Univerzita Jana Evangelisty Purkyně v Ústí nad Labem (UJEP)

I. a II. seminární práce

Počítačové zpracování signálu (KI/PZS)

Ondřej Švorc (F23209)

3. 2. 2025

I. seminární práce

1. Zadání | Výpočet tepové frekvence z EKG signálu

Ve zdrojové databázi najdete celkem 18 měření EKG signálu pro různé věkové skupiny. Signál obsahuje různé anomálie a nemusí být vždy centralizován podle vodorovné osy. EKG signál obsahuje dominantní peaky, které se nazývají R vrcholy. Vzdálenost těchto vrcholů určuje dobu mezi jednotlivými tepy. Počet tepů za minutu je tedy počet R vrcholů v signálu o délce jedné minuty. Navrhněte algoritmus, který bude automaticky detekovat počet R vrcholů v EKG signálech a prezentujte tepovou frekvenci při jednotlivých jízdách/měřeních. Váš algoritmus následně otestujte na databázi MIT-BIH https://physionet.org/content/nsrdb/1.0.0/ a prezentujte jeho úspěšnost vzhledem k anotovaným datům z databáze.

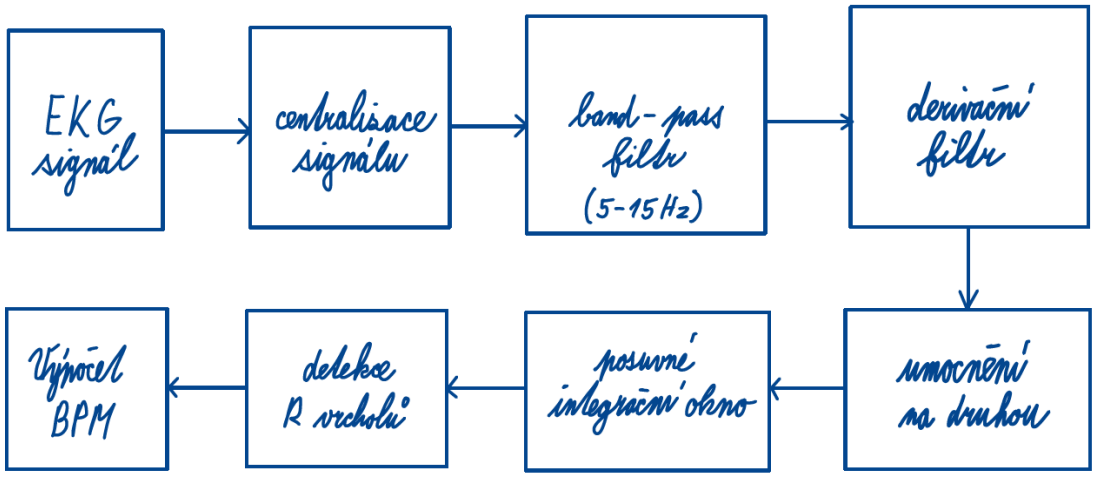
Vstupní data: https://physionet.org/content/butqdb/1.0.0/
Testovací databáze: https://physionet.org/content/nsrdb/1.0.0/
Grafické výstupy: Graf zobrazující tepovou frekvenci v závislosti na měření. Grafické schéma.

Úvodní myšlenkové pochody

Zadání ukrývá několik dílčích podproblémů, které mě, po jejich vyřešení, přivedly k tomu výslednému. Nejprve jsem si nastudoval, co to EKG je, jak EKG signál vypadá a jaký je jeho průběh. Tvrzení ze zadání o vstupním signálu, jako např. „obsahuje různé anomálie“ a „nemusí být vždy centralizován podle vodorovné osy“ mě vyvedly z omylu, že signál bude ze začátku vypadat stejně vzorově jako na obrázcích, s nimiž jsem se při studiu o EKG setkal. Přišlo mi tedy vhodné postupně zjistit, jak a proč anomálie vznikají, proč signál není centralizován dle vodorovné osy, jak ho centralizovaným učiním a jestli to má pro detekci R-vrcholů vlastně vůbec smysl.

Detekce R-vrcholů byl hlavní problém, jehož vyřešení vedlo k výpočtu tepové frekvence. Celé tvrzení o R-vrcholech ze zadání, tedy „Vzdálenost těchto vrcholů určuje dobu mezi jednotlivými tepy. Počet tepů za minutu je tedy počet R vrcholů v signálu o délce jedné minuty“ odhalilo závislost znalosti o výskytech a pozicích R-vrcholů na výpočtu tepové frekvence. Z toho vyplynulo, že pro výpočet tepové frekvence je nutné detekovat R-vrcholy.

K samotné detekci R-vrcholů jsem použil posloupnost několika kroků, které jsem vyhodnotil jak na základě zadání, tak i na základě poznatků o algoritmu s názvem Pan-Tompkinsův algoritmus, který se používá na detekci QRS komplexů. QRS komplexy jsou části EKG signálu odpovídající depolarizaci komor srdce (Q – právě první negativní kmit, R – každý pozitivní kmit, S – všechny negativní kmity za R kmitem).



Porozumění datům

Ze vstupních dat a ze shrnutí na URL adrese, z níž jsem vstupní data získal, jsem zjistil hned několik podstatných informací. Vzorkovací frekvence všech naměřených EKG signálů z tohoto datasetu je 1 000 Hz. K samotným EKG signálům jsou přiložena i akcelerometrická data, a ty mají vzorkovací frekvenci 100 Hz. Minimální délka záznamu je 24 hodin. Jak jsem později zjistil, aktivita v akcelerometrických datech měla vliv na anomálie EKG signálu.

V souborech ve formátu .dat se nacházel samotný EKG signál a akcelerometrická data (každý ve vlastním souboru). Soubory ve formátu .hea patřily ke stejnojmennému .dat souboru a obsahovaly hlavičková data.

```python
def ecg_info(ecg: Record) -> None:
    info = [
        ["Název souboru", ecg.record_name],
        ["Vzorkovací frekvence (Hz)", ecg.fs],
        ["Délka signálu (vzorky)", ecg.sig_len],
        ["Počet kanálů", ecg.n_sig],
        ["Názvy kanálů", ", ".join(ecg.sig_name)],
        ["Jednotky pro každý kanál", ", ".join(ecg.units)],
    ]
    info_df = pd.DataFrame(info, columns=["Informace", "Hodnota"])
    display(info_df)
```

	Informace	Hodnota
0	Název souboru	100001_ECG
1	Vzorkovací frekvence (Hz)	1000
2	Délka signálu (vzorky)	87087000
3	Počet kanálů	1
4	Názvy kanálů	ECG
5	Jednotky pro každý kanál	uV

Centralizace signálu

Centralizací signálu podle vodorovné osy znamená centralizaci okolo osy x, tedy bodu nula. Chtěl jsem, aby signál nebyl zbytečně posunutý nahoru nebo dolů. Odečtením průměrné hodnoty jsem zajistil, že signál bude správně vycentrovaný a obsahuje pouze užitečné informace o změnách v čase. Zároveň, když budou všechny signály kmitat okolo stejného bodu, bude jednodušší provést vyhodnocení funkčnosti finálního algoritmu.

```
1 def _center_signal(self, signal: np.ndarray) -> np.ndarray:
2     return signal - np.mean(signal)
```

Pásmová propust

Pásmová propust je zásadní pro izolaci frekvenčního rozsahu, kde se nachází QRS komplexy. Tvoří ji kombinace dolní propusti, která propouští nízké frekvence a potlačuje vysoké frekvence, a horní propusti, která propouští vysoké frekvence a potlačuje nízké frekvence. Zvolil jsem pásmovou propust s rozsahem 5–15 Hz na základě doporučení z odborných zdrojů, která dosahovala v mém případě nejvyšší úspěšnosti (jiné zdroje uváděly např. 8–12 Hz, 8–15, nebo 5–12 Hz). Má pásmová propust je tedy realizována jako kombinace horní propusti s mezní frekvencí 5 Hz a dolní propusti s mezní frekvencí 15 Hz.

Dozvěděl jsem se, že frekvence < 5 Hz obvykle zahrnují tzv. baseline wander, které je způsobeno dýcháním, pohybem pacienta anebo špatným kontaktem elektrod při měření. Dále jsem se objevil, že frekvence > 15 Hz obvykle zahrnují vysokofrekvenční šum, který je způsoben svalovými artefakty (např. napětím svalů při pohybu).

```python
def _bandpass_filter(
    self, signal: np.ndarray, lowcut_hz=5, highcut_hz=15, order=2
) -> np.ndarray:
    nyquist_freq = 0.5 * self.fs
    low = lowcut_hz / nyquist_freq
    high = highcut_hz / nyquist_freq
    b, a = butter(order, [low, high], btype="band")
    return filtfilt(b, a, signal)
```

Derivační filtr

Derivační filtr je navržen k identifikaci rychlých změn v signálu – což přesně odpovídá tomu, co hledáme u QRS komplexů. Derivace říká, jak rychle se signál mění, takže když signál prudce roste nebo klesá, derivace nabývá velké hodnoty.

