Analiza szeregów czasowych:

1. Dyskretna transformata Fouriera i zagadnienia pokrewne

P. F. Góra

http://th-www.if.uj.edu.pl/zfs/gora/

semestr letni 2007/08

Transformata Fouriera

$$G(f) = \int_{-\infty}^{\infty} g(t) e^{2\pi i f t}\, dt \tag{1}$$

Transformata odwrotna:

$$g(t) = \int_{-\infty}^{\infty} G(f) e^{-2\pi i f t}\, df \tag{2}$$

Trzeba pamiętać gdzie w używanej konwencji pojawia się 2π. Aby transformata Fouriera istniała, funkcja musi odpowiednio szybko zanikać w $\pm\infty$.

Własności transformaty Fouriera

Splot:

$$(g \star h)(t) = \int_{-\infty}^{\infty} g(\tau)h(t-\tau)\,d\tau \tag{3a}$$

$$g \star h \leftrightarrow G(f)H(f) \tag{3b}$$

Funkcja korelacji:

$$\text{Corr}(g,h) = \int_{-\infty}^{\infty} g(\tau+t)h(t)\,d\tau \tag{4a}$$

$$\text{Corr}(g,h) \leftrightarrow G(f)H^*(f) \tag{4b}$$

Twierdzenie Wienera-Chinczyna*

Funkcja autokorelacji:

$$\text{Corr}(g, g) \leftrightarrow |G(f)|^2 \tag{5}$$

Równość Parsevala

$$\int_{-\infty}^{\infty} |g(t)|^2 \, dt = \int_{-\infty}^{\infty} |G(f)|^2 \, df = \text{całkowita moc} \tag{6}$$

*Ang. Wiener-Khinchin Theorem.

Próbkowanie dyskretne

W praktyce nigdy nie mamy do czynienia z sygnałami ciągłymi, a tylko z szeregami czasowymi. Załóżmy, że mamy szereg *próbkowany* ze stałym krokiem Δ:

$$g_n = g(n\Delta)\,, \quad n = \ldots, -3, -2, -1, 0, 1, 2, 3, \ldots \tag{7}$$

Zauważmy, że samo próbkowanie wprowadza do układu pewną charakterystyczną częstotliwość, zwaną *częstotliwością Nyquista*:

$$f_{\text{Nyq}} = \frac{1}{2\Delta}\,. \tag{8}$$

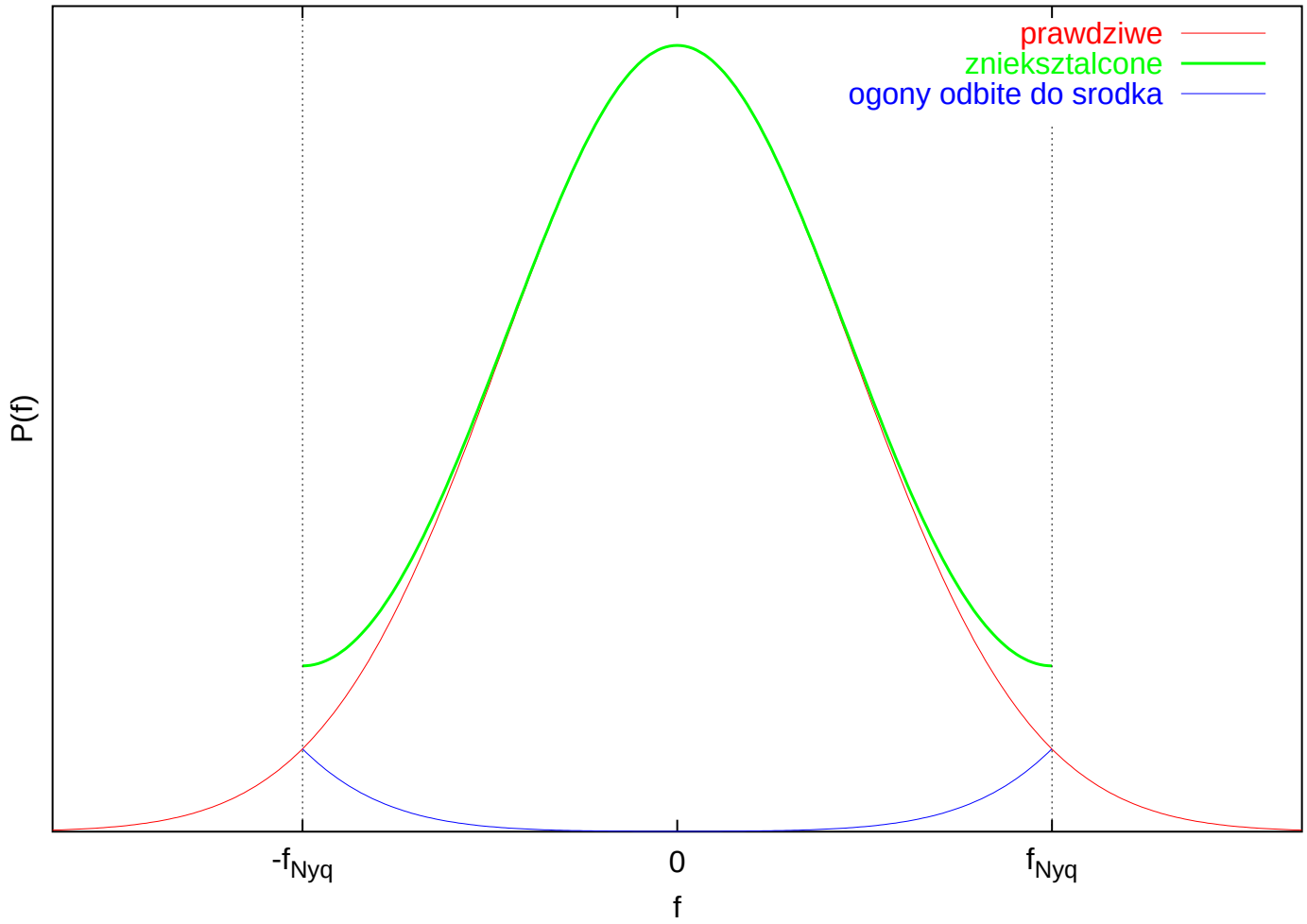
Jaki jest jej sens? Aby prawidłowo rozpoznać okres fali harmonicznej, trzeba ją spróbkować co najmniej dwa razy na okres. Jeśli zatem próbkujemy z krokiem Δ, możemy zaobserwować tylko częstotliwości $f \in [-f_{\text{Nyq}}, f_{\text{Nyq}}]$.

Aliasing

A co jeśli sygnał zawiera częstotliwości $f \notin [-f_{\text{Nyq}}, f_{\text{Nyq}}]$? Czy zostają one utracone? To byłoby pół biedy: Częstości spoza przedziału Nyquista, jeśli są obecne w sygnale, zafałszowują obserwowany rozkład częstości wewnątrz tego przedziału: „Ogony” zostają odbite do środka. Dlatego też jeśli podejrzewamy, że krok próbkowania jest za duży i nie można go zmniejszyć, należy odfiltrować częstości spoza przedziału Nyquista przed dalszą obróbką.

