Zadanie nr 4 - Przekształcenie Fouriera, Walsha-Hadamarda, kosinusowe i falkowe, szybkie algorytmy

Cyfrowe Przetwarzanie Sygnałów

Aneta Wiśniewska, 204029 Hanna Paluszkiewicz, 203962

04.06.2018

1 Cel zadania

Celem ćwiczenia jest zapoznanie się w praktyce z operacjami transformacji sygnałów dyskretnych przy użyciu przykładowych metod.

2 Wstęp teoretyczny

2.1 Teoria

W tej pracy zostały wybrane: górny wykres prezentujący część rzeczywistą amplitudy w funkcji częstotliwosci, a wykres dolny część urojoną. Została zaimplementowana dyskretna transformacja Fouriera, czyli algorytm z definicji oraz szybka transformacja Fouriera z decymacją w dziedzinie czasu (DIT FFT). Została zastosowana transformacja kosinusowa typu drugiego (DCT II) oraz szybka transformacja kosinusowa (FCT II).

Dyskretna transformacja Fouriera to przekształcenie macierzowe w postaci wzoru:

Odwrotne przekształcenie jest opisane wzorem:

$$A_k = \sum_{n=0}^{N-1} a_n w_N^{-kn},\ 0 \leqslant k \leqslant N-1$$

$$a_n = \frac{1}{N} \sum_{k=0}^{N-1} A_k w_N^{kn},\ 0 \leqslant n \leqslant N-1$$

x(n) - to wektor próbek sygnału, gdzie kolejne próbki są oddalone od siebie o czas delta t = 1/fpr (1/częstotliwość próbkowania). X(m) - to wektor-wynik transformacji sygnału x(n). Każda kolejna wartość opisuje udział składowej częstotliwosci m*f0 w sygnale próbkowanym. f0 opisuje wzór:

Istnieje także zapis zastosowaniem jądra transformacji prostej, któ-

$$f_0 = \frac{1}{T} = \frac{1}{\Delta t\, N} = \frac{f_{pr}}{N}$$

re przyjmuje postać: Wtedy zapisujemy:

$$W_N = e^{j\frac{2\pi}{N}}$$

transformacja prosta:

$$X(m)=\frac{1}{N}\sum_{n=0}^{N-1}x(n)W_N^{-mn}$$

transformacja odwrotna: dla:

$$x(n)=\sum_{m=0}^{N-1}X(m)W_N^{mn}$$

$$0\leq m,n\leq N-1$$

Postać macierzowa wzoru przyjmuje postać:

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ \dots \\ X_{N-1} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & \dots & W_N^{-(N-1)} \\ 1 & W_N^{-2} & W_N^{-4} & \dots & W_N^{-2(N-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & W_N^{-(N-1)} & W_N^{-2(N-1)} & \dots & W_N^{-(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_{N-1} \end{bmatrix}$$

Szybka transformacja Fouriera - Jest ulepszonym algorytmem DFT z mniejszym kosztem obliczeniowym. Klasyczna Dyskretna transformata Fouriera wymaga N*N mnożeń liczb zespolonych, natomiast FFT N log2 N

Decymacja w dziedzinie czasu - podział próbek wejściowych, zależnych od czasu, na próbki parzyste i nieparzyste z zapisem do oddziezlbych wektorów.
Równanie transformacji, gdy rozdzielimy próbki parzyste i nieparzyste do osobnych wektorów, zapisujemy :

$$X(m)=\frac{1}{N}\left(\sum_{n=0}^{N/2-1}x(2n)W_N^{-m(2n)}+\sum_{n=0}^{N/2-1}x(2n+1)W_N^{-m(2n+1)}\right)$$

f0 = fp/K gdzie f0 to częstotliwosć odcięcia

fp to częstotliwosć

$$X(m)=\frac{1}{N}\left(\sum_{n=0}^{N/2-1} x(2n)W_N^{-m(2n)}+W_N^{-m}\sum_{n=0}^{N/2-1} x(2n+1)W_N^{-m(2n)}\right)$$

Po wyciągnięciu WN-m przed sumę, równanie przyjmuje postać:

Jedna suma dotyczy próbek parzystych, a drugie nieparzystych.

Ponieważ zachodzi równosć: to poprzedni wzór możemy zapisać jako:

$$W_N^{-m(2n)}=(e^{j\frac{2\pi}{N}})^{-2mn}=e^{j\frac{2\pi}{N}(-2mn)}=e^{j\frac{2\pi}{N/2}(-mn)}=W_{N/2}^{-mn}$$

$$X(m)=\frac{1}{N}\left(\sum_{n=0}^{N/2-1} x(2n)W_{N/2}^{-mn}+W_N^{-m}\sum_{n=0}^{N/2-1} x(2n+1)W_{N/2}^{-mn}\right)$$

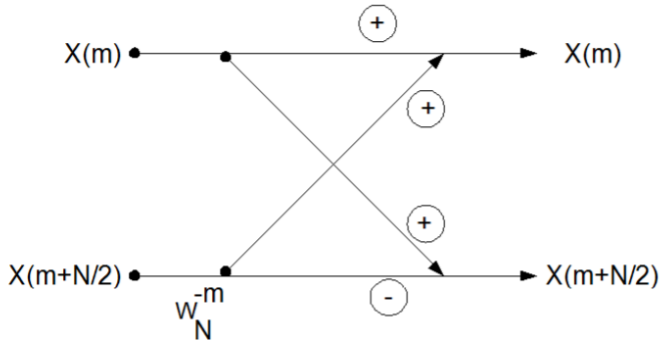
Sumowane wyrażenia są analogiczne do postaci wejciowej ale dotyczą tylko połowy próbek. Współczynnik m ma wartosć od 0 do N-1. Mamy: co oanqcza, że drugą połową wektora jest powtórzenie

$$W_{N/2}^{-(m+N/2)n}=W_{N/2}^{-mn}W_{N/2}^{-(N/2)n}=W_{N/2}^{-mn}e^{-j\frac{2\pi}{N/2}(N/2)n}=W_{N/2}^{-mn}e^{-j2\pi n}\quad=$$

$$=\quad W_{N/2}^{-mn}(\cos(2\pi n)-j\sin(2\pi n))=W_{N/2}^{-mn}(1-j*0)=W_{N/2}^{-mn}$$

pierwszej połowy. Pozwala to na wykonanie N/2 sumowań zamiast N. Stosując te same operacje dla N - nieparzystych. Teraz są mnożone maciezrzy to N/2xN/2 zamiast NxN. Trzeba jeszcze pomnożyć N razy przy składaniu wektorów, ale koszt nadal jest dużo mniejszy. Jako metodę obliczania FFT można zastosować algorytm motylkowy: W motylku wartość X(m+N/2) mnoży się przez odpowiedni współczynnik WN-m odczytany z wektora. Wyniki z prawej strony są uzyskiwane jako suma wartości ze strony lewej (X(m)) oraz jako różnica tych wartości (X(m+N/2)). Znak minus wynika z faktu, że w drugiej połowie wektora współczynników są wartości przeciwne do tych z pierwszej.

