

Data: 17.02.17

Marek Gadzalski 191422

Grzegorz Głąb 191425

Zadanie nr 4

Transformata Fouriera

1. Cel zadania

Celem zadania jest rozszerzenie aplikacji przygotowanej w poprzednich zadaniach o implementację szybkiej transformaty Fouriera.

2. Wstęp

2.1. Definicja transformaty Fouriera

Transformata Fouriera jest to przekształcenie funkcji w czasie (sygnału) na funkcję w dziedzinie częstotliwości, innymi słowy jest to rozbicie funkcji okresowej na jej składowe

funkcje sinus i cosinus. Ze względu na to, że komputery operują na zbiorach dyskretnych najczęściej stosowana jest dyskretna transformata Fouriera, którą opisują wzory:

- transformata Fouriera

$$X_m = \sum_{n=0}^{N-1} x_n \cdot \left(\cos\left(-2\pi m \frac{n}{N}\right) + i \sin\left(-2\pi m \frac{n}{N}\right) \right), k \in R$$

- odwrotna transformata Fouriera

$$x_n = \sum_{m=0}^{N-1} X_m \cdot \left(\cos\left(2\pi m \frac{n}{N}\right) + i \sin\left(2\pi m \frac{n}{N}\right) \right), k \in R$$

gdzie:

x_k – wartość k-tej próbki oryginalnego sygnału

X_k – wartość k-tej próbki transformaty

N – liczba próbek

n – dana próbka

k – częstotliwość próbkowania

Powyższe wzory wskazują, że transformata Fouriera operuje na liczbach zespolonych, gdzie amplituda to: $A = \sqrt{Re^2 + Im^2}/N$

przesunięcie fazy $\theta = \arctg\left(\frac{Im}{Re}\right)$

Transformata Fouriera (oraz jej odwrotność) obliczana jest z definicji wg poniższych wzorów:

$$X(m) = \frac{1}{N} \sum_{n=0}^{N-1} x_n \exp\left(-\frac{2i\pi mn}{N}\right)$$

$$x(n) = \sum_{m=0}^{N-1} X_m \exp\left(\frac{2i\pi mn}{N}\right)$$

Wyrażenie $\exp\left(\pm \frac{2i\pi mn}{N}\right)$ nazywa się jądrem funkcji, a $\exp\left(\frac{2i\pi}{N}\right)$ zastępowany jest symbolem (np. W), przez co wzory przybierają bardziej czytelną formę:

$$X(m) = \frac{1}{N} \sum_{n=0}^{N-1} x_n W^{-nm} \text{ oraz } x(n) = \frac{1}{N} \sum_{m=0}^{N-1} X_m W^{nm}$$

2.2. Szybka transformata Fouriera

Obliczanie transformaty Fouriera z definicji charakteryzuje się złożonością obliczeniową $O(N^2)$ co sprawia, że nie nadaje się ona do praktycznych zastosowań.

Transformatę Fouriera można rozbić na dwie składowe: parzyste i nieparzyste

$$X_m = \sum_{n=0}^{\frac{N}{2}-1} x_{2m} \exp\left(-\frac{2i\pi}{N} 2mn\right) + \sum_{n=0}^{\frac{N}{2}-1} x_{2m+1} \exp\left(-\frac{2i\pi}{N} (2m+1)n\right)$$

$$X_m = \sum_{n=0}^{\frac{N}{2}-1} x_{2m} \exp\left(-\frac{2i\pi}{N/2} mn\right) + \exp\left(-\frac{2i\pi}{N} m\right) \sum_{n=0}^{\frac{N}{2}-1} x_{2m+1} \exp\left(-\frac{2i\pi}{N/2} mn\right)$$

- część parzysta

$$E_m = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \exp\left(-\frac{2i\pi}{N/2} mn\right)$$

- część nieparzysta

$$O_m = \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \exp\left(-\frac{2i\pi}{N/2} mn\right)$$

zatem

$$X_m = E_m + \exp\left(-\frac{2i\pi}{N} m\right) O_m$$

Co więcej, ze względu na okresowość transformaty Fouriera:

$$E_{m+N/2} = E_m \text{ oraz } O_{m+N/2} = O_m$$

Pozwala to na zapisanie transformaty w postaci:

$$X_m = \begin{cases} E_m + \exp\left(-\frac{2i\pi}{N} m\right) O_m, & \text{dla } m \in (0, \frac{N}{2}) \\ E_{m-N/2} + \exp\left(-\frac{2i\pi}{N} m\right) O_{m-N/2}, & \text{dla } m \in (\frac{N}{2}, N) \end{cases}$$

A biorąc pod uwagę, że

$$\exp\left(-\frac{2i\pi}{N} \left(m + \frac{N}{2}\right)\right) = -\exp\left(-\frac{2i\pi}{N} m\right)$$

Ostatecznie transformatę Fouriera można uprościć do postaci:

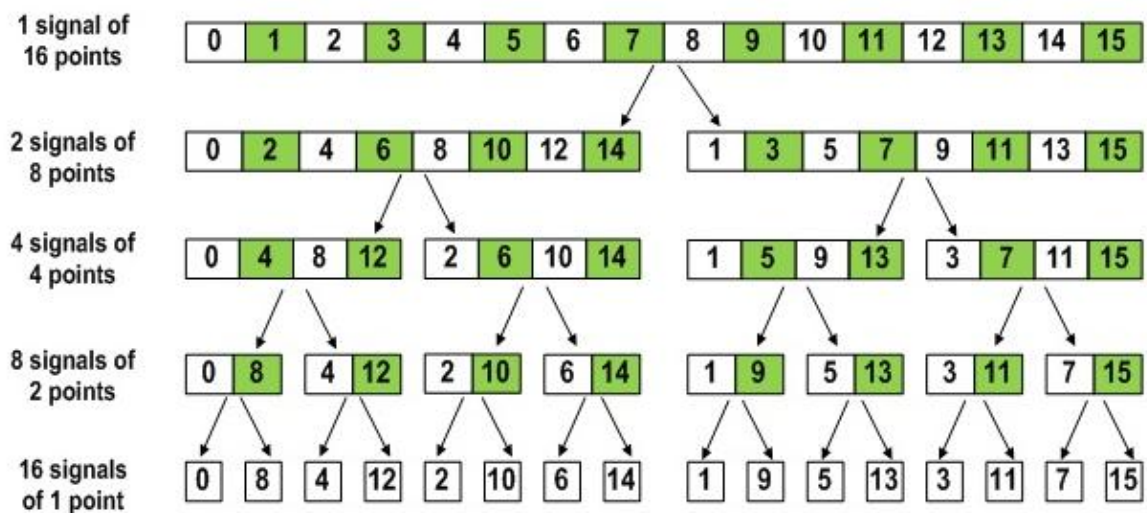
$$X_m = E_m + \exp\left(-\frac{2i\pi}{N} m\right) O_m$$

$$X_{m+\frac{N}{2}} = E_m - \exp\left(-\frac{2i\pi}{N} m\right) O_m$$

Wzór ten można wykorzystać rekurencyjnie (przy założeniu, że $n=2^x$), aż do osiągnięcia $N=1$, uzyskując przy tym złożoność $N\log(N)$.

