursuit have their bases function defined a priori). The HHT derives the frequency by differentiation, therefore the HHT has no uncertainty principle limitation on time or frequency resolution [10].

The Hilbert-Huang transform (HHT) was originally designed for study of fluid mechanics [11] and in the same year used in biomedical engineering [12]. The empirical mode decomposition (EMD) has been already extended into 2D form [8]. The list of domains, where is HHT used, and interesting articles will follow:

- biomedical engineering
  - Engineering analysis of biological variables: An example of blood pressure over 1 day [12]

	Fourier	Wavelet	HHT
Basis	a priori	a priori	Adaptive
Frequency	Convolution: global, uncertainty	Convolution: global, uncertainty	Differential: local, certainty
Presentation	Energy-frequency	Energy-time--frequency	Energy-time--frequency
Nonlinear	No	No	Yes
Nonstationary	No	Yes	Yes
Feature Extraction	No	Discrete: no Continuous: yes	Yes

Table 7.1: Comparison between Fourier transform, wavelet transform and HHT [10].

  - The local mean decomposition and its application to EEG perception data [33]
  - a New Tool for Nonstationary and Nonlinear Signals: The Hilbert-Huang Transform in Biomedical Applications [8]
  - Epileptic Seizure Detection Using Empirical Mode Decomposition [35]

- Mechanical engineering
  - An improved method for restraining the end effect in empirical mode decomposition and its applications to the fault diagnosis of large rotating machinery [28]
- Electrical engineering
  - The Application of a New Process Method for End Effects of EMD in the Insulator State Diagnosis [40]
- Economy
  - Identifying the oil price–macroeconomy relationship: An empirical mode decomposition analysis of US data [24]

The HHT was successfully used for processing non-stationary data in different fields, where the high frequency-time resolution is required. Therefore, the HHT seems to be suitable for the ERP detection and EEG signal processing.

Chapter 8

Application of HHT for EEG Processing

8.1 Creating Envelopes During EMD

When the EMD is performed on the data series, we are trying to create upper and lower envelopes by connecting local extrema with cubic spline. Though, some difficulties surface in the process. When we want to create an envelope which covers whole signal, we have to realize that some first and last points of the signal don't count as local extrema. The closest extremum to the beginning or the end of the signal belongs to the upper or lower envelope. Then the second closest extremum is the point from where the both envelopes are defined.

We have to add additional extremum points to extend the envelopes over the whole signal. It is the difficult part. We have to position them very carefully, because their incorrect location leads to imprecise estimate of the cubic spline (see figures 8.1 and 8.2). This overshoots or undershoots don't describe characteristics of the signal, but they could be propagated inward and corrupt the whole signal. The problem is described in [4; 6; 29; 40; 41] in detail.

To create a complete envelope and restrain the overshoot effect, several methods of additional extrema selection were proposed. They are described in following chapters.

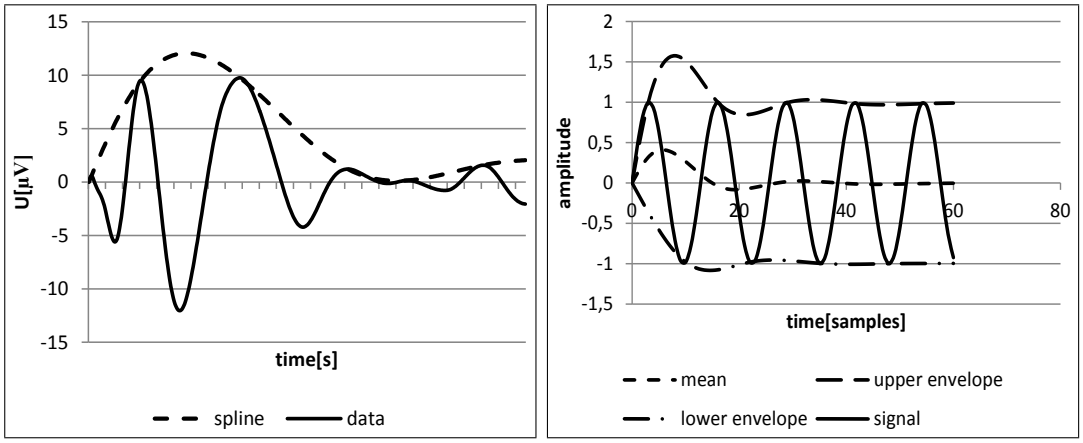

Figure 8.1: Example of a signal and the spline overshoot effect. Undershoot effects are better visible in figures 8.2.

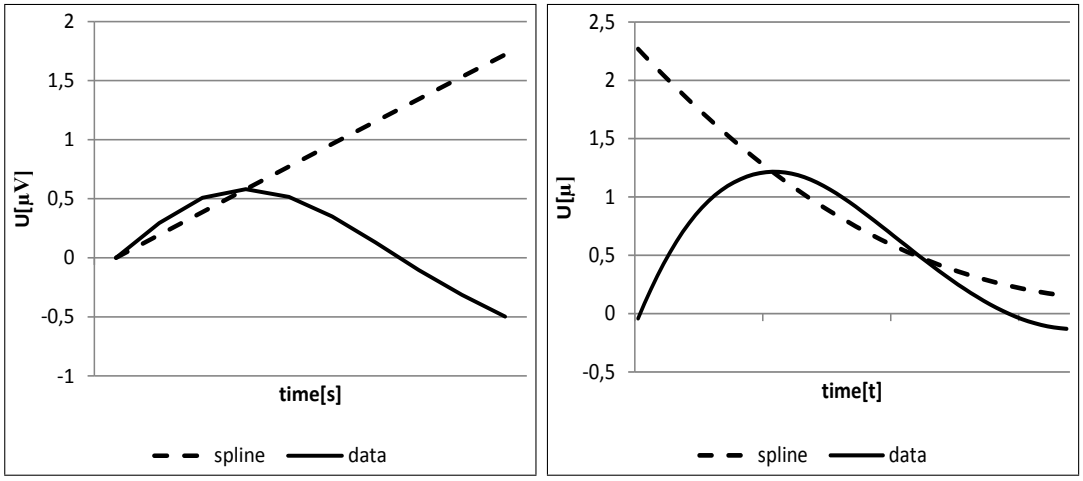

Figure 8.2: Detail of spline undershoot effects.

8.1.1 Mirror Method

Mirror method was proposed by Rilling [30] and described in [28]. The procedure is very simple. Additional extrema are mirror symmetric to the extrema that are closest to the beginning or end of the signal. The algorithm follows:

1. Locate the extremum closest to the begin of the signal (we found $Max(1)$). Then locate the extremum closest to $Max(1)$, this is $Min(1)$

2. Create new extremum on the begin of the data by creating $Min(0)$ respecting the mirror symmetry.

$$Min_x(0) = Max_x(1) - (Min_x(1) - Max_x(1)), \tag{8.1}$$

$$Min_y(0) = Min_y(1) \tag{8.2}$$

3. Repeat this process until the end of the signal is reached.

8.1.2 Slope-Base Method

The slope based method was proposed in [6] and described in [28]. This method also extends extrema, but adds one minimum and one maximum to the beginning or end of the signal. The new extrema are calculated from mathematically defined slopes created through the extrema. These slopes are derived from the distances between successive minimums and maximums and from amplitude differences. The method is shown in figure 8.4.

In the first step we have to calculate the slopes s_1 and s_2 for the signal $x(t)$ shown in the figure 8.4. The slopes are defined as:

$$s_1 = \frac{Max_y(2) - Min_y(1)}{Max_x(2) - Min_x(1)} \tag{8.3}$$

$$s_2 = \frac{Min_y(1) - Max_y(1)}{Min_y(1) - Max_x(1)} \tag{8.4}$$

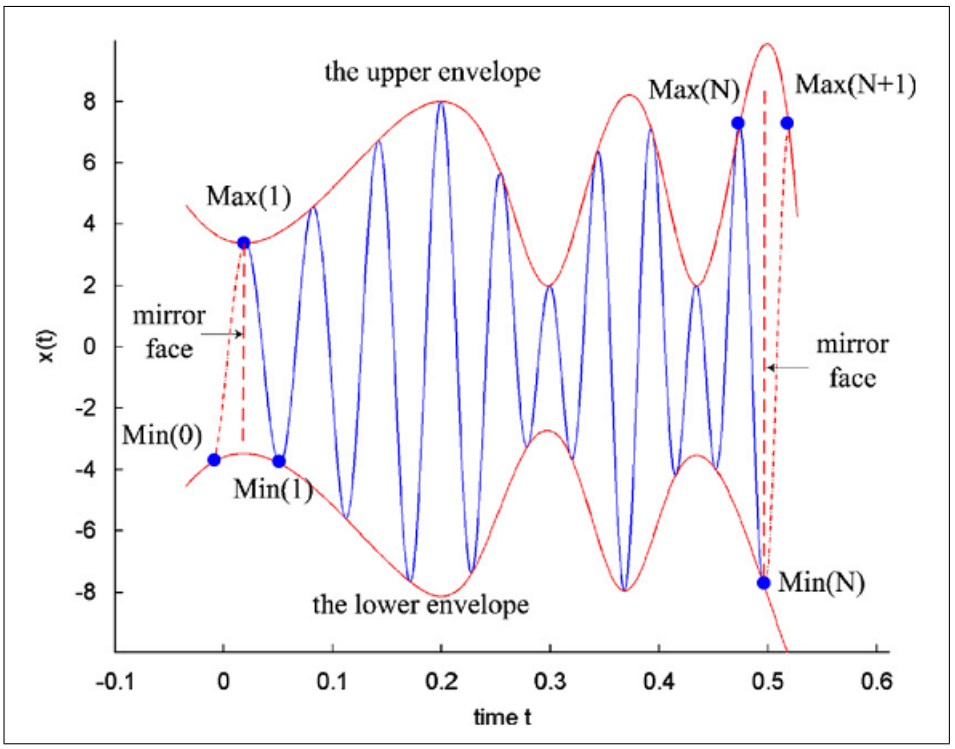

Figure 8.3: Demonstration of the mirror method from [28].

The x coordinates are defined as

$$\Delta t_{\max}(1) = Max_x(2) - Max_x(1) \tag{8.5}$$

$$\Delta t_{\min}(1) = Min_x(2) - Min_x(1) \tag{8.6}$$

$$Max_x(0) = Max_x(1) - \Delta t_{\max}(1) \tag{8.7}$$

$$Min_x(0) = Min_x(1) - \Delta t_{\min}(1) \tag{8.8}$$

Then we have to calculate the Y values of new maximum and minimum:

$$Min_y(0) = Max_y(1) - s_1 \cdot (Max_x(1) - Min_x(0)) \tag{8.9}$$

$$Max_y(0) = Min_y(0) - s_2 \cdot (Min_x(0) - Max_x(0)) \tag{8.10}$$

This procedure has to be repeated in order to generate additional extrema at the end of the signal. See more in [6; 28].

8.1.3 Drawbacks of Mirror and Slope-Based methods in EEG signal processing

When we are performing EMD on the EEG signal, we want to create envelopes covering the signal completely. The mirror method and slope based method create additional extrema to ensure this condition. The weak point of these two methods is the estimation of an additional extrema position on the time axis.

When edges of the processed signal contain time-short components of significantly higher frequency (in our case artifacts), the insufficiency of methods is apparent. The problem surfaces distinctively when we use artificial signals with randomly placed artifacts (see the figure 8.5).

The example signal in the figure 8.5 starts with relatively slow frequencies but there is an artifact (with high frequency and amplitude) in the short interval after the start. Unfortunately, this artifact includes all four extrema, which we use to estimate new extrema to extend envelopes, as you can see in figures 8.6 and 8.7.

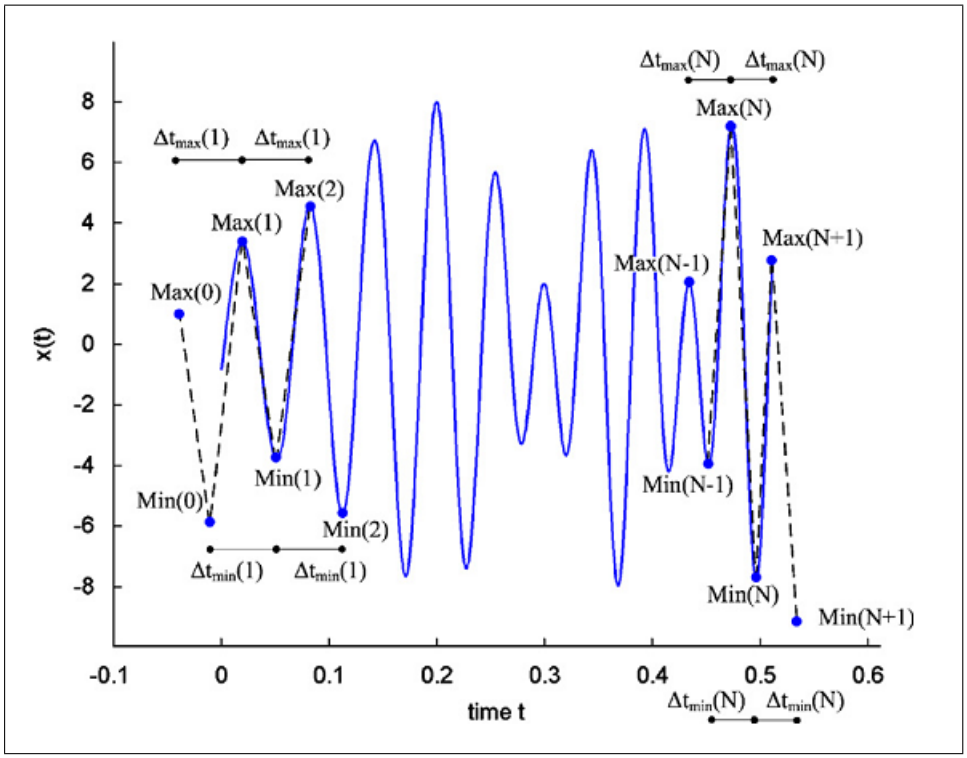

Figure 8.4: The illustration of the slope based method from [28].

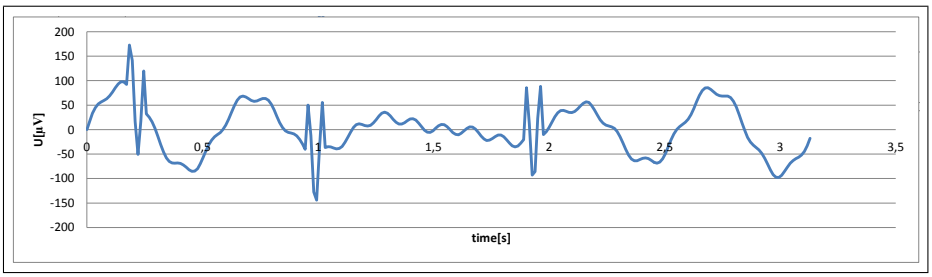

Figure 8.5: Example of artificial signal which represents the EEG signal with artifacts.

