Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Metody vyhodnocování elektrofyziologických experimentů

Plzeň, 2013 Tomáš Prokop

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 10.5.2013 …….

Tomáš Prokop

Abstrakt

There are several suitable methods for EEG signal processing such as time-frequency domain methods - Matching pursuit, Wavelet transform, short-time discrete Fourier transform or statistical methods - Linear Discriminant Analysis, or Principal Component Analysis. There is also one relatively new method designed for non-stationary signal processing known as Hilbert-Huang transform. It can be used to process EEG signal with some modifications.

The core of the Hilbert-Huang transform is an Empirical Mode Decomposition, which (in the sifting process) decomposes signal into Intrinsic Mode functions. I have designed two new stopping criteria to obtain better Intrinsic Mode functions during sifting process. I also created new classifiers to improve success rate of P3 component classification. These modifications were tested on real EEG data.

Obsah

1. Úvod

Počátky elektroencefalografie (EEG) lze datovat do roku 1875, kdy lékař Richard Caton publikoval své objevy o elektrické aktivitě v mozku králíků a opic. V dalších několika dekádách se zaznamenal značný pokrok v měření a vyhodnocování EEG. Přestože EEG je poměrně stará metoda vyšetření, stále je dnes hojně využívána. Novější metody jako je počítačová tomografie (CT), magnetická resonance (MRI) nebo pozitronová emisní tomografie (PET) mají velkou nevýhodu ve vysoké pořizovací ceně a špatné přenositelnosti zařízení. Výhodou EEG oproti novějším metodám je nízká cena vyšetření a velmi dobré časové rozlišení v řádu kHz.

EEG je náhodný, spojitý, nestacionární signál vytvářený aktivitou různých částí mozku nebo jiných částí nervového systému. EEG signál vznikne složením stovek aktivit různých neurálních zdrojů, což znesnadňuje vyčlenění určitého neurokognitivního procesu. Nervové odpovědi spjaté se specifickými senzorovými, kognitivními nebo motorickými událostmi se nazývají ERP (evokované potenciály, event-related potential) a jsou nedílnou součástí EEG signálu. V posledních letech se výzkum EEG ubírá právě směrem analýzy a hledání ERP komponent.

Výzkumná skupina při Katedře informatiky a výpočetní techniky Fakulty aplikovaných věd Západočeské university v Plzni spolupracuje s Fakultní nemocnicí v Plzni, společností Škoda Auto a Fakultou dopravní Českého vysokého učení technického v Praze na výzkumu v oblasti EEG/ERP. Součástí výzkumu je i provozování EEG/ERP laboratoře, EEG portálu a výzkum a vývoj metod detekce ERP komponent a zpracování EEG signálu [19].

K detekci ERP komponent a analýze EEG signálu se používá řada metod. K nejčastěji používaným patří Matching pursuit (MP) a waveletová transformace (WT). Tyto metody rozkládají signál použitím předdefinovaných komponent (mateřské wavelety, Gáborovy atomy) [19]. Dále se k analýze EEG signálu používají statistické metody, například analýza hlavních komponent (Principal Component Analysis, PCA) nebo analýza nezávislých komponent (Independent component analysis, ICA). Poměrně novou metodou je Hilbert Huangova transformace (HHT), která byla navržena N. E. Huangem a poprvé zveřejněna v roce 1998 v [20]. Metoda byla navržena pro zpracování nestacionárních nelineárních signálů, jakým je i EEG signál.

HHT rozkládá signál na vlastní modální funkce (IMF, Intrinsic Mode Functions), které jsou na rozdíl od předdefinovaných funkcí používaných k rozkladu signálu metodami MP a WT definovány samotným signálem. HHT dále transformuje původní signál na analytický signál, z kterého lze získat okamžité vlastnosti signálu (okamžitou frekvenci, amplitudu a fázi). Tyto vlastnosti jsou základními příznaky používanými k detekci ERP komponent.

Výzkum využití HHT k detekci ERP komponent byl publikován v [19], kde byly navrženy a otestovány modifikace různých částí HHT. Zároveň v závěru práce byl určen směr dalšího vývoje a nastíněny možné modifikace několika částí algoritmu

HHT. Cílem mé diplomové práce je zaměřit se na možnosti dalšího výzkumu HHT navržené ve výše zmíněné disertační práci. Jmenovitě se jedná o revizi implementace HHT (refactoring, okomentování kódu), návrh a otestování dodatečných zastavovacích podmínek procesu prosívání (sifting) a navržení nového klasifikátoru. Dále bych chtěl integrovat implementaci HHT pod software Matlab.

2. Elektroencefalografie

Elektroencefalografie je elektrofyziologická metoda používaná lékaři a vědci k monitorování mozkové aktivity. EEG má výborné časové rozlišení u monitorovaných potenciálů, zatímco přiřazení těchto potenciálů na místo svého původu je nepřesné. [2]

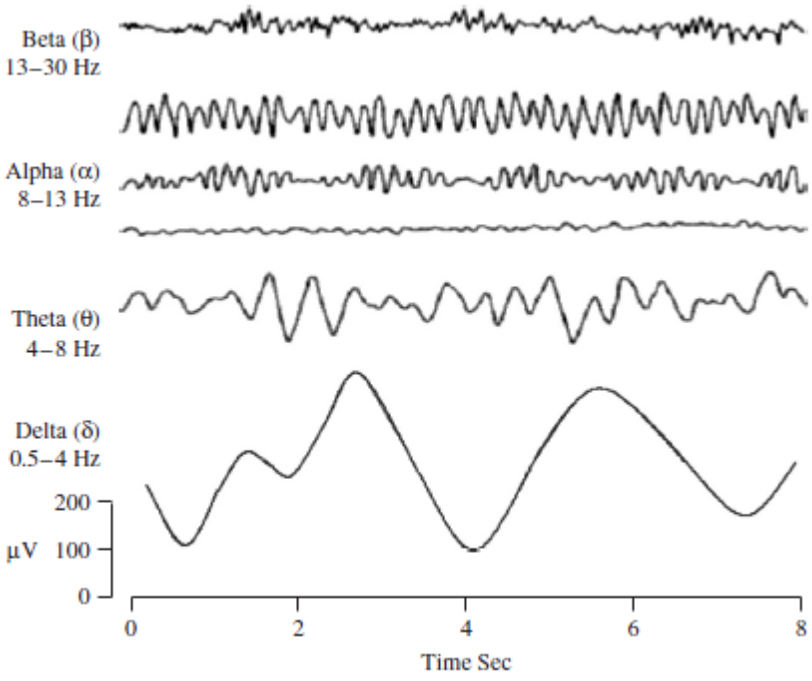
V lékařské praxi se používá měření EEG k [2]:

- vyšetření, zda pacient trpí epilepsií nebo migrénami
- potvrzení nebo vyloučení mozkové smrti
- stanovení prognózy u pacientů v kómatu
- monitorování mozkové aktivity během hluboké anestezie
- EEG biofeedback terapii pro lidi, kteří trpí poruchami učení, hyperaktivitou nebo zhoršenou koncentrací

2.1. Hlavní EEG rytmy

Mozkové vlny můžeme rozdělit do čtyř základních skupin podle jejich frekvenčního rozsahu (viz obrázek 1):

1. **δ vlny** – frekvence je v rozsahu od 0,5Hz do 4Hz, amplituda obvykle od 10μV do 300μV. Delta vlny jsou hlavně spojovány s hlubokým spánkem. Mohou se ale objevit i v bdělém stavu. Mohou být snadno zaměněny s artefakty. [23]
2. **θ vlny** – frekvence je v rozsahu od 4Hz do 7,5Hz, amplituda obvykle více než 20μV. Théta vlny hrají důležitou roli v kojeneckém období a dětství. Dospělí mají théta vlny pouze během ospalosti a spánku. [23]
3. **α vlny** – frekvence je v rozsahu od 8Hz do 13Hz, amplituda od 30μV do 50μV. Jsou nejlépe pozorovatelné u subjektů se zavřenýma očima při fyzickém uvolnění a uvolněném podvědomí bez pozornosti a koncentrace. [23]
4. **β vlny** – mají frekvenci obvykle od 13Hz do 30Hz a amplitudu od 5μV do 30μV. Tyto vlny se objevují u dospělých, kteří aktivně přemýšlí nebo řeší konkrétní problém. [23]



Obrázek 1: Hlavní mozkové vlny [1]

2.2. Artefakty

Artefakty jsou signály s původem jinde než v mozku, které se objevují v EEG signálu. Dělí se na dvě kategorie [3]:

- **Artefakty s biologickým původem:** například aktivita svalů, pohyb očí, mrknutí oka, tlukot srdce a jiné. [2]
- **Artefakty z okolního elektromagnetického pole:** například 50 Hz (60 Hz) z rozvodné sítě.

3. Evokované potenciály

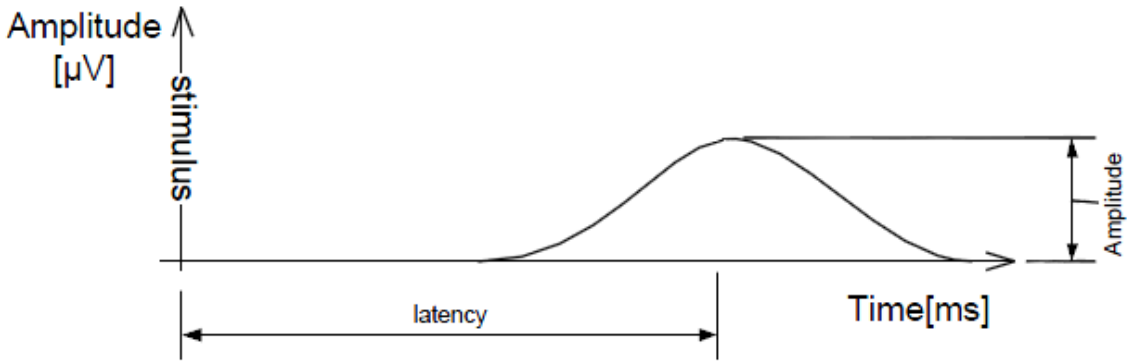
Evokovaný potenciál (ERP) je měřená odpověď mozku, která je přímým výsledkem specifické senzorické, motorické nebo kognitivní události. Formálněji je to každá stereotypní elektrofyziologická odpověď na stimul [4].

3.1. Průběh ERP

Následující vlastnosti popisují ERP průběh (viz obrázek 2):

- latence
- frekvence
- amplituda

Amplituda je mírou nervové aktivity odpovědi na stimul [19]. Latence je čas mezi výskytem stimulu a výskytem odpovědi na stimul. Je to tedy čas, který potřebuje mozek na zpracování informace.[2]



Obrázek 2: Vlastnosti ERP průběhu[5]

3.2. Značení ERP komponent

Názvy ERP komponent se skládají z jednoho písmene, po kterém následuje jedno či více číslic. Písmeno označuje polaritu ERP:

- **P** – kladná (positive)
- **N** – záporná (negative)
- **C** – nemá jednoznačně přiřazenou polaritu

Pokud je za písmenem jedna číslice, pak označuje pořadí vlny. Například N4 komponenta je čtvrtá negativní komponenta. Pokud jsou za znakem tři číslice, jedná se o označení latence. Například P100 je pozitivní vlna, která se zpravidla objeví 100ms po stimulu. [3]

3.3. Hlavní ERP komponenty

Následuje stručný přehled několika hlavních ERP komponent.

3.3.1. Visuální komponenty

- **C1** je první ERP komponenta, která se objeví po visuálním stimulu. Nemá jednoznačnou polaritu. Začíná 40 – 60ms po stimulu a vrcholu dosahuje v 80 – 100ms po stimulu. Je značně ovlivněna parametry stimulu. [4]
- **P1** vlna následuje po C1. P1 začíná 60-90ms po stimulu a vrcholí ve 100 – 130ms. Latence P1 se zásadně mění v závislosti na kontrastu stimulu. [4]
- **N1** následuje po P1 vlně. Má několik subkomponent. První subkomponenta vrcholí 100 – 150ms po stimulu a obvykle se objeví alespoň dvě další N1 komponenty, které vrcholí 150 – 200ms po stimulu. [4]
- **P2** vlna se objeví po N1 vlně. Tato komponenta je větší pro stimuly, které obsahují cílové příznaky. Je často špatně rozpoznatelná od přesahující N1, N2 a P3 vlny. [4]
- **P3** V časovém rozsahu P3 komponenty se může objevit několik rozdílných komponent. Dvě hlavní jsou P3a a P3b. [4]

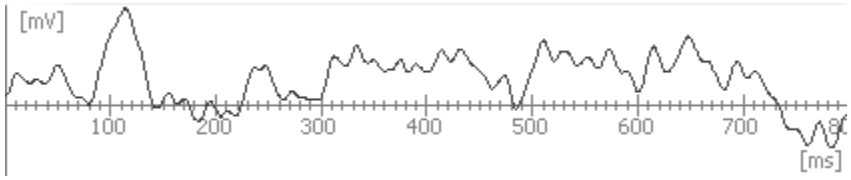
3.3.2. Sluchové komponenty

- **N1** Stejně jako visuální N1 vlna má 3 subkomponenty, které vrcholí přibližně 75ms, 100ms a 150ms po stimulu. Je citlivá na pozornost. [4]
- **N2** obsahuje několik různých komponent. První N2 vlna se nazývá základní N2 a je spojená s opakujícím se necílovým podnětem. Pokud je subjekt vystaven jinému stimulu (zvaný deviant), amplituda bude větší. Pokud takový podnět přísluší k úkolu, objeví se pozdější N2 komponenta zvaná N2b. [4]
- **P3** V časovém rozsahu P3 komponenty se může objevit několik rozdílných komponent. Dvě hlavní jsou P3a a P3b. Obě jsou vyvolány nepředvídatelnými, výjimečnými změnami v rozsahu nebo intenzitě tónu. P3b komponenta se objeví pouze v případě, že tyto změny přísluší k úkolu. [4]

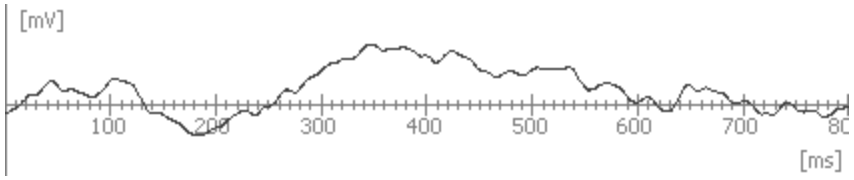
3.4. Průměrování

Amplituda ERP vlny je poměrně malá (do 30μV), ale pozadí EEG má amplitudu i 100μV. Proto je nezbytné použít průměrování ke zvýraznění ERP komponenty a potlačení pozadí EEG [23]. Průměrování je běžná metoda používaná pro zvýraznění ERP průběhu a potlačení šumu (viz obrázek 3).

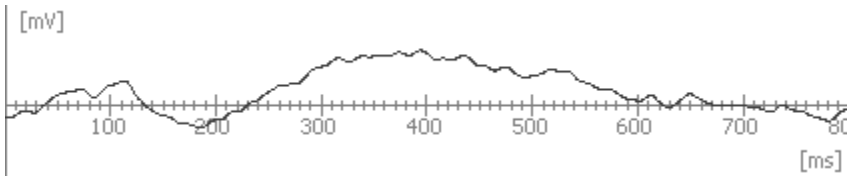
Průměrují se samozřejmě epochy obsahující stejnou ERP komponentu. Obecně není dobré použít všechny cílové epochy z ERP experimentu, protože některé ERP průběhy mohou být značně posunuté nebo ovlivněné nedetekovanými artefakty nebo některé cílové epochy nemusí obsahovat ERP vlnu. K určení, které epochy jsou vhodné pro průměrování, se často používá statistická metoda ANOVA (Analysis of Variance, analýza rozptylu). [3]



První epocha



Průměr sedmi epoch



Průměr patnácti epoch

Obrázek 3: Průměrování epoch obsahujících P3 vlnu [2]

3.5. Baseline korekce

Při měření amplitudy ERP vlny je obvykle amplituda ovlivněna průměrnou amplitudou před podnětem. Tato amplituda se nazývá hodnota baseline (pokud není nulová). Baseline má značný vliv na výsledek všech metod zpracování ERP (detekční algoritmy, průměrování, atd.). Proto je nutné kompenzovat základ v každé epoše. V [4] je doporučeno vypočítat průměrnou amplitudu v rozsahu 200ms před začátkem stimulu a odečíst tuto hodnotu od každé funkční hodnoty epochy. [3]

4. Metody vhodné pro zpracování EEG signálu

Metody vhodné k analýze EEG signálu lze rozdělit do třech skupin – statistické metody (např. lineární diskriminační analýza, analýza hlavních komponent, analýza nezávislých komponent, aj.), časově-frekvenční metody (např. Fourierova transformace, waveletová transformace, matching pursuit) a umělé neuronové sítě.

4.1. Lineární diskriminační analýza

Lineární diskriminační analýza (LDA), známá také jako Fisherův lineární diskriminant, je metoda používaná zejména k redukci dimenzí a klasifikaci dat do tříd. LDA snadno zvládne případy, kdy jsou frekvence rozptylů uvnitř tříd rozdílné. Metoda maximalizuje poměr rozptylu mezi třídami ku rozptylu uvnitř tříd v každém konkrétním vzorku dat, a tím garantuje maximální oddělitelnost tříd [5]. Základním předpokladem LDA je normální rozdělení nezávislých proměnných. Metoda se často používá pro vstupní data s velkým počtem dimenzí, kde nevadí předpoklad Gaussova rozdělení vstupních dat.

Předpokládejme, že máme data zobrazitelná v dvourozměrném systému a data jsou klasifikována do C tříd. Taková situace vypadá následovně [3]:

$$set\ 1 = \begin{bmatrix} s1_{11} & s1_{12} \\ \cdots & \cdots \\ s1_{m1} & s1_{m2} \end{bmatrix}, set\ 2 = \begin{bmatrix} s2_{11} & s2_{12} \\ \cdots & \cdots \\ s2_{n1} & s2_{n2} \end{bmatrix}, \ldots, set\ C = \begin{bmatrix} sC_{11} & sC_{12} \\ \cdots & \cdots \\ sC_{o1} & sC_{o2} \end{bmatrix}$$

Nad těmito daty se provedou následující operace:

- μ_i je vektor středních hodnot setu i pro i = 1, 2, …, C
- Mi je počet vzorků uvnitř setu i pro i = 1, 2, …, C
- $M = \sum_{i=0}^{C} Mi$ je celkový počet vzorků

Pak,

- Matice rozptylu uvnitř tříd je definována jako:

$$S_w = \sum_{i=1}^{C} \sum_{j=1}^{M_i} (y_j - \mu_i)(y_j - \mu_i)^T \tag{4.1}$$

- Matice rozptylu mezi třídami je definována jako:

$$S_b = \sum_{i=1}^{C} (\mu_i - \mu)(\mu_i - \mu)^T, \tag{4.2}$$

kde $\mu = \frac{1}{C} \sum_{i=0}^{C} \mu_i$ je střední hodnota všech dat.

LDA vypočítá transformaci, která maximalizuje rozptyl mezi třídami, zatímco minimalizuje rozptyl uvnitř tříd [3]:

$$maximalizace \frac{\det(S_b)}{\det(S_w)} \tag{4.3}$$

4.1.1. Detekce ERP s LDA

Z každé epochy získáme N-dimenzionální vektor příznaků. Poté manuálně rozdělíme vektory příznaků do dvou řad – jednu obsahující ERP průběh a druhou řadu bez ERP. Vypočte se LDA a stanoví se lineární funkce (nadrovina), která rozdělí N-dimenzionální prostor do dvou podprostorů. Jeden pro každou řadu. Jeden podprostor obsahuje vektory příznaků epoch, které obsahují průběh ERP. V druhém podprostoru jsou všechny ostatní vektory příznaků. [3]

Nechť $f(p_1, p_1, \ldots, p_N) = 0$ je nadrovina. Během procesu klasifikace jsou pro každý vektor příznaků každé epochy substituovány položky do rovnice nadroviny a poté vypočítány. V případě, že je výsledek vyšší než nula, patří vektor příznaků k první řadě. V opačném případě patří k druhé řadě. [3]

4.2. Analýza hlavních komponent

Analýza hlavních komponent (PCA, Principal Component Analysis) je matematická metoda, způsob jak identifikovat vzory v datech a vyjádřit data takovým způsobem, aby se zvýraznily podobnosti a rozdíly [6]. Je založena na statistickém popisu náhodné proměnné. Stejně jako LDA se využívá v klasifikaci dat (extrakce příznaků) a k redukci dimenzí. Její výhodou je malá ztráta informace při kompresi, pokud jsme již nalezli vzory například redukcí dimenzí [6].

