Szybka transformacja Fouriera

Doc. dr Antoni KRECZMAR

Klasyczna transformacja Fouriera odgrywa zasadniczą rolę we współczesnej matematyce, w szczególności w analizie harmonicznej. Jak pisze prof. Maurin, zarówno w rachunkach, jak i rozważaniach teoretycznych oddaje ona nieocenione usługi, a pewien wybitny matematyk stwierdził: „ludzie wymyślili wiele transformacji, ale transformację Fouriera stworzył Bóg" (matematyk ten jest ateistą). Młodszą siostrą klasycznej transformacji Fouriera jest dyskretna transformacja Fouriera, która odgrywa zasadniczą rolę we współczesnej informatyce.

Aby w możliwie prosty sposób zdefiniować dyskretną transformację Fouriera, rozważmy dowolny wektor $a = [a_0, a_1, \ldots, a_{n-1}]$, którego współrzędne są liczbami zespolonymi, przez P oznaczmy wielomian

$$P(z) = a_0 + a_1 z + \ldots + a_{n-1} z^{n-1}.$$

Niech ω będzie pierwotnym pierwiastkiem n-tego stopnia z jedności, wówczas ciąg $\omega^0, \omega^1, \ldots, \omega^{n-1}$ określa wszystkie pierwiastki n-tego stopnia z jedności. Otóż dyskretna transformacja Fouriera jest to funkcja F, która danemu wektorowi a przyporządkowuje wektor $b = [b_0, b_1, \ldots, b_{n-1}]$ o współrzędnych określonych przez wartości wielomianu P w kolejnych punktach $\omega^0, \omega^1, \ldots, \omega^{n-1}$, tzn.

$$b = F(a) = [P(\omega^0), P(\omega^1), \ldots, P(\omega^{n-1})].$$

Wiadomo, że iloczyn liczb zespolonych $w_1 = r_1(\cos\alpha_1 + i\sin\alpha_1)$ i $w_2 = r_2(\cos\alpha_2 + i\sin\alpha_2)$ danych w postaci trygonometrycznej jest liczbą $w_1 \cdot w_2 = r_1 \cdot r_2(\cos(\alpha_1+\alpha_2) + i\sin(\alpha_1+\alpha_2))$. Tak więc pierwiastkiem n-tego stopnia z 1 $= \cos 0 + i \sin 0$ będzie każda taka liczba $z = \cos\varphi + i\sin\varphi$, że $n \cdot \varphi = 2k\pi$, gdzie k jest liczbą całkowitą. Liczby $\omega_k = \cos\frac{2k}{n}\pi + i\sin\frac{2k}{n}\pi$; $k = 0, \ldots, n-1$ są więc wszystkimi pierwiastkami n-tego stopnia z 1. Zauważmy, że $(\omega_1)^k = \omega_k$ — taki pierwiastek, którego potęgami są wszystkie inne pierwiastki n-tego stopnia z 1 nazywamy pierwiastkiem pierwotnym.

Czyli współrzędne wektora b są postaci $b_j = \sum_{k=0}^{n-1} a_k(\omega^k)^j$.

Bardzo ważną cechą klasycznej transformacji Fouriera jest to, że transformacja odwrotna ma postać bardzo zbliżoną do transformacji zwykłej. Dyskretna transformacja Fouriera ma podobną własność. Mianowicie, łatwo pokazać, że

$$F^{-1}(a) = \frac{1}{n}[P(\omega^{-0}), P(\omega^{-1}), \ldots, P(\omega^{-(n-1)})].$$

Wystarczy wykazać, że $F^{-1}(F(a)) = a$. Oznaczmy przez c_k $(0 \le k < n)$ k-tą współrzędną wektora $F^{-1}(F(a))$. Wówczas:

$$n \cdot c_k = \sum_{j=0}^{n-1} b_j\omega^{-jk} = \sum_{j=0}^{n-1}\sum_{i=0}^{n-1} a_i\omega^{ij}\omega^{-jk} = \sum_{i=0}^{n-1} a_i \sum_{j=0}^{n-1} \omega^{j(i-k)}.$$

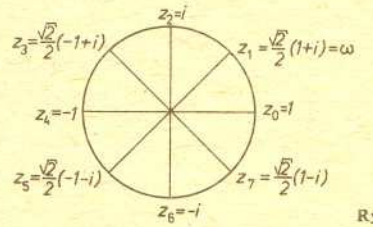
Ponieważ jednak ω jest pierwiastkiem n-tego stopnia z jedności, zatem dla $i \neq k$ $\sum_{j=0}^{n-1}\omega^{j(i-k)} = \frac{\omega^{n(i-k)}-1}{\omega^{i-k}-1} = 0$, oraz dla $i = k$ $\sum_{j=0}^{n-1}\omega^{j(i-k)} = n$.

