Akademia Górniczo-Hutnicza w Krakowie
Wydział Fizyki i Informatyki Stosowanej
Bartosz Rogowski, II rok, 303793
28 maja 2020

Metody numeryczne
Sprawozdanie nr 11 z zajęć laboratoryjnych
Szybka transformata Fouriera

1. Wstęp teoretyczny

Funkcje okresowe można aproksymować szybciej, wykorzystując tę właśnie cechę – okresowość. W tym celu używa się rozwinięcia funkcji w szereg Fouriera, a ogólniej – **zespolony szereg Fouriera**:

$$f(x) \sim \sum_{n=-\infty}^{+\infty} c_n \cdot e^{inx},$$
$$c_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t) \cdot e^{-int}\, dt. \tag{1}$$

> Uwaga: Zakładając, że przedział zbieżności to $x \in [-\pi, \pi]$.
> i to jednostka urojona (taka, że $i^2 = -1$)

W tym miejscu warto przypomnieć, że

$$e^{\pm inx} = \cos(nx) \pm i\sin(nx)\,, \tag{2}$$

a w tej postaci lepiej widać okresowość funkcji trygonometrycznych.

Parametr c_n ze wzoru (1) można zapisać również jako:

$$c_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t) \cdot [\cos(nt) - i\sin(nt)]\, dt = \frac{a_n - ib_n}{2}, \tag{3}$$

a definiując „rozszerzenia” współczynników a_n, b_n

$$\begin{gathered} a_{-n} = -a_n, n = 0, \dots, \infty \\ b_{-n} = -b_n, n = 1, \dots, \infty \\ b_0 = 0, \end{gathered} \tag{4}$$

można zapisać

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{+\infty}[\cos(nt) + i\sin(nt)] = \sum_{n=-\infty}^{+\infty} c_n \cdot e^{inx}. \tag{5}$$

Niech

$$E_n(x) = e^{inx}, n \in \mathbb{Z}. \tag{6}$$

Funkcje te tworzą ciąg ortogonalnych jednomianów eksponencjalnych, z których można utworzyć wielomian interpolacyjny, który może posłużyć do interpolacji:

$$f(x) = P(x) = \sum_{n=0}^{N-1} c_n \cdot E_n(x). \tag{7}$$

Ponieważ w implementacji nie można posłużyć się ciągłą serią danych (punkty pomiarowe/dane są przecież dyskretne), zachodzi potrzeba zmiany sposobu liczenia transformat (całkę trzeba zamienić na jej dyskretny odpowiednik). Aby wyznaczyć współczynnik, używa się wzoru:

$$c_n = \sum_{k=0}^{N-1} f_k(x) \cdot E_n(-x), \tag{8}$$

> Uwaga:
> Minus we wzorze występuje z powodu minusa w eksponencie we wzorze (1).

a przyjmując siatkę węzłów równoodległych:

$$x_k = \frac{2\pi k}{N}, k = 0, \dots, N-1, \tag{9}$$

otrzymamy wzór

$$c_n = \sum_{k=0}^{N-1} f_k(x) \cdot E_n(-x_k) = \sum_{k=0}^{N-1} f_k(x) \cdot e^{-\frac{2i\pi nk}{N}}. \tag{10}$$

Stosując oznaczenie

$$\omega_n = e^{-\frac{2i\pi}{n}} \tag{11}$$

oraz

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \omega_n^3 & \cdots & \omega_n^{N-1} \\ 1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \cdots & \omega_n^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{N-1} & \omega_n^{2(N-1)} & \omega_n^{3(N-1)} & \cdots & \omega_n^{(N-1)(N-1)} \end{bmatrix}, \tag{12}$$

problem ten można sprowadzić do macierzy:

$$\mathbf{W}f = c. \tag{13}$$

> Uwaga:
> f – wektor wartości funkcji w węzłach – czyli $f(x_k)$
> c – wektor współczynników (patrz: wzór (10))

