04

Cyfrowe Przetwarzanie Sygnałów

FFT – szybka transformata Fouriera

prof. dr hab. inż. Tomasz Zieliński, dr inż. Jarosław Bułat

2025.03.26

1a. FFT składane od drugiego poziomu „motylków" (2pkt)

Wykonaj zadanie 1a jeżeli Twój numer indeksu jest „parzysty", w przeciwnym wypadku wybierz 1b.

Złożoność obliczeniowa N-punktowej transformaty DFT określonej zależnością:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{-kn},\ k = 0, 1, 2,..., N - 1,\ W_N = e^{j\frac{2\pi}{N}} \quad (1)$$

to $O(N^2)$. Jedno zespolone N-punktowe DFT można wykonać w następujący sposób:

$$X(k) = X_1(k) + X_2(k) \quad (2)$$

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n) W_N^{-k(2n)} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_N^{-k(2n+1)},\ k = 0, 1, 2,..., N - 1 \quad (3)$$

$$W_N^{-2kn} = W_{\frac{N}{2}}^{-kn} \quad (4)$$

$$X(k) = \left(\sum_{n=0}^{\frac{N}{2}-1} x(2n) W_{\frac{N}{2}}^{-kn} \right) + W_N^{-k} \left(\sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_{\frac{N}{2}}^{-kn} \right),\ k = 0, 1, 2,..., N - 1. \quad (5)$$

Zauważ, że obliczenia w nawiasach z (5) to transformata DFT (1) o długości $N/2$ więc zależność można zapisać:

$$X(k) = DFT(x(2n)) + W_N^{-k} DFT(x(2n+1)),\ k = 0, 1, 2,..., \frac{N}{2} - 1$$
$$X\left(k + \frac{N}{2}\right) = DFT(x(2n)) + W_N^{-\left(k+\frac{N}{2}\right)} DFT(x(2n+1)),\ k = 0, 1, 2,..., \frac{N}{2} - 1 \quad (6)$$

Złożoność obliczeniowa (1) to $O(N^p)$ natomiast złożoność (6) jest mniejsza i wynosi $2*O((N/2)^2)$. Dokładny opis znajdziesz w TZ, podrozdział 9.5.1, równania (9.35)-(9.40).

I teraz wszystko staje się jasne ;-) Schemat przedstawiony od (2) do (6) można zastosować do (6) i dalej, głębiej, aż do momentu, gdy wektor wejściowy $\mathbf{x}$ będzie składał się tylko dwóch próbek.

Wygeneruj sygnał $\mathbf{x}$, losowy, o długości 1024 próbek. Oblicz $\mathbf{X}$ za pomocą funkcji DFT(...). Następnie wyznacz $\mathbf{X}_{fft}$ za pomocą (6): $\mathbf{X}_{fft}=\mathbf{X}_1+\mathbf{c}\mathbf{X}_2$, dodając do siebie osobno obliczone widma DFT(...) próbek parzystych i nieparzystych (te drugie z korektą **c**). Następnie widma $\mathbf{X}_1$ oraz $\mathbf{X}_2$ wyznacz ponownie za pomocą (2): $\mathbf{X}_1=\mathbf{X}_{11}+\mathbf{c}\mathbf{X}_{12}$, $\mathbf{X}_2=\mathbf{X}_{21}+\mathbf{c}\mathbf{X}_{22}$ (czyli podziel próbki o numerach parzystych na te o numerach parzystych i nieparzystych, podobnie zrób z próbkami o numerach nieparzystych). Zauważ, że $\mathbf{X}_{fft}$ oraz $\mathbf{X}$ ma długość 1024, $\mathbf{X}_1$ i $\mathbf{X}_2$ to wektory o długości 512 natomiast $\mathbf{X}_{11}$, $\mathbf{X}_{12}$, $\mathbf{X}_{21}$, $\mathbf{X}_{22}$ mają długość 256 próbek.

Porównaj czy wynik uzyskany we wszystkich 3 sposobach jest taki sam.

1b. FFT za pomocą rekurencji (2 pkt)

Złożoność obliczeniowa transformacji DFT N-punktowej to $O(N^p)$. Jedno zespolone N-punktowe DFT można wykonać jako złożenie dwóch $N/2$-punktowych DFT (plus pewne obliczenia związane ze ,,składaniem'') co skutkuje obniżeniem złożoności do $2*O((N/2)^2)$. Następnie można rozbić obliczenia na cztery $N/4$-punktowe DFT. Dla $N=2^p$ można zejść w ten sposób do wielu DFT o długości $N=2$.

