UNIVERZITA J. E. PURKYNĚ V ÚSTÍ NAD LABEM
Přírodovědecká fakulta

Seminární práce I

z předmětu

Počítačové zpracování signálu (KI/PZS)

Výpočet tepové frekvence z EKG signálu

Detekce anomálií v signálech

ZS 2024/25

Kristina Gavrina

Osobní číslo: F22015

I. Výpočet tepové frekvence z EKG signálu

Zadání:

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. Vás 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.

1. Data

Nejprve jsem si stáhla data a k jejich čtení použila knihovnu wfdb. Data obsahují 18 složek, v každé z nich je soubor .dat, který obsahuje samotný signál, a soubor .hea, který obsahuje metadata o tomto signálu.

Pro zobrazení informací o každém signálu ve formě tabulky jsem napsala funkci extract_ecg_metadata() s využitím knihovny pandas.

```python
def extract_ecg_metadata(record_name: str) -> pd.DataFrame:
    """
    Načte záznam EKG a extrahuje metadata.
    """
    # Načtení dat
    record = wfdb.rdrecord(record_name)

    # Vytvoření tabulky s metadaty
    data = [
        ["Název souboru", record.record_name],  # Název souboru
        ["Vzorkovací frekvence (Hz)", record.fs],  # Vzorkovací frekvence
        ["Délka signálu (vzorky)", record.sig_len],  # Délka signálu (počet vzorků)
        ["Počet kanálů", record.n_sig],  # Počet kanálů
        ["Názvy kanálů", ", ".join(record.sig_name)],  # Názvy kanálů
        ["Jednotky", "uV"],  # Jednotky měření (uV pro EKG)
        ["ADC Gain", record.adc_gain[0] if record.adc_gain is not None else "N/A"],  # Zesilovací koeficient
        ["Baseline (posun)", record.baseline[0] if record.baseline is not None else "N/A"]  # Posun základní linie
    ]

    # Vytvoření DataFrame
    df = pd.DataFrame(data, columns=["Informace", "Hodnota"])

    return 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	uV
6	ADC Gain	1.996
7	Baseline (posun)	-12200

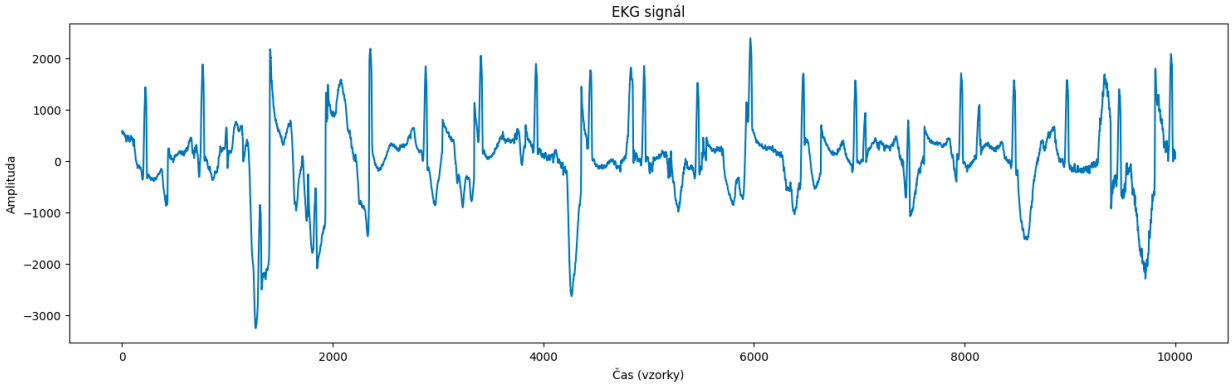
2. Zpracování signálu

Než začneme detekovat R-vlny, je nutné signál předzpracovat.

1. Centralizace signálu

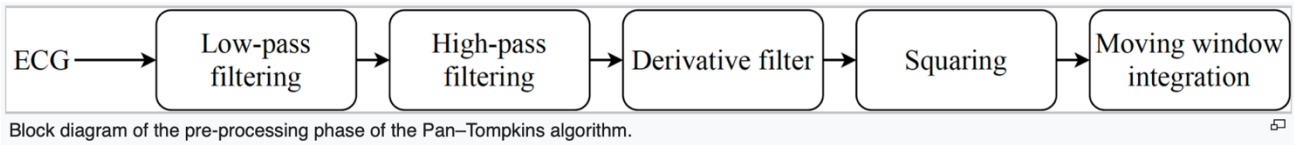
První fáze zpracování: centralizace signálu na vodorovné ose. To je nezbytné, protože signál může obsahovat výkyvy způsobené pohybem pacienta, jeho dýcháním nebo kontaktním šumem elektrod. Kvůli tomu mohou být R-peaky posunuté nahoru nebo dolů. Centralizace signálu usnadňuje další filtraci těchto výkyvů. Obvykle se centralizace signálu dosahuje odečtením průměrné hodnoty signálu.