Mějme vstupní signál $X_i \in R^n$. PCA předpokládá, že suma vzorků vstupního signálu je rovna nule, tedy $\sum_{i=1}^{k} X_i = \mathbf{0}$, kde $\mathbf{0} \in R^n$ je nulový vektor. Abychom toho docílili, musíme od každého vzorku odečíst průměr $\frac{1}{k}\sum_{i=1}^{k} X_i$. Cílem PCA je nalézt $m < n$ vektorů $Y_j \in R^n$ takových, že reprezentace vzorků X_i lineární kombinací Y_j má ve smyslu nejmenších čtverců minimální reziduum. Tedy, aproximujeme každé X_i lineární kombinací Y_j:

$$Z_i = \sum_{j=1}^{m} w_{ij} Y_j, \tag{4.4}$$

kde Z_i je aproximace X_i taková, že residuum

$$g(Y_j, w_{ij}) = \sum_{i=1}^{k} \|X_i - Z_i\|^2 \tag{4.5}$$

je minimální. Vektory Y_j řešící tento problém jsou rovné vlastním vektorům matice $S = XX^T$ odpovídajícím m největším vlastním číslům.

4.3. Analýza nezávislých komponent

Analýza nezávislých komponent (ICA, Independent Component Analysis) je statistická metoda, která odhaluje skryté faktory, které tvoří základ sady náhodných proměnných, měření nebo signálů [7]. ICA se snaží ze vstupního signálu separovat nezávislé komponenty, které jsou ve vstupním signálu smíchány. ICA zdánlivě souvisí s PCA a faktorovou analýzou (FA), ale je silnějším nástrojem, protože je schopna najít základní

faktory nebo zdroje i tam, kde PCA a FA selže. Předpokládá se, že komponenty nemají normální rozdělení a jsou vzájemně nezávislé.

4.4. Fourierova transformace

Fourierova transformace (FT) slouží k převodu periodického signálu z časové oblasti do frekvenční oblasti. Předpokladem metody jsou periodická vstupní data. Pokud tato podmínka není splněna, lze vstupní signál považovat za právě jednu periodu periodického signálu. FT je nedílnou součástí mnoha jiných, složitějších metod (např. i HHT). Existuje v několika různých variantách ve spojité i diskrétní verzi.

4.4.1. Spojitá Fourierova transformace

Spojitá Fourierova transformace (CFT) je dána následujícím vztahem:

$$F(f) = \int_{-\infty}^{+\infty} x(t) \cdot e^{-i2\pi ft} dt, \tag{4.6}$$

kde x(t) je původní signál, t je čas a f je frekvence. Nevýhodou CFT je ztráta časové informace. Proto nelze použít CFT k detekci ERP [3].

4.4.2. Diskrétní Fourierova transformace

Diskrétní Fourierova transformace (DFT) pro posloupnost čísel x_0, x_1, …, x_{N-1} je dána následujícím vztahem:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N} \tag{4.7}$$

Klasická DFT vyžaduje n^2 komplexních násobení. Proto se v praxi využívá algoritmu rychlé Fourierovy transformace (FFT), který má složitost O(N·logN).

4.4.3. Krátkodobá analýza signálu

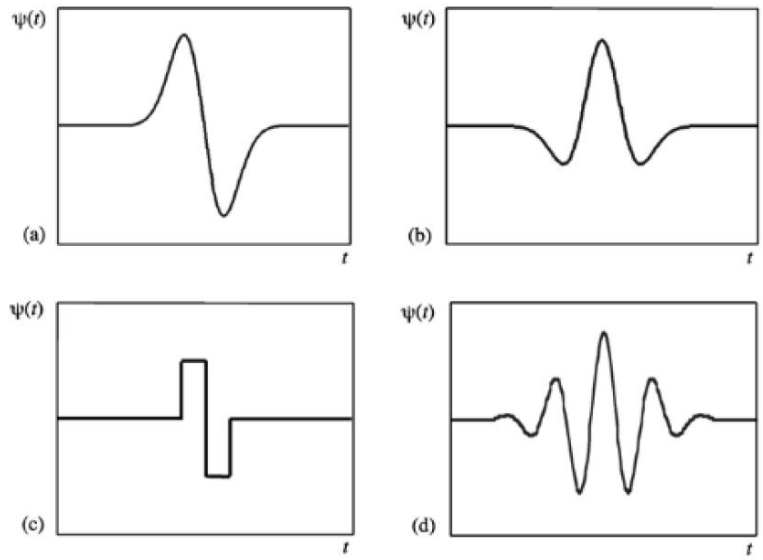
FFT se používá k analýze periodických signálů. EEG signál ale periodický není. Pokud vypočítáme FFT na celém signálu, ztratíme informaci o čase, která je pro detekci ERP podstatná. Proto se k analýze EEG signálu používá krátkodobá Fourierova transformace (STFT). Její princip spočívá v rozdělení vstupního signálu symetrickým oknem na menší části, na kterých je posléze vypočítána Fourierova transformace. Okno je funkce, která je nenulová pouze na krátkém intervalu [3]. Nejpoužívanějšími okénkovými funkcemi jsou pravoúhlé okno, Hammingovo okno nebo Hannovo okno. STFT pro spojitý signál (CSTFT) lze vyjádřit následovně:

$$STFT\{x(t)\} \equiv F(\tau, f) = \int_{-\infty}^{+\infty} x(t) w(t-\tau) e^{-jwt} dt, \tag{4.8}$$

kde w(t) je okno, x(t) je vstupní signál v časové oblasti, τ je časový posun okna w(t) a f je analyzovaná frekvence [3]. STFT existuje samozřejmě i ve verzi pro diskrétní signál (DSTFT). DSTFT se využívá k detekci SSVEP (steady state visually evoked potentials).

4.5. Waveletová transformace

Jednou z často používaných metod k analýze a zpracování nestacionárního signálu je waveletová transformace (WT). WT má dobrou časovou a frekvenční lokalizaci, která je nezbytná pro extrakci příznaků a následnou detekci ERP. WT je možné použít pro zpracování spojitého signálu (CWT) i diskrétního signálu (DWT). Základní myšlenkou WT je rozložit časovou funkci do tzv. wavelet. Některé základní wavelety jsou vidět na obrázku 4. [3]



Obrázek 4: Některé často používané wavelety: (a) Gaussova vlna, (b) mexický klobouk, (c) Haarův wavelet, (d) Morlet [3]

4.5.1. Spojitá waveletová transformace

Nechť *ψ(t)* je wavelet. Pak její funkční hodnoty pro dilataci *a* a translaci *b* lze vyjádřit následovně:

$$\psi_{u,s}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{4.9}$$

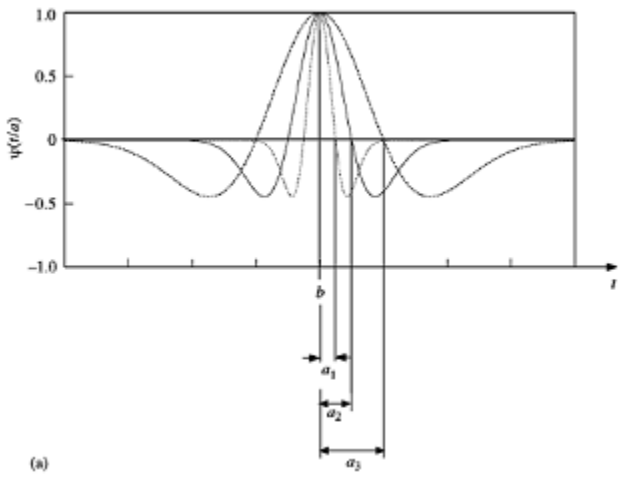
Spojitá waveletová transformace signálu *f* pro dilataci *a* a translaci *b* wavelety *ψ* je definována v [8] následovně:

$$WT(f,u,s) = \int_{-\infty}^{+\infty} f(t)\frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right)dt \tag{4.10}$$

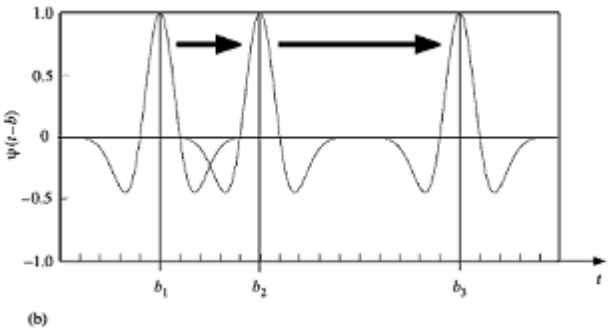
Principy CWT budu demonstrovat na waveletě mexický klobouk, který je definován následovně:

$$\psi\left(\frac{t-b}{a}\right) = \left[1 - \frac{(t-b)^2}{a}\right]\cdot e^{-\frac{1}{2}\left[\frac{t-b}{a}\right]^2}, \tag{4.11}$$

kde *a* (dilatace) odpovídá frekvenci a *b* (translace) popisuje posun wavelety po signálu viz obrázky 5 a 6.



Obrázek 5: Dilatace wavelety (Mexický klobouk). [9]

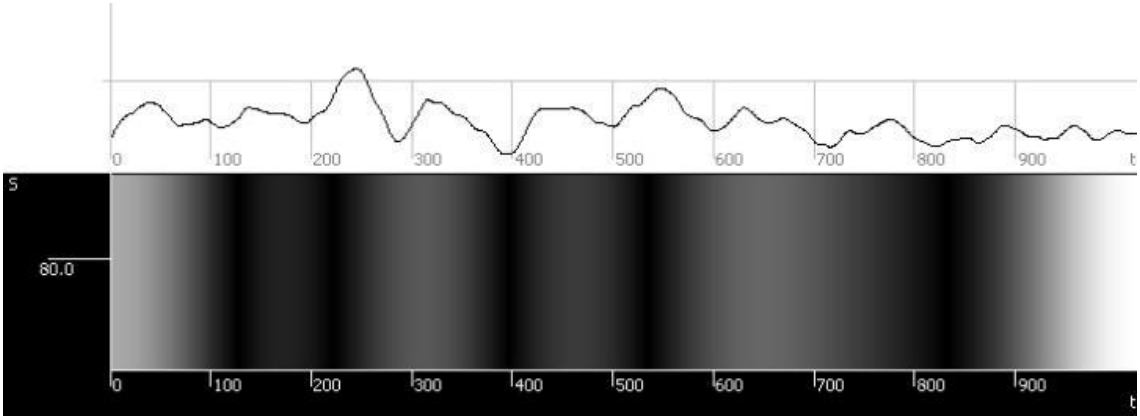


Obrázek 6: Translace wavelety (Mexický klobouk). [9]

K detekci ERP lze využít následující algoritmus CWT [3]:

1) Nastavení mateřské wavelety, počáteční a koncové hodnoty dilatace, krok dilatace a translace.
2) Výpočet součtu hodnot korelace pro aktuální dilataci a pro každou translaci tak, aby byl pokryt celý signál.
3) Zvýšení hodnoty dilatace o zvolený krok a pokračování krokem 2).
4) Algoritmus končí, když dilatace dosáhne maximální hodnoty zvolené na začátku algoritmu.

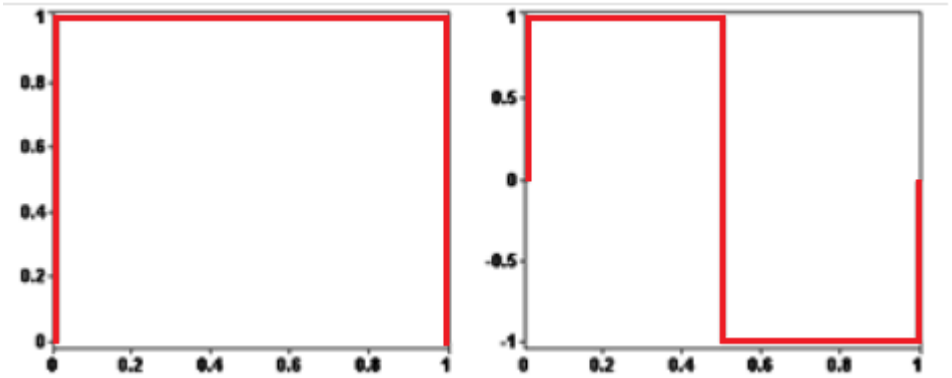
Výsledek CWT je zobrazen ve scalogramu, kde každý koeficient reprezentuje stupeň korelace mezi transformovanou waveletou a vstupním signálem. Scalogram je šedotónový. Nejvyšší hodnoty jsou zobrazeny bílou barvou (viz. Obrázek 7). [3]



Obrázek 7: Vstupní signál a jeho scalogram. [10]

4.5.2. Diskrétní waveletová transformace

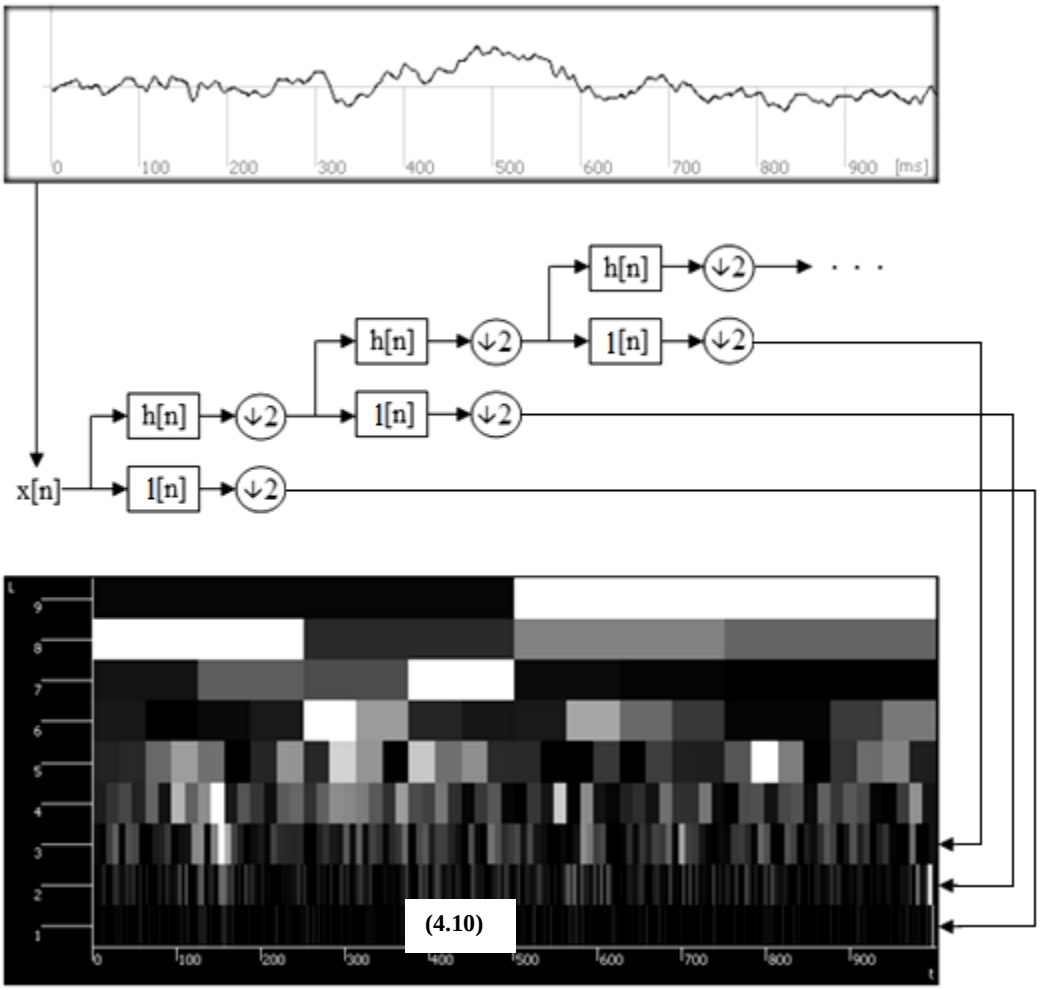
Spojitá waveletová funkce z předchozí kapitoly je zde nahrazena dvěma diskrétními signály - waveletovou funkcí a škálovací funkcí (viz obrázek 8). [3]



Obrázek 8: Haarův wavelet - škálovací funkce vlevo, wavelet napravo [10]

Vzhledem k omezenému spektru skupiny waveletových funkcí můžeme interpretovat proces konvoluce s waveletovou funkcí jako omezenou pásmovou propust. Z hlediska zpracování digitálního signálu můžeme považovat DWT za banku filtrů, kterými dekomponujeme signál do množiny dílčích frekvencí. Nejpomalejší základní frekvenční komponenty jsou detekovány škálovací funkcí. Waveletová funkce může být popsána horní propustí a škálovací funkce filtrem dolní propust. Příslušné koeficienty jsou určeny konvolucí signálu s odpovídající analyzující funkcí. Škála je nepřímo úměrná frekvenci, to znamená, že nízkým frekvencím odpovídají vysoké hodnoty škálovací funkce a velká dilatace waveletové funkce. Waveletovou analýzou na velkých měřítkách získáme ze signálu globální informaci (approximační komponentu). Naopak DWT na malých měřítkách získáme detailní informaci (detailní komponentu) vyjadřující rapidní změny v signálu. [3]

Výpočet koeficientů DWT je implementován postupnou aplikací waveletové funkce (filtr horní propust) a škálovací funkce (filtr dolní propust) na vstupní signál použitím Mallatovova dekompozičního schématu (viz obrázek 9) [3].



Obrázek 9: Principy DWT [10]

Pro každou úroveň dekompozice *p* je výstupem filtru horní propust $h_d(k)$ takzvaná detailní komponenta vstupního signálu $D_p(n)$. Approximační komponenta $A_p(n)$ je pak výstupem nízko-frekvenčního filtru $l_d(k)$. Použitím konvoluce a následného podvzorkování platí následující rovnice [11]:

$$D_p(n) = \sum_{k=0}^{L-1} h_d(k) A_{p-1}(2n-k) \tag{4.12}$$

$$A_p(n) = \sum_{k=0}^{L-1} l_d(k) A_{p-1}(2n-k) \tag{4.13}$$

pro n = 0, 1, …, N/2, kde $A_0(n) = x(n)$ je analyzovaný signál a obě sekvence $h_d(k)$ a $l_d(k)$ definují dekompoziční filtry.

4.5.3. Detekce ERP waveletovou transformací

Když hledáme průběh ERP v signálu, vypočítáme nejdříve korelaci mezi waveletou (která je v měřítku odpovídajícímu ERP průběhu) a EEG/ERP signálem v místě, kde lze ERP očekávat. Pokud bychom hledali ERP vlny kdekoli v signálu, mohlo by dojít

k chybné detekci ERP. Koeficienty WT jsou ovlivněny shodou naškálované wavelety a signálu a také amplitudou signálu. Pokud je stupeň korelace vyšší než stanovený práh, považujeme ERP průběh za detekovaný. [12]

Samozřejmě platí, že čím více je použitý wavelet podobnější detekovanému ERP průběhu, tím vyšší je stupeň korelace v případě, že vstupní signál tento ERP průběh obsahuje. Nejjednodušší myšlenkou je vytvořit model ERP průběhu, který chceme detekovat, a použít ho jako waveletu. Bohužel wavelet je matematicky definován a každý wavelet musí splnit striktní podmínky: [3]

1. Energie wavelety musí být konečná:

$$E = \int_{-\infty}^{+\infty} \Psi(t)^2 dt < \infty, \tag{4.14}$$

   kde E je energie a Ψ je wavelet.

2. Pokud $\Psi'(f)$ je Fourierova transformace $\Psi(f)$, tedy

$$\Psi'(f) = \int_{-\infty}^{+\infty} \Psi(t) \cdot e^{-i2\pi f t} dt, \tag{4.15}$$

   pak musí být splněna následující podmínka:

$$\int_{0}^{+\infty} \frac{\Psi'(f)^2}{f} df < \infty \tag{4.16}$$

   Tato podmínka vlastně říká, že průměr všech funkčních hodnot funkce $\Psi(t)$ musí být roven nule.

Kvůli těmto podmínkám a faktu, že průměrná hodnota všech funkčních hodnot ERP průběhu není rovna nule, je nutné udělat řadu testů a zvolit waveletu a hodnotu korelačního prahu empiricky. [3]

Nevýhodou CWT je její vysoká výpočetní náročnost, která lineárně roste s počtem vzorků signálu. Zvýšením počtu vzorků vstupního signálu nemá takový dopad na čas výpočtu v případě DWT.

