Metody numeryczne

Nikodem Kocjan

21.05.2024

Aproksymacja syngału okresowego przy użyciu FFT

Spis treści

1 Cel ćwiczenia

Celem ćwiczenia jest zastosowanie FFT do odszumienia sygnału periodycznego.

2 Wstęp teorytyczny

Fourierowska transformata (FFT) jest algorytmem umożliwiającym przekształcenie sygnału z dziedziny czasu do dziedziny częstotliwości. FFT jest szczególnie użyteczna w analizie sygnałów, ponieważ umożliwia identyfikację składników częstotliwościowych sygnału, co jest trudne do osiągnięcia w dziedzinie czasu.

2.1 Szybka Transformata Fouriera (FFT)

Algorytm FFT jest efektywną implementacją DFT, redukującą złożoność obliczeniową z $O(N^2)$ do $O(N \log N)$. Dzięki temu FFT jest powszechnie stosowana w analizie sygnałów, przetwarzaniu obrazów oraz w wielu innych dziedzinach inżynierii i nauki.

2.2 Zastosowanie FFT do odszumiania sygnału

W ramach tego ćwiczenia naszym zadaniem jest zastosowanie FFT do odszumienia sygnału periodycznego. Proces ten składa się z kilku kroków:

1. Generowanie zaszumionego sygnału:
   - Sygnał okresowy bez szumu jest opisany równaniem:
     $$y_0(i) = \sin(\omega \cdot i) + \sin(2\omega \cdot i) + \sin(3\omega \cdot i), \tag{1}$$
     gdzie $i = 0, 1, \ldots, N-1$ jest numerem próbki sygnału, a $\omega = \frac{4\pi}{N}$ jest częstotliwością.
   - Szum jest generowany jako zmienna losowa o rozkładzie równomiernym w przedziale $(-1, 1]$:
     $$\Delta = 2\left(\frac{\mathrm{rand}()}{\mathrm{RAND\_MAX} + 1.0}\right) - 1. \tag{2}$$
   - Zaszumiony sygnał jest sumą sygnału okresowego i szumu:
     $$y(i) = y_0(i) + \Delta. \tag{3}$$
2. Wyznaczanie transformaty sygnału przy użyciu FFT:
   - FFT sygnału liczymy za pomocą procedury `gsl_fft_complex_radix2_forward`.
   - Odwrotną transformację wykonujemy za pomocą `gsl_fft_complex_radix2_backward`.
3. Dyskryminacja współczynników FFT:

- Wartości współczynników transformaty są wyzerowywane, jeśli ich moduł jest mniejszy niż połowa maksymalnego modułu:

$$\text{threshold} = \frac{\max |c_k|}{2}. \tag{4}$$

4. Wyznaczanie przefiltrowanego sygnału:

   - Po wykonaniu odwrotnej FFT, przefiltrowany sygnał jest normalizowany przez podzielenie przez N.

Do generowania sygnału zastosowano wyrażenie:

$$y_0(i) = \sin(\omega \cdot i) + \sin(2\omega \cdot i) + \sin(3\omega \cdot i),$$

gdzie $\omega = \frac{4\pi}{N}$. Sygnał ten został zaszumiony przez dodanie pseudolosowej wartości Δ dla każdej próbki.

2.3 Dyskryminacja

Dyskryminacja została przeprowadzona przez wyzerowanie tych współczynników, których moduł był mniejszy niż połowa maksymalnego modułu. Po dyskryminacji, sygnał został poddany odwrotnej FFT, a wynik został unormowany przez podzielenie przez N.

3 Deklaracja zmiennych oraz implementacja bibliotek

Opisane zagadnienie rozwiązano za pomocą programu Visual Studio Code w języku C++. Wykorzstane zostały biblioteki:

- iostream - standardowa biblioteka C++
- cstdlib - użycie rand() oraz RAND_MAX
- cmath - funkcje pow oraz sin
- vector - użycie wektorów do zapisywania danych
- fstream - zapis do plików tekstowych

Wykorzystana została tak że zewnętrzna biblioteka GNU zawierająca wbudowane funkcje do obliczania transformaty.

