ANALIZA CZĘSTOTLIWOŚCIOWA

na procesorach sygnałowych

Opracowanie: Grzegorz Szwoch

Politechnika Gdańska, Katedra Systemów Multimedialnych

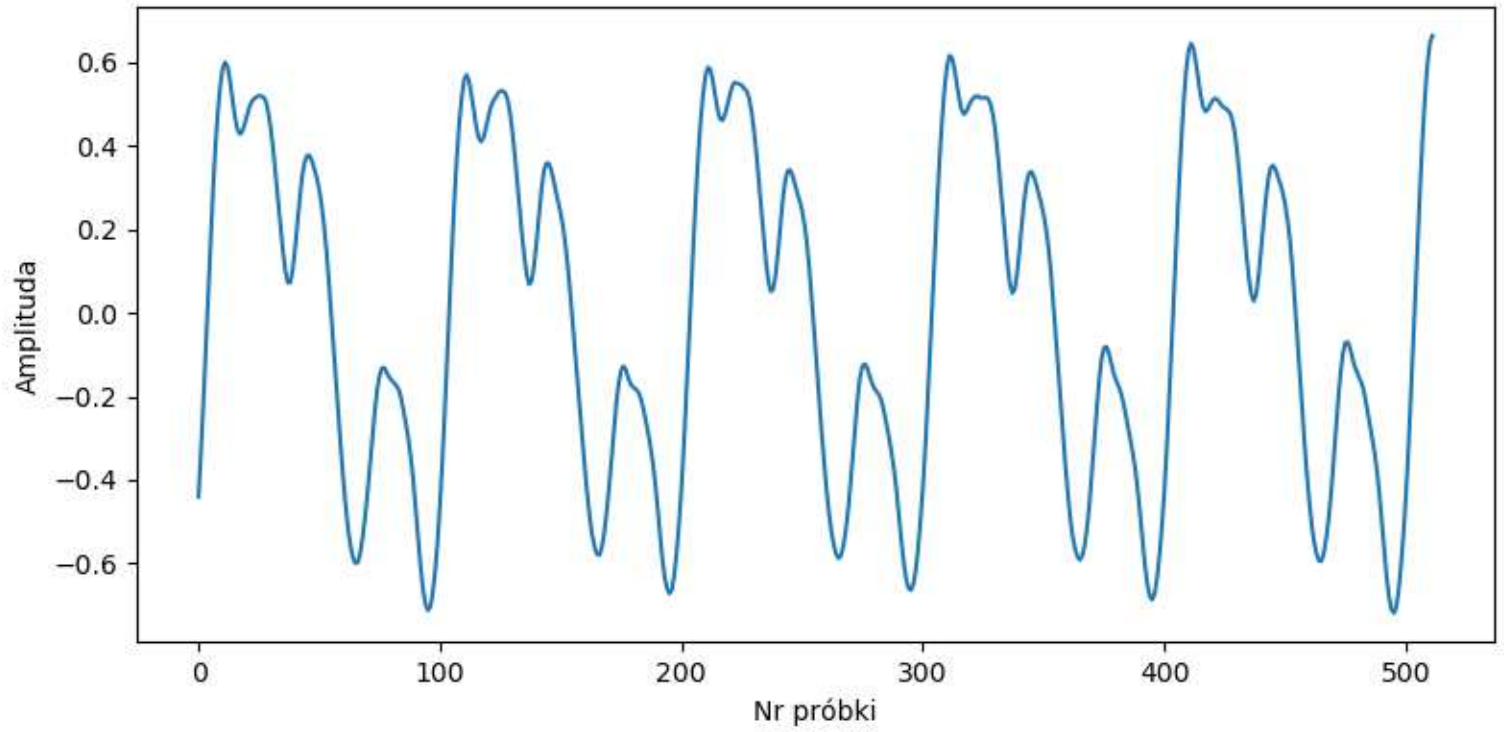
Wprowadzenie

- Próbki określają wartości sygnału cyfrowego w dziedzinie czasu.
- Wiele operacji cyfrowego przetwarzania sygnałów musi być wykonanych w dziedzinie częstotliwości. Nie chcemy modyfikować całego sygnału, a jedynie wybrane składowe.
- Analiza częstotliwościowa (widmowa) – określenie składowych częstotliwościowych, z których zbudowany jest sygnał.

