Szybka transformacja Fouriera
(FFT – Fast Fourier Transform)

Plan wykładu:

1. Transformacja Fouriera, iloczyn skalarny

2. DFT - dyskretna transformacja Fouriera

3. FFT – szybka transformacja Fouriera

   a) algorytm Cooleya-Tukeya (radix-2)
   b) algorytm PFA
   c) wielowymiarowe FFT

4. Przykłady zastosowań:
   - mnożenie wielomianów,
   - odszumianie sygnału,
   - rozwiązanie równania Poissona,
   - całkowanie

- jeśli funkcja f(x) jest okresowa wówczas zamiast wielomianów do jej **interpolacji** (**aproksymacji**) lepiej użyć wielomianów trygonometrycznych tj. rozwinąć funkcję w **szereg Fourieral**

- dla funkcji okresowej o okresie $x_{max}=2\pi$:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos(kx) + b_k \sin(kx))$$

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(kx)dx \qquad b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(kx)dx$$

Jak liczymy czynnik normalizacyjny?

$$\int_{-\pi}^{\pi} \sin^2 x\, dx = \int_{-\pi}^{\pi} \cos^2 x\, dx = \frac{2\pi}{2} = \pi$$

- funkcję możemy też zapisać w postaci **zespolonego szeregu Fouriera**

$$E_k(x) = e^{Ikx} = \cos(kx) + I\ \sin(kx) \qquad I = \sqrt{-1}$$

$$f(x) \sim \sum_{k=-\infty}^{\infty} \widehat{f}(k) e^{Ikx}$$

$$\widehat{f}(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x) e^{-Ikx} dx$$

Jak liczymy czynnik normalizacyjny?

$$\int_{-\pi}^{\pi} |e^{Ikx}|^2\, dx = \int_{-\pi}^{\pi} 1\, dx = 2\pi$$

- jeśli funkcja f(x) jest rzeczywista wówczas „zwykły” szereg Fouriera jest częścią rzeczywistą zespolonego szeregu Fouriera

$$\widehat{f}(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)\left[\cos(kx) - I\sin(kx)\right]dx = \frac{1}{2}(a_k - Ib_k), \qquad k \geq 0$$

- dla ciągu współczynników $[a_k]_{k=0}^{\infty}$ $[b_k]_{k=1}^{\infty}$

definiujemy

$$b_0 = 0$$
$$a_{-k} = a_k \qquad c_k = \frac{1}{2}(a_k - Ib_k)$$
$$b_{-k} = -b_k$$

co prowadzi do zależności pomiędzy szeregiem rzeczywistym i zespolonym

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{n}\left[a_k\cos(kx) + b_k\sin(kx)\right] = \sum_{k=-n}^{n} c_k e^{Ikx}$$

- dla sygnału rzeczywistego współczynniki a_k lub b_k mogą być wszystkie równe 0

Funkcje

$$E_k(x) = e^{Ikx}, \quad k = 0, \pm 1, \pm 2, \ldots$$

generują ciąg ortonormalnych funkcji w zespolonej przestrzeni Hilberta.
Iloczyn skalarny w tej przestrzeni:

$$\langle f, g \rangle = \frac{1}{2\pi} \int_{-\pi}^{\pi} f^*(x) g(x) dx$$

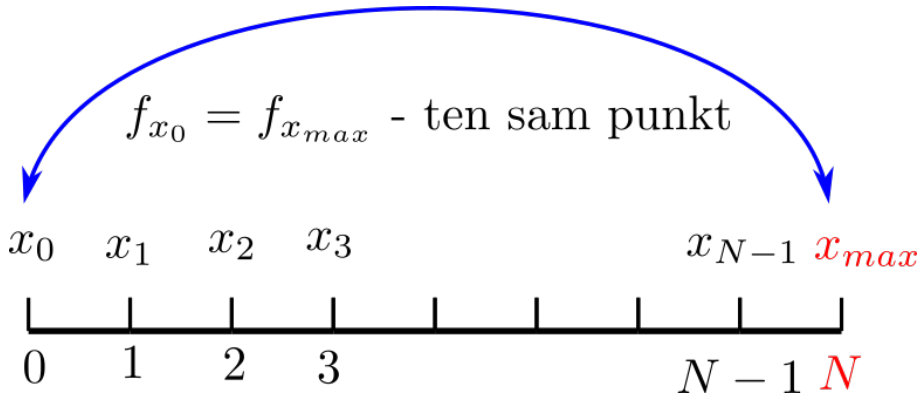
$$\begin{aligned}
\langle E_k, E_n \rangle &= \frac{1}{2\pi} \int_{-\pi}^{\pi} E_k^*(x) E_n(x) dx \\
&= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-Ikx} e^{Inx} dx \\
&= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{I(n-k)x} dx \\
&= \frac{1}{2\pi} \frac{e^{I(n-k)x}}{I(n-k)} \Big|_{x=-\pi}^{x=\pi} = 0, \qquad \Longleftrightarrow \; n \neq k
\end{aligned}$$

Funkcje En stanowią bazę ortogonalną.