Kernel jsem zvolil dle koeficientů z diferenciální rovnice níže.

```python
def _derivative_filter(self, signal: np.ndarray) -> np.ndarray:
    # Diferenční rovnice derivátoru podle Pan-Tompkins:
    # y(n) = (1/8) * [2*x(n) + x(n-1) - x(n-3) - 2*x(n-4)]
    kernel = np.array([2, 1, 0, -1, -2]) / 8.0
    return np.convolve(signal, kernel, mode="same")
```

$$y(n) = \frac{1}{8}[2x(n) + x(n-1) - x(n-3) - 2x(n-4)]$$

https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=53269
(pdf strana 17, skutečná strana 12)

Umocnění na druhou

Umocnění na druhou je jednoduchý, zato účinný krok, který má hned několik důsledků. Jeho aplikací jsem zvýraznil velké hodnoty (strmé sklony v QRS komplexu) více než menší hodnoty (tím je šum). Hodnoty blízko nule, konkrétně hodnoty na intervalu (0, 1), se na základě definice umocňování zmenší (např. $0.1^2 = 0.01$). Dále jsem umocněním zajistil, že všechny hodnoty budou kladné.

```python
def _squaring(self, signal: np.ndarray) -> np.ndarray:
    return np.power(signal, 2)
```

Posuvné integrační okno

Posuvné integrační okno slouží k vytvoření hladkého signálu, ve kterém je snadnější identifikovat QRS komplexy. Předchozí kroky zvýraznili hranice QRS komplexů a tento vytvoří ze signálu hladký signál, ve kterém jsou QRS komplexy jasně rozpoznatelné jako výrazné „hrboly“. Detekce R vrcholů pak probíhá jednoduše pomocí nastavení prahové hodnoty, která odděluje skutečné srdeční tepy od zbytku signálu

```python
def _moving_window_integration(
    self, signal: np.ndarray, window_ms=150
) -> np.ndarray:
    MS_IN_SECOND = 1000.0
    window_size = int(self.fs * (window_ms / MS_IN_SECOND))
    kernel = np.ones(window_size) / window_size
    return np.convolve(signal, kernel, mode="same")
```

Detekce R vrcholů

Konečným krokem je samotná detekce R vrcholů na základě již předzpracovaného a integrovaného signálu. Používám zde metodu find_peaks z knihovny scipy, která identifikuje lokální maxima podle dvou kritérií – minimální vzdálenosti mezi vrcholy a minimální výšky vrcholu.

```python
def _detect_r_peaks(self, signal_integrated: np.ndarray) -> np.ndarray:
    DISTANCE_SECONDS = 0.2
    HEIGHT_RATIO = 0.3
    min_distance = int(DISTANCE_SECONDS * self.fs)
    height_threshold = HEIGHT_RATIO * np.max(signal_integrated)
    peaks, _ = find_peaks(
        signal_integrated, distance=min_distance, height=height_threshold
    )
    return peaks
```

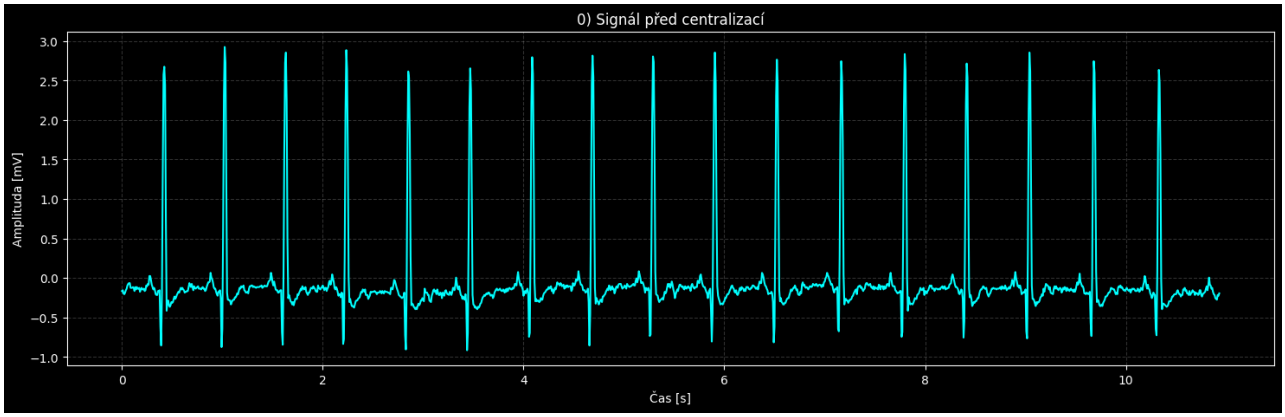
Výpočet tepové frekvence

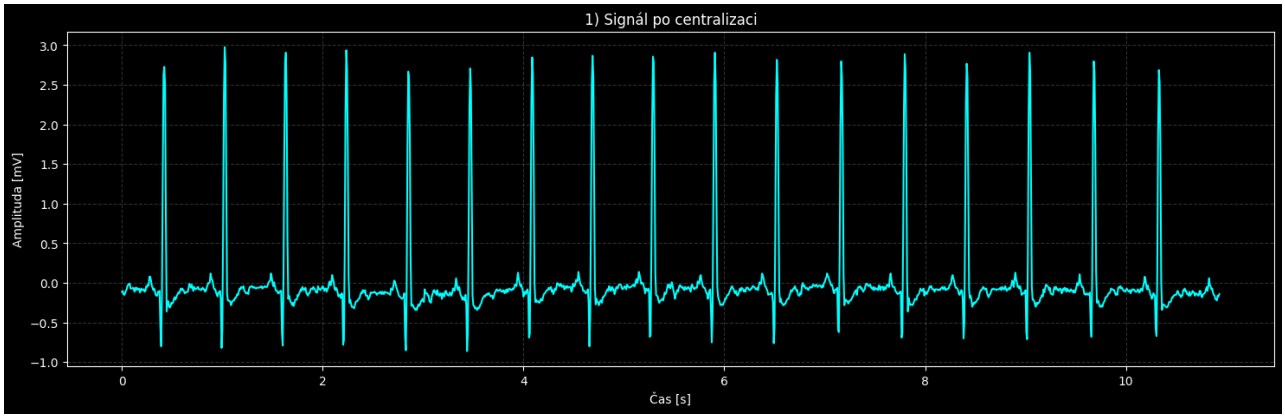
K výpočtu srdeční frekvence v jednotkách úderů za minutu (BPM) jsem použil následující postup. Vzorec vychází z poměru počtu detekovaných R vrcholů k délce analyzovaného signálu:

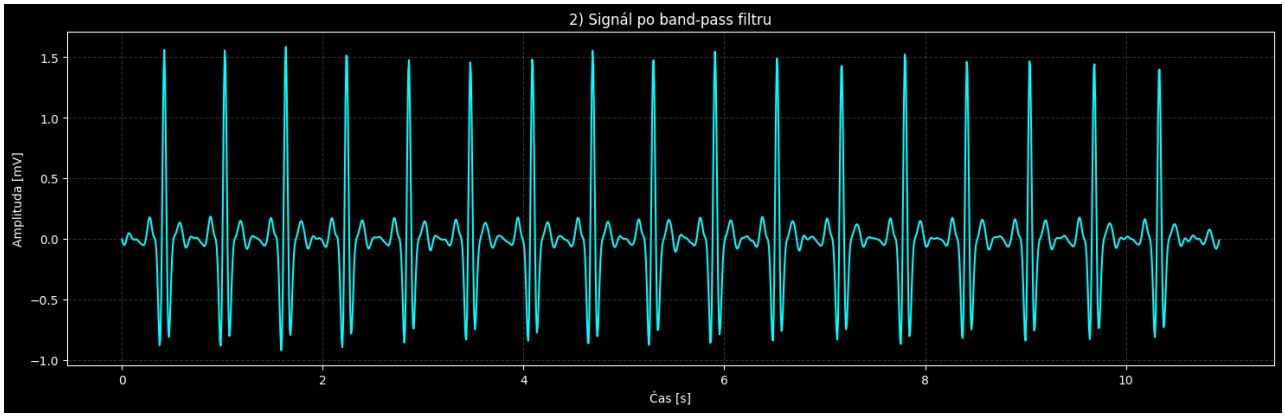
```python
def _calculate_bpm(self, r_peaks_indices: np.ndarray) -> float:
    SECONDS_PER_MINUTE = 60.0
    signal_length_seconds = len(self.ecg_signal) / self.fs
    bpm = (len(r_peaks_indices) / signal_length_seconds) * SECONDS_PER_MINUTE
    return bpm
```

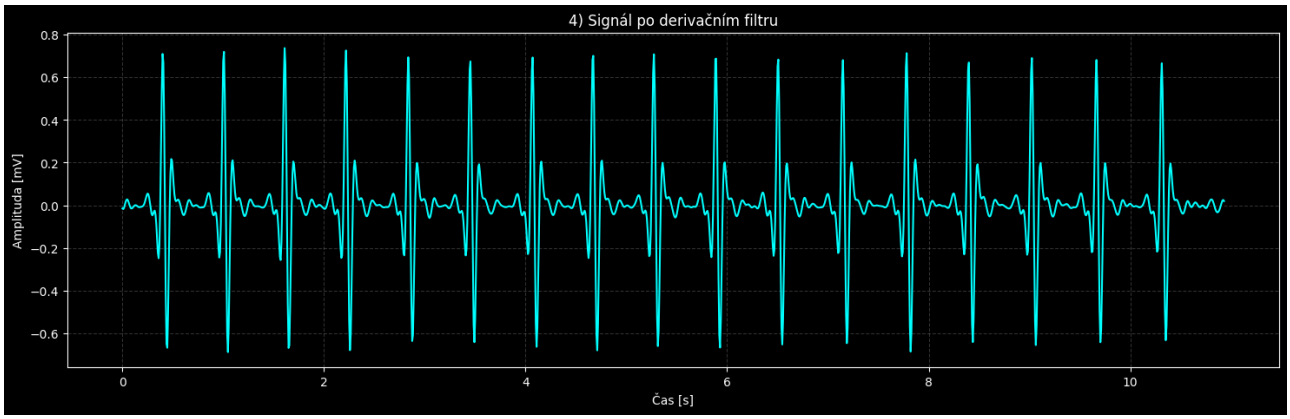
Výsledky na části testovacího vzorku

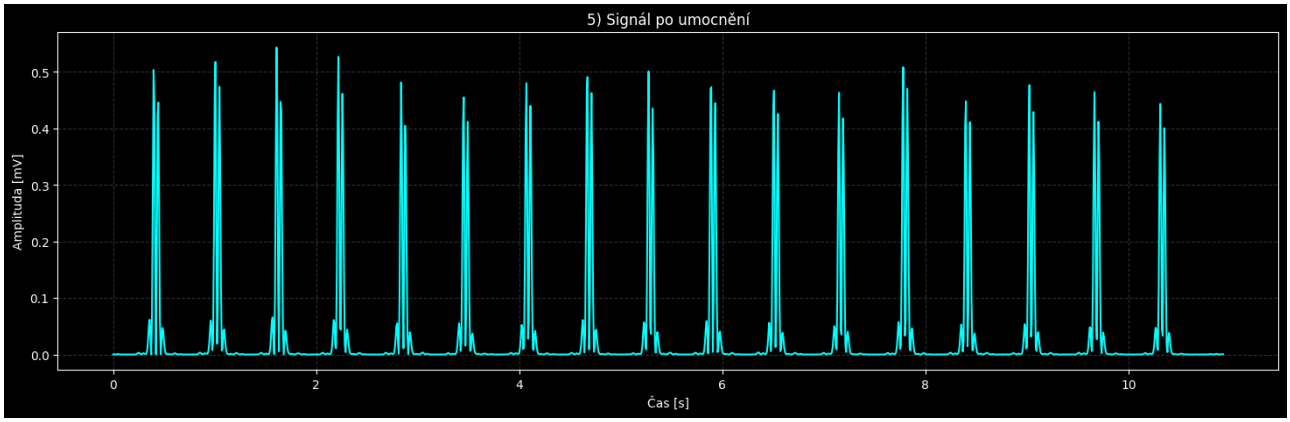
Algoritmus jsem otestoval na prvních cca 11 sekundách testovacího vzorku s označením 16265. V dalším kroku pak na všech testovacích vzorcích.