Zastosowanie rekurencji powoduje zwiększone zapotrzebowanie na pamięć (na każdym poziomie rekurencji tworzone są nowe tablice z danymi). Powyższy algorytm można usprawnić stosując wykorzystując oryginalną tablicę z danymi i jedynie zamieniać miejscami wartości w trakcie obliczeń.

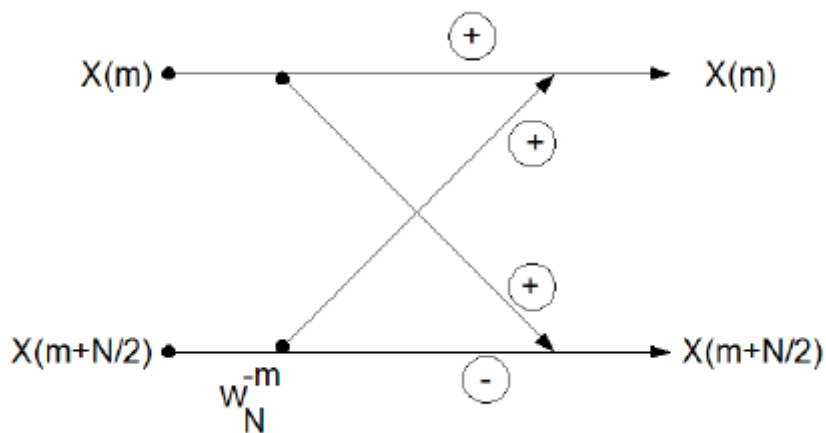
Pierwszym etapem jest dekompozycja (decymacja) w czasie (rysunek 1). W każdym kroku dekompozycji próbki parzyste i nieparzyste są rozdzielane. Etap ten pozwala na odwzorowanie działu rekurencyjnej wersji FFT. Decymacja odbywa się poprzez odwrócenie bitów (bit-reversal) w indeksach tablicy z danymi.



Rysunek 1 Decymacja w czasie 16-to elementowej tablicy z danymi

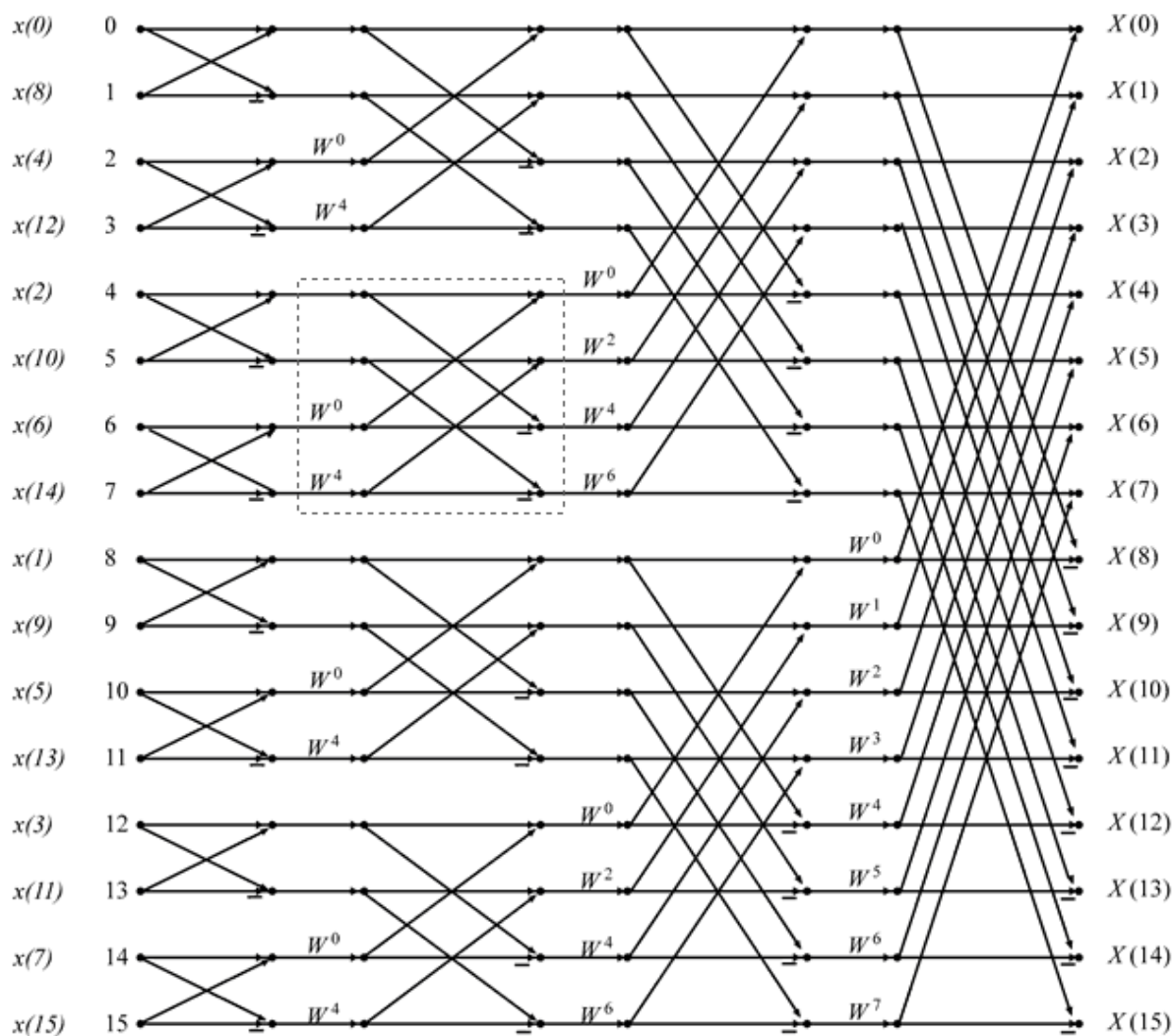
Kolejnym etapem jest obliczanie transformat poszczególnych części tablicy danych. Obliczenia wykonywane są w odwrotnej kolejności niż następowała dekompozycja danych (rysunek 1). Początkowo dane traktowane są jako jednoelementowe tablice (transformata jednej próbki jest równa co do wartości tej próbki), następnie dane składane są do coraz większych zbiorów.