4 Opis działania

4.1 Krótki opis programu

Program składa się głównie z funkcji main, oraz dwóch funkcji zewnętrznych, mających na celu uproszczenie kodu.

4.1.1 sygnal_okresowy

Funkcja odpowiada za wyznaczenie wartosci sygnalu okresowego dla danych wartosici i oraz w.

```
double sygnal_okresowy(int i, double w){
    return sin(w * i) + sin(2 * w * i) + sin(3 * w * i);
}
```

Fragment kodu 1: Wyznaczenie wartosci sygnalu okresowego

4.1.2 modul

Funkcja odpowiada za wyznaczenie wyznaczenie wartosci modulu liczby zespolonej, dla danej wartosci rzeczywistej oraz urojonej

```
double modul(double rzecz, double uroj){
    return sqrt(pow(rzecz,2) + pow(uroj,2));
}
```

Fragment kodu 2: Wyznaczenie modulu liczby zespolonej

4.1.3 Główna funkcja programu

Po zainicjalizowaniu stałych wyznaczane są oraz zapisywane do pliku tekstowego wartości niezaburzone funkcji.

```
    ofstream y0plik("y0.txt");
    for(int i = 0; i < N; i++){
        y0.push_back(sygnal_okresowy(i,w));
        y0plik << y0[i] << " , ";
    }
    y0plik.close();
```

Fragment kodu 3: Wyznaczenie wartosci niezaburzonych

Następnie z pomocą funkcji tworzącej liczby pseudolosowe ustawiane są wartości zaburzone funkcji.

```
    ofstream yplik("y.txt");
    for(int i = 0; i < N;i++){
        DELTA = 2 * (rand() / (RAND_MAX + 1.0)) - 1;
        y.push_back(y0[i] + DELTA);
        yplik << y[i] << " , ";
    }
    yplik.close();
```

Fragment kodu 4: Wyznaczenie wartosci zaburzonych

Na podstawie tablicy zawierającej wartości zaburzone funkcji, uzupełniania jest tablica dane, odpowiedzialna za przechowywanie wartości rzeczywistych oraz urojonych funkcji zespolonej.

```
    ofstream daneplik("dane.txt");
    for(int i = 0; i < 2*N; i++){
        if(i % 2 == 0){
```

```
            dane[i] = y[i/2];
        }else {
            dane[i] = 0;
        }
        daneplik << dane[i] << " , ";
    }
    daneplik.close();
```

Fragment kodu 5: Uzupelnienie wartosic liczb zespolonych do tablicy dane

Z użyciem biblioteki GSL obliczana jest transformata sygnału. Rezultaty nadpisują tablice dane. Następnie wartości rzeczywiste oraz urojone zapisywane są do plików tekstowych.

```
    gsl_fft_complex_radix2_forward(dane, stride, N);

    ofstream FFTplikrzecz("fftrzecz.txt");
    ofstream FFTplikuroj("ffturoj.txt");

    for(int i = 0; i < 2 * N; i++){
        if(i % 2  == 0){
            FFTplikrzecz << dane[i] << ",";
        }else {
            FFTplikuroj << dane[i] << ",";
        }
    }
    FFTplikrzecz.close();
    FFTplikuroj.close();
```

Fragment kodu 6: Transformata sygnalu

Po wyznaczeniu transformaty, wyznaczane są moduły liczb zespolonych. Największy zostaje zapisany do zmiennej max_modul, z pomocą której wyznaczony zostanie threshold. Jego zadaniem jest wskazanie, ponizej jakich wartosci, wyznaczona czesc funkcji ma zostac wyzerowana.

```
double max_modul = 0.0;
    ofstream modulyTAB("moduly.txt");
    for(int i = 0; i < N;i++){
        moduly[i] = modul(dane[2*i],dane[2*i+1]);
        modulyTAB << moduly[i] <<",";
        if (moduly[i] > max_modul) {
            max_modul = moduly[i];
        }
    }
    modulyTAB.close();
    double threshold = max_modul / 2.0;
    for (int i = 0; i < N; i++) {
        if (moduly[i] < threshold) {
            dane[2 * i] = 0;
            dane[2 * i + 1] = 0;
        }
    }
```