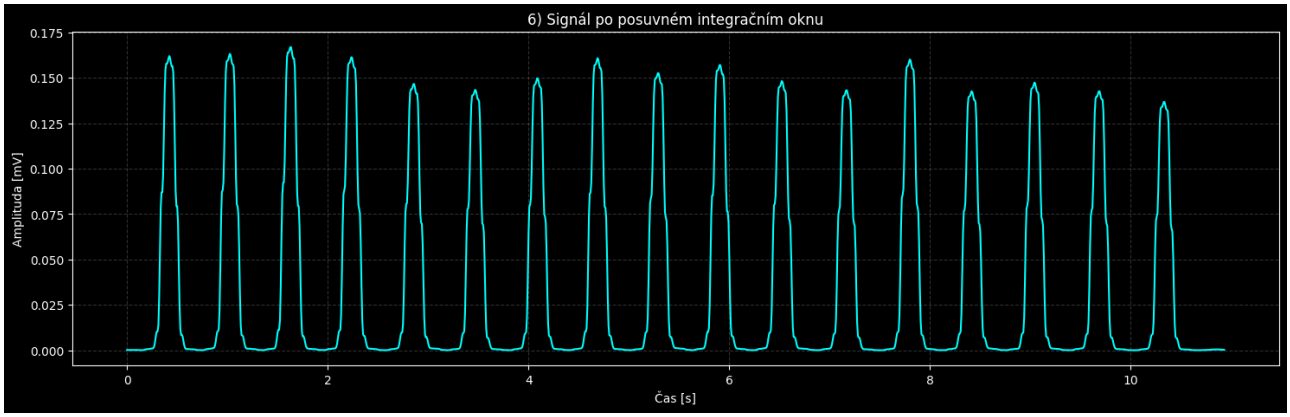


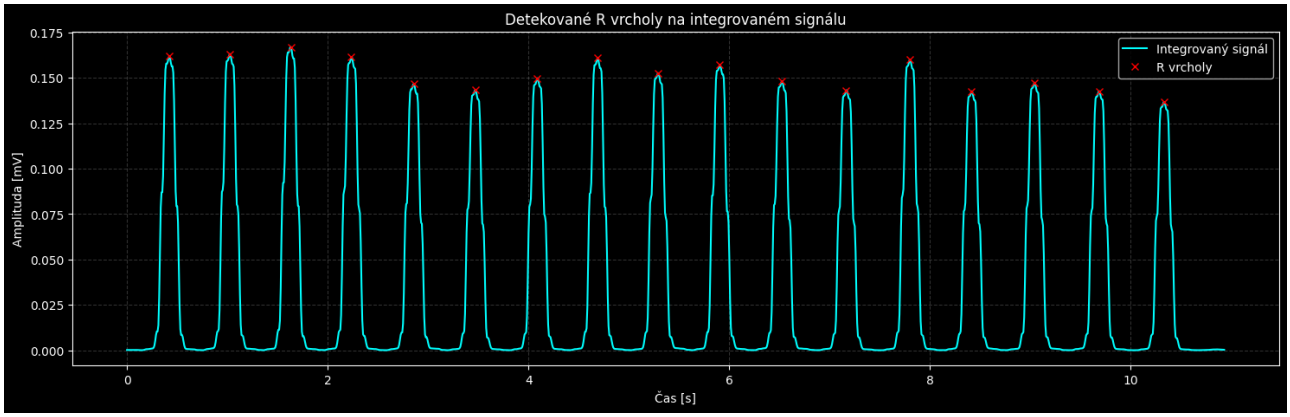


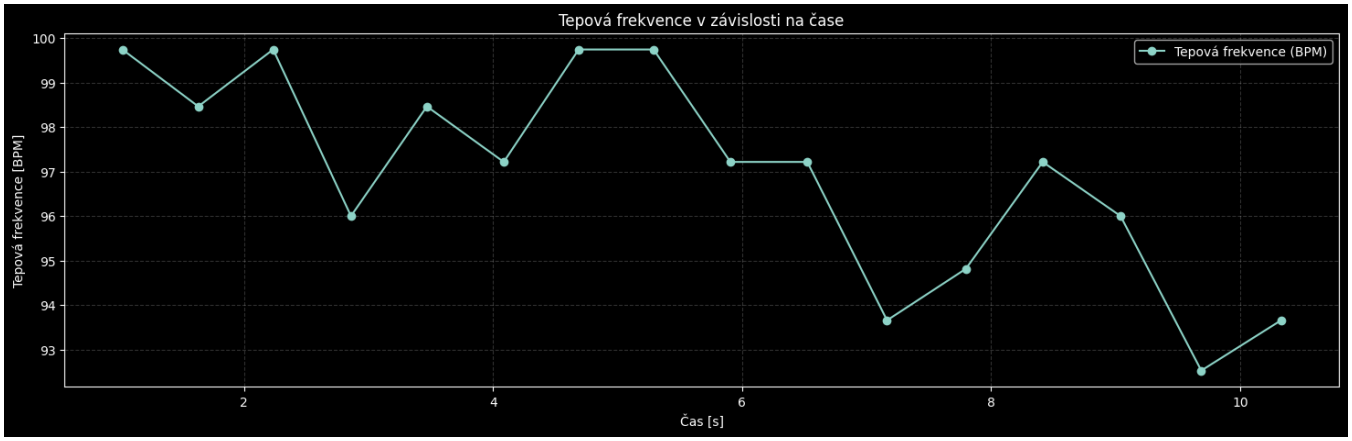



4) Signál po derivačním filtru
Amplituda [mV]
Čas [s]


5) Signál po umocnění
Amplituda [mV]
Čas [s]


6) Signál po posuvném integračním oknu
Amplituda [mV]
Čas [s]


Detekované R vrcholy na integrovaném signálu
Integrovaný signál
R vrcholy
Amplituda [mV]
Čas [s]




```
Počet detekovaných R-peaků: 17
Odhadovaná tepová frekvence (BPM): 93.25714285714285
```

	Soubor	Správně	Špatně	Anotovány	Úspěšnost (%)
0	16265	17	0	18	94.44

Odhad tepové frekvence (93 BPM) na základě prvních 11 sekund vzorku 16265 poskytuje pouze krátkodobý pohled na srdeční aktivitu. Tento výsledek může být samozřejmě ovlivněn okamžitým fyziologickým stavem pacienta (např. stres při začátku měření). I přesto mi přišlo dobré toto otestování provést.

Výsledky na celém testovacím vzorku

Algoritmu jsem otestoval na celém testovacím vzorku s označením 16265.

```
Počet detekovaných R-peaků: 100478
Odhadovaná tepová frekvence (BPM): 65.78081354748603
```

	Soubor	Správně	Špatně	Anotovány	Úspěšnost (%)
0	16265	100460	18	100955	99.51

Zde lze vidět, že tento výsledek je výrazně nižší než původních 93 BPM a lépe reflektuje dlouhodobou tepovou frekvenci pacienta, která ukazuje, že je převážně v klidu (65-66 BPM). Pouze 0.49 % všech R vrcholů bylo detekováno špatně, což na výsledné BPM pravděpodobně nemá významný vliv. Pokud bychom chtěli zjistit opravdu přesný výsledek (nikoliv jen odhad na základě daného algoritmu), výpočet pro tepovou frekvenci by musel brát v potaz i indexy špatně detekovaných R vrcholů.

Výsledky na všech testovacích vzorcích

Algoritmus jsem nakonec otestoval na všech testovacích vzorcích. Zajímavé je, že pokud jsem ho otestoval pouze na prvních cca 11 sekundách, úspěšnost se blížila **90 %**. Zatímco, když jsem ho otestoval na celých signálech, úspěšnost se blížila sotva **60 %**.

Zvláštní je pro mě též měření vzorku 18177, kdy detekce R vrcholů v prvních 11 sekundách sklízí 100% úspěšnost, ale při detekci na celém signálu naopak téměř nulovou (zde zaokrouhleno na 2 desetinná místa). Možnou příčinou je chybné porovnávání s anotacemi, tedy chyba ve funkci s názvem evaluate, která se stará o získání dat pro tento výstup.