```python
def center_signal(ecg_signal):
    """
    Centrovaní signálu - odstranění střední hodnoty.
    """
    return ecg_signal - np.mean(ecg_signal)
```



2. Pan-Tompkinsův algoritmus.

Druhá fáze zpracování – algoritmus Pan-Tompkins.



Block diagram of the pre-processing phase of the Pan–Tompkins algorithm.

Tento algoritmus zahrnuje:

1) Filtrování signálu
2) Derivaci signálu
3) Umocnění signálu
4) Integrované klouzavé okno
5) Detekci R-peaků

2.1. Filtrování signálu

V této dílčí fázi čistím signál od nízkofrekvenčního šumu, způsobeného pohybem pacienta a změnami polohy elektrod, a vysokofrekvenčního šumu, který vzniká v důsledku činnosti senzorů, svalové aktivity a elektromagnetického rušení.

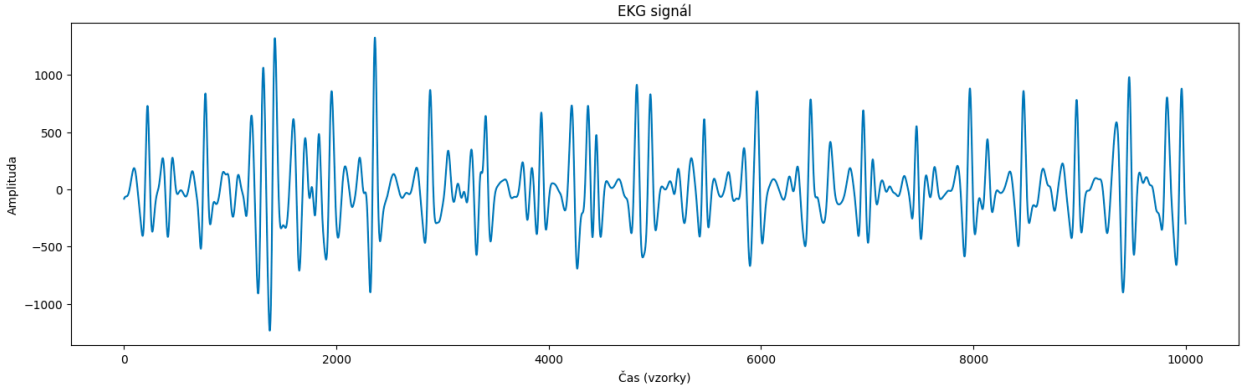
Abych nemusela vytvářet samostatně filtr pro odstranění vysokých frekvencí (high-pass filter) a filtr pro odstranění nízkých frekvencí (low-pass filter), používám pásmový filtr (band-pass filter), který kombinuje oba tyto typy filtrace.

```python
def bandpass_filter(signal, fs, lowcut=5, highcut=15, order=2):
    """
    Pásmová propust Butterworthova filtru.
    """
    nyquist = 0.5 * fs  # Nyquistova frekvence
    low = lowcut / nyquist
    high = highcut / nyquist