Przykład

Nagranie dźwięku klarnetu – wykres czasowy.

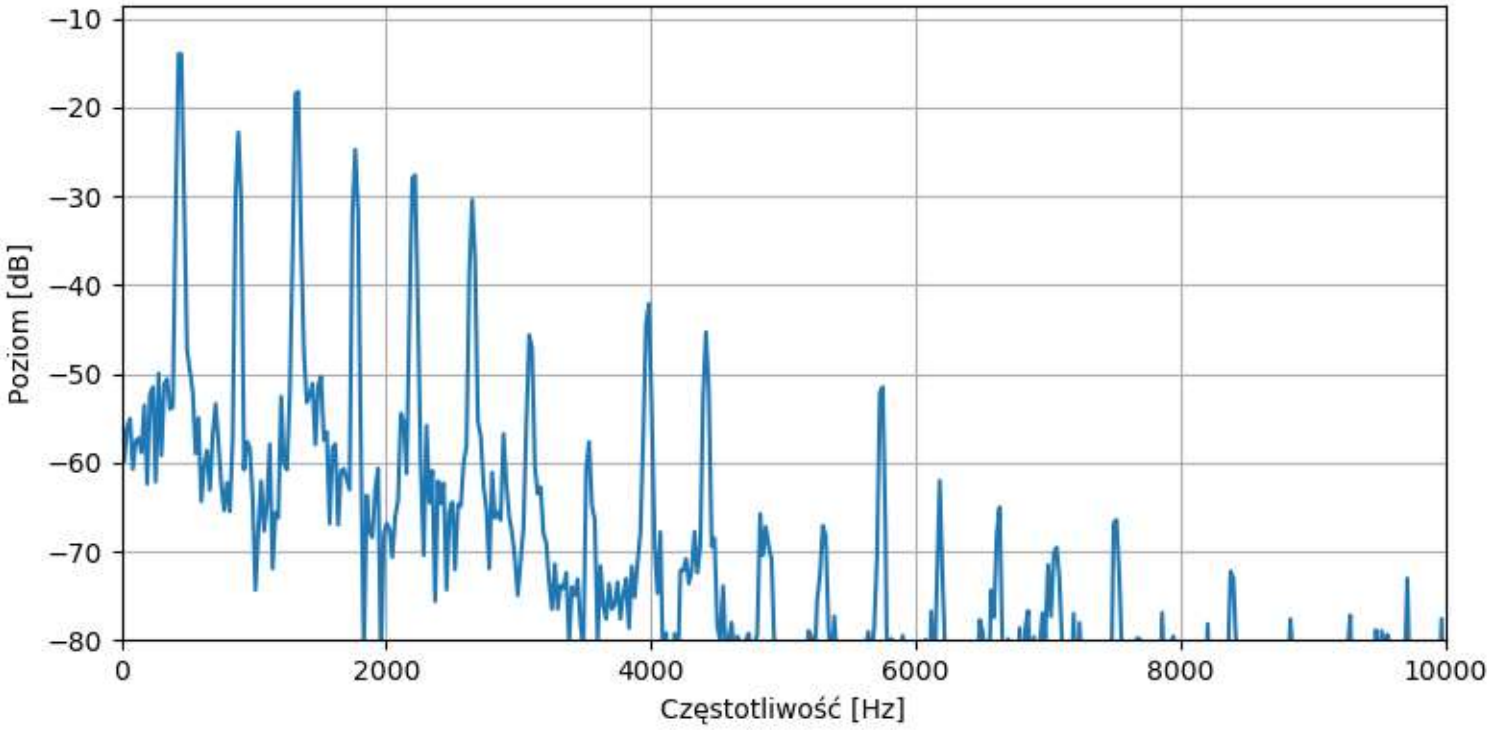
Nie dowiemy się w ten sposób jaka jest struktura sygnału.



Przykład

Wynik analizy częstotliwościowej dźwięku klarnetu.

- Widzimy strukturę sygnału – suma harmonicznych (sinusów).
- Możemy określić wysokość dźwięku: wynika z częstotliwości pierwszej składowej.