Detekce R vrcholů (na prvních 11 sekundách)

	Soubor	Správně	Špatně	Anotovány	Úspěšnost (%)
0	16265	17	0	18	94.44
1	16272	11	0	12	91.67
2	16273	18	0	18	100.00
3	16420	17	0	17	100.00
4	16483	18	0	18	100.00
5	16539	14	0	14	100.00
6	16773	13	0	28	46.43
7	16786	13	0	14	92.86
8	16795	5	7	24	20.83
9	17052	12	0	12	100.00
10	17453	14	0	15	93.33
11	18177	20	0	20	100.00
12	18184	14	0	16	87.50
13	19088	22	0	23	95.65
14	19090	22	0	22	100.00
15	19093	21	0	22	95.45
16	19140	22	0	22	100.00
17	19830	22	0	23	95.65

Celková úspěšnost: 87.28%

Detekce R vrcholů

	Soubor	Správně	Špatně	Anotovány	Úspěšnost (%)
0	16265	100460	18	100955	99.51
1	16272	209	461	97146	0.22
2	16273	88884	7	90097	98.65
3	16420	4323	13	102436	4.22
4	16483	83946	3904	104561	80.28
5	16539	51395	16	108674	47.29
6	16773	81936	1	112897	72.58
7	16786	99896	3	101739	98.19
8	16795	4126	64	87678	4.71
9	17052	62729	82	88002	71.28
10	17453	100466	33	101173	99.30
11	18177	3	3	117004	0.00
12	18184	87	252	102672	0.08
13	19088	89038	149	117880	75.53
14	19090	78255	30	81953	95.49
15	19093	70574	4532	83670	84.35
16	19140	91428	24	96992	94.26
17	19830	27064	49	111263	24.32

Celková úspěšnost: 57.27%

I. seminární práce

2. Zadání | Detekce anomálií v signálech

Ve zdrojové databázi najdete celkem 18 měření EKG obsahující úplné (3 signály) nebo částečné anotace událostí (P,T vlny a QRS komplex). Záznamy EKG obsahují i části, které jsou porušeny vlivem anomálií (vnější rušení, manipulace s pacientem apod.). Navrhněte způsob, jak detekovat tyto úseky a prezentujte statistiku výskytu úseků v měřeních.

Vstupní data: https://physionet.org/content/butqdb/1.0.0/
Grafické výstupy: Tabulka se statistickými ukazateli o výskytu anomálií, případně shoda s anotacemi, vizualizace vybraných úseků, které byly algoritmem detekovány

Úvodní myšlenkové pochody

Pokládal jsem si otázku „co je to anomálie“ a přemýšlel jsem nad tím, kde by mohly být informace o anomáliích uloženy. Dále jsem přemýšlel nad možným řešením, přičemž jako první mě napadla detekce dle nějaké prahové hodnoty. Zaujala mě též část „manipulace s pacientem“ a začal jsem hledat, jestli jsou i tyto data k dispozici.

Porozumění datům

Objevil jsem, že např. 100001_ACC.dat (a jemu podobné soubory) obsahuje akcelerometrická data, která zachycují pohyb pacienta. Hlavička tohoto souboru je strukturována následovně: první řádek „100001_ACC 3 100 8708700“ udává, že data jsou měřena ve třech kanálech, vzorkovací frekvence je 100 Hz a celkový počet vzorků je 8708700. Následující tři řádky specifikují parametry právě jednoho z kanálů.

Přirozeně mě nejprve zaujal CSV soubor, u něhož jsem měl podezření, že by mohl obsahovat anotace. Skutečně tomu tak bylo a potvrzovala to i stránka, z níž jsem data stáhl (viz sekce Data Description). Každý takový soubor má celkem 12 sloupců, přičemž každá trojice sloupců, čteno zleva doprava, je strukturována následovně:

1. sloupec = počátek vzorku
2. sloupec = konec vzorku
3. sloupec = hodnocení kvality vzorku,
    a. kde hodnota vzorku nabývá celkem 4 hodnot, a to:
        i. 0 = signál nemá anotace
        ii. 1 = signál je čitelný (bez, nebo téměř bez, anomálií)
        iii. 2 = signál je hůře čitelný (obsahuje anomálie, ale stále čitelný)
        iv. 3 = signál je nečitelný (příliš mnoho anomálií)

Příkladem je např. soubor 100001_ANN.csv, jehož první řádek vypadal následovně:
198868,320282,1,19526,28694,2,7048,17209,1,7048,17209,1

Segment vzorku od 198868 do 320282 má anotaci 1 = tedy čitelný
Segment vzorku od 19526 do 28694 má anotaci 2 = tedy hůře čitelný
Segment vzorku od 7048 do 17209 má anotaci 1 = tedy čitelný

Poslední trojice se může zdát duplicitní s tou předchozí, a v tomto případě tomu tak skutečně je, ale to je proto, že poslední trojice slouží jako tzv. konsenzus. Pokud jsem to pochopil správně, mělo by to být hodnocení od jiného než původního anotátora, který buď výsledky vyhodnotil stejně, nebo jinak.

Řešení

Dozvěděl jsem se, že existuje tzv. Pravidlo tří sigma (kde sigma = směrodatná odchylka), které říká, že u normálně rozděleného signálu se přibližně 68 % hodnot nachází v intervalu jedné sigma od průměru, 95 % ve dvou sigma a 99,7 % ve třech sigma. Někdy se toto pravidlo nazývá pravidlo 68-95-99,7, tedy proto tyto konkrétní procenta.

Z tohoto důvodu jsem se rozhodl nastavit prahovou hodnotu jako trojnásobek směrodatné odchylky. Tímto způsobem zachytím hodnoty, které spadají mimo očekávaný rozsah – tedy extrémní odchylky, jež by měly indikovat šum v EKG signálu. Uvědomuji si, že EKG signál nemusí být dokonale normálně rozdělen, ale tento přístup byl asi ten nejjednodušší, s nímž jsem přišel.

Signál jsem pouze centralizoval, bez dalších kroků z Pan-Tompkinsova algoritmu v předešlé úloze. Dále jsem vypočítal prahovou hodnotu jako trojnásobek směrodatné odchylky. Nakonec jsem porovnal absolutní hodnotu centralizovaného signálu se zmíněnou prahovou hodnotou. Pokud je hodnota větší než prahová hodnota, označuji ji jako anomálii (příznakem True). Pokud je menší, tak False.

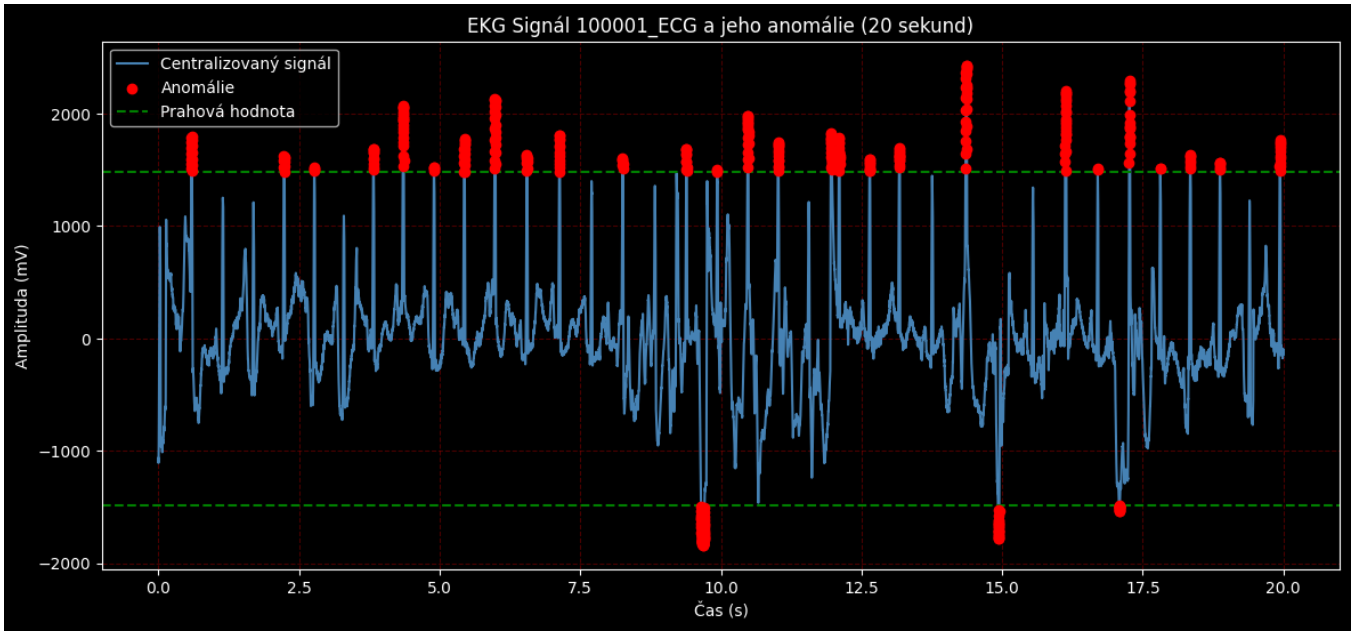
Pro přesnější shodu by bylo nutné použít sofistikovanější metody, které by zohlednily i další faktory, např. zmíněná akcelerometrická data. Respektive, místo staticky zvolené prahové hodnoty by řešení mohlo být vylepšeno použitím adaptivní prahové hodnoty, která by se získávala na základě lokálních vlastností signálu (např. v momentech vyšší pohybové aktivity by se práh mohl automaticky zvýšit). Mezi EKG signálem a akcelerometrickými daty pravděpodobně bude existovat korelace a otázkou by bylo, zda by se vliv akcelerometru projevil okamžitě nebo s určitým zpožděním. Protože tato data nejsou v řešení zohledněna, nakonec jsem se rozhodl neimplementovat porovnání s anotacemi, na rozdíl od první a poslední seminární práce.

```python
def _centralize_signal(self, signal: np.ndarray) -> np.ndarray:
    return signal - np.mean(signal)

def _calculate_threshold(self) -> float:
    std_dev = np.std(self._centralized_signal)
    return 3 * std_dev

def _detect_anomalies(self) -> np.ndarray:
    return np.abs(self._centralized_signal) > self._threshold

```



II. seminární práce

Zadání | Klasifikace zvukových záznamů

Ve zdrojové databázi najdete celkem 208 hlasových záznamů písmene a. Pomocí Vámi vybrané techniky v časové nebo frekvenční oblasti klasifikujte zvukové záznamy na dobré a patologické. V případě patologických poté klasifikujte jednotlivé poruchy. Jejich výčet najdete buď v hlavičkových souborech nebo v propisu databáze. Pro klasifikaci do jednotlivých skupin použijte veškeré techniky, které jste si v rámci kurzu osvojili včetně Fourierovy a kepstrální analýzy. Úspěšnost Vašeho postupu porovnejte s anotacemi, resp. rozřazením do skupin, které provedli experti, kteří data pořizovali.

