PCPS — Zastosowania

Jan Kowalski

13 listopada 2020

Spis treści

Spis rysunków

Jak ogarnąć środowisko w jakim pracujemy

Całość rozwiązania ma znajdować się w pliku '`zastosowania.m`'. Edytor należy włączyć za pomocą:

```
$ make edit
```

dla sekcji preferujących edytor `vim` lub:

```
$ make edit-kate
```

jeśli preferuje się edytor kate (graficzny).

W pliku tym piszemy równocześnie trzy rzeczy:

- kod programu interpretowany przez GNU Octave (wolna implementacja Matlab),

- komentarze do kodu, za pomocą linii zaczynających się od '`%`' (jak standardowy komentarz w Octave/Matlab),
- wnioski umieszczane w raporcie, za pomocą linii zaczynających się od '`%%`'.

  Raport kompilujemy za pomocą:

  ```
  $ make
  ```

  Raport można wyświetlić za pomocą:

  ```
  $ make show
  ```

  Cały raport generowany jest automatycznie. Zarówno kod programu jak i wnioski dodawane są automatycznie do raportu. Wykresy robione są automatycznie, ale program **musi** umieścić dane do wykresów w odpowiednich plikach. Nazwy plików znajdują się w opisach zadań, np. '`x_1`'. Dane należy zapisywać za pomocą funkcji `writedata()`. Funkcja ma następujący prototyp:

  `writedata(`*name*`, `*x*`, `*y*`)`

  name to nazwa pliku do jakiego mają być zapisane dane, *x* to wektor danych na osi x, *y* to wektor danych na osi y. Dla wykresów czasowych wektor *x* ma reprezentować czas w sekundach. Dla wykresów gęstości widmowej mocy (GWM) wektor *x* ma reprezentować częstotliwość w hercach, a wektor *y* wartości GWM. Przykład:

  ```
  t = [0; 0.1; 0.2; 0.3];
  x = [1; 2; 3; 4];
  writedata('x_1', t, x);
  ```


Fragmenty oznaczone takim znakiem są na dodatkowe punkty, bez nich można uzyskać 10 punktów.

Deklaracje wstępne

```
function writedata(name, x, y)
        x = vec(x);
        y = vec(y);
        if (length(x) != length(y))
                fprintf(stderr, "error: signal: '%s', length(x) = %d, length(y) =
%d", name, length(x), length(y));
                return;
        end
        f = fopen(name, "w");
        y(find(isinf(y))) = -1e6;
        fprintf(f, "%f %f\n", [x y]');
        fclose(f);
end

if (!exist("audioread"))
        function [x, f_s] = audioread(file)
                [x, f_s] = wavread(file);
```

```
        end
end
```

1 Przetwarzanie sygnału audio

1.1 Zadanie

1. Odczytaj sygnał audio z pliku '`x.wav`' za pomocą funkcji `audioread()`.
2. Zapisz przebieg czasowy sygnału do pliku '`x_1`'.
3. Wyznacz GWM sygnału, pomijając pierwszą sekundę, i zapisz do pliku '`X_1`'. Należy użyć funkcji `pwelch()`.
4. Wyznacz spektrogram sygnału, i zapisz go jako macierz w pliku '`S`'. Należy użyć funkcji `specgram()`. Wektory częstotliwości i czasu zapisać w plikach '`SF`' i '`ST`'.
5. Odsłuchaj sygnał używając:

   ```
   $ make play-x
   ```
6. Na podstawie przebiegu czasowego, GWM i spektrogramu odpowiedz we wnioskach na pytania:
   - czy sygnał jest stacjonarny?
   - co w sygnale jest zakłóceniem?
   - czy zakłócenia są stacjonarne?
7. zaproponuj 2 różne filtry usuwające zakłócenie, charakterystyki amplitudowe filtów zapisz do plików '`H_1`' oraz `H_2`. GWM przefiltrowanych danych zapisz do plików '`X_2`' oraz '`X_3`'. Zapisz sygnały wyjściowe jako '`y1.wav`' oraz '`y2.wav`', używając funkcji `audiowrite()`. Zagwarantuj, że nie następuje nasycenie w tych sygnałach.