- dysponując tablicą wartości funkcji f i g w węzłach siatki, iloczyn wewnętrzny (skalarny) można zapisać w postaci dyskretnej:

$$\Delta x = \frac{x_{max}}{N}$$

$$x_j = \Delta x \cdot j, \quad j = 0, 1, 2, \ldots, N$$



$$\begin{aligned}
\langle f, g \rangle &= \sum_{j=0}^{N-1} f^*(x_j) g(x_j) \Delta x \\
&= \sum_{j=0}^{N-1} f^*(x_j) g(x_j) \frac{x_{max}}{N} \\
&= x_{max} \left[\frac{1}{N} \sum_{j=0}^{N-1} f^* \left(j \frac{x_{max}}{N} \right) g \left(j \frac{x_{max}}{N} \right) \right] \\
&= x_{max} \left[\frac{1}{N} \sum_{j=0}^{N-1} f_j^* g_j \right]
\end{aligned}$$

$$\frac{\langle f, g \rangle}{x_{max}} = \langle f, g \rangle_N$$

- funkcje eksponencjalne mają być periodyczne w dowolnie określonym przedziale [0,x_{max}]

stosujemy podstawienie: $\kappa = k\,\Delta\kappa$ $x_j = j\Delta x$

$$E_k(x) = e^{i\kappa\, x} = e^{i\kappa(x+x_{max})} = e^{i\kappa x}\underbrace{e^{i\kappa x_{max}}}_{=1}$$

$$e^{i\kappa\, x_{max}} = e^{ik\Delta\kappa N\Delta x} = e^{ik2\pi}$$

$$\Delta\kappa N\Delta x = 2\pi$$

$$\boxed{\Delta\kappa = \frac{2\pi}{N\Delta x}}$$

- element odległości w przestrzeni odwrotnej

$$\boxed{E_k(x_j) = e^{i\kappa x_j} = e^{i\frac{2\pi\, k\, j}{N}}}$$

- w wykładniku brak zależności od długości

Własności iloczynu wewnętrznego/skalarnego:

$$\langle f, f\rangle_N \geq 0$$

$$\langle f, g\rangle_N = \langle g, f\rangle_N^*$$

$$\langle \alpha f + \beta g, h\rangle_N = \alpha\langle f, h\rangle_N + \beta\langle g, h\rangle_N$$

oraz związek z normą euklidesową

$$\|f\|_2 = \sqrt{\langle f, f\rangle_N}$$

Dla każdego

$$N \geq 1 \quad \Longrightarrow \quad \langle E_k, E_m\rangle_N = \begin{cases} 1 & \frac{k-m}{N} \in \boldsymbol{Z} \\ 0 & \frac{k-m}{N} \notin \boldsymbol{Z} \end{cases}$$

Dowód

$$\frac{1}{N}\sum_{j=0}^{N-1} E_k^*\left(\frac{2\pi j}{N}\right) E_m\left(\frac{2\pi j}{N}\right) = \frac{1}{N}\sum_{j=0}^{N-1}\left[e^{-2\pi I(k-m)/N}\right]^j$$

$$e^{-2\pi I(k-m)/N} = 1 \Leftrightarrow \frac{k-m}{N} \in \boldsymbol{Z}$$

Uwaga:
brak węzła j=N
bo jest on równoważny j=0
$E_k(x)$ są okresowe

Dla pozostałych przypadków można się posłużyć wyrażeniem na sumę szeregu:

$$\sum_{j=0}^{N-1} \lambda^j = \frac{\lambda^N - 1}{\lambda - 1}, \quad \lambda \neq 1$$

co daje

$$\frac{e^{2\pi I(k-m)} - 1}{e^{2\pi I(k-m)/N} - 1} = 0$$

$$\cos\left[2\pi(k-m)\right] = 1$$

$$\sin\left[2\pi(k-m)\right] = 0$$

ze względu na postać licznika.

Funkcje $E_k(x)$ tworzą ciąg ortogonalnych (ortonormalnych) **jednomianów eksponencjalnych**, z których można utworzyć „wielomian”:

$$\begin{aligned} P(x) &= \sum_{k=0}^{N-1} c_k e^{ik\Delta\kappa\, x} = \sum_{k=0}^{N-1} c_k (e^{i\Delta\kappa\, x})^k \\ &= \sum_{k=0}^{N-1} c_k E_k(x) \end{aligned}$$

Wielomian eksponencjalny może posłużyć do interpolacji/aproksymacji funkcji f(x).