Vstupní data: https://physionet.org/content/voiced/1.0.0/
Grafické výstupy: Grafy demonstrující práci se signálem v časové nebo frekvenční oblasti, vizualizace klasifikace v prostoru nebo pomocí Vámi zvolených parametrů. Tabulka s úspěšností klasifikace na jednotlivé skupiny a patologické signály.

Úvodní myšlenkové pochody

Ze zadání jsem vyčetl hned několik podúloh – rozdělení na dobré a patologické. Zjednodušeně, zdravý/nemocný, i když rozumím tomu, že patologický zvukový signál nemusí nutně znamenat, že jej říkal člověk s vadou řeči. Nejprve budu muset zjistit, jaké všechny vady řeči se mohou objevit a jak se značí nepatologický zvukový signál. To vše budu získávat z hlavičkových souborů. Pak se pokusím porozumět dostupným datům a pustím do prvotního řešení. Nakonec vymyslím způsob, jak jej porovnat s anotacemi a zjistit, zdali se jedná o zdravého či nemocného pacienta.

Porozumění datům

Ze stránky physionet se dozvídám, že všechny nahrávky mají totožnou vzorkovací frekvenci (8000 Hz) a stejně dlouhý vzorek. Při přepočtu na sekundy má každá zvuková nahrávka cca 5 sekund (přesněji 38080 / 8000 = 4,76 s = 4 sekundy 760 milisekund).

Co mě překvapilo byla jednotka NU. Tato zkratka znamená Normalized Unit, tedy normalizovaná jednotka. Nicméně, nikde není na první pohled uvedeno, na jaké rozpětí hodnot byly zvukové nahrávky normalizovány. Provedl jsem tedy postupné přečtení maximální a minimální hodnoty každého signálu, přičemž jsem si držel informaci o zatím nejvyšším maximu a minimu, a pokud měl signál vyšší maximální hodnotu než globální maximum, stávala se právě maximální hodnota signálu novým globálním maximem. Podobně to funguje i s globálním minimem. Nakonec se při zaokrouhlení ukazuje, že signál byl normalizován na rozpětí (-1, 1).

```python
# NU = Normalized Unit
# Zde zjišťuji, na jaké rozmezí byla amplituda hlasu normalizována.

folder = "data"
global_max = -np.inf
global_min = np.inf

for file in os.listdir(folder):
    if file.endswith(".dat"):
        file_path = os.path.join(folder, file[:-4])
        record: Record = wfdb.rdrecord(file_path)
        signal = record.p_signal[:, 0]
        file_max = np.max(signal)
        file_min = np.min(signal)
        if file_max > global_max:
            global_max = file_max
        if file_min < global_min:
            global_min = file_min

# Dedukce je taková, že rozmezí je na intervalu od -1 do 1.
print("Celková maximální hodnota signálů:", global_max)
print("Celková minimální hodnota signálů:", global_min)
```

```
Celková maximální hodnota signálů: 0.999969482654695
Celková minimální hodnota signálů: -0.9999999997671729
```

	Informace	Hodnota
0	Název souboru	voice001
1	Vzorkovací frekvence (Hz)	8000
2	Délka signálu (vzorky)	38080
3	Počet kanálů	1
4	Názvy kanálů	voice
5	Jednotky pro každý kanál	NU

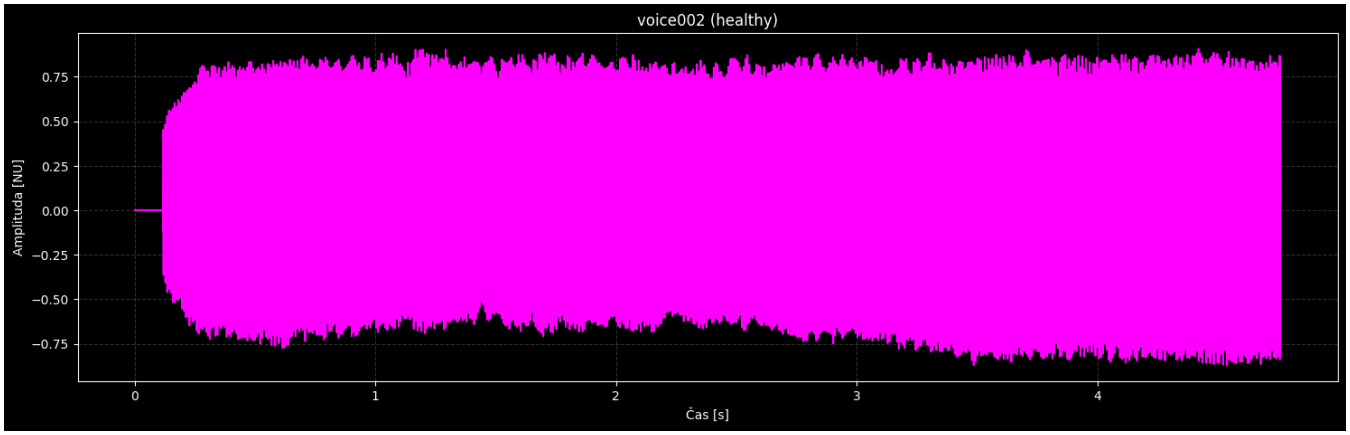
Dále mě zajímalo, jaké všechny vady řeči se ve zvukových nahrávkách mohou objevit. Na základě hlavičkových souborů jsem našel 4 vady řeči, přičemž 2 z nich znamenali pravděpodobně totéž a jednalo se o překlep – hyperkineti dysphonia a hyperkinetic dysphonia. Celkem tedy 3 označení poruch řeči a 1 označení pro ty pacienty, kteří poruchu řeči nemají (healthy).

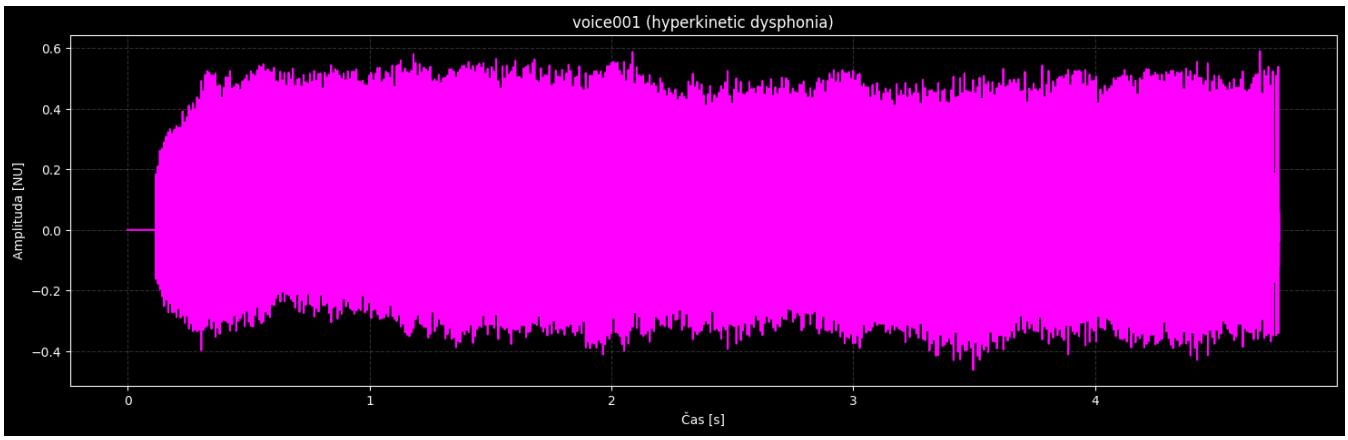
```python
def get_possible_diagnoses() -> set[str]:
    results = set()
    folder = "data"
    for file in os.listdir(folder):
        if file.endswith(".hea"):
            file_no_extension = file[:-4]
            file_path = os.path.join(folder, file_no_extension)
            header = wfdb.rdheader(file_path)
            comment: str = header.comments[0]
            diagnoses = comment.split("<diagnoses>:")[1].split("<")[0].strip()
            results.add(diagnoses)
    return results
```

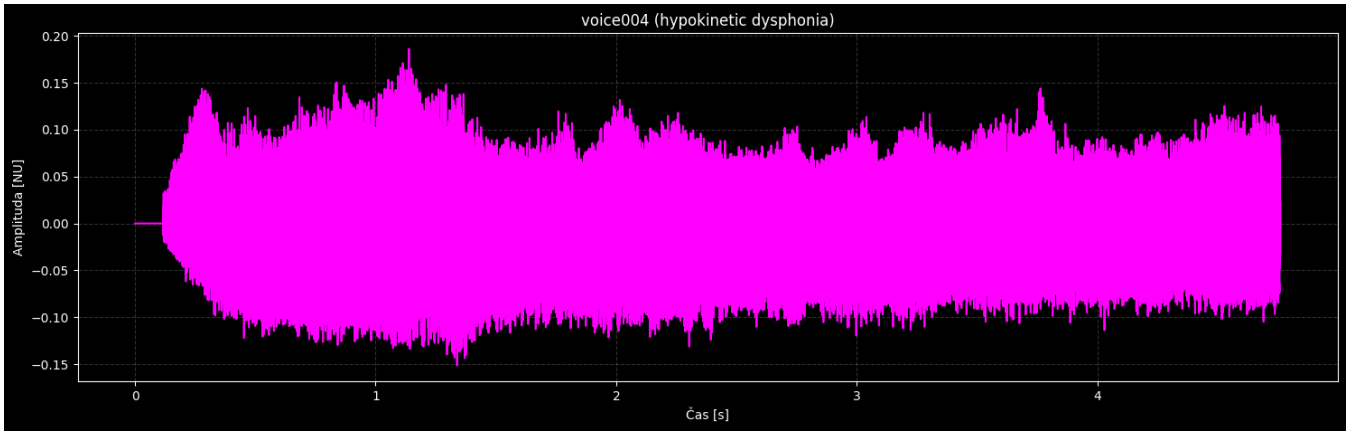
```
{'healthy',
 'hyperkineti dysphonia',
 'hyperkinetic dysphonia',
 'hypokinetic dysphonia',
 'reflux laryngitis'}
```