Fragment kodu 7: Wyznaczenie modulow oraz wyzerowanie odpowiednich danych

Po następujących zmianach wyznaczana jest transformata odwrotna oraz normalizacja zmiennych, rezultaty zostają zapisane do pliku tekstowego.

```
    for (int i = 0; i < 2 * N; i++) {
        dane[i] /= N;
    }

    ofstream splotplik("splot.txt");
    for (int i = 0; i < N; i++) {
        splotplik << dane[2 * i] << " , ";
    }
    splotplik.close();
```

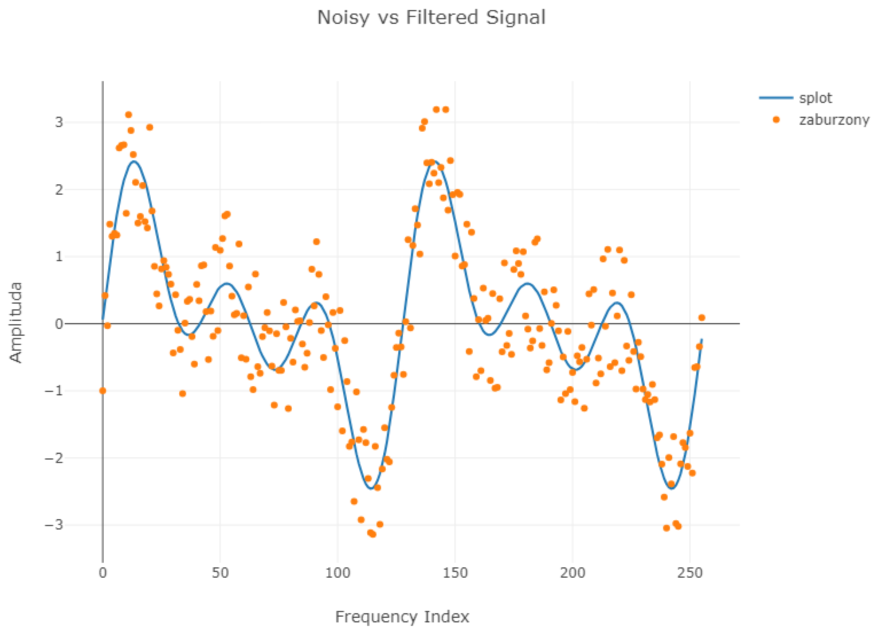
Fragment kodu 8: Transformata odwrotna

5 Analiza wyników

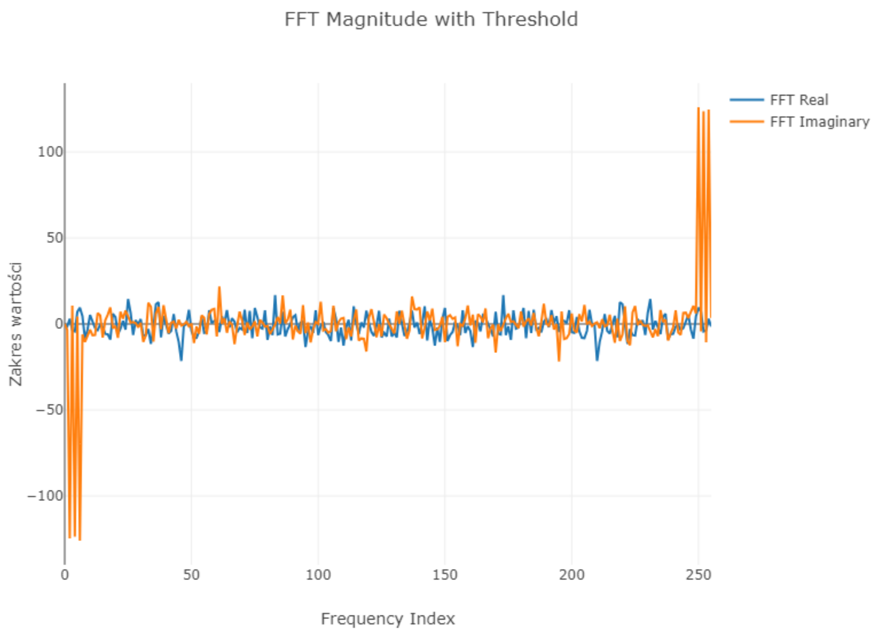
Badanie zostało przeprowadzone dla kolejno $k = 8, 10, 12$. W celu lepszej analizy, dla każdej wartości k.Otrzymane wyniki obejmują wykresy:

- Rzeczywistej i urojonej części FFT.
- Modułów współczynników FFT z progiem dyskryminacji.
- Sygnału zaburzonego i odszumionego.
- Sygnału niezaburzonego i odszumionego.