Przekształcenie Fouriera

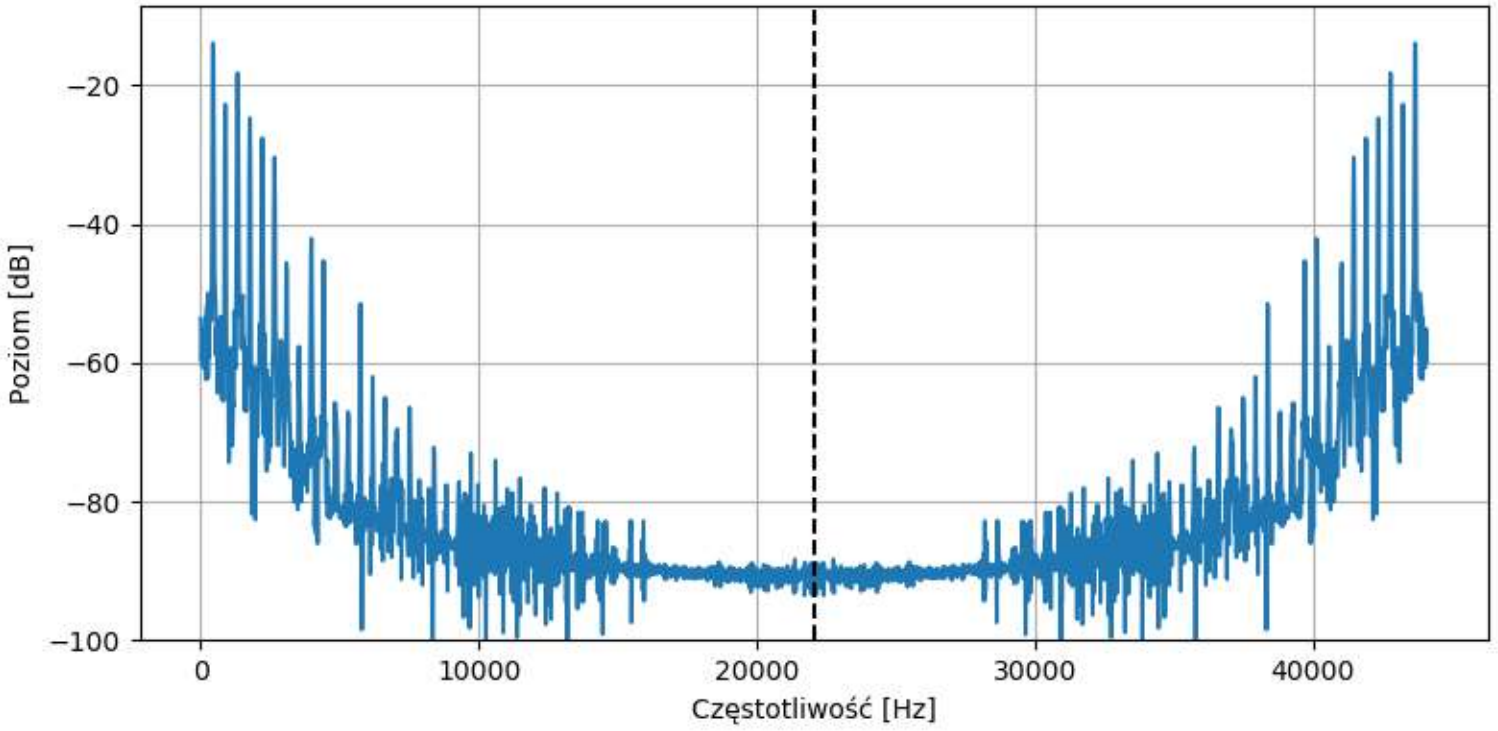
- Operacja przekształcenia (transformacji) Fouriera przekształca *N* próbek sygnału w *N* próbek widma.
- Transformata Fouriera dla sygnału określa widmo (spektrum) sygnału – przez analogię do widma światła.
- Odwrotne przekształcenie Fouriera – w drugą stronę, z widma do próbek sygnału.
- Można przekształcić sygnał w widmo, przetworzyć je i otrzymać próbki przetworzonego sygnału.
  Jest to przetwarzanie częstotliwościowe (*spectral processing*).