Ciąg współczynników oznaczonych jako c w równaniu (13) definiuje **dyskretną transformatę Fouriera (ang. DFT – Discrete Fourier Transform)**. Analiza złożoności takiego sposobu zdradza, że potrzeba $\mathrm{O}(N^2)$ operacji na jego wykonanie. Dla dużych zestawów danych sposób ten staje się mało wydajny (co jest konieczne w zastosowaniach, o których za chwilę). Aby przyspieszyć ten proces, stosuje się właśnie **szybką transformatę Fouriera (ang. FFT – Fast Fourier Transform)**. Jej zaletą jest to, że skraca liczbę operacji, bowiem złożoność **FFT** wynosi $\mathrm{O}(N \cdot \log_2 N)$, a dla dużych zestawów danych zależność ta jest stosunkowo niewiele większa od zależności liniowej, zatem jest to znacznie wydajniejsza metoda. Wzór (10) można rozpisać dalej jako sumowanie po parzystych i nieparzystych elementach:

$$c_n = \sum_{m=0}^{\frac{N}{2}-1} f_{2m}(x) \cdot e^{-\frac{2i\pi n(2m)}{N}} + \sum_{m=0}^{\frac{N}{2}-1} f_{2m+1}(x) \cdot e^{-\frac{2i\pi n(2m+1)}{N}}. \tag{14}$$

Prawą (pomarańczową) sumę można rozpisać dalej jako:

$$\sum_{m=0}^{\frac{N}{2}-1} f_{2m+1}(x) \cdot e^{-\frac{2i\pi n(2m+1)}{N}} = \sum_{m=0}^{\frac{N}{2}-1} f_{2m+1}(x) \cdot e^{-\frac{2i\pi n\left(m+\frac{1}{2}\right)}{\frac{N}{2}}} =$$

$$= \sum_{m=0}^{\frac{N}{2}-1} f_{2m+1}(x) \cdot e^{-\frac{2i\pi nm}{\frac{N}{2}} - \frac{i\pi n}{\frac{N}{2}}} = \sum_{m=0}^{\frac{N}{2}-1} f_{2m+1}(x) \cdot e^{-\frac{2i\pi nm}{\frac{N}{2}}} \cdot e^{-\frac{i\pi n}{\frac{N}{2}}}. \tag{15}$$

Można dowieść, że gdy m przekroczy połowę N, wartości eksponenty zaczną się powtarzać. Ponadto analizując wzory (14) i (15) można dojść do wniosku, że prawy składnik ze wzoru (14, zaznaczony na pomarańczowo) różni się tylko o stałą (niebieska stała we wzorze (15)). Można więc zapisać:

$$c_n = p_k + q_k \cdot \varphi_k. \tag{16}$$

> Uwaga:
> Oznaczenia odpowiadają kolorowym członom we wzorach (14) oraz (15).

Korzystając z powtarzalności (wspomnianej wyżej), można otrzymać następujące zależności:

$$p_{k+\frac{N}{2}} = p_k, \tag{17}$$

$$q_{k+\frac{N}{2}} = q_k, \tag{18}$$

$$\varphi_{k+\frac{N}{2}} = -\varphi_k. \tag{19}$$

Podział taki jak we wzorze (14) należy powtarzać, dopóki nie otrzymany zostanie pojedynczy element, tzn.

$$k \rightarrow \begin{cases} 2m+0 \rightarrow \begin{cases} 4m+0 \rightarrow \begin{cases} \ldots \rightarrow \begin{cases} \ldots \\ \ldots \end{cases} \\ \ldots \rightarrow \begin{cases} \ldots \\ \ldots \end{cases} \end{cases} \\ 4m+2 \rightarrow \begin{cases} \ldots \rightarrow \begin{cases} \ldots \\ \ldots \end{cases} \\ \ldots \rightarrow \begin{cases} \ldots \\ \ldots \end{cases} \end{cases} \end{cases} \\ 2m+1 \rightarrow \begin{cases} 4m+1 \rightarrow \begin{cases} \ldots \rightarrow \begin{cases} \ldots \\ \ldots \end{cases} \\ \ldots \rightarrow \begin{cases} \ldots \\ \ldots \end{cases} \end{cases} \\ 4m+3 \rightarrow \begin{cases} \ldots \rightarrow \begin{cases} \ldots \\ \ldots \end{cases} \\ \ldots \rightarrow \begin{cases} \ldots \\ \ldots \end{cases} \end{cases} \end{cases} \end{cases} \tag{20}$$