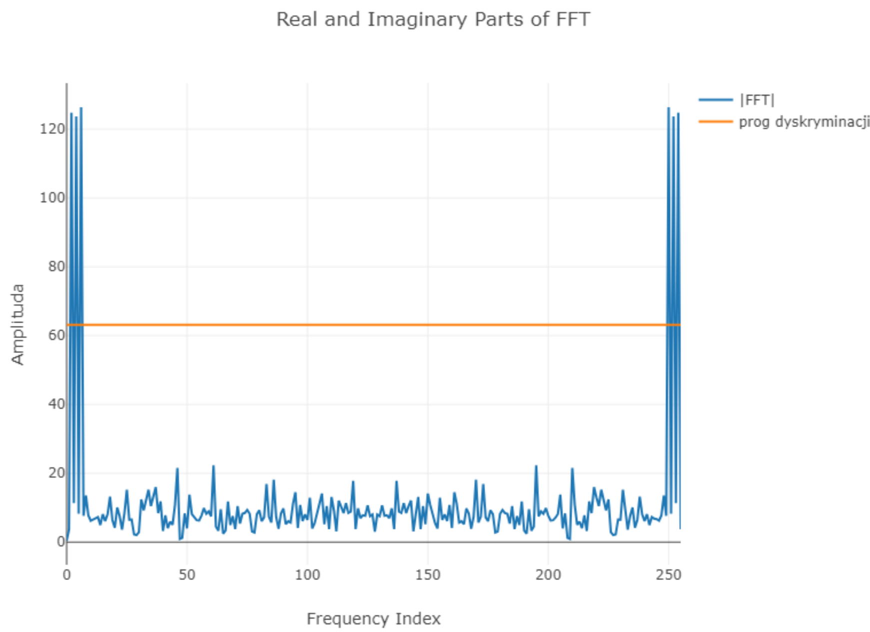
5.1 Analiza wyników dla k = 8



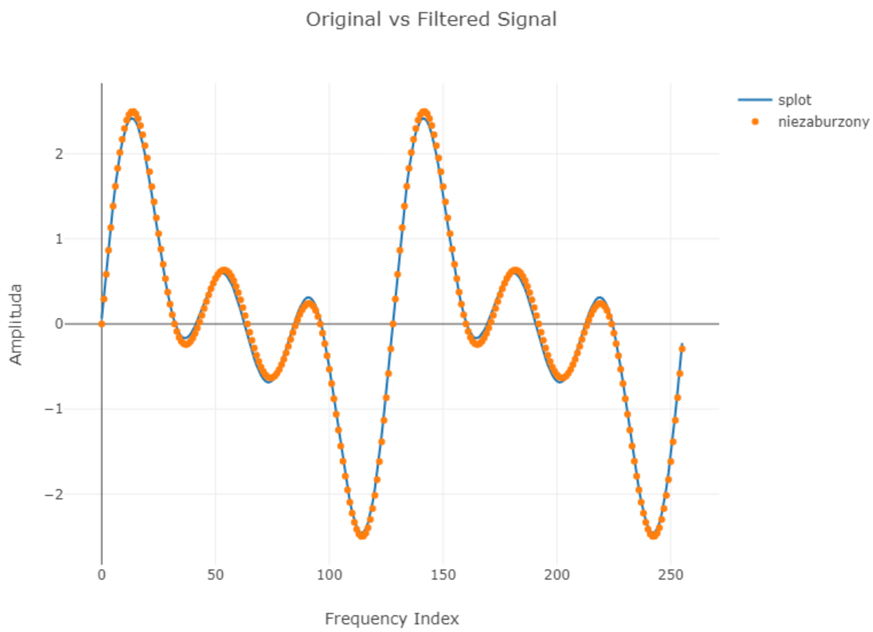
Rysunek 1: Wykres przedstawiający wartości sygnału zaburzonego i odszumionego dla $k = 8$