4.6. Algoritmus Matching pursuit

4.6.1. Základní principy

Matching pursuit (MP) je algoritmus, který dekomponuje vstupní signál na součet takzvaných atomů, které jsou vybírány ze slovníku [3]. To znamená, že vstupní signál x může být vyjádřen atomy g_n a adekvátními konstantami a_n následovně:

$$x \approx \sum_{n=0}^{N-1} a_n g_n \tag{4.17}$$

Během každé iterace je vybrán atom, který nejtěsněji aproximuje vstupní signál. Tento atom je pak odečten od vstupního signálu a zbytek signálu je vstupním signálem další iterace algoritmu. Celkový součet atomů úspěšně vybraných v iteracích algoritmu je aproximací původního signálu. Čím více se provede iterací, tím přesnější dostaneme odhad původního signálu [13]. S rostoucím počtem iterací MP algoritmu jde chyba rozdílu mezi vstupním signálem a jeho odhadem k nule.

MP algoritmus je nejčastěji spojován s Gáborovými atomy. Gáborovy atomy jsou definovány jako Gaussovo okno:

$$g(t) = e^{-\pi t^2}, \tag{4.18}$$

které je modulováno sinovou funkcí následovně:

$$g_{(s,u,v,w)}(t) = g\left(\frac{t-u}{s}\right) \cdot \cos\left(vt + w\right) \tag{4.19}$$

Každý atom je jednoznačně definovaný uspořádanou čtveřicí (s, u, v, w), kde s je měřítko, u je posun, v je frekvence a w je fázový posun.

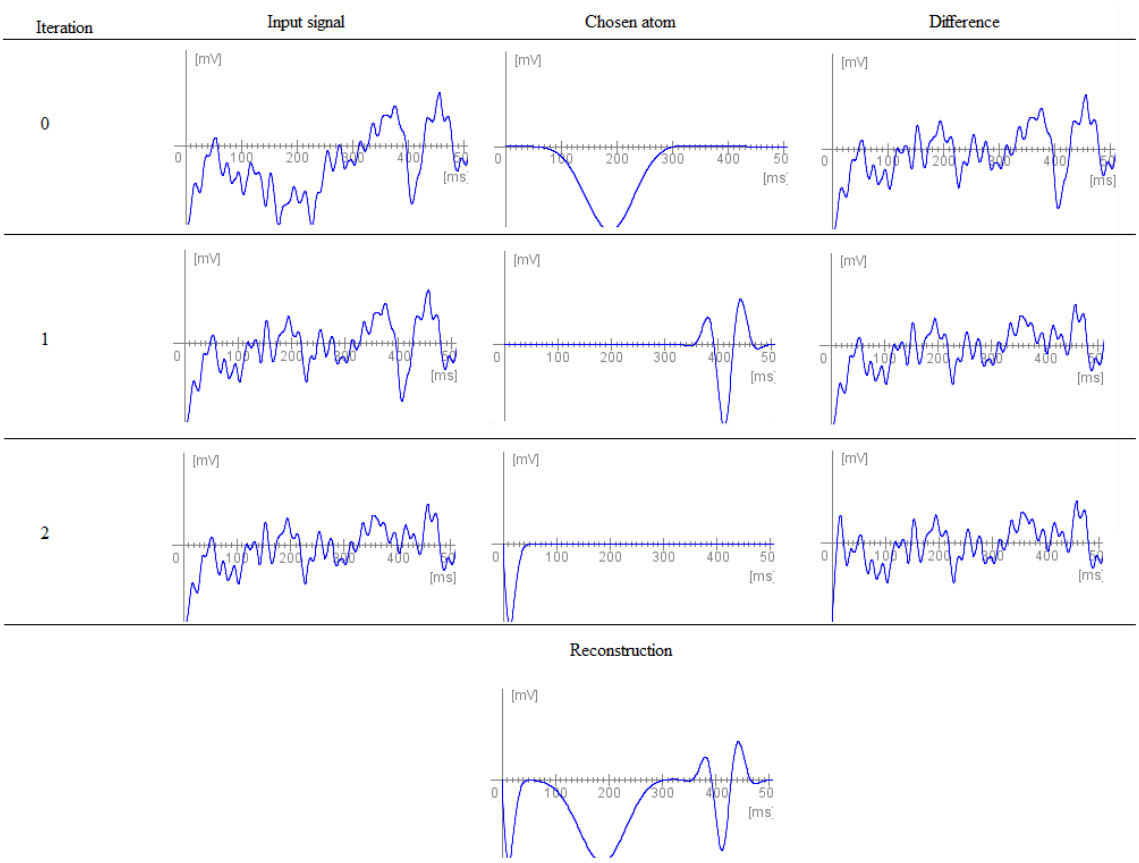
Nechť $\gamma = (s, u, v, w)$ je modulační vektor a $\gamma_k = (s, u, v, w)$ je modulační vektor vybraný v k-tém kroku algoritmu a nechť f je vstupní signál. Kritérium pro vybrání γ_k v každé iteraci je skalární součin $\langle f, \gamma \rangle$, který je maximální pro γ_k. Vstupní signál po jedné iteraci lze vyjádřit následovně:

$$f = \langle f, g_{\gamma_0} \rangle \cdot g_{\gamma_0} + Rf, \tag{4.20}$$

kde Rf je rozdíl mezi f a g_{γ_0}. Následující výraz je zobecněním pro M iterací MP algoritmu.

$$f = \sum_{i=0}^{M-1} \langle R^i \cdot f, g_{\gamma_i} \rangle \cdot g_{\gamma_i} + R^M \cdot f, \tag{4.21}$$

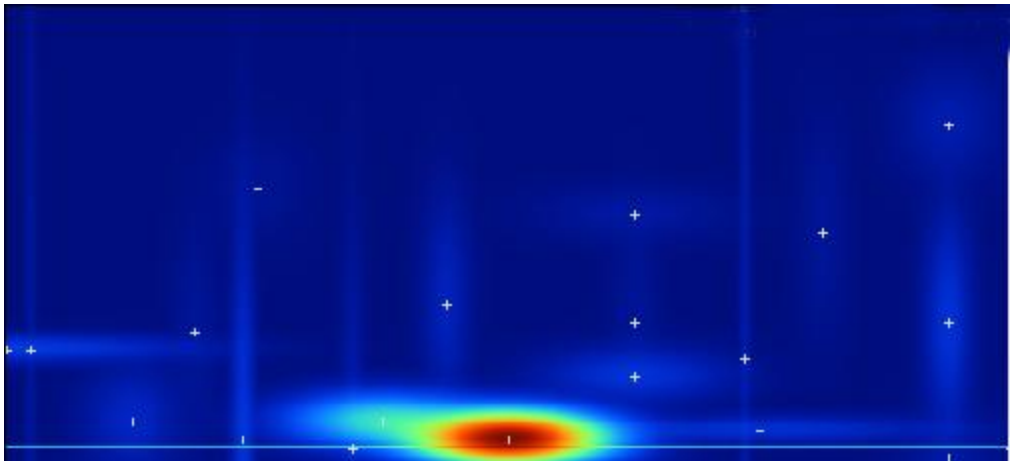
Kde $R^M \cdot f$ je rozdíl mezi f a součtem všech M Gaborových atomů. Na obrázku 10 je ukázka tří iterací MP algoritmu. U každé iterace je vidět signál, který vstupuje do iterace, vybraný atom a rozdíl mezi vstupním signálem a vybraným atomem. V dolní části je vidět zrekonstruovaný signál z vybraných atomů. [3]



Obrázek 10: Příklad tří iterací MP algoritmu [3]

4.6.2. Vizualizace výstupu

Výsledkem MP algoritmu je dvojdimenzionální matice. Řádky zde představují čísla atomů a sloupce se skládají z parametrů atomu (měřítko, posun, frekvence a fázový posun). Tato forma výstupu je vhodná pro další zpracování, protože obsahuje všechny podstatné informace, ale nehodí se pro konečný výstup. Většinou se pro finální výstup používá Wigner-Villeho transformace, která umožňuje analyzovat výsledek MP algoritmu pouhým okem. Více o Wigner-Villeho transformaci v [14 a 15]. Na obrázku 11 je vidět příklad Wingner-Villeho transformace výstupu MP algoritmu. [3]

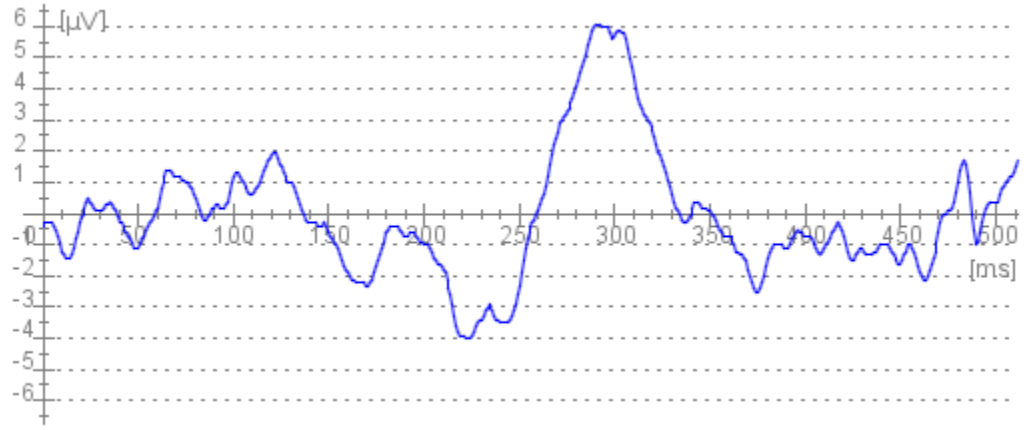


Obrázek 11: Wigner-Villeho transformace výstupu MP algortimu [16]

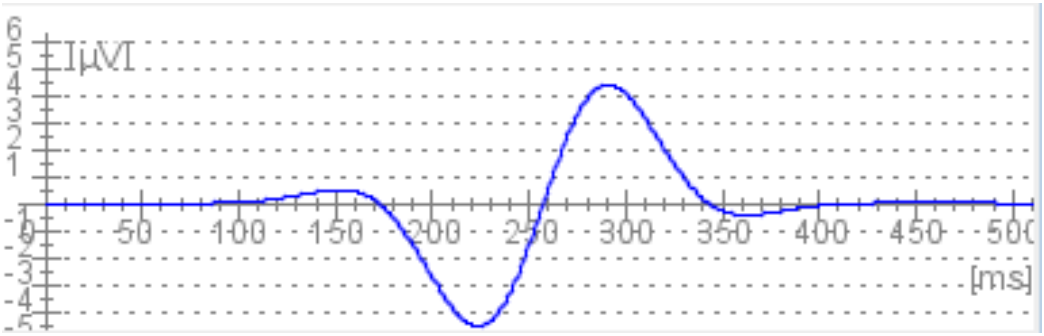
4.6.3. Detekce ERP klasickým MP algoritmem

Princip MP algoritmu je dekomponovat signál na individuální atomy. Zpočátku je atomy aproximován směr signálu a v pozdějších iteracích jsou dekomponovány detaily signálu. Během nahrávání mozkové aktivity se ERP objeví jako trendy signálu, které jsou narušeny EEG signálem [19]. Po několika iteracích je vstupní signál aproximován atomy do té míry, že je zvýrazněn směr signálu [17].

Každý atom je podle rovnice (4.19) unikátně popsán uspořádanou čtveřicí (s, u, v, w). Kromě toho je po spuštění MP algoritmu dostupný modulus pro každý atom. Modulus je stupeň korelace mezi atomem a vstupním signálem v iteraci algoritmu. Z těchto hodnot může být určen směr atomu v čase. Přesnost aproximace originálního signálu může být určena v závislosti na hodnotě modulu. Čím vyšší je hodnota modulu, tím lépe je původní signál aproximován. Hodnota modulu je vysoká v případě, že se v EEG signálu objeví ERP, protože ERP průběh reflektuje směr signálu [19]. Zároveň hodnota posunu (u ve výše zmíněné uspořádané čtveřici) odpovídá předpokládané poloze ERP v signálu [16]. Myšlenka detekce ERP komponent MP algoritmem byla publikována v [18]. Na obrázcích 12 a 13 je vidět příklad detekce P3 komponenty MP algoritmem.



Obrázek 12: Signál obsahující P3 komponentu [16]

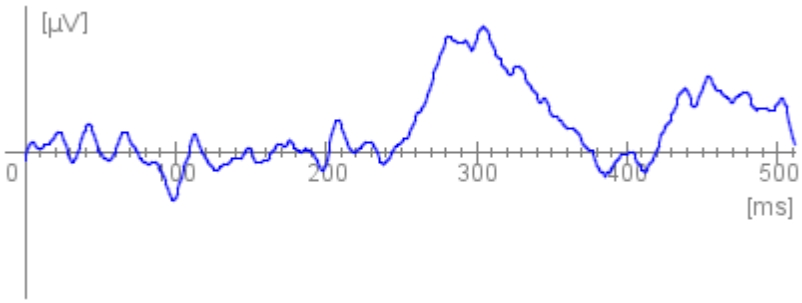


Obrázek 13: Gáborův atom, který nejlépe aproximuje P3 komponentu [16]

4.6.4. Detekce ERP modifikovaným MP algoritmem

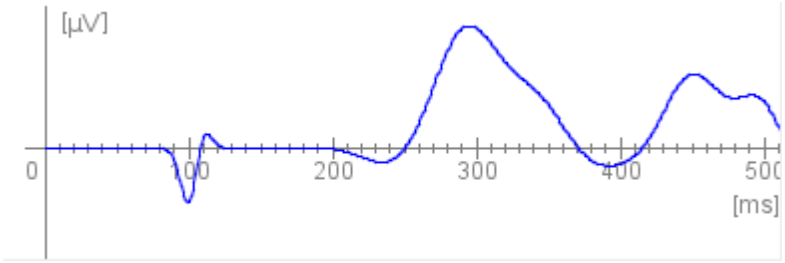
Základní myšlenkou modifikovaného MP algoritmu není zakládat detekci ERP komponenty na klasifikaci vektoru příznaků (vektory příznaků jsou zde parametry

Gáborových atomů), ale použít MP k filtraci vstupního signálu a poté vypočítat korelaci mezi filtrovaným (zrekonstruovaným) signálem a modelem ERP komponenty. [19]

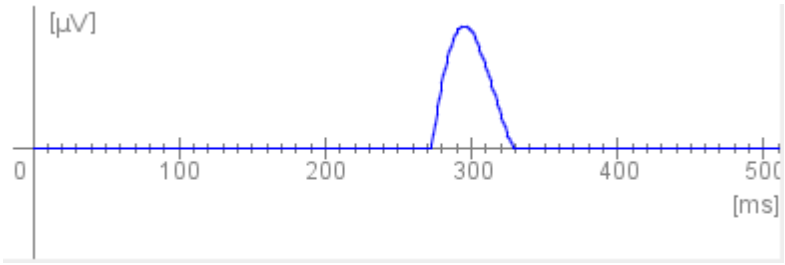


Obrázek 14: vstupní signál [19]

Nejdříve aproximujeme vstupní signál (obrázek 14) několika Gáborovými atomy. Pak z atomů zrekonstruujeme signál. Ztrátu informace způsobenou aproximací považujeme za filtraci vstupního signálu (obrázek 15).



Obrázek 15: Zrekonstruovaný signál - 5 Gáborových atomů [19]



Obrázek 16: Model ERP komponenty v odpovídající oblasti [19]

Vlastností MP algoritmu je potlačení šumu. Pokud ale provedeme velké množství iterací algoritmu, může se původní šum začít objevovat. Rekonstruovaný signál kopíruje trend původního signálu. Toho lze využít, protože ERP komponenty jsou (pokud nejsou zatížené artefakty) částí trendu signálu. Další fáze algoritmu je samotná detekce, kdy je použit model ERP komponenty (obrázek 16). Model získáme například zprůměrováním dostatečného množství epoch obsahujících hrubý ERP signál nebo filtrací ERP komponenty z jedné epochy. Model ERP komponenty je posunut na zrekonstruovaný signál na přibližný začátek oblasti, kde lze ERP komponentu očekávat. Poté se vypočítá korelace mezi modelem ERP komponenty a zrekonstruovaným signálem pro každý relevantní posun – od začátku do konce oblasti, kde se ERP

komponenta obvykle vyskytuje. Uloží se maximální hodnota korelace a příslušný posun. Po vypočtení všech možných korelací máme uložené globální maximum, které porovnáme s prahovou hodnotou. Pokud je hodnota maximální korelace vyšší než zvolený práh, ERP komponenta je detekována v odpovídající oblasti. [19]

5. Hilbert-Huangova transformace

HHT neboli Hilbert-Huangova transformace je metoda zpracování signálu navržená speciálně pro nelineární nestacionární signály. Metoda se skládá ze dvou částí – Empirical Mode Decomposition (EMD) a Hilbertovy spektrální analýzy (HAS).

5.1. Nestacionární signál

Nestacionární signál je takový signál, jehož frekvence se mění v čase (například lidská řeč nebo EEG signál). Oproti tomu ve stacionárním signálu se nemění frekvenční obsah, tedy všechny frekvenční komponenty existují v každém čase.

Pokud vypočteme statistiku EEG signálu v různých časech, zjistíme, že se výrazně liší. Tento jev nastává, protože EEG signál se skládá z různých mozkových rytmů, artefaktů atd.

5.2. Empirical Mode Decomposition

Při EMD je původní signál rozkládán na množinu Intrinsic Mode funkcí (IMF) a zbytek signálu. Většina dat nejsou IMF. Každá IMF musí splnit následující podmínky:

1) Počet křížení nuly a počet extrémů si je v celém objemu dat roven nebo je rozdílný maximálně o jedna.
2) Střední hodnota obálky definované lokálními maximy a minimy je nulová v každém bodě. [17; 20]

Splnění těchto podmínek je nezbytné pro definici okamžité frekvence. Každá IMF reprezentuje jednoduchý oscilační mód jako protějšek k jednoduché harmonické funkci, ale je ze své definice mnohem obecnější.[19]

V [21] bylo dokázáno, že první podmínka, kterou musí funkce splnit, aby byla IMF, je odvoditelná z druhé podmínky. Proto lze definovat IMF pouze jako funkci, jejíž střední hodnota obálky definované lokálními maximy a minimy je nulová v každém bodě.

V každém čase mohou data obsahovat více než jeden oscilační mód, a proto nelze pouhou Hilbertovou transformací plně popsat frekvenci. Proces hledání IMF se nazývá prosívání (sifting). Algoritmus EMD je následující [19]:

1) Inicializace zbytku na původní signál $r_0(t) = x(t)$ a čítače IMF i=1
2) Nalezení i-té IMF:
    a) Inicializace signálu $h_0=r_{i-1}(t)$ a inicializace čítače k=1
    b) Nalezení lokálních extrémů v signálu $h_{k-1}(t)$
    c) Vytvoření horní obálky proložením nalezených maxim kubickou křivkou
    d) Vytvoření dolní obálky proložením nalezených minim kubickou křivkou
    e) Výpočet průměru $m_{k-1}(t)$ zprůměrováním horní a dolní obálky
    f) Výpočet $h_k(t) = h_{k-1}(t) - m_{k-1}(t)$
    g) Kontrola zastavovací podmínky
        i. Kritéria splněna: $IMF_i(t) = h_k(t)$
        ii. Jinak k++, pokračuj bodem 2a

3) Nový zbytek je $r_i(t) = r_{i-1}(t) - IMF_i(t)$
4) Kontrola zastavovací podmínky EMD
    a) $r_i(t)$ má alespoň 2 extrémy, pak i++ a pokračování bodem 2
    b) Jinak konec dekompozice a $r_i(t)$ je zbytek signálu po dekompozici

5.2.1. Zastavovací podmínky

Každá IMF musí splnit druhou podmínku (viz. 5.2.), to může být ale problém. Jen velmi těžko lze dosáhnout absolutně nulové střední hodnoty průměru obálek. Při zvyšujícím se počtu iterací algoritmu hledání IMF (bod 2 algoritmu EMD) se postupně přibližuje střední hodnota nule. Tím se ale vyrovnávají amplitudy jednotlivých vln. Pokud chceme bezpodmínečně splnit tuto podmínku, můžeme předpokládat, že amplitudy jsou konstantní a tím ztratíme důležitou informaci o signálu [19]. Pokud chceme dosáhnout striktně nulové střední hodnoty, musíme také provést více iterací algoritmu než při určité toleranci. Z těchto důvodů se často využívá jako zastavovací podmínka normální rozdělení (SD).

$$SD = \sum_{t=0}^{T} \frac{|h_{k-1}(t) - h_k(t)|^2}{h_{k-1}^2(t)} \tag{5.1}$$

Druhou možností je použít Cauchyho testu konvergence (CCT) [22].

$$CCT = \frac{\sum_{t=0}^{T} |h_{k-1}(t) - h_k(t)|^2}{\sum_{t=0}^{T} h_{k-1}^2(t)} \tag{5.2}$$

Proces prosívání končí, když je výsledek zastavovací podmínky menší než zadaný práh.