Podstawową jednostką obliczeniową jest tzw. motylek



Rysunek 2 Motylek szybkiej transformaty Fouriera

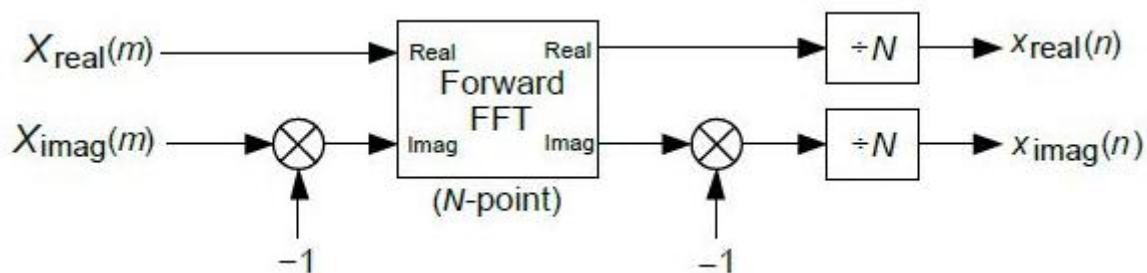
Na każdym etapie motylki obliczane są w obrębie określonego zbioru danych a całość obliczeń na 16 elementowej tablicy przedstawiona jest na rysunku 3



Rysunek 3 Szybka transformata Fouriera

2.3. Odwrotna transformata Fouriera

Odwrotną transformatę Fouriera można obliczyć korzystając z transformaty wprost. Jednym ze sposobów jest wykorzystanie sprzężenia zespolonego. Próbkę poddawane są sprzężeniu zespolonemu przed i po transformacji.



Rysunek 4 Wykorzystanie sprzężenia zespolonego do obliczania odwrotnej transformaty Fouriera

3. Opis implementacji

Do aplikacji dodany został pakiet **singals.fourier**, który zawiera wszystkie klasy potrzebne do implementacji transformaty Fouriera w przygotowanej aplikacji. Implementacje obejmuje transformatę Fouriera z definicji (**DefinitionFourierTransform.java**) oraz szybką transformatę Fouriera (**FastFourierTransform.java**) która zawiera dwa warianty:

- rekurencyjną
- *in situ*

Aplikacja korzysta wyłącznie z transformaty *in situ*. Odwrotna transformata obliczana jest przy użyciu sprzężenia zespolonego.

Transformata obliczana jest dla każdego sygnału dyskretnego i zapisywana jest w postaci tablicy liczb zespolonych jako pole klasy. Jeśli liczba próbek sygnału nie jest potęgą liczby 2, sygnał jest rozszerzany do następnej potęgi liczby 2 i wypełniany zerami. Okno wyświetlające sygnały zostało poszerzone o nową zakładkę w której wyświetlane są dwa wykresy transformaty Fouriera: amplituda i przesunięcie w fazie. Dodatkowo każdy sygnał dyskretny można odtworzyć na podstawie jego transformaty.

Do wszystkich implementacji transformaty Fouriera zostały napisane testy jednostkowe. Wyniki obliczeń były porównywane z gotową implementacją szybkiej transformaty Fouriera (rekurencyjną) (<http://introcs.cs.princeton.edu/java/97data/FFT.java.html>). Dodatkowo w pakiecie **singals.fourier** znajduje się osobna aplikacja konsolowa, porównująca szybkość działania przygotowanych implementacji fft.

4. Instrukcja Obsługi

Program składa się z okna głównego i okien wykresów. W tym rozdziale opisano instrukcję obsługi z podziałem na przypadki użycia.

Sygnaly sinusoidalne

Sygnaly prostokątne

Szumy

Radar

Sygnal sinusoidalny

Amplituda: 1

Częstotliwość [Hz]: 10

Przesunięcie fazy [n]: 0

Czas trwania[s]: 1

Czas początkowy: 0

Generuj Sygnal Pseudociągły

Wygenerowane sygnały pseudociągłe

id	typ sygnału	czas początkowy [s]	czas trwania [s]	amplituda	okres [s]	wypełnienie ...
Wyczyść						