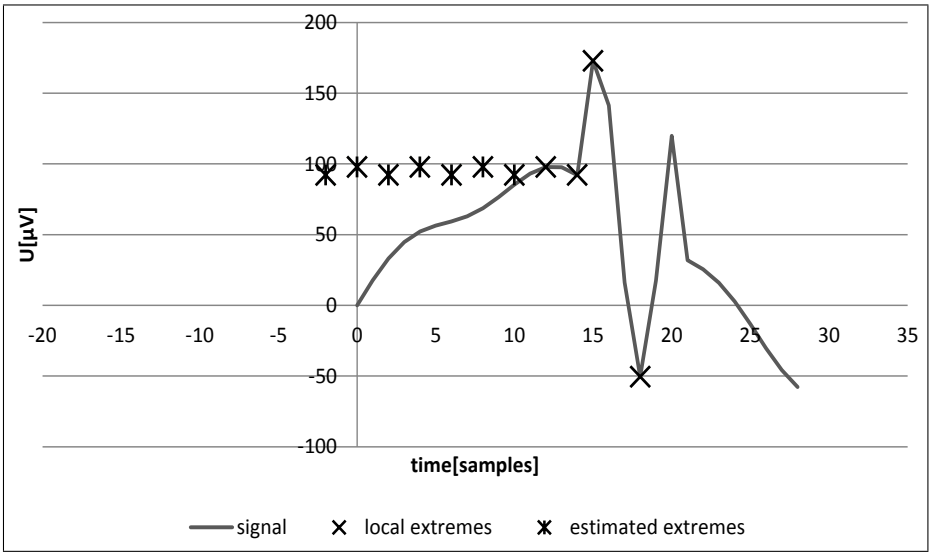

Figure 8.6: Example of poorly estimated additional extrema point with mirror method.

When we calculate a new extremum with mirror method according to algorithm described in chapter 8.1.1, we get a new minimum with x-coordinate:

$$Min_x(0) = Max_x(1) - (Min_x(1) - Max_x(1)) = 13 - (15 - 13) = 11$$

Index of a sample was used as the x-coordinate. The new minimum is at the position of the 11^{th} sample. Therefore we cannot create the envelope covering all the data. Similar problem appears when we use the slope based method to estimate x-coordinates of new extrema:

$$Max_x(0) = Max_x(1) - \Delta t_{\max}(1) = Max_x(1) - (Max_x(2) - Max_x(1))$$

$$Max_x(0) = 13 - (16 - 13) = 10$$

$$Min_x(0) = Min_x(1) - \Delta t_{\min}(1) = Min_x(1) - (Min_x(2) - Min_x(1))$$

$$Min_x(0) = 15 - (19 - 15) = 11$$

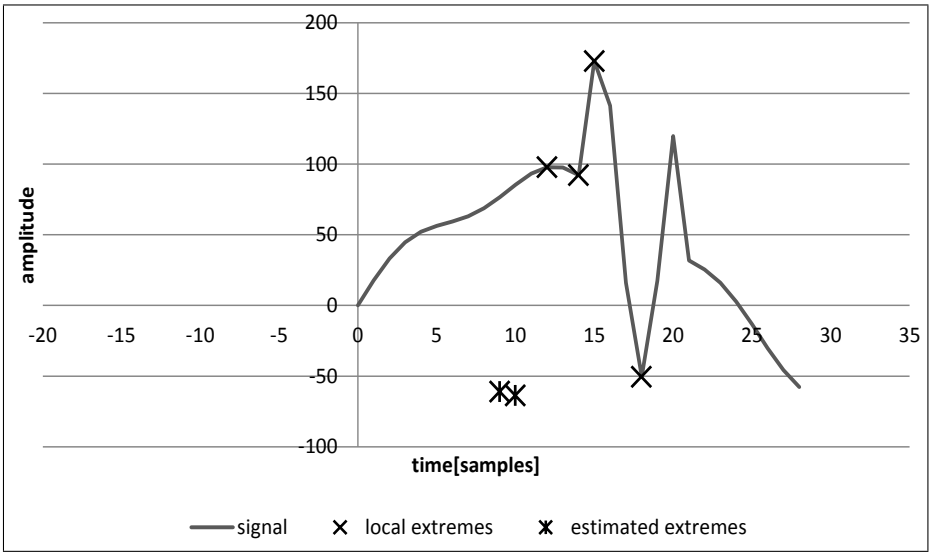

Figure 8.7: Example of poorly estimated additional extrema with slope based method.

We also use the index of the sample as the x-coordinate. Newly estimated extrema have its x-coordinate before the beginning of the signal. So we cannot construct the proper envelope for the sifting process (similarly described in [41].

The mirror method and slope-based method are not suitable for processing the EEG signal, because the EEG signal could contain significant changes of amplitude and frequency (artifacts). Both methods are not able to estimate additional extrema, which could determine points of complete envelopes. A significant change of amplitude and frequency is present in the processed signal; therefore, it is neccessary to design other methods for estimating additional extrema.

Chapter 9

Proposed Modifications of HHT

9.1 Methods for Estimating Additional Extrema Points

Because there is no suitable method for estimating additional extrema points, I have proposed two methods for the task. The first one is much simplier than the other, derived from the Mirror method and adapted for EEG signal processing.

9.1.1 First/Last Points Method

The principle of the method is very simple. To create a complete envelope all over the signal, we add simply the first and last point of the signal to the set of maximums and minimums. Then we can create complete envelopes. This simple approach enables to create complete envelopes but could cause significant over/undershoot effects in figures 8.2;

9.1.2 Requirements on the New Method for Estimating Additional Extrema

The new method for estimating additional extrema points has to deal with several requirements. These requirements were established during testing with First/Last, Slope-based and Mirror methods on real EEG data. Established re-

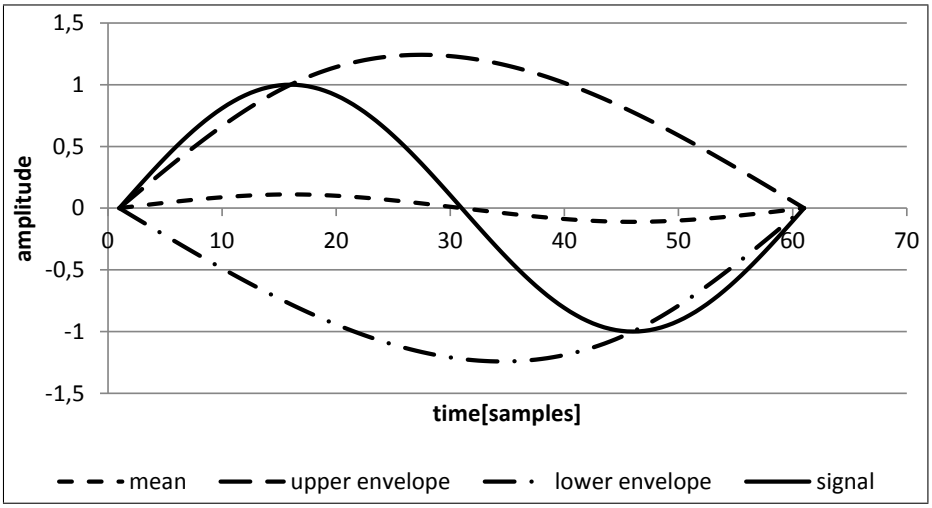

Figure 9.1: Misplaced additional extrema created by First/Last method

quirements represent drawbacks of tested methods which has to be solved to adapt the EMD for EEG signal processing. List of established requirements follows:

1. artifacts in the beginning/end of the signal (described in section 8.1.3)
2. minimize overshoot effect
3. envelopes mean of IMF should be zero

Mean of envelopes

The mean calculated from the upper and lower envelope should be zero, this is one of conditions defining the IMF (section 7.1). This condition depends on calculated envelopes derived from detected local extrema and additional extrema which are placed at the beginning/end of the processed signal. So the additional extrema could affect the mean of the envelope.

I have tested this phenomena on a simple sinus wave which is definitely an IMF. The mean value of the envelope the simple sinus wave should be equal to zero in its every point. But misplaced additional extrema can cause non-zero mean values of the envelopes. An example of such misplaced extrema and their impact on the mean of envelopes can be seen in the figure 9.1.

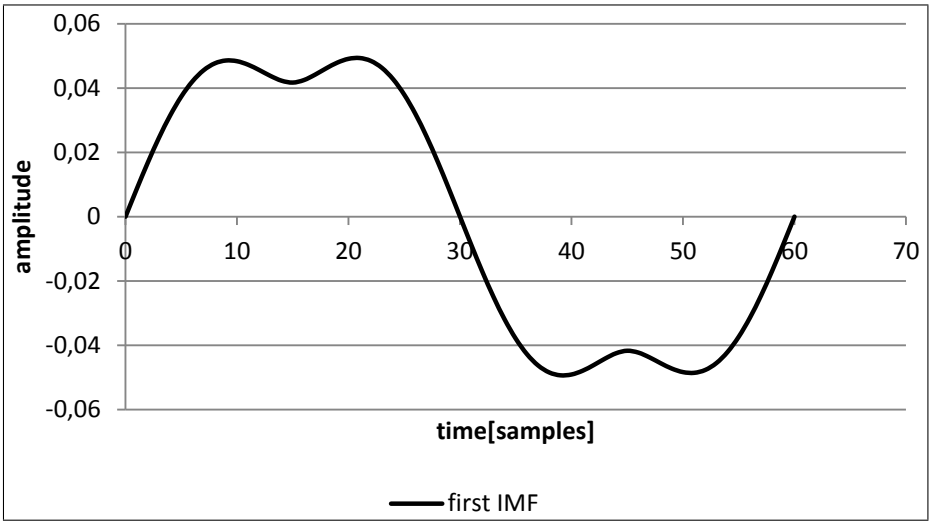

Figure 9.2: Example of distorted IMF, caused by misplaced additional extrema.

To calculate the mean of envelopes correctly is crucial for all stopping criteria applied during the sifting process. When the EMD sifts the simple sinus wave, it should stop immediately, because its envelopes mean is equal to zero and therefore the mean won't change in further iterations of the sifting process (the mean is subtracted from the signal see section 7.2). But when the envelopes mean is inaccurate due to misplaced additional extrema, the EMD continues with sifting and the envelopes mean varies from one iteration to another. Due to the variation of the envelopes mean, the EMD decomposes the simple sinus wave into several IMFs instead of single IMF, because the residue contains fluctuations from the envelopes mean. This affected the convergence speed of the EMD.

Minimization of the Overshoot effect

The overshoot/undershoot effect is a situation when the envelopes don't follow the trend of the signal precisely (see more in [29; 30]). a similar situation can be seen in the figure 9.1. When the overshoot/undershoot effect is present in every iteration of the sifting process, then it propagates inward and causes distortion of created IMFs (like the first IMF of the testing sinus wave in the figure 9.2). More iterations with overshoot/undershoot effect is performed, the more significant

distortion we get. Therefore, the overshoot/undershoot effect is more dangerous when the EMD converges slowly.

Additional extrema affect the overshoot/undershoot effect significantly in the beginning and the end of the processed signal. With better estimated additional extrema, the over/under-shoot effect could be minimized like in the figure 9.4.

9.1.3 Modified Mirror Method

According to established requirements introduced in section 9.1.2, I have designed a method based on the Mirror method (section 8.1.1). Its algorithm is very simple:

1. Locate the closest minimum and maximum at the signal beginning.
2. Create a new extrema that it would preceed the beginning of the signal and respect the mirror symmetry, as in equations 9.1 and 9.2.
3. Locate the first extremum of the signal. (In our case it is $Max(0)$.) It means that the signal before extremum has ascending trend.
4. Therefore, we have to ensure that the value of the new minimum $Min_y(0)$ is lower or equal than the first value of the signal($x(0)$). If the $Min_y(0)$ is greater than the first signal value($x(0)$), we simply replace $Min(0)$ with $x(0)$. This ensures that the envelope will respect the trend at the beginning of the signal(see the figure 9.3).

$$Min_x(0) = -Min_x(1), \qquad Min_y(0) = Min_y(1) \tag{9.1}$$

$$Max_x(0) = -Max_x(1), \qquad Max_y(0) = Max_y(1) \tag{9.2}$$

An accurate local extrema detection (see figure 11.3 and section 11.3) are essential for the Modified mirror method. Without the accurate local extrema detection, the estimated additional extrema would be a mirror images of points which are not considered as suitable extrema for envelopes (see more in section 9.2).

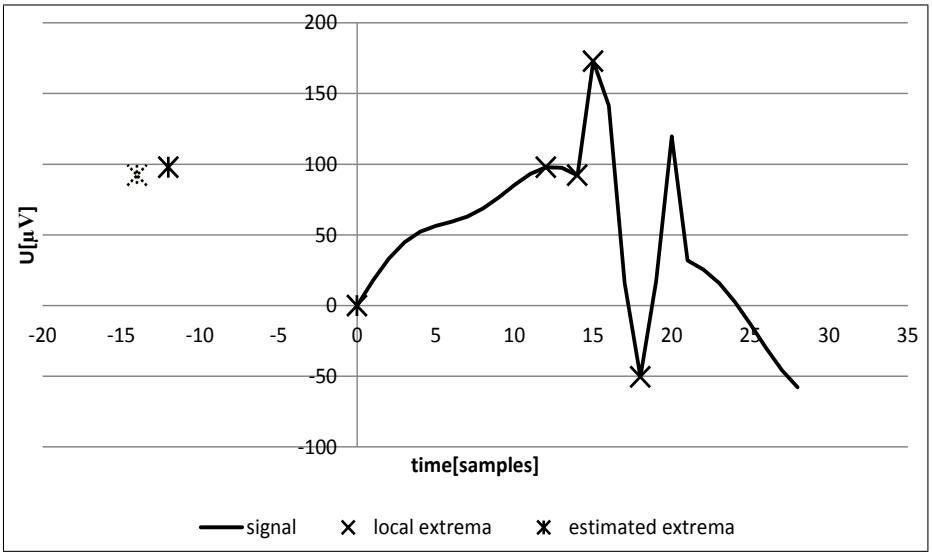

Figure 9.3: Modified Mirror Method

When the envelopes are created with additional extrema estimated by Modified mirror method, the envelopes mean is equal to zero at every point for testing simple sinus wave (figure 9.4. This means, that Modified mirror method makes the sifting process more stable, ensures its better convergence speed, and prevents from distortion of created IMFs due to the over/undershoot effect.

The modified mirror method minimizes the over/shoot effect, because it ensures that created envelopes are closer to the detected local extrema (figure 9.4). Then the EMD decomposes the simple sinus wave into single IMF which is exactly the same as the input wave.

9.2 Local Extrema Detection in the EEG Signal

Local extrema detection is essential for creating envelopes during the sifting process (part of the EMD)and significantly affects the stopping criterion of entire EMD (see more in [26; 29; 30]). The EMD ends when the residue after sifting has less than two extrema(section 7.2).