Jak widać na powyższym schemacie, każdy podział wyznacza dwa kolejne. Liczba podziałów będzie proporcjonalna do $O(\log_2 N)$. Następnie wystarczy wymnożyć otrzymane współczynniki N razy zgodnie z powyższymi formułami, a więc złożoność **FFT** – $O(N \cdot \log_2 N)$ bierze się właśnie dzięki takim przekształceniom. Zakłada się istnienie liczby sygnałów, która jest potęgą dwójki (algorytm działa wtedy najwydajniej). Np. dla 1000 węzłów dodanie 24 zer

będzie optymalniejsze, niż choćby kolejne dzielenia tysiąca. Taki algorytm obliczania **FFT** został podany przez J. W. Cooleya oraz J. Tukeya w latach 60. ubiegłego wieku (nosi również nazwę *radix-2*) [1]. Jest on jednym z najważniejszych algorytmów, ponieważ „szybko" oblicza **DFT**, a ona z kolei jest powszechnie używana do:

- rozwiązywania cząstkowych równań różniczkowych
- odszumiania sygnałów
- kompresji plików (audio oraz obrazy)
- transmisji danych,

a to tylko kilka przykładów zastosowań.

Aby pokazać w jaki sposób odszumić sygnał, konieczne jest wprowadzenie kilku definicji.

Splot dwóch funkcji jest definiowany jako:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(r)g(r-t)dr. \tag{21}$$

Traktując $f(t)$ jako sygnał zależny od czasu (najczęściej zawierający szum – oscylacje wartości), a $g(t)$ jako funkcję wagową, to ich splot można rozpatrywać jako pewne uśrednienie funkcji f. Jest to ważne założenie w odszumianiu sygnałów. **FFT** splotu jest łatwa do obliczenia, bowiem:

$$FFT\{f(t) * g(t)\} = FFT\{f\} \cdot FFT\{g\} = f(k) \cdot g(k)\ [*]. \tag{22}$$

Jako funkcję wagową można przyjąć np. rozkład Gaussa dany wzorem:

$$g(t) = G_{\mu,\sigma}(t) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(t-\mu)^2}{2\sigma^2}}. \tag{23}$$

Uwaga:
μ – wartość oczekiwana (odpowiada za przesunięcie krzywej)
σ – odchylenie standardowe (odpowiada za smukłość kształtu krzywej)

Rzeczywiste dane są zbierane w przedziale czasowym $[0, t_{max}]$, a przedział zbieżności musi być symetryczny, zatem konieczne jest „rozbicie" $g(t)$ na dwa przypadki:

$$g(k) = \begin{cases} g_1(k) = FFT\{g(t>0)\} \\ g_2(k) = FFT\{g(t<0)\} \end{cases}$$
$$g_2(k) = FFT^{-1}\{g(t>0)\}. \tag{24}$$

FFT^{-1} – czyli **odwrotna szybka transformata Fouriera** jest liczona niemal identycznie jak we wzorze (10), z tym, że eksponenta jest dodatnia i cała suma jest dodatkowo dzielona przez N.

Zatem g to suma:

$$g(k) = g_1(k) + g_2(k) = FFT\{g(t)\} + FFT^{-1}\{g(t)\}. \tag{25}$$

Jeśli założyć, że ma się do czynienia z funkcją zespoloną, warto operować na $2N-1$ węzłach, wtedy elementy o indeksach parzystych będą stanowić część rzeczywistą, a nieparzyste – urojoną, a implementacja splotu będzie wyglądać następująco (zakładając, że funkcje te zostały już uprzednio „obłożone" odpowiednim przekształceniem **Fouriera**):

$$f_{2j} = f_{2j} \cdot f_{2j+1} - g_{2j} \cdot g_{2j+1} \tag{26}$$

[*] Z tego zapisu widać, że **FFT** zamienia dziedzinę (np. z czasu na częstotliwość).

$$f_{2j+1} = f_{2j} \cdot g_{2j+1} + g_{2j} \cdot f_{2j+1}. \quad (27)$$

Uwaga:
$j = 0, \dots, N-1$

Ostatnim krokiem jest wykonanie **odwrotnej FFT** na tak przygotowanej tablicy f:

$$FFT^{-1}\{f(k)\} = f(t) * g(t), \quad (28)$$

a tak otrzymany splot będzie stanowił aproksymującą funkcję – odszumiony sygnał.

