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) szybkie mnożenie wielomianów

Jeśli funkcja f(x) jest okresowa wówczas zamiast wielomianów do jej interpolacji (aproksymacji) lepiej użyć wielomianów trygonometrycznych tj. rozwinąć funkcję w **szereg Fouriera.** Dla funkcji okresowej o okresie 2π:

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

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)cos(kt)dt$$

$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)sin(kt)dt$$

Funkcję możemy też zapisać w postaci zespolonego szeregu Fouriera

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

$$\widehat{f}(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)e^{-ikt}dt$$

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

$$\begin{aligned} \widehat{f}(k) &= \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)[cos(kt) - isin(kt)]dt \\ &= \frac{1}{2}(a_k - ib_k) \\ k &\geq 0 \end{aligned}$$

Dla ciągu współczynników

$$[a_k]_{k=0}^{\infty} \qquad [b_k]_{k=1}^{\infty}$$

definiujemy

$$b_0 = 0 \quad a_{-k} = a_k \quad b_{-k} = -b_k$$

$$c_k = \frac{1}{2}(a_k - ib_k)$$

Co prowadzi do zależności pomiędzy szeregiem rzeczywistym i zespolonym

$$\frac{a_0}{2} + \sum_{k=1}^{n}(a_k cos(kx) + b_k sin(kx)) = \sum_{k=-n}^{n} c_k e^{ikx}$$

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:

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

$$\begin{aligned} \langle E_k, E_n \rangle &= \frac{1}{2\pi} \int_{-\pi}^{\pi} E_k(x) E_n^*(x) dx \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{ikx} e^{-inx} dx \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i(k-n)x} dx \\ &= \frac{1}{2\pi} \frac{e^{i(k-n)x}}{i(k-n)} \Bigg|_{x=-\pi}^{x=\pi} = 0 \end{aligned}$$

Dysponując tablicą wartości funkcji f i g w węzłach siatki, iloczyn wewnętrzny można zapisać w postaci dyskretnej:

$$\langle f, g \rangle_N = \frac{1}{N} \sum_{j=0}^{N-1} f(2\pi j/N) g^*(2\pi j/N)$$

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

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

Dla każdego

$$N \geq 1$$

$$\langle E_k, E_m \rangle_N = \begin{cases} 1 & \frac{k-m}{N} = liczba \quad calkowita \\ 0 & \frac{k-m}{N} \neq liczba \quad calkowita \end{cases}$$

Dowód

$$\frac{1}{N} \sum_{j=0}^{N-1} E_k\left(\frac{2\pi j}{N}\right) E_m^*\left(\frac{2\pi j}{N}\right) =$$

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

$$e^{2\pi i(k-m)/N} = 1 \Leftrightarrow \frac{k-m}{N} = liczba\ calkowita$$

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

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

ze względu na postać licznika.

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

$$\begin{aligned} P(x) &= \sum_{k=0}^{n} c_k e^{ikx} = \sum_{k=0}^{n} c_k (e^{ix})^k \\ &= \sum_{k=0}^{n} c_k E_k(x) \end{aligned}$$

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

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

$$x_j = \frac{2\pi j}{N}, \; j = 0, 1, \ldots, N-1$$

Wielomian interpolujący ma wówczas postać

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

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

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

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

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

W metodzie najmniejszych kwadratów wielomian ten może posłużyć do aproksymacji funkcji f(x) gdy n<N.

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	$Nlog_2N$
1024	1048576	10240
4096	16777216	49152
16384	268435456	229375

Szybka transformacja Fouriera

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:

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

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

Osobno grupujemy składniki

a) parzyste $j = 2m$

b) nieparzyste $j = 2m+1$

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

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

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

$$c_k = p_k + \varphi_k q_k$$

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

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

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

Korzystamy teraz z okresowości funkcji p_k oraz q_k:

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

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(N²/4)

b) dodatkowo oszczędzamy czas wyznaczając tylko współczynniki dla

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

ponieważ

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

Kolejnym krokiem w FFT jest podział sum w p_k oraz w q_k na sumy zawierające tylko elementy parzyste i nieparzyste. Po podziale liczba elementów w każdej z dwóch powstałych sum jest dwukrotnie mniejsza niż w elemencie macierzystym.
Proces **rekurencyjnego** podziału kończymy gdy liczba elementów jest równa 1.

Inne algorytmy FFT

1) **PFA** (prime factors algorithm) – $N=N_1N_2$, gdy N_1, N_2 są liczbami pierwszymi. DFT jest obliczane jako dwuwymiarowe FFT

2) **Algorytm Winograda/Radera** – DFT jest sformułowane w postaci cyklicznego splotu. Modyfikacja Winograda algorytmu FFT działa gdy $N=p^m$, p -liczba pierwsza.

3) **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.

4) **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.

5) **Wielowymiarowa FFT**

$$N = N_1 N_2 \dots N_d$$

$$c_{k_1,k_2,\dots,k_d} = \sum_{j_1=0}^{N_1} \sum_{j_2=0}^{N_2} \dots \sum_{j_d=0}^{N_d} f_{j_1,j_2,\dots,j_d} exp\left(-i2\pi\left(\frac{j_1k_1}{N_1} + \frac{j_2k_2}{N_2} + \dots + \frac{j_dk_d}{N_d}\right)\right)$$

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

Zastosowania FFT:

1) interpolacja, aproksymacja
2) szybkie mnożenie, dzielenie wielomianów
3) cyfrowe przetwarzanie sygnału (np. odszumianie – widmo częstotliwości)
4) kompresja danych
5) analiza sygnałów czasowych
6) rozwiązywanie równań różniczkowych (rów. Poissona)

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

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

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

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

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

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

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

$$\tilde{\boldsymbol{a}} = [a_0, a_1, \ldots, a_{n-1}, a_n, \ldots, a_{2n-1}]$$

$$\tilde{\boldsymbol{b}} = [b_0, b_1, \ldots, b_{n-1}, b_n, \ldots, b_{2n-1}]$$

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