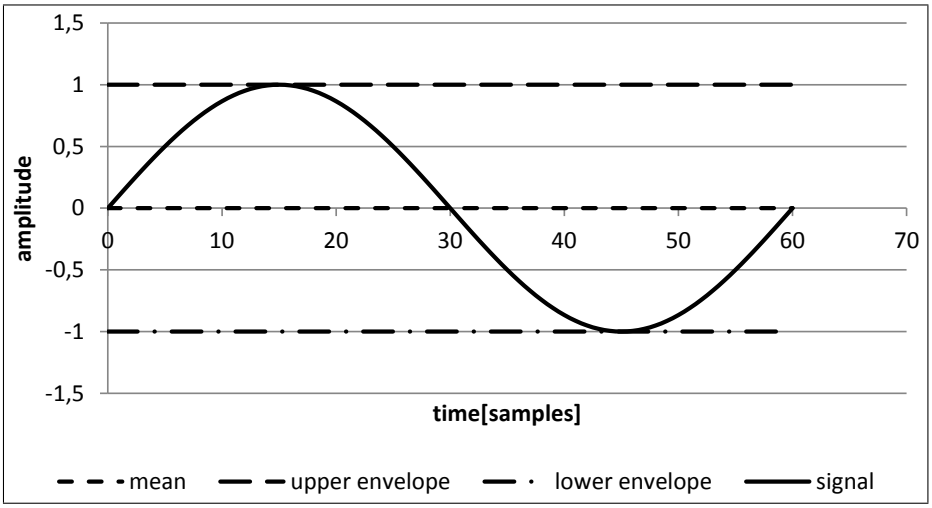

Figure 9.4: Envelopes created with additional extrema estimated by Modified mirror method

9.2.1 Inflection Point Method

Another very simple method for local extrema detection is based on detection of inflection points. To locate the local extremum, it is necessary to test three data points. When the middle one is greater/lower than the other, it is marked as the local maximum/minimum. This method considers every inflection point as a local extremum.

Tests on the real EEG data acquired in our laboratory revealed some drawbacks of this approach. It seems that not every inflection point is a local extremum. Some of inflection points could be created by signal preprocessing techniques like filtering, averaging etc. The amplitude of false detected inflection points is insignificantly higher/lower than the amplitude of surrounding points. The absolute value of the difference could be lower than the resolution of used analog-to-digital converter.

Moreover, some of the extrema cannot be detected with this method at all. When we have set of two (or more) points with equal amplitude and their amplitude is higher/lower than amplitude of theirs surrounding points, one of these points should be marked as an extremum. But when using this method, no points are marked as extrema, because there is no point of inflection. Therefore, I have

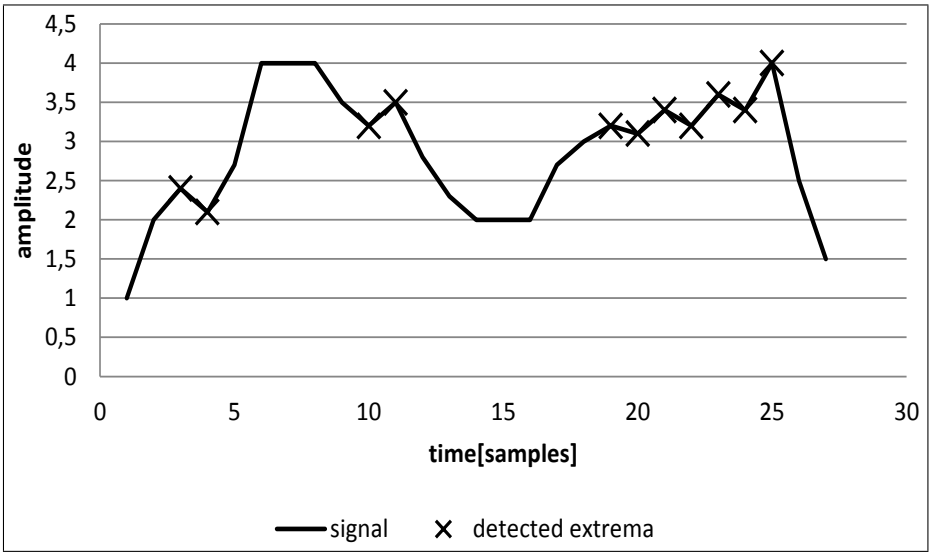

Figure 9.5: Detected extrema with Inflection Point Method

to designed more suitable method for local extrema detection. The method is described in the next section 9.2.2.

9.2.2 Delta Difference Method

The upper and lower envelope should cover all the data within. [10]. This condition cannot be fulfilled, when the extrema are incorrectly detected; as you can see in the figure 9.5. Therefore, I have decided to design a new, more suitable method for extrema detection. First I have stated two requirements which the new method has to fulfill:

- Ignore insignificant changes in the amplitude.
- Detect the extrema, even when there is no inflection point.

The following algorithm can fulfill these two requirements:

1. Initialize threshold δ, sample index $i = 0$
2. Initialize base value with first sample and $max_p = null$

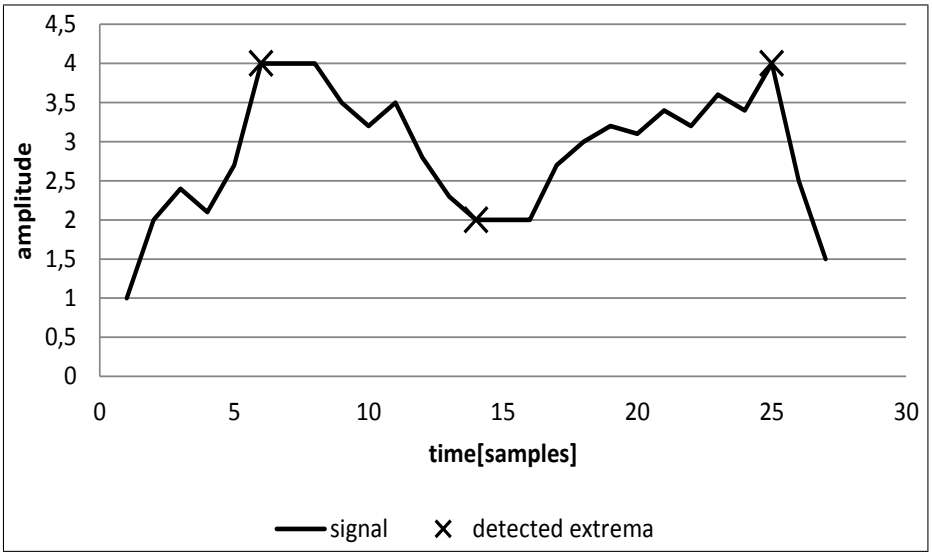

Figure 9.6: Extrema detected by Delta Difference Method

3. Store amplitude of the i-th sample into $y = x[i]$
4. If $y_i > (y_{i-1} - \delta)$ then accept i-th sample as new potential extremum max_p
5. If $(max_p - y_i) > \delta$ then max_p is the maximum, start search for another one, $max_p = null$
6. If $max_p == null$ and $base < y_i$ then $base = y_i$
7. i++, continue with 3.

The δ parameter represents the tolerance for small amplitude fluctuations. When it is set to zero, the method detects exactly the same extrema as the Inflection Point method introduced in section (9.2.1). This method marks the sample as a maximum when some of previous and next samples has the amplitude lower and the difference between amplitudes is greater than value of the δ parameter. An example of detected extrema can be seen in the figure 9.6.

By a simple comparison of detected extrema on figures 9.5 and 9.6 we could intuitively recognize that detected extrema by Delta Difference method are more convenient for creating envelopes.

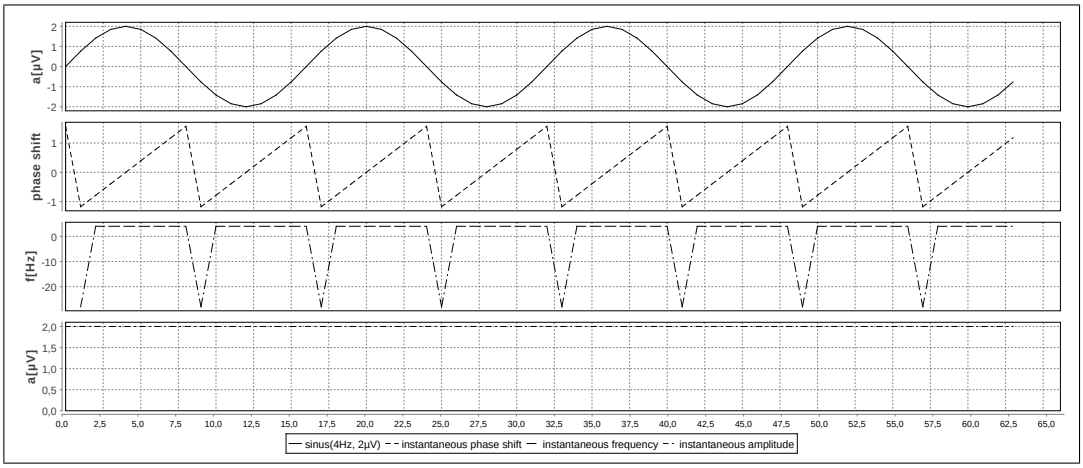

Figure 9.7: Hilbert transform results of simple sinus wave using simple arctan

9.3 Instantaneous Frequency Calculation from the Analytic Signal

After performing Hilbert transform on detected IMFs, the instantaneous amplitude and frequency are calculated according to equations 7.4, 7.5 and 7.6. To calculate an instantaneous amplitude is simple, but calculate the instantaneous phase shift, which is needed for instantaneous frequency, could be difficult in some cases.

Let us consider a simple sinus function with frequency 4Hz and amplitude $2\mu V$, which represents an IMF. After performing the Hilbert transform, we get the analytic signal. From acquired analytic signal we can calculate the instantaneous amplitude and the instantaneous phase shift (see figure 9.8).

In the next step we want to calculate the instantaneous frequency according to the equation 7.6. The instantaneous phase shift is stored as an array of doubles. Therefore, we calculate the instantaneous frequency as

$$f(i) = \frac{[\theta(i) - \theta(i-1)] * f_s}{2\pi} \tag{9.3}$$

But there is a numerical issue in the process of the phase shift calculation. When we calculate the phase shift with ordinary arctan function, the result fits in the interval $(-\pi/2; \pi/2)$ for every half period of the sinus function. Therefore, the difference of the phase shift calculated from the last and the first point of the interval is $(-\pi/2 - \pi/2) = -\pi$. When we get the negative difference between two phase shift values, the calculated instantaneous frequency would be negative (see figure 9.7. But negative frequency is meaningless, therefore we have to minimize or eliminated this phenomena.

First step to minimize negative frequencies, is to use arctan function which respects quadrants. This function is usually called arctan2 and it accepts two arguments - x and y coordinates and returns the angle between them in the correct quadrant. The return value fits in a range from $-\pi$ to π radians. Using the arctan2 function, we can minimize the crossing which produces mentioned negative frequencies (compare figures 9.8 and 9.7).

When we use arctan2, we get even higher values of negative frequencies at the crossing, because the difference at the crossing point is -2π now. But it is possible to correct the values of negative instantaneous frequency simply. We could assume that the processed signal has almost the same frequency all the time. This assumption is based on properties of IMFs. If the difference of two phase shift values is close to the value of -2π, we could assume, that at this point the signal transcendes from one period to another. The negative frequency f(i) can be easily replaced with the average of f(i-1) and f(i+1). Using these improvements we get better instantaneous frequencies, as you can see on figures 9.8 and 9.7;

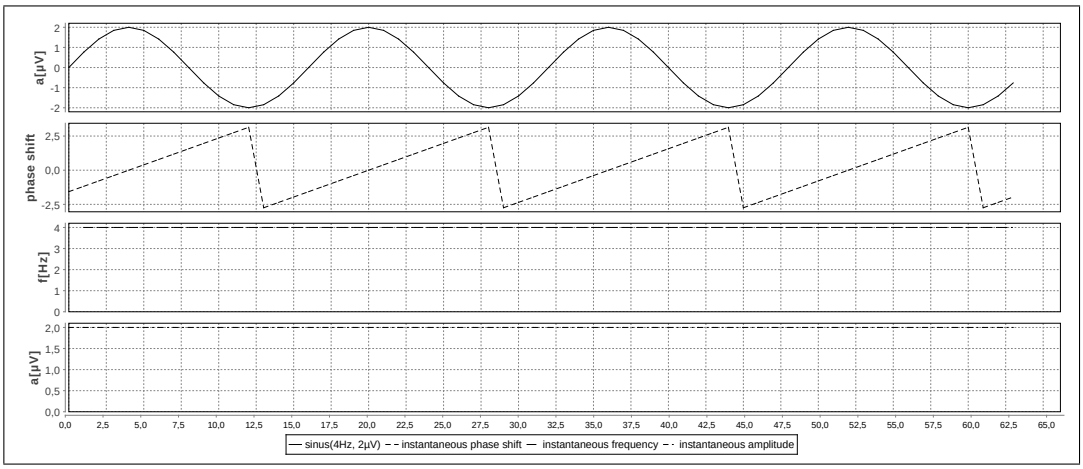

Figure 9.8: Hilbert transform results of simple sinus wave using arctan2

Chapter 10

Implementation

When I started my research in this area, there was only one freely accessible implementation of the HHT. It's written in Matlab. In our laboratory we intend to use the HHT for many purposes, such as live signal processing during experiments and integrating it into our Portal of EEG/ERP Experiments http://eegdatabase.kiv.zcu.cz/home.html. Therefore, I have decided to create my own implementation of the HHT in Java. We have already used this programming language for our development of the portal and other tools.

I have implemented the HHT as a part of my experimental software. Its implementation is more likely a library than an application. There is no need to implement a complex application, because the HHT module will be integrated into other tools developed in our research group. My implementation is open to modifications and suitable for testing the HHT.

My implementation could be divided into three main modules:

1. core of the HHT
2. logging and visualization
3. testing

Modules will be briefly described in the following sections.

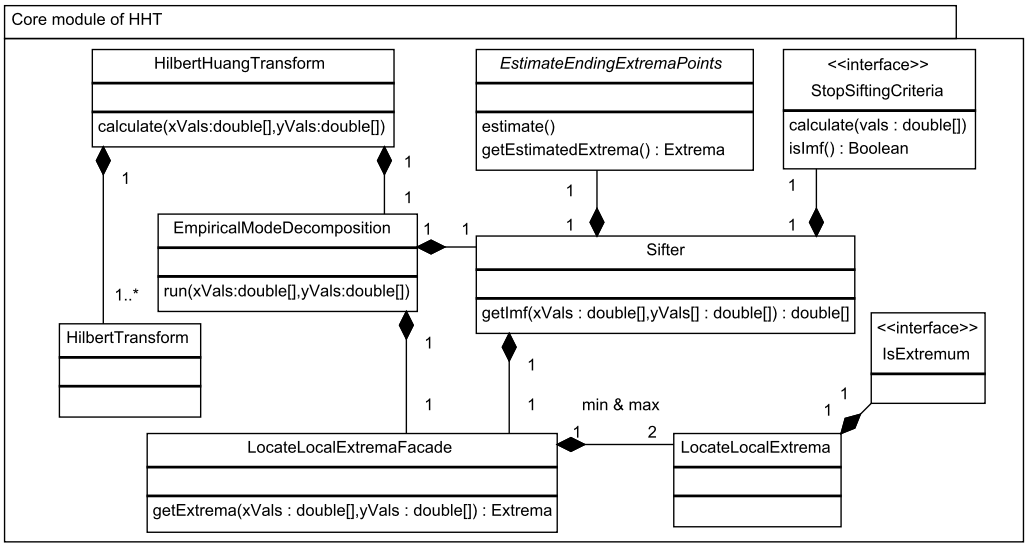

Figure 10.1: Class diagram of Core module of HHT

10.1 Core Module of HHT

The core module contains all necessary classes for the Hilbert-Huang transform. The class structure is displayed in the class diagram (figure 10.1). The class diagram doesn't contain all of them, but only the most important. As you can see, all the important parts (extrema detection, stopping criteria, estimation of extrema points) of the EMD can be simply replaced with different implementation.