5.3. Hilbertova transformace

Výsledkem Hilbertovy transformace je analytický signál získaný ze sekvence reálných dat. Analytický signál $x = x_r + i*x_i$ je signál skládající se z reálné části x_r reprezentující původní data a imaginární části x_i, která představuje Hilbertovu transformaci. Imaginární část analytického signálu je původní signál s fází posunutou o 90°. Signál transformovaný Hilbertovou transformací si zachovává amplitudu a frekvenci původního signálu a fáze nového signálu se odvíjí od fáze původního signálu. Například pokud aplikujeme transformaci na funkci sinus, získáme cosinus.

Hilbertova transformace se využívá k získání okamžitých vlastností signálu, hlavně okamžité amplitudy a frekvence. Okamžitá amplituda je amplituda komplexní Hilbertovy transformace. Okamžitá frekvence vyjadřuje poměr změny úhlu okamžité fáze. V případě sinusové křivky jsou okamžitá amplituda a frekvence konstantní. [19]

5.3.1. Výpočet standardního analytického signálu o stejném počtu vzorků

HHT aproximuje analytický signál tak, že vypočítá rychlou Fourierovu transformaci nad vstupní sekvencí x, nahradí koeficienty FFT odpovídající záporným frekvencím nulami a vypočte inverzní FFT nad takto upraveným výsledkem. Algoritmus výpočtu analytického signálu je následující:

1) Výpočet FFT nad vstupními daty a uložení výsledku do vektoru x.
2) Vytvoření vektoru h s následujícími hodnotami:
   - 1 pro i = 1, (n/2) + 1
   - 2 pro i = 2, 3, …, n/2
   - 0 pro i = (n/2) + 2, …, n
3) Vynásobení každého prvku vektoru x odpovídajícím prvkem vektoru h.
4) Výpočet inverzní FFT na datech z kroku 3 a vrácení prvních n elementů výsledku.

Výsledkem tohoto algoritmu je analytický signál Z(t) definovaný následovně:

$$Z(t) = X(t) + iY(t) = a(t)e^{i\theta(t)}, \tag{5.3}$$

kde X(t) je původní signál a Y(t) je výsledek Hilbertovy transformace signálu X(t). Jak již bylo řečeno, snažíme se získat okamžité vlastnosti signálu - amplitudu, fázi a frekvenci. Tyto vlastnosti signálu Z(t) jsou definovány následovně:

$$a(t) = \sqrt{X(t)^2 + Y(t)^2} \tag{5.4}$$

$$\theta(t) = arctan\left(\frac{Y(t)}{X(t)}\right) \tag{5.5}$$

$$\omega(t) = \frac{d\theta(t)}{dt} \tag{5.6}$$

kde $a(t)$ je okamžitá amplituda, $\theta(t)$ je okamžitá fáze a $\omega(t)$ je požadovaná okamžitá frekvence [19].

5.4. Aplikace HHT na zpracování EEG

5.4.1. Vytvoření obálky EEG signálu

Když vytváříme obálku signálu, snažíme se jí pokrýt celý signál. Jenže několik počátečních a koncových bodů se nepočítá za lokální extrémy. Obálku definují až druhé nejbližší extrémy. Proto je nutné přidat další body (lokální extrémy), abychom pokryli celý signál. Tyto body se musí správně umístit, jinak obálka špatně kopíruje signál. Vzniká takzvaný overshoot nebo undershoot efekt. Tyto efekty nepopisují charakteristiky signálu a mohou být dále propagovány a znehodnotit tak celý signál.

5.4.2. Metody odhadu přidaných extrémů

Metodou zrcadlení (anglicky Mirror Method) se odhadují přidané extrémy na okrajích signálu. Nejdříve se nalezne první lokální extrém (například *Max(1)*) a jemu nejbližší extrém (*Min(1)*). Pak odhadnutý přidaný extrém *Min(0)* bude umístěn následovně:

$$Min_x(0) = Max_x(1) - \left(Min_x(1) - Max_x(1)\right) \tag{5.7}$$

$$Min_y(0) = Min_y(1) \tag{5.8}$$

Druhou metodou je tzv. metoda základního sklonu (anglicky Slope-Base Method). Tato metoda přidává jedno minimum a jedno maximum na začátek nebo konec signálu. Umístění nových extrémů je vypočítáno z matematicky definovaných sklonů vytvořených z extrémů. Tyto sklony jsou odvozeny od vzdálenosti mezi následnými minimy a maximy a z rozdílu amplitudy. Více o metodě v [19].

Obě metody odhadnou a přidají další extrémy tak, aby vytvořené obálky při výpočtu EMD kompletně pokryly signál. Jejich slabinou je ale odhad umístění těchto extrémů na časové ose. Tento problém nastává hlavně v případech, kdy okraje signálu obsahují krátkodobé komponenty o výrazně vyšší frekvenci (artefakty) [19].

6. Modifikovaná Hilbert-Huangova transformace

Základní algoritmus modifikované HHT zůstává nezměněný. Modifikovaná HHT používá jiný algoritmus odhadu přidaných extrémů na okrajích signálu, vylepšenou metodu detekce lokálních extrémů a jiný výpočet okamžitých vlastností signálu. Tyto modifikace výrazným způsobem zlepšují pokrytí signálu obálkami a minimalizují overshoot a undershoot efekty a tím výrazným způsobem zlepšují detekci ERP komponent v EEG signálu.

6.1. Metody detekce lokálních extrémů

6.1.1. Metoda inflexních bodů

Tato jednoduchá metoda detekuje inflexní body v EEG signálu. Testujeme vždy trojici po sobě jdoucích bodů signálu. Pokud je prostřední bod větší resp. menší než oba krajní body, je považován za lokální maximum resp. minimum. Jenže ne každý inflexní bod je extrémem. Některé inflexní body mohou vzniknout při předchozím zpracování signálu (filtrace, průměrování atd.). Amplituda takovýchto inflexních bodů je jen nepatrně vyšší/nižší než amplituda okolních bodů.

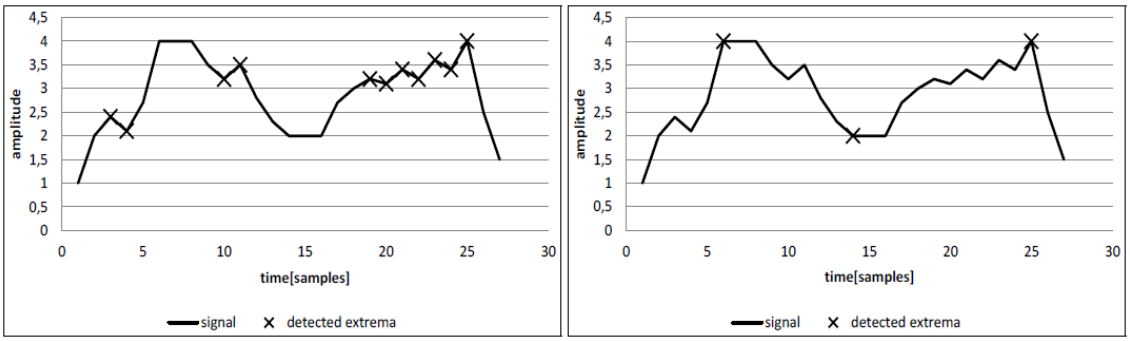
Některé extrémy nelze touto metodou detekovat. Pokud máme například dva body se stejnou amplitudou a jejich amplituda je vyšší/nižší než amplituda jejich okolních bodů, měl by být jeden z těchto bodů označen za lokální extrém. Ale protože tyto body nejsou inflexní, metoda žádný z nich neoznačí za extrém. [19]

6.1.2. Metoda delta-diference

Tato metoda na rozdíl od předchozí ignoruje extrémy s nepatrně větší/menší amplitudou než mají okolní body a dokáže detekovat extrémy i v případě, že se nejedná o inflexní bod. Metodu lze popsat následujícím algoritmem [19]:

1) Inicializace prahu δ a indexu vzorku $i = 0$
2) Inicializace základní hodnoty hodnotou prvního vzorku a $max_p = null$
3) Uložení amplitudy i-tého vzorku do $y = x[i]$
4) Pokud $y_i > (y_{i-1} - \delta)$, pak akceptuj i-tý vzorek jako nový potenciální extrém max_p
5) Pokud $(max_p - y_i) > \delta$, pak max_p je maximum, začni hledat další, $max_p = null$
6) Pokud $max_p = null$ a $základ < y_i$, pak $základ = y_i$
7) i++, pokračuj bodem 3)

Parametr δ je mírou tolerance pro malé fluktuace amplitudy. Pokud je parametr δ nastaven na nulu, pak metoda nalezne stejné extrémy jako metoda inflexních bodů [19]. Na obrázku 17 je vidět rozdíl mezi metodami v detekovaných extrémech. Je vidět, že extrémy nalezené touto metodou jsou vhodnější pro vytvoření obálky.



Obrázek 17: Detekované extrémy - vlevo metoda inflexních bodů, napravo metoda delta diference [19]

6.2. Modifikovaná metoda zrcadlení

Modifikovanou metodu zrcadlení popisuje následující algoritmus [19]:

1) Nalezení nejbližšího minima a maxima na začátku signálu
2) Vytvoření nového extrému, který bude předcházet začátku signálu a bude respektovat zrcadlovou symetrii jako v rovnicích [6.1] a [6.2]
3) Nalézt první extrém signálu. V tomto případě Max(0). To znamená, že signál měl před extrémem vzestupný směr.
4) Proto se musíme ujistit, že hodnota nového minima $Min_y(0)$ je menší nebo rovna první hodnotě signálu (x(0)). Pokud je minimum $Min_y(0)$ větší než první hodnota signálu, jednoduše nahradíme Min(0) bodem x(0).

$$Min_x(0) = -Min_x(1), \qquad Min_y(0) = Min_y(1) \tag{6.1}$$

$$Max_x(0) = -Max_x(1), \qquad Max_y(0) = Max_y(1) \tag{6.2}$$

Pro tuto metodu je důležité přesně detekovat extrémy v signálu. V opačném případě může metoda špatně odhadnout polohu nových extrémů. To by mělo za příčinu špatné pokrytí signálu obálkou a zhoršenou pozdější detekci ERP komponenty. Proto je vhodné používat k nalezení lokálních extrémů výše popsanou metodu delta diference.

6.3. Výpočet okamžité frekvence z analytického signálu

Mějme jednoduchou sinovou funkci s frekvencí 4Hz a amplitudou 2μV. Když vypočítáme fázový posun obyčejnou arctan funkcí, výsledek spadá do intervalu (-π/2, π/2) pro každou půlperiodu sinu. Potom rozdíl fázového posunu prvního a posledního bodu intervalu je (- π/2 - π/2) = - π. To znamená, že pokud máme záporný rozdíl dvou fázových posunů, dostaneme také zápornou okamžitou frekvenci. Jenže záporná frekvence je nesmyslná. [19]

Abychom odstranili tento problém, použijeme místo arctan funkce tzv. arctan2 funkci, která respektuje kvadranty. Jejím vstupem jsou dva argumenty – souřadnice x a y a vrací úhel mezi nimi ve správném kvadrantu v rozsahu od – π do π radiánů.

U funkce arctan2 dostaneme v místě přechodu mezi kvadranty ještě vyšší záporné hodnoty (-2π). Ale teď je snadné upravit záporné hodnoty. Můžeme předpokládat, že

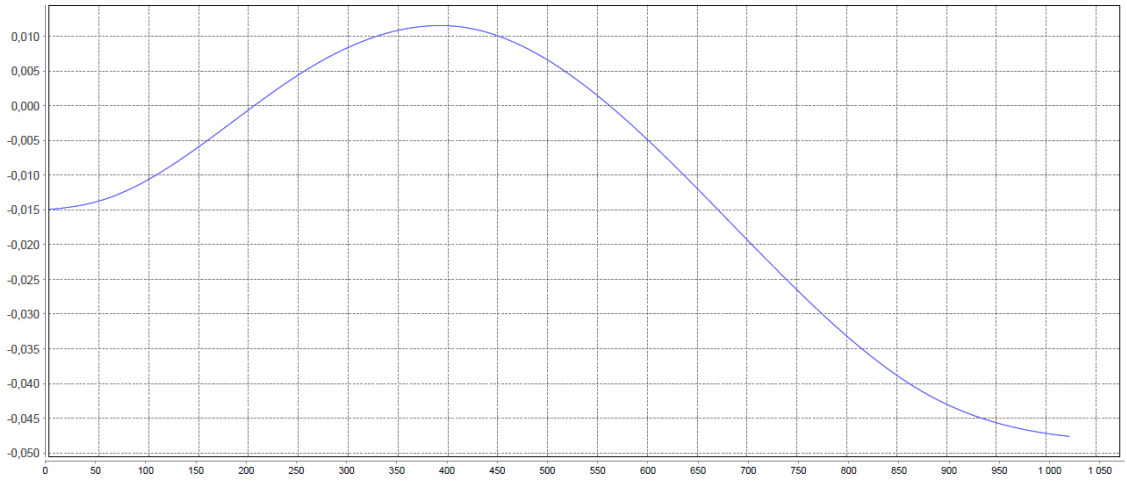
zpracovávaný signál má po celou dobu téměř stejnou frekvenci. Tento předpoklad je založen na vlastnostech IMF. Pokud je rozdíl fázových posunů blízko hodnotě -2π, můžeme předpokládat, že v tomto bodě přechází signál z jedné periody do druhé. V tomto případě můžeme nahradit zápornou frekvenci f(i) za průměr frekvencí f(i-1) a f(i+1). [19]

6.4. Další možné modifikace

V [19] byly popsány všechny výše uvedené modifikace a také nastíněn směr dalšího výzkumu. Jednou z možností je otestovat vliv dodatečných zastavovacích podmínek v procesu prosívání (viz 5.2.) na výběr IMF a na úspěšnost klasifikace. Konečnou fází celého procesu je klasifikace, tedy určení, zda daný signál obsahuje ERP komponentu. V této práci se pokusím detekovat P3 komponentu v reálných EEG datech a vytvořit k tomuto účelu nový klasifikátor.

7. Dodatečné zastavovací podmínky

Každá IMF musí mít střední hodnotu obálky nulovou v každém bodě. Tuto podmínku je složité striktně dodržet, a proto je nahrazena hodnotou normálního rozdělení nebo Cauchyho testu konvergence. IMF splňující tuto novou podmínku nemusí být ideální. Je možné, že by se v některé z dalších iterací dala najít lepší IMF, která by lépe vystihovala trend signálu a zlepšila tak pozdější klasifikaci. Proto jsem se pokusil otestovat dvě dodatečné zastavovací podmínky. První dodatečná podmínka je založena na normálním rozdělení a druhá na průměrné hodnotě průměrné křivky. V praxi je pak funkce v aktuální iteraci prohlášena za IMF, pokud splňuje podmínku 2) z oddílu 5.2. a zároveň dodatečnou podmínku.



Obrázek 18: Průměrná křivka bez dodatečné podmínky

7.1. Průměrná vzdálenost průměrné křivky od nuly

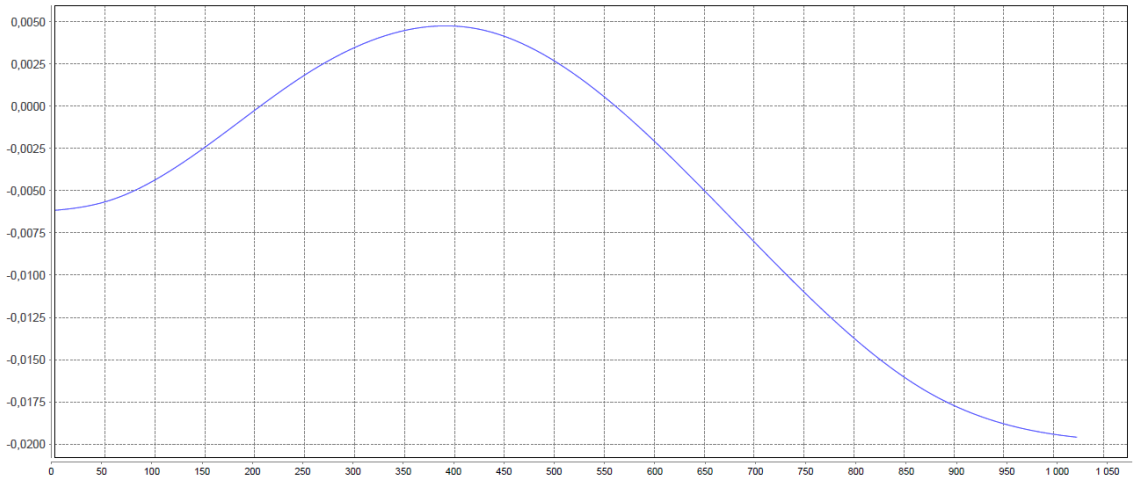
Normální rozdělení křivky lze jednoduše popsat jako průměrnou vzdálenost funkčních hodnot křivky od průměru:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N}} \tag{7.1}$$

Pokud za x_i dosadíme funkční hodnoty průměrné křivky obálek signálu, pak by měl být průměr roven nule pro každou IMF. Tento předpoklad vychází z podmínky, že střední hodnota obálky definované lokálními maximy a minimy je nulová v každém bodě. Pokud má být průměr obálek roven nule v každém bodě, měl by být nulový i celkový průměr obálek. Samozřejmě vlivem různých chyb (odhad přidaných extrémů, nalezení lokálních extrémů, numerické chyby atd.) nemusí být průměr přesně nula, ale měl by se k nule blížit. Za $\bar{x}$ proto dosadíme nulu a získáme tak průměrnou vzdálenost průměrné křivky od nuly:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} {x_i}^2}{N}} \tag{7.2}$$

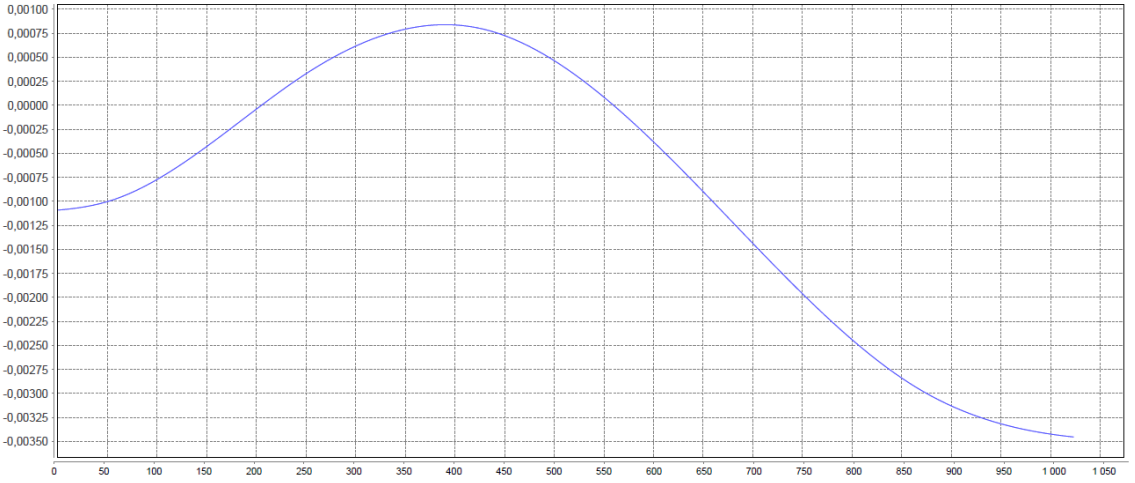
Kritérium je splněno ve chvíli, kdy tato hodnota je menší než zvolený práh.



Obrázek 19: Průměrná křivka pro test průměrné vzdálenosti s prahem 0.01

7.2. Průměrná hodnota průměrné křivky

Pokud má být střední hodnota obálek nulová v každém bodě, měl by být i průměrná hodnota průměrné křivky obálek rovna nule. Samozřejmě kvůli různým nepřesnostem nemusí být průměr přesně roven nule, ale měl by se nule blížit. Pokud je hodnota průměru menší než zadaný práh, je podmínka splněna.