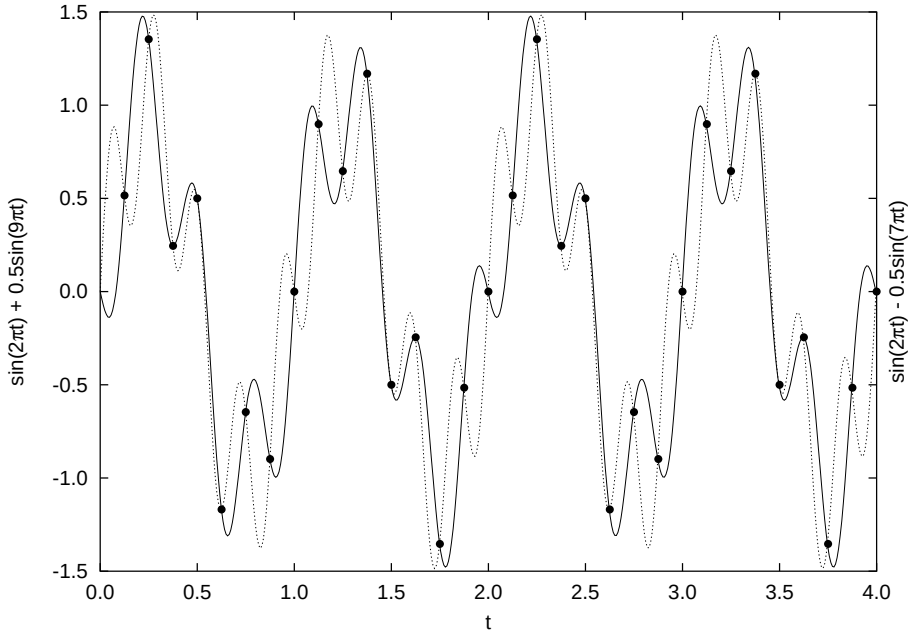
Praktycznym testem na nieobecność aliasingu jest to, że widmo mocy[†] zanika na granicach przedziału Nyquista.

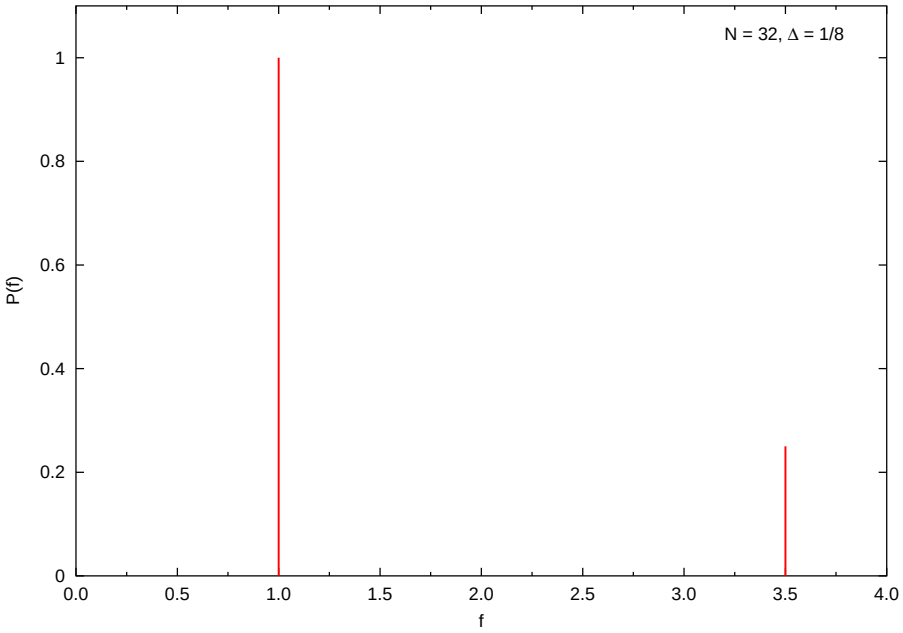
[†]Wybiegając przed orkiestrę.


prawdziwe
zniekształcone
ogony odbite do srodka
P(f)
-f_Nyq
0
f_Nyq
f

Formalne źródło aliasingu:

Jeśli Δ jest krokiem próbkowania, dwie fale o częstotliwościach f_1, f_2 dają takie same próbki, jeśli $f_1 - f_2 = k/\Delta$: $\exp(2\pi i f_1 n\Delta) = \exp(2\pi i(f_2 + k/\Delta)n\Delta) = \exp(2\pi i f_2 n\Delta + 2\pi i k n) = \exp(2\pi i f_2 n\Delta)$.





Twierdzenie Shannona[‡] o próbkowaniu[§]:

Jeżeli funkcja jest *pasmowo ograniczona* (ang. *bandwidth limited, band-limited*), to znaczy jeśli zawiera tylko częstotliwości z przedziału $[-f_{\text{Nyq}}, f_{\text{Nyq}}]$, i jeśli dany jest nieskończony ciąg próbek z krokiem Δ, to

$$\forall t: \quad g(t) = \Delta \sum_{n=-\infty}^{\infty} g_n \frac{\sin\left(2\pi f_{\text{Nyq}}(t - n\Delta)\right)}{\pi(t - n\Delta)}. \tag{9}$$

To bardzo mocne twierdzenie: Pozwala odtworzyć wartości funkcji pasmowo ograniczonej w nieprzeliczalnie wielu punktach na podstawie jej znajomości w dyskretnych punktach. Szumy (procesy stochastyczne) i funkcje nieciągłe nie są pasmowo ograniczone.

[‡]Zwane także *Twierdzeniem Shannona-Kotelnikowa*.

[§]Ang. *Shannon sampling theorem*.

Skończone szeregi czasowe

W praktyce nigdy nie mamy nieskończonego szeregu czasowego — dysponujemy zaledwie skończonym ciągiem o długości N. Co zrobić z transformatą Fouriera, a w szczególności jak zapewnić jej zbieżność? Stosowane są dwie konwencje:

- Zakładamy, że szereg czasowy zanika do zera przy końcach — a jeśli nie zanika, obkładamy go odpowiednią funkcją okna zapewniającą to zanikanie.

- Zakładamy, że dostępny ciąg skończony jest okresem podstawowym nieskończonego ciągu okresowego. Jest to motywowane tym, iż w tej konwencji czyste przebiegi harmoniczne *mają* transformaty (ciągła transformata Fouriera funkcji sinus istnieje tylko w sensie dystrybucyjnym).

My przyjmujemy tę drugą konwencję. Zgodnie z nią udajemy, że badamy szereg postaci

$$\ldots, \underbrace{g_0, g_1, g_2, \ldots, g_{N-1}}_{\text{kopia } -1}, \underbrace{g_0, g_1, g_2, \ldots, g_{N-1}}_{\text{prawdziwe dane}}, \underbrace{g_0, g_1, g_2, \ldots, g_{N-1}}_{\text{kopia } 1}, \ldots \tag{10}$$

Ponieważ mamy N próbek wejściowych, także dyskretną transformatę Fouriera (DFT, ang. *Discrete Fourier Transform*) możemy obliczyć tylko w N punktach. Dla ustalenia uwagi niech N będzie parzyste. Przyjmujemy, że DFT obliczać będziemy dla częstotliwości

$$f_n = \frac{n}{N\Delta}, \quad n = -\frac{N}{2}, \ldots, \frac{N}{2}. \tag{11}$$

$$G(f_n) = \int_{-\infty}^{\infty} g(t)\, e^{2\pi i f_n t}\, dt$$

$$= \lim_{M\to\infty} \frac{1}{2M+1} \sum_{s=-M}^{M} \int_{(s-1)(N-1)\Delta}^{s(N-1)\Delta} g(t)\, e^{2\pi i f_n t}\, dt = \int_{0}^{(N-1)\Delta} g(t) e^{2\pi i f_n t} dt \tag{12}$$