EmpiricalModeDecomoposition class

This class performs the EMD on input data and returns a list of double arrays which represents IMFs. It requires instances of classes LocateLocalExtremaFacade and Sifter. LocateLocalExtremaFacade is required, because the decomposition stops when there are less than two extrema in the residue. The decomposition is executed by calling the *run(...)* method.

Sifter class

The Sifter class performs the sifting process on input data and returns detected IMF as an array of double For the sifting process, the Sifter needs instances of EstimateEndingExtremaPoints, StopSiftingCriteria, LocateLocalExtremaFacade classes. Internally the Sifter uses an instance of the Enveloper class to create envelopes (upper and lower). The IMF itself is calculated by calling the method *getImf(. . .)*.

LocateLocalExtremaFacade class

The easiest way to obtain both sets of local extrema (minima and maxima) is to use this. Internally the class uses the *LocateLocalExtrema* class which iterates through the input data and checks via *IsExtremum* interface, whether a point belongs to local extrema. Therefore, the LocateLocalExtremaFacade class requires two instances of IsExtremum interface - for minima and for maxima detection. There are two implementations of the local extrema detection method. *MaximumFinder* and *MinimumFinder* represents the inflection point method (section 9.2.1) and ExtremaLocator represents the Delta-difference method. ExtremaLocator requires instances of MaximumLocatorHelper or MinimumLocatorHelper class.

StopSiftingCriteria interface

The interface represents necessary methods to stop the sifting process. Sifting process should be stopped when an IMF is detected.

I have implemented the Standard Deviation (StandardDeviation class) stopping condition and Cauchy convergence test (CauchyConvergence class), both rules were described in section 7.2.1.

HilbertHuangTransform class

HilbertHuangTransform class applies the HHT on the input data. First, it runs the EMD and after retrieving all IMFs, it performs the Hilbert transform on every

single one of them. The result is saved as a list of instances of HilbertTransform class, one per each IMF.

10.2 Configuration of Empirical Mode Decomposition

An instance of the EmpiricalModeDecomposition class can be created in the source code by composing required instances of classes together. Another way to create an instance is to use an XML file as the configuration file. For this purpose I have used a part of the Spring Framework, the XmlBeanFactory. It instantiates Java Beans from a XML file (see example on page 97). The XML file also contains the configuration settings which could be easily accessed this way.

10.2.1 How to run the HHT easily

The easiest way to run the HHT and save its results, is to use static methods in the class HhtSimpleRunner. The methods are :

1. runHht(String resultPath, String emdCfg, double[] xVals, double[] yVals, int samplingFrequency, ResultsConfigs resultsCfg)
2. runHht(String resultPath,String emdCfg, double[] xVals, double[] yVals, int samplingFrequency)
3. runHhtDataFromTxt(String resultPath, String emdCfg, String dataFile, int samplingFrequency, int samplesCount)

where *resultPath* is the path where the results and logs should be saved ; *emdCfg* is a filename of the EMD configuration file; *dataFile* is a name of the file with the source signal; resultsCfg describes, which result data are going to be saved and whether a PDF preview should be created.

10.3 Module for Logging and Visualization

During my delevoplent of the method, I needed to record every step - such as calculated IMFs, each iteration of sifting process, envelopes, or convergence of the sifting criteria - for debugging purposes. Therefore it was necessary to implement a flexible logging mechanism and simple visualization of logged data.

10.3.1 Using Aspect-Oriented Programming

I wanted to keep the actual code of the HHT separated from the logging part which would lose its usefullness after conclusion of the development. I have decided to use the aspect-oriented programming (AOP) technique, namely the AspectJ library. This AOP library was designed for Java. AOP makes it possible to remove the logging part when the development of my method is finished. Also, the classes implementing the HHT will be more understandable and readable without the logging code.

Because of my focus on the EMD, I needed to store data about the convergence of sifting criteria and resulting envelopes, their mean values, and processed signal into separated files. Therefore I have implemented three aspects SiftingConvergenceLogger, EnvelopesLogger and EmdLogger.

SiftingConvergenceLogger

This aspect logs the data about the convergence of the sifting process. Its poincut (called isImf) is linked to the interface method StopSiftingCriteria.isImf method, so all results of implementation of this interface are automatically logged. After each call of the isImf method, the current value of the criterion evaluation is logged. If it rises (instead of expected decline, which would signify improving convergence), it is logged as a warning with the current value and the previous value of the stopping criterion. These information is very useful for testing new stopping criteria and for setting suitable threshold values of the stopping criteria.

EnvelopesLoggger

This aspect logs the created envelopes, their mean and the input data for each iteration of the sifting process. Used pointcut (called meanCurve) is linked to the method Enveloper.countEnvelopesMeanCurve (the class is responsible for calculating envelopes). This method calculated the mean from the upper and lower envelope. After calculating, the mean curve, the mean itself, the upper and lower envelope, and input data are written into a file by EnvelopesFileAppender (section 10.3.2.1).

EmdLogger

The EmdLogger aspect logs values and classes of input parameters and results of all invoked methods at every place of my implementation of the HHT. During the logging, it indents methods called within the superior method. Stored data are very useful for debugging. The detail level of logged information can be set in the log4j.properties file (section 10.3.2). List of classes and packages to be logged, can be listed in the properties file as well.

10.3.2 Logging

For logging itself I have used most popular logging library for Java *log4j*. I have chosen this library because it is simple to use and offers great amount of configuration options. It could be configured via its properties file and also by changes in the source code in run-time. An user can choose the level of details to be logged (TRACK, DEBUG, INFO, WARNING, ERROR) by each logger (see listing 10.1), even for each class or package (see listing 10.2).

Listing 10.1: Example of log4j levels for each logger

```
log4j.logger.hht.aspects.SiftingConvergenceLogger = INFO, SCL
log4j.logger.hht.aspects.EnvelopesLogger = DEBUG, ENVELOPES
log4j.logger.testing.classificators = TRACE, CLASS
```

Listing 10.2: Example of log4j levels for each class

```
log4j.logger.hht.emd.sifting.stoppingCriteria = TRACE
```

```
log4j.logger.hht.emd.sifting.Enveloper = TRACE
log4j.logger.hht.emd.EmpiricalModeDecomposition = TRACE
```

For saving logs into a file, the log4j offers so called Appenders. Mostly I have been using the FileAppender, which can be set to overwrite logging files and to use a specified layout for logs. The filename of the logger can be read from system properties. It is very useful when the HHT is performed several times on different data and it is necessary to save logs into different files or different directories. It is very simple to use the system property in log4j.properties file. A placeholder has to be inserted (${placeholder.name}) into the file and the value of the system property altered accordingly. (see listing 10.3).

Listing 10.3: Example configuration of FileAppender

```
log4j.appender.SCL=org.apache.log4j.FileAppender
log4j.appender.SCL.File=${sifting.log.filename}
log4j.appender.SCL.append=false
log4j.appender.SCL.layout=org.apache.log4j.PatternLayout
log4j.appender.SCL.layout.ConversionPattern=%5p%6.6r[%t]%x - %m%n
```

10.3.2.1 EnvelopesFileAppender

It was necessary to store each sifting iteration into separate file for further analysis of the EMD and sifting process. The name of the file should consist of the index of currently detected IMF and the index of the iteration of the sifting process. Therefore, I have designed a new appender called EnvelopesFileAppender. This appender automatically increases the counter of IMFs and also it counts iterations of the sifting process. An user needs only to set the path where the files should be stored. The filenames will be like path/imf_0001_iteration_0001. An example of the configuration follows (listing 10.4). The complete log4j.properties file used for logging during testing can be seen in listing 2 on page 98.

Listing 10.4: Configuration of EnvelopesFileAppender

```
log4j.appender.ENVELOPES=data.log.EnvelopesFileAppender
log4j.appender.ENVELOPES.path=log//imfs//
log4j.appender.ENVELOPES.append=false
```

```
log4j.appender.ENVELOPES.separateFiles=true
log4j.appender.ENVELOPES.layout=org.apache.log4j.PatternLayout
log4j.appender.ENVELOPES.layout.ConversionPattern=%m
```

10.3.3 Visualization

The process of testing designed modifications requires the possibility to view and analyse saved logs (see 10.3.2). Vector images seem to be the most suitable form of visualization of collected log data. Therefore I have decided to save them into PDF files. This task has been accomplished with use of the JFreeChart library. I have wrapped it and designed a number of classes to visualize IMFs, envelope logs and time frequency maps.

ImfMapToSignalPdf class

The frequency and amplitude of each sample are saved in the imfs_map directory after the Hilbert trasnform is performed. The files are named imf_map_[index]_[ampl—freq].txt. a PDF file is built from those files. There is the amplitude displayed at the first chart, and the frequency is displayed in the second line. Third chart shows the original signal. The path to those files has to be passed to the main method. An example of the input can be seen on page 99.

ImfsToPdf class

This class creates PDF files from each iteration of the sifting process. Iterations are saved along with envelopes, section 10.3.2.1). a single file is created for each IMF. Such file contains the iterations, which lead to identification of the IMF. The main method requires the path where the data are stored. An example output PDF file can bee seen on page 100.

ImfsToSignalPdf class

The ImfsToSignalPdf class is able to save IMFs of the decomposed signal into PDF files. A file with IMFs is passed as its input parameter. The main method requires the filename of the file where IMFs are saved, to be passed as its input

along with a name of the output file. An example of the result can be seen on page 101.

MapsToPdf class

To create the time-frequency map from files containing frequency and amplitude was implemented the MapsToPdf class. In these two files is stored frequency/amplitude for each sample. Example of the PDF map could be seen on page 102.

10.4 Module for Testing

During tests of the designed modifications, the HHT had to be performed on all the test data with different configuration profiles. Results of each profile was evaluated according to criteria established in section 11.2. Classes described in following section serve this purpose.

10.4.1 Processing Data With Different Configurations of the HHT

The HhtDataRunner class was designed to process all the input data with different settings. The main method requires the configuration file as its input (see listing 10.5.

Listing 10.5: Configuration file for HhtDataRunner class

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="runnerCfg">
  <logpath>relative/path/where/to/store/logs/</logpath>
  <dataPath>path/to/the/directory/with/input/data</dataPath>
  <samplingFrequency>1000</samplingFrequency>
  <!-- read 1024 samples from input file -->
  <useNSamples>1024</useNSamples>
  <resultPath>path/where/to/store/results/</resultPath>
  <cfgsList>
    <name>path/to/emd/cfg/file/EmdCfg1.xml</name>
  </cfgsList>
```