Rysunek 2: Wykres przedstawiający wartości rzeczywistej i urojonej części FFT dla $k = 8$

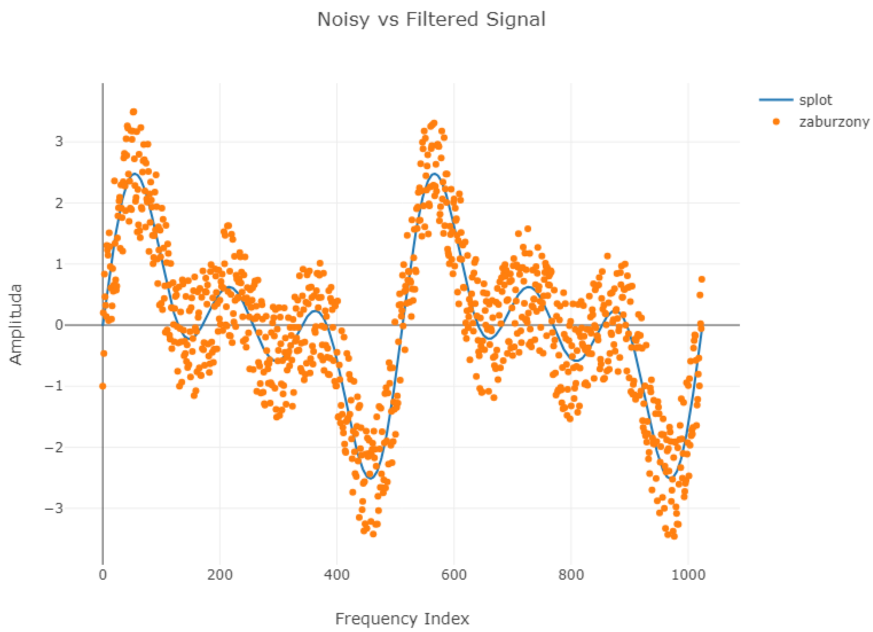


Rysunek 3: Wykres przedstawiający wartości modułów współczynników FFT z progiem dyskryminacji dla $k = 8$

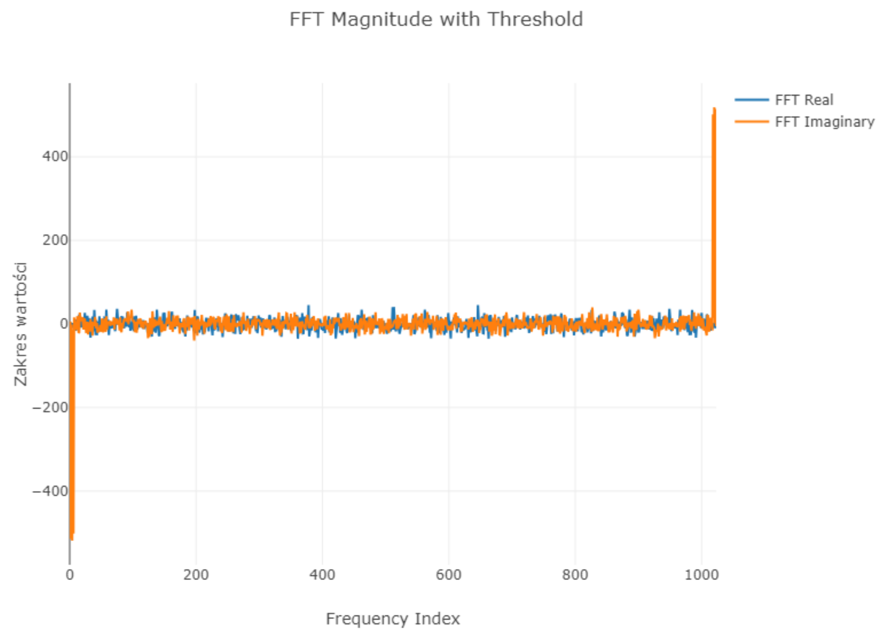


Rysunek 4: Wykres przedstawiający wartości sygnału niezaburzonego i odszumionego dla $k = 8$