Całkę w (12) przybliżam w najprostszy możliwy sposób, przybliżając funkcję podcałkową funkcją schodkową:

$$G(f_n) \simeq \sum_{k=0}^{N-1} g_k\, e^{2\pi i f_n t_k} \Delta \simeq \Delta \sum_{k=0}^{N-1} g_k\, e^{2\pi i k n/N}\,. \tag{13}$$

Liczby

$$G_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} g_k \, e^{2\pi i k n/N} \tag{14}$$

nazywam *dyskretnymi składowymi fourierowskimi* funkcji g. W tej konwencji transformata odwrotna ma postać

$$g_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} G_n \, e^{-2\pi i k n/N} \, . \tag{15}$$

Dyskretna wersja tożsamości Parsevala ma postać

$$\sum_{k=0}^{N-1} |g_k|^2 = \sum_{n=0}^{N-1} |G_n|^2 \, . \tag{16}$$

DFT jako operacja liniowa

Zauważmy, że wzór (14) można zapisać w postaci

$$G_n = \sum_{k=0}^{N-1} W_{nk}\, g_k\,, \tag{17a}$$

$$W_{nk} = \frac{1}{\sqrt{N}} e^{2\pi i k n/N}\,. \tag{17b}$$

Liczby W_{nk} można zinterpretować jako elementy pewnej macierzy. Wobec tego zwartą postacią (17) jest

$$\mathbf{G} = \mathbf{W}\mathbf{g}\,, \tag{18}$$

gdzie $\mathbf{G}, \mathbf{g} \in \mathbb{C}^N$, $\mathbf{W} \in \mathbb{C}^{N\times N}$. Wydaje się, że koszt numeryczny obliczania DFT wynosi $O(N^2)$, czyli tyle, ile wynosi koszt mnożenia wektora przez macierz.

Własności macierzy $\mathbf{W}$

$$\left(\mathbf{W}\mathbf{W}^{\dagger}\right)_{ls} = \sum_{k=0}^{N-1} \mathbf{W}_{lk} \left(\mathbf{W}^{\dagger}\right)_{ks} = \sum_{k=0}^{N-1} \mathbf{W}_{lk} \left(\mathbf{W}_{sk}\right)^{*} = \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi i (l-s)k/N} \tag{19}$$

Jeśli $l = s$, wszystkie wyrazy pod sumą są równe 1 i wynik wynosi 1. Jeśli $l - s = m \neq 0$,

$$\left(\mathbf{W}\mathbf{W}^{\dagger}\right)_{ls} = \frac{1}{N} \sum_{k=0}^{N-1} \left(e^{2\pi i m/N}\right)^{k} = \frac{1 - \left(e^{2\pi i m/N}\right)^{N}}{N(1 - e^{2\pi i m/N})} = \frac{1 - e^{2m\pi i}}{N(1 - e^{2\pi i m/N})} = 0\,. \tag{20}$$

$$\left(\mathbf{W}\mathbf{W}^{\dagger}\right)_{ls} = \delta_{ls}.$$

Widzimy, że macierz $\mathbf{W}$ jest *unitarna*. (Uwaga na przyjętą normalizację!)

Podsumowanie

DFT jest (z dokładnością do normalizacji) unitarnym przekształceniem *wektora próbek* (sygnału) w *wektor składowych fourierowskich*.

DFT można zatem traktować jako przedstawienie wektora próbek w innej bazie, mianowicie w bazie rozpiętej przez sinusy i kosinusy o częstotliwościach

$$0,\ 1/(N\Delta),\ 2/(N\Delta),\ \ldots,\ 1/(2\Delta).$$

Alternatywnie — w bazie funkcji

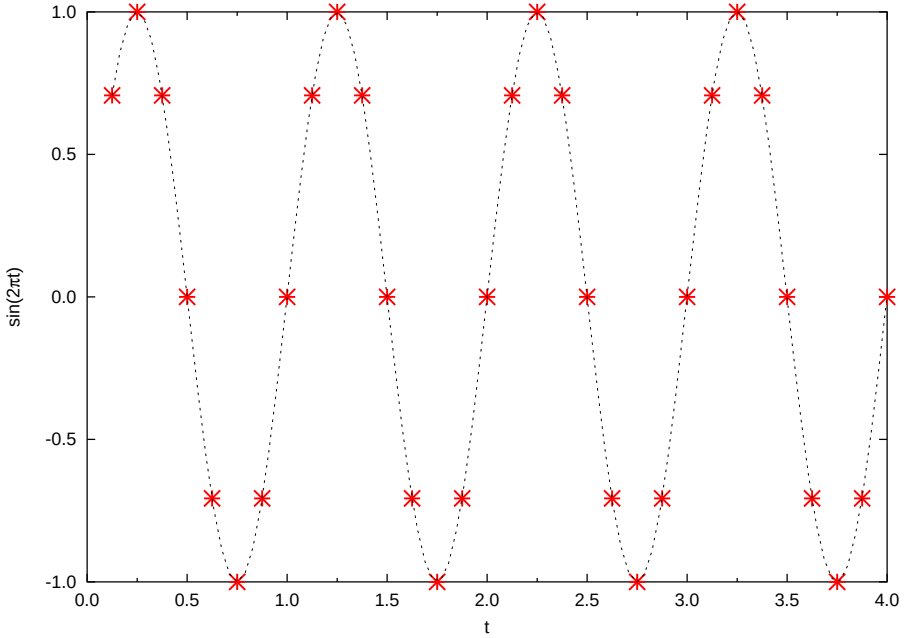
$$\exp\left(-\frac{2\pi i}{2\Delta}t\right),\ \ldots,\ \exp\left(-\frac{2\pi i}{N\Delta}t\right),1,\exp\left(\frac{2\pi i}{N\Delta}t\right),\ldots,\ \exp\left(\frac{2\pi i}{2\Delta}t\right)$$

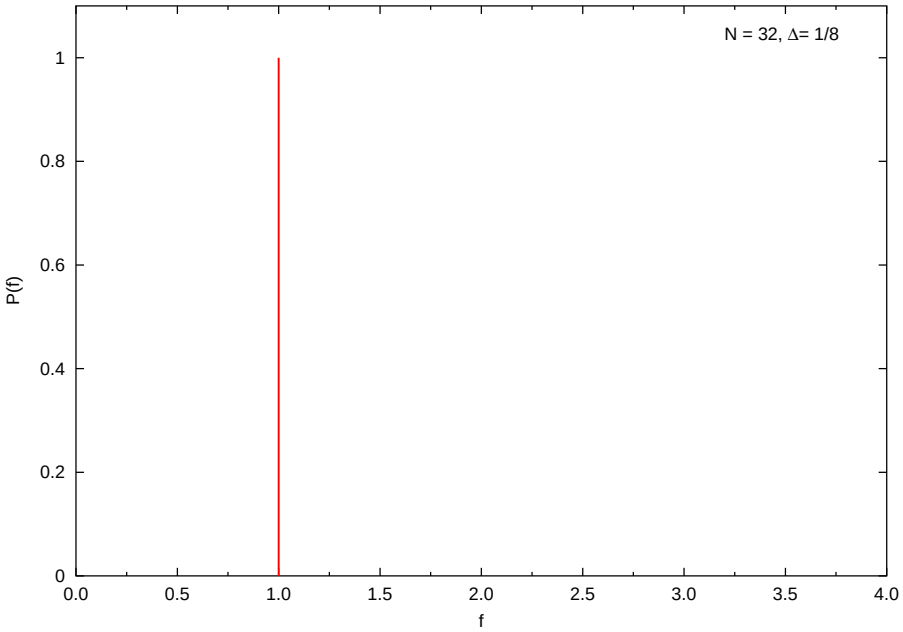
Dzięki symetriom macierzy $\mathbf{W}$*, koszt obliczania DFT można* znacznie *zredukować (algorytm FFT).*