   Sygnał można odtworzyć za pomocą:

   ```
   $ make play-y1
   ```

   oraz

   ```
   $ make play-y2
   ```

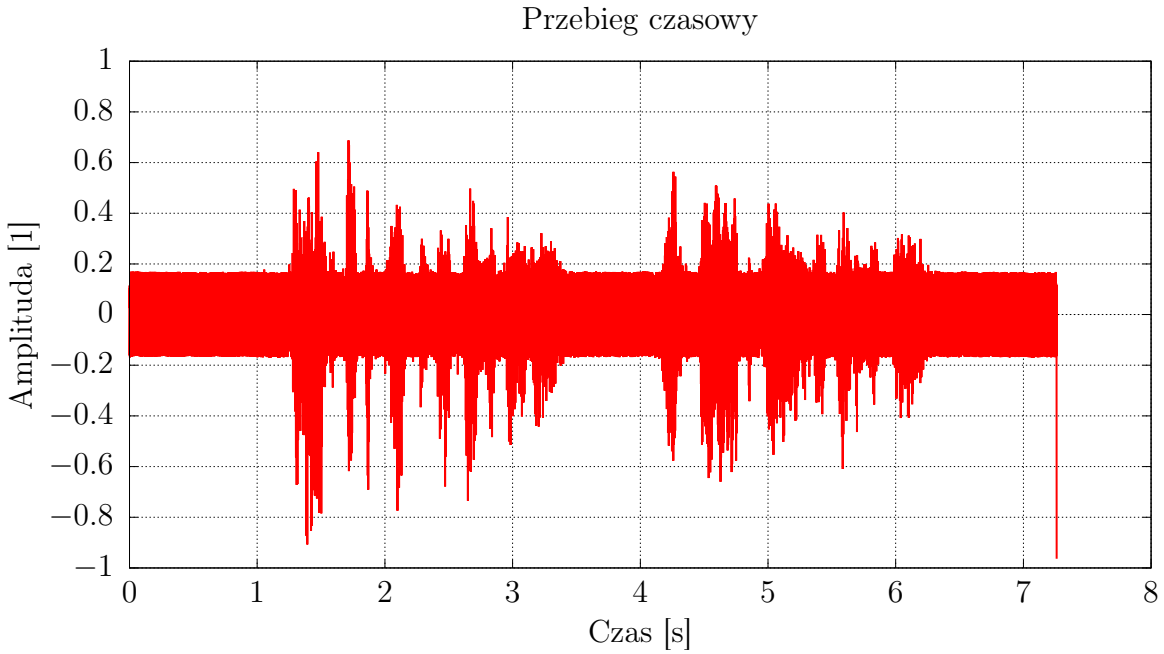
1.2 Rozwiązanie

Przebieg czasowy sygnału został umieszczony na rysunku 1. Spektrogram sygnału został umieszczony na rysunku 2.

Charakterystyki filtrów zostały umieszczone na rysunku 3.