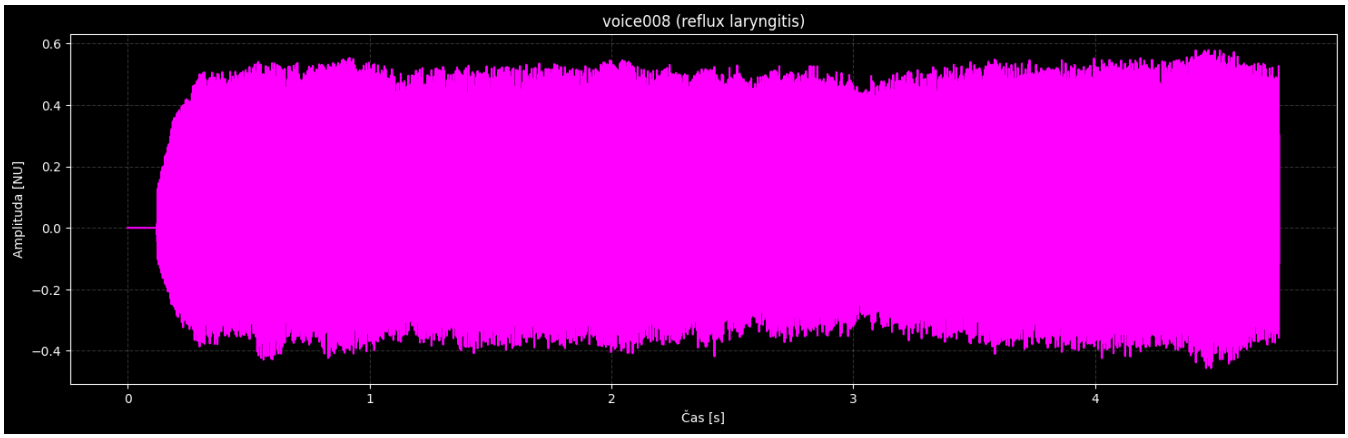
Manuálně jsem pak dohledal ke každé kategorii alespoň 1 zvukovou nahrávku, kterou jsem si vizualizoval grafem a také přehrál zvukově, abych na vlastní uši slyšel, jak daná patologie zní. Pokud by hlavičkové soubory nebyly na první pohled lidsky čitelné, zvolil bych programový přístup. Zde mi to ale přišlo zbytečné.

```
# healthy (voice002)
# hyperkineti dysphonia = hyperkinetic dysphonia (voice001)
# hypokinetic dysphonia (voice004)
# reflux laryngitis (voice008)
```









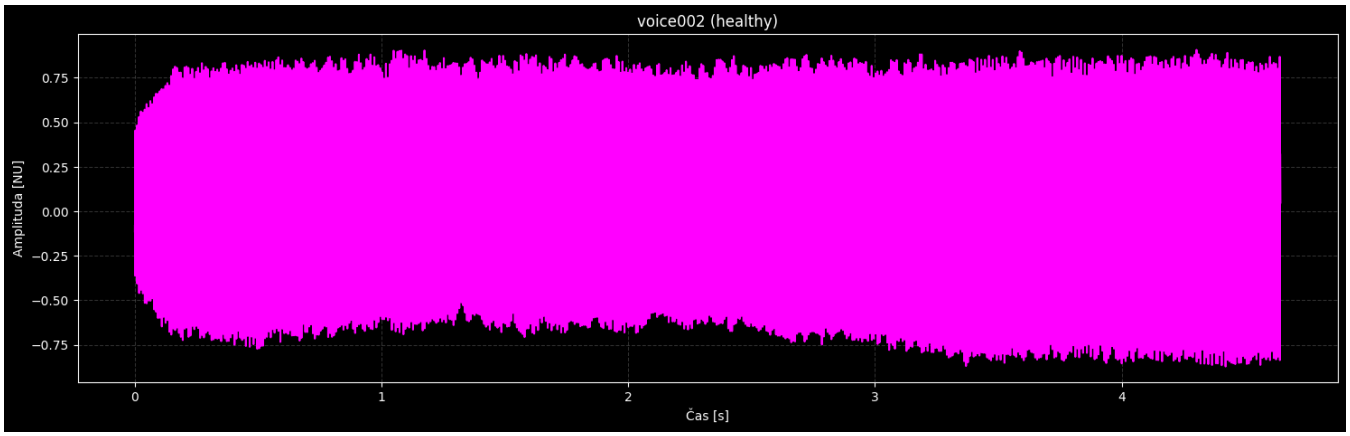
Na první pohled jdou vidět u všech nahrávek odlišná minima a maxima. To samé platí i o počátečním tichu.

Odstranění ticha

Tato funkce odstraňuje případné tiché úseky jak na začátku, tak i případně na konci zvukového signálu. Vrací indexy vzorků, jejichž absolutní hodnota překračuje zadaný práh. Pokud jsou nalezeny alespoň dva takové vzorky, funkce vrátí oříznutý signál mezi prvním a posledním vzorkem. V opačném případě vrací původní signál.

Z obrázků z předchozího kroku lze vidět, že ticho se nachází pouze na začátku a na konci již ne. Nicméně, jedná se o pár šetření, které bych nerad generalizoval na celou datovou sadu.

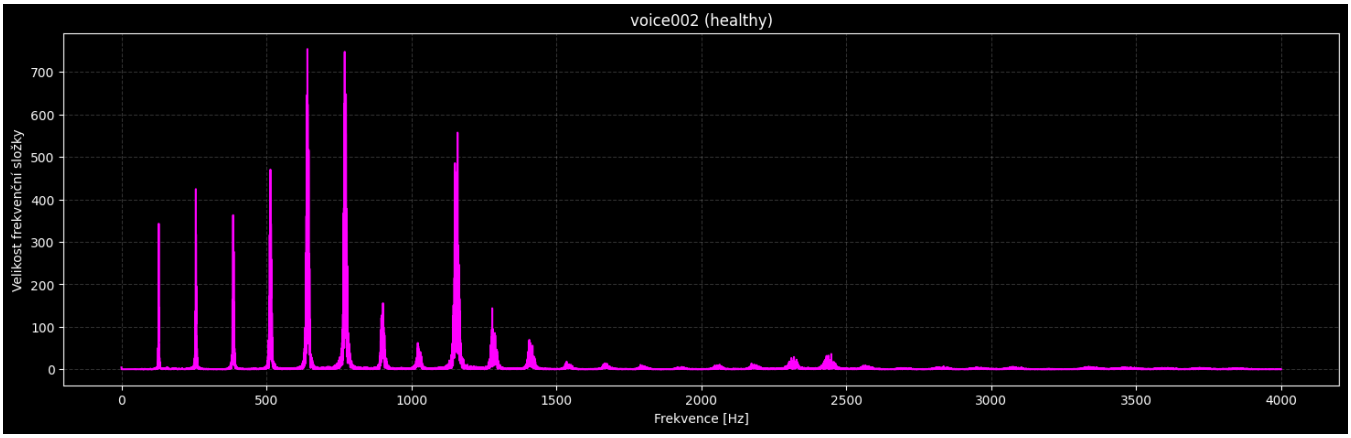
Stejný signál s označením healthy z předchozí kroku pak vypadá následovně, bez tichého úseku na začátku.



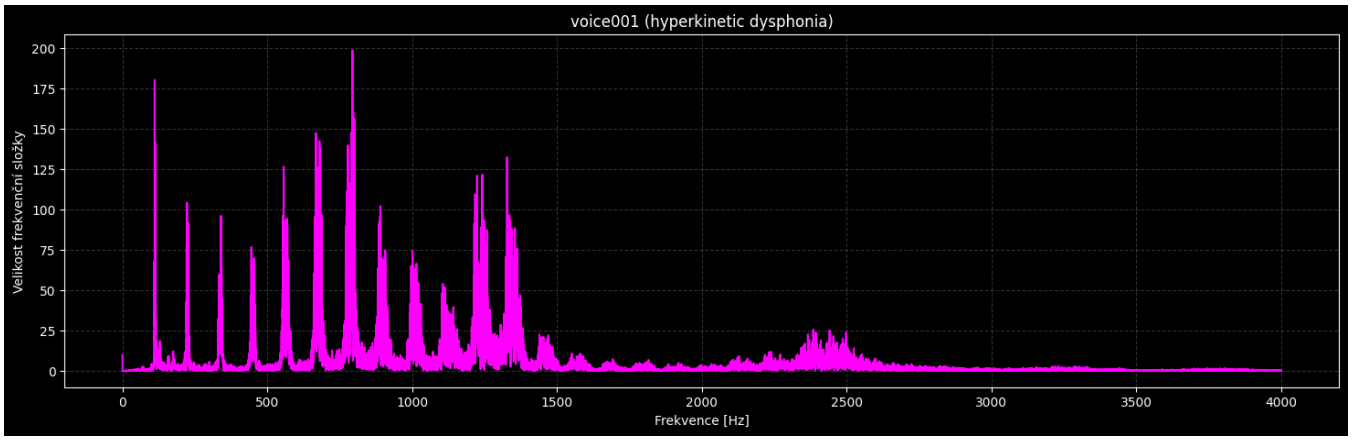
Zobrazení frekvenčního spektra

Na obrázcích níže jsou frekvenční spektra všech zmíněných hlasových záznamů.