Transformacja kosinusowa - wykorzystanie funkcji cosinus o różnych fazach i okresach. DCT II opisuje wzór:



$$X_m = c(m) \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi(2n+1)m}{2N}\right) \quad , \qquad 0 \le m, n \le N-1$$

$$x(n) = \sum_{m=0}^{N-1} c(m) X(m) \cos\left(\frac{\pi(2n+1)m}{2N}\right) \quad , \qquad 0 \le m, n \le N-1$$

$$c(0) = \sqrt{\frac{1}{N}} \quad , \qquad c(m) = \sqrt{\frac{2}{N}} \qquad \text{dla} \quad 0 \le m \le N-1$$

a przekształcenie odwrotne przyjmuje postać:

Sygnał y(n) to sygnał x(n) o zmienionej kolejności próbek, gdzie w pierwszej połowie znajdują się próbki parzyste, a w drugiej próbki nieparzyste w odwróconej kolejnosci.

Szybka transformacja kosinusowa - zastosowanie szybkich algorytmów dyskretnej transformacji Fouriera, stosowanych przy przekształceniu równania DCT do postaci pozwalającej wyodrębnić z niego człon stanowiący DFT.
Gdy mamy sygnał dyskretny opisany jako funkcja można zdefiniować sygnał y(n) wzorami:

$$y(n) = x(2n) \quad \text{i} \quad y(N-1-n) = x(2n+1) \quad \text{dla} \quad 0 \le n \le \frac{N}{2} - 1$$

a przekształcenie odwrotne przyjmuje postać:

$$x(n)=\sum_{m=0}^{N-1} c(m)X(m)\cos\left(\frac{\pi(2n+1)m}{2N}\right) \quad , \qquad 0\le m, n\le N-1$$

$$c(0)=\sqrt{\frac{1}{N}} \quad , \qquad c(m)=\sqrt{\frac{2}{N}} \qquad \text{dla} \quad 0\le m\le N-1$$

Transformatę sygnału x(n) można obliczyćprzy pomocy wzoru: Przy

$$X(m)=Re[c(m)e^{-j\Pi m/2N}DFT_N(y(n))]$$

czym przekształcenie odwrotne opisuje równanie:

$$y(n)=Re[IDFT_N\{c(m)e^{j\Pi m/2N}X(m)\}]$$

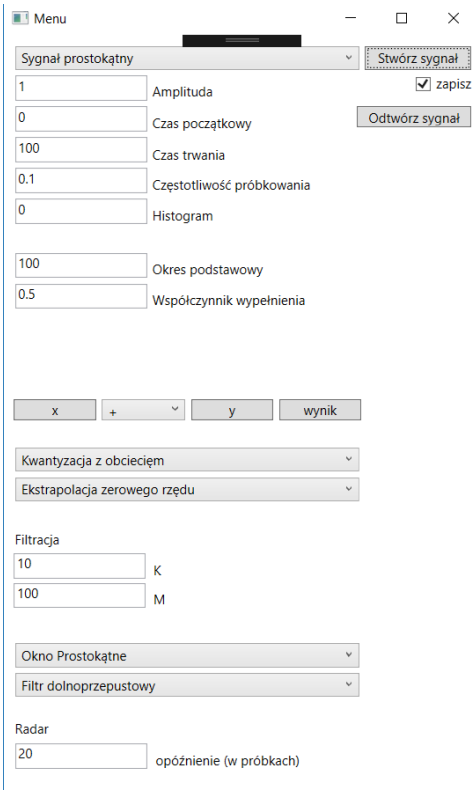
Sygnał x(n) odtwarzamy przy pomocy zmiany kolejności próbek w sygnale y(n).

2.2 Instrukcja obsługi aplikacji

Aplikacja do generacji szumów zawiera interfejs graficzny, który służy do obsługi przez użytkownika. Wygląd został przedstawiony na poniższym rysunku.

Na górze okienka znajduje się wysuwana lista możliwych do generacji sygnałów. Obok znajduje się chceckbox, po zaznaczeniu którego sygnał zostanie zapisany do pliku. Niżej jest przycisk do generacji sygnałów oraz lista parametrów wykresu. Tutaj wpisuje się dane wpływające na sygnał. Pola umożliwiają ustawienie charakterystycznych parametrów sygnału. Na ich podstawie program wylicza wartości amplitudy sygnału w określonym czasie oraz wyświetla graficzną reprezentację sygnału w postaci wykresu funkcji amplitudy od czasu i histogramu.

Na dole okienka znajdują się przyciski do operacji na dwóch sygnałach (dodawanie, odejmowanie itp, a także splot, korelacja). Po kliknięciu w x i y wybieramy odpowiednio pierwszy i dugi składnik działania. Po wcisnięciu przycisku "wynik" program liczy wynik działania i wywietla jego graficzną reprezentację. Poniżej znajdują się wysuwane listy z opcjami kwantyzacji, ekstrapolacji. Niżej znajdują się okienka służące do wpisywania parametróW filtracji. Pod nimi znajdują się do wyboru warianty okna i filtru. Na samym dole okna jest okienko do wpisywania parametrów radaru.



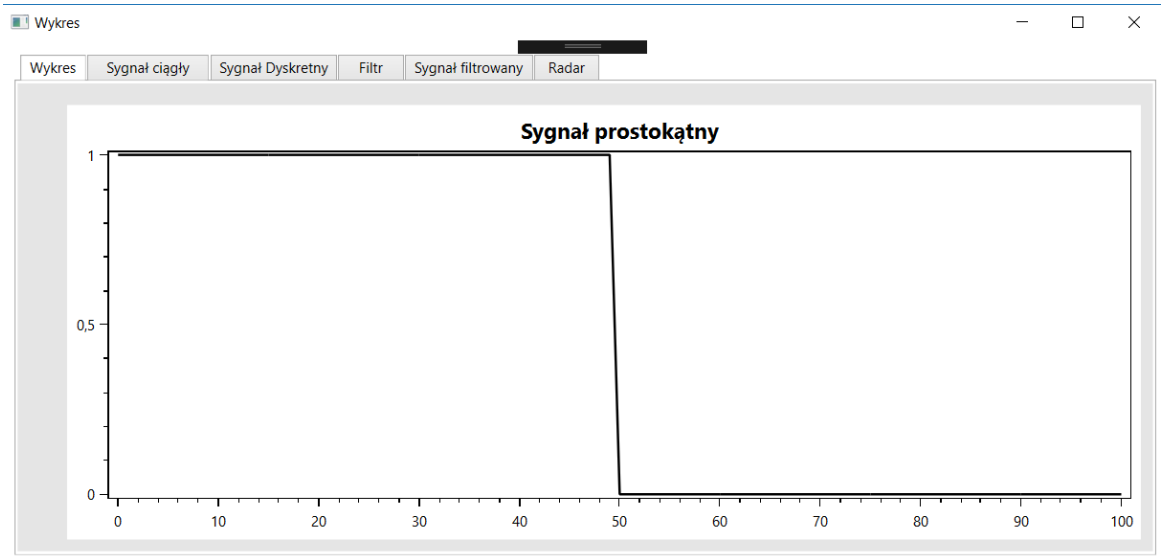
Rysunek 1: Widok główny aplikacji

2.2.1 Generowanie sygnału

Aby wygenerować sygnał użytkownik musi kliknąć w generuj sygnał lub w przypadku innych operacji wynik.
Po wygenerowaniu sygnału pojawiają się dodatkowe okienko aplikacji.
W kolejnych zakładkach okienka są inne wykresy.

2.2.2 Odczyt sygnału z pliku

Oprócz generacji i zapisu do pliku, program umożliwia odczyt z pliku sygnału będącego wynikiem dyskretyzacji (bez kwantyzacji) wygenerowanego sygnału ciągłego oraz sygnału będącego wynikiem operacji na dwóch sygnałach dyskretnych.
Tak jak w przypadku generacji, sygnał jest reprezentowany graficznie w postaci histogramu i wykresu funkcji.