Poniższa funkcja realizuje zespoloną transformację Fouriera poprzez podział w dziedzinie czasu DIT (ang. *Decimation i Time*).

```
function X = dit( x )

N = length(x);
x = x(:);                  % macierz wertykalna
X1 = fft( x(1:2:N) );      % próbki parzyste
X2 = fft( x(2:2:N) );      % próbki nieparzyste

X = zeros( size(x) );
k = (0:N/2-1)';
X(1:N/2) = X1 + exp( j*2*pi/N .*-k ) .* X2;
X(N/2+1:N) = X1 + exp( j*2*pi/N .* -(k+N/2) ) .* X2;
```

Działanie funkcji można zweryfikować programem (powyższą funkcję zapisz do pliku `dit.m`):

```
clear all;
close all;

N = 256;
x = randn( N, 1 );
X1 = fft(x);               % oryginalne DFT
X2 = dit(x);               % DFT ,,sklejane'' z dwóch połówek
mean( abs(X1-X2) )         % błąd
```

Funkcja `dit(...)` wykonuje tylko pierwszy z p etapów podziału. Każdy następny etap powinien być wykonany na zmiennej `X1` i `X2`. Wykorzystując `dit(...)` zaimplementuj algorytm radix-2 DIT FFT dla długości $N=2^p$. Wykorzystaj rekurencję.

2. Transformata Fouriera sygnałów rzeczywistych (1.5 pkt)

Przeanalizuj i uruchom kod programu:

```
% cps_06_fftapp1_start.m
clear all; close all; % "mycie rak"
fpr = 1000; % czestotliwosc probkowania (Hz)
N = 100; % liczba probek sygnalu, 100 lub 1000
dt=1/fpr; t=dt*(0:N-1); % chwile probkowania sygnalu, os czasu

% Signal
f0=50; x = sin(2*pi*f0*t); % sygnal o czestotliwosciach f0 = 50,100,125,200 Hz
figure; plot(t,x,'bo-'); xlabel('t [s]'); title('x(t)'); grid; pause

% FFT spectrum
X = fft(x); % FFT
f = fpr/N *(0:N-1); % os częstotliwosci
figure; plot(f,1/N*abs(X),'bo-'); xlabel('f [Hz]'); title('|X(k)|'); grid; pause
```

Sprawdź wartości amplitudy i częstotliwości sygnału wygenerowanego w programie i spróbuj je odczytać z rysunku widma FFT sygnału (znaleźć je na nim). Zwróć uwagę na opis osi częstotliwości oraz na "sprzężoną", hermitowską symetrię widma względem częstotliwości $\frac{fpr}{2}$ - oś symetrii

$$X\left(\frac{fpr}{2}+f0\right)=X^*\left(\frac{fpr}{2}-f0\right),\ X\left(\frac{N}{2}+f0\right)=X^*\left(\frac{N}{2}-k\right)$$

(w Matlabie centrum symetrii jest w punkcie $x(N/2+1)$, dlatego, np. $x(N)=conj(x(2))$).

Wyjaśnij pochodzenie tej symetrii (przypomnij sobie szczegóły dotyczące powtarzania się funkcji bazowych w macierzy transformacji DFT).

Zmień amplitudę sygnału na 10, 100, 1000 oraz obserwuj wartości modułu widma. Dlaczego wartości maksimów widma są dwa razy niższe niż spodziewałeś się? (Przypomnij sobie równość: $cos(\alpha) = 0.5e^{j\alpha} + 0.5e^{-j\alpha}$. Jaki wzór jest dla sinusa?)

Zmień wartość częstotliwości sygnału f0 na 50, 100, 125, 200 Hz oraz obserwuj wartości argumentów maksimów widma. Czy to są te same wartości? Wytłumacz pochodzenie rozmycia widma dla sygnału o częstotliwości f0=125 (przypomnij sobie znaczenie funkcji okien: jakiego okna teraz używamy?).