2. Zadanie do wykonania

2.1 Opis problemu

Naszym zadaniem było wygenerowanie sztucznego sygnału z pseudolosowym szumem:

$$f(t) = f_0(t) + \Delta, \quad (29)$$

$$f_0(t) = \sin(\omega t) + \sin(2 \cdot \omega t) + \sin(3 \cdot \omega t), \quad (30)$$

Uwaga:
$\omega = \frac{2\pi}{T}$ – pulsacja

$$\Delta = \frac{rand()}{RAND_{MAX} + 1.0} - \frac{1}{2} = Rand\left(-\frac{1}{2}, \frac{1}{2}\right), \quad (31)$$

a następnie odtworzenie f_0 za pomocą algorytmu odszumiającego opisanego powyżej. W tym celu konieczne było użycie **FFT** za pomocą funkcji *gsl_fft_complex_radix2_forward* oraz *gsl_fft_complex_radix2_backward* (**FFT-1**) znajdujących się w bibliotece *GSL*. Wywołanie tych funkcji jest identyczne – jako argumenty przyjmują tablicę zawierającą sygnał, krok między elementami (u nas 1) oraz liczbę próbek (potęga dwójki – założenie algorytmu).

W treści zostały podane następujące parametry:

$T = 1$	okres
$t_{max} = 3 \cdot T$	koniec przedziału
$k = 8, 10, 12$	rząd liczby danych (wykładnik dwójki)
$N = 2^k$	liczba węzłów
$dt = \frac{t_{max}}{N}$	krok czasowy (dane są dyskretne)
$\sigma = \frac{T}{20}$	parametr funkcji wagowej (rozkład Gaussa) – odchylenie standardowe
$\mu = 0$	parametr funkcji wagowej (rozkład Gaussa) – wartość oczekiwana

Tabela 1. Parametry podane w treści zadania.

Wartości funkcji w węzłach f oraz funkcji wagowych g_1, g_2 były trzymane w tablicach długości $2N$ [†]. Tablice te (wartości) zostały poddane **transformacji Fouriera**, następnie został policzony splot (według wzorów (26) i (27)), a na końcu **odwrotna FFT** dla f. Ostatnim krokiem była normalizacja wartości f zgodnie we wzorem:

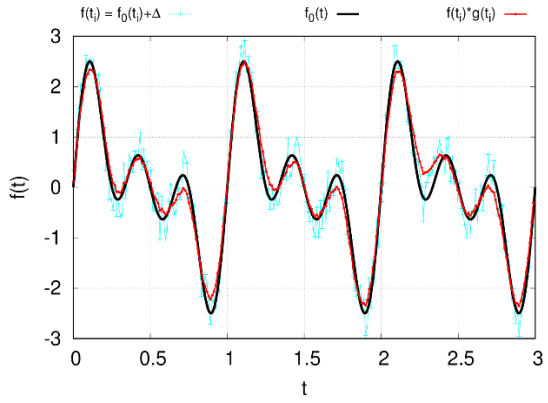
[†] Konieczność „podziału” na elementy rzeczywiste i urojone (w naszym przypadku były to zera, a więc tablice o elementach nieparzystych zostały wyzerowane)

$$f(t_i) = \frac{2{,}5 \cdot f(t_i)}{f_{max}}. \tag{32}$$

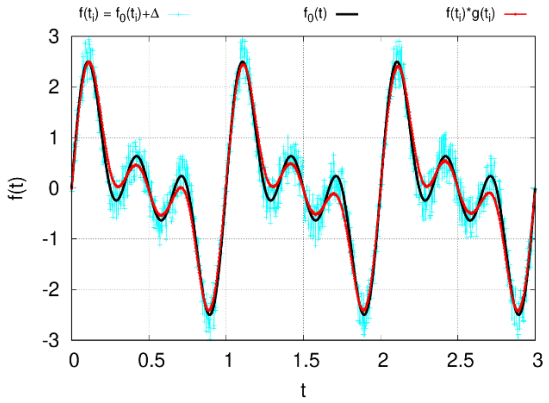
Uwaga:
$i = 0, \dots, N-1$
$t_i = i \cdot dt$
f_{max} – element rzeczywisty (indeks parzysty) o największym module

2.2 Wyniki

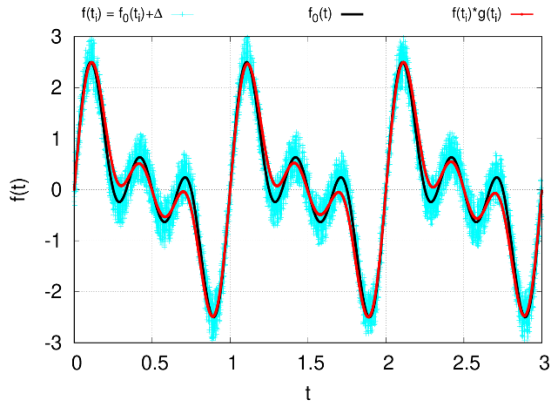
Dla trzech podanych (w tabeli 1) wartości k do trzech plików zostały zapisane wartości (sztucznie) zaszumionego sygnału oraz wartości funkcji wygładzonej i znormalizowanej zgodnie ze wzorem (32). Specjalnie przygotowany skrypt pozwolił wygenerować wykresy za pomocą programu Gnuplot. Na poniższych rysunkach zestawiono zaszumioną funkcję, oryginał f_0 oraz splot – odszumioną funkcję.