Obrázek 20: Průměrná křivka pro test průměrné hodnoty s prahem 0,001

7.3. Porovnání kriterií

Může se zdát, že obě podmínky jsou si dost podobné, ale není tomu tak. Mějme například sinusovou křivku na intervalu 2π a její hodnoty x = [0; 0,5; 1; 0,5; 0; -0,5; -1; -0,5]. Její průměr je

$$\bar{x} = \frac{0{,}5 + 1 + 0{,}5 - 0{,}5 - 1 - 0{,}5}{8} = 0.$$

Ale hodnota průměrné vzdálenosti od nuly je

$$\sigma = \sqrt{\frac{3}{8}} = 0{,}61.$$

V tomto případě by byla druhá podmínka splněna, zatímco první pravděpodobně ne. Na první pohled se zdá být první dodatečná podmínka lepším kritériem, protože lépe popisuje tvar průměrné křivky. Jak je ale vidět na obrázcích 18, 19 a 20, tvar křivky se výrazně nemění. To je způsobeno tím, že v algoritmu siftingu odečítáme od signálu průměrnou křivku. V každé iteraci se pak naleznou extrémy v přibližně stejném místě na ose x a na ose y budou o něco blíže nule, obálky signálu budou mít přibližně stejný tvar a jejich průměrná křivka také. Mění se hlavně meze, ve kterých se křivka pohybuje. U obrázku 18 (bez dodatečné podmínky) jsou mez přibližně v rozsahu od -0.0475 do 0,012, u obrázku 19 (průměrná vzdálenost od nuly) od -0,02 do 0,005 a na obrázku 20 (průměrná hodnota průměrné křivky) od -0,0035 do 0,0009. Obě dodatečné podmínky pomáhají v závislosti na hodnotě prahu striktněji dodržet nulovou střední hodnotu obálky v každém bodě.

Tabulka 1 sleduje vliv dodatečných podmínek na hledání IMF. V prvním sloupci je název podmínky, ve druhém hodnota prahu, ve třetím průměrný počet iterací, ve čtvrtém sloupci je čas nutný k nalezení jedné IMF a v posledním sloupci je průměrný počet nalezených IMF. Vliv dodatečných podmínek byl testován na 40 různých datech a výsledky zprůměrovány. Z naměřených výsledků je patrné, že se snižujícím se prahem roste nejen počet iterací algoritmu, ale i počet nalezených IMF.

Podmínka	Práh	Iterace	Čas [ms]	Počet IMF
Bez dodat. podmínky	-	29,08	12,68	6,95
Průměrná vzdálenost průměrné křivky od nuly	0,1	29,12	12,52	6,95
Průměrná vzdálenost průměrné křivky od nuly	0,01	30,54	12,79	7,075
Průměrná vzdálenost průměrné křivky od nuly	0,001	104,61	36,12	8
Průměrná hodnota průměrné křivky	0,01	29,16	12,34	6,95
Průměrná hodnota průměrné křivky	0,001	34,64	13,92	7,025
Průměrná hodnota průměrné křivky	0,0001	55,25	20,52	7,425

Tabulka 1: Vliv dodatečných podmínek na hledání IMF

8. Popis implementace

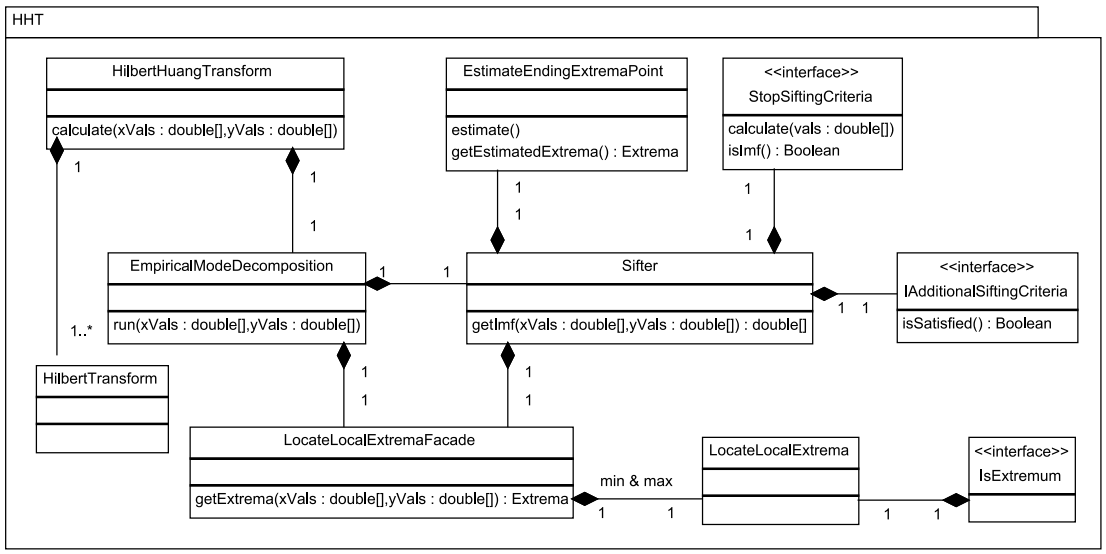
Autor původní implementace se snažil o co nejlepší modularitu programu. Proto je snadné vyměnit různé části implementace bez větších zásahů do kódu programu. Jednotlivé moduly logicky kopírují algoritmus. V této kapitole budou popsány jen ty nejdůležitější, nově vytvořené nebo změněné části programu.

Implementaci tvoří i celá řada konfiguračních souborů. Jejich nevýhodou je, že pokud se změní některé části kódu, musí se změnit vždy několik konfiguračních souborů. Na druhou stranu umožňují uživateli více kontrolovat jednotlivé výpočetní části algoritmu a nastavovat velké množství parametrů jak algoritmu HHT, tak i následné klasifikace.

Implementace HHT je psaná v jazyce Java. Jazyk byl zvolen původním tvůrcem s ohledem na další projekty, kde bylo plánované využití HHT, a které jsou většinou v tomto programovacím jazyce. Program je designován jako knihovna využitelná v dalších projektech, ale obsahuje i třídy pro snadné spuštění z konzole.

8.1. Modul hht

Celý algoritmus HHT je oddělen od klasifikace. Po výpočtu HHT se výsledky uloží do souboru a klasifikátor je pak pro potřeby klasifikace znovu načte. Na obrázku 18 je vidět diagram tříd. Diagram zobrazuje jen nejdůležitější třídy.



Obrázek 21: diagram tříd HHT algoritmu

8.1.1. Třída HilbertHuangTransform

Je hlavní třídou celého algoritmu. Nejdůležitější metodou této třídy je metoda `calculate(…)`, která nejdříve inicializuje a spustí výpočet EMD, a poté na každou získanou IMF aplikuje Hilbertovu transformaci.

Nejsnazší způsob jak spustit celý algoritmus HHT je zavolat jednu z metod třídy `HhtSimpleRunner`, které spustí HHT. Jedním parametrem je i cesta ke

konfiguračnímu souboru, který obsahuje konfiguraci HHT algoritmu. Po skončení výpočtu HHT třída uloží výsledky do souborů.

8.1.2. Třída EmpiricalModeDecomposition

Základní třída, která řídí dekompozici vstupního signálu na IMF. Třída vyžaduje instanci tříd `Sifter` a `LocateLocalExtremaFacade`. Proces dekompozice spouští metoda `run(…)`. Metoda ve while cyklu nejdříve získá jednu IMF, uloží ji a nakonec odečte IMF od signálu z předešlé iterace. Cyklus končí, pokud je počet nalezených extrémů roven nule. Výstupem třídy je seznam polí s funkčními hodnotami IMF.

8.1.3 Třída Sifter

Třída hledá ve vstupním signálu IMF. K nalezení IMF potřebuje `Sifter` instance tříd `EstimateEndingExtremaPoints`, `StopSiftingCriteria`, `LocateLocalExtremaFacade` a volitelně (může být `null`) `IAdditionalSiftingCriteria`. Hlavní je metoda `getImf(double[] xVals, double[] yVals)`, která ve while cyklu nalezne lokální extrémy ve vstupním signálu, odhadne přidané extrémy a pak vytvoří horní a dolní obálku signálu. Nakonec se odečte průměrná křivka vytvořená z horní a dolní obálky od signálu. Cyklus končí, pokud je detekována IMF. Metoda pak vrací IMF jako pole jejích funkčních hodnot.

O rozhodnutí, zda je funkce v aktuální iteraci IMF, se stará metoda `isImf()`, která vrací `true`, pokud se jedná o IMF. Aby byla funkce IMF, musí být hodnota jejího normálního rozdělení nebo Cauchyho testu konvergence menší než zadaná prahová hodnota. O to se stará metoda `IsImf()` třídy `StopSiftingCriteria`. Nová implementace nekontroluje první podmínku popsanou v oddílu 5.2. protože tato podmínka podle [21] vychází z druhé podmínky. Pokud instance třídy implementující rozhraní `IAdditionalSiftingCriteria` není `null`, metoda kontroluje, zda je splněna dodatečná podmínka. V tomto případě musí být splněny obě podmínky (funkce je IMF a zároveň hodnota dodatečné podmínky je menší než práh). Některé IMF mohou být označeny za nevyhovující a proces siftingu bude pokračovat dalšími iteracemi, dokud nebudou splněny obě podmínky.

8.1.4. Třída HilbertTransform

Výstupem EMD je seznam IMF. Z každé IMF je nutné vypočítat analytický signál, tedy provést Hilbertovu transformaci. O to se stará třída `HilbertTransform`, která vypočítá analytický signál ze vstupní IMF a poté i okamžité vlastnosti signálu (viz oddíly 5.3.1. a 6.3.). Výstupem je pole okamžitých frekvencí, amplitud a fází. Tyto okamžité atributy signálu jsou většinou zapsány do souboru pro potřeby klasifikace.

8.2. Modul testing

Nejdůležitější součástí tohoto modulu je systém klasifikátorů, kterých je zde implementováno několik. Klasifikátory jsou spouštěny třídou implementující rozhraní `ClassificatorRunner`. Taková třída musí implementovat metodu `processAllSubDirs`, která zjistí, jestli v uživatelem definovaném adresáři existují data z HHT (soubor s okamžitými frekvencemi a amplitudami), a pak spustí klasifikaci

s danými klasifikátory. O uložení výsledků se starají třídy implementující rozhraní `ClassificationDataStorage`. Výsledky se většinou ukládají do HTML souborů.

8.3. Logování, vizualizace, ukládání výsledků

Projekt využívá k logování log4j knihovnu, která umožňuje spouštět implementaci s různým stupněm logování. Výsledky pro jednotlivá data jsou ukládány do samostatné složky. Každá taková složka obsahuje pdf soubory s vizualizovanými IMF, mapou IMF a originálním signálem. Dále jsou ve složce textové soubory s amplitudami, frekvencemi a logy. Výsledky klasifikace jsou ukládány do HTML souborů do samostatné složky definované v konfiguračním souboru klasifikátoru.

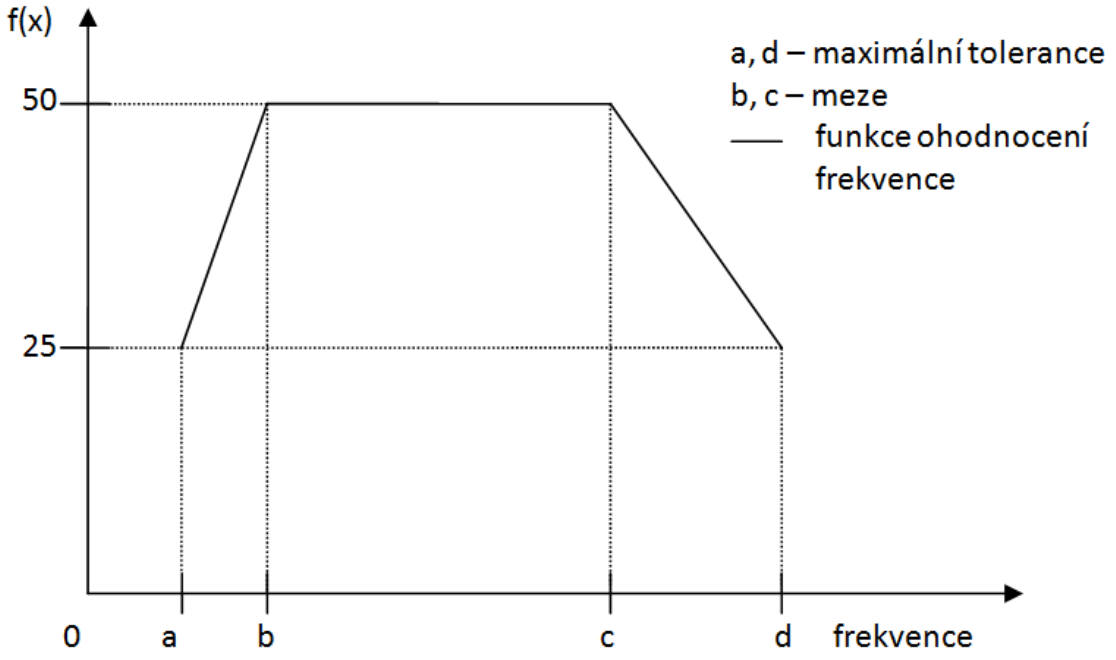
9. Klasifikace

Jedním z hlavních cílů diplomové práce je vytvořit nový klasifikátor, který bude dávat lepší a stabilnější výsledky než stávající klasifikátory. Výsledkem HHT jsou soubory s uloženými okamžitými frekvencemi a amplitudami jednotlivých IMF. Proto klasifikátor může pro klasifikaci používat pouze tyto příznaky nebo příznaky vytvořené z okamžitých frekvencí a amplitud.

Původní knihovna obsahuje čtyři klasifikátory, z nichž výrazně nejlepší výsledky dává klasifikátor `FreqAmplTreshold`, který byl použit jako výchozí pro nové klasifikátory. Vstupem klasifikátoru jsou uložené okamžité frekvence a amplitudy. Klasifikátor vyžaduje zadání rozsahu frekvence a práh amplitudy pro klasifikaci. Tento klasifikátor vždy zpracovává okamžité frekvence jedné IMF v definovaném rozsahu a vypočítá průměrnou okamžitou frekvenci. Pokud je frekvence uvnitř zadaných mezí, klasifikátor vypočítá průměrnou okamžitou amplitudu stejné IMF ve stejném rozsahu, a pokud je větší než zadaný práh, prohlásí ERP komponentu za nalezenou.

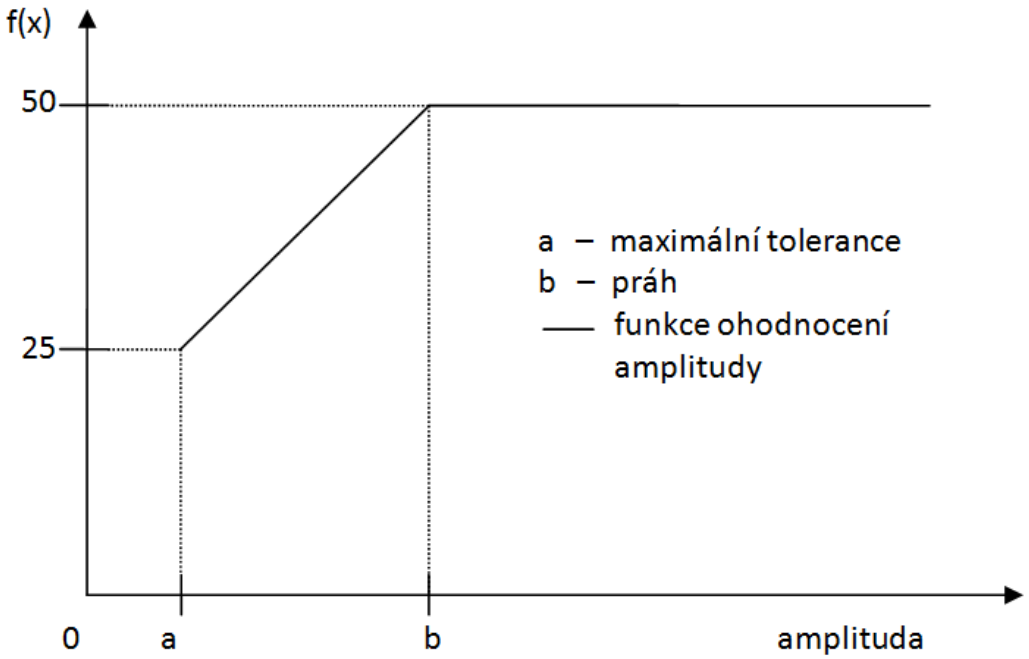
9.1. Váhový klasifikátor

Váhový klasifikátor vychází z výše popsaného klasifikátoru a dále ho vylepšuje. Původní klasifikátor má definovaný rozsah frekvencí, kterých může ERP komponenta dosahovat, a práh amplitudy, kterou musí komponenta s frekvencí v zadaném rozsahu překročit, aby byla v signálu detekována. Klasifikátor tak ostře vymezuje vlastnosti ERP komponenty. Jenže vlastnosti ERP komponenty jsou velmi rozdílné u různých lidí a je tak těžké přesně určit hranice, kdy část signálu s určitými vlastnostmi prohlásit za ERP komponentu či ne.



Obrázek 22: Funkce ohodnocení frekvence

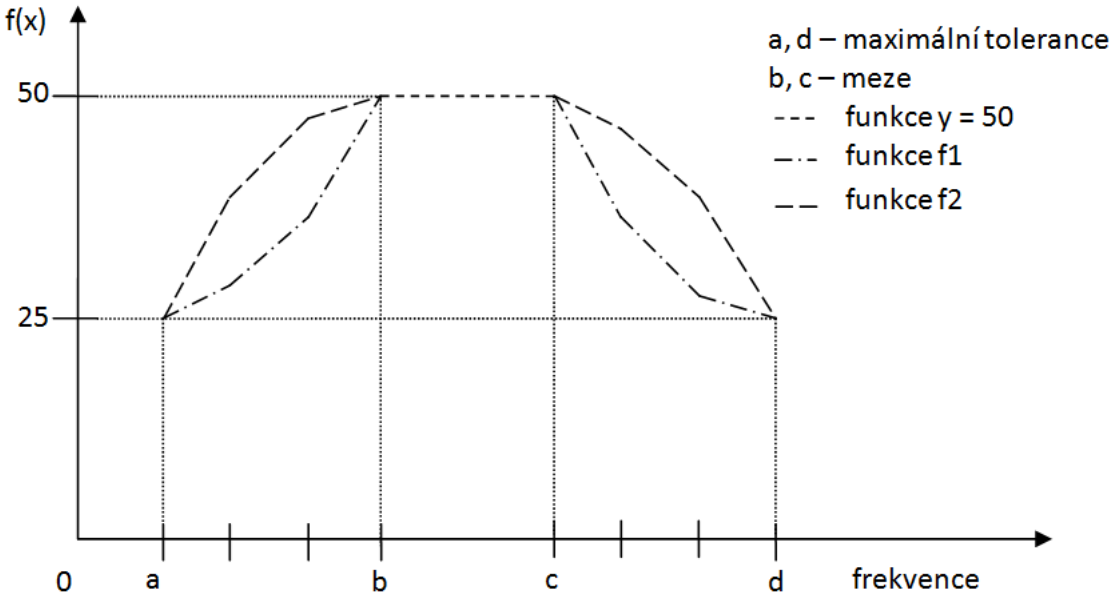
Váhový klasifikátor klasifikuje ERP komponenty podle dvou příznaků – průměrné frekvence a průměrné amplitudy na zadaném intervalu. Tento klasifikátor má kromě základních mezí zadanou i toleranci, při které lze uvažovat, že by se mohlo jednat o ERP komponentu. Z testování vyšlo, že vzdálenost |mez - tolerance| by neměla být větší než 0,7, protože čím bude vzdálenost větší, tím bude pravděpodobnější, že klasifikátor chybně detekuje komponentu v signálu, kde ve skutečnosti není. Váhový klasifikátor přiděluje každému příznaku body (váhu) v závislosti na jeho hodnotě a zvolené váhové funkci. Pokud je každý příznak v základních mezích, je ohodnocen padesáti body, pokud je od základních mezí vzdálen maximálně o hodnotu tolerance, je ohodnocen jen poměrnou částí bodů (mezi 25 a 50) v závislosti na zvolené funkci, jinak je přiděleno nula bodů. Komponenta je detekována, pokud součet bodů za jednotlivé příznaky překročí prahovou hodnotu definovanou uživatelem. Na obrázku 22 je znázorněno ohodnocení příznaku za frekvenci a na obrázku 23 za amplitudu v případě lineárního přidělení bodů mezi tolerancí a mezí.