Załóżmy, że jej wartości są określone na siatce zbudowanej z równoodległych węzłów:

$$x_j = \frac{2\pi j}{N}, \qquad j = 0, 1, \ldots, N-1 \quad \Longrightarrow \quad x_j = 0, \frac{2\pi}{N}, \frac{2 \cdot 2\pi}{N}, \ldots, \frac{(N-1) \cdot 2\pi}{N}$$

Wielomian interpolujący ma wówczas postać

$$f(x) = P(x) = \sum_{k=0}^{N-1} c_k E_k(x)$$

Uwaga:
nie uwzględniamy punktu 2π bo jest on powtórzeniem wezła x_0

Współczynniki znajdziemy licząc iloczyny skalarne (lewa i prawa strona) z kolejnymi jednomianami E_m

$$\langle f, E_m \rangle = \sum_{k=0}^{N-1} c_k \langle E_k, E_m \rangle = \sum_{k=0}^{N-1} c_k \delta_{k,m} = c_m$$

Ciąg współczynników c_m wyznaczanych zgodnie z powyższym wzorem definiuje **dyskretną transformatę Fouriera** (DFT – to wynik przekształcenia).

$$f(x_j) = P(x_j) = \sum_{k=0}^{N-1} \langle f, E_k \rangle E_k(x_j)$$

W metodzie najmniejszych kwadratów wielomian ten może posłużyć do aproksymacji funkcji f(x), gdy stopień wielomianu aproksymującego jest mniejszy od N-1.

DFT można zapisać wykorzystując postać macierzową

$$Ef = c$$

$$\begin{pmatrix} E_0(x_0) & E_0(x_1) & \dots & E_0(x_{N-1}) \\ E_1(x_0) & E_1(x_1) & \dots & E_1(x_{N-1}) \\ \vdots & \vdots & \ddots & \vdots \\ E_{N-1}(x_0) & E_{N-1}(x_1) & \dots & E_{N-1}(x_{N-1}) \end{pmatrix} \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{N-1}) \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{pmatrix}$$

Transformatę można znaleźć wykonując „tylko” mnożenie wektora przez macierz. Ale w ten sposób należy wykonać $O(N^2)$ operacji arytmetycznych.

Jednakże macierz E ma specyficzną postać – jej elementy są ze sobą ściśle powiązane co można wykorzystać w celu zmniejszenia nakładu obliczeń.

Dzięki FFT liczba wykonywanych operacji może zmaleć do wartości $O(N\log_2 N)$.

N	N^2	$N \cdot log_2(N)$
1024	1048576	10240
4096	16777216	49152
16384	268435456	229375

Algorytm radix-2

Najprostszy algorytm FFT to radix-2 (**Cooley-Tukey**) opracowany w latach 60 XX wieku w celu szybkiej analizy danych sejsmologicznych.

Naszym zadaniem jest obliczenie współczynników transformaty Fouriera (DFT) c_k, ale wykonując jak najmniej obliczeń.

Zakładamy że całkowita liczba węzłów jest potęgą 2:

$$\begin{aligned} x_j &= \frac{2\pi}{N} j \\ j &= 0, 1, 2, \ldots, N-1 \\ N &= 2^r,\ r \in \boldsymbol{N} \end{aligned}$$

$$\begin{aligned} c_k &= \langle E_k, f \rangle = \sum_{j=0}^{N-1} E_k^*(x_j) f(x_j) \\ &= \sum_{j=0}^{N-1} f(x_j) \exp\left(-I x_j k\right) \\ &= \sum_{j=0}^{N-1} f_j \exp\left(-I \frac{2\pi}{N} jk\right) \end{aligned}$$

Osobno grupujemy składniki

parzyste $j = 2m$

nieparzyste $j = 2m + 1$

$$c_k = \sum_{m=0}^{\frac{N}{2}-1} f_{2m} exp\left(-I\frac{2\pi}{N}(2m)k\right) + \sum_{m=0}^{\frac{N}{2}-1} f_{2m+1} exp\left(-I\frac{2\pi}{N}(2m+1)k\right)$$

$$c_k = \sum_{m=0}^{\frac{N}{2}-1} f_{2m} exp\left(-I\frac{2\pi}{N/2}mk\right) + exp\left(-I\frac{2\pi}{N}k\right) \sum_{m=0}^{\frac{N}{2}-1} f_{2m+1} exp\left(-I\frac{2\pi}{N/2}mk\right)$$

$$c_k = p_k + \varphi_k q_k$$

$$p_k = \sum_{m=0}^{\frac{N}{2}-1} f_{2m} exp\left(-I\frac{2\pi}{N/2}mk\right)$$

$$q_k = \sum_{m=0}^{\frac{N}{2}-1} f_{2m+1} exp\left(-I\frac{2\pi}{N/2}mk\right)$$

$$\varphi_k = exp\left(-I\frac{2\pi}{N}k\right)$$

Korzystamy teraz z okresowości wyrazów p_k oraz q_k:

$$p_{k+N/2} = p_k \qquad q_{k+N/2} = q_k$$

(nie musimy wyznaczać wszystkich współczynników – tylko połowę)

Natomiast czynnik fazowy ma następującą własność:

$$\begin{aligned}
\varphi_{k+N/2} &= \exp\left(-I\frac{2\pi}{N}\left(k+\frac{N}{2}\right)\right) \\
&= \exp\left(-I\frac{2\pi}{N}k\right)\exp\left(-I\frac{2\pi}{N}\frac{N}{2}\right) \\
&= -\exp\left(-I\frac{2\pi}{N}k\right) = -\varphi_k
\end{aligned}$$