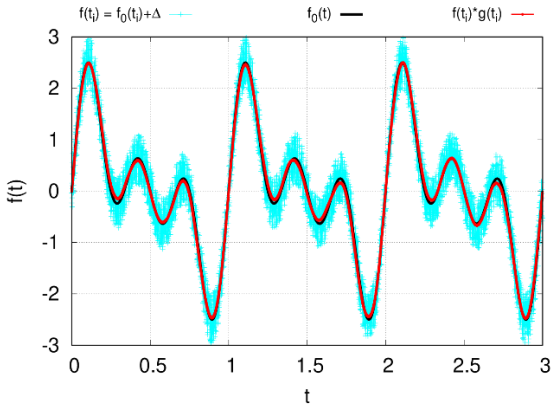
Rysunek 1. Rezultat dla 2^8 próbek.



Rysunek 2. Rezultat dla 2^{10} próbek.



Rysunek 3. Rezultat dla 2^{12} próbek.



Rysunek 4. Rezultat dla 2^{12} próbek, $\sigma = \frac{T}{40}$.

Dodatkowo, by przetestować jakość dopasowania, zmieniona została wartość parametru σ odpowiadającego za kształt funkcji g, a więc i kształt f (splot). Rezultat został zestawiony na rys. 4 w celu porównania z rys. 3, w którym zastosowano σ podaną w treści zadania (tabela 1).

W tabeli 2 zestawiono największe moduły w zależności od liczby próbek.

liczba próbek $N = 2^k$	$k = 8$	$k = 10$	$k = 12$
f_{max}	52840,2	734220	$1{,}10917 \cdot 10^7$

Tabela 2. Największe moduły dla podanych liczb próbek.

3. Wnioski

- Szybka Transformata Fouriera – FFT – jest jednym z najważniejszych algorytmów z uwagi na dość prostą implementację, ale przede wszystkim na dość szerokie pole zastosowań (zwłaszcza w przetwarzaniu sygnałów, danych, itp.).
- Jednym z ważniejszych zastosowań FFT jest odszumianie sygnałów, co zostało udowodnione na rysunkach 1 – 4.
- Jakość otrzymanych wyników zależy od kilku parametrów. Wpływ na to mają m.in. liczba węzłów oraz częstość próbkowania – czyli odległości między węzłami, ponieważ im większe te wartości, tym lepszy rezultat (potwierdzają to kolejne wykresy 1 – 3 dla rosnącej liczby próbek).
- Ponieważ sigma (odchylenie standardowe) wpływa na kształt funkcji, więc to od tego parametru również zależy dopasowanie krzywej do punktów węzłowych t_i. Im mniejsze odchylenie, tym krzywa aproksymująca bardziej przypomina zaszumiony sygnał (od pewnego stopnia w splocie również występuje szum, funkcja przechodzi przez większą liczbę węzłów), natomiast duże wartości σ skutkują błędnym odtworzeniem funkcji, dlatego ważne jest znalezienie optymalnej wartości (np. dla 2^{12} próbek $\sigma = \frac{T}{40}$ wydaje się bardzo dobrym parametrem (patrz: rysunek 4)). Należy także pamiętać, żeby dostosować odchylenie do liczby próbek – ten sam parametr wpływa inaczej na rezultat dla różnych zestawów danych (np. na rys. 1 widać drobny szum podczas, gdy na rys. 2 i 3 funkcja zdaje się być gładka).

4. Bibliografia

W części teoretycznej posiłkowano się poniższymi materiałami:
https://www.youtube.com/watch?v=nl9TZanwbBk [data dostępu: 25 maja 2020]
https://www.youtube.com/watch?v=E8HeD-MUrjY [data dostępu: 25 maja 2020]
https://www.youtube.com/watch?v=htCj9exbGo0 [data dostępu: 26 maja 2020]

[1] źródło: https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm [data dostępu: 26 maja 2020]