A więc rzeczywiście $F^{-1}(F(a)) = a$.

Po co informatykom dyskretna transformacja Fouriera? O tym, drodzy Czytelnicy, dowiecie się w dalszej części niniejszego artykułu. Na razie zajmiemy się szybkim sposobem obliczania takich transformacji.

Algorytm szybkiego obliczania dyskretnych transformacji Fouriera w literaturze fachowej nosi skrótową nazwę FFT, od angielskiego Fast Fourier Transform. Załóżmy najpierw, że n jest potęgą liczby 2 (takie założenie w informatyce nie zmniejsza ogólności zastosowań). Niech zatem $n = 2^m$ dla pewnego całkowitego $m > 0$. Zadanie nasze polega na wyznaczeniu wartości wielomianu P w n punktach $\omega^0, \omega^1, \ldots, \omega^{n-1}$ (dla transformacji odwrotnej są to te same punkty, ale w trochę innej kolejności, co tylko w nieznacznym stopniu modyfikuje sposób postępowania).

Z twierdzenia Bezout wiemy, że dla dowolnego zespolonego z' wartość $P(z')$ równa się reszcie z dzielenia P przez dwumian $(z-z')$. No dobrze, ale zastanówmy się teraz, jak można obliczyć wartość wielomianu P w dwóch punktach z' i z''. Oczywiście można to zrobić dla każdego $z = z'$ i $z = z''$ oddzielnie. Można także podzielić najpierw P przez wielomian $(z-z')\cdot(z-z'')$, otrzymaną resztę stopnia pierwszego $az+b$ podzielić przez $(z-z')$ otrzymując $P(z')$, a następnie przez $(z-z'')$ otrzymując $P(z'')$. Takie dzielenie w ogólnym przypadku jest bardzo pracochłonne, ale dla naszego zbioru punktów można postąpić jeszcze sprytniej. Ponieważ $n = 2^m$, zatem pierwiastki n-tego stopnia z jedności rozłożą się na okręgu jednostkowym symetrycznie (rys. 1).



Rys. 1

Pogrupujmy je parami tak, aby iloczyn $(z-z')\cdot(z-z'')$ był zawsze wielomianem bez składnika liniowego, czyli aby był postaci z^2+a. Okazuje się, że jest to możliwe. Dla $n = 8$ takie pary mogą być następujące:

$$(z-z_0)(z-z_4) = (z-1)(z+1) = (z^2-1) = (z^2-z_0),$$

$$(z-z_2)(z-z_6) = (z-i)(z+i) = (z^2+1) = (z^2-z_4),$$

$$(z-z_1)(z-z_5) = \left(z-\frac{\sqrt{2}}{2}(1+i)\right)\left(z+\frac{\sqrt{2}}{2}(1+i)\right) =$$

$$= (z^2-i) = (z^2-z_2),$$

$$(z-z_3)(z-z_7) = \left(z-\frac{\sqrt{2}}{2}(-1+i)\right)\left(z+\frac{\sqrt{2}}{2}(-1+i)\right) =$$

$$= (z^2+i) = (z^2-z_6).$$

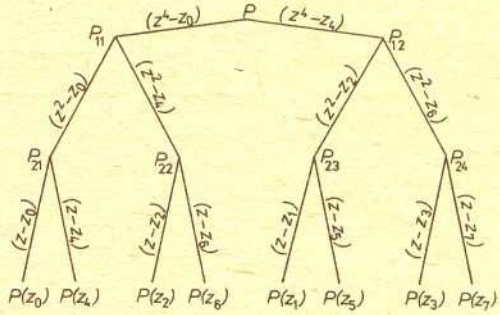
Tak więc, jeżeli podzielimy wielomian P przez wielomiany (z^2-1), (z^2+1), (z^2-i), (z^2+i), a następnie otrzymane reszty przez odpowiednie dwumiany $(z-z_k)$, to w ten sposób możemy obliczyć łatwo wszystkie wartości $P(z_k)$, $k = 0, 1, \ldots, 7$.