Uwagi

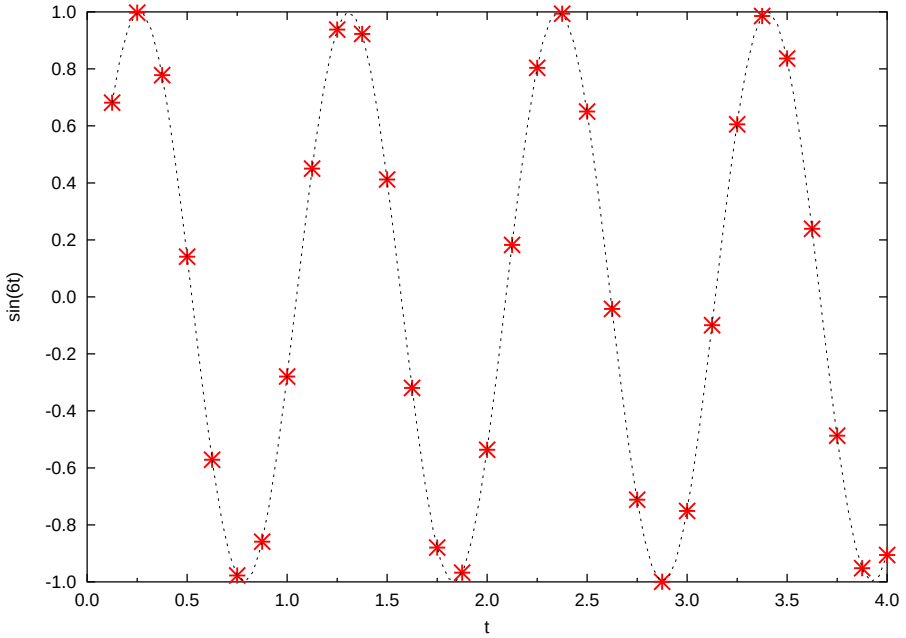
- Dyskretne współczynniki fourierowskie są okresowe, w szczególności $G_{-N/2} = G_{N/2}$. Jest to konsekwencja założenia o okresowości sygnału.
- Ujemne częstotliwości pojawiają się dlatego, że funkcje sinus i kosinus o tym samym okresie traktujemy jako niezależne.
- Niekiedy rozważa się transformaty rozpięte na samych sinusach (transformata sinusowa) lub na samych kosinusach (transformata kosinusowa).
- „Sygnał” może być w ogólności zespolony. Jeżeli sygnał jest rzeczywisty, można jego transformatę obliczyć szybciej robiąc sygnał zespolony o połowie długości. Podobnie można jednocześnie liczyć transformaty dwu sygnałów rzeczywistych. Transformaty rozwikłuje się korzystając z własności symetrii DFT.

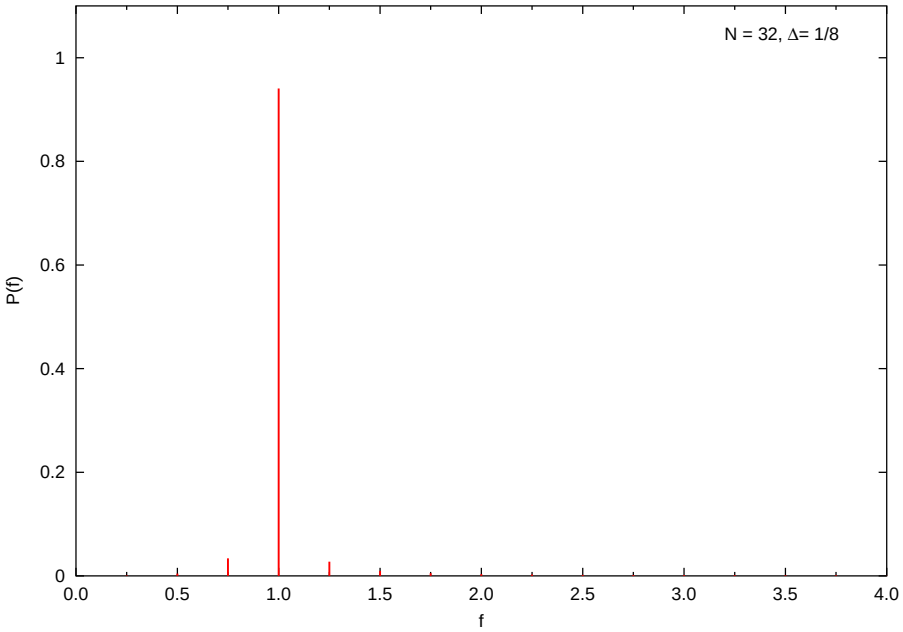
Transformata sygnału „zgranego” z funkcjami bazowymi



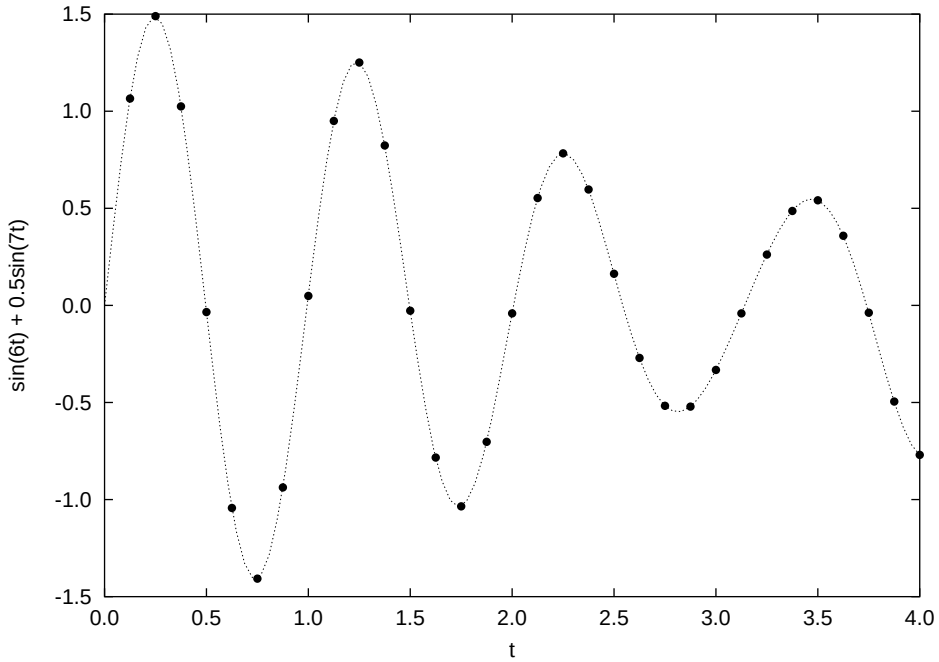


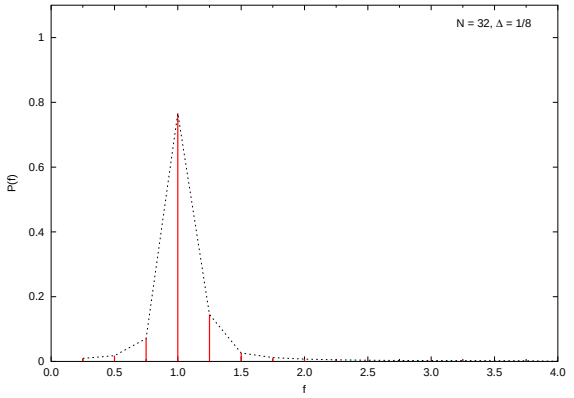
Transformata sygnału „niezgranego” z funkcjami bazowymi