```
  <!-- list of regular expressions
       describing, which data will be processed -->
  <fileNameList>
    <name>10_Cz_?ep\[\d{3}\]_(target|nonTarget){1}\.txt</name>
    <name>20_Cz_?ep\[\d{3}\]_(target|nonTarget){1}\.txt</name>
    <name>30_Cz_?ep\[\d{3}\]_(target|nonTarget){1}\.txt</name>
  </fileNameList>
</configuration>
```

The input data should be stored as simple text files (in my case each testing subject have its own folder). The directory with subjects is passed as the *logpath* parameter. The fileNamelist accepts the regular expressions, which makes possible to store target/non-target and variously preprocessed data into the one subject's directory.

Results of performed HHTs are stored into the directory sets in the *resultsPath* element. For each configuration is created the directory, which name is same as base filename from EMD configuration file (in our case it will be EmdCfg1). In this directory are stored all processed data.

10.4.2 Acquiring Iterations Count From the EMD

The EmpiricalDecompositionClass contains the Counter class, which counts the iterations during the EMD and the sifting process. The counter could limit the iterations count during EMD, when it has set up the *maxValue* property. To limit the iterations count is necessary, because some of configurations don't have to converge. The iterations count is logged into the *itCount.log* file in logs path. This file contains only the iteration count.

For summarising all iterations count of all configurations was implemented IterationCounter class with main method. In the configuration file could be defined several "counter" which could provide different information about sifting. This "counters" view could be done with regular expressions, same way as is shown in the example on page 99.

Success:90,0000%

success[%]	success	ERPs correctly detected	no ERPs correctly detected	classificator	configuration	data name
90,0000%	36/40	18/20	18/20	FreqAmpl4	Cauchy_d_0.1_c_1.0E-4	30_Cz_baseline_ep[000]_

Figure 10.2: Preview of the classifiers result

10.4.3 Classification of Processed Data

For evaluating the modifications of HHT was essential to test the classification. For this purpose I had designed and implemented several classes.

RunClassificationAll class

This class classify all the processed data into two classes target/non-target. Used classifiers have to implement the Classifier interface. Configuration is Spring bean file, which should contains list of used classifiers, list of directories (with data) and list of regular expression representing groups of data, which will be classified. The example of the configuration file could be seen on page 103.

The results of the classifiers process are stored as HTML pages for easier manipulating. Name of each HTML file consists of configuration name, processed pattern (name of the data) and used classifier for example we have configuration named EmdCfg1.xml, used data are 20_Cz (twenty averages from Cz electrode) and classifiers name is Class1 then the filename will be EmdCfg1_20_Cz_Class1.html. Example of the created HTML file is in the figure 10.2.

ClassifiersSucessViewer class

For gathering information about classification reliability of designed classifiers and used configuration of HHT is designed the ClassifiersSucccessViewer class. This class process the classifiers results (HTML pages described in section 10.4.3) and create the previews according to configuration file. Using the example of the

configuration file on page 103 creates for you previews of configurations, classifiers and data. The page contains references to the stored HTML files, so it is easy to analyse the acquired data (see figure 10.3).

10.5 Summary

I have implemented the HHT as a part of the experimental software in Java. The algorithm can be used as a part of a more complex library, because it offers great modularity and almost every part of it could be easily replaced with another implementation. This feature makes the library very suitable for testing new methods to estimate additional extrema, stopping criteria, etc.

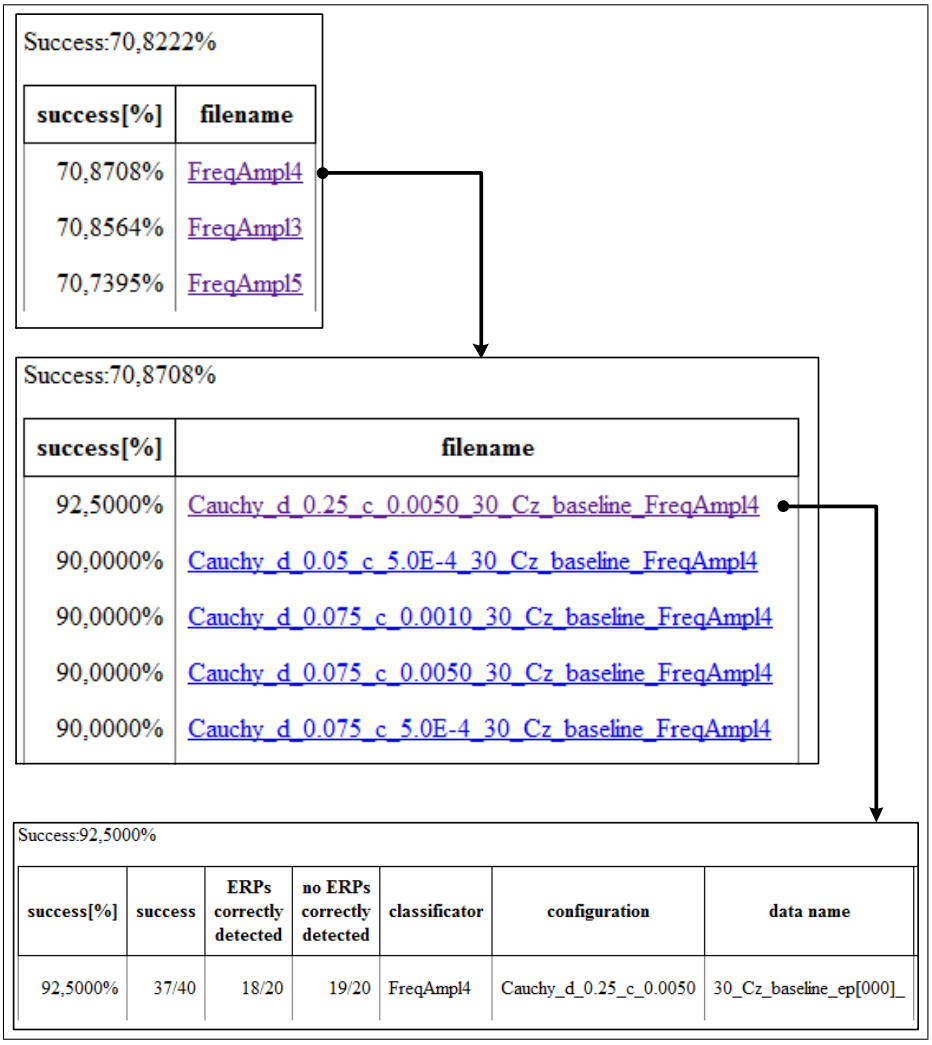

Figure 10.3: Success preview of used classifiers

Chapter 11

Results and Evaluation

11.1 Testing data

I have tested and evaluated my improvements of the HHT on real EEG data acquired in our laboratory. We have performed 20 Odd-Ball experiments (the experiment is described in 4.4) with 20 different subjects. We recorded the data with a device called BrainAmp. Its sampling frequency $1kHz$ was sufficient for our experiment.

I have decided to evaluate proposed methods by its ability to detect P3 wave. The tests were performed on 1s (1000 samples) long epochs. The original data consists of 30 epochs of the target stimulus responses and 90 epochs of the non-target stimulus responses, though subsets of 30 epochs were selected at random for the test.

Testing data contained a single trial (one epoch) and averaged epochs (up to 2nd, 5th, 10th, 20th and 30th). I have calculated several averages from data according to table 11.1. First 100 samples (0.1s) were included in the average used for the baseline correction.

11.2 Evaluation

My aim is to prove that proposed improvements of the EMD are beneficial for processing EEG signals. I chose three criteria:

Averaged Epochs	Created Averages Count
1	4
2	4
5	6
10	3
20	1
30	1

Table 11.1: Used averages count

- the average iterations count during the sifting process,
- the average count of created IMFs,
- the average classification reliability.

11.2.1 Average Iterations Count During the Sifting Process

The count of the sifting process iterations represents explicitly the speed of EMD. The aim of the criterion is to prove that proposed methods decrease the iterations count with same or better classification reliability (see section 11.2.3). The average iteration count is collected from all testing data including averaged epochs (2, 5, 10, 20, 30).

11.2.2 Average Count of Created IMFs

The number of created IMFs corresponds with the speed of sifting process. The more IMFs is created, the more iterations are neccessary to complete the sifting process. This criterion also reflects the "quality" of the decomposition. A reasonable number of IMFs has to be achieved with the decomposition. The average count of produced IMFs is calculated from all testing data including averaged epochs (2, 5, 10, 20, 30).

11.2.3 Average Classification Reliability

Automatic classification reliability represents directly the quality of decomposed IMFs. The more successful the classification is, the more IMF components retain enough physical sense of both amplitude and frequency. Therefore I designed simple classifier described in section 11.2.3.1.

Classifiers were applied on testing data including averaged epochs (2, 5, 10, 20, 30) and their results were summarized. The average classification reliability represents the evaluation.

11.2.3.1 Designed Classifier

The designed classifier has to reflect the variability of the P3 component. The P3 wave can vary in frequency, amplitude and latency among persons and even among trials.

I have assumed that the P3 wave has to occur between 150ms and 650ms after occurance of stimuli. Its frequency lies between 0.2Hz and 3Hz. I have designed a simple classifier which algorithm is as follows:

1. i = 0
2. Store i-th IMF into the imf_i
3. Calculate the mean frequency between 150ms and 650ms from the imf_i
4. If the calculated frequency mean fits into range $\langle 0.2Hz, 3Hz \rangle$ then continue with 4a, else continue with 5.
   (a) Calculate mean amplitude between 150ms and 650ms from the imf_i.
   (b) If the mean amplitude is greater than threshold then the P3 wave is detected.
5. If there is another IMF increase i and continue with 2, else there is no P3 and the algorithm ends.

I have used a set of three classifiers for the evaluation. They differ only in their mean amplitude threshold value. These three classifiers were selected from a

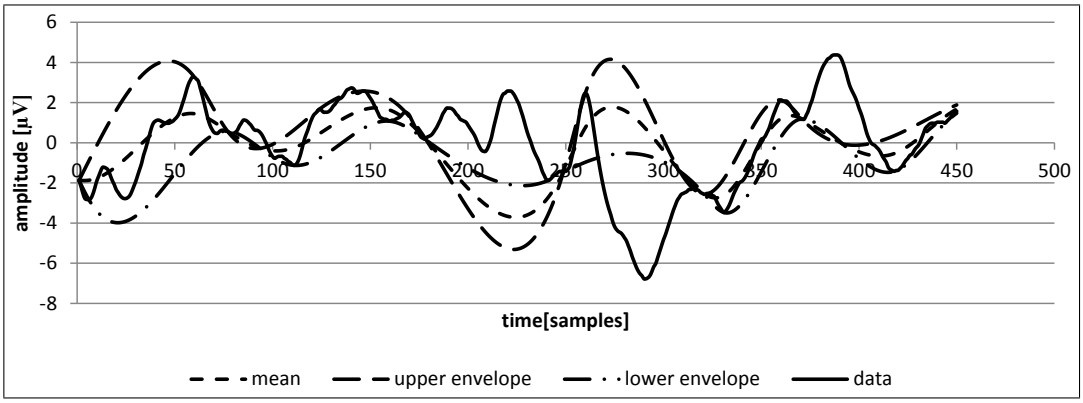

Figure 11.1: Envelopes created using Inflection Point Method.

greater set of classifiers according to their rate of successful recognition. Threshold values were set to 3.5, 3.25 and 3.0. Classifiers weren't adapted for testing data. Their settings reflects the description of the P3 wave.

11.3 Extrema Detection Methods Comparison

I have presented two methods for local extrema detection in the EEG signal. The Delta-difference method was designed as a result of requirements described in section 9.2.2. These requirements emerged from testing on real EEG data. As you can see at figure 11.1, when using the Inflection point method to detect extrema, the envelopes don't follow the trend of the signal. This drawback is fixed when local extrema are detected by Delta-difference method. In that case the envelopes surround the signal more tightly and follow its trend as intended (see figure 11.2).

The first-last method for estimating additional extrema point(see 9.1.1) was applied in both cases, stopping criteria were same. Therefore the decomposition is only affected by detected extrema, not by estimated extrema. It made possible to evaluate the speed of proposed method and its classification reliability. Results are presented in table 11.2;

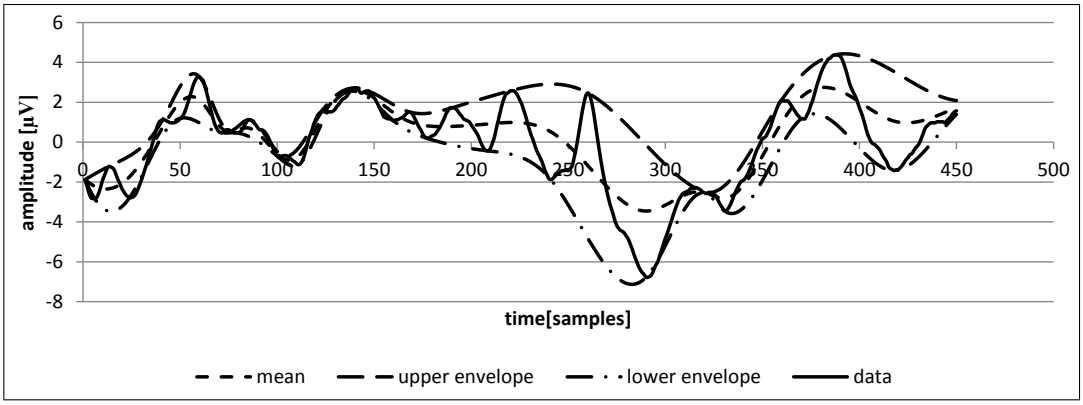

Figure 11.2: Envelopes created using Delta-difference method.

Used Stopping Criteria	Extrema Detection Method	Average Iterations Count	Average IMFs Count	Classification reliability [%]
Cauchy	Delta-difference method ($\delta = 0.05$)	278.5	13.2	63.8
	inflexion points	318.2	14.3	64.2
SD	Delta-difference method ($\delta = 0.05$)	943.8	19.0	63.0
	inflexion points	958.8	19.8	66.0

Table 11.2: Comparison of extrema detection methods.

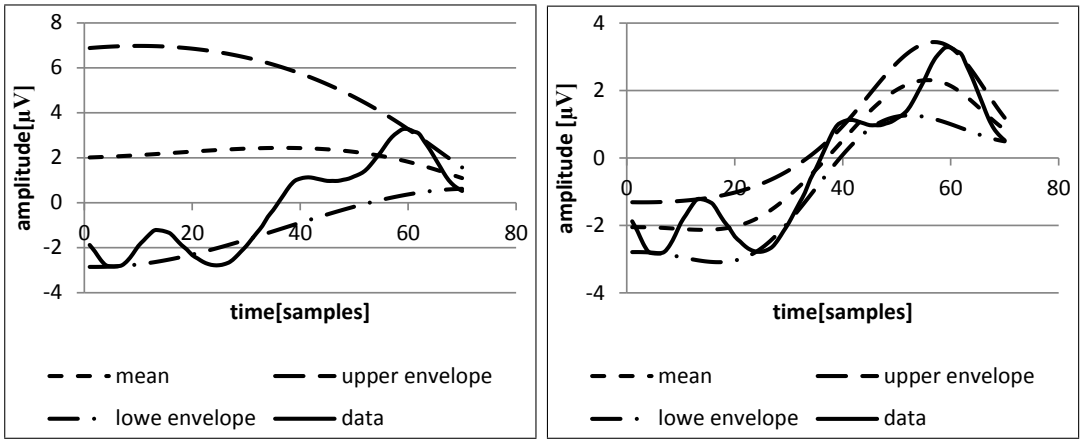

Figure 11.3: Comparison of Improved mirror method using Inflection point method (left) and Delta-difference method (right) for extrema detection.

The Delta-difference method speeded up the convergence of the EMD (see table 11.2), but it decreased the classification reliability insignificantly. This method was although proposed to detect "correct" local extrema, not to improve classification reliability. Correct extrema detection is essential for the trend of envelopes, moreover, it is crucial for estimating additional extrema with the improved mirror method (see figure 11.3). Without Delta-difference method it would be almost impossible to use the Improved mirror method for processing the EEG signal.

11.4 Additional Extrema Methods Comparison

As shown in section 11.3, the extrema detection is crucial for the modified mirror method. Therefore, I had to use the Delta-difference method to detect extrema while comparing the Modified mirror method and First-last.

The Improved mirror method is supposed to decompose the signal in a better way than the First-last method. I set δ to 0.05 while performing the comparison.

The Modified mirror method improves significantly the EMD for EEG processing. The method increased the classification reliability greatly. It means that decomposed IMFs correspond more to the signal. The method increases the speed

Used Stopping Criteria	Stopping Criteria Value	Additional Extrema Method	Average Iterations Count	Average IMFs Count	Classification reliability [%]
SD	10.0	Modified Mirror	162.4	5.7	77.0
Cauchy	0.001	Modified Mirror	42.4	5.1	74.7
Cauchy	0.001	First-last	278.5	13.2	63.8
SD	10.0	First-last	943.8	19.0	63.0

Table 11.3: Comparison of additional extrema methods.

Used Stopping Criteria	Stopping criteria value	δ [μV]	Average Iterations Count	Average IMFs Count	Classification reliability [%]
Cauchy	0.001	0.001	63.5	6.1	71.9
		0.01	53.2	5.7	74.1
		0.05	42.3	5.1	74.7
		0.1	38.4	4.8	72.8
SD	10	0.001	265.5	7.1	75.4
		0.05	162.4	5.7	77.0
		0.1	142.4	5.4	76.5

Table 11.4: Influence of δ parameter on speed and "quality" of EMD.

of the sifting process rapidly. The average iterations count is approximately six times smaller when the Modified mirror method estimates additional extrema, instead of First-last method. Results are presented in table 11.3.

11.5 Influence of the δ Parameter on the EMD

The value of δ parameter expresses the tolerance for small amplitude fluctuations(see section 9.2.2). These fluctuations don't bear important information about processed signal. Therefore they can be considered as a noise or interference. The *delta* parameter of Delta-difference method makes it possible to ignore small amplitude fluctuations. Several tests were performed with different values of δ parameter(see table 11.4).

The δ parameter affects the speed of EMD(table 11.4). The greater value of δ parameter makes the lower average iterations count drop, because the EMD

Stopping Criteria	Max Iterations Count
Cauchy	500
SD	700

Table 11.5: Max iterations count during sifting process.

Cauchy	0.0001	0.0005	0.001	0.005	0.01	0.05	0.1	0.5
SD	0.1	0.2	0.5	1	2	5	10	20
	30	50						
δ	0.001	0.005	0.01	0.05	0.075	0.1	0.25	0.75
	1.0	1.25	1.5					

Table 11.6: Values for EMD configurations.

ignores more details while creating envelopes. The δ parameter influences also the classification reliability (see table 11.4). Therefore, I have tested several configurations of stopping criteria and the δ parameter. Results can be seen in section 11.6.

11.6 Recommended Configurations for the EMD

The aim of this section is to recommend some configurations of EMD. They could serve as starting point for other researchers.

I had to established an additional criterion for testing proposed configurations - the *maximum iterations count*. This criterion represents the stability of the sifting process (its convergence). Configurations exceeding the maximum iterations count even in single case, are considered as unsuitable for ERP detection. Criterion values were established empirically with regards of time consumption (see table 11.5). Additional 200 iterations were used with the Standard Deviation stopping criterion (well known for its problematic convergence[9]).

Testing configurations were created for EMD stopping criteria (Cauchy convergence test and Standard deviation). I have proposed a set of stopping criteria values for each stopping criterion (table 11.6). The values of δ parameter can be seen in the table 11.6.

Stopping criteria value	δ [μV]	Average Iterations Count	Average IMFs Count	Classification reliability [%]	Average classification reliability
0.005	0.25	15.2	3.6	92.5	71.1
0.001	0.075	35.9	4.4	90.0	72.7
0.0005	0.075	42.0	4.4	90.0	73.9
0.0005	0.05	49.2	4.7	90.0	73.7
0.0001	0.1	97.1	4.7	90.0	74.0

Table 11.7: Suitable configurations for 30 averaged epochs.

Suitable EMD configurations were selected for 10, 20 and 30 averaged epochs (tables 11.9, 11.8 and 11.7). Configurations were selected according to following criteria:

1. classification reliability with corresponding averaged epochs
2. averaged classification reliability

The *NA* signifies that the averaged classification reliability couldn't be determined, because the method didn't provide its result before reaching iterations limit for all testing data.

The standard deviation as stopping criterion failed in my test. It exceeded the maximum iteration limit with some testing data samples in almost every case. Therefore, I don't recommend it to be used as the stopping criterion, because of its convergence instability. The Cauchy convergence test seems to be a better choice.

The most successful configurations are selected in table 11.10. The configurations offer best average classification reliability for all testing data.

11.7 Comparison HHT with WT and MP

Best achieved results from HHT, WT and MP were compared in table 11.11. When the HHT was performed with the modified EMD, it achieved similar results

Stopping criteria value	δ [μV]	Average Iterations Count	Average IMFs Count	Classification reliability [%]	Average classification reliability
0.005	1.5	8.1	2.3	85.0	73.3
0.0001	1.5	41.5	2.5	85.0	72.2
0.0001	0.75	50.2	3.3	85.0	71.6
0.001	1.5	14.0	2.4	85.0	71.4
0.0005	0.5	97.1	4.7	85.0	71.2

Table 11.8: Suitable configurations for 20 averaged epochs.

Stopping criteria value	δ [μV]	Average Iterations Count	Average IMFs Count	Classification reliability [%]	Average classification reliability
0.0001	0.05	64.4	3.6	78.3	72.5
0.0005	0.075	50.6	4.9	78.3	73.9
0.0001	0.01	150.3	6.0	79.2	NA
0.0005	0.25	41.5	4.2	77.5	71.6
0.0005	0.05	47.0	4.7	77.5	73.1

Table 11.9: Suitable configurations for 10 averaged epochs.

Stopping criteria value	δ [μV]	Average Iterations Count	Average IMFs Count	Classification reliability [%]
0.0001	0.1	97.1	4.7	74.0
0.0005	0.075	50.6	4.9	73.9
0.0005	0.05	49.2	4.7	73.7
0.005	1.5	8.1	2.3	73.3
0.0005	0.1	47.0	4.7	73.1

Table 11.10: Most reliable settings

Averaged Epochs	Method	correctly detected		false positive detection	false negative detection
		count	[%]		
10	MP	31	77.5	7	2
	CWT	33	82.0	6	1
	HHT	31	77.5	2	7
	oHHT	24	60.0	7	7
20	MP	34	85.0	3	3
	CWT	34	85.0	3	3
	HHT	34	85.0	3	3
	oHHT	26	65.0	5	7
30	MP	36	90.0	2	2
	CWT	37	92.5	2	1
	HHT	37	92.5	1	2
	oHHT	28	70.0	3	9

Table 11.11: Comparison of MP, CWT and modified and original HHT

as the CWT and modified MP algorithm. It is significant improvement, because the average classification reliability of the originally proposed EMD was lower.

My modifications of the HHT (and the modified mirror method) improved the sifting process. The resulting IMFs correspond more to the original EEG signal. Therefore, the classification reliability has increased. After modification, the HHT achieves same results as the MP and CWT in the ERP detection task.

Chapter 12

Conclusion

In my PhD thesis, I have described the Hilbert-Huang transform in detail. My aim was to use HHT for ERP detection, because it offers precise time resolution. But during my tests on real EEG data acquired in our laboratory, I had encountered several drawbacks in EMD. Therefore, I had to focused on the Empirical Mode decomposition, which is the core of HHT.

Described drawbacks were related to the envelope calculation process (part of EMD). Created envelopes have to cover the whole input signal (see section 8.1). Points of envelopes are derived from local maxima and minima, therefore, it is necessary to add additional points in order to create complete envelope covering the signal from its first to its last sample. But additional extrema points have to be chosen carefully. The envelopes could suffer from overshoot effect if additional points were chosen incorrectly. The Mirror method(section 8.1.1) and Slope-based method (section 8.1.2) were already designed for this specific task. But they weren't designed for the EEG signal processing. Both of them are not able to estimate additional extrema points, when the edge of the signal contains time-short components of significantly higher frequency, such as like artifacts (see section 8.1.3). Therefore, I have designed the Improved mirror method (section 9.1.3), which makes it possible to create complete envelopes, even when the edges of the signal contain artifacts.

For Improved mirror method and envelopes creation process, the local extrema detection is the essential part. When processing the discrete signal, not every inflection point can be considered as a local extremum and not every local

extremum has its highest/lowest value in its surroundings. To detect more suitable local extrema, I have designed Delta-difference method (see section 9.2.2). It makes possible to ignore small amplitude fluctuation in the signal and to detect local extrema not having their highest/lowest amplitude value in their surroundings. With Delta-difference method, the envelopes surround the signal more tightly and follow its trend (section 11.3).

I have implemented the modified mirror method and Delta-difference method. Both methods were tested on data (section 11.1) acquired in our laboratory during odd-ball experiments. These two improvements significantly increase the speed of EMD (section 11.4). Average iterations count during the sifting process is approximately six times lower. Moreover, designed modifications increase classification reliability because created IMFs corresponds to the decomposed signal with their more physical properties (amplitude, frequency). Now the classification reliability of the ERP detection by HHT is comparable with Continuous wavelet transform and with Modified Matching pursuit [25] (introduced in section 6.2.2). Comparison of CWT, MP and HHT can be seen in section 11.7. During the testing, I have selected some configurations of EMD to be suitable for ERP detection and EEG signal processing (section 11.6).

My achievements in short:

- implementation of the HHT in Java
- proposition of the new method for local extrema detection in a discrete signal
- proposition of the new method for additional extrema estimation in order to minimize the overshoot effect during envelopes calculation
- implementation of proposed modifications
- verification of proposed modifications on real EEG signals
- HHT with my modifications is far more suitable for ERP detection than the original HHT

12.1 Future Work

My work on the HHT could be divided into two parts. First, I have implemented the HHT as a part of the testing software written in Java. I suggest to include the HHT module into an open source library to offer the HHT algorithm to other researches who use Java for their projects. Also, I made future improvement and refactoring of the source code easier by positioning standard eclipse marks. I recommend to conduct subsequent performance tests and identify weak points of my implementation.

The second part of my contribution is focused on improving the EMD and classification. My idea is to design an additional stopping criteria which could be combined with Standard deviation and Cauchy convergence stopping criteria. The newly designed criteria is based on tests conducted on the EEG data acquired in our laboratory. The implementation of the sifting process makes it easy to change stopping criteria and combine them together. New criteria could be, for example, the mean or standard deviation of the mean curve calculated from envelopes.

I would gladly encourage the further development of new methods for estimating additional extrema, because, as I have shown, suitably positioned additional extrema could significantly improve the speed and reliability of the EMD. It should be possible to use a neural networks to estimate additional extrema [41].

Also, it is necessary to establish more features for classification than the medium amplitude and frequency of the P3 wave. New classification features will increase the classification reliability and lead to a design of more suitable classifier. This features could be based on earlier ERP waves like N1 or P2. The possibility of combining the time-locked spectral averaging (section 5.2.2) with the HHT should be explored in order to acquire these features.

12.1.1 Future Work Summary

1. refactoring of the current implementation of HHT
2. performance tests and optimization of current implementation
3. release the HHT open-source library

4. design new stopping criteria and test their combination
5. test neural networks and their ability to estimate additional extrema
6. focus on the classification feature extraction using time-locked spectral averaging
7. design more suitable classifiers

References

[1] P. S. Addison. *The Illustrated Wavelet Transform Handbook,*. IOP Publishing, 2002. 27, 29

[2] Adriano O. Andrate, Peter Kyberd, and Slawomir J. Nasuto. The aplication of the hilbert spectrum to the analysis of electromyographic signals. **9**[178]:2176–2193, 2008. 40

[3] S. R. Bernadis. Eeg artifacts. http://emedicine.medscape.com/article/1140247-overview, 2008. v, 7, 8, 9, 10, 21

[4] Junsheng Cheng, Dejie Yu, and Yu Yang. Application of support vector regression machines to the processing of end effects of hilbert-huang transform. *Mechanical Systems and Signal Processing*, **21**[3]:1197 – 1211, 2007. 42

[5] F. Cong, T. Sipola, X. Xu, T. Huttunen-Scott, H. Lyytinen, and T. Ristaniemi. Concatenated trial based hilbert-huang transformation on event-related potentials. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1 –5, july 2010. 38, 39

[6] Marcus Dätig and Torsten Schlurmann. Performance and limitations of the hilbert-huang transformation (hht) with an application to irregular water waves. **31**, 2004. 42, 44, 46

[7] P. Durka. *Matching Pursuit and Unification in EEG Analysis*. ARTECH HOUSE, INC., Nordwood, 2007. 1, 32

[8] Rui Fonseca-Pinto. *A New Tool for Nonstationary and Nonlinear Signals: The Hilbert-Huang Transform in Biomedical Applications*. InTech, 2011. 40, 41

[9] N Huang and Nii O. Attoh-Okine. *The Hilbert-Huang Transform in Engineering*. CRC Press, 2005. 37, 38, 81

[10] N. E. Huang and et al. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. **Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences**[454], 1998. 2, 36, 38, 39, 40, 41, 56

[11] Norden E. Huang, Zheng Shen, Steven R. Long, Manli C. Wu, Hsing H. Shih, Quanan Zheng, Nai-Chyuan Yen, Chi C. Tung, and Henry H. Liu. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, **454**[1971]:903–995, March 1998. 36, 40

[12] Wei Huang, Zheng Shen, Norden E. Huang, and Yuan Cheng Fung. Engineering analysis of biological variables: An example of blood pressure over 1 day. *Proceedings of the National Academy of Sciences*, **95**[9]:4816–4821, 1998. 40

[13] Boris Kotchoubey and Simone Lang. Event-related potentials in an auditory semantic oddball task in humans. *Neuroscience Letters*, **310**[2-3]:93 – 96, 2001. 17

[14] Xiaoli Li, Duan Li, Zhenhu Liang, Logan J. Voss, and Jamie W. Sleigh. Analysis of depth of anesthesia with hilbert-huang spectral entropy. *Clinical Neurophysiology*, **119**[11]:2465 – 2475, 2008. 21, 37

[15] H Liang, S Bresser, and R Desimone. Empirical mode decomposition: a method for analyzing neural data. *Neurocomputing*, **65**:801–807, 2005. 37

[16] H Liang, Z Lin, and RW McCallum. Artifact reduction in electrogastrogram based on the empirical model decomposition method. **38**:35–41, 2000. 21

[17] Chin-Feng Lin, Shan-Wen Yeh, Yu-Yi Chen, Tsung-Ii Peng, Jung-Hua Wang, and Shun-Hsyung Chang. A hht-based time frequency analysis scheme for clinical alcoholic eeg signals. In *Proceedings of the 9th WSEAS international conference on Multimedia systems & signal processing*, pages 53–58, Stevens Point, Wisconsin, USA, 2009. World Scientific and Engineering Academy and Society (WSEAS). 37, 40

[18] R. Liu. Empirical mode decomposition: A useful technique for neuroscience?, 2002. 36

[19] S.J. Luck. *An Introduction to the Event-Related Potential Technique*. The MIT Press, Cambridge,, 2005. 14, 15, 16, 17, 18, 19

[20] Steven J. Luck. *Ten Simple Rules for Designing and Interpreting ERP Experiments*. The MIT Press, 2004. 13

[21] Mathworks. Signal processing toolbox - hilbert, 2008. 38

[22] P. Mautner and R. Mouček. Using matching pursuit method to detection of erp components. In *Proceedings of the 9th Conference Kognice a umělý život*, pages 201–205, Bezručovo náměstí 13, Opava, Czech Republic, 2009. Slezská univerzita v Opavě, Filozoficko-přírodovědecká fakulta v Opavě. 31

[23] J. B. Ochoa. Eeg signal classification for brain computer interface applications. `http://bci.epfl.ch/publications/baztarricadiplomaproject.pdf`, 2002. 6, 7

[24] Gbadebo Oladosu. Identifying the oil price-macroeconomy relationship: An empirical mode decomposition analysis of us data. *Energy Policy*, **37**[12]:5417 – 5426, 2009. 41

[25] T. Řondík. Using matching pursuit algorithm with its own dictionary for erp in eeg signal detection. In *Proceedings of the 10th Conference Kognice a umělý život*, pages 329–332, Bezručovo náměstí 13, Opava, Czech Republic, 2010. Slezská univerzita v Opavě, Filozoficko-přírodovědecká fakulta v Opavě. v, 31, 32, 33, 86

[26] M. C. Peel, G. E. Amirthanathan, G. G. S. Pegram, T. A. Mcmahon, and F. H. S. Chiew. Issues with the Application of Empirical Mode Decomposition Analysis. In *MODSIM 2005 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand, December, 2005.*, pages 170–176, 2005. 54

[27] A. Procházka and E. Hošťálková. Zpracování biomedicínských signálů a obrazů pomocí wavelet transformace. *Automatizace: odborný časopis pro automatizaci, měření a inženýrskou informatiku*, **6**:397–401, 2007. 27, 28

[28] Liangsheng Qu and Fangji Wu. An improved method for restraining the end effect in empirical mode decomposition and its applications to the fault diagnosis of large rotating machinery. *Journal of Sound and Vibration*, **314**:586–602, 2008. vi, 37, 41, 44, 45, 46, 47

[29] R.T. Rato, M.D. Ortigueira, and A.G. Batista. On the hht, its problems, and some solutions. *Mechanical Systems and Signal Processing*, **22**[6]:1374 – 1394, 2008. Special Issue: Mechatronics. 42, 52, 54

[30] Gabriel Rilling, Patrick Flandrin, and Paulo Gon¸calv‘es. On empirical mode decomposition and its algorithms. In *IEEEEURASIP workshop on nonlinear signal and image processing NSIP03 Grado I*, **3**, pages 8–11, 2003. 44, 52, 54

[31] Jr. S. Lawrence Marple. Computing the discrete-time "analytic" signal via fft, 1999. 38, 39

[32] S. Sanei and J. Chambers. *EEG Signal Processing*. John Wiley and Sons, New York,, 2007. 6, 7, 11, 12, 18

[33] Jonathan S Smith. The local mean decomposition and its application to eeg perception data. *Journal of The Royal Society Interface*, **2**[5]:443–454, 2005. 41

[34] P. Soukal. *Methods for automatic detection of ERP components*. Master's thesis, West Bohemia University, Faculty of Applied Sciences, Pilsen,, 2010. v, 27, 28, 29

[35] A.K. Tafreshi, A.M. Nasrabadi, and A.H. Omidvarnia. Epileptic seizure detection using empirical mode decomposition. In *Signal Processing and Information Technology, 2008. ISSPIT 2008. IEEE International Symposium on*, pages 238 –242, dec. 2008. 41

[36] Yin Qing Tan, F. Ibrahim, M. Moghavvemi, and J. Ibrahim. Development of an eeg amplifier for brain-computer-interface. In Ratko Magjarevic, Fatimah Ibrahim, Noor Azuan Abu Osman, Juliana Usman, and Nahrizul Adib Kadri, editors, *3rd Kuala Lumpur International Conference on Biomedical Engineering 2006*, **15** of *IFMBE Proceedings*, pages 378–382. Springer Berlin Heidelberg, 2007. 6

[37] M. Teplan. Fundamentals of eeg measurement. *Measurement Science review*, **2**, 2002. 4, 5, 6

[38] C.A. Valens. Really friendly guide to wavelets. http://pagesperso-orange.fr/polyvalens/clemens/wavelets/wavelets.html, 2002. 27

[39] J.J. Vidal. Real-time detection of brain events in eeg. *Proceedings of the IEEE*, **65**[5]:633 – 641, may 1977. 12

[40] Zhai Xueming, Li Ling, Zhu Yongli, and Li Lamei. The application of a new process method for end effects of emd in the insulator state diagnosis. In *Information Technology and Applications, 2009. IFITA '09. International Forum on*, **2**, pages 764 –767, may 2009. 41, 42

[41] Jian Xun and Shaoze Yan. A revised hilbert-huang transformation based on the neural networks and its application in vibration signal analysis of a

deployable structure. *Mechanical Systems and Signal Processing*, **22**[7]:1705 – 1723, 2008. 38, 42, 49, 87

[42] Yue Min Zhu, Francoise Peyrin, and Robert Goutte. The wigner-ville transform - description of a new tool for signal and image processing. *Annales des Telecommunications*, 1987. 32

Author's Publications

[43] J. Ciniburk. Emd overshoot efect in erp detection: Erp detection related specifics of the empirical mode decomposition in eeg analysis. In *ICEIS (5)*, **5 HCI**, pages 238–241. SciTePress, 2010.

[44] Jindřich Ciniburk. Suitability of huang hilbert transformation for erp detection. In *Proceedings of the 9th international PhD workshop on systems and control*, 2008.

[45] Jindřich Ciniburk. Weka a evokované potenciály. In *Informatika v škole a praxi*, 2008.

[46] Jindřich Ciniburk. Možnosti využití hilbert-huangovy transformace pro detekci evokovaných potenciálů. In *Kognice a umělý život IX*, 2009.

[47] Jindřich Ciniburk. Možnosti omezení překmitů v průběhu hilbert-huangovy tranformace při detekci evokovaných potenciálů. In *Kognice a umělý život X*, 2010.

[48] Jindřich Ciniburk, Martin Hošna, and Pavel Mautner. Segmentace eeg signálů. In *Informatika v škole a praxi*, pages 12–15, 2007.

[49] Jindřich Ciniburk, Martin Hošna, Pavel Mautner, Václav Matoušek, and Roman Mouček. Eeg signal segmentation. In *TRANSTEC*, pages 270–274, 2007.

[50] Jindřich Ciniburk, Martin Hošna, Pavel Mautner, Václav Matoušek, and Roman Mouček. *Segmentation of EEG Signal*, pages 43–62. Neural Network World, 2007.

[51] Jindřich Ciniburk and Pavel Mautner. Segmentation eeg with neural network. In *Proceedings of the 8th international PhD workshop on systems and control a young generation viewpoint*, 2007.

[52] Martin Hošna, Pavel Mautner, and Jindřich Ciniburk. Detekce a odstranění artefaktů v eeg signálu. In *Informatika v škole a praxi*, 2007.

[53] Tomáš Řondík, Jindřich Ciniburk, Pavel Mautner, and Roman Mouček. Erp components detection using wavelet transform and matching pursuit algorithm. In *Proceedings of 3rd Driver Car Interaction Interface 2010 Conference*, 2010. 24

Appendix A

Listing 1: Example of configuration file

