# Sprawozdanie – Laboratorium nr 11

# Odszumianie sygnału przy użyciu FFT – splot funkcji

Marcin Urbanowicz

19 maja 2021

## 1. Wstęp teoretyczny

Podczas laboratoriów zajmowaliśmy się odszumaniem sygnałów przy pomocy Szybkiej Transformaty Fouriera (FFT).

### 1.1. Splot funkcji

Jest to działanie określone dla dwóch funkcji lub (jak to było w naszym przypadku) sygnałów dające w wyniku inną, którą można postrzegać jako zmodyfikowaną wersję oryginalnych funkcji.

W naszym przypadku mieliśmy do czynienia ze splotem dwustronnym dwóch funkcji, który jest definiowany następująco:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(x-\tau)d\tau \qquad (1)$$

Gdzie: $f$ i $g$ są funkcjami bezwzględnie całkowalnymi w przedziale (-∞, ∞), czyli należą do przestrzeni $L^1(R)$.

Sploty funkcji są wykorzystywane w statystyce, elektronice, rozwiązywaniu równań różniczkowych, cyfrowym przekształcaniu obrazów i sygnałów.

### 1.2. Transformacja Fouriera

Transformata Fouriera przetwarza funkcję z danej przestrzeni w taki sposób aby uzyskać jej własności okresowe, częstotliwościowe (tak zwane spektrum). Przekształcenie to jest całkowicie odwracalne (istnieje transformata odwrotna). Transformację Fouriera definiujemy w następujący sposób:

$$F(s) = \int_{-\infty}^{\infty} f(t) \cdot e^{-2\pi its} dt \qquad (2)$$

W praktyce często zmienna *t* oznacza czas (w sekundach), a argument transformaty *s* oznacza częstotliwość (w Hz = 1/s)

Transformacja Fouriera znajduje wiele zastosowań w życiu codziennym: w inżynierii, fizyce i matematyce. Rozkład funkcji na funkcje trygonometryczne jest rzeczą, która przydaje się w rozwiązywaniu wielu problemów, takich jak przetwarzanie sygnałów, kompresja lub rozwiązywanie równań różniczkowych.

### 1.3. Szybka transformacja Fouriera

Jest ro algorytm wyznaczania dyskretnej transformaty Fouriera oraz transformaty do niej odwrotnej.

Zakładając, że $x_0, x_1, \dots, x_{N-1}$ są liczbami zespolonymi, wtedy dyskretną transformatę Fouriera definiujemy w następujący sposób:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk} \quad k = 0, 1, \dots, N-1 \tag{3}$$

Obliczanie sum za pomocą powyższego wzoru jest bardzo pracochłonne, gdyż ma on złożoność O ( $N^2$ ).

Algorytmy szybkiej transformacji Fouriera opiera się na dzieleniu transformaty na mniejsze (metodzie dziel i zwyciężaj), dzięki temu złożoność algorytmu można sprowadzić do poziomu O( Nlog(N) ), czyli znacznie zwiększyć wydajność. Najbardziej powszechnie stosowanym jest algorytm Cooleya – Tukeya.

Algorytm ten jest stosowany w podobnych obszarach jak transformata Fouriera, jednak na tych laboratoriach skupiliśmy się na jego zastosowaniu przy przetwarzaniu sygnałów.

# 2. Zadanie do wykonania

## 2.1. Opis problemu

Naszym zadaniem jest odszumienie sygnału, który jest dany wzorem:

$$f(t) = f_0(t) + \Delta \tag{4}$$

Gdzie:

- $f_o(t) = \sin(1 \cdot \omega t) + \sin(2 \cdot \omega t) + \sin(3 \cdot \omega t)$ - jest sygnałem niezaburzonym ($\omega = \frac{2\pi}{T}$ - pulsacja)
- $\Delta$ jest liczbą pseudolosową z zakresu [-0.5, 0.5]

Jako funkcję wagową przyjmujemy funkcję gaussowską:

$$g(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{t^2}{2\sigma^2}} \tag{5}$$

Te funkcje składają się na splot, który jest definiowany wzorem (1) . Splot ten traktujemy jako uśrednienie funkcji sygnału $f$ funkcją wagową $g$. Fakt ten wykorzystamy do wygładzenia sygnału, do którego wykorzystamy szybką transformację Fouriera:

$$FFT\{f(t) * g(t)\} = FFT\{f(t)\} \cdot FFT\{g(t)\} = f(k) \cdot g(k) \tag{6}$$

$$f * g = FFT^{-1}\{f(k) \cdot g(k)\} \tag{7}$$

Przy przeprowadzeniu zadania, zmieniając liczbę *k*, która mogła być liczbą ze zbioru {8, 10, 12} przyjęliśmy następujące parametry:

- $N_k = 2^k$
- Okres: $T = 1.0$
- maksymalny okres rejestracji sygnału: $t_{max} = 3T$
- krok czasowy: $dt = t_{max}/N$
- parametr: $\sigma = T/20$

Dla każdego *k* tworzyliśmy trzy tablice **f, g1, g2** o długości 2N, które początkowo zawierały następujące informacje:

- **f** - sygnał z szumem
- **g1, g2** – wagi

Pierwszym zadaniem było przekształcenie tych tablic posługując się transformacją i transformacją odwrotną w następujący sposób:

- $f_k = FFT\{f\}$
- $g1_k = FFT\{g1\}$
- $g2_k = FFT^{-1}\{g2\}$

Następnie trzeba było wyznaczyć transformatę splotu (wzór 6), którą trzeba było zapisać do tablicy f, którą następnie trzeba było poddać transformacji odwrotnej, dzięki której uzyskalibyśmy wygładzoną funkcję *f*(t)

Do pliku zapisywaliśmy znormalizowany odszumiony sygnał splotu ($f \cdot 2.5/f_{max}$, gdzie $f_{max}$ jest elementem tablicy f o największym module).

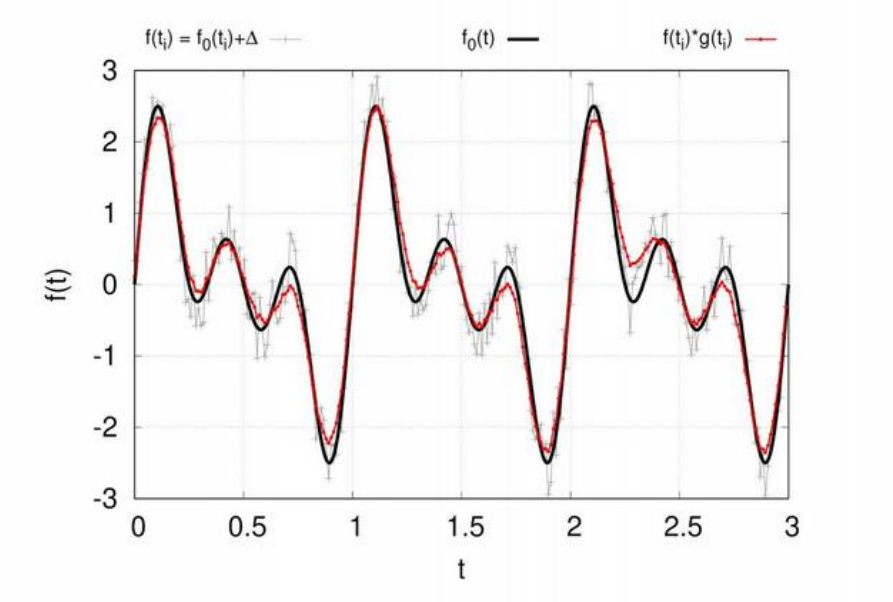
W implementacji posługiwałem się biblioteką GSL, a konkretnie pochodzącymi z niej dwoma procedurami:

- **gsl_fft_complex_radix2_forward(gsl_complex_packed_array y, size_t stride, size_t N)** – Szybka Transformacja Fouriera
- **gsl_fft_complex_radix2_backward(gsl_complex_packed_array y, size_t stride, size_t N)** Transformacja odwrotna do Szybkiej Transformacji Fouriera

## 2.2. Wyniki

Uzyskałem następujące wyniki:

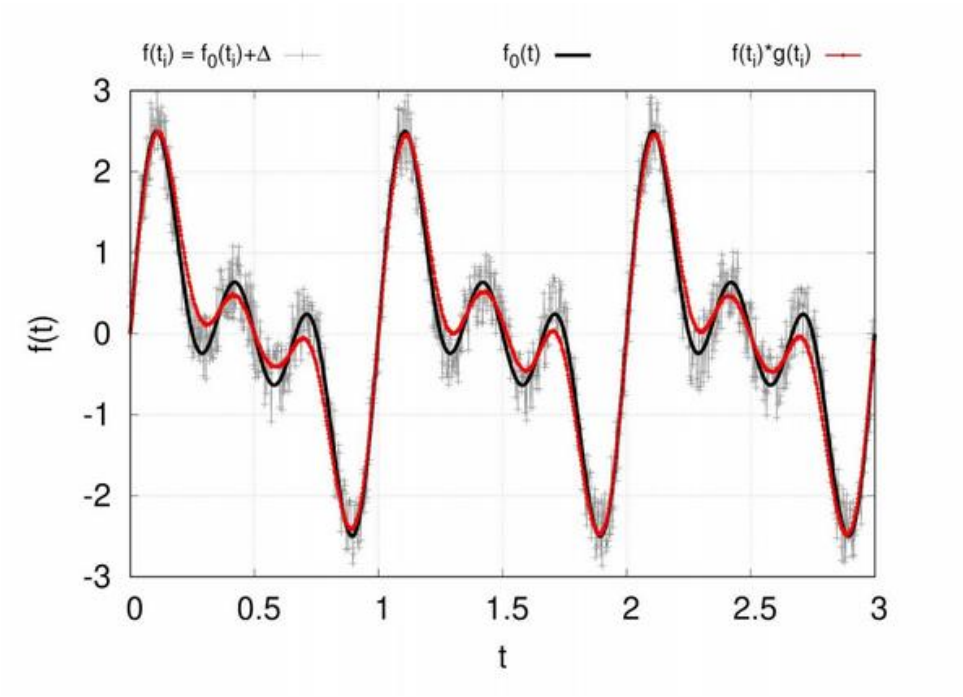
- **Dla k = 8**




Rysunek 1: Wynik odszumiania sygnału (funkcja szara) przy użyciu FFT, liczba próbek wejściowych $N_k = 2^8$

Dla *k* = 8, możemy zauważyć, że wykres czerwony jest niejednolitą linią, co pokazuje małą skuteczność odczumiania w tej sytuacji.

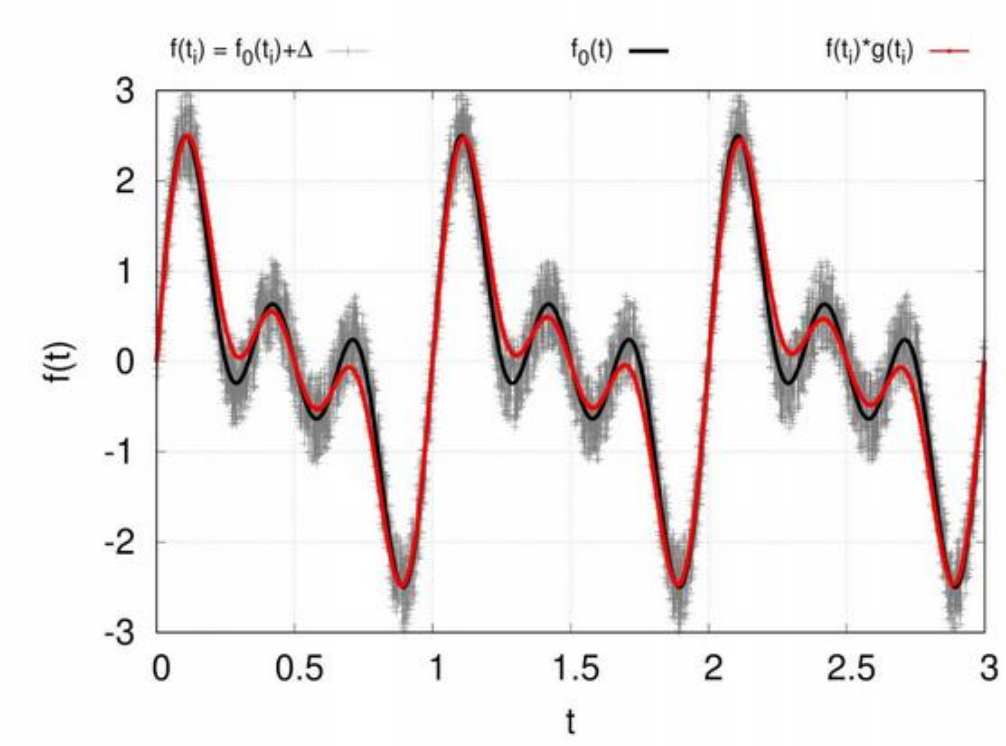
- **Dla k = 10**




Rysunek 2: Wynik odszumiania sygnału (funkcja szara) przy użyciu FFT, liczba próbek wejściowych $N_k = 2^{10}$

Dla *k* = 10, wykres czerwony jest widocznie gładszy niż ten, który jest na *rysunku 1*.

- **Dla k = 12**



Rysunek 3: Wynik odszumiania sygnału (funkcja szara) przy użyciu FFT, liczba próbek wejściowych $N_k = 2^{12}$

Dla k = 12, funkcja jest najlepiej wygładzona w porównaniu z pozostałymi.

## 3. Wnioski

Jak możemy zauważyć Szybka Transformacja Fouriera umożliwia szybkie oraz w miarę dokładne odszumianie sygnału wejściowego. Możemy zauważyć, że dokładność przybliżenia zwiększa się wraz z ilością węzłów (częstością próbkowania). Niestety w żadnym z powyższych przypadków nie otrzymaliśmy wiernego przybliżenia funkcji rzeczywistej (sygnału bez szumu) – nie dla każdego momentu czasu wartości tych funkcji są sobie równe.