Widmo sygnału rzeczywistego

- Przekształcenie Fouriera działa dla sygnałów zespolonych i rzeczywistych.
- Widmo jest zawsze zbiorem liczb zespolonych.
- Zazwyczaj przekształcamy sygnały rzeczywiste.
- Znaczenie *N* próbek widma sygnału rzeczywistego:
  - wartość 1: składowa stała, suma wartości próbek
  - wartości od 2 do $N/2$: składowe widma sygnału
  - wartość $N/2 + 1$: składowa Nyquista, powinna być zerowa
  - wartości od $N/2 + 2$ do N; lustrzana kopia pierwszej połowy widma (symetria hermitowska).
- Widmo z *N* próbek ma $(N/2 + 1)$ unikalnych wartości.

Widmo sygnału rzeczywistego

Spośród *N* wartości widma potrzebujemy (*N*/2 + 1) wartości.



Widmo amplitudowe

- Widmo $X(f)$ zawiera wartości zespolone.
- Najczęściej interesuje nas widmo amplitudowe $A(f)$, czyli moduł widma zespolonego:

$$A(f) = |X(f)|$$

- Widmo mocy sygnału jest kwadratem modułu widma:

$$P(f) = |X(f)|^2$$

- Widma amplitudowe i mocy często wyrażamy w skali logarytmicznej, w decybelach (dB):

$$A(f) = 10\log_{10}|X(f)|$$

$$P(f) = 10\log_{10}|X(f)|^2 = 20\log_{10}|X(f)|$$

Widmo amplitudowe

- Aby obliczyć amplitudę składowej widmowej, należy podzielić moduł widma przez liczbę próbek i uwzględnić fakt, że widmo sygnału rzeczywistego rozkłada się na „dwie połowy”.
- Amplituda składowej o indeksie *n*:

$$A[n] = \frac{2}{N}\left|X[n]\right|$$

- Nie dotyczy pierwszej wartości (składowej stałej) oraz składowej Nyquista – te dwie nie mają pary.
- Wartość składowej stałej podzielona przez *N*
  = wartość średnia sygnału w analizowanym okresie.

Częstotliwości próbek widma

- Na podstawie *N* próbek sygnału obliczamy *N* próbek widma.
- Widmo pokrywa zakres częstotliwości [0, f_S].
- Zatem *n*-ta wartość widma odpowiada częstotliwości:

$$f[n] = n\frac{f_S}{N}$$

- Odstęp między próbkami widma jest równy stosunkowi częstotliwości próbkowania do liczby próbek widma.
- Wyznacza on rozdzielczość częstotliwościową analizy widmowej:

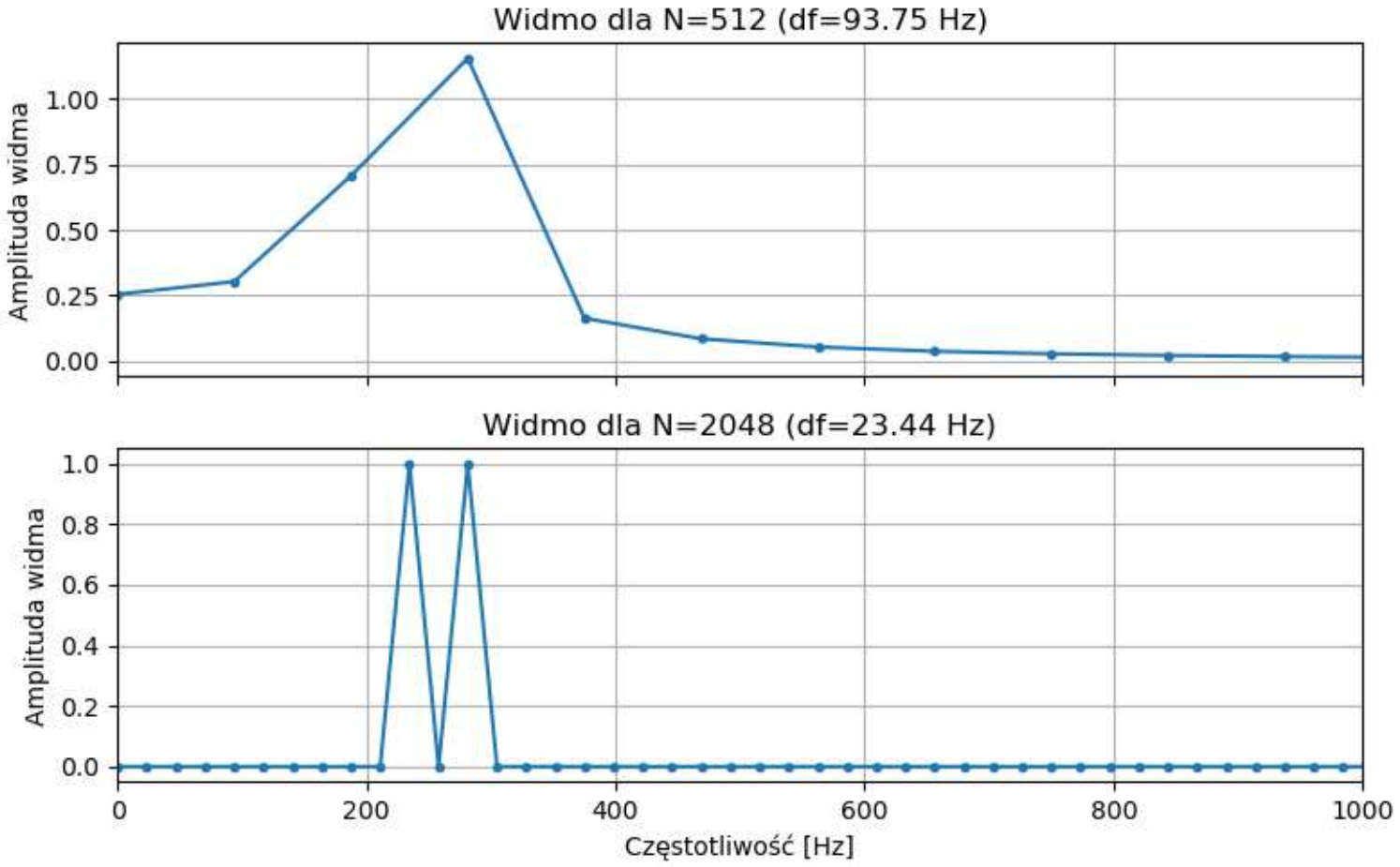
$$df = \frac{f_S}{N}$$

Rozdzielczość częstotliwościowa

- Znaczenie rozdzielczości częstotliwościowej ($df = f_S / N$):
  - nie można rozróżnić dwóch składowych widmowych, jeżeli odstęp między nimi jest mniejszy niż *df* („wpadają” one do tego samego przedziału analizy),
  - niedokładność (błąd) określania częstotliwości składowej sygnału wynosi maksymalnie $\pm df/2$.
- Większa liczba próbek sygnału (*N*) zwiększa rozdzielczość.
- Można sztucznie poprawić rozdzielczość uzupełniając sygnał zerami na końcu (*zero padding*). Nie dostajemy w ten sposób więcej danych (wartości są interpolowane), ale mamy lepszą rozdzielczość.