    # Vytvoření koeficientů Butterworthova filtru
    b, a = butter(order, [low, high], btype='band')

    # Dvojnásobná filtrace pro minimalizaci fázového zkreslení
    filtered_signal = filtfilt(b, a, signal)

    return filtered_signal
```



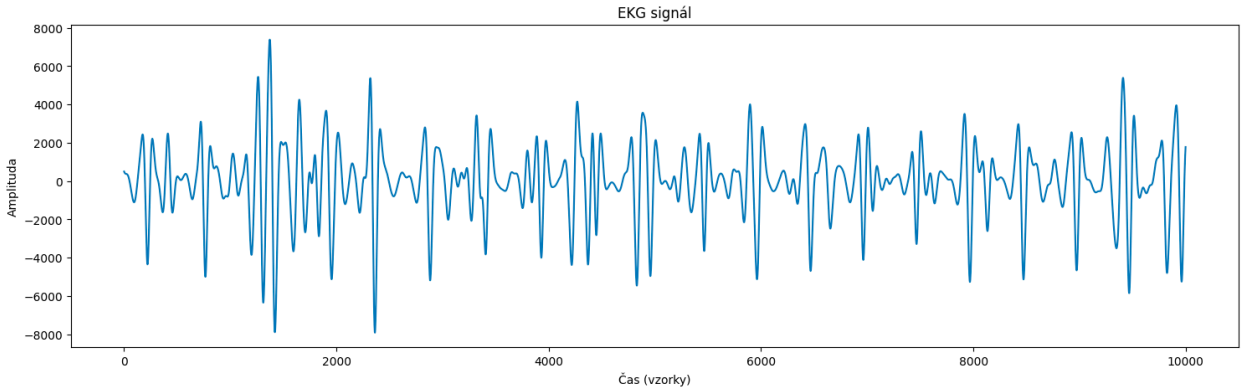
2.2. Derivace signálu.

Derivace signálu je potřebná k zvýraznění QRS komplexu a potlačení P a T vln. Výsledkem je signál, ve kterém jsou R-píky lépe viditelné.

V algoritmu Pana Tompkinse se používá pětibodová diferenciace.

$$Y(n) = \frac{1}{8}(-x(n-2) - 2x(n-1) + 2x(n+1) + x(n+2))$$

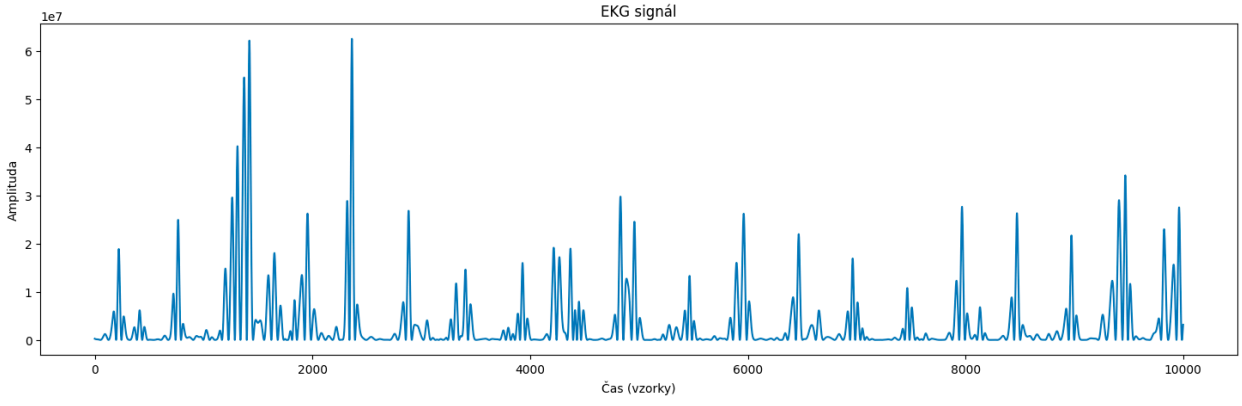
```
def derivative_filter(signal):
    """
    y(n)= 1/8* [-x(n-2)-2x(n-1)+2x(n+1)+x(n+2)]
    """
    kernel = np.array([-1, -2, 0, -2, -1])
    return np.convolve(signal, kernel, mode='same')
```



2.3. Umocnění na druhou

Dalším krokem je umocnění signálu na druhou, aby se signál stal kladným a aby se zvýraznil rozdíl mezi R-píky a šumem.

$$y(n) = x(n)^2$$

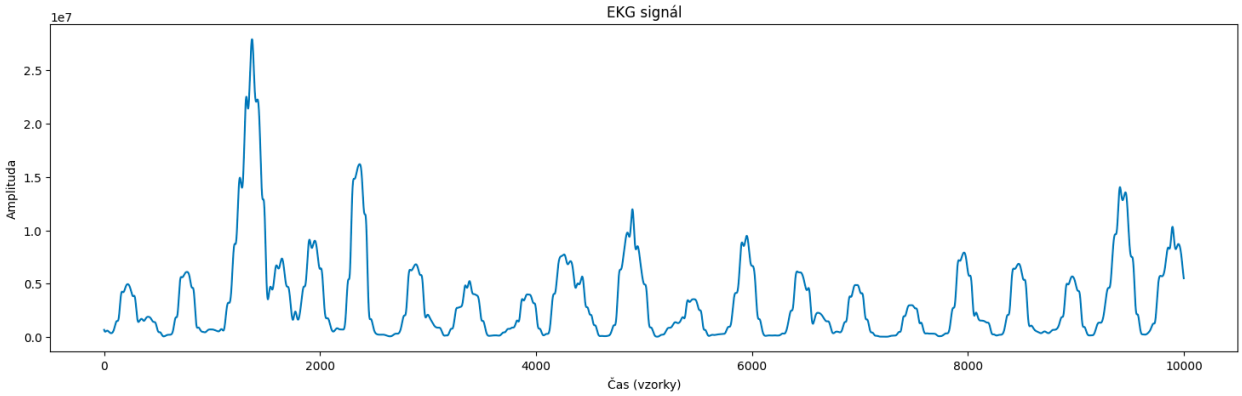


2.4. Integrální klouzavé okno

Po umocnění signálu na druhou stále obsahuje drobný šum. K jeho odstranění se používá vyhlazování signálu (integrální klouzavé okno), které vytváří hladký obal QRS komplexů. N – šířka okna.

$$N = \frac{150 \times fs}{1000}$$