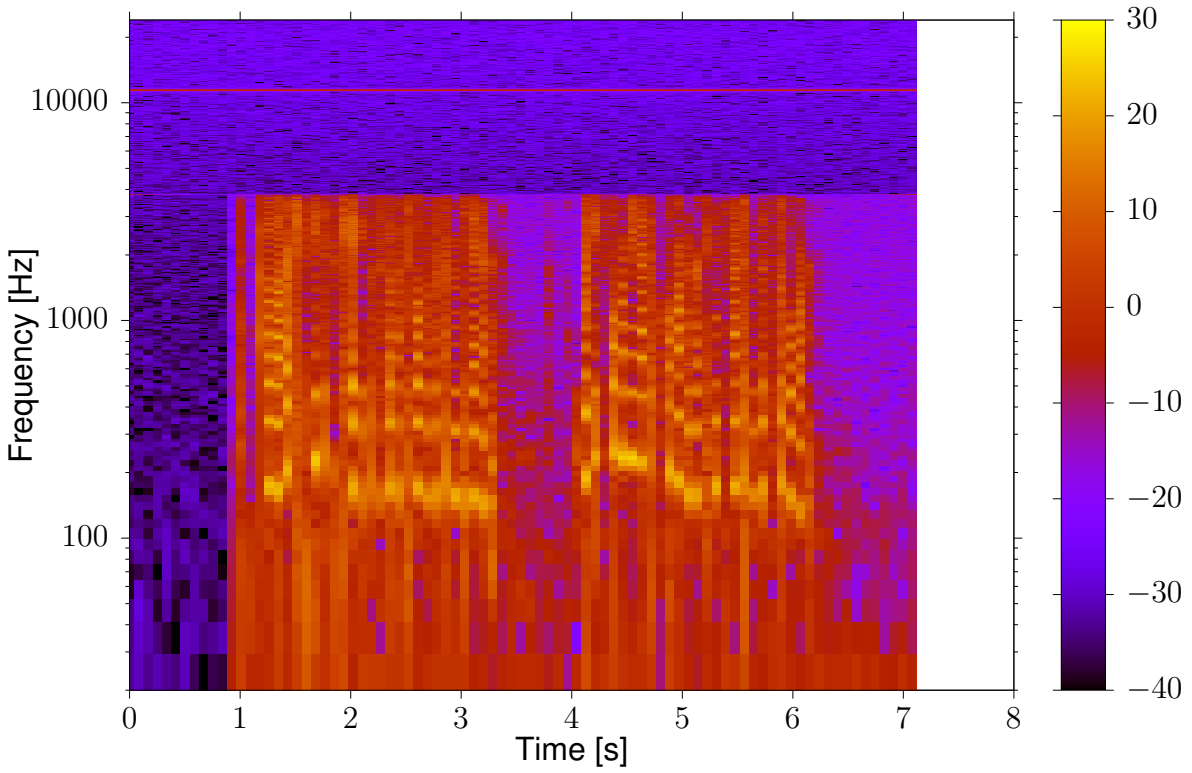
GWM sygnałów przed i po filtracji zostały umieszczone na rysunku 4.

```
[x_1, f_s] = audioread("x.wav");
t = (0:length(x_1) - 1) / f_s;
```

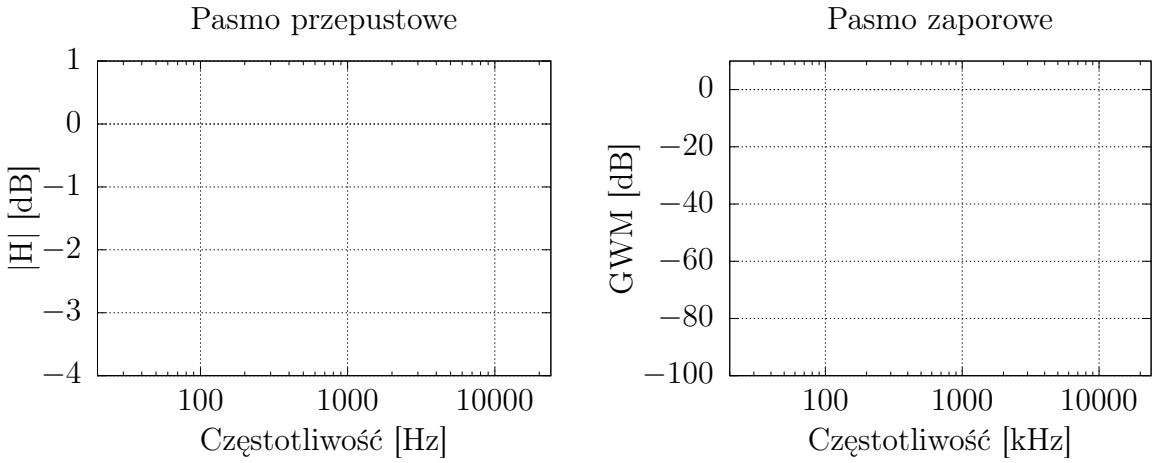


Rysunek 1: Przebieg czasowy sygnału oryginalnego.