Filtry

Okno: Hamming'a

Rząd Filtru [ilość próbek]: 1024

Filtr: Dolnoprzepustowy

Częstotliwość próbkowania[Hz]: 1024

Częstotliwość odcięcia niskiego [Hz]: 100

Częstotliwość odcięcia wysokiego [Hz]: 500

Utwórz Filtr

Próbkowanie Sygnałów

Częstotliwość próbkowania[Hz]: 1024

Typ kwantyzacji: Brak kwantyzacji

Typ konwersji: zero-order hold

Liczba bitów kwantyzacji: 8

Próbkuj sygnał

Konwertuj

id	typ sygnału	czas początkowy [s]	czas trwania [s]	amplituda	okres [s]
Wyczyść					

Operacje na sygnałach

Transformaty

+

SPLIT

KORELACJA

Odwróć kolejność

Porównaj sygnały

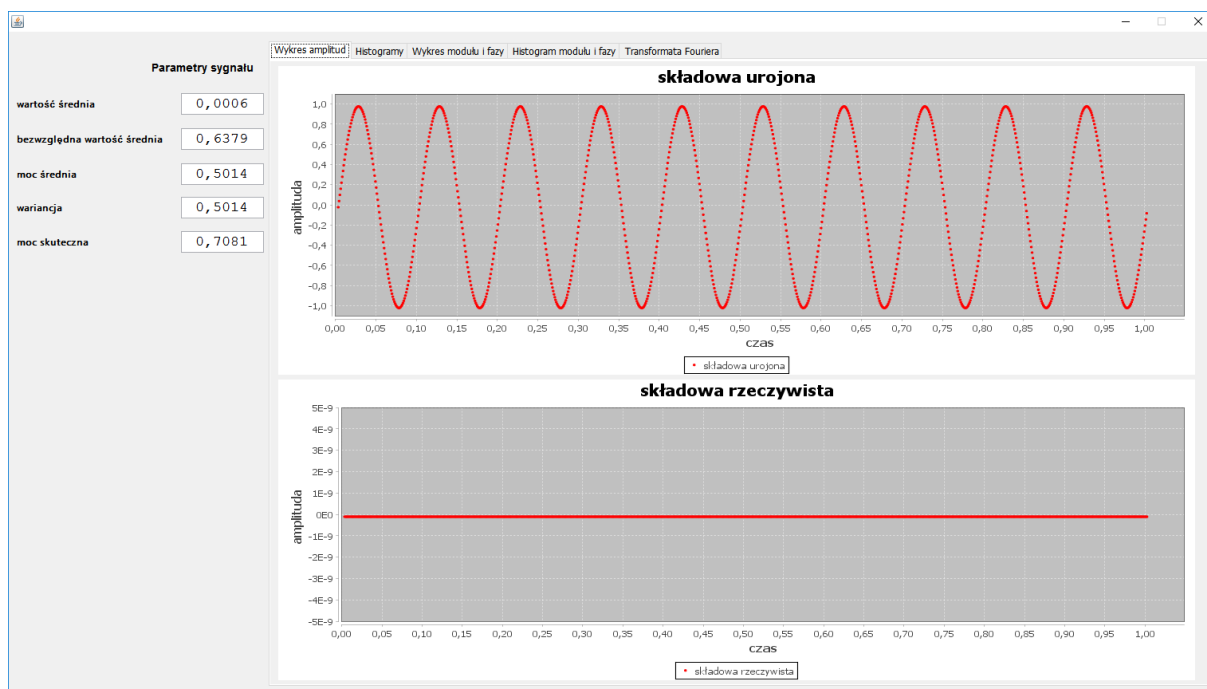
Wyświetl sygnał

Zapisz

Odczytaj

liczba przedziałów histogramu: 20

Rysunek 5: Okno główne




Rysunek 6: Okno sygnału

4.1. Przypadek użycia – generowanie sygnału pseudo ciągłego i spróbkowanego

Scenariusz pokazuje w jaki sposób wygenerować sygnał w programie:

1. Użytkownik wybiera typ sygnału i wprowadza parametry, które będzie miał wygenerowany sygnał pseudo ciągły. Po wybraniu opcji „Generuj Sygnał Presudociągły” wygenerowany sygnał pokazuje się w tabelce po prawej stronie w obszarze o nazwie „Wygenerowane sygnały pseudociągłe”


Sygnały

Sygnały sinusoidalne

Sygnały prostokątne

Szumy

Radar

Sygnał sinusoidalny

▼

Amplituda

1

Częstotliwość [Hz]

10

Przesunięcie fazy [n]

0

Czas trwania[s]

1

Czas początkowy

0

Generuj Sygnał Pseudociągły

Rysunek 7: wybór typu i parametrów sygnału

Wygenerowane sygnały pseudociągłe						
id	typ sygnału	czas początkowy [s]	czas trwania [s]	amplituda	okres [s]	wypełnienie ...
1	sin	0.0	1.0	1.0	0.1	

Rysunek 8: rezultat akcji "Generuj Sygnał Pseudociały"

2. Użytkownik zaznacza w tabelce w obszarze „Wygenerowane sygnały pseudociągłe” wybrany sygnał i następnie w obszarze „Próbkowanie Sygnałów” wybiera parametry kwantyzacji i próbkowania oraz wybiera akcję „Próbkuj Sygnał”. W rezultacie powstaje sygnał spróbkowany, który jest widoczny w tabeli w obszarze „Próbkowanie Sygnałów”.

Próbkowanie Sygnałów

Częstotliwość próbkowania[Hz]

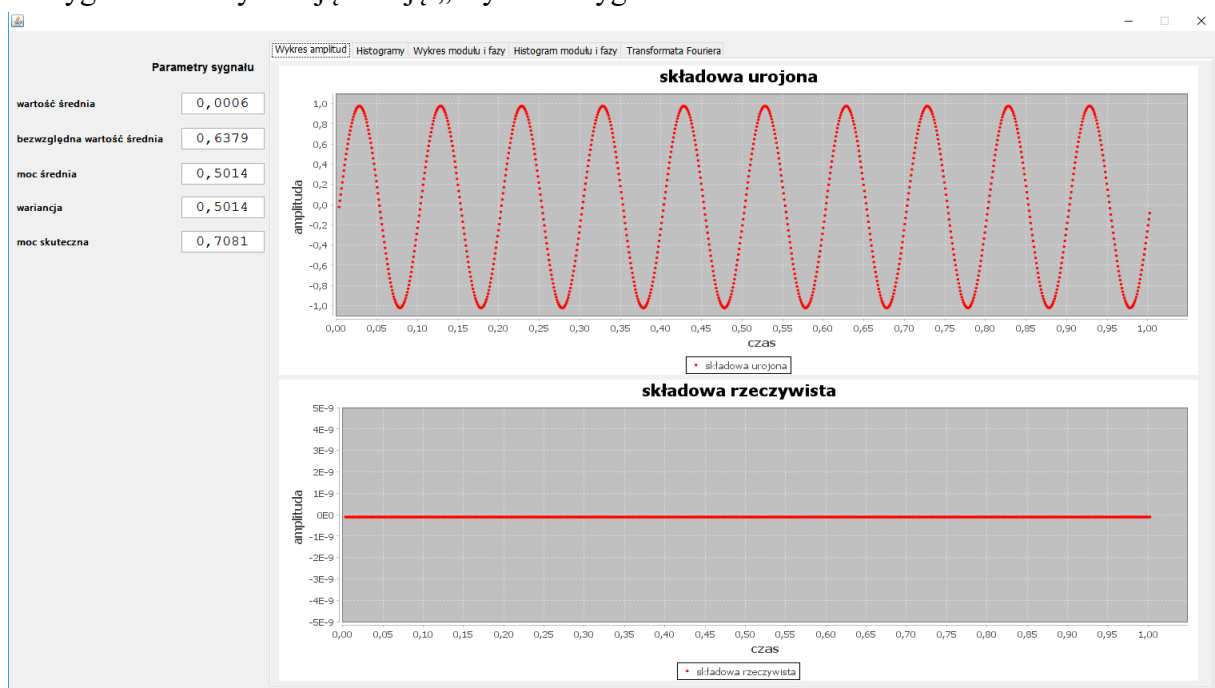
Typ kwantyzacji Typ konwersji

Liczba bitów kwantyzacji

id	typ sygnału	czas początkowy [s]	czas trwania [s]	amplituda	okres [s]
1	sin	0.0	1.0	1.0	0.1

Rysunek 9: Próbkowanie sygnału pseudociągłego

- Użytkownik może wyświetlić sygnał spróbkowany, zaznaczając go w tabeli „Próbkowanie Sygnałów” i wybierając akcję „Wyświetl sygnał”



Rysunek 10: wyświetlanie sygnału (amplituda w czasie)



Rysunek 11: Wyświetlanie sygnału (rezultat FFT w decymacji czasu)

4.2. Przypadek użycia – generowanie sygnału pochodnego z wykorzystaniem operacji na sygnałach

Scenariusz pokazuje jak wykonać przykładową operację na dwóch sygnałach spróbkowanych:

1. Użytkownik wybiera dwa sygnały spróbkowane z tabeli w obszarze „Próbkowanie Sygnałów” i wybiera w obszarze „Operacje na Sygnałach” jedną z dostępnych akcji. W rezultacie powstaje nowy sygnał pochodny, widoczny w tabeli w obszarze „Próbkowanie Sygnałów”.

Próbkowanie Sygnałów

Częstotliwość próbkowania[Hz]

Typ kwantyzacji Typ konwersji

Liczba bitów kwantyzacji

id	typ sygnału	czas początkowy [s]	czas trwania [s]	amplituda	okres [s]
1	sin	0.0	1.0	1.0	0.1
2	sin	0.0	1.0	1.0	0.5
3	sygnał pochodny	0.0	1.0	1.99999...	