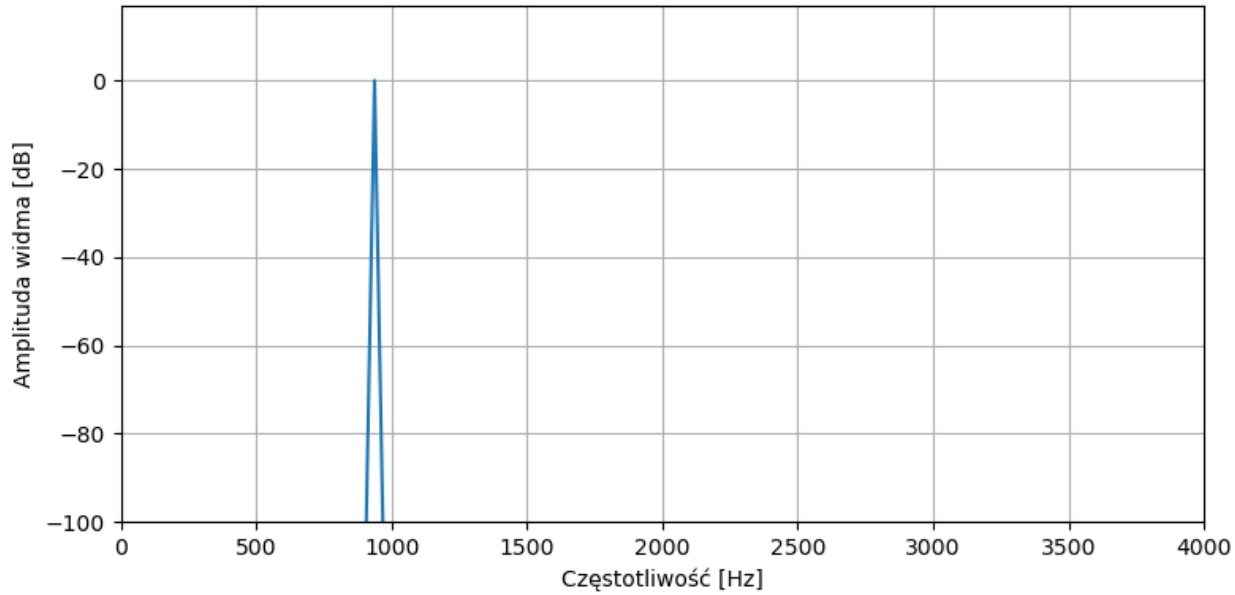
Rozdzielczość częstotliwościowa

Przykład: suma sinusów *f*1 = 234,375 i f2 = 281,25 Hz



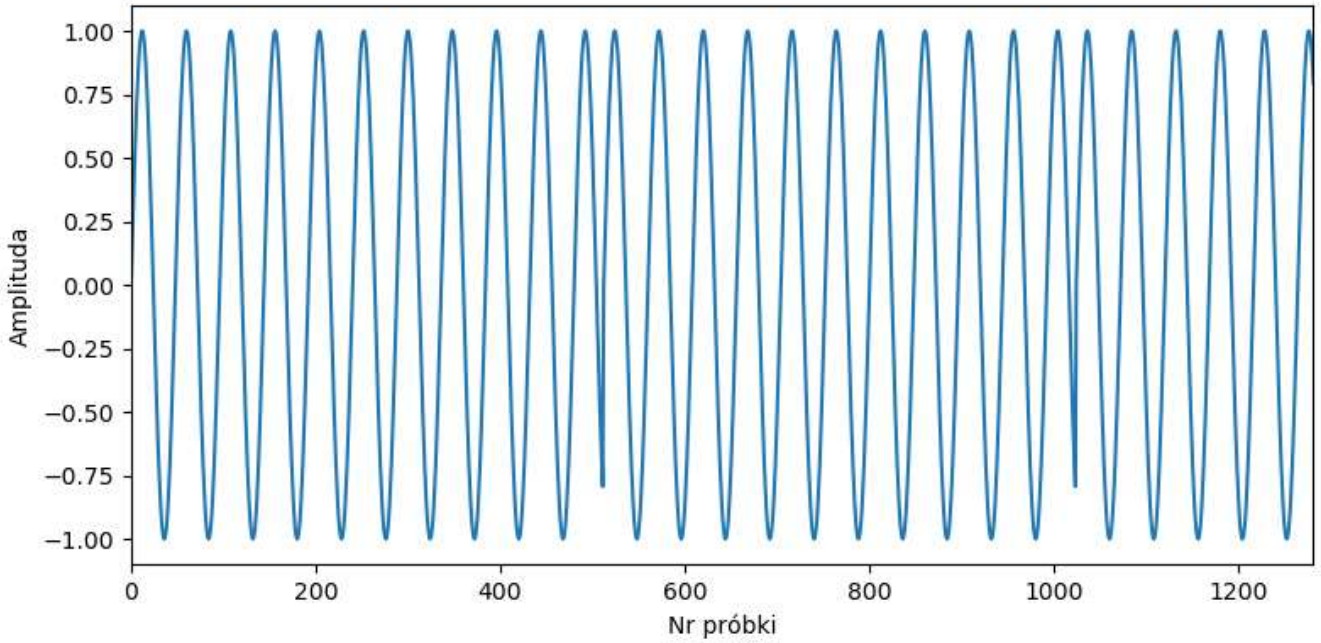
Okresowość sygnału

- Przekształcenie Fouriera zakłada, że sygnał jest okresowy.
- Fragment sygnału poddawany przekształceniu jest traktowany jako okres sygnału.
- Przykład widma, gdy długość okna analizy jest wielokrotnością okresu sygnału (f = 937,5 Hz; N = 512):