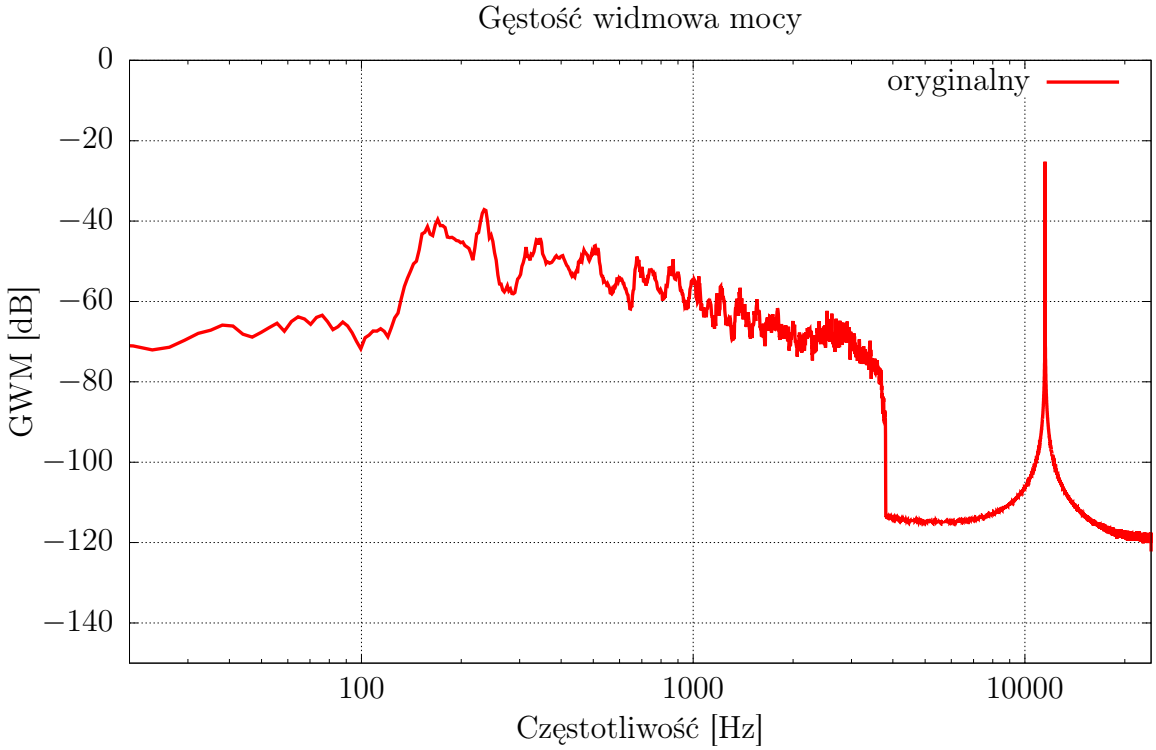
```
[S, F, T] = specgram(x_1, 4096, f_s, 4096);
S = 10*log10(abs(S));
F = vec(F);
T = vec(T);
save('S', 'S');
save('SF', 'F');
save('ST', 'T');

writedata('x_1', t, x_1);
[X_1, f] = pwelch(x_1(f_s:end), 16384, [], 16384, f_s);
writedata('X_1', f, 10*log10(abs(X_1)));
```



Rysunek 2: Spektrogram sygnału oryginalnego.



Rysunek 3: Charakterystyki filtrów.



Rysunek 4: Porównanie GWM przed i po filtracji.

2 Przetwarzanie sygnału EKG

2.1 Zadanie

1. Odczytaj sygnał audio z pliku 'ekg' za pomocą funkcji `load()`.
2. Zapisz przebieg czasowy sygnału do pliku '`y_1`'
3. Wyznacz GWM sygnału, pomijając pierwszą sekundę, i zapisz do pliku '`Y_1`'. Należy użyć funkcji `pwelch()`.
4. Wyznacz spektrogram sygnału, i zapisz go jako macierz w pliku '`S2`' (za pomocą funkcji `save()`). Należy użyć funkcji `specgram()`. Wektory częstotliwości i czasu zapisać w plikach '`SF2`' i '`ST2`' (za pomocą funkcji `save()`).
5. Na podstawie przebiegu czasowego, GWM i spektrogramu odpowiedz we wnioskach na pytania:
   - czy sygnał jest stacjonarny?
   - co w sygnale jest zakłóceniem?
   - czy zakłócenia są stacjonarne?
6. zaproponuj 2 różne filtry usuwające zakłócenie, charakterystyki amplitudowe filtów zapisz do plików '`H_3`' oraz `H_4`. GWM przefiltrowanych danych zapisz do plików '`Y_2`' oraz '`Y_3`'. Zapisz przebiegi czasowe sygnału do plików '`y_2`' oraz '`y_3`'.