```
def moving_window_integration(signal, fs):
    """
    Aplikuje klouzavý průměr na signál.
    """
    window_size=150
    n = int(window_size * fs / 1000 )
    window = np.ones(n) / n
    return np.convolve(signal, window, mode='same')
```



2.5. Detekce R-píků

Posledním krokem v algoritmu Pana Tompkinse je detekce R-peaků. K tomu používám funkci find_peaks() z knihovny scipy.signal. Tato funkce hledá maximální hodnoty v signálu, čímž identifikuje R-píky.

Pro použití této funkce je třeba nastavit height - minimální výšku píku, aby byl odmítnut zbývající šum, a distance - minimální vzdálenost mezi p-píky.

Vzdálenost se vypočítá podle vzorce:

$$\text{distance} = \frac{RR\ interval \times fs}{1000}$$

Podle zdroje Medscape[1] je normální RR interval v rozmezí 0,6–1,2 sekundy. Proto volím RR interval 0,4 sekundy, aby bylo možné zachytit i nestandardní hodnoty.

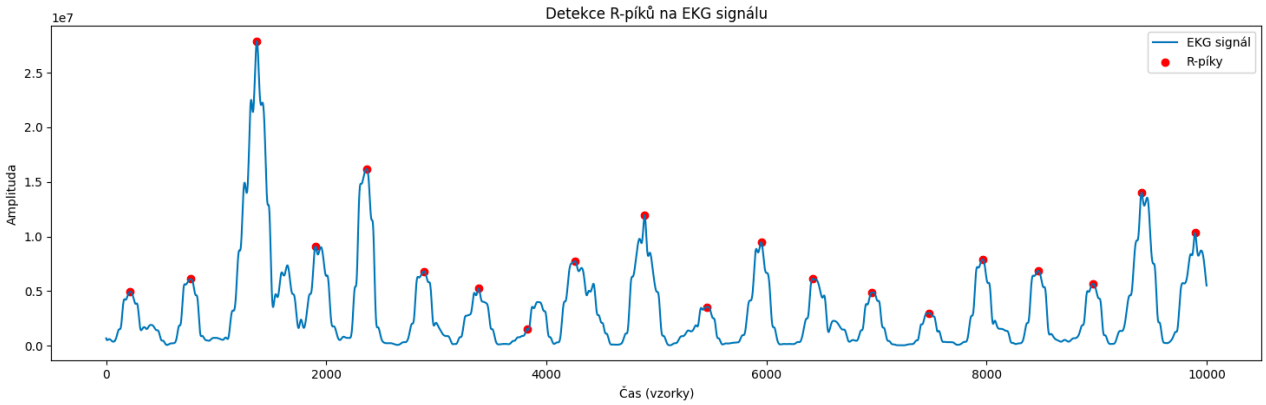
Prahovou hodnotu počítám podle vzorce:

$$\text{height} = \text{median}(\text{vyhlazený signál}) + 0.3 \times \max(\text{vyhlazený signál})$$