Obrázek 23: Funkce ohodnocení amplitudy

Na úspěšnost klasifikace má vliv tvar funkce na intervalech <a, b> u amplitudy a frekvence a <c, d> u frekvence. V základním nastavení jsou použity dvě lineární funkce. První je rostoucí a na intervalu <a, b> je v rozsahu funkčních hodnot <25, 50>, druhá je klesající a na intervalu <c, d> je v rozsahu funkčních hodnot <50, 25>. Výsledná funkce pro ohodnocení frekvence může být vyjádřena následovně:

$$f(x) = \begin{cases} \frac{25 * (x - a)}{b - a} + 25, & x \in < a, b > \\ 50, & x \in (b, c) \\ \frac{-25 * (x - c)}{d - c} + 50, & x \in < c, d > \\ 0, & jinak \end{cases} \quad (9.1)$$



Obrázek 24: Další možnosti ohodnocení příznaků

Někdy by mohlo být výhodnější přidělovat hodnotám amplitud nebo frekvencí v toleranci více (popřípadě méně) bodů než u předchozí funkce. Místo použití složitějších nelineárních funkcí jsem zvolil postup, kdy je interval rozdělen na třetiny a v každé třetině je použita jiná lineární funkce (viz obrázek 24). Na obrázku 24 jsou vidět dvě použité funkce f1 a f2 pro ohodnocení frekvence. Ohodnocení amplitudy probíhá obdobně. Funkce f1 více zvýhodňuje hodnoty příznaku v mezích tolerance než předchozí funkce, u f2 je tomu naopak. U funkce f1 může častěji docházet k chybnému nedetekování ERP komponenty v signálu, než v případě funkce [9.1]. U funkce f2 může naopak častěji docházet k chybné detekci ERP komponenty v části signálu, kde se nevyskytuje. Funkce f1 a f2 na intervalu <a, b> lze vyjádřit následovně:

$$f1(x) = \begin{cases} \frac{11 * (x-a)}{k} + 25, & x \in < a, a+k > \\ \frac{9 * (x-k-a)}{k} + 36, & x \in < a+k, a+2k > \\ \frac{5 * (x-2k-a)}{k} + 45, & x \in < a+2k, b > \end{cases} \quad (9.2)$$

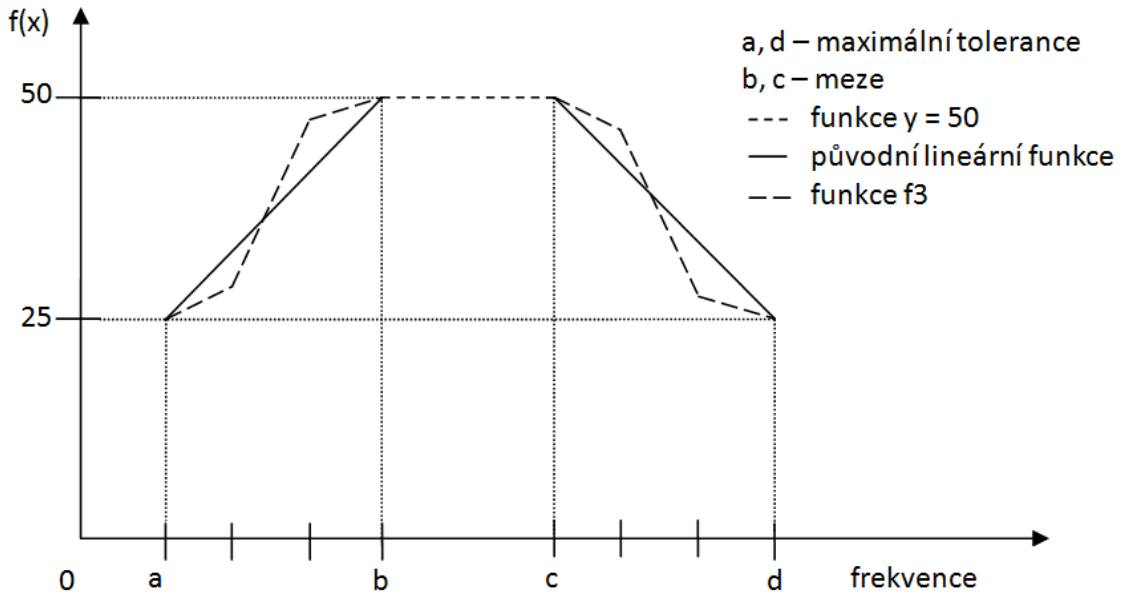
$$f2(x) = \begin{cases} \frac{5 * (x-a)}{k} + 25, & x \in < a, a+k > \\ \frac{7 * (x-k-a)}{k} + 30, & x \in < a+k, a+2k > \\ \frac{13 * (x-2k-a)}{k} + 37, & x \in < a+2k, b > \end{cases} \quad (9.3)$$

kde $k = \frac{1}{3}(b - a)$. Na intervalu <c, d> vypadají funkce f1 a f2 následovně:

$$f1(x) = \begin{cases} \frac{-5 * (x - c)}{l} + 50, & x \in < c, c + l > \\ \frac{-9 * (x - l - c)}{l} + 45, & x \in < c + l, c + 2l > \\ \frac{-11 * (x - 2l - c)}{l} + 36, & x \in < c + 2l, d > \end{cases} \quad \textbf{(9.4)}$$

$$f2(x) = \begin{cases} \frac{-13 * (x - c)}{l} + 50, & x \in < c, c + l > \\ \frac{-7 * (x - l - c)}{l} + 37, & x \in < c + l, c + 2l > \\ \frac{-5 * (x - 2l - c)}{l} + 30, & x \in < c + 2l, d > \end{cases} \quad \textbf{(9.5)}$$

kde $l = \frac{1}{3}(d - c)$. Funkce $f1(x) = 0$ *a* $f2(x) = 0$ pro $x \in (0, a) \cup (d, \infty)$ a $f1(x) = 50$ *a* $f2(x) = 50$ pro $x \in (b, c)$.



Obrázek 25: Funkce f3

Výhodným kompromisem mezi funkcemi f1 a f2 je funkce f3, která zvýhodní hodnoty příznaku blízko mezím (nebo prahu) a naopak znevýhodní hodnoty blízko toleranci. Taková funkce je zobrazena na obrázku 25. Její výhodou by měla být menší šance chybného nalezení ERP komponenty jako u funkce f2 a chybného nenalezení jako je tomu u funkce f1 z obrázku 24. Následuje předpis funkce f3 na intervalu <a, b>:

(9.6)

$$f3(x) = \begin{cases} \frac{5 * (x-a)}{k} + 25, & x \in < a, a+k > \\ \frac{15 * (x-k-a)}{k} + 30, & x \in < a+k, a+2k > \\ \frac{5 * (x-2k-a)}{k} + 45, & \in < a+2k, b > \end{cases}$$

kde $k = \frac{1}{3}(b-a)$, a na intervalu <c, d>:

$$f3(x) = \begin{cases} \frac{-5 * (x-c)}{l} + 50, & x \in < c, c+l > \\ \frac{-15 * (x-l-c)}{l} + 45, & x \in < c+l, c+2l > \\ \frac{-5 * (x-2l-c)}{l} + 30, & x \in < c+2l, d > \end{cases} \quad \textbf{(9.7)}$$

kde $l = \frac{1}{3}(d-c)$. Funkce $f3(x) = 0$ pro $x \in (0, a) \cup (d, \infty)$ a $f3(x) = 50$ pro $x \in (b, c)$.

Pro všechny výše zmíněné funkce platí, že hodnota tolerance a je z intervalu $(0, b >$. Tolerance a nemůže být záporná, protože pak bychom uvažovali, že se v signálu vykytují záporné frekvence. Záporná frekvence samozřejmě nedává smysl. Hodnota tolerance d je větší nebo rovna c a je z intervalu $(c, c+1 >$. Vyšší hodnoty tolerance nemá smysl uvažovat, protože pak bychom chybně detekovali velké množství ERP komponent. Pro parametry a, b, c a d platí 0 <= a <= b < c <= d <= c+1. Pokud platí a = b a c = d, pak dává klasifikátor stejné výsledky jako původní `FreqAmplTreshold` klasifikátor.

Body za jednotlivé příznaky jsou sečteny a porovnány s klasifikačním prahem. Pokud je součet větší než práh, ERP komponenta je nalezena. Hodnotu prahu je dobré volit větší nebo rovnu 75, protože pokud je v nejhorším případě jeden příznak roven toleranci (a nebo d), druhý musí být v mezích. Při nízké hodnotě prahu budou samozřejmě často chybně detekovány komponenty v signálu, kde ve skutečnosti nejsou.

9.1.1. Implementace

Váhový klasifikátor implementuje třída `WeightClassificator` v balíku `classificators`. Třída dědí od abstraktní třídy `AbstractFreqAmplSingle-Classificator` a překrývá její metodu `containsERP()`, která vrací `true`, pokud byla nalezena ERP komponenta v signálu, jinak vrací `false`. Metoda `contains-ERP()` implementuje následující algoritmus:

1) Inicializace čítače i = 0
2) Načtení okamžitých frekvencí i-té IMF v zadané oblasti
3) Výpočet průměrné frekvence
4) Bodové ohodnocení průměrné frekvence funkcí
5) Pokud získáno 0 bodů

    a. pokud je i < počet IMF, i++ a pokračuj bodem 2)
    b. jinak komponenta není detekována, return false
6) Načtení okamžitých amplitud i-té IMF v zadané oblasti
7) Výpočet průměrné amplitudy
8) Bodové ohodnocení průměrné amplitudy funkcí a přičtení k ohodnocení za frekvenci
9) Pokud je počet bodů <= prahu
    a. pokud je i < počet IMF, i++ a pokračuj bodem 2)
    b. jinak komponenta není detekována, return false
10) Komponenta nalezena, return true

Klasifikátor má hodně možností nastavení, proto je nejsnazší pustit ho s konfiguračním souborem, který je přiložen a v něm měnit hodnoty parametrů. Část konfiguračního souboru s nastavením klasifikátoru může vypadat následovně:

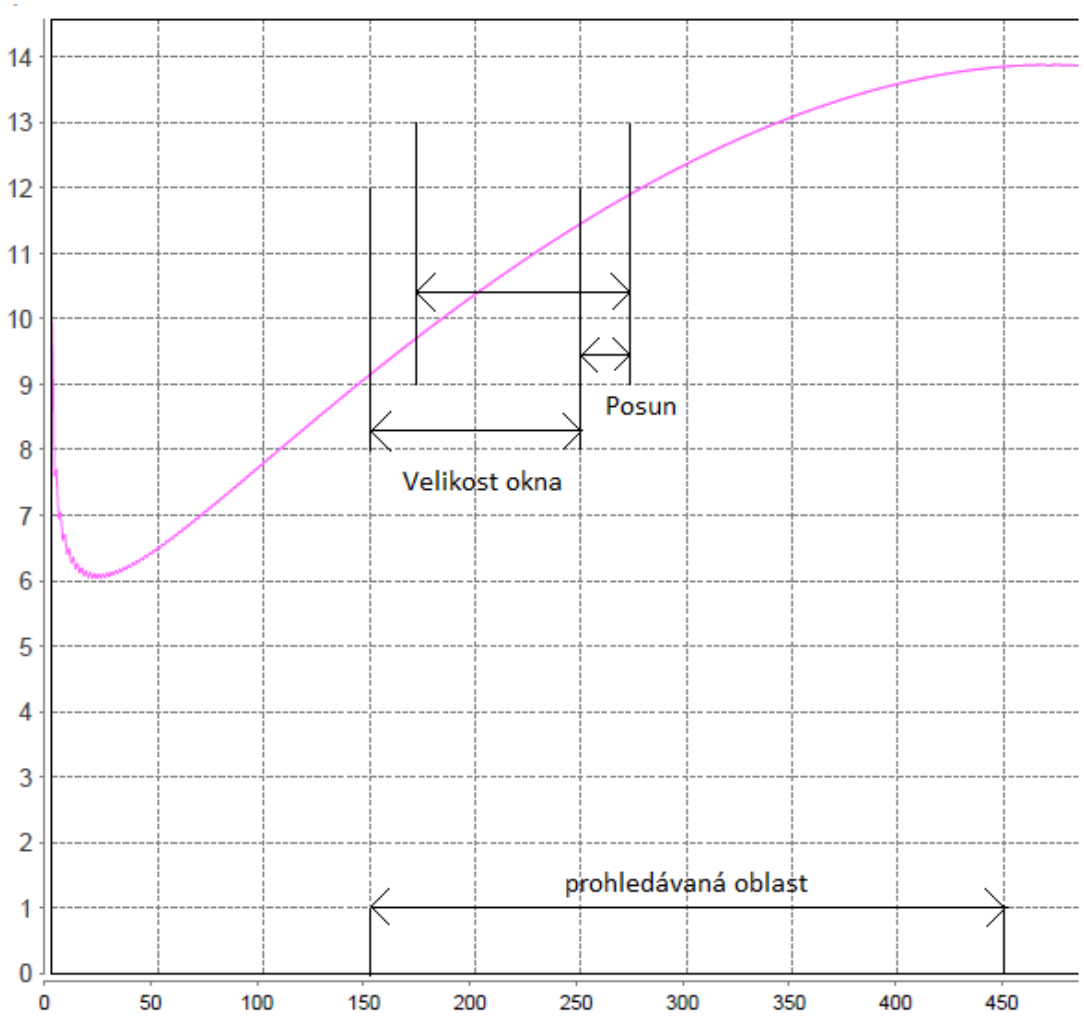
```
<bean id="weight" class="testing.classificators.WeightClassificator">
        <property name="freqMeanFrom" value="0.2"/>
        <property name="freqMeanFromTol" value="0.1"/>
        <property name="freqMeanTo" value="3"/>
        <property name="freqMeanToTol" value="3.5"/>
        <property name="fromSample" value="150"/>
        <property name="toSample" value="650"/>
        <property name="amplMeanThreshold" value="3.0"/>
        <property name="amplMeanThresholdTol" value="2.5"/>
        <property name="classificationTreshold" value="80"/>
        <property name="id" value="WeightClassificator"/>
        <property name="amplZet" value="true"/>
        <property name="leftZet" value="true"/>
        <property name="rightZet" value="true"/>
 </bean>
```

Parametry v konfiguračním souboru odpovídají těmto parametrům klasifikátoru:

- *freqMeanFromTol* je tolerance frekvence a
- *freqMeanFrom* je mez frekvence b
- *freqMeanTo* je mez frekvence c
- *freqMeanToTol* je tolerance frekvence d
- *amplMeanThresholdTol* je tolerance amplitudy a
- *amplMeanThreshold* je práh amplitudy b
- *fromSample*, *toSample* definují od jakého do jakého vzorku budou načítány okamžité frekvence a amplitudy
- *classificationTreshold* je práh bodů pro detekci komponenty
- *id* je název klasifikátoru
- Poslední tři možná nastavení definují, jaká funkce se má použít v intervalu <a, b> u amplitudy a frekvence a <c, d> u frekvence. Nastavení je nepovinné, pokud

není zvoleno jinak, je použita lineární funkce z obrázků 22 a 23. Možná nastavení jsou:

- amplLinear je základní lineární funkce, amplSubinterval je funkce f1, amplSubinterval + amplExpLike je f2, amplZet je funkce f3 pro amplitudu
- leftLinear je základní lineární funkce, leftSubinterval je funkce f1, leftSubinterval + leftExpLike je f2, leftZet je funkce f3 pro frekvenci v intervalu <a, b>
- rightLinear je základní lineární funkce, rightSubinterval je funkce f1, rightSubinterval + rightExpLike je f2, rightZet je funkce f3 pro frekvenci v intervalu <c, d>



Obrázek 26: Posun okna v signálu

9.2. Okénkový klasifikátor

Okénkový klasifikátor také vychází z klasifikátoru `FreqAmplTreshold`. Původní klasifikátor průměruje hodnoty příznaků v poměrně velkém okolí (například od 150-650ms). Výhodnější by mohlo být zvolit menší okno pevné velikosti a s ním postupně prohledávat část signálu, kde by se mohla ERP komponenta vyskytovat (viz obrázek

26). Průměrné hodnoty v menším okně nebudou tolik zkreslené okolním signálem. ERP komponenta je detekována, pokud příznaky v okně jsou v definovaných mezích.

Není nutné okno posouvat po jednom vzorku, protože průměrná hodnota nebude několika málo vzorky výrazně ovlivněna. P3 komponenta se totiž nachází v signálu s nízkými okamžitými frekvencemi. Okénko je výhodné používat k vyhodnocení amplitudy. U frekvence je to zbytečné, protože v místě ERP komponenty je okamžitá frekvence přibližně konstantní. Ideálně by velikost okna měla být 150 – 300ms, posun okna 10 – 20ms a prohledávaná oblast mezi 100 – 700ms. Stejně jako předchozí klasifikátor se i tento nejsnáze nastaví konfiguračním souborem. Část konfiguračního souboru s nastavením klasifikátoru může vypadat následovně:

```
<bean id="window" class="testing.classificators.WindowClassificator">
        <property name="freqMeanFrom" value="0.2"/>
        <property name="freqMeanTo" value="3.0"/>
        <property name="fromSample" value="150"/>
        <property name="toSample" value="650"/>
        <property name="amplMeanThreshold" value="3.3"/>
        <property name="windowLen" value="300"/>
        <property name="shift" value="20"/>
        <property name="id" value="WindowClassificator"/>
 </bean>
```

Parametry v konfiguračním souboru odpovídají těmto parametrům klasifikátoru:

- *freqMeanFrom* je dolní mez frekvence
- *freqMeanTo* je horní mez frekvence
- *amplMeanThreshold* je práh amplitudy
- *fromSample*, *toSample* definují od jakého do jakého vzorku budou načítány okamžité frekvence a amplitudy
- *windowLen* je velikost okna
- *shift* je posun okna
- *id* je jméno klasifikátoru

9.3. Spuštění klasifikátorů

O spouštění klasifikátorů se starají třídy implementující interface `ClassificatorRunner`. Třídy musí implementovat jedinou metodu `processAllSubDirs(…)`, která prohledává adresářovou strukturu v zadaném místě a hledá složky obsahující data zpracovaná HHT s názvem odpovídajícím vzoru. Spouštěč klasifikátorů také většinou obsahuje seznam klasifikátorů, které budou použity. V cyklu pak předá cestu k datům jednotlivým klasifikátorům a spustí metodu `containsERP()`. Výsledky získané z jednotlivých klasifikátorů zpracuje a uloží pro následnou vizualizaci.

V původní knihovně jsou dvě implementace rozhraní `ClassificatorRunner` – `SimpleClassificatorRunner` a `MultiDataClassficatorRunner`. Cílem bylo

vytvořit novou implementaci, která by sjednotila výsledky klasifikátorů a zlepšila tak úspěšnost klasifikace.

9.3.1. ORClassificatorRunner

Nový spouštěč klasifikátorů se liší od původních tím, že sjednocuje výsledky jednotlivých klasifikátorů tak, aby se dosáhlo co nejlepší úspěšnosti klasifikace. Jinak se ale vyhodnocují signály obsahující target stimul a jinak signály s nontarget stimulem. Signál s nontarget stimulem neměl obsahovat ERP komponentu, zatímco signál s target stimulem ano. `ORClassificatorRunner` vyhodnocuje target a nontarget měření zvlášť. Pokud alespoň jeden spuštěný klasifikátor nalezne komponentu v target signálu, je komponenta nalezena. Pokud alespoň jeden spuštěný klasifikátor nenalezne komponentu v nontarget signálu, není komponenta detekována. V podstatě tento spouštěč klasifikátorů funguje také jako klasifikátor, jehož příznaky jsou výsledky jednotlivých klasifikátorů. `ORClassificatorRunner` má tu výhodu, že může eliminovat nevýhody jednotlivých klasifikátorů. Pokud jeden klasifikátor často nalézá komponenty v necílovém signálu a druhý naopak nenalézá komponenty v cílovém, použití tohoto spouštěče klasifikátorů může potlačit nevýhody obou klasifikátorů. Spouštět lze i stejný klasifikátor s různou konfigurací.