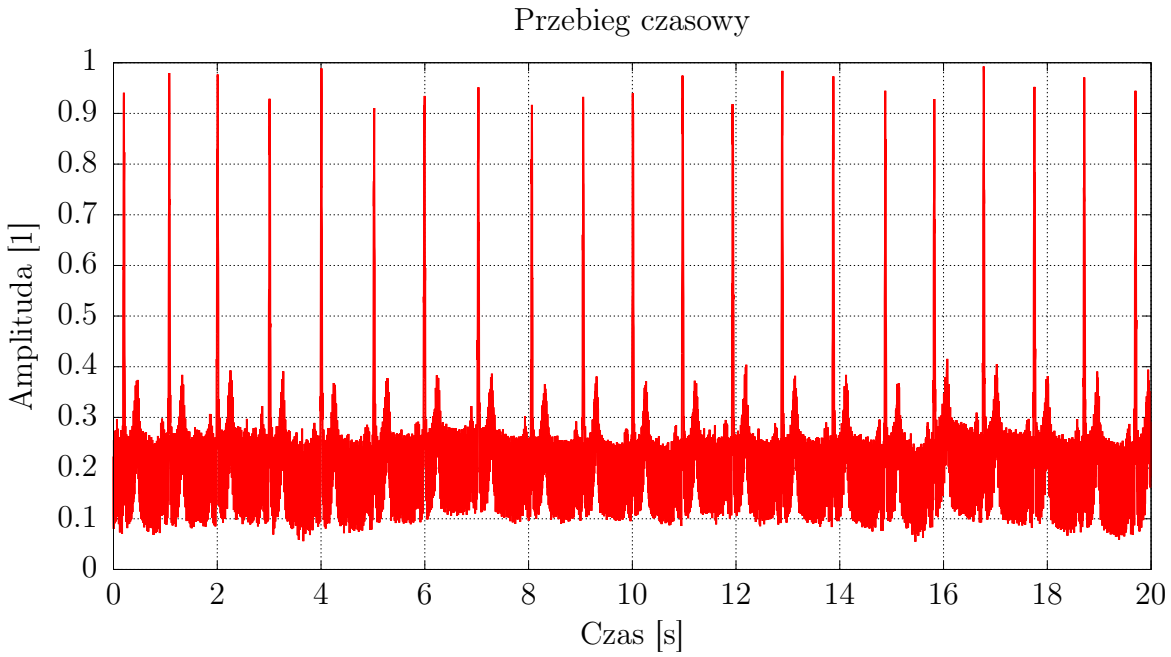
2.2 Rozwiązanie

Przebieg czasowy sygnału został umieszczony na rysunku 8. Spektrogram sygnału został umieszczony na rysunku 6.