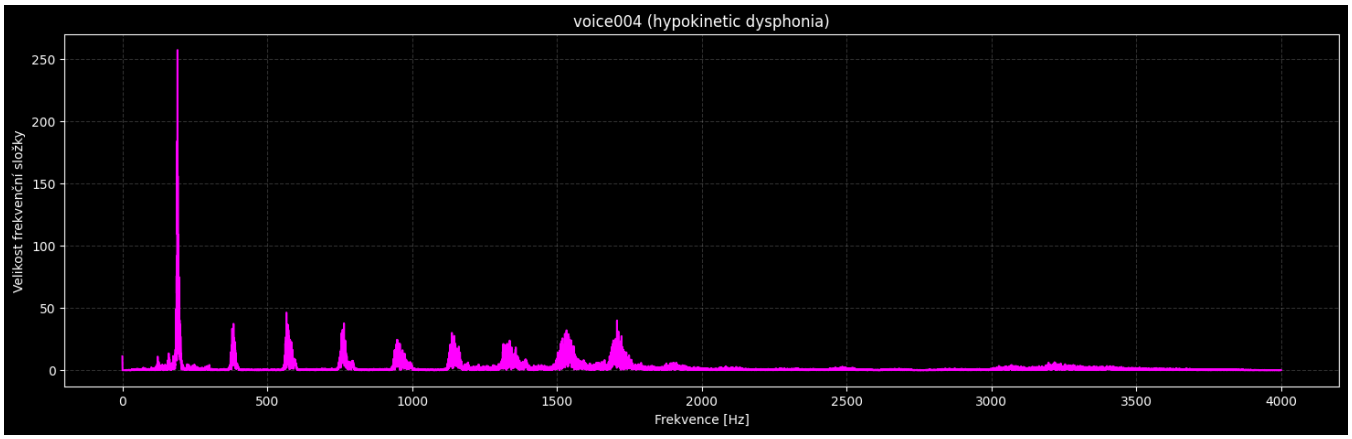
Zdravý hlas (voice002) má pravidelné a ostré vrcholy, jež ukazují na čistý/stabilní tón. Tyto vrcholy postupně klesají s vyšší frekvencí a dosahují vysokých hodnot. To značí silný a nepřerušovaný hlas.



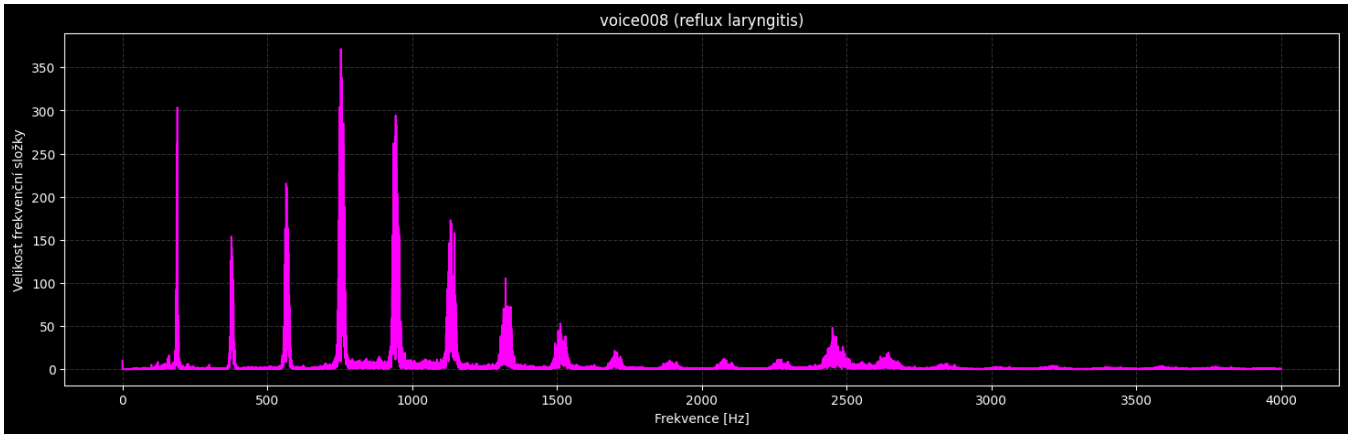
Hlas s hyperkinetickou dysfonií (voice001) je naopak méně pravidelný. Vrcholy jsou menší a rozmazanější a mezi nimi je více šumu. Tento jev by odpovídal nekontrolovaným svalovým pohybům, které logicky narušují plynulost hlasu.



Hypokinetická dysfonie (voice004) ukazuje velmi slabé vrcholy. To znamená, že hlas je slabý a málo výrazný kvůli nízké svalové aktivitě.



Refluxní laryngitida (voice008) má spektrum s částečně pravidelnými vrcholy, mezi nimiž je patrný šum. Takový stav odráží, jak jsem se dočetl, podráždění hlasivek způsobené refluxem.



Celkově lze pozorovat, že zdravý hlas má nejpravidelnější a nejvýraznější spektrum, zatímco hlasové poruchy se vyznačují nepravidelnostmi, vyšší úrovní šumu a nižší amplitudou frekvenčních složek.

Shannonova entropie ve frekvenčním spektru

Funkce níže je přímou aplikací Shannonova vzorce entropie na frekvenční spektrum signálu. To znamená, že místo toho, abych počítal entropii nějakého obecného pravděpodobnostního rozdělení, vezmu výsledek rychlé Fourierovy transformace signálu a zjistím, kolik „energie“ je přítomno v jednotlivých frekvencích. Tuto „energii“ normalizuji (aby součet všech hodnot byl roven 1) a interpretuju ji jako pravděpodobnostní rozdělení. Na tomto rozdělení pak vypočítám entropii podle vzorce. Tento krok dělám proto, abych získal hodnotu, který popisuje, jak je „energie“ signálu rozložena napříč různými frekvencemi.

```python
def shannon_spectral_entropy(magnitude: np.ndarray) -> float:
    # https://en.wikipedia.org/wiki/Entropy_(information_theory)
    # https://cs.wikipedia.org/wiki/Informační_entropie
    p = magnitude / np.sum(magnitude)
    p = p[p > 0]
    return -np.sum(p * np.log2(p))
```

$$S = -\sum_i P_i \log P_i = -\operatorname{E}_P[\log P],$$

Předzpracování signálu a Fourierova analýza

Následující funkce slouží k převodu časového signálu do frekvenční domény. Stejně jako u EKG signálu, i zde signál nejprve centralizuji okolo vodorovné osy. Dále se signál násobí s Hammingovým oknem, které slouží k tomu, aby se na koncích vzorku eliminovaly náhlé přechody. Když totiž provedeme Fourierovu transformaci, implicitně předpokládáme, že se signál periodicky opakuje donekonečna. Pokud začátek a konec signálu nejsou stejné, vznikne náhlá nespojitost, která se projeví jako šum – tento jev se nazývá „spectral leakage“. Když aplikujeme Hammingovo okno, upravujeme signál tak, aby jeho okraje nebyly ostré a tomuto jevu se předešlo. Pomocí FFT se signál převede do frekvenční domény, kde se získá amplitudové spektrum (pouze pro kladné frekvence). Pokud je amplitudové spektrum prázdné, funkce vrátí prázdné hodnoty. Nakonec se spektrum normalizuje a spolu s frekvenční osou a počtem vzorků se vrací jako výstup.

```python
def compute_fft_features(signal: np.ndarray, sr: int = 8000):
    n = len(signal)
    signal = signal - np.mean(signal)
    windowed = signal * np.hamming(n)
    fft_result = np.fft.fft(windowed)
    magnitude = np.abs(fft_result[: n // 2])
    if not np.any(magnitude):
        return None, None, None, n
    magnitude = magnitude / np.max(magnitude)
    freqs = np.fft.fftfreq(n, 1 / sr)[: n // 2]
    return fft_result, magnitude, freqs, n
```

Získání charakteristik spektra

Z této funkce vychází několik důležitých charakteristik spektra. Vypočítá se spektrální centroid, což je vážený průměr frekvencí a ukazuje, kde je soustředěná většina energie. Dále se určí spread, tedy rozptyl energie okolo tohoto středu, a skewness, která měří asymetrii rozložení. Nakonec se spočítá již zmíněná spektrální entropie pomocí funkce spectral_entropy.

```python
def calculate_spectral_features(magnitude: np.ndarray, freqs: np.ndarray):
    centroid = np.sum(freqs * magnitude) / np.sum(magnitude)
    spread = np.sqrt(np.sum(((freqs - centroid) ** 2) * magnitude) / np.sum(magnitude))
    skewness = skew(magnitude)
    entropy = spectral_entropy(magnitude)
    return centroid, spread, skewness, entropy
```

Kepstrální analýza

Funkce níže provádí kepstrální analýzu signálu. Nejprve vezme logaritmus amplitudového spektra (což zvýrazní rozdíly v energii), pak provede inverzní FFT, aby získala tzv. cepstrum – spektrum spektra. Z cepstra vybere oblast od indexu 20 do přibližně jedné třetiny délky signálu a najde v ní nejvýraznější vrchol, který indikuje dominantní periodickou složku. Dále vypočítá průměr prvních několika cepstrálních koeficientů.

```python
def calculate_cepstral_features(fft_result: np.ndarray, n: int):
    log_magnitude = np.log(np.abs(fft_result) + 1e-10)
    cepstrum = np.fft.ifft(log_magnitude).real
    ceps_start = 20
    cepts_end = n // 3
    if cepts_end <= ceps_start:
        return None, None
    cepts_segment = cepstrum[ceps_start:cepts_end]
    peak_idx_local = np.argmax(np.abs(cepts_segment))
    peak_idx_global = peak_idx_local + ceps_start
    cep_peak = cepstrum[peak_idx_global]
    cep_mean = np.mean(np.abs(cepstrum[1:14]))
    return cep_peak, cep_mean
```

Analýza amplitudového spektra

V této funkci analyzuji amplitudové spektrum signálu tak, že nejprve najdu významné vrcholy, které mi ukazují, kde je energie signálu nejvyšší, a pak spočítám rozdíly mezi frekvencemi těchto vrcholů. Díky těmto rozdílům mohu odhadnout, jak pravidelně se tyto frekvenční složky opakují jakou mají variabilitu jejich rozdíly. Průměr rozdílů pak považuji za odhad základní frekvence (f0). Pokud signál neobsahuje dostatek vrcholů, nastavím hodnoty tak, aby bylo jasné, že nelze spolehlivě určit jakési uspořádané a pravidelné rozložení tónů v signálu.

```python
def analyze_peak_features(magnitude: np.ndarray, freqs: np.ndarray):
    peaks, _ = find_peaks(magnitude, height=0.1, prominence=0.05)
    if len(peaks) > 1:
        peak_freqs = freqs[peaks]
        freq_diffs = np.diff(peak_freqs)
        if len(freq_diffs) == 0 or np.mean(freq_diffs) == 0:
            harmonic_regularity = float("inf")
        else:
            harmonic_regularity = np.std(freq_diffs) / np.mean(freq_diffs)
        fundamental_freq_variability = (
            np.std(freq_diffs) if len(freq_diffs) > 1 else float("inf")
        )
        f0 = np.mean(freq_diffs) if len(freq_diffs) > 0 else 0
    else:
        harmonic_regularity = float("inf")
        fundamental_freq_variability = float("inf")
        f0 = 0
    return f0, harmonic_regularity, fundamental_freq_variability

```

Klasifikace patologií

Tato funkce analyzuje hlasový signál a snaží se rozhodnout, zda je hlas „zdravý“ nebo vykazuje známky patologie na základě doposud představených funkcí.