Uwagi:

a) współczynniki p_k oraz q_k można wyliczyć dzięki DFT nakładem $O(N/2)^2=O(N^2/4)$

b) dodatkowo oszczędzamy czas wyznaczając tylko współczynniki dla

$$k < \frac{N}{2}$$

ponieważ

$$c_k = \begin{cases} p_k + \varphi_k q_k, & k < \frac{N}{2} \\ p_{k-\frac{N}{2}} - \varphi_k q_{k-\frac{N}{2}}, & k \geq \frac{N}{2} \end{cases}$$

c) kolejnym krokiem w FFT jest podział sum w p_k oraz w q_k na sumy zawierające tylko elementy parzyste i nieparzyste

d) po podziale liczba elementów w każdej z dwóch powstałych sum jest dwukrotnie mniejsza niż w elemencie macierzystym

e) proces **rekurencyjnego** podziału kończymy, gdy liczba elementów jest równa 1

FFT z rozkładem na czynniki pierwsze (PFA - Prime Factor Algorithm)

Liczbę naturalną możemy zapisać jako $N = r_1 r_2 \dots r_p$

gdzie: r_i są liczbami pierwszymi przykład $N = 56 = 2 \cdot 2 \cdot 2 \cdot 7$

Idea algorytmu PFA polega na zastąpieniu obliczeń DFT w jednym wymiarze (skala N^2), na obliczeniu iloczynu p transformat DFT skalujących się jak

$$r_1^2,\ r_2^2, \dots, r_p^2,$$

Algorytm PFA zmniejsza nakład obliczeń z N^2 do $N(r_1+r_2+\dots+r_p)$

Zakładamy, że funkcja f(x) jest stablicowana w n węzłach

$$x_j = \frac{2\pi j}{n}, \quad j = 0, 1, \dots, n-1$$

Funcję rozwijamy w bazie wielomianów eksponencjalnych

$$F(x) = \sum_{k=0}^{n-1} c_k e^{i\,kx}$$

Wykorzystując własności wielomianów, współczynniki rozwinięcia zapisujemy w postaci

$$c_k = \frac{1}{n}\sum_{j=0}^{n-1} f(x_j)e^{-i\,kx_j}, \quad k = 0, 1, \ldots, n-1$$

Oznaczenia:

$$\left.\begin{array}{l} w = exp\left(-\frac{i\,2\pi}{n}\right) \\ a_j = \frac{1}{n} f\left(\frac{2\pi j}{n}\right) \end{array}\right\} c_k = \sum_{j=0}^{n-1} a_j w^{kj}$$

Rozkładamy liczbę węzłów na iloczyn liczb pierwszych:

$$n = r_1 r_2 \ldots r_p$$

i wprowadzamy kolejne oznaczenia:

$$n_\nu = r_{\nu+1} r_{\nu+2} \ldots r_p = \prod_{i=\nu+1}^{p} r_i$$

$$\nu = 0, 1, \ldots, p-1$$

$$n_p = 1$$

$$n = r_1 r_2 \ldots r_\nu\, n_\nu$$

Zmienną k zapisujemy w postaci:

$$k = \alpha_1 n_1 + \alpha_2 n_2 + \ldots + \alpha_p n_p$$

$$\begin{aligned} \alpha_1 &\in \{0, 1, \ldots, r_1 - 1\} \\ \alpha_2 &\in \{0, 1, \ldots, r_2 - 1\} \\ &\vdots \\ \alpha_p &\in \{0, 1, \ldots, r_p - 1\} \end{aligned}$$

$$k_\nu = \sum_{i=\nu+1}^{p} \alpha_i n_i$$

$$\frac{j_\nu}{n_\nu} = \sum_{i=\nu}^{p-1} \frac{l_{i+1}}{n_i}$$

$$k_\nu < n_\nu, \ \nu = 0, 1, \ldots, p$$

i podobnie ułamek j/n:

$$\frac{j}{n} = \frac{l_1}{n_0} + \frac{l_2}{n_1} + \ldots \frac{l_p}{n_{p-1}}$$

$$\begin{aligned} l_1 &\in \{0, 1, \ldots, r_1 - 1\} \\ l_2 &\in \{0, 1, \ldots, r_2 - 1\} \\ &\vdots \\ l_p &\in \{0, 1, \ldots, r_p - 1\} \end{aligned}$$

W wykładniku funkcji **exp** mamy wyraz

$$\begin{aligned} \frac{k \cdot j}{n} &= \left(\sum_{i=1}^{p} \alpha_i n_i \right) \left(\sum_{\nu=0}^{p-1} \frac{l_{\nu+1}}{n_\nu} \right) \\ &= \sum_{\nu=0}^{p-1} \frac{l_{\nu+1}}{n_\nu} \left(\sum_{i=\nu+1}^{p} \alpha_i n_i \right) + M \\ &= \sum_{\nu=0}^{p-1} \frac{l_{\nu+1} k_\nu}{n_\nu} + M \end{aligned}$$

$$M \in \boldsymbol{Z}$$

podstawmy ten wynik do rozwinięcia

Wykorzystujemy uzyskany wynik do obliczenia w^{kj}

$$
\begin{aligned}
w^{kj} &= exp\left(-2\pi i\,\frac{kj}{n}\right) \\
&= exp\left(-2\pi i\,\left(\sum_{\nu=0}^{p-1}\frac{l_{\nu+1}k_\nu}{n_\nu}+M\right)\right) \\
&= \prod_{\nu=0}^{p-1} exp\left(-2\pi i\,\frac{l_{\nu+1}k_\nu}{n_\nu}\right) \\
&= \prod_{\nu=0}^{p-1} w_\nu^{k_\nu l_{\nu+1}}
\end{aligned}
$$