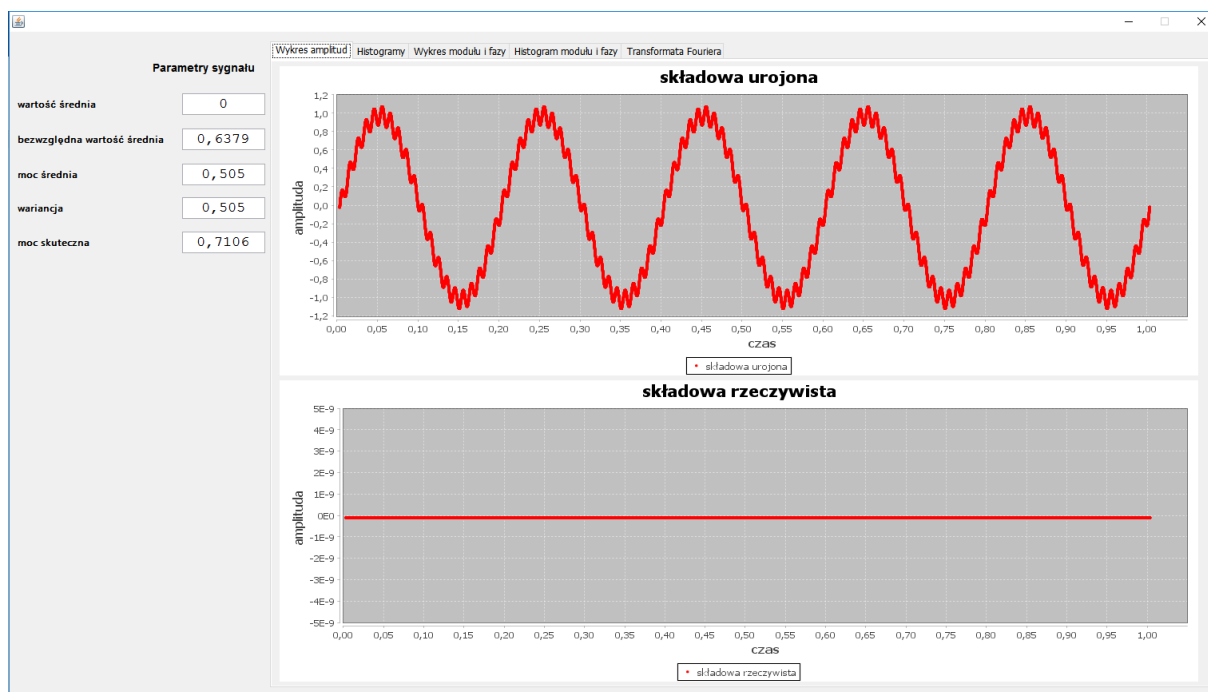
Operacje na sygnałach

☐ Transformaty

☐ Odwróć kolejność

5 10 15 20

Rysunek 12: Operacje na sygnałach



Rysunek 13: Rezultat operacji sumy sygnałów

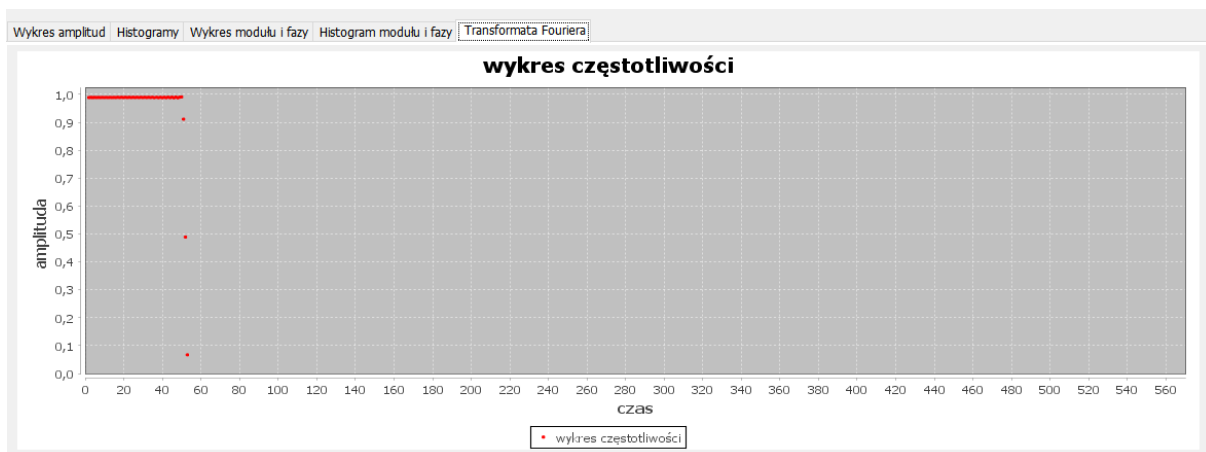
4.3. Przypadek użycia – filtrowanie sygnału

Ten przypadek użycia pokazuje w jaki sposób używać funkcjonalności filtrowania sygnałów. Do wykonania tej operacji potrzebny jest istniejący sygnał spróbkowany lub sygnał pochodny, dostępny w tabeli w sekcji „Próbkowanie Sygnałów”.

1. Użytkownik w obszarze „Filtry” wybiera typ filtru, funkcję okna oraz parametry filtrowania i uruchamia akcję „Utwórz Filtr”. W rezultacie powstaje sygnał filtru, jako sygnał pochodny, widoczny w tabeli w sekcji „Próbkowanie Sygnałów”.

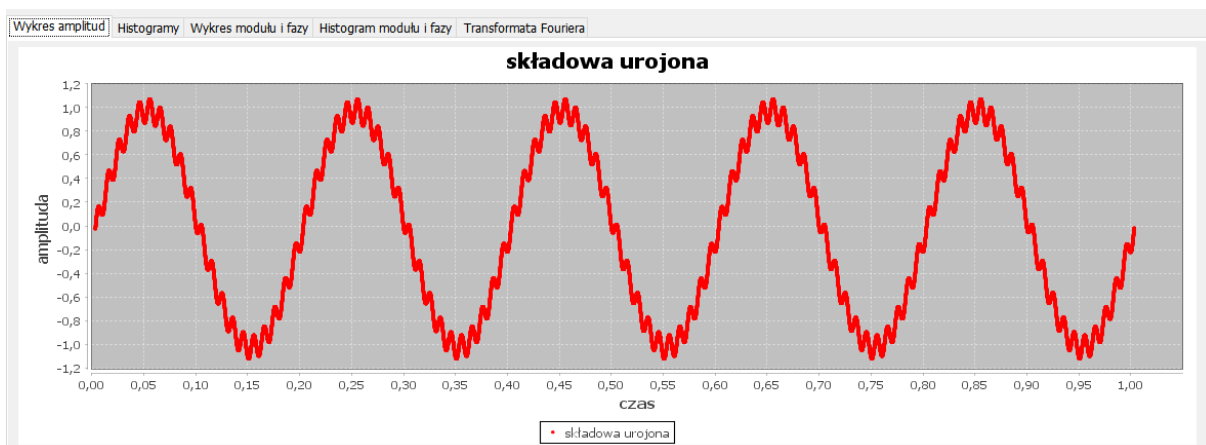


Rysunek 14: Wykres sygnału filtru (amplituda w dziedzinie czasu)