```
<?xml
version="1.0" encoding="UTF-8"?> <beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean  id="emd" class="hht.emd.EmpiricalModeDecomposition" >
        <constructor-arg  ref="sifter"/>
        <constructor-arg ref="extremaLocator"/>
        <property name="counter" ref="counter"/>
</bean>
<bean id="counter" class="hht.emd.sifting.IterationCounter">
        <property name="max" value="3"/>
</bean>
<bean id="extremaLocator"
        class="hht.emd.sifting.extrema.locators.LocateLocalExtremesFacade" >
        <constructor-arg ref="maxLocator" />
        <constructor-arg ref="minLocator" />
</bean>
<bean id="minLocator"
        class="hht.emd.sifting.extrema.locators.ExtremeLocator">
        <constructor-arg ref="minHelper"/>
</bean>
<bean id="maxLocator"
        class="hht.emd.sifting.extrema.locators.ExtremeLocator">
        <constructor-arg ref="maxHelper"/>
</bean>
<bean id="minHelper"
        class="hht.emd.sifting.extrema.locators.MinimumLocatorHelper">
        <property name="delta" value="0.05"/>
</bean>
<bean id="maxHelper"
        class="hht.emd.sifting.extrema.locators.MaximumLocatorHelper">
        <property name="delta" value="0.05"/>
</bean>
<bean id="sifter" class="hht.emd.sifting.Sifter">
        <constructor-arg ref="Or1"/>
        <constructor-arg ref="EndPointMirrorEstimator2"/>
        <constructor-arg ref="extremaLocator"/>
</bean>
<bean id="EndPointMirrorEstimator2"
        class="hht.emd.sifting.extrema.estimators.EndPointMirror2"/>