Nejprve nastavím prahové hodnoty a parametry (na základě odborných zdrojů a také metodou pokus/omyl), které definují, jaké charakteristiky signálu považovat za indikaci zdravého hlasu. Poté převedu časový signál do frekvenční domény a získám jeho spektrum. Následuje výpočet základních spektrálních rysů, jako je centroid, spread, skewness a entropie, které popisují rozložení energie v signálu. Dále provedu kepstrální analýzu, kde se zaměřím na periodické složky signálu, a nakonec analyzuji frekvenční vrcholy, abych odhadl základní frekvenci a její pravidelnost. Na závěr vyhodnotím, zda je splněno dostatečné množství kritérií, a pokud ano, funkce vrátí hodnotu True, což znamená, že hlas je považován za zdravý; pokud ne, vrátí False a hlas je považován za patologický.

```python
def is_voice_healthy(signal: np.ndarray) -> bool:
    """ """
    # Prahové hodnoty a požadovaný počet splněných kritérií
    CENTROID_THRESHOLD = 1300.0
    SPREAD_THRESHOLD = 700.0
    SKEWNESS_THRESHOLD = 1.4
    CEPSTRAL_PEAK_THRESHOLD = 4.5
    ENTROPY_THRESHOLD = 3.9
    F0_MIN = 70.0
    F0_MAX = 300.0
    REGULARITY_THRESHOLD = 0.4
    FREQ_VARIABILITY_THRESHOLD = 8.0
    CEPS_MEAN_THRESHOLD = 0.55
    REQUIRED_CONDITIONS = 5
    sr = 8000

    fft_result, magnitude, freqs, n = compute_fft_features(signal.copy(), sr)
    if magnitude is None:
        return False

    centroid, spread, skewness, entropy = calculate_spectral_features(magnitude, freqs)

    cep_peak, cep_mean = calculate_cepstral_features(fft_result, n)
    if cep_peak is None:
        return False

    f0, harmonic_regularity, fundamental_freq_variability = analyze_peak_features(
        magnitude, freqs
    )

    conditions = [
        centroid < CENTROID_THRESHOLD,
        spread < SPREAD_THRESHOLD,
        abs(skewness) < SKEWNESS_THRESHOLD,
        cep_peak > CEPSTRAL_PEAK_THRESHOLD,
        entropy < ENTROPY_THRESHOLD,
        F0_MIN < f0 < F0_MAX,
        harmonic_regularity < REGULARITY_THRESHOLD,
        fundamental_freq_variability > FREQ_VARIABILITY_THRESHOLD,
        cep_mean < CEPS_MEAN_THRESHOLD,
    ]

    return sum(conditions) >= REQUIRED_CONDITIONS
```

Výsledky na všech vzorcích

Nakonec jsem otestoval finální algoritmus (zapouzdřený ve funkci is_voice_healthy) na anotovaných datech. Vyextrahoval jsem z hlavičkových souborů informaci o patologii. U 151 z 208 souborů jsem správně určil, že se jednalo buď o patologický nebo nepatologický signál. U zbývajících 57 z 208 algoritmus určil patologii nebo nepatologii signálu chybně. Celková úspěšnost určení, zdali je pacient zdravý či nemocný pak byla okolo **73 %**.

Souhrnná statistika:

	Metrika	Hodnota
0	Správně	151.000000
1	Špatně	57.000000
2	Celkem	208.000000
3	Úspěšnost (%)	72.596154

Závěr

V 1. seminární práci jsem se zaměřil na detekci R-vrcholů pro výpočet tepové frekvence z EKG signálu. Postupoval jsem podle kroků inspirovaných Pan-Tompkinsovým algoritmem, který zahrnoval centralizaci signálu, pásmovou propust, derivační filtr, umocnění na druhou a posuvné integrační okno. Algoritmus jsem otestoval na různých částech záznamů a jeho úspěšnost byla vysoká na krátkých úsecích (cca **90 %**), ale výrazně klesala při aplikaci na celé signály (kolem **60 %**).

Ve 2. seminární práci jsem navrhl jednoduchý algoritmus pro detekci anomálií v EKG signálu. Použil jsem pravidlo tří sigma – hodnoty překračující trojnásobek směrodatné odchylky jsem považoval za anomálie. Porovnání s anotacemi jsem nakonec nerealizoval, protože by bylo nutné zahrnout sofistikovanější přístup s využitím akcelerometrických dat, která zachycují pohyb pacienta a mají pravděpodobně okamžitý vliv na EKG signál. Vzhledem k omezenému rozsahu práce jsem se rozhodl zaměřit na jednodušší detekci a porovnání s anotacemi jsem provedl až v jiných seminárních úlohách.

Ve 3. seminární práci jsem se věnoval klasifikaci hlasových záznamů na zdravé a patologické. Použil jsem techniky Fourierovy a kepstrální analýzy a odstranění ticha pro předzpracování signálu. Algoritmus dosáhl **73 %** úspěšnosti při rozpoznávání patologických a nepatologických hlasů. To ukazuje, že základní spektrální charakteristiky jsou užitečné pro klasifikaci, ale existuje prostor pro vylepšení pomocí složitějších modelů nebo kombinace více parametrů.

Zdroje

Popis EKG – WikiSkripta. (n.d.). *WikiSkripta*. https://www.wikiskripta.eu/w/Popis_EKG

Contributors to Wikimedia projects. (2005, April 18). *QRS complex*. Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/QRS_complex

Contributors to Wikimedia projects. (2019, July 1). *Pan–Tompkins algorithm*. Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Pan–Tompkins_algorithm

Tarantíková, L. (2017). *Analýza EKG signálů* [Bakalářská práce, Západočeská univerzita v Plzni]. OTIK. https://otik.uk.zcu.cz/bitstream/11025/27683/1/BP-Tarantikova-Analyza-EKG-signalu.pdf

Novák, J. (2019). *Zpracování a analýza biomedicínských signálů* [Diplomová práce, VUT Brno]. https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=53269

Svoboda, P. (2020). *Analýza EKG signálů s využitím metod strojového učení* [Diplomová práce, VUT Brno]. https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=68750

n.a. (n.d.). *Baseline Wander in ECG Signals*. *ScienceDirect*. https://www.sciencedirect.com/topics/computer-science/baseline-wander

Kubíček, A. (2018). *Analýza EKG signálů s využitím Matlabu* [Bakalářská práce, ČVUT]. https://dspace.cvut.cz/bitstream/handle/10467/76084/F3-BP-2018-Kubicek-Adam-BP_Kubicek_elektronicka_opravena.pdf

n.a. (n.d.). *Elektrokardiografické měření*. *VOŠ ČR*. https://www.vovcr.cz/odz/tech/221/page15.html

Harvard Health Publishing. (n.d.). *What your heart rate is telling you*. Harvard Medical School. https://www.health.harvard.edu/heart-health/what-your-heart-rate-is-telling-you

Medical News Today. (n.d.). *What is a dangerous heart rate?*. https://www.medicalnewstoday.com/articles/what-is-a-dangerous-heart-rate#ideal-heart-rates

Heart Foundation New Zealand. (n.d.). *How to check your pulse heart rate*. https://www.heartfoundation.org.nz/wellbeing/managing-risk/how-to-check-your-pulse-heart-rate

Portál.cz. (n.d.). *Jednotlivé poruchy hlasu*. https://obchod.portal.cz/o-portalu/aktuality-z-portalu/jednotlive-poruchy-hlasu

n.a. (2020). *Dysfonie a její léčba*. *Otorinolaryngologie.cz*. https://www.otorinolaryngologie.cz/content/uploads/2020/02/ppp-dysfonie.pdf

Masarykova univerzita. (n.d.). *Funkční a psychogenní poruchy hlasu*. https://is.muni.cz/elportal/estud/pedf/js09/orl/web/pages/7_3_funkcni_a_psychogenni_poruchy_hlasu.html

n.a. (n.d.). *Základy elektroniky – Elektromagnetická interference*. https://www.isibrno.cz/~joe/elektronika/elektronika_9.pdf

Masarykova univerzita. (2017). *Statistika II: Spektrální analýza*. https://is.muni.cz/el/sci/jaro2017/Z2069/um/um/54271981/Statistika_II_3_spektralni_analyza.pdf

Contributors to Wikimedia projects. (n.d.). *Informační entropie*. Wikipedia. https://cs.wikipedia.org/wiki/Informa%C4%8Dn%C3%AD_entropie

StatQuest with Josh Starmer. (2019, May 14). *Entropy (for data science) Clearly Explained!!!* [Video]. YouTube. https://www.youtube.com/watch?v=YtebGVx-Fxw

StatQuest with Josh Starmer. (2020, January 10). *Intuitively Understanding the Shannon Entropy* [Video]. YouTube. https://www.youtube.com/watch?v=0GCGaw0QOhA

WikiSofia. (n.d.). *Analýza řečového signálu*. https://wikisofia.cz/wiki/Anal%C3%BDza_%C5%99e%C4%8Dov%C3%A9ho_sign%C3%A1lu

Contributors to Wikimedia projects. (n.d.). *Cepstrum*. Wikipedia. https://cs.wikipedia.org/wiki/Cepstrum

Contributors to Wikimedia projects. (n.d.). *68–95–99.7 rule*. Wikipedia. https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule

Contributors to Wikimedia projects. (n.d.). *Pravidlo tří sigma*. Wikipedia. https://cs.wikipedia.org/wiki/Pravidlo_t%C5%99%C3%AD_sigma