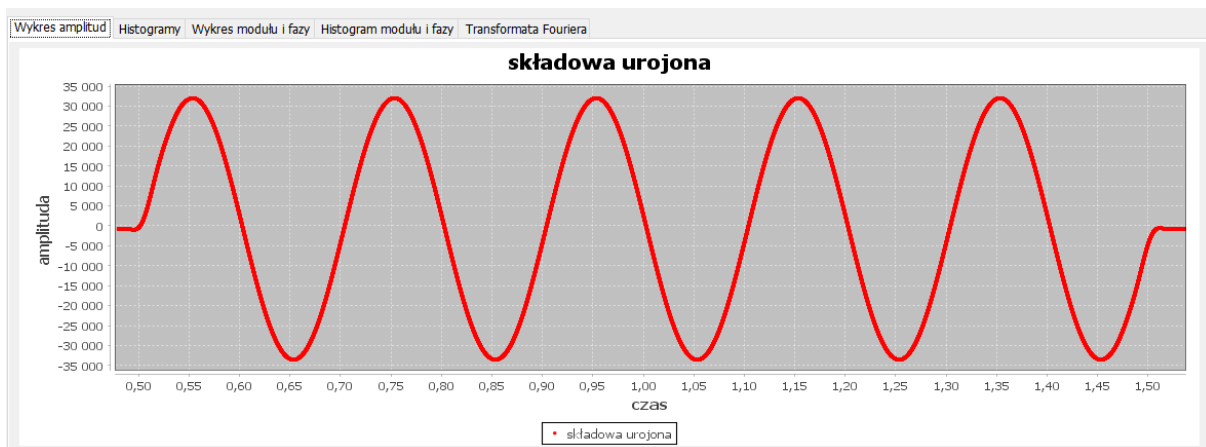


Rysunek 15: Wykres sygnału filtru (FFT - amplituda w dziedzinie częstotliwości)

2. Aby odfiltrować sygnał z użyciem wygenerowanego sygnału filtru, w obszarze „Próbkowanie Sygnałów” użytkownik zaznacza sygnał wybranego filtru i sygnał do odfiltrowania, oraz w sekcji „Operacje na Sygnałach” wybiera akcję „splot”. Powstaje sygnał pochodny, widoczny w tabeli w sekcji „Próbkowanie Sygnałów”, który jest wynikiem filtrowania.



Rysunek 16: Sygnał przed filtrowaniem (amplituda w czasie)



Rysunek 17: Wynik filtrowania (amplituda w czasie)

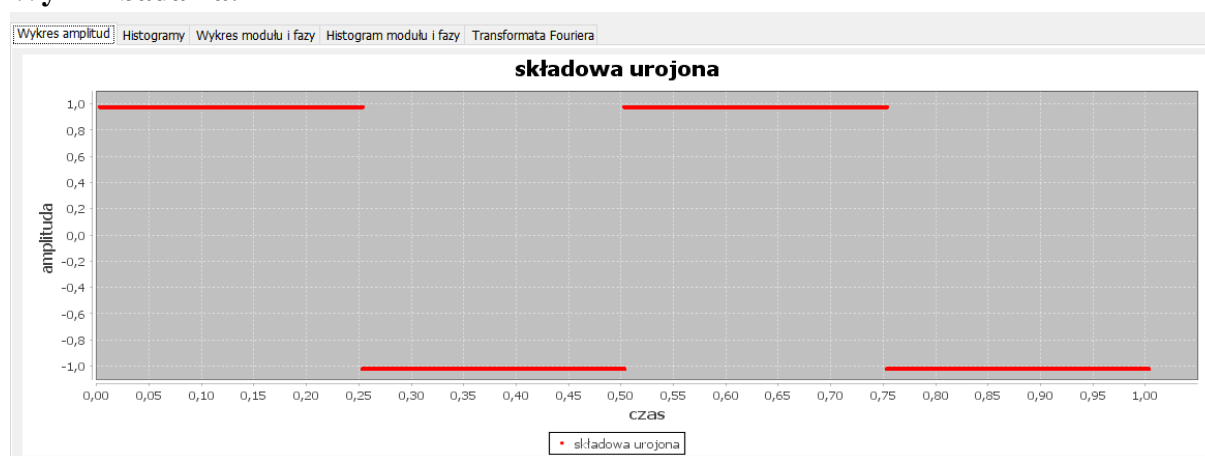
5. Wyniki działania programu

Ten rozdział opisuje przebieg przeprowadzonych badań i ich wyniki przy użyciu implementacji opisywanego programu. Jest to jedynie kilka typowych eksperymentów związanych z przetwarzaniem sygnałów. Opisywany program umożliwia budowanie wielu różnych scenariuszy badań, nie tylko tych zamieszczonych w tym dokumencie.

5.1. Wyniki działania transformaty Fouriera na przykładzie sygnału prostokątnego.

Scenariusz tego badania ma na celu pokazanie w jaki jest wynik zastosowania transformaty Fouriera dla sygnału prostokątnego. Taki sygnał powinien zostać rozłożony na wiele sygnałów sinusoidalnych, począwszy od sygnałów z wysoką amplitudą i niską częstotliwością, oraz wraz ze wzrostem częstotliwości malejącą amplitudą.

Wyniki badania:



Rysunek 18: Sygnał prostokątny (amplituda w dziedzinie czasu)



Rysunek 19: Wynik FFT w decymacji czasu (amplituda w dziedzinie częstotliwości)

Wniosek:

Wyniki badania są całkowicie zgodne z oczekiwaniami.

5.2. Porównanie działania dyskretnej transformaty Fouriera i szybkiej transformaty Fouriera.

Scenariusz tego badania polega na wykonaniu dodatkowego programu testowego, który bez obsługowo i bez użycia graficznego interfejsu użytkownika, używając zaimplementowanych w zadaniu algorytmów wygeneruje sygnał szumu gaussowskiego o ilości próbek 8912 i wykona w sekwencji dyskretną transformatę Fouriera i następnie szybką transformatę Fouriera (FFT w decymacji czasu) – implementację rekurencyjną, oraz finalnie wykona szybką transformatę Fouriera (FFT w decymacji czasu) – implementację *in situ* dla tego samego sygnału szumu.