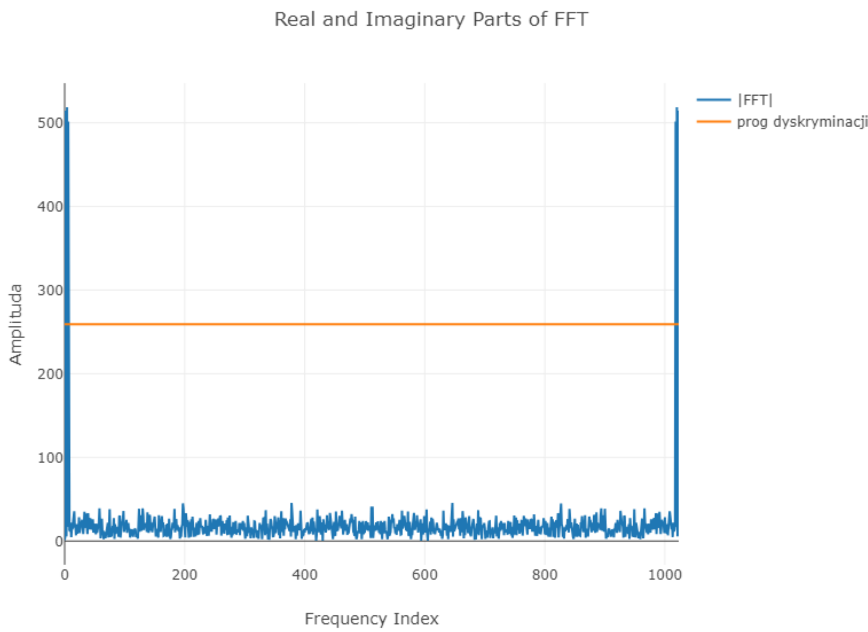
5.2 Analiza wyników dla k = 10



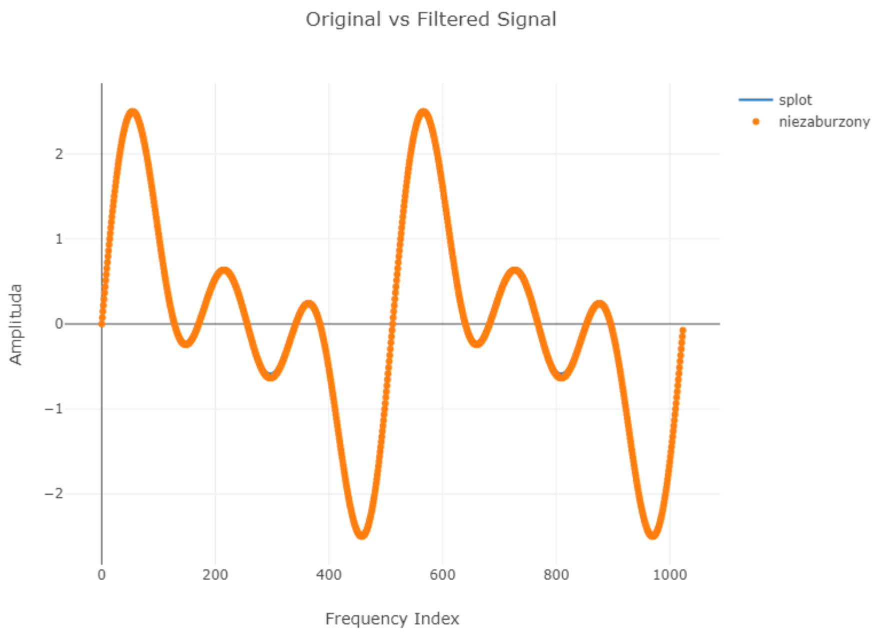
Rysunek 5: Wykres przedstawiający wartości sygnału zaburzonego i odszumionego dla $k = 10$