Zauważmy jednak dalej, że dzielenie przez (z^2-z_0), (z^2-z_4), (z^2-z_2) i (z^2-z_6) można wykonać rozumując podobnie. Mianowicie pogrupujmy parami te wielomiany tak, aby iloczyn $(z^2-z')\cdot(z^2-z'')$ był dwumianem postaci (z^4+a), co w dalszym ciągu jest możliwe:

$$(z^2-z_0)(z^2-z_4) = (z^2-1)(z^2+1) = (z^4-1) = (z^4-z_0),$$

$$(z^2-z_2)(z^2-z_6) = (z^2-i)(z^2+i) = (z^4+1) = (z^4-z_4).$$

I znowu dzieląc P przez (z^4-z_0) i (z^4-z_4) otrzymamy dwie reszty stopnia trzeciego, które dzieląc przez dwumiany (z^2-z_k), $k = 0, 4, 2, 6$, dadzą reszty stopnia pierwszego, a z tych reszt można już obliczyć $P(z_k)$ dla $k = 0, 1, \ldots, 7$, w sposób uprzednio opisany. Całość tego postępowania wygodnie jest przedstawić w postaci drzewa (rys. 2).



Rys. 2

To jeszcze nie koniec „chwytów" technicznych prowadzących do algorytmu FFT. Otóż dzielenie wielomianu P przez dwumiany (z^4-z_0) i (z^4-z_4) jest bardzo proste. Wystarczy zauważyć, że reszta z dzielenia wielomianu

$$P(z) = a_0 + a_1 z + \ldots + a_{2k-1} z^{2k-1}$$

przez dwumian (z^k-c) jest postaci

$$(*)\quad r(z) = (a_0+ca_k) + (a_1+ca_{k+1})z + \ldots + (a_{k-1}+ca_{2k-1})z^{k-1}.$$

Faktycznie, stopień wielomianu r jest mniejszy niż k oraz

$$P(z) = (a_k + a_{k+1}z + \ldots + a_{2k-1}z^{k-1})(z^k - c) + r(z).$$

Mamy zatem resztę z dzielenia wielomianu P przez $(z^4 - z_0)$ postaci

$$(a_0 + a_4) + (a_1 + a_5)z + (a_2 + a_6)z^2 + (a_3 + a_7)z^3$$

oraz resztę z dzielenia wielomianu P przez $(z^4 - z_4)$ postaci

$$(a_0 - a_4) + (a_1 - a_5)z + (a_2 - a_6)z^2 + (a_3 - a_7)z^3.$$

Dzielenie tak otrzymanych reszt przez dwumiany $(z^2 - z_k)$, $k = 0, 4, 2, 6$, wykonujemy podobnie, tzn. stosując wzór (*). Ogólny schemat postępowania w przypadku $n = 8$ przedstawiony jest poniżej, natomiast przykład wykonania takich obliczeń dla konkretnego wielomianu w długiej tabelce.

3	1	2	1	2	1	1	1
5	2	3	2	1	0	1	0
8	4	2	0	1+i	0	1−i	0
12	4	2	2	1+i	1+i	1−i	1−i

$F([3, 1, 2, 1, 2, 1, 1, 1]) = [12, 1+i, 2, 1-i, 4, 1+i, 2, 1-i]$

współczynniki wielomianu P

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
współczynniki wielomianu P_{11}				współczynniki wielomianu P_{12}			
$b_0 = a_0 + a_4$	$b_1 = a_1 + a_5$	$b_2 = a_2 + a_6$	$b_3 = a_3 + a_7$	$b_4 = a_0 - a_4$	$b_5 = a_1 - a_5$	$b_6 = a_2 - a_6$	$b_7 = a_3 - a_7$
Współczynniki wielomianu P_{21}		Współczynniki wielomianu P_{22}		Współczynniki wielomianu P_{23}		Współczynniki wielomianu P_{24}	
$c_0 = b_0 + b_2$	$c_1 = b_1 + b_3$	$c_2 = b_0 - b_2$	$c_3 = b_1 - b_3$	$c_4 = b_4 + ib_6$	$c_5 = b_5 + ib_7$	$c_6 = b_4 - ib_6$	$c_7 = b_5 - ib_7$
$P(z_0)$	$P(z_4)$	$P(z_2)$	$P(z_6)$	$P(z_1)$	$P(z_5)$	$P(z_3)$	$P(z_7)$
$d_0 = c_0 + c_1$	$d_1 = c_0 - c_1$	$d_2 = c_2 + ic_3$	$d_3 = c_2 - ic_3$	$d_4 = c_4 + \frac{\sqrt{2}}{2}(1+i)c_5$	$d_5 = c_4 - \frac{\sqrt{2}}{2}(1+i)c_5$	$d_6 = c_6 + \frac{\sqrt{2}}{2}(-1+i)c_7$	$d_7 = c_6 - \frac{\sqrt{2}}{2}(-1+i)c_7$