Rysunek 2: Okna po generacji sygnału

2.3 Opis implementacji

Aplikacja została napisana w wysokopoziomowym języku programowania - C#. Do rysowania wykresów została wykorzystana zewnątrzna biblioteka OxyPlot. Program został napisany przy pomocy metodyki obiektowej i stosuje metody numeryczne.

3 Eksperymenty i wyniki

Poniżej znajdują się wszystkie przeprowadzone eksperymenty - możliwe do uzyskania w aplikacji sygnaly i wyniki.

3.1 Eksperyment nr 1

Eksperyment nr 1 - DFT

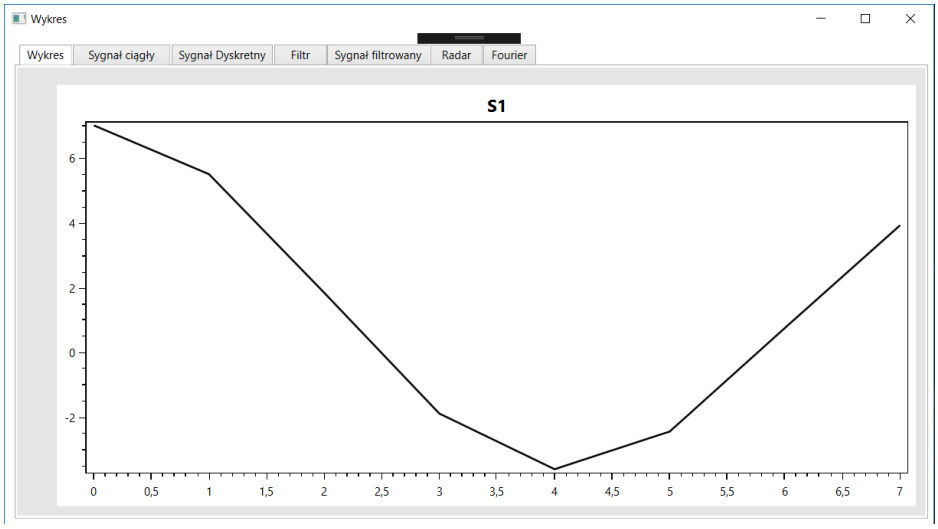
3.1.1 Założenia

Program wykona obliczenia niezbędne do obliczenia dyskretnej transformaty Fouriera 2.1.

3.1.2 Przebieg

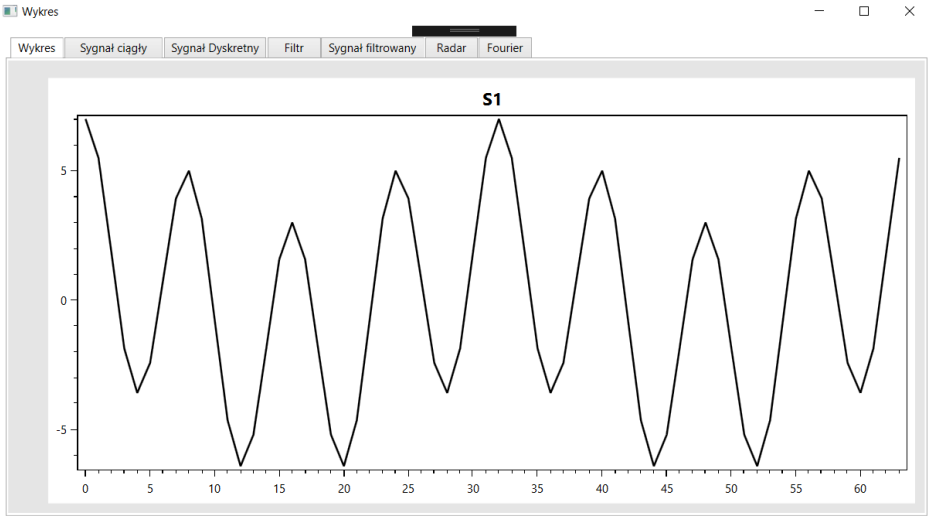
Do generacji synału zostały podane parametry:

Sygnał 1: [n:] 3



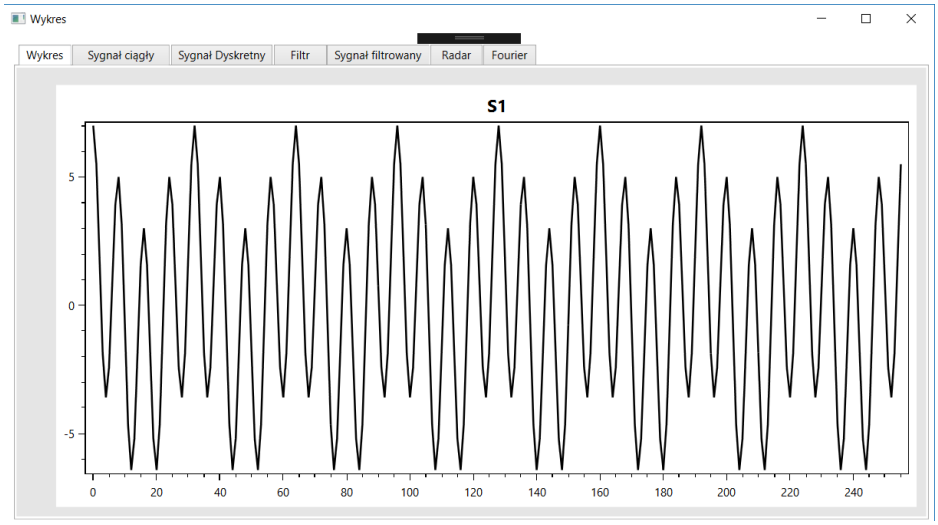
Rysunek 3: Wykres funkcji

Sygnał 2: [n:] 6



Rysunek 4: Wykres funkcji

Sygnał 3: [n:] 8

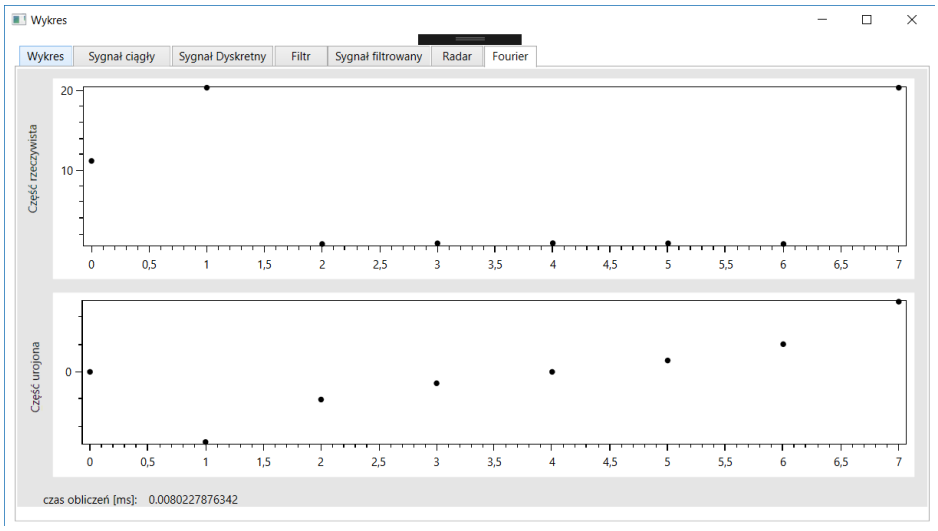


Rysunek 5: Wykres funkcji

3.1.3 Rezultat

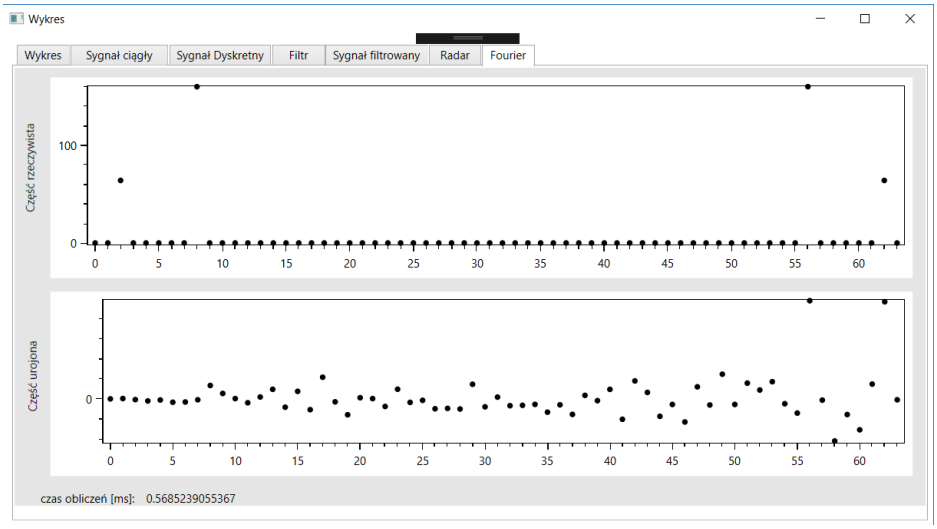
Rezultaty przedstawiają zamieszczone poniżej zrzut ekranu z programu. Czas wykonania oraz wykres dyskretnej transformaty Fouriera.

Sygnał 1:



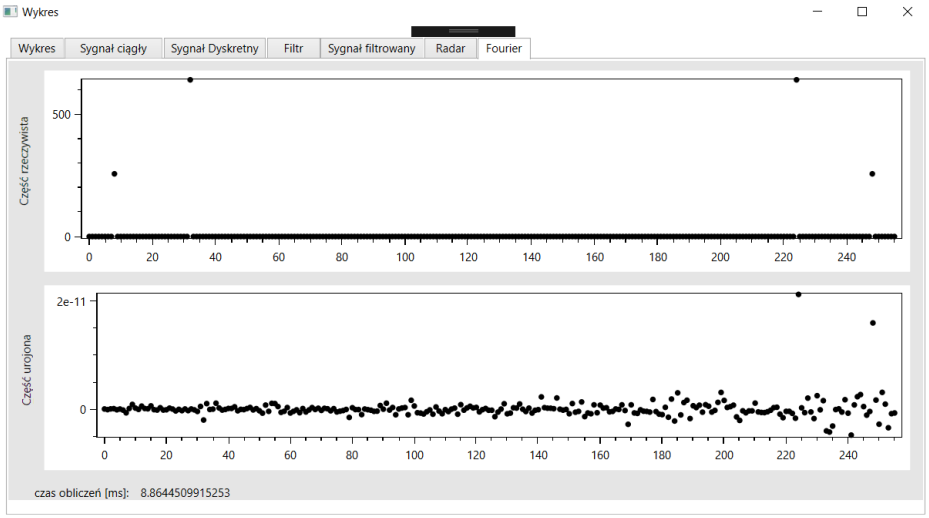
Rysunek 6: Wykres DFT dla sygnału 1

Sygnał 2:



Rysunek 7: Wykres DFT dla sygnału 2

Sygnał 3:



Rysunek 8: Wykres DFT dla sygnału 3

3.2 Eksperyment nr 2

Eksperyment nr 2 - FFT

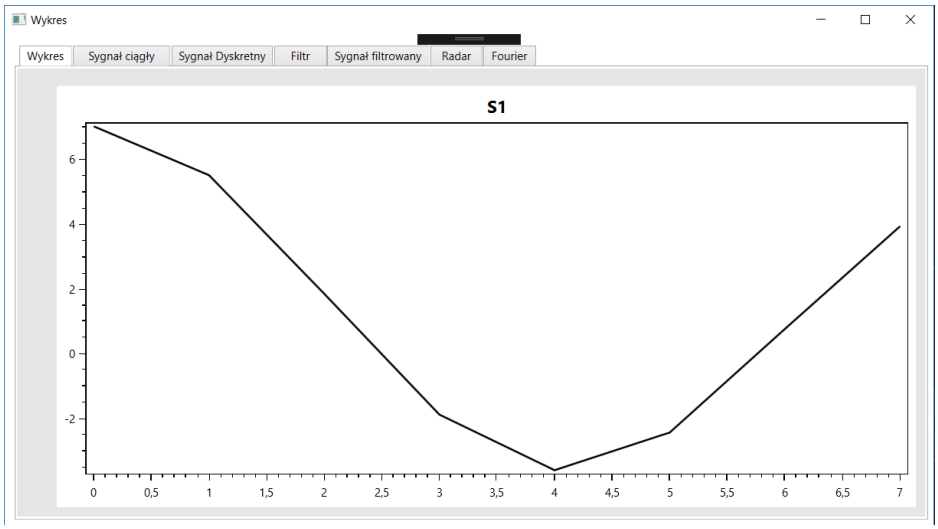
3.2.1 Założenia

Program wykona obliczenia niezbędne do obliczenia dyskretnej transformaty Fouriera 2.1.

3.2.2 Przebieg

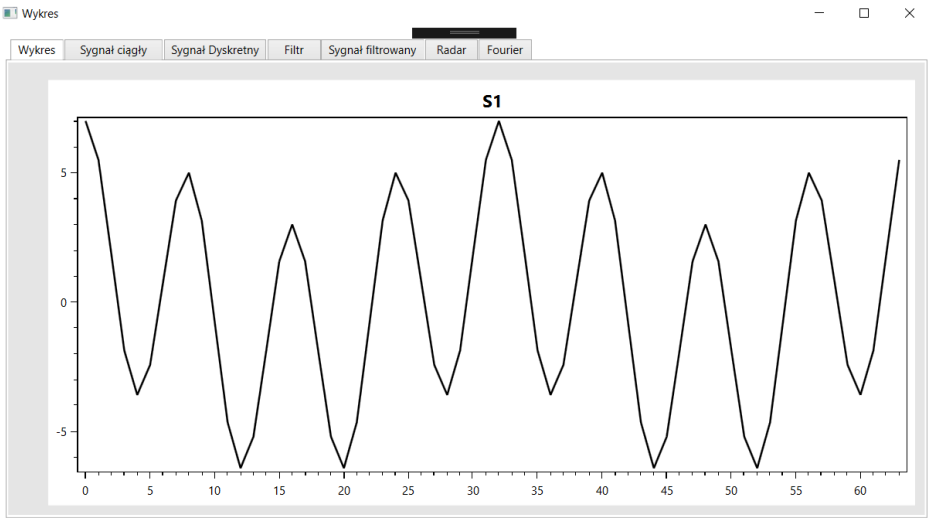
Do generacji synału zostały podane parametry:

Sygnał 1: [n:] 3



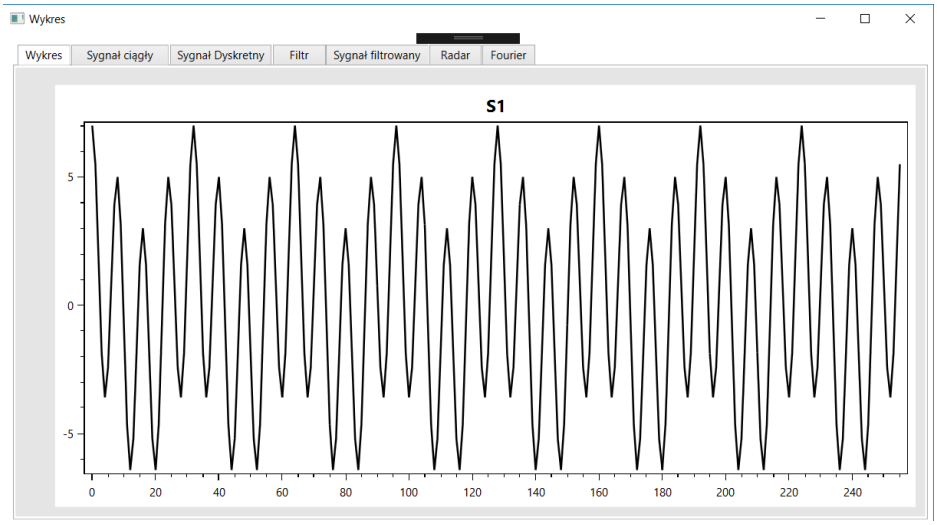
Rysunek 9: Wykres funkcji

Sygnał 2: [n:] 6



Rysunek 10: Wykres funkcji

Sygnał 3: [n:] 8

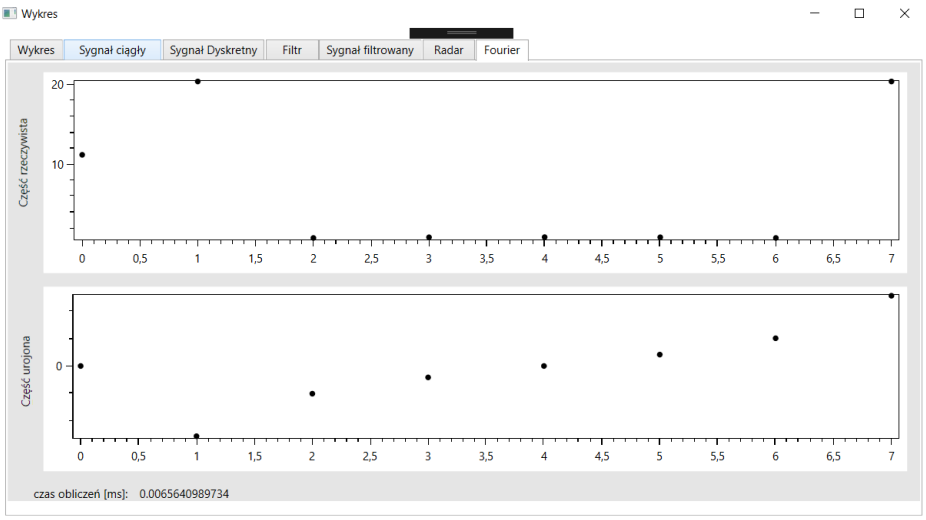


Rysunek 11: Wykres funkcji

3.2.3 Rezultat

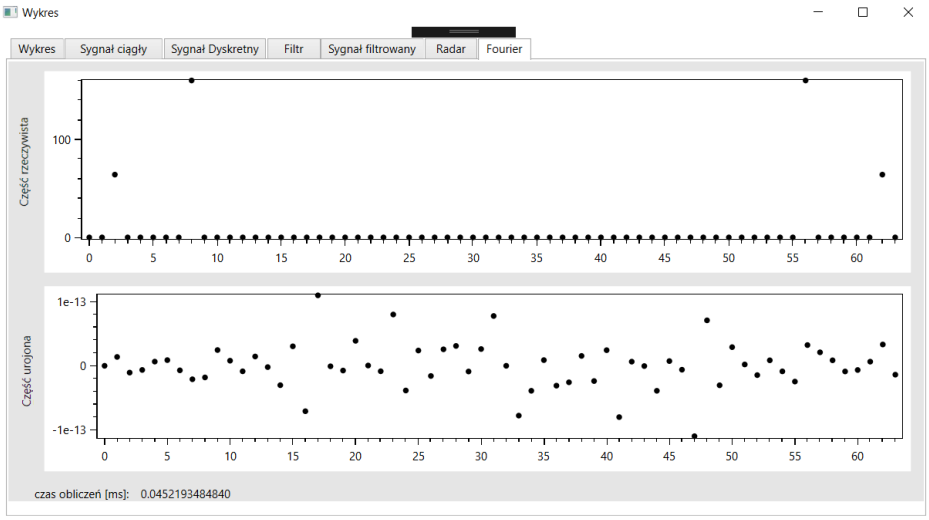
Rezultaty przedstawiają zamieszczone poniżej zrzut ekranu z programu. Czas wykonania oraz wykres dyskretnej transformaty Fouriera.

Sygnał 1:



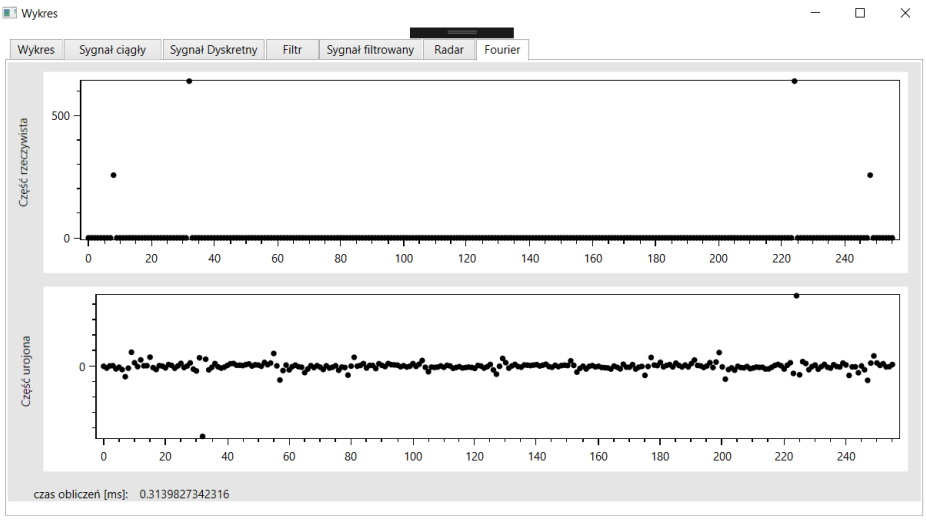
Rysunek 12: Wykres FFT dla sygnału 1

Sygnał 2:



Rysunek 13: Wykres FFT dla sygnału 2

Sygnał 3:



Rysunek 14: Wykres FFT dla sygnału 3

3.3 Eksperyment nr 3

Eksperyment nr 3 -transformacja kosinusowa typu drugiego

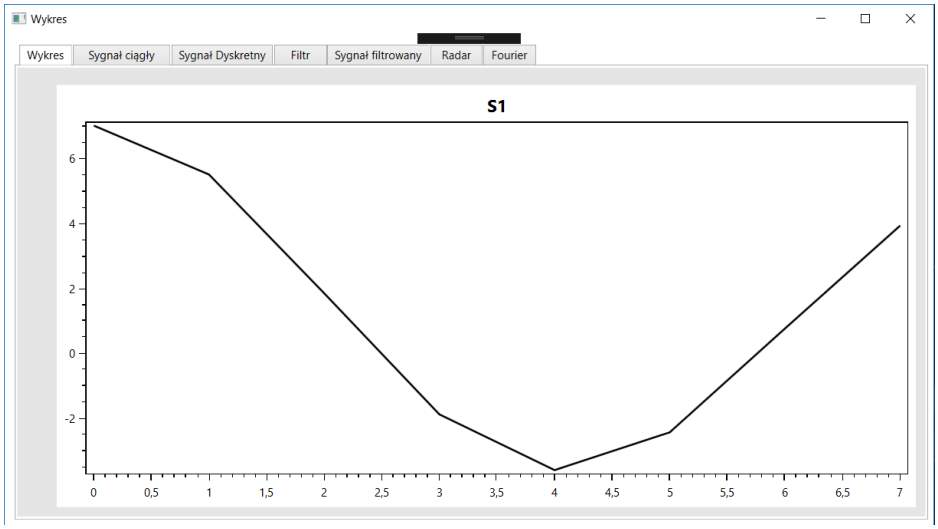
3.3.1 Założenia

Program wykonuje transformację kosinusową typu drugiego

3.3.2 Przebieg

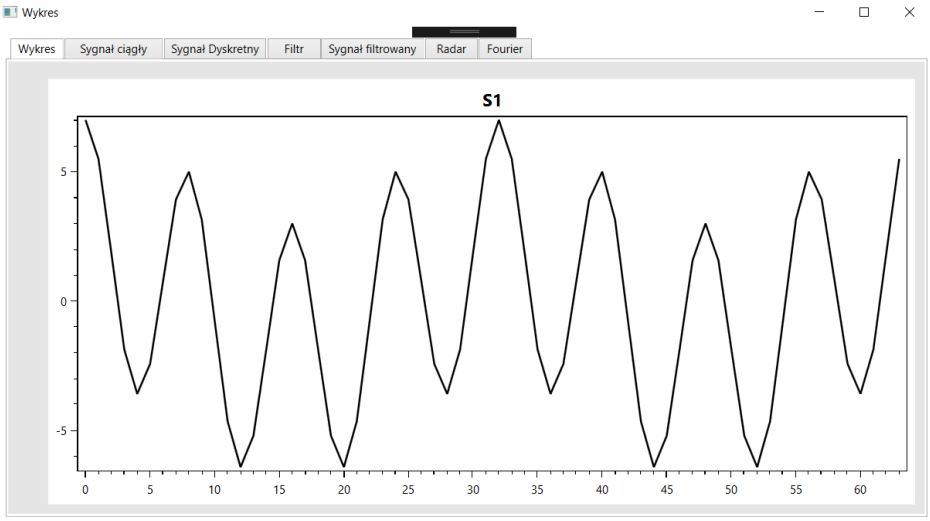
Do generacji synału zostały podane parametry:

Sygnał 1: [n:] 3



Rysunek 15: Wykres funkcji

Sygnał 2: [n:] 6

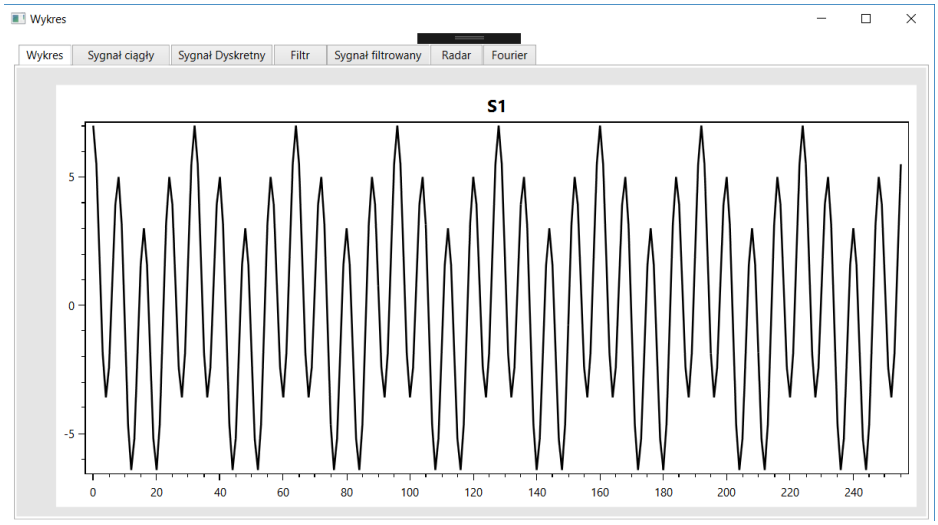


Rysunek 16: Wykres funkcji

Sygnał 3: [n:] 8

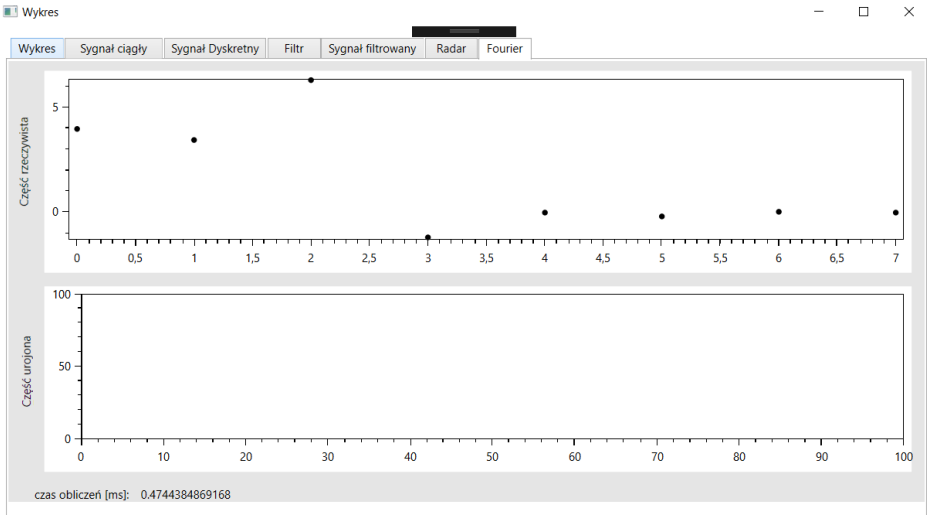
3.3.3 Rezultat

Rezultaty przedstawiają zamieszczone poniżej zrzut ekranu z programu. Czas wykonania oraz wykres transformaciji kosinusowej typu drugiego