10. Integrace knihovny do softwaru Matlab

Software Matlab umožňuje spouštět příkazy javy, javovské knihovny i pracovat s javovskými objekty přímo z prostředí Matlabu od verze 2006b. Pro snadné spuštění implementace HHT přímo z Matlabu je nutné udělat několik kroků. Nejdříve je nutné zkopírovat implementaci HHT i s knihovnami, které využívá, do adresáře Matlabu například C:\Program Files\MATLAB\R2012a\java\jarext\hht. Pak je nutné přidat cestu ke knihovně do souboru classpath.txt. To lze jednoduše po zadání příkazu `edit classpath.txt` do konzole Matlabu. Do souboru přidáme nový řádek:

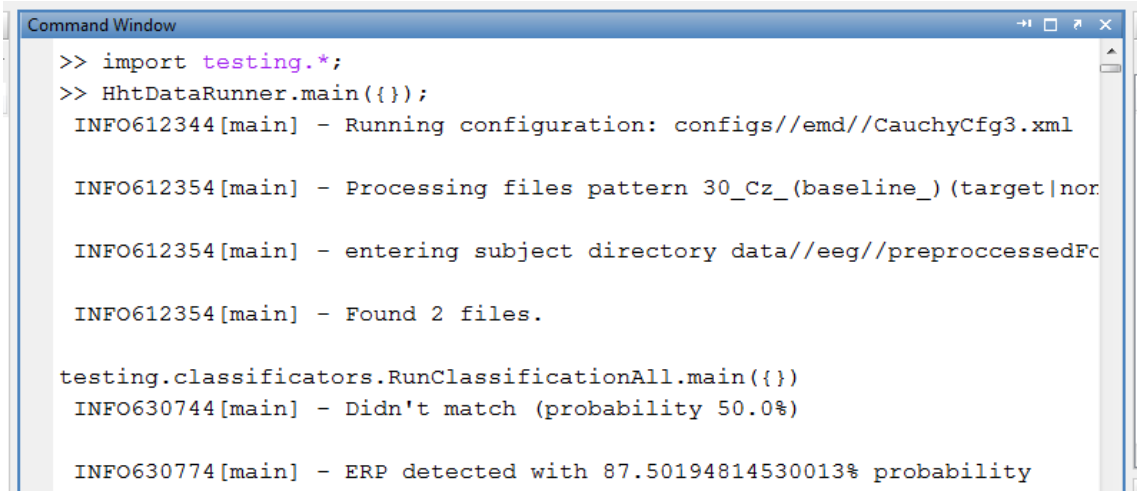
```
$matlabroot/java/jarext/hht/hht.jar
```

Naše implementace HHT využívá *commons-math* knihovnu ve verzi 2.1. Distribuce Matlabu také obsahuje tuto knihovnu, ale ve starší verzi. Proto je nutné ji přepsat na novější. Knihovna je v adresáři java\jarext v kořenovém adresáři Matlabu. Pro snadné spuštění knihovny jsou nutné konfigurační soubory. Aby je bylo možné využívat, musí se nakopírovat služka configs do pracovního adresáře Matlabu, například Documents\MATLAB. Nyní je možné plně využívat implementaci HHT v prostředí Matlabu (viz obrázek 27). Například spuštění HHT:

```
import testing.*;
HhtDataRunner.main({});
```

nebo spuštění klasifikace:

```
testing.classificators.RunClassificationAll.main({})
```



Obrázek 27: Spuštění HHT a klasifikace z konzole Matlabu

11. Testování

Veškeré testy byly prováděny se stejnými daty jako v [19]. K dispozici jsem měl měření od dvaceti lidí, od každého člověka jedno cílové a jedno necílové měření. Dohromady tedy 20 cílových a 20 necílových měření. Použitá data byla předzpracována průměrováním ze třiceti epoch a baseline korekcí.

11.1. Vliv dodatečných podmínek na úspěšnost klasifikace

Dodatečné podmínky by měly pomoci nalézt lepší IMF v procesu siftingu. IMF nalezené při použití dodatečných podmínek budou rozdílné od IMF nalezených bez jejich použití. Proto bude odlišný analytický signál vypočítaný z těchto IMF a budou se lišit i hodnoty okamžitých amplitud a frekvencí. To samozřejmě ovlivní i hodnoty příznaků a následnou klasifikaci.

Dodatečné podmínky byly testovány se třemi nejlepšími původními nastaveními původního `FreqAmplTreshold` klasifikátoru. Aby byly výsledky vypovídající, byla data opakovaně zpracována HHT s různou konfigurací EMD. Celkově tak byla HHT spuštěna šestkrát s různými zastavovacími podmínkami EMD - třikrát normální rozdělení a třikrát Cauchyho test konvergence (viz 5.2.1.). To znamená, že každá hodnota dodatečné podmínky byla klasifikována osmnáctkrát.

11.1.1. Průměrná hodnota průměrné křivky

V tabulce 2 je úspěšnost klasifikace pro různé hodnoty dodatečné podmínky průměrná hodnota průměrné křivky. V tabulce je zanesen počet zlepšení a zhoršení klasifikace a o kolik procent se maximálně zlepšila nebo zhoršila klasifikace jednoho z použitých klasifikátorů, dále úspěšnost nejlepšího klasifikátoru a průměrná schopnost klasifikace. Řádky jsou srovnané podle nejlepší klasifikace a druhotně podle průměrné klasifikace. Při použití dodatečné podmínky se změní výsledné okamžité amplitudy a frekvence. Ve výsledku to znamená změnu úspěšnosti klasifikace. V tabulce je vidět, že například pro hodnotu dodatečné podmínky 0,0005 se zhorší průměrná schopnost klasifikace a navíc došlo k častému zhoršení klasifikace, ale ve výsledku jsme schopni správně klasifikovat ERP komponentu o 5% lépe než bez dodatečné podmínky. Klasifikace bez použití dodatečné podmínky je s hodnotou 0,05 až na předposledním místě.

Hodnota podmínky	Počet zlepšení klasifikace	Maximální zlepšení [%]	Počet zhoršení klasifikace	Maximální zhoršení	Průměrná klasifikace [%]	Nejlepší klasifikace [%]
0,01	3	5	0	0	84,0278	92,5
0,005	4	5	2	2,5	84,0278	92,5
0,0005	5	5	9	7,5	81,9444	92,5
0,001	3	5	9	7,5	82,2222	90
0,05	0	0	0	0	83,3333	87,5
Bez podm.					83,3333	87,5
0,0001	3	2,5	11	7,5	81,3889	87,5

Tabulka 2: Úspěšnost klasifikace s dodatečnou podmínkou - průměrná hodnota průměrné křivky

V tabulce 3 jsou četnosti klasifikace s danou úspěšností. Pokud porovnáme počet zhoršení klasifikace z tabulky 2 a počet klasifikátorů s úspěšností 85% a vyšší z tabulky

3, zjistíme, že ke zhoršení klasifikace došlo většinou u méně úspěšných klasifikátorů. Naopak u nejlepších klasifikátorů došlo po použití dodatečné podmínky ke zlepšení klasifikace.

	Počet klasifikací s danou úspěšností				
Hodnota podmínky	85%	87,5%	90%	92,5%	Celkem
0,005	4	2	2	2	10
0,01	5	2	1	2	10
0,0005	3	2	1	1	7
0,001	4	2	1	0	7
0,05	5	5	0	0	10
Bez podm.	5	5	0	0	10
0,0001	4	2	0	0	6

Tabulka 3: Četnosti úspěšností klasifikace s dodatečnou podmínkou – průměrná hodnota průměrné křivky

11.1.2. Průměrná vzdálenost průměrné křivky od nuly

Tabulka 4 obsahuje totožné informace jako tabulka 2 pro různé hodnoty dodatečné podmínky průměrná vzdálenost průměrné křivky od nuly. Druhá dodatečná podmínka dává totožné výsledky jako předchozí. Průměrná klasifikace se pohybuje přibližně v rozmezí +- 1% od průměrné klasifikace bez použití podmínky, ale maximální klasifikace se zvýšila o 5% z 87,5% na 92,5%. Stejně jako u předchozí podmínky se původní klasifikace bez dodatečné podmínky umístila až na předposledním místě.

Hodnota podmínky	Počet zlepšení klasifikace	Maximální zlepšení [%]	Počet zhoršení klasifikace	Maximální zhoršení	Průměrná klasifikace [%]	Nejlepší klasifikace [%]
0,025	3	5	0	0	84,0278	92,5
0,0075	4	7,5	9	5	82,5000	92,5
0,05	3	2,5	0	0	83,7500	90
0,01	6	5	6	5	83,3111	90
0,005	6	5	8	12,5	82,0833	90
0,1	0	0	0	0	83,3333	87,5
Bez podm.					83,3333	87,5
0,001	3	5	8	5	82,6389	87,5

Tabulka 4: Úspěšnost klasifikace s dodatečnou podmínkou - průměrná vzdálenost průměrné křivky od nuly

Tabulka 5 udává četnosti klasifikace s danou úspěšností. V posledním sloupci je počet klasifikací s 85% a vyšší úspěšností. I když se například u hodnoty podmínky 0,01 zhoršila klasifikace v šesti případech, klasifikací s 85% a vyšší úspěšností je jen o jednu méně než u klasifikace bez použití dodatečné podmínky. To znamená, že zhoršení se opět týká pouze horších klasifikátorů, naopak u úspěšnějších klasifikátorů se schopnost klasifikace zvýšila.

	Počet klasifikací s danou úspěšností				
Hodnota podmínky	85%	87,5%	90%	92,5%	Celkem
0,025	5	2	1	2	10
0,0075	3	0	3	1	7
0,01	2	2	5	0	9
0,005	3	1	4	0	8
0,05	5	2	3	0	10
0,1	5	5	0	0	10
Bez podm.	5	5	0	0	10
0,001	8	4	0	0	12

Tabulka 5: Četnosti úspěšností klasifikace s dodatečnou podmínkou – průměrná vzdálenost průměrné křivky od nuly

11.1.3. Porovnání podmínek

Výhodou obou podmínek je, že ani v nejhorším případě nesníží maximální schopnost klasifikace ERP komponenty. Výsledek klasifikace tak bude lepší nebo stejný jako v případě bez použití dodatečné podmínky. Obě podmínky zhoršují klasifikaci u méně úspěšných klasifikátorů, zatímco u dobrých klasifikátorů dochází k výraznému zlepšení klasifikace až o 5%. V tabulce 6 je úspěšnost klasifikace nejlepšího klasifikátoru a nejlepších nastavení dodatečných podmínek. Pro srovnání je uveden jeden nejlepší klasifikátor při klasifikaci bez dodatečné podmínky. Zkratka MV v tabulce odpovídá první podmínce (průměrná hodnota průměrné křivky) a SD druhé podmínce (průměrná vzdálenost průměrné křivky od nuly). Z hodnot v tabulce je vidět, že všechny klasifikátory při použití dodatečné podmínky správně nedetekovaly o dvě komponenty více než v případě bez dodatečné podmínky. Obě podmínky dávají stejné výsledky a jsou si tak rovnocenné.

Podmínka	Hodnota podmínky	ERP správně detekována	ERP správně nedetekována	Úspěšnost	Úspěšnost [%]
MV	0,01	18/20	19/20	37/40	92,5
SD	0,025	18/20	19/20	37/40	92,5
MV	0,005	18/20	19/20	37/40	92,5
SD	0,0075	18/20	19/20	37/40	92,5
MV	0,0005	18/20	19/20	37/40	92,5
Bez podm.		18/20	17/20	35/40	87,5

Tabulka 6: Úspěšnost klasifikace nejlepších klasifikátorů při a bez použití dodatečné podmínky

11.2. Úspěšnost klasifikace nových klasifikátorů

Nové klasifikátory byly testovány stejně jako dodatečné podmínky na šesti konfiguracích EMD a výsledky porovnány se třemi nejlepšími původními nastaveními klasifikátoru `FreqAmplTreshold`. Pro všechny následující testy byla použita data zpracována HHT bez použití dodatečné podmínky.

11.2.1. Váhový klasifikátor

Váhový klasifikátor má mnoho možností nastavení. Lze vybírat ze čtyř funkcí pro tři parametry, měnit meze a tolerance parametrů, rozsah používaných vzorků signálu a práh klasifikace. Nastavení používaných klasifikátorů lze nalézt v příloze A.

V tabulce 7 je srovnání tří nejlepších konfigurací `FreqAmplTreshold` klasifikátoru (class1, class2, class3) a tří konfigurací váhového klasifikátoru (weight1l, weight2l, weight3l). V tomto případě byly u váhového klasifikátoru vždy použity lineární funkce podle obrázků 22 a 23 v sekci 9.1. Nastavení váhových klasifikátorů se lišila pouze ve třech parametrech následovně:

Weight1l:

```
<property name="amplMeanThreshold" value="3.0"/>
<property name="amplMeanThresholdTol" value="2.5"/>
<property name="classificationTreshold" value="75"/>
```

Weight2l:

```
<property name="amplMeanThreshold" value="3.2"/>
<property name="amplMeanThresholdTol" value="2.9"/>
<property name="classificationTreshold" value="75"/>
```

Weight3l:

```
<property name="amplMeanThreshold" value="3.3"/>
<property name="amplMeanThresholdTol" value="2.7"/>
<property name="classificationTreshold" value="80"/>
```

Výsledky testu hovoří jasně pro váhový klasifikátor, který má větší maximální, minimální i průměrnou úspěšnost klasifikace.

Klasifikátor	Maximální úspěšnost [%]	Minimální úspěšnost [%]	Průměrná úspěšnost [%]
Weight3l	90	82,5	87,0833
Weight1l	90	80	86,6667
Weight2l	90	82,5	86,2500
Class5	87,5	80	84,1667
Class4	87,5	77,5	83,7500
Class3	87,5	72,5	82,0833

Tabulka 7: Srovnání úspěšnosti klasifikace váhového klasifikátoru s původním klasifikátorem

Tabulka 8 sleduje vývoj úspěšnosti klasifikace váhového klasifikátoru s nastavením Weight3l při změně klasifikačního prahu. Hlavně v první části tabulky je vidět, že i jemná změna prahu může znamenat výraznější změnu úspěšnosti klasifikace (např. práh 79 a 80). Ideální hodnota prahu je v tomto případě od 78 do 82, protože v tomto rozmezí je nejvyšší maximální úspěšnost. Z testování i jiných nastavení klasifikátoru vyšly nejlépe hodnoty prahu mezi 75 a 83. Při volbě prahu by se mělo přihlížet ke vzdálenosti meze od tolerance. Čím je vzdálenost větší, tím by měl být větší i práh, aby nedocházelo k častým chybným detekcím ERP komponenty.

Práh	Maximální úspěšnost [%]	Minimální úspěšnost [%]	Průměrná úspěšnost [%]
75	87,5	82,5	86,2500
77	87,5	80	85,8333
78	90	80	85,4167
79	90	82,5	85,8333
80	90	82,5	87,0833
82	90	82,5	86,2500
85	87,5	82,5	85,0000
87	87,5	80	84,5833
90	87,5	80	84,1667
95	87,5	77,5	83,7500
99	87,5	77,5	84,5833

Tabulka 8: Vliv klasifikačního prahu na úspěšnost klasifikace

Na úspěšnost klasifikace váhového klasifikátoru má vliv i výběr váhové funkce. V tabulce 9 jsou úspěšnosti klasifikace váhového klasifikátoru s nastavením Weight3l při volbě různých váhových funkcí popsaných v sekci 9.1. Nejlépe vychází lineární funkce, která je použita v základu. Na lepší otestování by bylo nutné mít k dispozici více dat. Toto nastavení klasifikátoru bylo navíc původně vytvořeno jen se základní váhovou funkcí. Pro jiné funkce by bylo vhodné mírně změnit parametry a klasifikační práh, aby se dosáhlo stejné nebo případně i vyšší úspěšnosti.

Funkce	Maximální úspěšnost [%]	Minimální úspěšnost [%]	Průměrná úspěšnost [%]
Základní (lineární)	90	82,5	87,0833
f1	90	82,5	85,833
f3	90	82,5	85,833
f2	90	82,5	85,4167

Tabulka 9: Vliv váhové funkce na úspěšnost klasifikace

Největší vliv na úspěšnost klasifikace váhového klasifikátoru má volba prahu a tolerance amplitudy. I mírná změna vede k rapidnímu zlepšení nebo zhoršení klasifikace. Naopak mírná změna mezí a tolerancí frekvence neměla na klasifikaci téměř žádný vliv. Dalším zásadním parametrem, který ovlivní klasifikaci, je volba klasifikačního prahu.

11.2.2. Okénkový klasifikátor

Okénkový klasifikátor byl testován se třemi nastaveními. U všech nastavení byl posun okna o 10 vzorků. Nastavení klasifikátoru se lišila v hodnotě prahu amplitudy a velikosti okna následovně:

Window1:

```
<property name="amplMeanThreshold" value="3.15"/>
<property name="windowLen" value="225"/>
```

Window2:
<property name="amplMeanThreshold" value="3.25"/>
<property name="windowLen" value="200"/>
Window3:
<property name="amplMeanThreshold" value="3.4"/>
<property name="windowLen" value="200"/>

Výsledky testu úspěšnosti klasifikace okénkového klasifikátoru jsou v tabulce 10. Pro srovnání jsou uvedeny i výsledky tří nejlepších konfigurací `FreqAmplTreshold` klasifikátoru (class1, class2, class3). Okénkový klasifikátor dokáže klasifikovat data s maximální úspěšností 90%, zatímco původní klasifikátor pouze s 87,5% úspěšností. Vyšší je ve většině případů i minimální úspěšnost. Průměrná úspěšnost klasifikace je mírně vyšší než u původních klasifikátorů.

Klasifikátor	Maximální úspěšnost [%]	Minimální úspěšnost [%]	Průměrná úspěšnost [%]
Window1	90	80	85,8333
Window2	90	80	85,0000
Window3	90	80	84,5833
Class5	87,5	80	84,1667
Class4	87,5	77,5	83,7500
Class3	87,5	72,5	82,0833

Tabulka 10: Srovnání úspěšnosti klasifikace váhového klasifikátoru s původním klasifikátorem

Stejně jako u váhového klasifikátoru se i u okénkového výrazně neprojevuje na úspěšnosti klasifikace změna mezí frekvence. Klasifikaci výrazně ovlivňuje práh amplitudy a velikost okénka. V tabulce 11 je zaznamenán vliv velikosti okénka na úspěšnost klasifikace. Z tabulky je patrné, že velikost okna by se měla pohybovat kolem 175 až 250 vzorků. Původní klasifikátor počítal průměr amplitudy z přibližně 500 vzorků, což v některých případech zhoršilo klasifikaci.

Velikost okna	Maximální úspěšnost [%]	Minimální úspěšnost [%]	Průměrná úspěšnost [%]
150	87,5	80	84,5833
175	90	80	85,4167
200	90	82,5	85,4167
225	90	80	85,8333
250	90	80	85,4167
275	87,5	77,5	84,1667
300	87,5	77,5	83,7500

Tabulka 11: Vliv velikosti okna na úspěšnost klasifikace

11.2.3. Porovnání klasifikátorů

Oba nové klasifikátory převyšují svou klasifikační schopností původní klasifikátory. Jsou lepší v minimální, průměrné i v maximální úspěšnosti klasifikace, jak je vidět v tabulce 12. Žlutě jsou označeny nastavení váhového klasifikátoru, oranžově okénkového a modře původního klasifikátoru. Nejlépe si vede váhový klasifikátor. Je

možné, že váhový klasifikátor by při jiné konfiguraci podával ještě lepší výsledky, protože jsem zkoušel jen malou část kombinací různých parametrů. Průměrná úspěšnost klasifikace váhového klasifikátoru byla v průběhu testování různých nastavení téměř vždy (pro rozumné hodnoty parametrů) vyšší než 85%, zatímco u okénkového klasifikátoru dalo větší práci nastavit parametry tak, aby dával dobré výsledky.

Klasifikátor	Maximální úspěšnost [%]	Minimální úspěšnost [%]	Průměrná úspěšnost [%]
Weight3l	90	82,5	87,0833
Weight1l	90	80	86,6667
Weight2l	90	82,5	86,2500
Window1	90	80	85,8333
Window2	90	80	85,0000
Window3	90	80	84,5833
Class5	87,5	80	84,1667
Class4	87,5	77,5	83,7500
Class3	87,5	72,5	82,0833

Tabulka 12: Porovnání klasifikátorů

11.3. Výsledná klasifikace při použití dodatečných podmínek

Tabulka 13 obsahuje výsledný přehled úspěšnosti klasifikace EEG měření zpracovaných HHT s dodatečnými podmínkami i bez nich, klasifikovaných novými i původními klasifikátory. Žlutě je znázorněn váhový klasifikátor, oranžově okénkový a modře původní klasifikátor. Zkratka MV v tabulce odpovídá první dodatečné podmínce (průměrná hodnota průměrné křivky) a SD druhé dodatečné podmínce (průměrná vzdálenost průměrné křivky od nuly). Každý řádek v tabulce je souhrnem testu pro tři nastavení jednoho klasifikátoru, jednu dodatečnou podmínku, pokud byla použita, a šest různých nastavení EMD. Průměrná úspěšnost klasifikace je tak průměrem z osmnácti klasifikací a maximální a minimální úspěšnost je maximum respektive minimum z těchto osmnácti dílčích klasifikací. Tabulka je seřazena sestupně podle maximální, průměrné a minimální úspěšnosti. Z tabulky vyplývá, že použití nových klasifikátorů a zároveň dodatečné podmínky je nejlepší volbou. Klasifikace bez použití dodatečné podmínky a většina klasifikací s původním klasifikátorem se umístila až v druhé polovině tabulky. Velkou výhodou nových klasifikátorů je i rapidní zvýšení minimální úspěšnosti klasifikace o 5% až 10%.