Rysunek 6: Wykres przedstawiający wartości rzeczywistej i urojonej części FFT dla $k = 10$

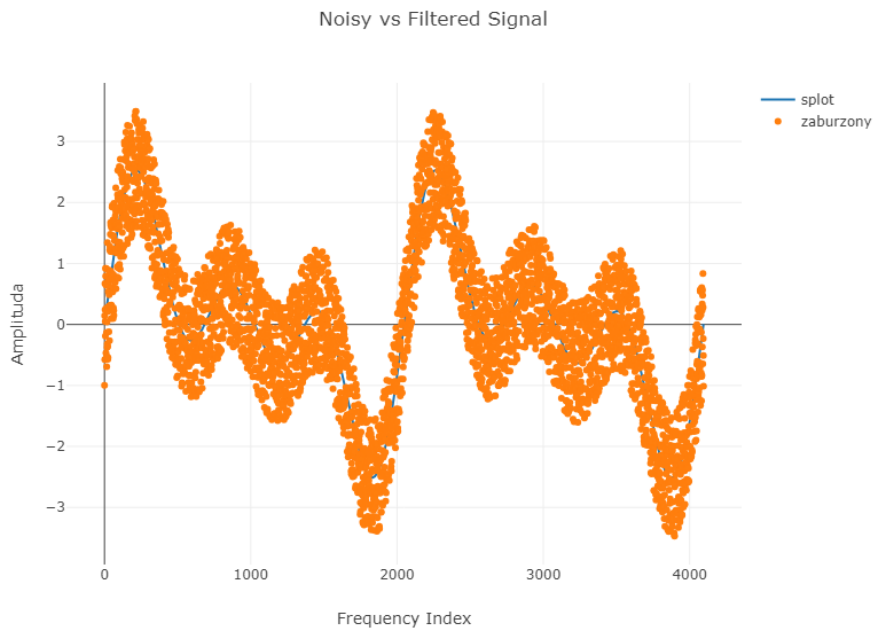


Rysunek 7: Wykres przedstawiający wartości modułów współczynników FFT z progiem dyskryminacji dla $k = 10$

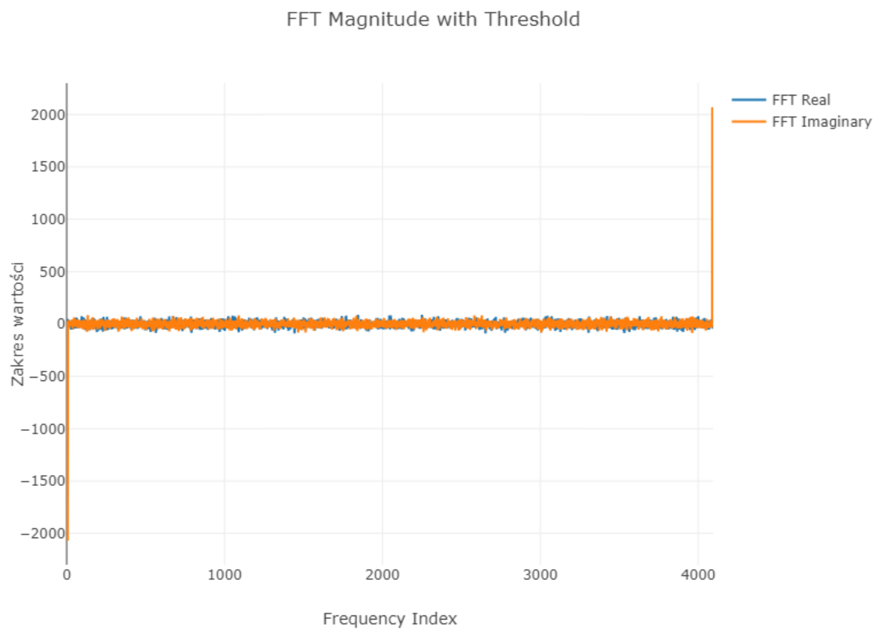


Rysunek 8: Wykres przedstawiający wartości sygnału niezaburzonego i odszumionego dla $k = 10$