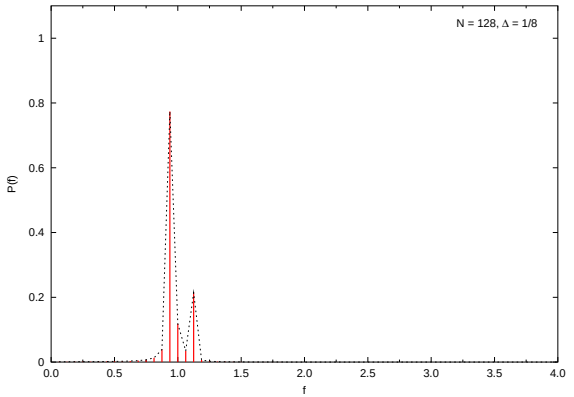




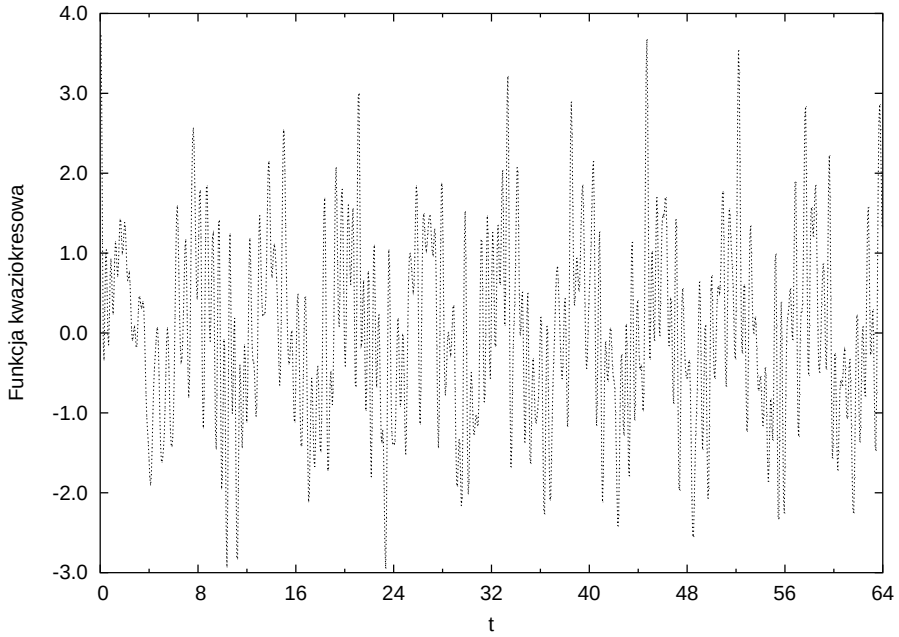
Zwiększenie ilości próbek przy tym samym kroku próbkowania poprawia rozdzielczość transformaty