Klasifikátor	Dodatečná podmínka	Hodnota podmínky	Maximální úspěšnost [%]	Minimální úspěšnost [%]	Průměrná úspěšnost [%]
Váhový	SD	0,01	95	77,5	88,0556
Okénkový	SD	0,01	95	80	86,9444
Okénkový	MV	0,01	95	80	85,6944
Váhový	SD	0,025	92,5	80	87,3611
Váhový	MV	0,01	92,5	80	87,0833
Váhový	SD	0,05	92,5	80	86,9444
Okénkový	SD	0,025	92,5	80	86,6667
Váhový	MV	0,005	92,5	77,5	86,6667
Okénkový	SD	0,005	92,5	77,5	86,3889
Váhový	SD	0,0075	92,5	75	86,3889
Okénkový	SD	0,0075	92,5	77,5	86,1111
Okénkový	MV	0,005	92,5	80	85,6944
Okénkový	SD	0,05	92,5	80	85,2778
Okénkový	MV	0,0001	92,5	77,5	84,3056
FreqAmplTreshold	MV	0,01	92,5	72,5	84,0278
FreqAmplTreshold	MV	0,005	92,5	72,5	84,0278
FreqAmplTreshold	SD	0,025	92,5	72,5	84,0278
Okénkový	MV	0,0005	92,5	75	83,8889
Váhový	MV	0,0005	92,5	75	83,75
FreqAmplTreshold	SD	0,0075	92,5	72,5	82,5
FreqAmplTreshold	MV	0,0005	92,5	75	81,9444
Váhový	Bez podm.		90	80	86,6667
Váhový	MV	0,05	90	80	86,6667
Váhový	MV	0,0001	90	82,5	85,5556
Okénkový	MV	0,001	90	80	85,4167
Váhový	SD	0,005	90	77,5	85,4167
Váhový	MV	0,001	90	77,5	85,2778
Okénkový	Bez podm.		90	80	85
Okénkový	MV	0,05	90	80	85
FreqAmplTreshold	SD	0,05	90	72,5	83,75
FreqAmplTreshold	SD	0,01	90	72,5	83,6111
FreqAmplTreshold	MV	0,001	90	72,5	82,2222
FreqAmplTreshold	SD	0,005	90	75	82,0833
Váhový	SD	0,001	87,5	77,5	84,0278
FreqAmplTreshold	Bez podm.		87,5	72,5	83,3333
FreqAmplTreshold	MV	0,05	87,5	72,5	83,3333
FreqAmplTreshold	SD	0,001	87,5	72,5	82,6389
FreqAmplTreshold	MV	0,0001	87,5	72,5	81,3889
Okénkový	SD	0,001	85	75	81,3889

Tabulka 13: Souhrn úspěšnosti klasifikace při použití dodatečných podmínek

11.4. Shrnutí testů

Výsledky testů ukázaly, že dodatečné podmínky pomáhají procesu siftingu nalézt kvalitnější IMF, a přiblížit se tak více její původní definici (oddíl 5.2.). Nalezením kvalitnější IMF ovlivníme výsledek Hilbertovy transformace. Vypočítané hodnoty okamžitých amplitud a frekvencí budou více odpovídat skutečnosti a to kladně ovlivní úspěšnost klasifikace. Obě podmínky zvyšují maximální úspěšnost klasifikace a jsou si v tomto ohledu rovnocenné.

Nově vytvořené klasifikátory poskytují vyšší maximální, minimální i průměrnou úspěšnost klasifikace nežli původní klasifikátory. Oba klasifikátory počítají pouze dva příznaky – průměrnou okamžitou amplitudu a frekvenci. V úvahu by připadalo rozšíření o nové příznaky. Po prozkoumání původních klasifikátorů jsem od rozšíření o další příznaky upustil. Jeden původní klasifikátor například používal jako další příznak čas vrcholu komponenty. Tento klasifikátor podával výrazně horší výsledky, protože vrchol komponenty není v poli okamžitých amplitud vždy jasně patrný.

Použitím dodatečných podmínek k nalezení IMF a následnou klasifikací novými klasifikátory jsme schopni výrazně zvýšit úspěšnost klasifikace. Výhodou je i vysoká průměrná úspěšnost klasifikace (až 88%) a i v nejhorším případě jsme schopni správně klasifikovat většinou 80% P3 komponent.

11.5. Práce do budoucna

Jak ukázaly výsledky mé práce, dodatečné podmínky jsou schopny pozitivně ovlivnit hledání IMF. Další vhodnou podmínkou by mohlo být například maximální hodnota průměrné křivky. I když osobně si myslím, že výsledky by byly obdobné, jako mají mnou implementované podmínky.

Výzkum by se měl zaměřit na další části algoritmu HHT, hlavně na odhad dodatečných extrémů. Použití arctan2 funkce pro určení okamžitých frekvencí částečně zamezilo generování záporných okamžitých frekvencí. Pokud je Hilbertovou transformací zpracovávána IMF obsahující vysoké frekvence, stále se často objevují záporné hodnoty okamžitých frekvencí. Proto by bylo vhodné tento problém odstranit.

Z implementačního pohledu je vhodné přepracovat a zjednodušit systém konfiguračních souborů a omezit ukládání a čtení z disku. Pro představu knihovna uložila pro data použitá k testování 305140 souborů v 100860 složkách o celkové velikosti 12,4GB.

12. Závěr

V rámci oborového projektu jsem se seznámil s metodami zpracování EEG signálu a detekce ERP komponent. V diplomové práci jsem se zaměřil na Hilbert-Huangovou transformaci, která je využívána ke zpracování nestacionárního signálu, kterým je i EEG. Ve své původní podobě algoritmus neumožňoval zpracovat EEG signál do té míry, aby bylo možné v signálu detekovat ERP komponenty s vysokou úspěšností. Proto byly navrženy modifikace HHT přímo pro EEG signál. V mé práci jsem se snažil modifikovaný algoritmus dále vylepšit.

Klíčovou fází HHT je EMD, která dekomponuje signál na IMF. Každá IMF musí mít střední hodnotu obálky definované lokálními maximy a minimy nulovou v každém bodě. Tuto podmínku je těžké striktně dodržet, a proto se v praxi používá k rozhodnutí, zda je funkce IMF, místo podmínky hodnota normálního rozdělení nebo Cauchyho test konvergence. Vybraná funkce většinou není ideální IMF. Abychom se více přiblížili nalezení ideální IMF, navrhl jsem a otestoval dvě dodatečné podmínky. Obě podmínky pomohly nalézt kvalitnější IMF, což ve výsledku vedlo k úspěšnější klasifikaci. Při hledání P3 komponenty byl nárůst maximální úspěšnosti klasifikace až o 5%.

K dispozici jsem měl implementaci HHT i testovací data Ing. Jindřicha Ciniburka Ph.D., který navrhl, implementoval a obhájil modifikace HHT v disertační práci [19]. S implementací jsem se seznámil, opravil drobné nedostatky, doplnil javadoc komentáře u důležitých metod a rozšířil implementaci o dodatečné podmínky a nový spouštěč klasifikace.

Dále jsem navrhl a implementoval dva nové klasifikátory, které využívají ke klasifikaci dvou příznaků – průměrné amplitudy a frekvence. Oba klasifikátory mají vyšší minimální, průměrnou a hlavně maximální úspěšnost klasifikace. Další zvýšení úspěšnosti klasifikace nastalo pro kombinaci dodatečných podmínek a nových klasifikátorů. Výsledkem je maximální úspěšnost klasifikace 95%. Nové klasifikátory i dodatečné podmínky byly testovány na reálných datech. Výsledný produkt lze jednoduše integrovat do prostředí softwaru Matlab a plně jej v něm využívat.

Literatura

[1] G. Schalk, P. Brunner, L.A. Gerhardt, H. Bischof, and J.R. Wolpaw. Brain-computer interfaces (BCIs): Detection instead of classification, *Journal of Neuroscience* Methods, číslo 167, 15 ledena 2008, str. 51-62.

[2] T. Rondik. Methods of ERP Signal Processing, diplomová práce. Západočeská universita v Plzni, 2010.

[3] T. Řondík. Methods for Detection of ERP Waveforms in BCI Systems, rigorózní práce, Západočeská universita v Plzni, 2012.

[4] S. J. Luck. An introduction to the Event-Related Technique, The MIT Press, Londýn, 2005. ISBN 0-262-12277-4

[5] Balakrishnama, S., Ganapathiraju, A. Linear Discriminant Analysis – a brief tutorial, Technical report, Institute for Signal and Information Processing, Department of Electrical and Computer Engineering, Mississippi State University.

[6] Lindsay I Smith, A tutorial on Principal Components Analysis, 26.2. 2002

[7] Hyvarinen Aj, Karhunen Oja E., editors. Independent Component Analysis. Wiley-Interscience, 2001

[8] S. Mallat. A wavelet tour of signal processing (Second edition). Academic Press, San Diego, 1999. ISBN 978-0-12-466606-1

[9] S. L. Marple. Computing the discrete-time 'analytic' signal via FFT, Conference Record of the Thirty-First Asilomar Conference on Signals, Systems & Computers, číslo 2, 1997, str. 1322-1325.

[10] P. Soukal. Methods for automatic detection of ERP components, diplomová práce, Západočeská universita v Plzni, 2010.

[11] A. Prochazka, E. Hostalkova. Biomedical signals and image processing using wavelet transform, *Automation: professional journal for automation, measurement and computer science engineering*, číslo 50, červen 2007, str. 397 – 401.

[12] T. Řondík, J. Ciniburk. Comparison of various approaches for P3 component detection using basic methods for signal processing, *2011 4th International Conference on Biomedical Engineering and Informatics*, New York: IEEE, 2010, str. 698-702. ISBN: 978-1-4244-9352-4

[13] T. Řondík, P. Mautner. Using ART2 for Clustering of Gabor Atoms Describing ERP P3 Waveforms, 2012.

[14] P. Durka. Matching Pursuit and Unification in EEG Analysis. ARTECH HOUSE, INC., Nordwood, 2007.

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

[16] T. Řondík. Using matching pursuit algorithm with its own dictionary for erp in eeg signal detection. Sborník z 10. ročníku konference Kognice a umělý život, strany 329-332, Bezručovo náměstí 13, Opava, Česká republika, 2010. Slezská univerzita v Opavě, Filozoficko-přírodovědecká fakulta v Opavě.

[17] 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.

[18]P. Mautner and R. Mouček. Using matching pursuit method to detection of erp components. Sborník z 9. ročníku konference Kognice a umělý život, strany 201- 205, Bezručovo náměstí 13, Opava, Česká republika, 2010. Slezská univerzita v Opavě, Filozoficko-přírodovědecká fakulta v Opavě.

[19] J. Ciniburk, Hilbert-Huang transform for ERP detection, disertační práce, Západočeská universita v Plzni, 2011.

[20] 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 spektrum 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, květen 1998.

[21]Z. Yang and L Yang. A new denition of the intrinsic mode function. World Academy of Science, Engineering and Technology, 60:822-825, 2009.

[22] N Huang and Nii O. Attoh-Okine. The Hilbert-Huang Transform in Engineering. CRC Press, 2005.

[23] S. Sanei and J. A. Chambers. EEG Signal Processing, Chippenham (Wiltshire): Antony Rowe Ltd., 2007. ISBN 0470025816

Příloha A – Konfigurační soubory

Konfigurační soubor klasifikátorů:

```
<?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">

  <bean id="class3" class="testing.classificators.FreqAmplTreshold">
          <property name="freqMeanFrom" value="0.2"/>
          <property name="freqMeanTo" value="3"/>
          <property name="fromSample" value="150"/>
          <property name="toSample" value="650"/>
          <property name="amplMeanThreshold" value="3.5"/>
          <property name="id" value="FreqAmpl3"/>
  </bean>

  <bean id="class4" class="testing.classificators.FreqAmplTreshold">
          <property name="freqMeanFrom" value="0.2"/>
          <property name="freqMeanTo" value="3"/>
          <property name="fromSample" value="150"/>
          <property name="toSample" value="650"/>
          <property name="amplMeanThreshold" value="3.25"/>
          <property name="id" value="FreqAmpl4"/>
  </bean>

  <bean id="class5" class="testing.classificators.FreqAmplTreshold">
          <property name="freqMeanFrom" value="0.2"/>
          <property name="freqMeanTo" value="3"/>
          <property name="fromSample" value="150"/>
          <property name="toSample" value="650"/>
          <property name="amplMeanThreshold" value="3.0"/>
          <property name="id" value="FreqAmpl5"/>
  </bean>

 <bean id="weight1l" class="testing.classificators.WeightClassificator">
          <property name="freqMeanFrom" value="0.2"/>
          <property name="freqMeanFromTol" value="0.1"/>
          <property name="freqMeanTo" value="3"/>
          <property name="freqMeanToTol" value="3.5"/>
          <property name="fromSample" value="150"/>
          <property name="toSample" value="650"/>
          <property name="amplMeanThreshold" value="3.0"/>
          <property name="amplMeanThresholdTol" value="2.5"/>
          <property name="classificationTreshold" value="75"/>
          <property name="id" value="WeightClassificator1l"/>
```

```
 </bean>

 <bean id="weight2l" class="testing.classificators.WeightClassificator">
        <property name="freqMeanFrom" value="0.2"/>
        <property name="freqMeanFromTol" value="0.1"/>
        <property name="freqMeanTo" value="3"/>
        <property name="freqMeanToTol" value="3.5"/>
        <property name="fromSample" value="150"/>
        <property name="toSample" value="650"/>
        <property name="amplMeanThreshold" value="3.2"/>
        <property name="amplMeanThresholdTol" value="2.9"/>
        <property name="classificationTreshold" value="75"/>
        <property name="id" value="WeightClassificator2l"/>
 </bean>

<bean id="weight3l" class="testing.classificators.WeightClassificator">
        <property name="freqMeanFrom" value="0.2"/>
        <property name="freqMeanFromTol" value="0.1"/>
        <property name="freqMeanTo" value="3"/>
        <property name="freqMeanToTol" value="3.5"/>
        <property name="fromSample" value="150"/>
        <property name="toSample" value="650"/>
        <property name="amplMeanThreshold" value="3.3"/>
        <property name="amplMeanThresholdTol" value="2.7"/>
        <property name="classificationTreshold" value="80"/>
        <property name="id" value="WeightClassificator3l"/>
 </bean>

 <bean id="window1" class="testing.classificators.WindowClassificator">
        <property name="freqMeanFrom" value="0.2"/>
        <property name="freqMeanTo" value="3.0"/>
        <property name="fromSample" value="150"/>
        <property name="toSample" value="650"/>
        <property name="amplMeanThreshold" value="3.15"/>
        <property name="windowLen" value="200"/>
        <property name="shift" value="10"/>
        <property name="id" value="WindowClassificator1"/>
 </bean>

 <bean id="window2" class="testing.classificators.WindowClassificator">
        <property name="freqMeanFrom" value="0.2"/>
        <property name="freqMeanTo" value="3.0"/>
        <property name="fromSample" value="150"/>
        <property name="toSample" value="650"/>
        <property name="amplMeanThreshold" value="3.25"/>
        <property name="windowLen" value="200"/>
        <property name="shift" value="10"/>
        <property name="id" value="WindowClassificator2"/>
 </bean>

 <bean id="window3" class="testing.classificators.WindowClassificator">
        <property name="freqMeanFrom" value="0.2"/>
        <property name="freqMeanTo" value="3.0"/>
```

```
        <property name="fromSample" value="150"/>
        <property name="toSample" value="650"/>
        <property name="amplMeanThreshold" value="3.4"/>
        <property name="windowLen" value="200"/>
        <property name="shift" value="10"/>
        <property name="id" value="WindowClassificator3"/>
  </bean>
```

Konfigurační soubor spouštění klasifikace:

```
<?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="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//weight_bez//"/>
        <property name="templateFilename" value="configs//classification//htmlTemplate.stg"/>
        <property name="logsDirName" value="classification"/>
</bean>

<util:list id="classifiers">
        <ref bean="weight1l"/>
        <ref bean="weight2l"/>
        <ref bean="weight3l"/>
</util:list>

<util:list id="directories">
        <value>c://hhtResults2//bez//CauchyCfg3//</value>
        <value>c://hhtResults2//bez//CauchyCfg6//</value>
        <value>c://hhtResults2//bez//CauchyCfg7//</value>
        <value>c://hhtResults2//bez//SDcfg3//</value>
        <value>c://hhtResults2//bez//SDcfg4//</value>
        <value>c://hhtResults2//bez//SDcfg8//</value>
</util:list>

<util:list id="patterns">
         <value>30_Cz_baseline_(target|nonTarget)</value>
</util:list>
</beans>
```

Konfigurační soubor EMD s dodatečnou podmínkou:

```
<?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="extremesLocator"/>
</bean>

<bean id="extremesLocator" class="hht.emd.sifting.extremes.locators.LocateLocalExtremaFacade" >
        <constructor-arg ref="maxLocator" />
        <constructor-arg ref="minLocator" />
</bean>

<bean id="minLocator" class="hht.emd.sifting.extremes.locators.DeltaDifferenceExtrema">
        <constructor-arg ref="minHelper"/>
</bean>

<bean id="maxLocator" class="hht.emd.sifting.extremes.locators.DeltaDifferenceExtrema">
        <constructor-arg ref="maxHelper"/>
</bean>

<bean id="minHelper" class="hht.emd.sifting.extremes.locators.MinimumLocatorHelper">
        <property name="delta" value="0.05"/>
</bean>

<bean id="maxHelper" class="hht.emd.sifting.extremes.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="extremesLocator"/>
        <!-- can be "addcrit1, addcrit2 or null-->
        <constructor-arg ref="addcrit2"/>
</bean>

<bean id="addcrit1"
class="hht.emd.sifting.stoppingCriteria.additionalCriteria.EnvelopeMeanCurveMeanValue">
        <constructor-arg value="0.0005"/>
</bean>

<bean id="addcrit2" class="hht.emd.sifting.stoppingCriteria.additionalCriteria.MeanDistanceFromZero">
        <constructor-arg value="0.0075"/>
</bean>

<bean id="ZeroEstimator" class="hht.emd.sifting.extremes.estimators.ZeroPoints"/>
```

```
<bean id="MirrorEstimator" class="hht.emd.sifting.extremes.estimators.Mirror"/>
<bean id="EndPointMirrorEstimator2" class="hht.emd.sifting.extremes.estimators.EndPointMirror2"/>

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

Příloha B – Programátorský manuál

Spuštění HHT je možné několika způsoby. Třída `HhtSimpleRunner` z balíku `hht` obsahuje následující metody pro spuštění algoritmu HHT:

```
runHht(String resultPath, String emdCfg, double[] xVals, double[] yVals, int samplingFrequency, ResultsConfigs resultsCfg)
```

- `resultPath` adresář, kam se uloží výsledky HHT
- `emdCfg` cesta ke konfiguračnímu souboru EMD
- `xVals` časy vzorků signálu
- `yVals` funkční hodnoty signálu
- `samplingFrequency` vzorkovací frekvence signálu
- `resultsCfg` nastavení ukládání výsledků – definuje, co se bude na konci algoritmu ukládat na disk

```
public static void runHht(String resultPath, String emdCfg, double[] xVals, double[] yVals, int samplingFrequency)
```

- spustí předchozí metodu s defaultním nastavením ukládání výsledků algoritmu

```
public static void runHhtDataFromTxt(String resultPath, String emdCfg, String dataFile, int samplingFrequency, int samplesCount)
```

- `dataFile` textový soubor s daty
- `samplesCount` počet vzorků

Druhou možností je spuštění metody `run()` třídy `HhtDataRunner` z balíku `testing`. Tato třída vyžaduje pro spuštění instanci třídy `Configuration`, kterou je možné vytvořit z konfiguračního souboru následovně:

```
Configuration cfg = XmlUtils.unmarshal(Configuration.class, "path//to//configuration//file");
```

Dalším krokem je spuštění samotné klasifikace. Třída `RunClassification` z balíku `testing.classificators` spustí klasifikaci dat z definovaných adresářů. Je možné naráz použít více klasifikátorů. Třída obsahuje statickou metodu `getFromCfg(String cfg)`, která vytvoří a vrátí instanci třídy `RunClassification`. Jejím parametrem je cesta ke konfiguračnímu souboru s nastavením klasifikace. Klasifikaci pak spouští metoda `run()`. Třída navíc umožňuje nastavit libovolný spouštěč klasifikace. Druhou možností je třída `RunClassificationAll` ze stejného balíku. Třída obsahuje statickou metodu `getFromCfg(String cfgFilename)`, která z konfiguračního souboru vytvoří a vrátí instanci třídy `RunClassificationAll`. Samotnou klasifikaci pak spouští metoda `run()`. Třídy se liší hlavně ve způsobu vizualizace dat. První ukládá výsledky do jednoho souboru, druhá ukládá výsledky jednotlivých klasifikátorů do samostatných souborů a nakonec vytvoří soubor s náhledem.