Ile operacji arytmetycznych na liczbach zespolonych trzeba wykonać w celu obliczenia $F(a)$ stosując algorytm FFT? Zauważmy, że dla $n = 2^m$ musimy wyznaczyć m kolejnych wierszy reprezentujących współczynniki odpowiednich reszt. Łatwo sprawdzić, że wyznaczenie takiego wiersza z poprzedniego wymaga n dodawań i n mnożeń. Zatem w sumie trzeba wykonać $2 \cdot 2^m \cdot m = 2 \cdot n \cdot \log(n)$ operacji arytmetycznych na liczbach zespolonych. Jest to kluczowa własność tego algorytmu. Mianowicie wyznaczenie $F(a)$ za pomocą algorytmu FFT wymaga wykonania tylko rzędu $n\log(n)$ operacji arytmetycznych, podczas gdy algorytmy tradycyjne wyznaczające dyskretną transformację Fouriera wymagają wykonania rzędu n^2 operacji arytmetycznych. Wróćmy teraz do pytania, po co w ogóle chcemy wyznaczać dyskretne transformacje Fouriera. Otóż pomocne są one niezwykle w obliczaniu splotu wektorów. Cóż to jest splot wektorów? Znacie zapewne tę operację, jakkolwiek być może sam termin jest Wam nieznany. Niech P i Q będą dwoma wielomianami stopnia $n-1$:

$$P(z) = a_0 + a_1 z + \ldots + a_{n-1}z^{n-1}, \quad Q(z) = b_0 + b_1 z + \ldots + b_{n-1}z^{n-1}.$$

Mnożąc te wielomiany otrzymamy wielomian stopnia $2n-2$

$$P(z)Q(z) = c_0 + c_1 z + \ldots + c_{2n-2}z^{2n-2}, \quad \text{gdzie}$$

$$c_0 = a_0 b_0,\ c_1 = a_0 b_1 + a_1 b_0,\ \ldots,\ c_{2n-2} = a_{n-1} b_{n-1}.$$

Otóż wektor $c = [c_0, c_1, \ldots, c_{2n-2}]$ nazywamy splotem wektorów $a = [a_0, a_1, \ldots, a_{n-1}]$, $b = [b_0, b_1, \ldots, b_{n-1}]$, co oznacza się najczęściej następująco: $c = a*b$.

Gdybyśmy chcieli wyznaczyć splot $a*b$ wprost z definicji, to musielibyśmy wykonać około $2n^2$ operacji arytmetycznych, co każdy z Was może łatwo sprawdzić. W celu wyznaczenia splotu można posłużyć się dyskretną transformacją Fouriera. Rozszerzmy mianowicie wektory a i b dołączając do nich n współrzędnych o wartościach 0, tj. $a = [a_0, a_1, \ldots, a_{n-1}, 0, \ldots, 0]$, $b = [b_0, b_1, \ldots, b_{n-1}, 0, \ldots, 0]$. Splot $a*b$ może mieć też $2n$ współrzędnych, jeżeli ostatnią ustalimy jako 0. Zachodzi wówczas bardzo ważne twierdzenie o splocie:

$$F(a*b) = F(a)F(b),$$

gdzie iloczyn po prawej stronie nie jest iloczynem skalarnym wektorów $F(a)$ i $F(b)$, ale iloczynem „po współrzędnych" (tzn. wynik nie jest skalarem, ale wektorem o współrzędnych będących iloczynami odpowiednich współrzędnych wektorów $F(a)$ i $F(b)$). Aby wyznaczyć splot $a*b$, obliczamy najpierw $F(a)$ i $F(b)$, następnie mnożymy te wektory „po współrzędnych" i wreszcie wyznaczamy transformację odwrotną $F^{-1}(F(a)F(b))$. Pierwsze dwie i ostatnia operacja mają koszt rzędu $n\log(n)$, a mnożenie „po współrzędnych" ma koszt rzędu n operacji arytmetycznych.