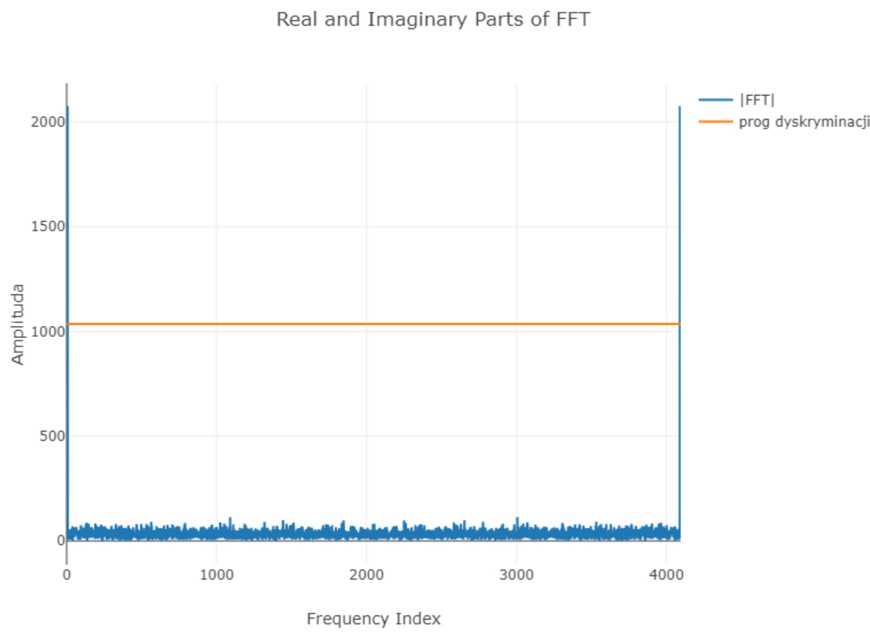
5.3 Analiza wyników dla k = 12



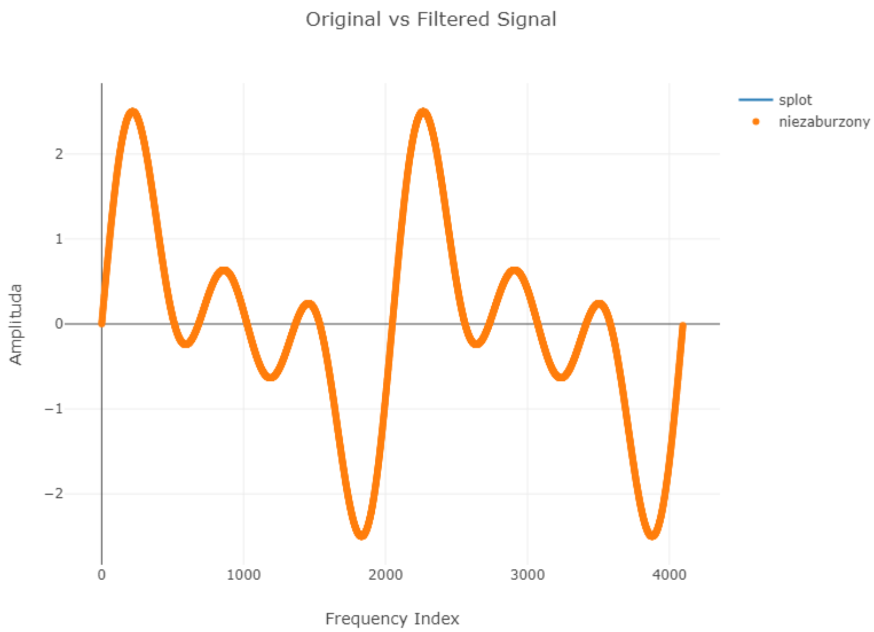
Rysunek 9: Wykres przedstawiający wartości sygnału zaburzonego i odszumionego dla $k = 12$



Rysunek 10: Wykres przedstawiający wartości rzeczywistej i urojonej części FFT dla $k = 12$



Rysunek 11: Wykres przedstawiający wartości modułów współczynników FFT z progiem dyskryminacji dla $k = 12$



Rysunek 12: Wykres przedstawiający wartości sygnału niezaburzonego i odszumionego dla $k = 12$

6 Wnioski