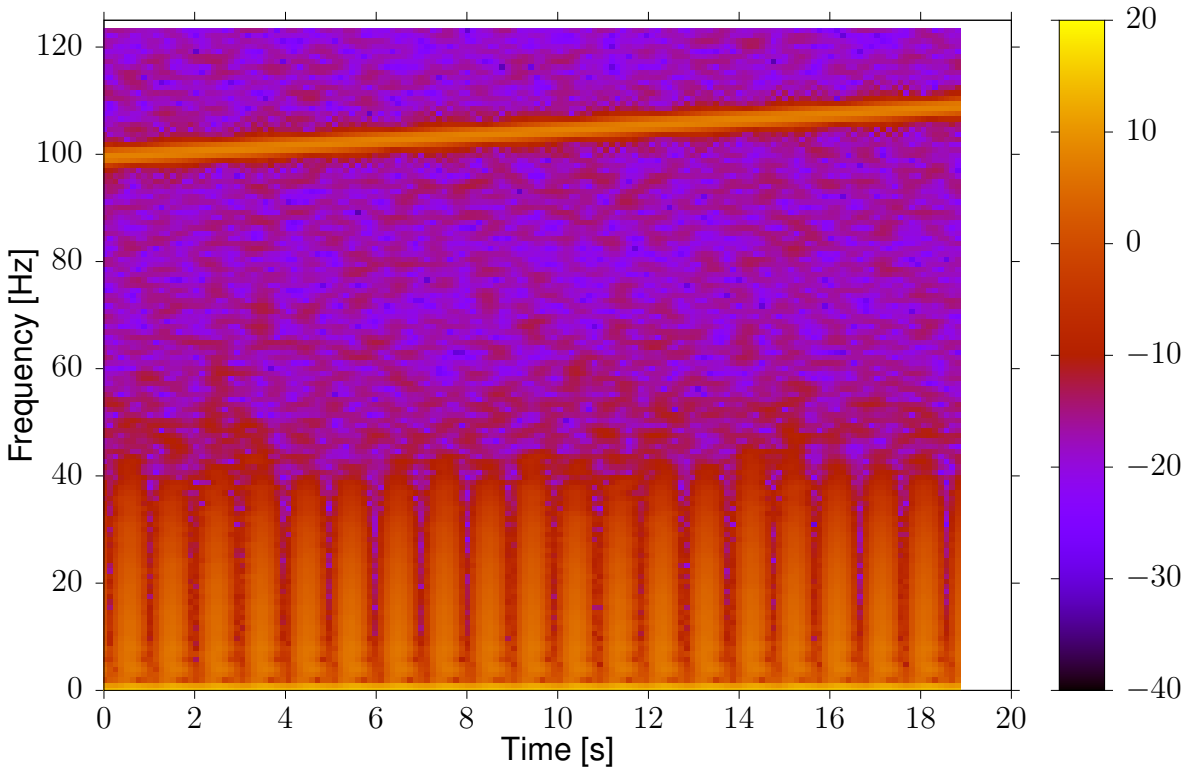
Charakterystyki filtrów zostały umieszczone na rysunku 3.

Porównanie przebiegów czasowych i GWM przed i po filtracji zostało umieszczone na rysunkach 8 i 9.

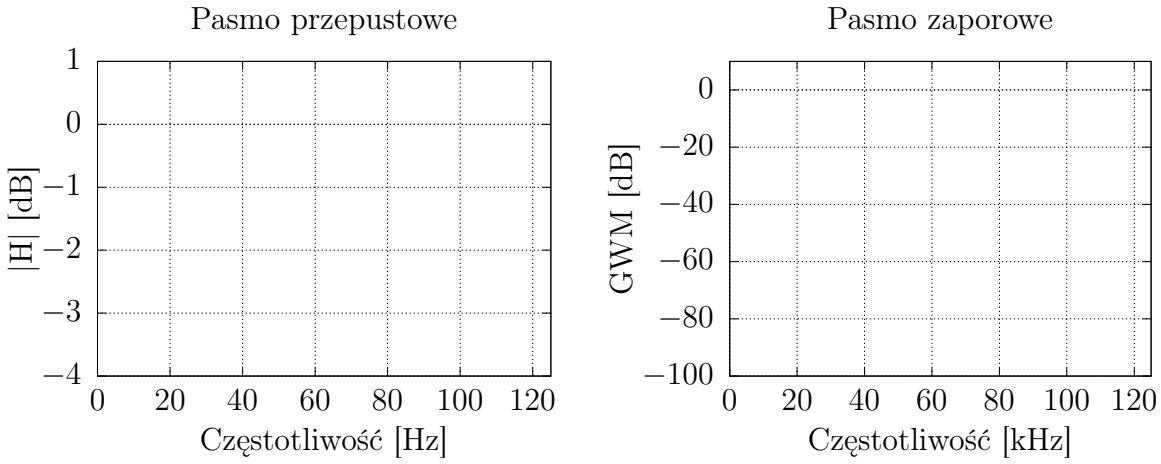
```
y_1 = load("ekg");
f_s = 250;
t = (0:length(y_1) - 1) / f_s;
writedata('y_1', t, y_1);

[S, F, T] = specgram(y_1, 256, f_s, 256, 256 - 32);
S = 10*log10(abs(S));
F = vec(F);
T = vec(T);
save('S2', 'S');
save('SF2', 'F');
save('ST2', 'T');

[Y_1, f] = pwelch(y_1(f_s:end), 512, [], 512, f_s);
writedata('Y_1', f, 10*log10(abs(Y_1)));
```