<bean id="Or1" class="hht.emd.sifting.stoppingCriteria.Or">
        <property name="criteria">
                <list>
                        <ref bean="CauchyConvergence" />
                </list>
        </property>
```

```
</bean>
<bean id="CauchyConvergence"
        class="hht.emd.sifting.stoppingCriteria.CauchyConvergence">
        <property name="deviationThreshold" value="0.001" />
</bean>
<bean id="MonotonicFunction"
        class="hht.emd.sifting.stoppingCriteria.MonotonicFunction">
        <property name="diferenceThreshold" value="0.005" />
</bean>
<bean id="ResultsConfigs" class="data.ResultsConfigs">
        <property name="genPdfFromImfs" value="false"/>
        <property name="genPdfFromImfsMaps" value="false"/>
        <property name="genImfsAllMapPdf" value="false"/>
        <property name="genImfMapToSignalPdf" value="false"/>
</bean>
</beans>
```

Listing 2: Example of the log4j property file

```
log4j.debug=false
log4j.rootLogger=INFO, EMD

log4j.logger.hht.aspects.SiftingConvergenceLogger=INFO, SCL
log4j.logger.hht.aspects.EnvelopesLogger=DEBUG, ENVELOPES
log4j.logger.testing.classificators=TRACE, CLASS
log4j.logger.EmdIterationsLogger = INFO, ItCount

#CLASS
log4j.appender.CLASS=org.apache.log4j.FileAppender
log4j.appender.CLASS.File=${class.log.filename}
log4j.appender.CLASS.append=false
log4j.appender.CLASS.layout=org.apache.log4j.PatternLayout
log4j.appender.CLASS.layout.ConversionPattern=%5p%6.6r[%t]%x - %m%n

#O Stdout
log4j.appender.O=org.apache.log4j.ConsoleAppender
log4j.appender.O.layout=org.apache.log4j.PatternLayout
log4j.appender.O.layout.ConversionPattern=%5p%6.6r[%t]%x - %m%n

# EMD
log4j.appender.EMD=org.apache.log4j.FileAppender
log4j.appender.EMD.File=${emd.log.filename}
log4j.appender.EMD.append=false
log4j.appender.EMD.layout=org.apache.log4j.PatternLayout
log4j.appender.EMD.layout.ConversionPattern=%5p%6.6r[%t]%x - %m%n

#SCL
log4j.appender.SCL=org.apache.log4j.FileAppender
log4j.appender.SCL.File=${sifting.log.filename}
log4j.appender.SCL.append=false
log4j.appender.SCL.layout=org.apache.log4j.PatternLayout
log4j.appender.SCL.layout.ConversionPattern=%5p%6.6r[%t]%x - %m%n

#ENVELOPES
#log4j.appender.ENVELOPES=data.EnvelopeAppender
log4j.appender.ENVELOPES=data.log.EnvelopesFileAppender
log4j.appender.ENVELOPES.path=${imfs.log.path}
log4j.appender.ENVELOPES.append=false
log4j.appender.ENVELOPES.separateFiles=true
log4j.appender.ENVELOPES.fileNamePrefix=matrix
log4j.appender.ENVELOPES.layout=org.apache.log4j.PatternLayout
log4j.appender.ENVELOPES.layout.ConversionPattern=%m

#Iterations count
log4j.appender.ItCount=org.apache.log4j.FileAppender
log4j.appender.ItCount.File=${imfs.itCount.filename}
log4j.appender.ItCount.append=false
```

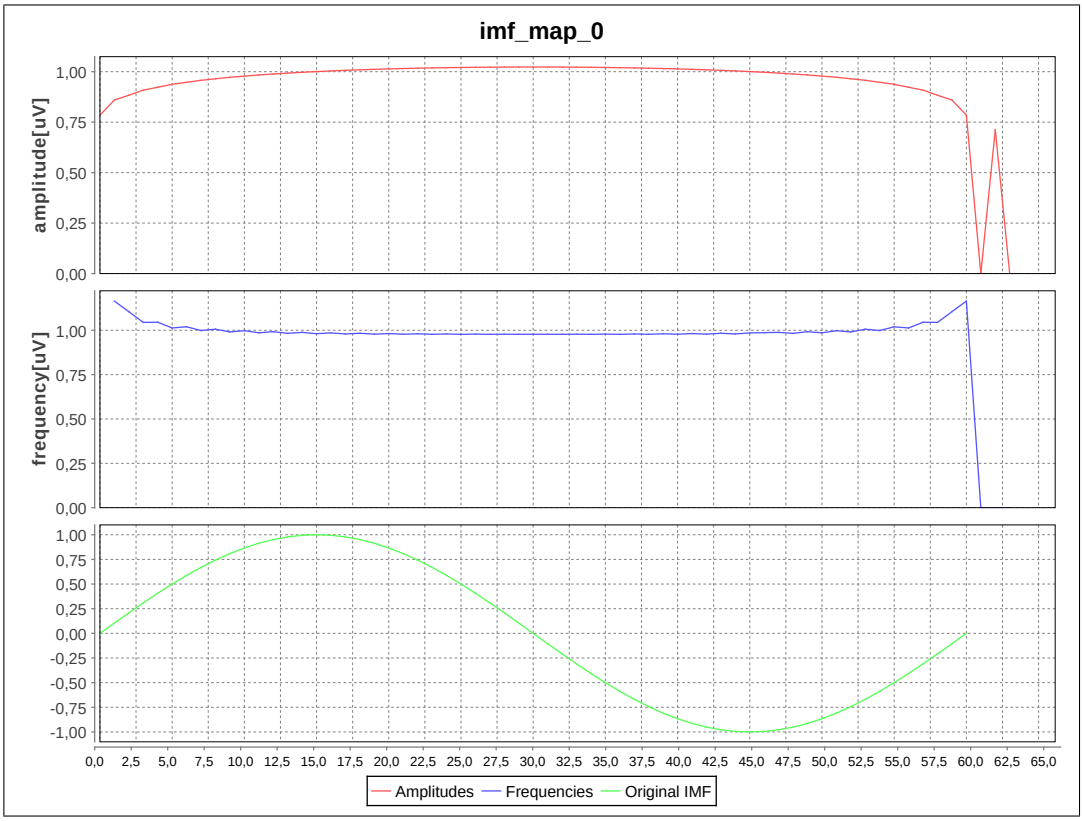

Figure 1: The visualization of HT results - single IMF.

```
log4j.appender.ItCount.layout=org.apache.log4j.PatternLayout
log4j.appender.ItCount.layout.ConversionPattern=%m%n

log4j.logger.hht.emd.sifting.Sifter=TRACE
log4j.logger.hht.emd.sifting.stoppingCriteria=TRACE
log4j.logger.hht.emd.sifting.Enveloper=TRACE
log4j.logger.hht.emd.sifting.extrema.Extremes=TRACE
log4j.logger.hht.emd.sifting.extrema.estimators=TRACE
log4j.logger.hht.emd.EmpiricalModeDecomposition=TRACE
log4j.logger.hht.emd.sifting.extrema.estimators.EndPointMirror=TRACE
```

Listing 3: Configuration file for IterationCounter class

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:util="http://www.springframework.org/schema/util"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

   xsi:schemaLocation="
    http://www.springframework.org/schema/beans
```

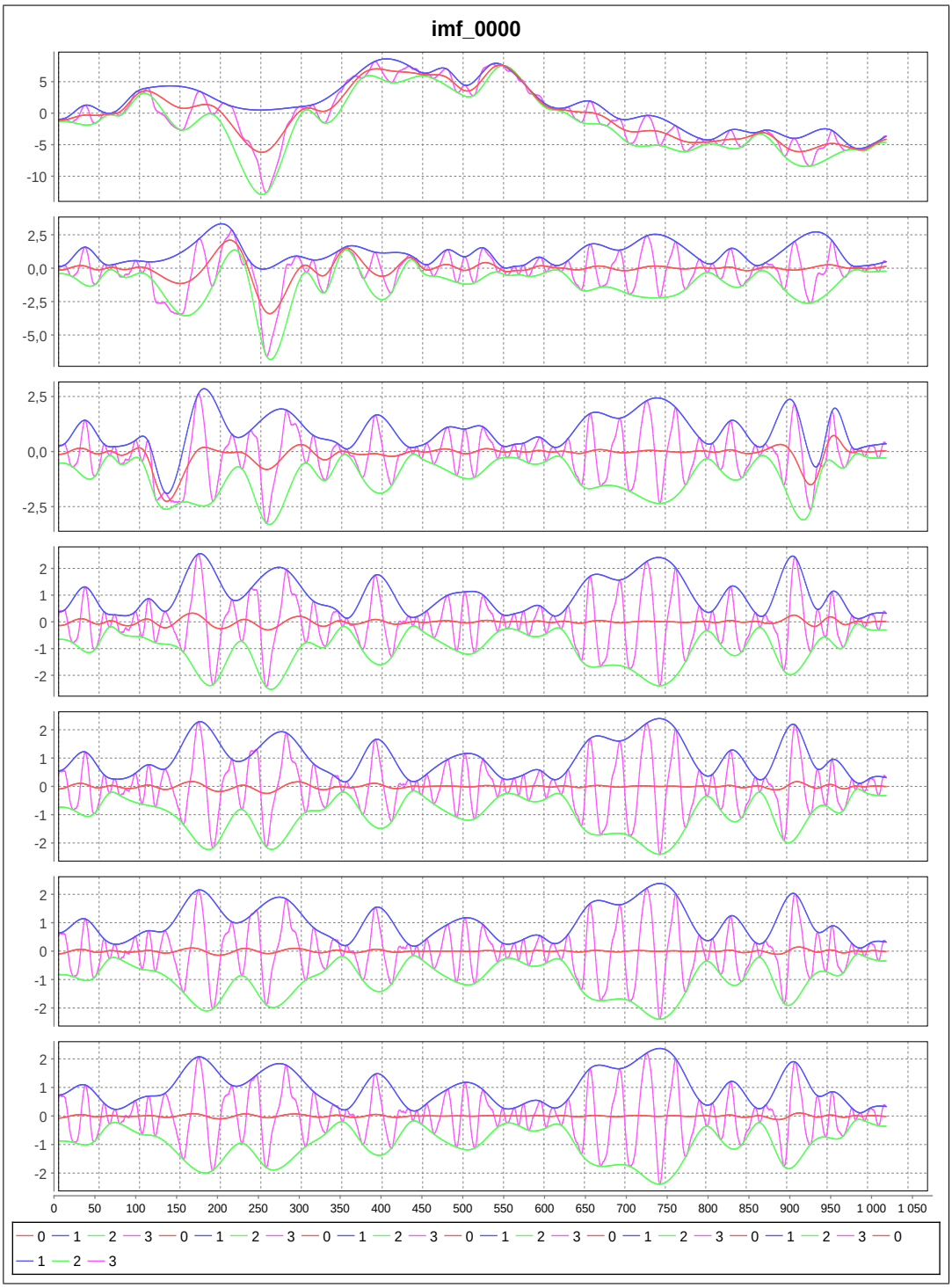

Figure 2: An example of sifting process iterations.