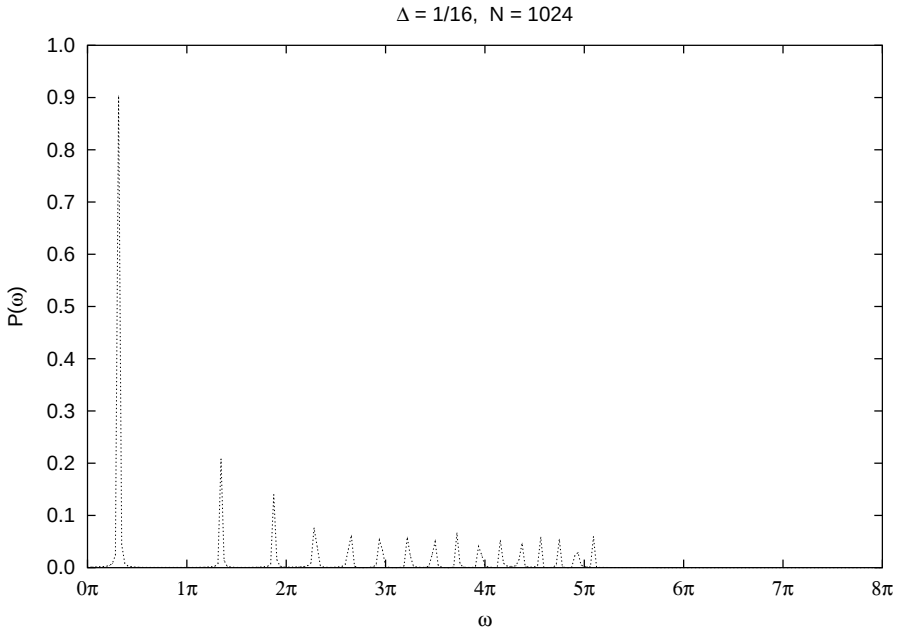




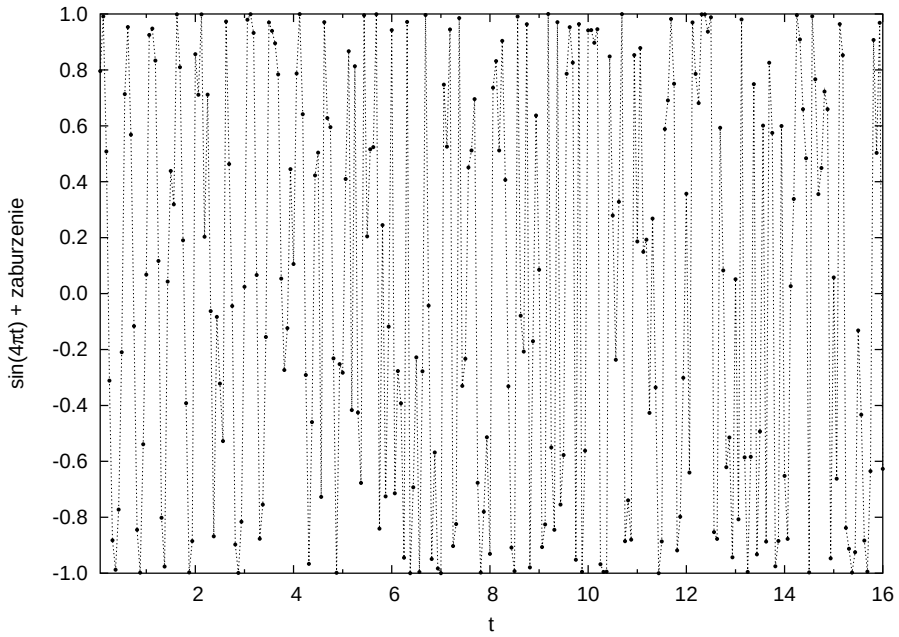


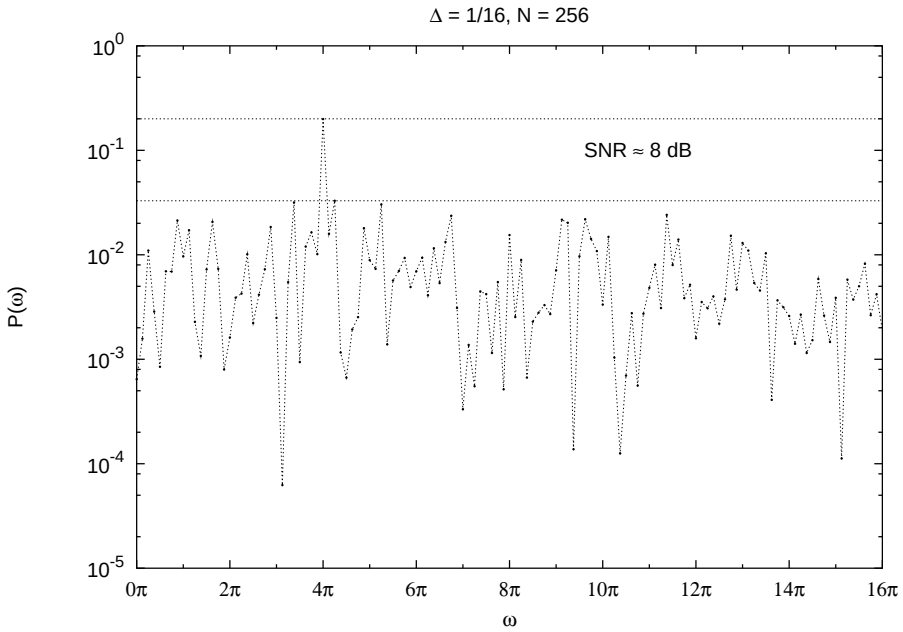
Funkcja kwaziokresowa





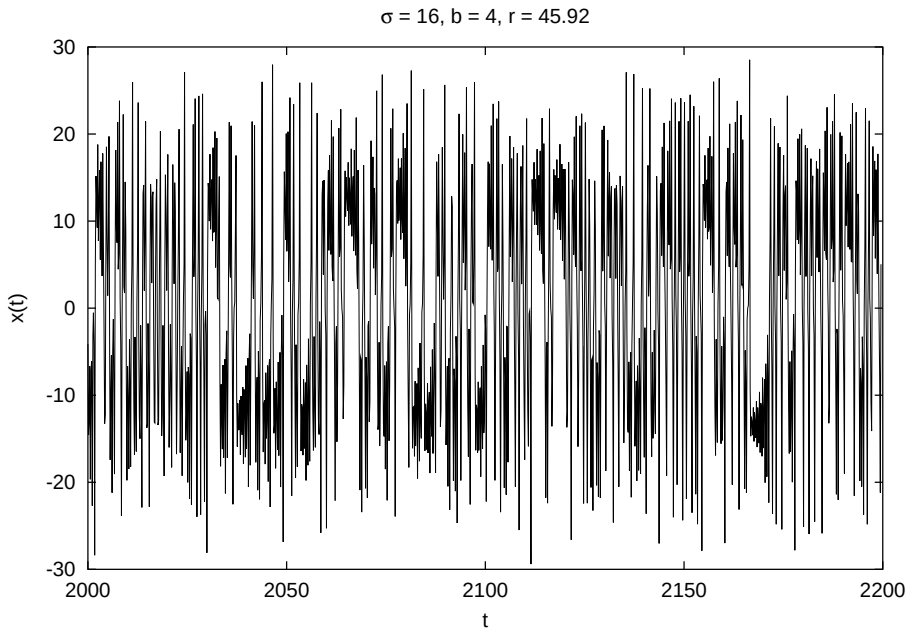
Funkcja okresowa z zaburzeniem sprzężonym nieliniowo:

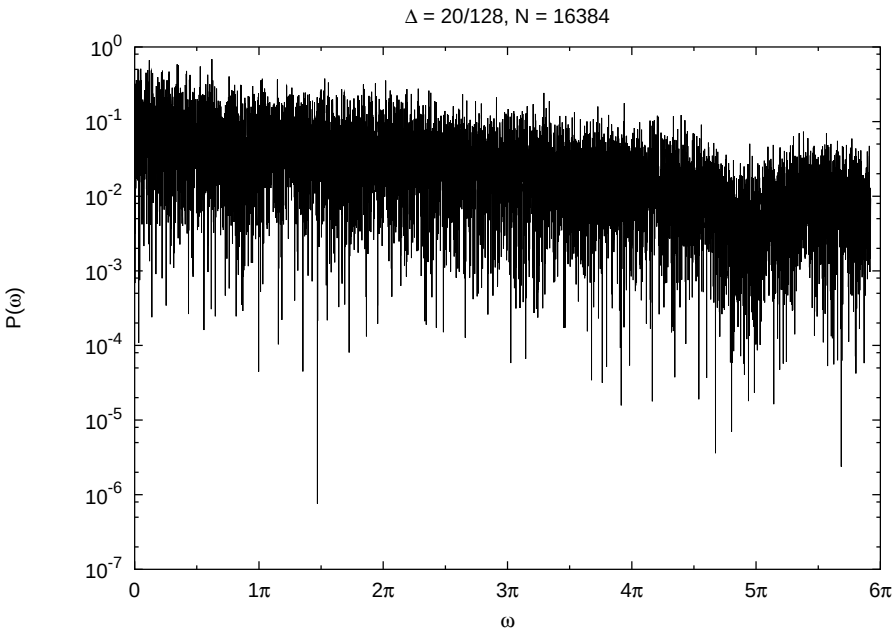




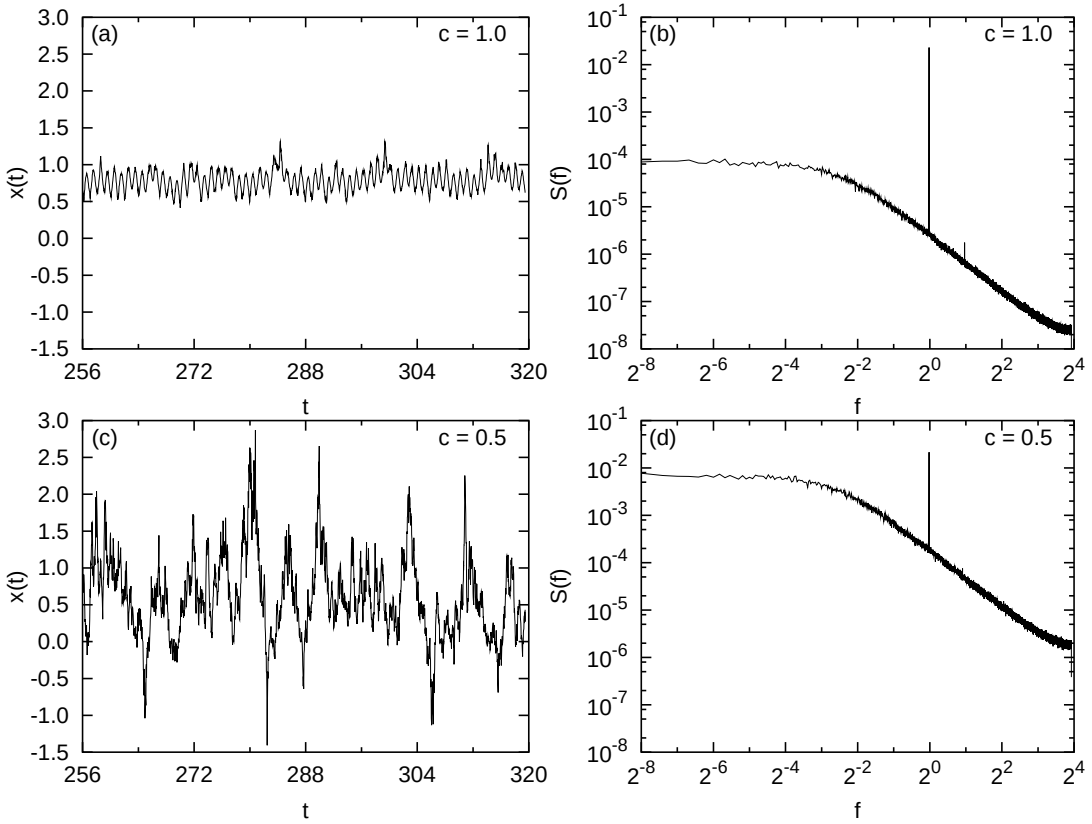
Sygnał nieliniowy — układ Lorentza

$$\dot{x} = \sigma(y - x)\,, \quad \dot{y} = -xz + rx - y\,, \quad \dot{z} = xy - bz$$





Kolejny przykład



Uwaga terminologiczna: macierz Vandermonde'a

Macierz realizująca DFT jest szczególnym przypadkiem macierzy Vandermonde'a $\mathbf{W} = \frac{1}{\sqrt{N}}\mathbf{V} \in \mathbb{C}^{N\times N}$, gdzie

$$\mathbf{V} = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ z_0 & z_1 & \cdots & z_{N-2} & z_{N-1} \\ z_0^2 & z_1^2 & \cdots & z_{N-2}^2 & z_{N-1}^2 \\ \vdots & \vdots & & \vdots & \vdots \\ z_0^{N-1} & z_1^{N-1} & \cdots & z_{N-2}^{N-1} & z_{N-1}^{N-1} \end{bmatrix} \tag{21}$$

W przypadku DFT, $z_k = \exp(2\pi ik/N)$.

Algorytm Fast Fourier Transform, FFT

Przykład: $N = 8$

$$\mathbf{G} = \frac{\mathbf{V}^{(8)}}{\sqrt{8}}\mathbf{g} = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \frac{1+i}{\sqrt{2}} & i & \frac{-1+i}{\sqrt{2}} & -1 & \frac{-1-i}{\sqrt{2}} & -i & \frac{1-i}{\sqrt{2}} \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & \frac{-1+i}{\sqrt{2}} & -i & \frac{1+i}{\sqrt{2}} & -1 & \frac{1-i}{\sqrt{2}} & i & \frac{-1-i}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & \frac{-1-i}{\sqrt{2}} & i & \frac{1-i}{\sqrt{2}} & -1 & \frac{1+i}{\sqrt{2}} & -i & \frac{-1+i}{\sqrt{2}} \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & \frac{1-i}{\sqrt{2}} & -i & \frac{-1-i}{\sqrt{2}} & -1 & \frac{-1+i}{\sqrt{2}} & i & \frac{1+i}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \end{bmatrix} \tag{22}$$

Widać *regularności*, ale trudno zobaczyć *symetrie*...

Permutujemy kolumny, *jednocześnie* permutując wektor danych — to *nie zmienia* wyniku.

$$\mathbf{G} = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i & \frac{1+i}{\sqrt{2}} & \frac{-1+i}{\sqrt{2}} & \frac{-1-i}{\sqrt{2}} & \frac{1-i}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & i & -i & i & -i \\ 1 & -i & -1 & i & \frac{-1+i}{\sqrt{2}} & \frac{1+i}{\sqrt{2}} & \frac{1-i}{\sqrt{2}} & \frac{-1-i}{\sqrt{2}} \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & i & -1 & -i & \frac{-1-i}{\sqrt{2}} & \frac{1-i}{\sqrt{2}} & \frac{1+i}{\sqrt{2}} & \frac{-1+i}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & -i & i & -i & i \\ 1 & -i & -1 & i & \frac{1-i}{\sqrt{2}} & \frac{-1-i}{\sqrt{2}} & \frac{-1+i}{\sqrt{2}} & \frac{1+i}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} g_0 \\ g_2 \\ g_4 \\ g_6 \\ g_1 \\ g_3 \\ g_5 \\ g_7 \end{bmatrix} \tag{23}$$

Dzięki zmianie kolejności zaczyna być coś widać!

Równanie (23) możemy przepisać w postaci (zapis blokowy)

$$\mathbf{G} = \frac{1}{\sqrt{8}} \begin{bmatrix} \mathbf{V}^{(4)} & \mathbf{\Omega}^{(4)}\mathbf{V}^{(4)} \\ \mathbf{V}^{(4)} & -\mathbf{\Omega}^{(4)}\mathbf{V}^{(4)} \end{bmatrix} \begin{bmatrix} g_0 \\ g_2 \\ g_4 \\ g_6 \\ g_1 \\ g_3 \\ g_5 \\ g_7 \end{bmatrix} \tag{24}$$

. . . gdzie oznaczyliśmy

$$\mathbf{V}^{(4)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}, \tag{25}$$

$$\mathbf{\Omega}^{(4)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1+i}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & \frac{-1+i}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \left(\frac{1+i}{\sqrt{2}}\right)^0 & & & \\ & \left(\frac{1+i}{\sqrt{2}}\right)^1 & & \\ & & \left(\frac{1+i}{\sqrt{2}}\right)^2 & \\ & & & \left(\frac{1+i}{\sqrt{2}}\right)^3 \end{bmatrix} \tag{26}$$

A zatem

$$\mathbf{G} = \frac{1}{\sqrt{8}} \begin{bmatrix} \mathbf{V}^{(4)} \begin{bmatrix} g_0 \\ g_2 \\ g_4 \\ g_6 \end{bmatrix} + \mathbf{\Omega}^{(4)} \mathbf{V}^{(4)} \begin{bmatrix} g_1 \\ g_3 \\ g_5 \\ g_7 \end{bmatrix} \\ \mathbf{V}^{(4)} \begin{bmatrix} g_0 \\ g_2 \\ g_4 \\ g_6 \end{bmatrix} - \mathbf{\Omega}^{(4)} \mathbf{V}^{(4)} \begin{bmatrix} g_1 \\ g_3 \\ g_5 \\ g_7 \end{bmatrix} \end{bmatrix} \tag{27}$$

Fragmenty oznaczone kolorami obliczamy tylko raz. Mnożenie przez $\mathbf{\Omega}^{(4)}$ wykonujemy w czasie liniowym. Zmniejszyliśmy czas obliczeń *o połowę*.