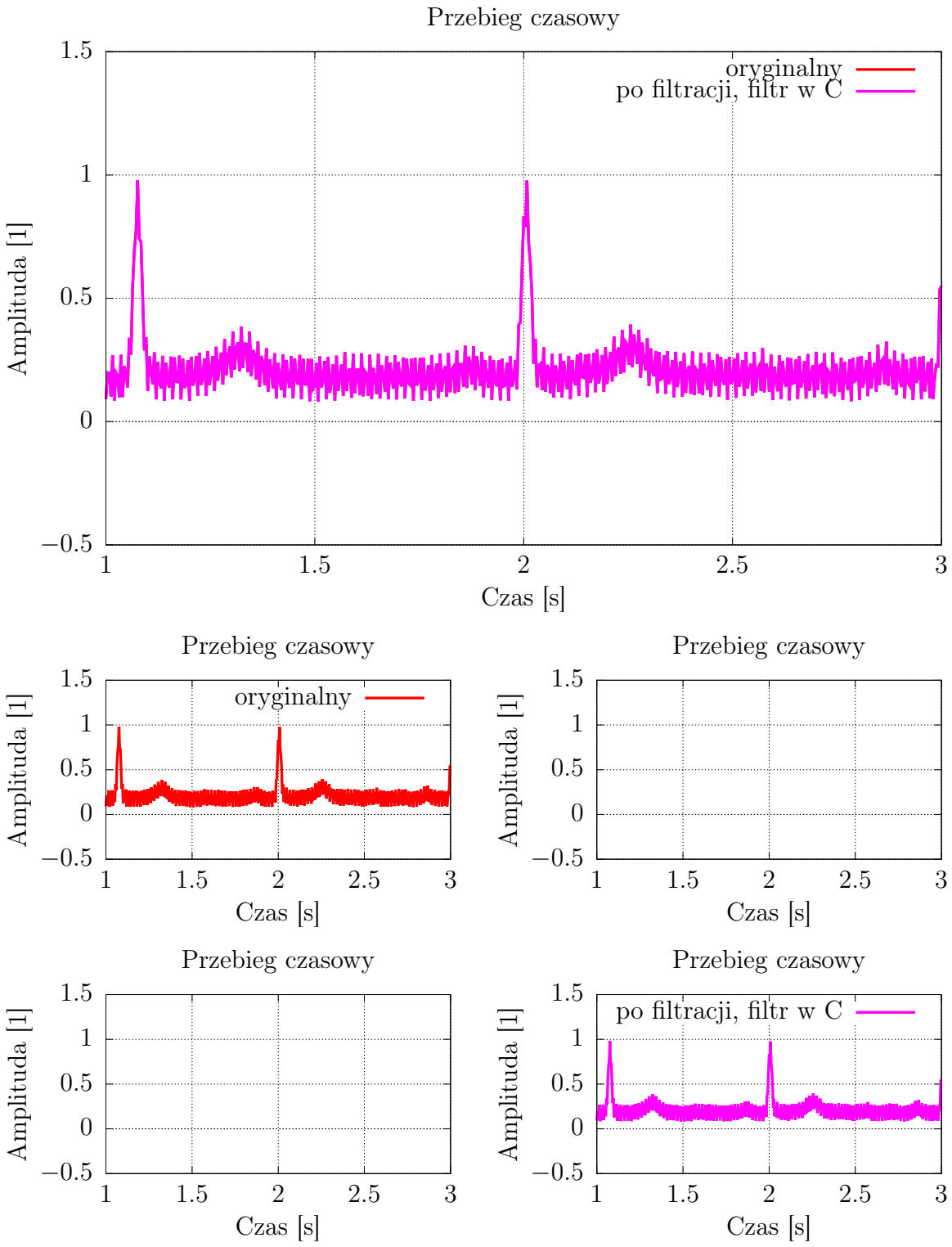
Rysunek 5: Przebieg czasowy sygnału oryginalnego.