```python
def detect_r_peaks(signal, fs):
    rr_interval = 400

    distance = int(rr_interval*fs/1000)
    height = height = np.median(signal) + 0.3 * np.std(signal)

    peaks, _ = find_peaks(signal, height=height, distance=distance)
    return peaks
```



3. Výpočet tepové frekvence

Po zpracování signálu lze přistoupit k výpočtu tepové frekvence.

Tepovou frekvenci počítám podle vzorce:

$$HR = \frac{60}{RR\ interval\ in\ seconds}$$

RR interval je vzdálenost mezi dvěma po sobě jdoucími R-píky. V mém kódu používám průměrnou hodnotu RR intervalů pro výpočet HR (BPM).

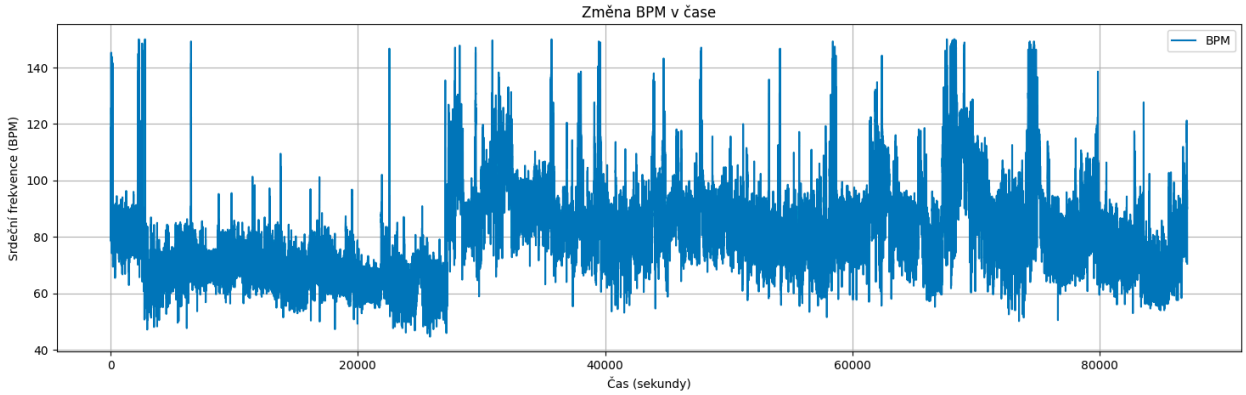
[1] https://emedicine.medscape.com/article/2172196-overview?form=fpf#a1

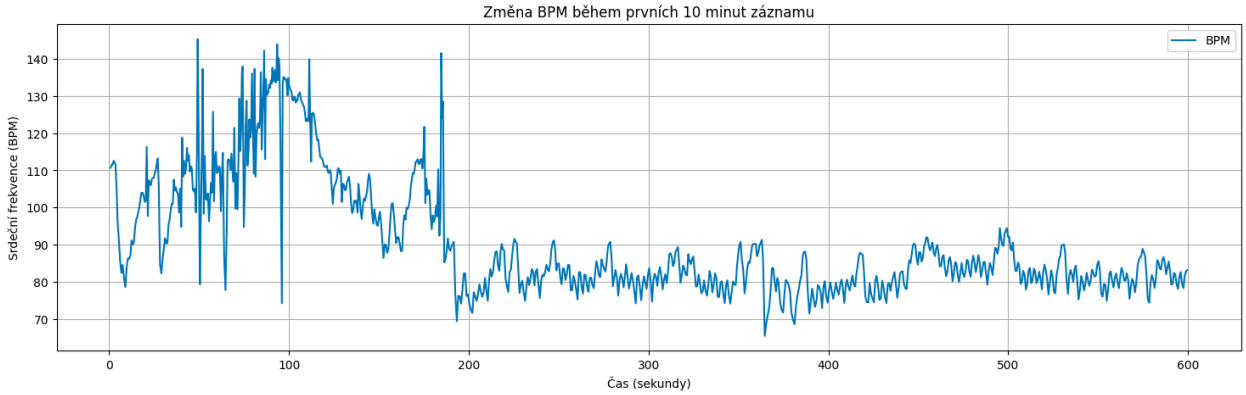
```python
def calculate_bpm(r_peaks, fs):
    """
    Vypočítá průměrnou srdeční frekvenci (BPM) na základě detekovaných R-vrcholů.
    """
    rr_intervals = np.diff(r_peaks) / fs

    mean_rr = np.mean(rr_intervals)