uwaga:

$$\exp\left(\pm i\,M2\pi\right)=1$$

Chcemy znaleźć wartość współczynników c_k

$$
\begin{aligned}
c_k &= \sum_{j=0}^{n-1}\frac{f(x_j)}{n}exp\left(-i\,k\frac{2\pi j}{n}\right) \\
&= \sum_{j=0}^{n-1}\frac{f(x_j)}{n}w^{kj}
\end{aligned}
$$

Wykorzystujemy teraz zależność pomiędzy wskaźnikiem **j** a $\mathbf{l_1,l_2,\ldots,l_p}$ w sumowaniu

$$j = l_1 + l_2 r_1 + \ldots + l_p r_1 r_2 \ldots r_p$$

Sumę po j możemy zapisać jako

$$\sum_{j=0}^{n-1} \quad \Longrightarrow \quad \sum_{l_1=0}^{r_1-1} \sum_{l_2=0}^{r_2-1} \cdots \sum_{l_p=0}^{r_p-1}$$

ponieważ każdą wartość j realizuje odpowiednia kombinacja wskaźników $\mathbf{l_1,l_2,\ldots,l_p}$.

To przejście pozwala zapisać współczynnik c_k jako
iloczyn p transformat DFT jednowymiarowych

$$\begin{aligned} c_k &= c(l_1, l_2, \ldots, l_p) \\ &= \sum_{l_1=0}^{r_1-1} \sum_{l_2=0}^{r_2-1} \cdots \sum_{l_p=0}^{r_p-1} c^{(0)}(l_1, l_2, \ldots, l_p) w_0^{k_0 l_1} w_1^{k_1 l_2} \ldots w_{p-1}^{k_{p-1} l_p} \end{aligned}$$

Startujemy od obliczenia jednowymiarowego DFT dla wartości funkcji w węzłach, których indeksy zależą od aktualnych wartości $l_1, l_2, \ldots, l_p$

$$c^0 = \frac{f(x_j)}{n}, \; j = l_1 + l_2 r_1 + \ldots + l_p r_1 r_2 \ldots r_p$$

Takich transformat będzie **n/r_p**, a wyznaczenie każdej z nich wiąże się z nakładem obliczeń rzędu $(r_p)^2$

$$c^{(1)}(l_1, l_2, \ldots, \alpha_p) = \sum_{l_p=0}^{r_p-1} c^{(0)}(l_1, l_2, \ldots, l_p) w_{p-1}^{k_{p-1} l_p}$$

następnie obliczamy

$$c^{(2)}(l_1, l_2, \ldots, \alpha_{p-1}, \alpha_p) = \sum_{l_{p-1}=0}^{r_{p-1}-1} c^{(1)}(l_1, l_2, \ldots, l_{p-1}, \alpha_p) w_{p-2}^{k_{p-2} l_{p-1}}$$

Po wyznaczeniu w ten sposób p transformat dostajemy żądany współczynnik c_k (procedurę powtarzamy dla każdej wartości k)

$$c_k = c^{(p)}(\alpha_1, \alpha_2, \ldots, \alpha_{p-1}, \alpha_p) = \sum_{l_1=0}^{r_1-1} c^{(p-1)}(l_1, \alpha_2, \ldots, \alpha_{p-1}, \alpha_p) w_0^{k_0 l_1}$$

Inne algorytmy FFT

- **Split-radix** – modyfikacja algorytmu **Cooleya-Tukeya**. W każdym kroku DFT jest wyrażana jako suma DFT dla N/2 oraz dwóch DFT dla N/4. Jest to najszybszy algorytm FFT.

- **DST** (discrete sine transform) oraz **DCT** (discrete cosine transform)
  - transformaty sinusowa i kosinusowa, opłaca się je stosować gdy transformację przeprowadzamy na funkcjach rzeczywistych. Unikamy w ten sposób operacji na liczbach zespolonych co jest kosztowne.

Wielowymiarowa FFT

Transformacja Fouriera jest operacją liniową, zatem w przypadku d-wymiarowym możemy dokonać d transformacji niezależnych.

Współczynnki wyznacza się stosując algorytm jednowymiarowego FFT kolejno dla każdego z wymiarów.

$$N = N_1 N_2 \dots N_d$$

$$c_{k_1,k_2,\dots,k_d} = \sum_{j_1=0}^{N_1} \sum_{j_2=0}^{N_2} \dots \sum_{j_d=0}^{N_d} f_{j_1,j_2,\dots,j_d} \exp\left(-i\,2\pi\left(\frac{j_1 k_1}{N_1} + \frac{j_2 k_2}{N_2} + \dots + \frac{j_d k_d}{N_d}\right)\right)$$

Zastosowania FFT:

1) interpolacja, aproksymacja

2) szybkie mnożenie wielomianów

3) cyfrowe przetwarzanie sygnału
(np. odszumianie, analiza widma częstotliwości)

4) kompresja danych

5) analiza sygnałów czasowych (korelacja, splot)

6) rozwiązywanie równań różniczkowych (rów. Poissona)

7) całkowanie (splot)

Szybkie mnożenie wielomianów przy użyciu FFT

Chcemy obliczyć iloczyn dwóch wielomianów

$$P(x) = \sum_{i=0}^{n-1} a_i x^i$$

$$Q(x) = \sum_{i=0}^{n-1} b_i x^i$$

Jeśli stopnie wielomianów są różne to je wyrównujemy dodając do wielomianu niższego stopnia współczynniki równe 0.