Rzecz w tym, iż $\mathbf{V}^{(4)}$ podlega analogicznej faktoryzacji (zob. (25)), przy jednoczesnej dalszej permutacji wektora wejściowego:

$$\mathbf{V}^{(4)} \begin{bmatrix} g_0 \\ g_2 \\ g_4 \\ g_6 \end{bmatrix} = \begin{bmatrix} \mathbf{V}^{(2)} & \mathbf{\Omega}^{(2)}\mathbf{V}^{(2)} \\ \mathbf{V}^{(2)} & -\mathbf{\Omega}^{(2)}\mathbf{V}^{(2)} \end{bmatrix} \begin{bmatrix} g_0 \\ g_4 \\ g_2 \\ g_6 \end{bmatrix} \tag{28}$$

$$\mathbf{V}^{(2)} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathbf{\Omega}^{(2)} = \begin{bmatrix} 1 & \\ & i \end{bmatrix} = \begin{bmatrix} i^0 & \\ & i^1 \end{bmatrix}$$

(analogicznie dla $[g_1, g_3, g_5, g_7]^T$)

W ten sposób

$$\mathbf{G} = \frac{1}{\sqrt{8}} \begin{bmatrix} \begin{bmatrix} \mathbf{V}^{(2)} \begin{bmatrix} g_0 \\ g_4 \end{bmatrix} + \boldsymbol{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_2 \\ g_6 \end{bmatrix} \\ \mathbf{V}^{(2)} \begin{bmatrix} g_0 \\ g_4 \end{bmatrix} - \boldsymbol{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_2 \\ g_6 \end{bmatrix} \end{bmatrix} + \boldsymbol{\Omega}^{(4)} \begin{bmatrix} \mathbf{V}^{(2)} \begin{bmatrix} g_1 \\ g_5 \end{bmatrix} + \boldsymbol{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_3 \\ g_7 \end{bmatrix} \\ \mathbf{V}^{(2)} \begin{bmatrix} g_1 \\ g_5 \end{bmatrix} - \boldsymbol{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_3 \\ g_7 \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \mathbf{V}^{(2)} \begin{bmatrix} g_0 \\ g_4 \end{bmatrix} + \boldsymbol{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_2 \\ g_6 \end{bmatrix} \\ \mathbf{V}^{(2)} \begin{bmatrix} g_0 \\ g_4 \end{bmatrix} - \boldsymbol{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_2 \\ g_6 \end{bmatrix} \end{bmatrix} - \boldsymbol{\Omega}^{(4)} \begin{bmatrix} \mathbf{V}^{(2)} \begin{bmatrix} g_1 \\ g_5 \end{bmatrix} + \boldsymbol{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_3 \\ g_7 \end{bmatrix} \\ \mathbf{V}^{(2)} \begin{bmatrix} g_1 \\ g_5 \end{bmatrix} - \boldsymbol{\Omega}^{(2)} \mathbf{V}^{(2)} \begin{bmatrix} g_3 \\ g_7 \end{bmatrix} \end{bmatrix} \end{bmatrix} \tag{29}$$

Dwuwymiarowe wektory oznaczone kolorami oblicza się tylko raz. W ten sposób złożoność obliczeniową zredukowaliśmy już *czterokrotnie*.

Ostatnia zagadka: W porównaniu z wyjściowym równaniem (22), w równaniu (29) dokonaliśmy permutacji
$[g_0, g_1, g_2, g_3, g_4, g_5, g_6, g_7]^T \rightarrow [g_0, g_4, g_2, g_6, g_1, g_5, g_3, g_7]^T$.
Co to za permutacja?

$$\begin{array}{c} 0 \\ 4 \\ 2 \\ 6 \\ 1 \\ 5 \\ 3 \\ 7 \end{array} = \begin{array}{c} 000_2 \\ 100_2 \\ 010_2 \\ 110_2 \\ 001_2 \\ 101_2 \\ 011_2 \\ 111_2 \end{array} \xrightarrow[\text{odwracam kolejność}]{} \begin{array}{c} 000_2 \\ 001_2 \\ 010_2 \\ 011_2 \\ 100_2 \\ 101_2 \\ 110_2 \\ 111_2 \end{array} = \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \tag{30}$$

Wektor wejściowy (“wektor próbek”) uporządkowany jest *w odwrotnej kolejności bitowej* (*bit reversal order*) indeksów.

Ten algorytm bardzo łatwo uogólnia się na dowolne $N = 2^s$, $s \in \mathbb{N}$.

Widać też, że znacznie redukujemy konieczność obliczania złożonych wyrażeń typu $\sin\left(\frac{n\pi}{N}\right)$, $\cos\left(\frac{n\pi}{N}\right)$ — konieczność wywoływania funkcji bibliotecznych $\sin(\cdot)$, $\cos(\cdot)$ została w ogóle wyeliminowana. Na każdym etapie faktoryzacji wystarczy *raz* wyliczyć odpowiedni pierwiastek z i, na co są gotowe wzory, potem zaś wyliczać tylko kolejne potęgi.

Jaki jest koszt numeryczny tego algorytmu?

Każda faktoryzacja zmniejsza ilość operacji o połowę.

Faktoryzacji można zrobić tyle, ile razy można podzielić macierz “na ćwiartki”.

Koszt numeryczny algorytmu FFT wynosi

$$O(N \log_2 N)$$

Na przykład dla $N = 65536 = 2^{16}$ zastosowanie algorytmu FFT zmniejsza złożoność obliczeniową w stosunku do liczenia “z definicji” ponad *cztery tysiące razy*.

Uwagi

- Zupełnie podobny algorytm można skonstruować dla $N = 3^s$, $N = 5^s$ i, ogólnie, dla $N = q^s$, gdzie q jest liczbą pierwszą. Koszt obliczeniowy wynosi wówczas $O(N \log_q N)$.
- Dobre biblioteki "szybko" obsługują także $N = q_1^{s_1} q_2^{s_2} \cdots q_m^{s_m}$, gdzie $q_1, q_2, \ldots, q_m$ są niskimi liczbami pierwszymi.
- Jeżeli analizujemy ciąg uzyskany z symulacji, to w zasadzie kontrolujemy jego długość i *koniecznie* należy zadbać o to, aby była ona potęgą niskiej liczby pierwszej, ewentualnie iloczynem jak najmniejszej ilości czynników typu $q_i^{s_j}$. Jeżeli mamy ciąg "doświadczalny", dla zapewnienia odpowiedniej szybkości obliczeń należy go albo obciąć, albo *uzupełnić zerami* do najbliższej "specjalnej" długości.

Inne uwagi

- DFT traktuje ciąg próbek jak ciąg zmiennych zespolonych. Jeśli liczymy transformatę sygnału rzeczywistego, można go przekształcić za sygnał "zespolony" o długości *dwa razy mniejszej*. Po dokonaniu transformacji, odwikłujemy transformatę korzystając z własności symetrii transformaty Fouriera: Jeżeli $\mathbf{g} \in \mathbb{R}^N$, $\mathbf{G}^*(f) = \mathbf{G}(-f)$. Daje to istotne przyspieszenie obliczeń.
- Podobnie można jednocześnie liczyć transormaty dwu sygnałów rzeczywistych — *bardzo przydatne* przy liczeniu splotu dwu takich sygnałów.
- Istnieją także "szybkie" algorytmy rozkładające sygnał wejściowy w bazie samych kosinusów (szybka transformata kosinusowa) lub samych sinusów (szybka transformata sinusowa). Transformat tych można z sensem używać jeśli sygnały wejściowe mają odpowiednie symetrie. Szybkiej dwuwymiarowej transformacji kosinusowej używa się w standardzie JPEG.

Jak zrobić transformację dwuwymiarową?

Bardzo prosto: Jeżeli $\mathbf{g} \in \mathbb{C}^{N \times N}$,

$$\mathbf{G} = \mathbf{W}\mathbf{g}\mathbf{W}^{\dagger} \tag{31}$$

Oczywiście jest na to odpowiedni “szybki” algorytm, z odpowiednim przyspieszeniem dla $\mathbf{g} \in \mathbb{R}^{N \times N}$.