    bpm = 60 / mean_rr

    return bpm
```





4. Testování přesnosti kódu

Poté, co byl kód pro detekci R-peaků a výpočet tepové frekvence dokončen, jsem jej otestovala na datech MIT-BIH Normal Sinus Rhythm Database (PhysioNet).

V těchto datech jsou kromě souborů .dat a .hea důležité také soubory .atr, které obsahují informace o skutečných R-peaků v signálu. Tyto hodnoty jsem použila ve funkci evaluate_detection(), abych je porovnala s R-píky nalezenými mým algoritmem zpracování signálu.

5. Výsledky testování

Bohužel výsledky byly horší, než jsem očekávala. Pouze u malé části signálů přesnost detekce přesáhla 50 %.

Pravděpodobnou příčinou je použití funkce np.isin(detected_r_peaks, true_r_peaks), která kontroluje pouze přesnou shodu indexů R-peaků. V reálných datech však mohou být místa výskytu R-peaků mírně posunuta, což vede ke snížení procenta úspěšné detekce.

	File	Correct	Wrong	Total	Success rate
0	16795	41760	45142	86902	48.054130
1	16420	33409	68677	102086	32.726329
2	19088	14414	85234	99648	14.464917
3	16786	0	101632	101632	0.000000
4	18184	166	102328	102494	0.161961
5	16483	12286	92050	104336	11.775418
6	19830	39645	70608	110253	35.958205
7	16265	114	100366	100480	0.113455
8	16272	36655	45704	82359	44.506368
9	19140	2055	94759	96814	2.122627
10	16273	107	89735	89842	0.119098
11	16539	80746	27536	108282	74.570104
12	17052	1965	85592	87557	2.244252
13	18177	29649	86547	116196	25.516369
14	19090	19353	62260	81613	23.713134
15	17453	57817	42855	100672	57.431063
16	19093	43637	31602	75239	57.997847
17	16773	1273	80714	81987	1.552685

II. Detekce anomálií v signálech

Zadání:

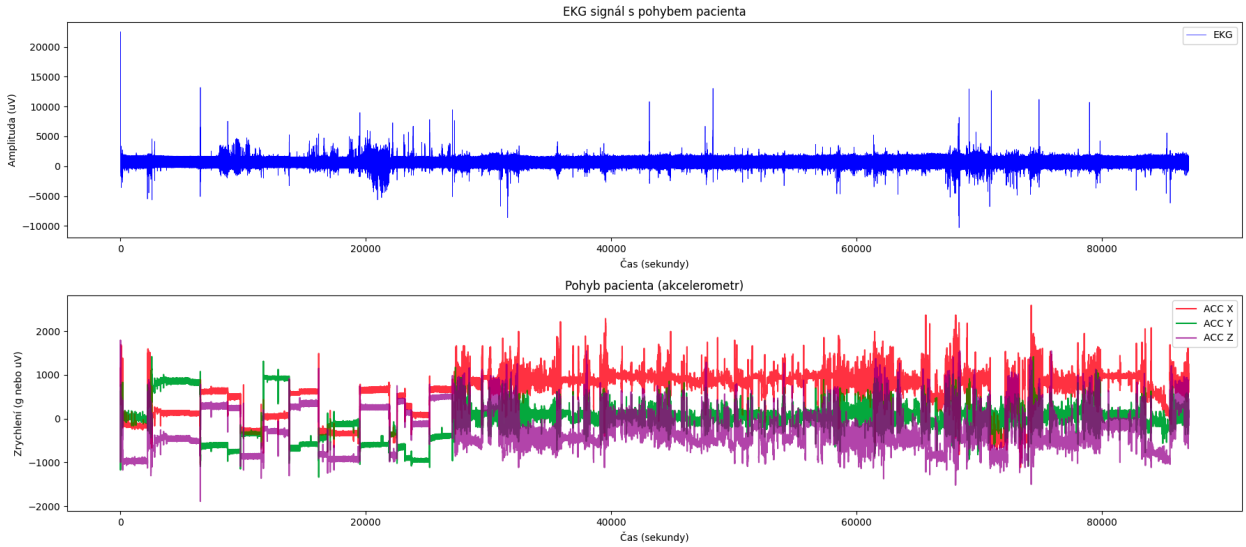