Iloczyn wielomianów

$$\begin{aligned} R(x) &= P(x)Q(x) = \sum_{i=0}^{n-1} a_i x^i \sum_{j=0}^{n-1} b_j x^j \\ &= \sum_{i,j=0}^{n-1} a_i b_j x^{i+j} \end{aligned}$$

Dokonujemy reindeksacji wskaźników

$$i + j = k \quad \rightarrow \quad j = k - i$$

Po reindeksacji dostajemy

$$R(x) = \sum_{k=0}^{2n-2} \underbrace{\left(\sum_{i=0}^{n-1} a_i b_{k-i} \right)}_{c_k} x^k = \sum_{k=0}^{2n-1} c_k x^k$$

$$c_k = \sum_{i=0}^{n-1} a_i b_{k-i}$$

$$c_{2n-1} = 0$$

Jeśli współczynniki wielomianów a_i oraz b_i potraktujemy jako współrzędne wektorów

$$\boldsymbol{a} = [a_0, a_1, \ldots, a_{n-1}]$$
$$\boldsymbol{b} = [b_0, b_1, \ldots, b_{n-1}]$$

to wektor **c** jest ich splotem:

$$\boldsymbol{c} = \boldsymbol{a} * \boldsymbol{b}$$

Korzystając z definicji tranformacji Fouriera dla splotu funkcji możemy zapisać

$$\boldsymbol{c} = FFT^{-1}\left[FFT(\tilde{\boldsymbol{a}}) FFT(\tilde{\boldsymbol{b}}) \right]$$

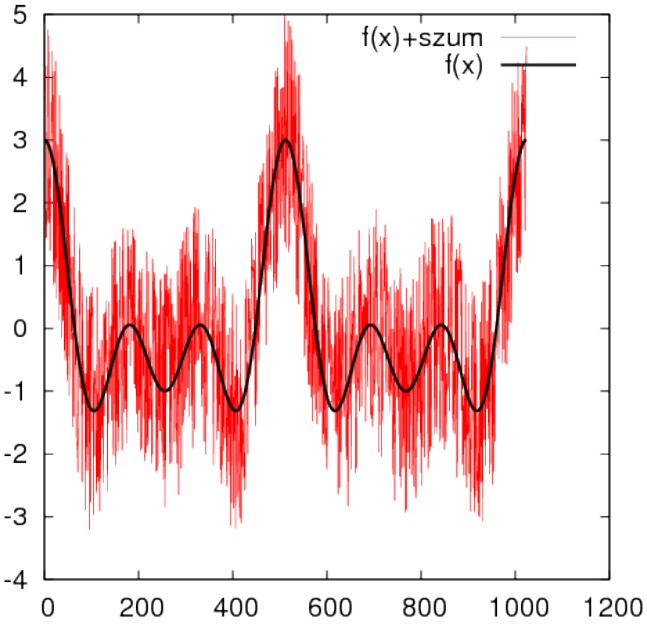
$$\begin{aligned} \tilde{\boldsymbol{a}} &= [a_0, a_1, \ldots, a_{n-1}, a_n, \ldots, a_{2n-1}] \\ \tilde{\boldsymbol{b}} &= [b_0, b_1, \ldots, b_{n-1}, b_n, \ldots, b_{2n-1}] \end{aligned}$$

Wektory a i b powiększamy więc dodatkowe elementy zerujemy (bo ich nie ma):

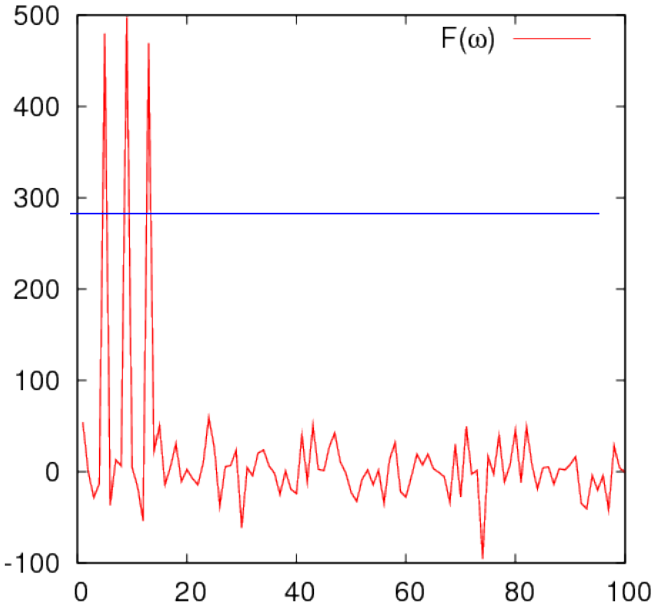
$$a_i, b_i = 0 \Leftrightarrow i > n-1$$