Wyniki przeprowadzonego badania zaprezentowano poniżej w formie listingu programu:

```
Definition transform
68178[ms]
-----
RecursiveFFt
26[ms]
-----
FFT
26[ms]
-----
```

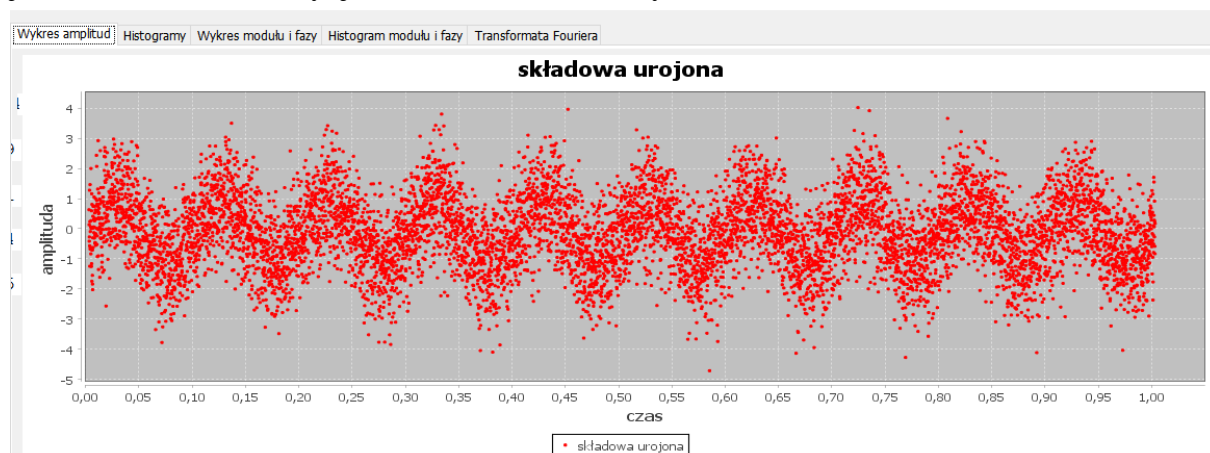
Wniosek:

Algorytm DFT w porównaniu do algorytmu FFT jest tak uderzająco i skandalicznie niewydajny, że jego zastosowanie we współczesnej rzeczywistości jest praktycznie niemożliwe. Jeżeli chodzi o porównanie implementacji algorytmu FFT rekursywnego i implementacji *in situ*, to nie ma praktycznie różnicy w szybkości działania.

5.3. Skuteczność filtrowania i transformaty Fouriera przy odsumianiu sygnałów.

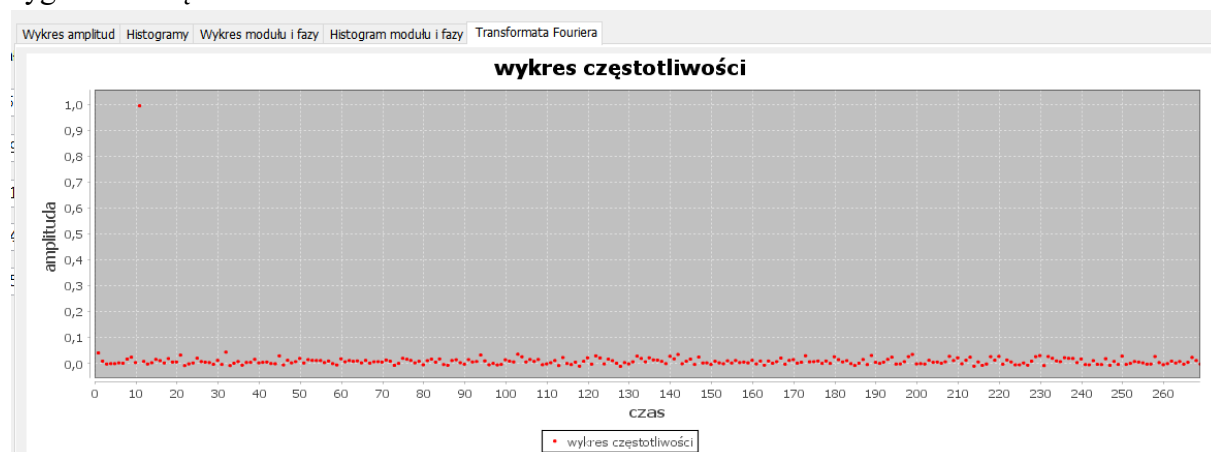
To badanie polega na odfiltrowaniu szumów gaussowskich od sygnału okresowego przy zastosowaniu filtrów, operacji splotu i wizualizacji operacji na podstawie wyniku transformaty Fouriera.

W tym badaniu sygnał wejściowy w dziedzinie czasu na pierwszy rzut oka nie jest jednoznacznie okresowy, jest znacznie zaszumiony.



Rysunek 20: sygnał wejściowy (amplituda w czasie)

Z wykorzystaniem transformaty Fouriera da się zauważyć wyraźna okresowość tego sygnału w częstotliwości 10Hz

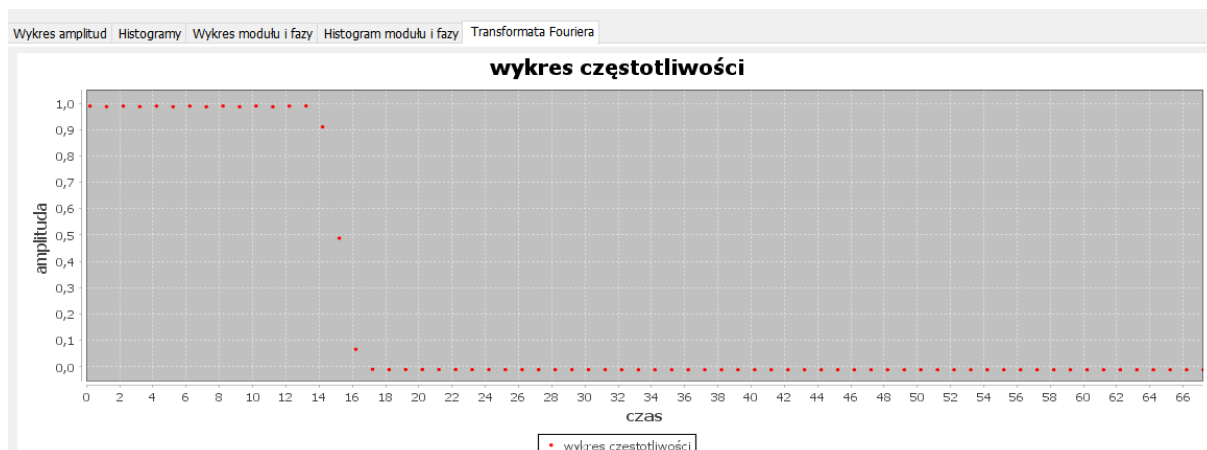


Rysunek 21: Sygnał wejściowy - wynik FFT (amplituda w dziedzinie częstotliwości)

Do wykonania badania generowany jest filtr z użyciem funkcji okna, w tym przypadku wygenerowano filtr dolnoprzepustowy o częstotliwości odcięcia 15Hz z użyciem funkcji okna Hamminga, żeby pozbyć się większości szumów przy zachowaniu stosunkowo niskiego zniekształcenia sygnału okresowego.