Ve zdrojové databázi najdete celkem 18 měření EKG obsahující úplné (3 signály) nebo částečné anotace událostí (P,T vlny a QRS komplex). Záznamy EKG obsahují i části, které jsou porušeny vlivem anomálií (vnější rušení, manipulace s pacientem apod.). Navrhněte způsob, jak detekovat tyto úseky a prezentujte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

1. Data

Pro tento úkol jsem použila nejen soubory EKG, ale také soubory ANN. Soubor ANN.csv obsahuje anotace. Jsou v něm 3 anotace a každá má tři sloupce (počáteční index úseku, konečný index úseku, hodnocení kvality od 0 do 3).

	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	uV
6	ADC Gain	1.996
7	Baseline (posun)	-12200

	Informace	Hodnota
0	Název souboru	100001_ACC
1	Vzorkovací frekvence (Hz)	100
2	Délka signálu (vzorky)	8708700
3	Počet kanálů	3
4	Názvy kanálů	ACCx, ACCy, ACCz
5	Jednotky	uV
6	ADC Gain	17.6975
7	Baseline (posun)	-13052



2. Metoda detekce anomálií

Pro identifikaci anomálií jsem zvolila prahovou metodu.

Prvním krokem je určení prahu, při jehož překročení budou hodnoty považovány za anomálie. Pro stanovení prahových hodnot jsem použila metodu „3 sigma“.

Průměrná hodnota (μ) – reprezentuje „centrální“ hodnotu dat, která udává jejich celkovou úroveň.

Směrodatná odchylka (σ) – vyjadřuje, jak moc se hodnoty odchylují od průměru.

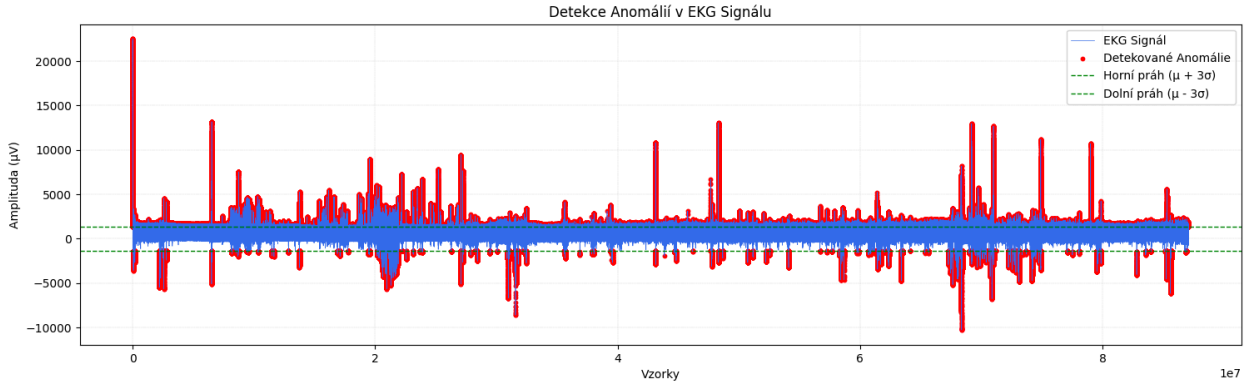
Podle zákona normálního rozdělení se 99,7 % všech hodnot nachází v rozmezí (μ ± 3σ). Pokud hodnota překročí tento interval, je považována za anomálii.

Nejprve jsem vypočítala průměr (μ) a směrodatnou odchylku (σ) pomocí funkcí z knihovny NumPy. Poté jsem určila dolní a horní práh podle vzorců:

$$Upper = \mu + 3\sigma$$
$$Lower = \mu - 3\sigma$$

Nakonec jsem identifikovala všechny hodnoty, které překračují stanovené prahy.