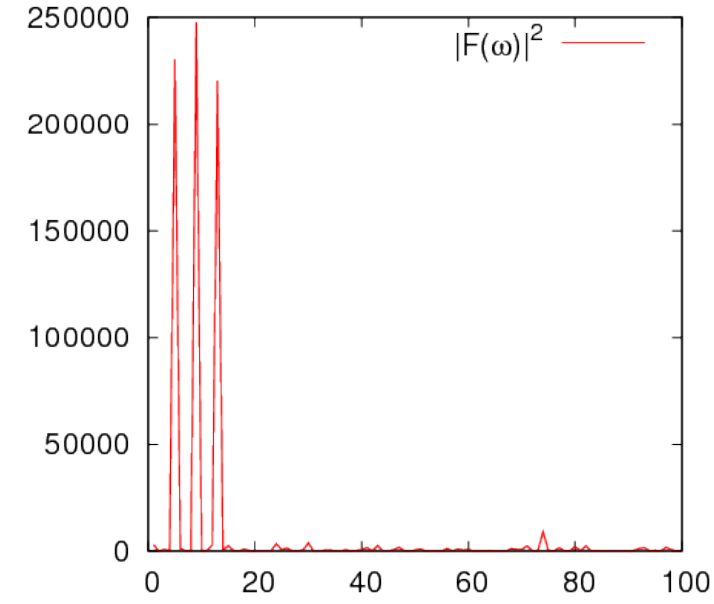
Przykład:
Filtracja sygnału (aproksymacja)

$$f(x) = cos(x) + cos(2x) + cos(3x)$$



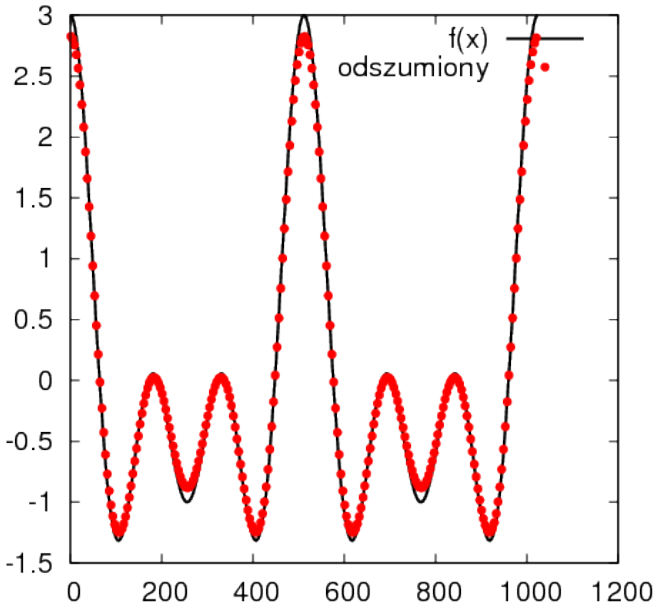
FFT





Dyskryminacja szumu

FFT^{-1}



Rozwiązywanie równania Poissona (2D, 3D)

$$\nabla^2 V_r = \rho_r$$

dokonujemy transformacji (FFT) całego równania (do przestrzeni odwrotnej)

$$k^2 V_k = \rho_k$$

skąd już łatwo wyznaczymy V_k

$$V_k = \frac{\rho_k}{k^2}$$

i gotowe rozwiązanie (w przestrzeni rzeczywistej)

$$V_r = FFT^{-1}\{V_k\}$$

Uwaga: musimy jeszcze uwzględnić warunki brzegowe (WB)

1) jeśli WB są typu Dirichleta

$$V_r|_{brzeg} = V_b \neq 0$$

to wykonujemy DST-FFT dla wnętrza obszaru, a WB uwzględniamy dokonując transformat potencjału na brzegach – wyniki dodajemy

2) jeśli WB są typu Neumanna

$$\frac{\partial V_r}{\partial \vec{n}}|_{brzeg} = 0$$

to wówczas stosujemy DCT-FFT dla wnętrza obszaru. WB są automatycznie spełnione.