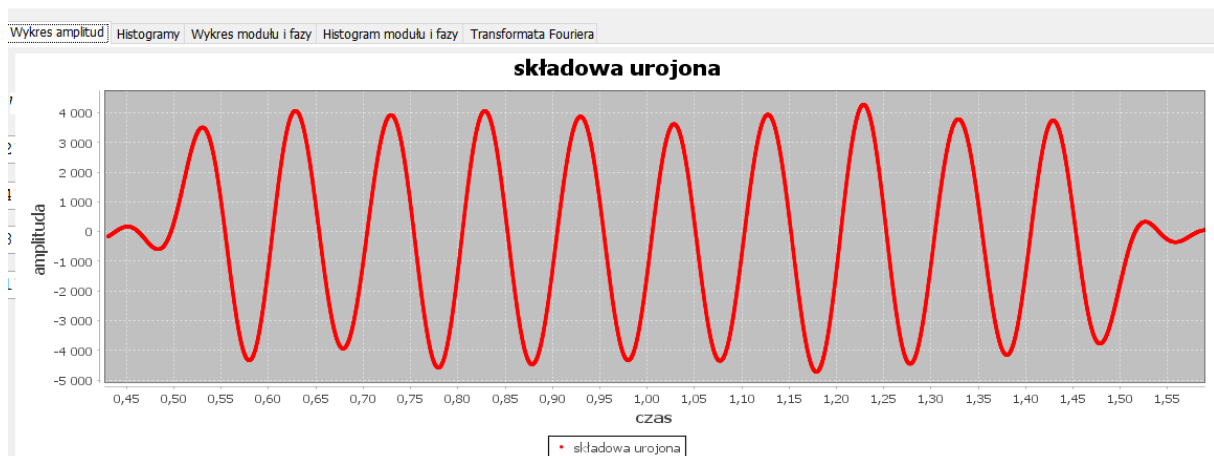


Rysunek 22: Wykres filtru dolnoprzepustowego (amplituda w czasie)

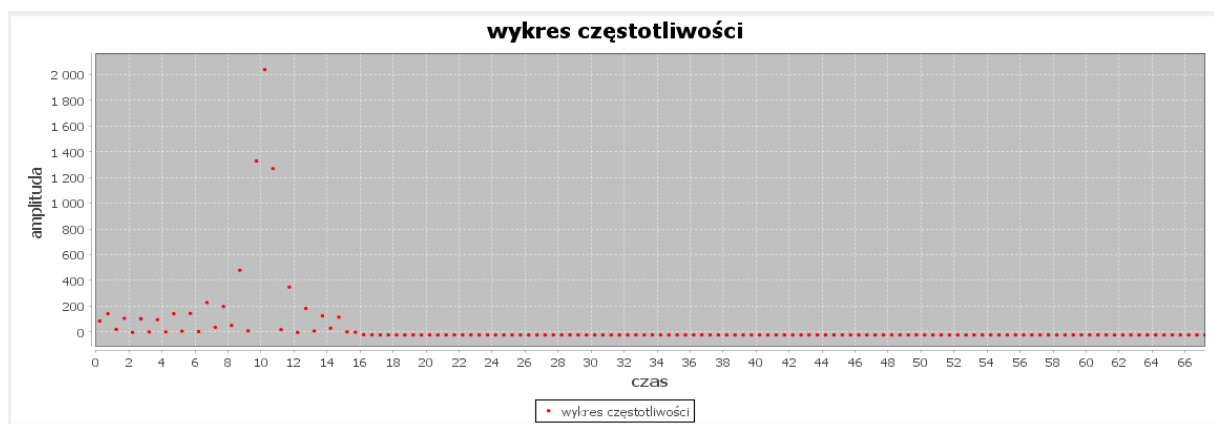


Rysunek 23: Filtr dolnoprzepustowy w dziedzinie częstotliwości

Ostatnim krokiem badania jest wykonanie operacji splotu sygnału z sygnałem filtra co powoduje powstanie sygnału odszumionego.



Rysunek 24: Sygnał odszumiony (amplituda w dziedzinie czasu)



Rysunek 25: Sygnał odszumiony w dziedzinie częstotliwości.

Wniosek:

Transformata Fouriera bardzo skutecznie pozwala zidentyfikować sygnał okresowy i oddzielić go od szumów. Przy wykorzystaniu sygnału filtra i funkcji okna można w dużej części pozbyć się szumów z sygnału. Filtrowanie nie jest jednak idealne i sygnał odszumiony jest często zniekształcony. Należy obrać optymalne parametry, filtr i funkcję okna, żeby sygnał był jak najmniej zniekształcony, nigdy jednak nie będzie idealnie odwzorowany.

6. Wnioski

W tym rozdziale zostały zamieszczone najważniejsze wnioski z przeprowadzonej implementacji i wykonanych badań, opisanych w poprzednich rozdziałach tego dokumentu:

- Dyskretna Transformata Fouriera (DFT) jest stosunkowo łatwa w implementacji, ale jej efektywność działania jest bardzo niska (złożoność kwadratowa) co sprawia, że przy współczesnych sygnałach o wysokiej rozdzielczości jest praktycznie nieprzydatna.

- Szybka Transformata Fouriera (FFT) jest znacznie efektywniejsza w działaniu od DFT. Jest też bardziej skomplikowana w implementacji, ale walor efektywności sprawia, że jest niesamowicie przydatna.
- Implementacja FFT *in situ* jest praktycznie taka sama co do efektywności jak implementacja rekurencyjna. FFT *in situ* jest natomiast efektywniejsza w wykorzystaniu pamięci. Zwłaszcza przy sygnałach o znacznej ilości próbek, implementacja *in situ* jest lepszym wyborem, mimo większej trudności implementacji.
- Transformata Fouriera jest bardzo skutecznym narzędziem do identyfikacji sygnałów okresowych w sygnałach zaszumionych i w sygnałach złożonych z wielu składowych okresowych. Przy zastosowaniu odpowiedniej filtracji można skutecznie oddzielić sygnały od siebie, lecz jest to zawsze obarczone pewnym zniekształceniem. Zniekształcenie zależy natomiast od doboru filtru, funkcji okna i parametrów filtracji.