```python
def detect_anomalies(signal, upper_threshold, lower_threshold):
    """
    Detekuje anomálie v signálu na základě horního a dolního prahu.
    """
    anomaly_indices = np.where((signal > upper_threshold) | (signal < lower_threshold))[0]
    return anomaly_indices
```

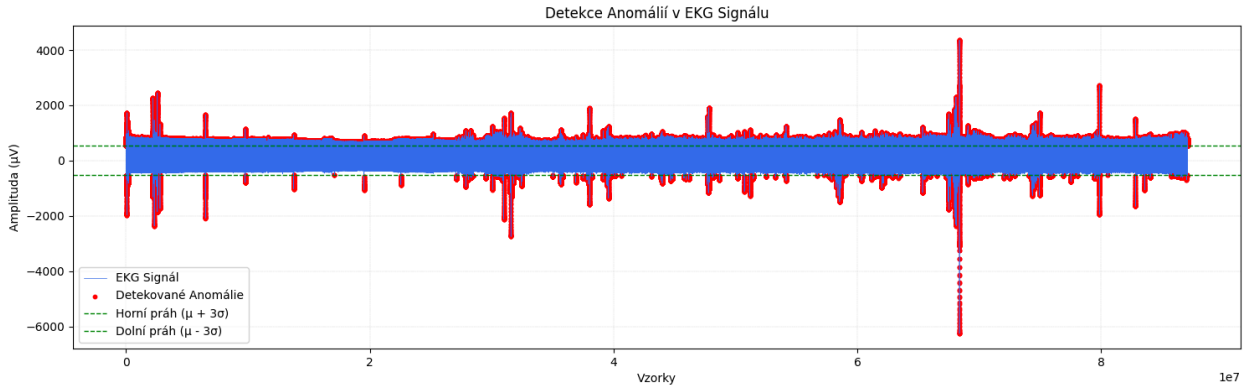


A posledním krokem byl výpočet úspěšnosti algoritmu.

	Metrika	Hodnota
0	Počet anomálií (dle anotací)	1147
1	Počet detekovaných anomálií	2289938
2	Počet shodných anotací	102
3	Úspěšnost detekce	8.89 %

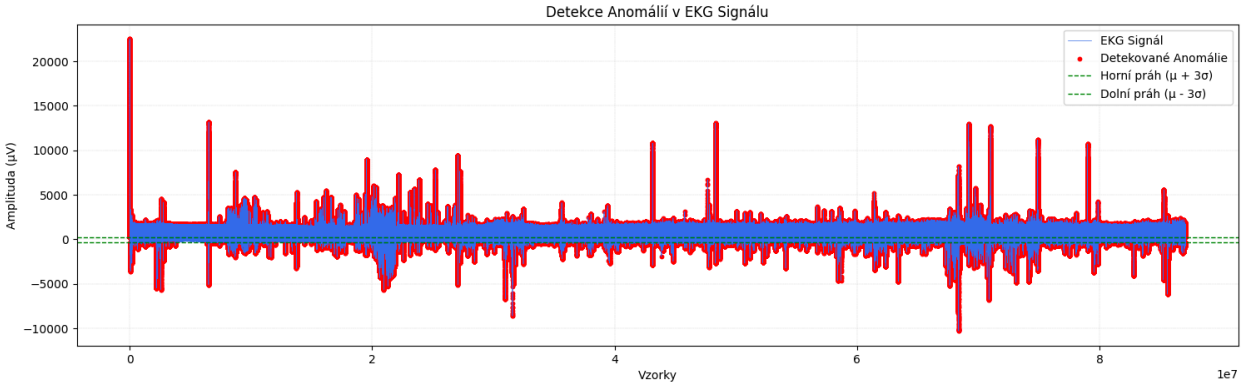
Po prohlédnutí výsledků jsem se rozhodla nejprve zpracovat signál pomocí části algoritmu Pana Tompkinse. Centralizovala jsem signál a odstranila vysokofrekvenční a nízkofrekvenční šumy.

Předzpracování signálu před detekcí anomálií mírně zvýšilo počet identifikovaných anomálií, ale nemělo vliv na procentuální úspěšnost.



	Metrika	Hodnota
0	Počet anomálií (dle anotací)	1147
1	Počet detekovaných anomálií	2697280
2	Počet shodných anotací	102
3	Úspěšnost detekce	8.89 %

Protože předzpracování signálu neovlivnilo úspěšnost detekce, rozhodla jsem se vypočítat prahové hodnoty pomocí mediánové absolutní odchylky (MAD) místo směrodatné odchylky (σ). Ani tato metoda však nezlepšila konečný výsledek.



	Metrika	Hodnota
0	Počet anomálií (dle anotací)	1147
1	Počet detekovaných anomálií	15332475
2	Počet shodných anotací	102
3	Úspěšnost detekce	8.89 %

Literatura

A real-time QRS detection algorithm. (n.d.). https://www.robots.ox.ac.uk/~gari/teaching/cdt/A3/readings/ECG/Pan+Tompkins.pdf

Lawrence Rosenthal, M. (2024, October 29). *Normal electrocardiography (ECG) intervals*. Normal Electrocardiography Intervals. https://emedicine.medscape.com/article/2172196-overview?form=fpf#a1

Wikimedia Foundation. (2025, January 21). *Heart rate*. Wikipedia. https://en.wikipedia.org/wiki/Heart_rate