Na zakończenie naszego spotkania z szybką transformacją Fouriera podam jeszcze kilka ważnych faktów dotyczących samej transformacji, jak i jej zastosowań.

Niektórzy Czytelnicy mogą mieć wątpliwości, czy istnieje łatwy sposób znajdowania tej cudownej permutacji początkowej ciągu $z_0, z_1, \ldots, z_{n-1}$ (patrz dolny wiersz w rys. 2). Jest to permutacja rzeczywiście cudowna, albowiem na każdym piętrze drzewa mamy do czynienia tylko z dwumianami postaci $z^k - c$, gdzie c jest pewnym pierwiastkiem n-tego stopnia z jedności. Otóż istnieje łatwy sposób wyznaczania tej permutacji. Mianowicie, niech $n = 2^m$ i rozważmy dowolną liczbę k, $0 \leqslant k < n$. Taką liczbę można przedstawić w układzie dwójkowym jako $b_0 b_1 \ldots b_{m-1}$ ($b_i = 0$ lub 1). Niech teraz $rev(k)$ będzie liczbą z tego samego przedziału, o reprezentacji dwójkowej odwrotnej, tzn. $b_{m-1} \ldots b_1 b_0$. Permutacja przyporządkowująca liczbie k liczbę $rev(k)$ jest szukaną permutacją. Dla $n = 8$ mamy permutację (000), (100), (010), (110), (001), (101), (011), (111), a więc 0, 4, 2, 6, 1, 5, 3, 7 dziesiętnie. Pozostawiamy Czytelnikowi sprawdzenie, że permutacja rev ma żądane własności.

Twierdzenie o splocie można zinterpretować w następujący sposób. Mamy dane dwa wielomiany P i Q. Wyznaczamy ich wartości w $2n$ punktach $\omega^0, \omega^1, \ldots, \omega^{2n-1}$ (tu oczywiście ω jest pierwiastkiem stopnia $2n$), co odpowiada wyznaczeniu $F(a)$ i $F(b)$. Następnie mnożymy parami ich wartości w tych punktach $(F(a) \cdot F(b))$. Ostatecznie mając takie $2n$ wartości szukamy wielomianu stopnia co najwyżej $2n-1$, który w tych punktach przyjmuje wyznaczone wartości $F^{-1}(F(a) \cdot F(b))$. Ponieważ istnieje dokładnie tylko jeden wielomian o tej własności, który w punktach $\omega^0, \omega^1, \ldots, \omega^{2n-1}$ przyjmuje wartości $P(\omega^i)Q(\omega^i)$ dla $i = 0, \ldots, 2n-1$, zatem musi to być iloczyn wielomianów $P(z)Q(z)$. A więc użyta przez nas do pomnożenia dwóch wielomianów liczba działań jest rzędu $n \cdot \log n$, podczas gdy przy „normalnym" mnożeniu jest ona rzędu n^2.

Kilka słów o zastosowaniach szybkiej transformacji Fouriera. Autorzy podstawowego podręcznika ze złożoności obliczeniowej A. Aho, J. Hopcroft, J. Ullman (*Projektowanie i analiza algorytmów komputerowych*, PWN, 1983) piszą, że transformacja ta jest w informatyce wszechobecna. Jest to niewątpliwie prawda. Pokazaliśmy, jak ją stosować do obliczania splotów wektorów i iloczynu wielomianów. Większość operacji na wielomianach wykonuje się za pomocą tej transformacji. Stosuje się ją także dla arytmetyki wieloprecyzyjnej (gdy liczby trzeba reprezentować w komputerze za pomocą bardzo długiego ciągu bitów). Wiele zadań numerycznych rozwiązuje się za pomocą tej transformacji, na przykład niektóre zadania prowadzące od układów równań różniczkowych cząstkowych do układów równań liniowych. Czy Fourier mógł przewidzieć, że jego transformacja znajdzie zastosowanie w tomografii komputerowej? Ale nasi Czytelnicy po przeczytaniu niniejszego artykułu będą już wiedzieć, że tomografia komputerowa (nagrodzona zresztą Noblem w dziedzinie medycyny), która pozwala w niesłychanie precyzyjny a jednocześnie prawie nieszkodliwy sposób dokonać wewnętrznej obserwacji pacjenta, też korzysta z algorytmu FFT.