Pokaż tylko pierwszą połowę (1/2) widma i poprawnie wyskaluj jego moduł (*2/N):

`k=1:N/2+1; plot(f(k),2/N*abs(X(k)));` Zmień wartość `N` z `100` na `1000`: obejrzyj widma FFT dla `f0 = 50,100,125,200` Hz; Wyskaluj ostatni moduł widma FFT w decybelach `X=20*log10(2/N*abs(X(k)))` oraz obejrzyj go dla `N=100` oraz `N=1000`;

3. Transformata Fouriera sygnałów rzeczywistych (1.5 pkt)

Transformata DFT jest zespolona, jednak często transformacji poddaje się sygnały rzeczywiste np.: dźwięk, zdjęcia cyfrowe, etc... Dlatego, aby jeszcze bardziej przyspieszyć działanie tego algorytmu można wykorzystać symetrię widma i:

1. wykonać dwie N-punktowe transformacje rzeczywistego sygnału za pomocą jednej N-punktowej transformaty zespolonej lub
2. wykonać jedną N-punktową transformacje rzeczywistego sygnału za pomocą N/2-punktowej transformaty zespolonej.

Jeżeli przedostatnia cyfra numeru Twojej legitymacji studenckiej jest liczbą nieparzystą wykonaj pierwszy punkt, jeżeli jest parzysta wybierz punkt drugi. Przyjmij N=1024, wygeneruj losowe dane o rozkładzie normalnym, wykonaj transformatę rzeczywistą a następnie porównaj otrzymane wyniki do transformaty zespolonej. Jakie przyspieszenie algorytmu uzyskałeś tą metodą?

Dodatkowe przyspieszenie transformacji Fouriera sygnałów rzeczywistych wykorzystuje właściwość symetrii widma: dla rzeczywistego sygnału $x(n)$, $n=0,1,2,\ldots,N\text{-}1$, jego transformata $X(k)$, $k=0,1,2,\ldots,N\text{-}1$ ma następującą właściwość:

$$X(k) = X^*(N - k),\ k = 1, 2, \ldots, N - 1$$

czyli:

$$\mathrm{Re}\{X(k)\} = \mathrm{Re}\{X(N\text{-}k)\},\ \mathrm{Im}\{X(k)) = \text{-}\mathrm{Im}(X(N\text{-}k)),\ k=1,2,...,N\text{-}1.$$

Dlatego, dla dwóch niezależnych sygnałów (**punkt 1**) $x_1(n)$ i $x_2(n)$, $n=0,1,2,...,N\text{-}1$, można uzyskać ich transformaty Fouriera w następujący sposób:

$y(n)=x_1(n)+jx_2(n)$

$\mathbf{Y}=fft(\mathbf{y})$

wykorzystując symetrię widma względem $k=N/2$ można ,,odzyskać'' transformaty Fouriera sygnałów $\mathbf{x_1}$ i $\mathbf{x_2}$ w następujący sposób:

$X_{1r}(k) = 0.5*(Y_r(k)+Y_r(N\text{-}k)),\ k=1,2,3,...,N\text{-}1,$ $X_{1i}(k) = 0.5*(Y_i(k)\text{-}Y_i(N\text{-}k)),\ k=1,2,3,...,N\text{-}1$

$X_{2r}(k) = 0.5*(Y_i(k)+Y_i(N\text{-}k)),\ k=1,2,3,...,N\text{-}1,$ $X_{2i}(k) = 0.5*(Y_r(N\text{-}k)\text{-}Y_r(k)),\ k=1,2,3,...,N\text{-}1$

gdzie $\mathbf{Y_r}$ i $\mathbf{Y_i}$ to odpowiednio część rzeczywista i urojona wektora $\mathbf{Y}$, podobnie jak $\mathbf{X_{1r}}$, $\mathbf{X_{1i}}$, $\mathbf{X_{2r}}$ i $\mathbf{X_{2i}}$ są odpowiednio częścią rzeczywistą i urojoną wektorów $\mathbf{X_1}$ i $\mathbf{X_2}$. Zerowe prążki widma odzyskuje się następująco:

$X_{1r}(0)=Y_r(0),$ $X_{1i}(0)=0$

$X_{2r}(0)=Y_i(0),$ $X_{2i}(0)=0$

Ostatecznie, transformaty Fouriera wektorów $\mathbf{x_1}$ i $\mathbf{x_2}$ uzyskuje się łącząc wektory:

$\mathbf{X_1}=\mathbf{X_{1r}}+j\mathbf{X_{1i}}$

$\mathbf{X_2}=\mathbf{X_{2r}}+j\mathbf{X_{2i}}$

Wersja alternatywna (**punkt 2**) polega na wyznaczeniu N-punktowego sygnału rzeczywistego za pomocą pojedynczej N/2-punktowej FFT: niech $y(n)=x(2n)+jx(2n+1)$, $n=0,1,2,...,N/2\text{-}1$ a $\mathbf{Y}$ to transformata Fouriera $\mathbf{y}$ o długości $N/2$. Wtedy operację $\mathbf{X}=fft(\mathbf{x})$ można otrzymać w dwóch krokach:

1) $$X(k)=\frac{1}{2}\left[Y(k)+Y^*\left(\frac{N}{2}-k\right)\right]+\frac{1}{2}je^{-\frac{j2\pi k}{N}}\left[Y^*\left(\frac{N}{2}-k\right)-Y(k)\right]$$, k=1,2,3,...,N/2-1.

Widmo dla $k=N/2,N/2+1,...,N\text{-}1$ jest sprzężone, symetryczne do pierwszej połówki. Załóż, że $Y(512)=Re(Y(0))-Im(Y(0))$.

2) $$X(0)=Re(Y(0))+Im(Y(0))$$

4. Implementacja algorytmu radix-2 (opcja, +1 pkt)

Wygeneruj w Matlabie zespolony wektor **x** o rozmiarze 1x1024, charakterze losowym (rozkład normalny) oraz nieskorelowanej części rzeczywistej i urojonej. Zapisz go w formacie Matlaba do pliku x.mat oraz w formacie w który odczytasz w języku C/C++ (plik xcpp.dat).

Napisz program (w języku Matlab) o nazwie myFFT, który wczyta x.mat oraz xcpp.dat wykona transformatę Fouriera obu sygnałów uzyskując $\mathbf{X_1}$ oraz $\mathbf{X_2}$. Porównaj oba wyniki. Jeżeli są istotnie różne to oznacza, że utraciłeś część informacji podczas zapisu do pliku xcpp.dat – skoryguj ten błąd.

Napisz program w języku C/C++ implementujący szybką transformatę Fouriera (FFT) za pomocą algorytmu *radix-2* w dwóch wersjach – na zmiennych typu float oraz double. Następnie wczytaj sygnał **x** z pliku xcpp.dat, wykonaj transformację sygnału w precyzji float oraz double, zapisz wyniki w oddzielnych plikach, w formacie który będziesz mógł odczytać w środowisku Matlab.

W języku Matlab, wczytaj transformacje wykonane w języku C/C++ i porównaj wynik do wzorcowej implementacji transformaty Fouriera wykonanej w języku Matlab (wektor $\mathbf{X_1}$).

Wykorzystaj opis algorytmu *radix-2* z wykładu.

5. STFT - spectrogram FFT sygnałów zmiennych w czasie (opcja, +0.5 pkt)

Przeanalizuj kod programu `Listing 6_3.m` oraz zaobserwuj jak jest wyznaczana i pokazywana w nim macierz krótko-czasowej transformacji Fouriera (STFT) $|X(t, f)|$ (X1 w programie). Ustaw `x=x2` (sygnał LFM) oraz `x=x3` (sygnał SFM). Uruchom program. Obejrzyj kolorowy obraz czasowo-częstotliwościowej macierzy STFT. Spróbuj odczytać z niego informację, dotyczącą zmian częstotliwości analizowanego sygnału w funkcji czasu: od-do w hercach dla sygnału LFM `x2` oraz częstotliwość środkowa oraz głębokość modulacji dla sygnału SFM `x3`. Podczas analizy sygnału SFM `x=x3` użyj okien o różnej długości `Mwind`: bardzo krótkich (zła rozdzielczość częstotliwościowa - rozmycie w osi częstotliwości) oraz bardzo długich (zła rozdzielczość czasowa - rozmycie w osi czasu) — zauważ znacząco różne kształty widm w obu przypadkach. Napisz swoją własną funkcję `myspectrogram()`, mającą takie same parametry wywołania jak funkcja Matlaba `spectrogram()`.