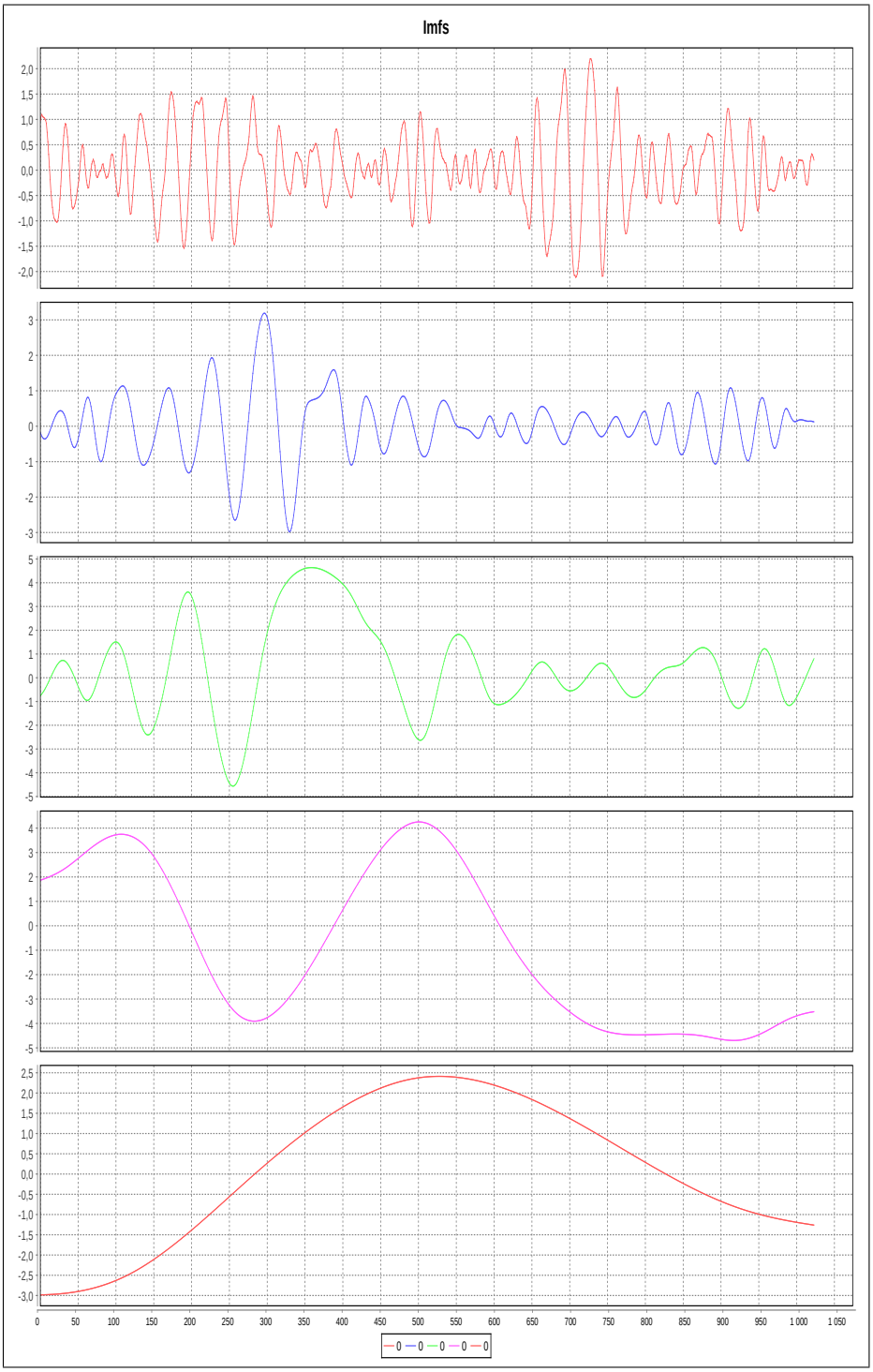

Figure 3: An example of decomposed IMFs of the EEG signal.

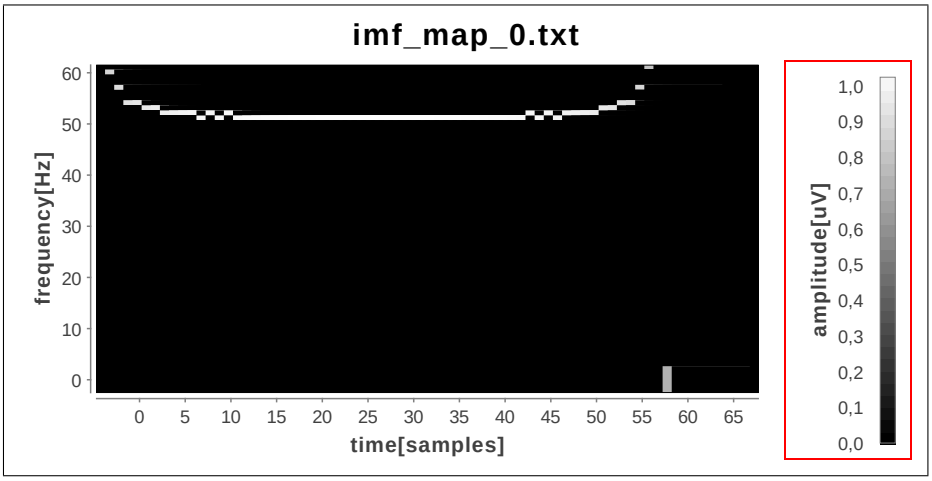

Figure 4: Time-Frequency map

```
____http://www.springframework.org/schema/beans/spring-beans.xsd
____http://www.springframework.org/schema/util
____http://www.springframework.org/schema/util/spring-util-2.0.xsd">

  <!-- list of directories with processed data-->
  <util:list id="directories">
    <value>i:/hhtResultsTesting/Cauchy_d_0.0010_c_0.0010</value>
        <value>i:/hhtResultsTesting/Cauchy_d_0.0010_c_0.0050</value>
  </util:list>

  <bean id="baseline" class="testing.iterationCount.IterationCounter">
        <property name="cfgDirectories" ref="directories" />
        <property name="dataDirectoriesPatterns">
          <list>
                <value>\d+_Cz_baseline_(ep\[\d{3}\]_)?(target|nonTarget)</value>
                <value>\d+_Cz_(ep\[\d{3}\]_)?(target|nonTarget)</value>
          </list>
        </property>
        <property name="outputFile" value="baselineVsNotBaseline.html"/>
  </bean>

  <bean id="averages" class="testing.iterationCount.IterationCounter">
        <property name="cfgDirectories" ref="directories" />
        <property name="dataDirectoriesPatterns">
          <list>
                <value>10_Cz_baseline_(ep\[\d{3}\]_)?(target|nonTarget)</value>
                <value>20_Cz_baseline_(ep\[\d{3}\]_)?(target|nonTarget)</value>
                <value>30_Cz_baseline_(ep\[\d{3}\]_)?(target|nonTarget)</value>
                <value>05_Cz_baseline_(ep\[\d{3}\]_)?(target|nonTarget)</value>
                <value>02_Cz_baseline_(ep\[\d{3}\]_)?(target|nonTarget)</value>
                <value>01_Cz_(ep\[\d{3}\]_)?(target|nonTarget)</value>
          </list>
        </property>
        <property name="outputFile" value="averages.html"/>
  </bean>

  <util:list id="counters">
        <ref bean="baseline"/>
        <ref bean="averages"/>
  </util:list>
```

```
</beans>
```

Listing 4: Configuration file for HhtDataRunner class

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:util="http://www.springframework.org/schema/util"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
___http://www.springframework.org/schema/beans
___http://www.springframework.org/schema/beans/spring-beans.xsd
___http://www.springframework.org/schema/util
___http://www.springframework.org/schema/util/spring-util-2.0.xsd">

        <!-- import the list of the used classifiers -->
        <import resource="classifiers.xml"/>

  <bean id="RunClassificationAll"
     class="testing.classificators.RunClassificationAll">
    <property name="dataDirectories" ref="directories"/>
    <property name="patterns" ref="patterns"/>
    <property name="classifiers" ref="classifiers"/>
        <property name="resultsPath" value="data//results//classifiers//"/>
        <property name="templateFilename" value="configs//classification//htmlTemplate.stg"/>
        <property name="logsDirName" value="classification"/>
  </bean>

  <util:list id="classifiers">
        <ref bean="classifier1"/>
        <ref bean="classifier2"/>
        <ref bean="classifier3"/>
  </util:list>

  <util:list id="directories">
        <value>i://hhtResults//CauchyCfg6_3//</value>
        <value>i://hhtResults//SDCfg4_3//</value>
  </util:list>

  <util:list id="patterns">
        <value>10_Cz_baseline_(ep\[\d{3}\]_)(target|nonTarget)</value>
        <value>10_Cz_(ep\[\d{3}\]_)(target|nonTarget)</value>
        <value>20_Cz_baseline_(ep\[\d{3}\]_)?(target|nonTarget)</value>
        <value>20_Cz_(ep\[\d{3}\]_)?(target|nonTarget)</value>
        <value>30_Cz_baseline_(ep\[\d{3}\]_)?(target|nonTarget)</value>
        <value>30_Cz_(ep\[\d{3}\]_)?(target|nonTarget)</value>
  </util:list>
</beans>
```

Listing 5: Configuration file for ClassifiersSucessViewwer class

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
                xmlns:util="http://www.springframework.org/schema/util"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xsi:schemaLocation="
________________http://www.springframework.org/schema/beans
________________http://www.springframework.org/schema/beans/spring-beans.xsd
________________http://www.springframework.org/schema/util
________________http://www.springframework.org/schema/util/spring-util-2.0.xsd">

        <import resource="classifiers.xml"/>

        <bean id="ClassifiersSuccessViewer" class="testing.classificators.view.ClassifiersSuccessViewer">
                <property name="dataDir" value="c://hhtResults//classifiers//"/>
                <property name="patterns" ref="configsPatterns"/>
```

```
        <property name="resultsDir" value="c://hhtResults//classifiers//configs" />
        <property name="next" ref="configsSum" />
</bean>

<bean id="configsSum" class="testing.classificators.view.ClassifiersSuccessViewer">
        <property name="dataDir" value="c://hhtResults//classifiers//configs"/>
        <property name="resultsDir" value="c://hhtResults//classifiers//configs" />
        <property name="next" ref="classifiers" />
</bean>

<bean id="classifiers" class="testing.classificators.view.ClassifiersSuccessViewer">
        <property name="dataDir" value="c://hhtResults//classifiers//"/>
        <property name="resultsDir" value="c://hhtResults//classifiers//classifiers" />
        <property name="patterns" ref="classifiersPattern"/>
        <property name="next" ref="classifiersSum" />
</bean>
<bean id="classifiersSum" class="testing.classificators.view.ClassifiersSuccessViewer">
        <property name="dataDir" value="c://hhtResults//classifiers//classifiers"/>
        <property name="resultsDir" value="c://hhtResults//classifiers//classifiers" />
        <property name="next" ref="averages" />
</bean>
<bean id="averages" class="testing.classificators.view.ClassifiersSuccessViewer">
        <property name="dataDir" value="c://hhtResults//classifiers//"/>
        <property name="resultsDir" value="c://hhtResults//classifiers//averages" />
        <property name="patterns" ref="averagesPattern"/>
        <property name="next" ref="averagesSum" />
</bean>
<bean id="averagesSum" class="testing.classificators.view.ClassifiersSuccessViewer">
        <property name="dataDir" value="c://hhtResults//classifiers//averages"/>
        <property name="resultsDir" value="c://hhtResults//classifiers//averages" />
</bean>

<util:list id="averagesPattern">
                        <bean class="testing.classificators.view.Value">
                                <property name="value" value=".*_10_.*"/>
                                <property name="filename" value="10"/>
                        </bean>
                        <bean class="testing.classificators.view.Value">
                                <property name="value" value=".*_20_.*"/>
                                <property name="filename" value="20"/>
                        </bean>
                        <bean class="testing.classificators.view.Value">
                                <property name="value" value=".*_30_.*"/>
                                <property name="filename" value="30"/>
                        </bean>
</util:list>

<util:list id="classifiersPattern">
                        <bean class="testing.classificators.view.Value">
                                <property name="value" value=".*FreqAmpl4.*"/>
                                <property name="filename" value="FreqAmpl4"/>
                        </bean>
                        <bean class="testing.classificators.view.Value">
                                <property name="value" value=".*FreqAmpl5.*"/>
                                <property name="filename" value="FreqAmpl5"/>
                        </bean>
                        <bean class="testing.classificators.view.Value">
                                <property name="value" value=".*FreqAmpl6.*"/>
                                <property name="filename" value="FreqAmpl6"/>
                        </bean>
</util:list>

<util:list id="configsPatterns">
                        <bean class="testing.classificators.view.Value">
                                <property name="value" value="CauchyCfg1_.*"/>
                                <property name="filename" value="CauchyCfg1"/>
                        </bean>
```

```
                        <bean class="testing.classificators.view.Value">
                            <property name="value" value="CauchyCfg2_.*"/>
                            <property name="filename" value="CauchyCfg2"/>
                        </bean>
                        <bean class="testing.classificators.view.Value">
                            <property name="value" value="CauchyCfg3_.*"/>
                            <property name="filename" value="CauchyCfg3"/>
                        </bean>
    </util:list>
</beans>
```