Całkowanie poprzez liczenie splotu

Chcemy obliczyć całkę oddziaływania dwóch gęstości ładunku/materii

$$C = \int_a^b d\vec{r}_1 \int_a^b d\vec{r}_2 \frac{\rho_1(\vec{r}_1)\rho_2(\vec{r}_2)}{|\vec{r}_1 - \vec{r}_2|}$$

która zawiera osobliwość.

$$|\vec{r}_1 - \vec{r}_2|^{-1} = f(\vec{r}_1 - \vec{r}_2) = f(\vec{r}_2 - \vec{r}_1)$$

Całkę możemy zapisać nieco inaczej

$$C = \int_a^b d\vec{r}_1 \rho_1(\vec{r}_1) V(\vec{r}_1) \quad \Longrightarrow \quad V(\vec{r}_1) = \int_a^b d\vec{r}_2 \rho_2(\vec{r}_2) f(\vec{r}_1 - \vec{r}_2)$$

Potencjał $V(r_1)$ jest splotem dwóch funkcji: gęstości i funkcji f.
Można więc wykorzystać tu twierdzenie o splocie i jego transformacie:

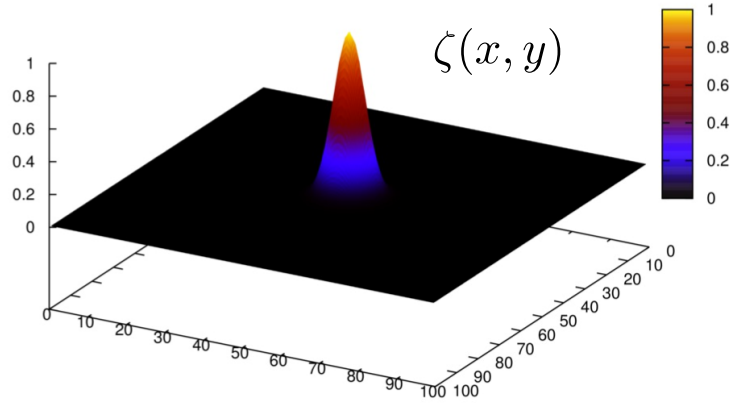
$$FFT\{V\} = FFT\{\rho_2 * f\} = FFT\{\rho_2\} \cdot FFT\{f\}$$

Następnie wykonujemy transformację odwrotną ostatniego iloczynu co daje poszukiwany potencjał V(r).

Generowanie pola prędkości w równaniu adwekcji

Równanie adwekcji

$$\frac{du}{dt} = -\vec{V} \cdot \nabla u$$



Zadajemy funkcję wirowości

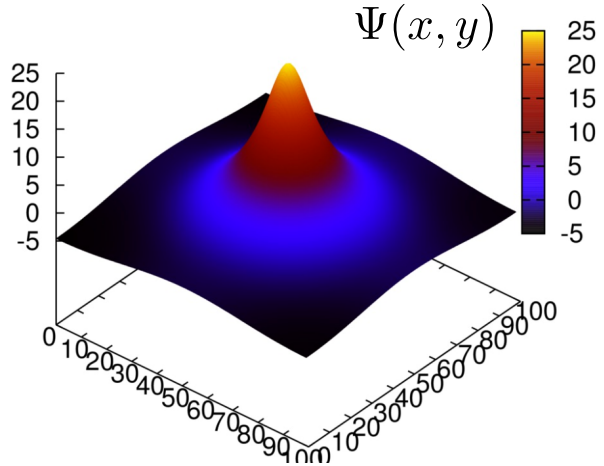
$$\zeta(x,y) = \exp\left(-\frac{(x-x_0)^2 + (y-y_0)^2}{2\sigma^2}\right)$$

Rozwiązując równanie Poissona dla funkcji strumienia Ψ(x,y)

$$\nabla^2 \Psi = \zeta$$

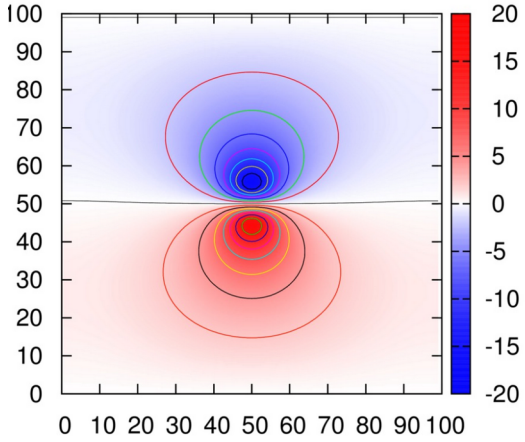
z **WB** (**DCT - FFTW**)

$$\frac{\partial \Psi}{\partial \vec{n}} = 0$$

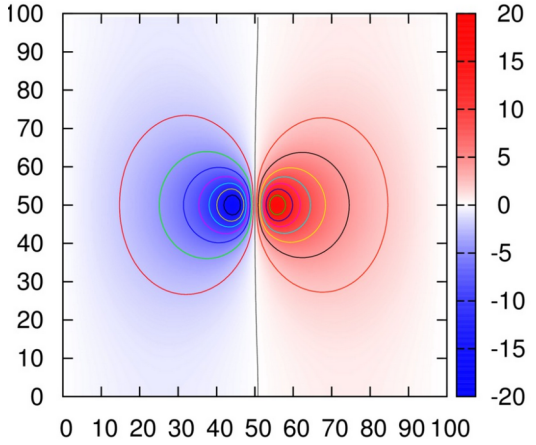


możemy otrzymać pole prędkości

$$V_x = \frac{\partial \Psi}{\partial y}$$



$$V_y = -\frac{\partial \Psi}{\partial x}$$



Rozwiązanie - plamka oleju poruszająca się po powierzchnii cieczy

$$\frac{du}{dt} = -\vec{V} \cdot \nabla u$$