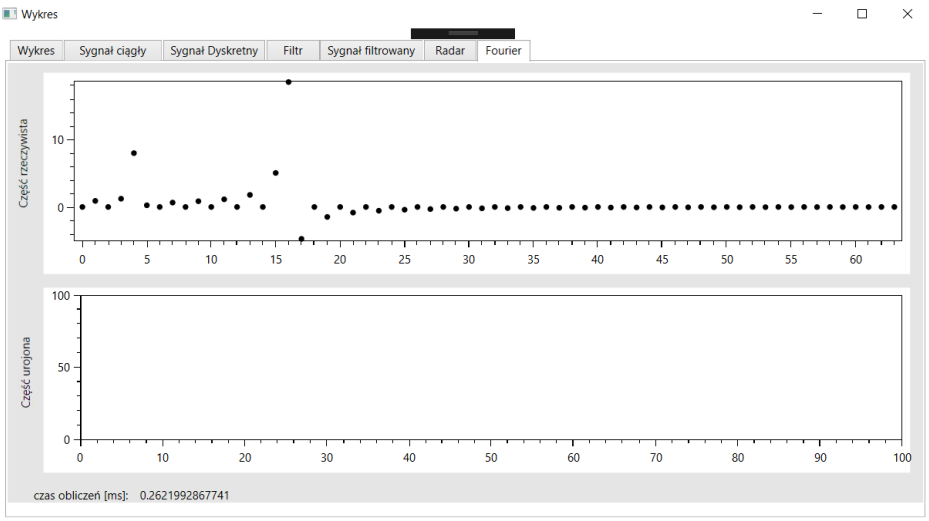
Rysunek 17: Wykres funkcji

Sygnał 1:



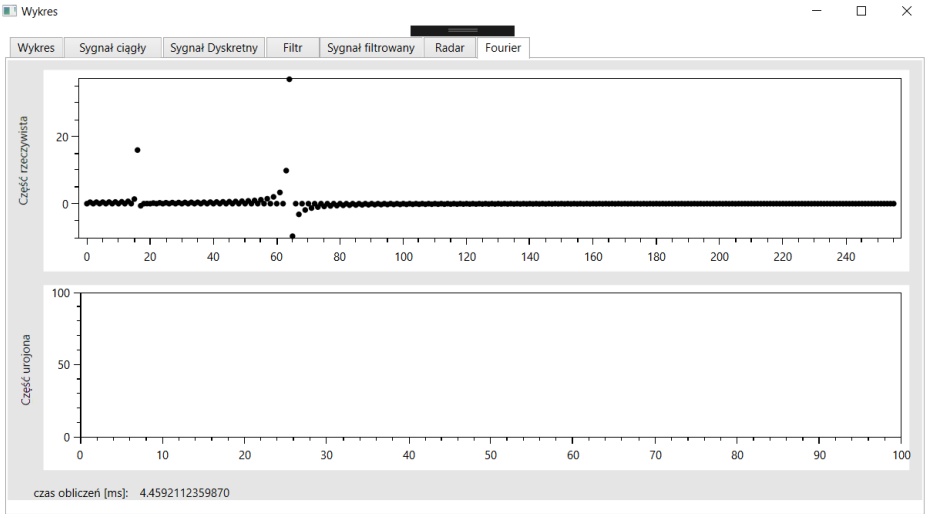
Rysunek 18: Wykres DCT II dla sygnału 1

Sygnał 2:



Rysunek 19: Wykres DCT II dla sygnału 2

Sygnał 3:



Rysunek 20: Wykres DCT II dla sygnału 3

3.4 Eksperyment nr 4

Eksperyment nr 4 - Szybka transformacja kosinusowa

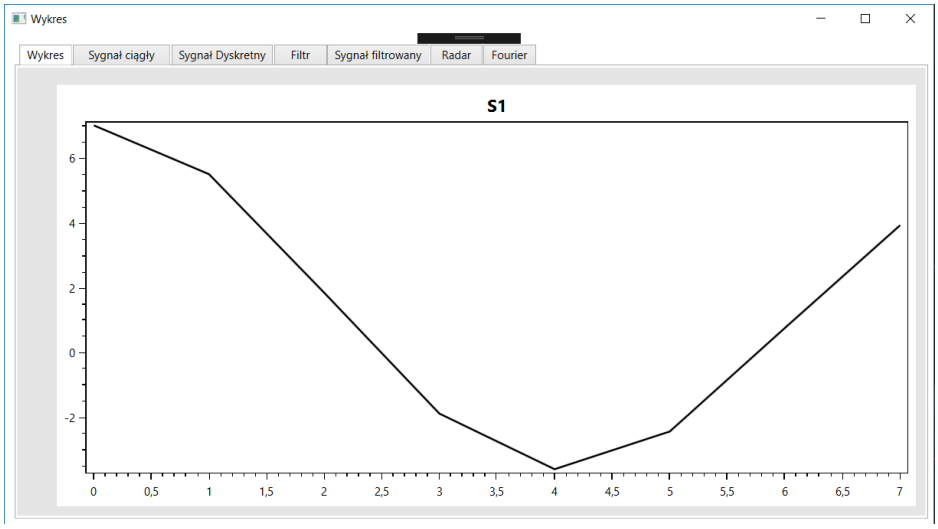
3.4.1 Założenia

Program wykonuje transformację kosinusową typu drugiego

3.4.2 Przebieg

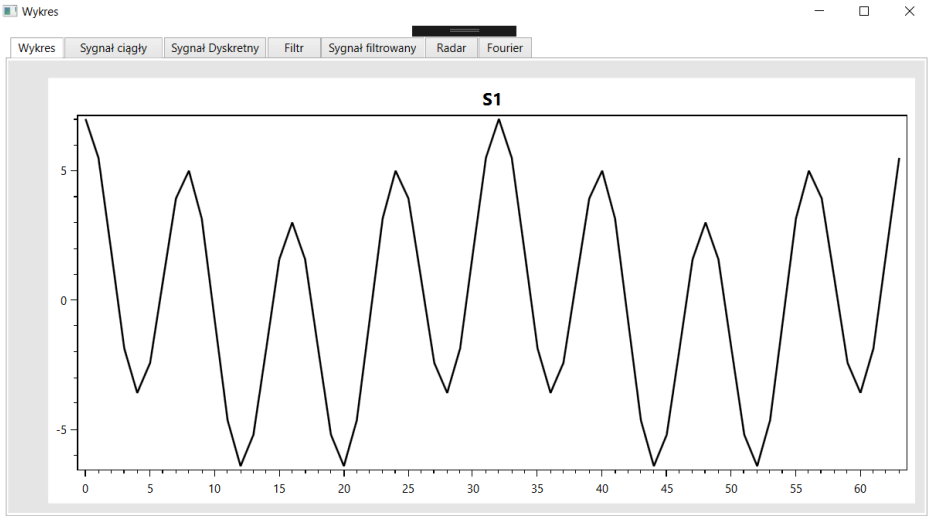
Do generacji synału zostały podane parametry:

Sygnał 1: [n:] 3



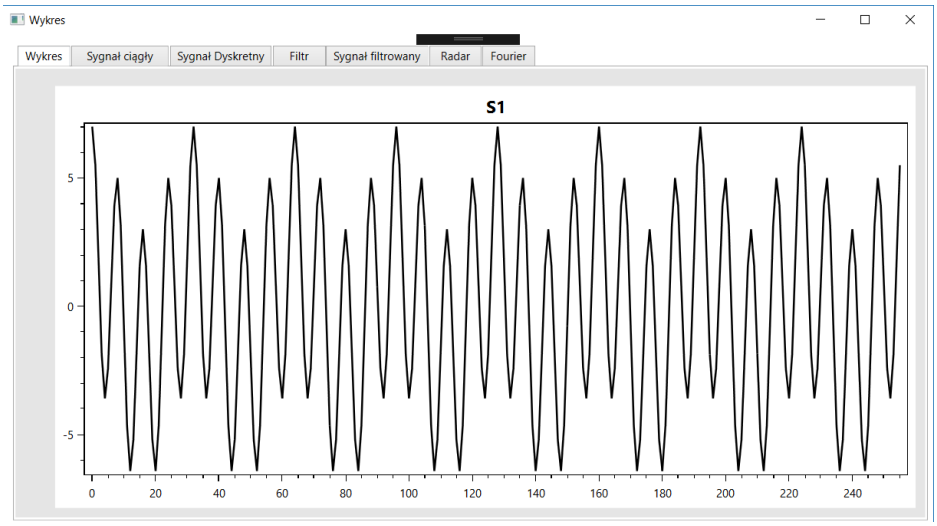
Rysunek 21: Wykres funkcji

Sygnał 2: [n:] 6



Rysunek 22: Wykres funkcji

Sygnał 3: [n:] 8

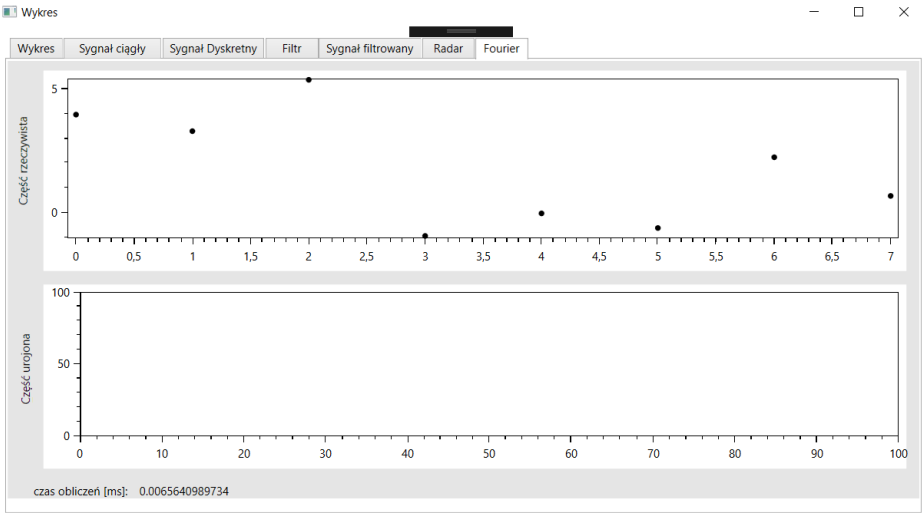


Rysunek 23: Wykres funkcji

3.4.3 Rezultat

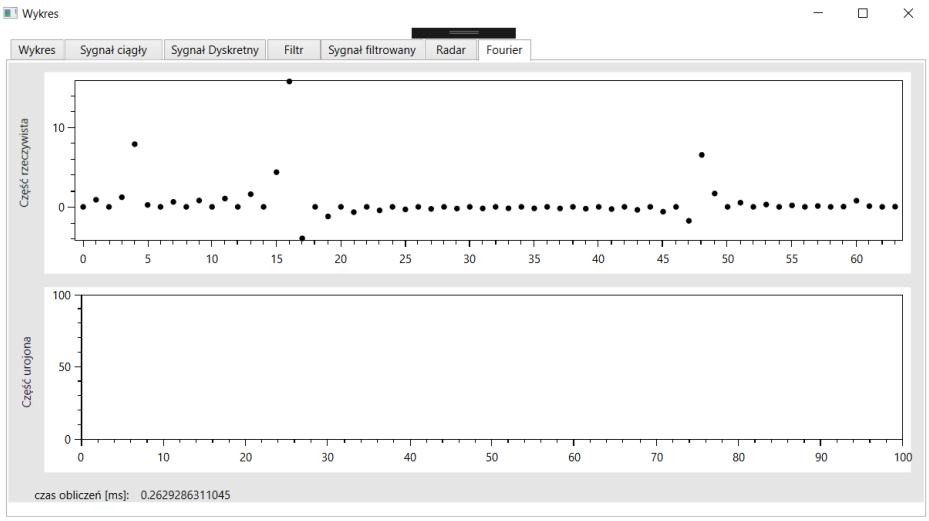
Rezultaty przedstawiają zamieszczone poniżej zrzut ekranu z programu. Czas wykonania oraz wykres transformaciji kosinusowej typu drugiego

Sygnał 1:



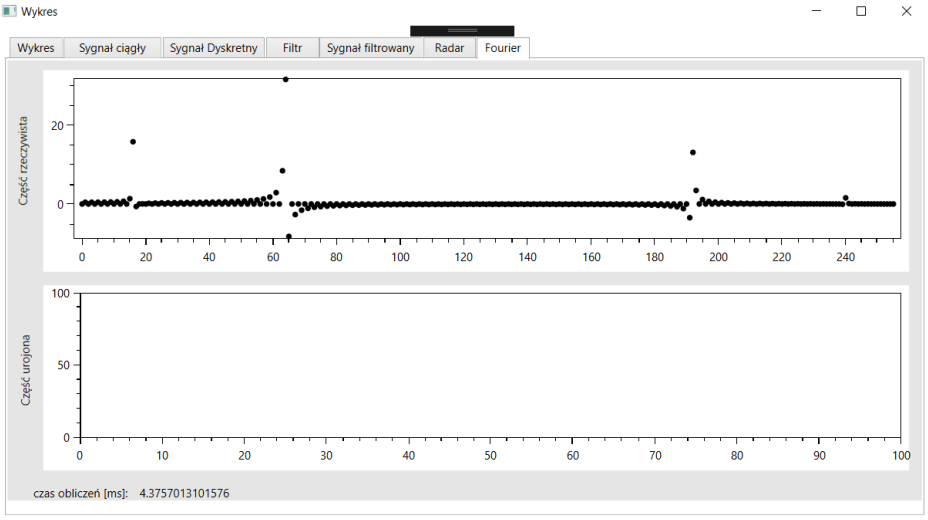
Rysunek 24: Wykres FCT II dla sygnału 1

Sygnał 2:



Rysunek 25: Wykres FCT II dla sygnału 2

Sygnał 3:



Rysunek 26: Wykres FCT II dla sygnału 3

4 Podsumowanie

W poniższych tabelach zostały zebrane czasy wykonania i metody dla każego z naszych sygnałów. Sygnał 1:

Metoda	Czas
DFT	0,56825
FFT	0,00656
$DCTII$	0,47444
$FCTII$	0,00656

Sygnał 2:

Metoda	Czas
DFT	0,056852
FFT	0,045219
$DCTII$	0,26220
$FCTII$	0,26293

Sygnał 3:

Metoda	Czas
DFT	0,31398
FFT	0,26199
$DCTII$	4,445921
$FCTII$	4,37570

5 Wnioski

Pary algorytmów DFT i FFT oraz DCT II i FCT II dają te same wyniki. Różnią się kosztem obliczeniowym i czasem wykonania. W naszym programie zgodnie z założeniami FFT okazało się szybsze niż DFT, jednak FCT II wbrew przewidywaniom potrzebuje dłuższego czasu wykonania niż DCT II. Algorytm nadal ma mniejszy koszt obliczeniowy ale kwestie programistyczne zadecydowały, że jego osiągi są niższe niż można by się spodziewać. Prawdopodobnie jest to spowodowane użyciem w algorytmie mało wydajnego DFT, co za tym idzie o więcej mnożeń.

Literatura

[1] FTIMS Politechnika Łódzka. *Zadanie nr 4 - Przekształcenie Fouriera, Walsha-Hadamarda, kosinusowe i falkowe, szybkie algorytmy*, Wikamp.