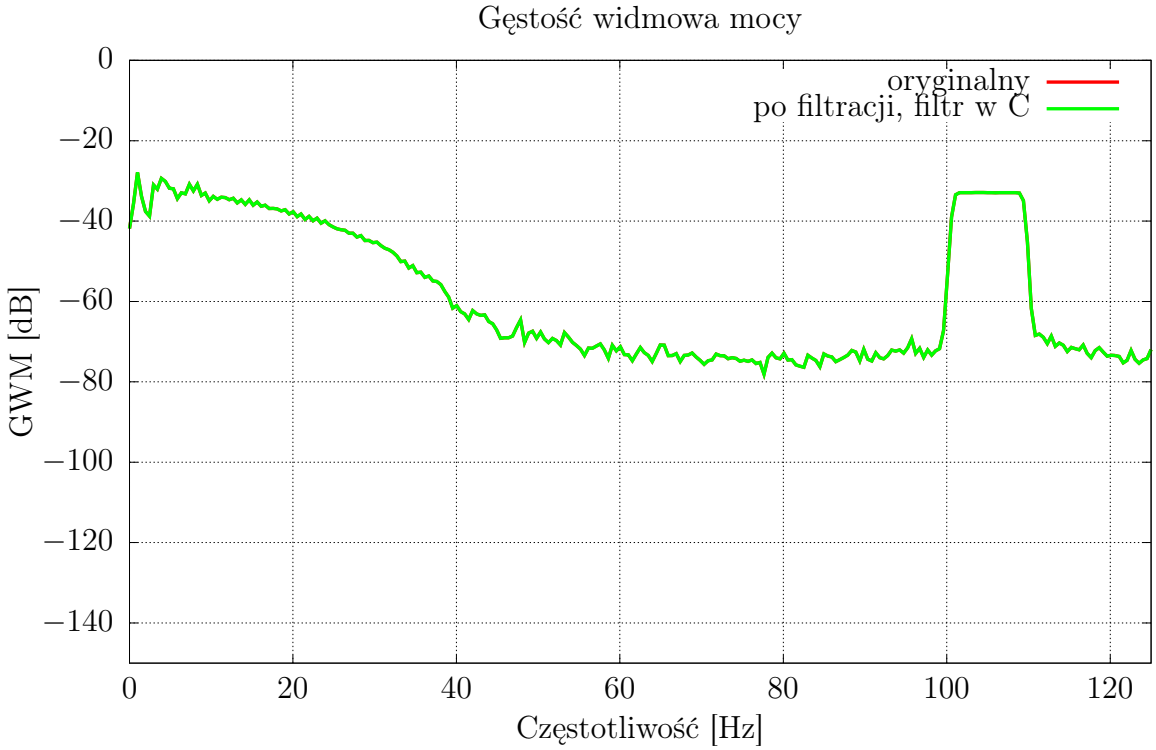
Rysunek 6: Spektrogram sygnału oryginalnego.



Rysunek 7: Charakterystyki filtrów.



Rysunek 8: Porównanie przebiegów czasowych przed i po filtracji.



Rysunek 9: Porównanie GWM przed i po filtracji.

3 Realizacja filtru cyfrowego

3.1 Zadanie

Zaimplementuj w języku C jeden z filtrów używanych dla sygnału EKG. Dane należy czytać ze standardowego wejścia, jako liczby rzeczywiste zapisane w ASCII. Dane wyjściowe należy zapisywać także w formacie ASCII, ze znakiem nowej linii ('\n') pomiędzy liczbami.

Implementacja ma znajdować się w pliku '`filter.c`'. Edytor należy włączyć za pomocą:

```
$ make edit-filter
```

dla sekcji preferujących edytor `vim` lub:

```
$ make edit-filter-kate
```

jeśli preferuje się edytor kate (graficzny).

3.2 Rozwiązanie

Wykresy przebiegów czasowych i GWM znajdują się na rysunkach 8 i 9.

```
#include <stdio.h>

int main(void)
{
        double x, y;

        while (!feof(stdin)) {
                if (scanf("%lf", &x) < 1)
                        continue;
                y = x;
                printf("%f\n", y);
        }
}
```