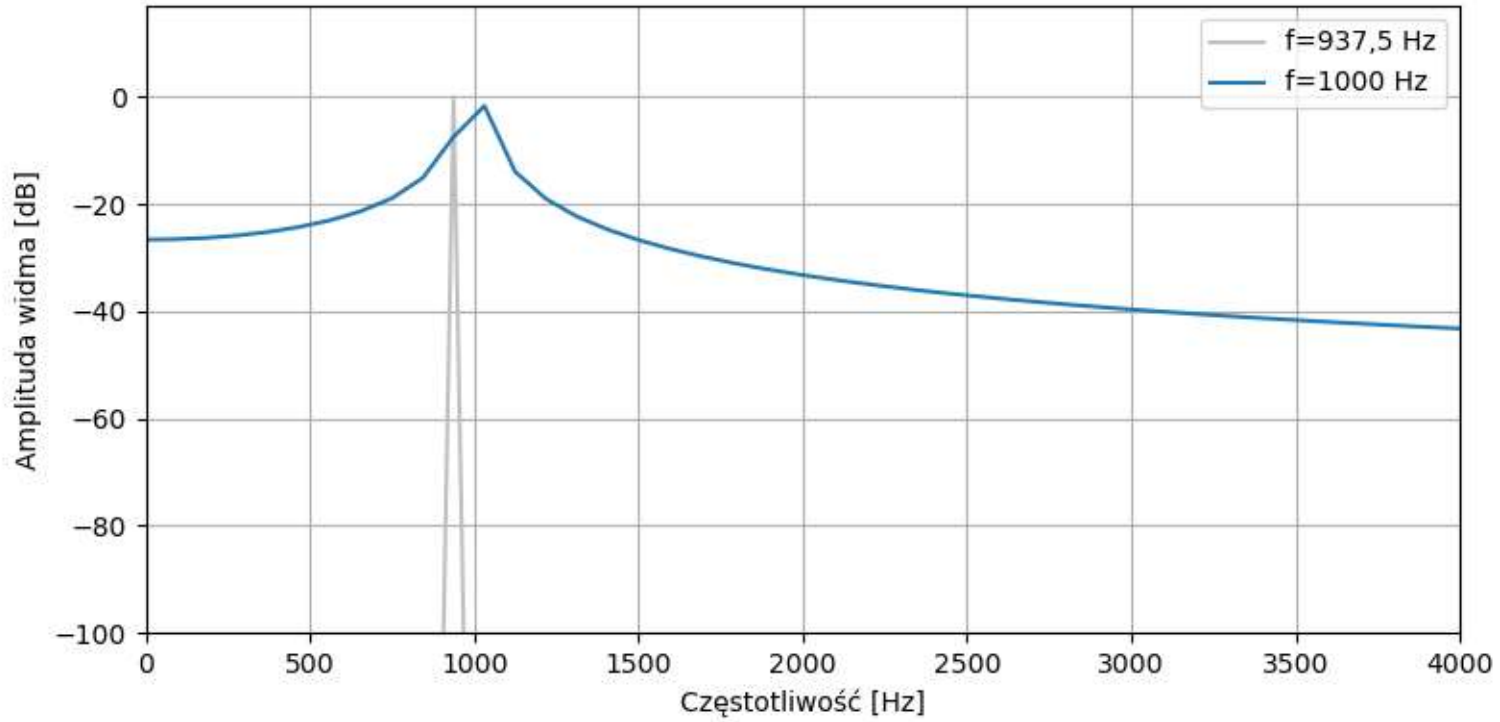
Okresowość sygnału

- Co w przypadku, gdy długość okna analizy nie jest wielokrotnością okresu sygnału?
- Wtedy obliczana jest transformata sygnału będącego „zapętleniem” okna analizy.
- Przykład dla f = 1000 Hz, N = 512:



Przecieki widma

Jak wygląda widmo takiego sygnału?

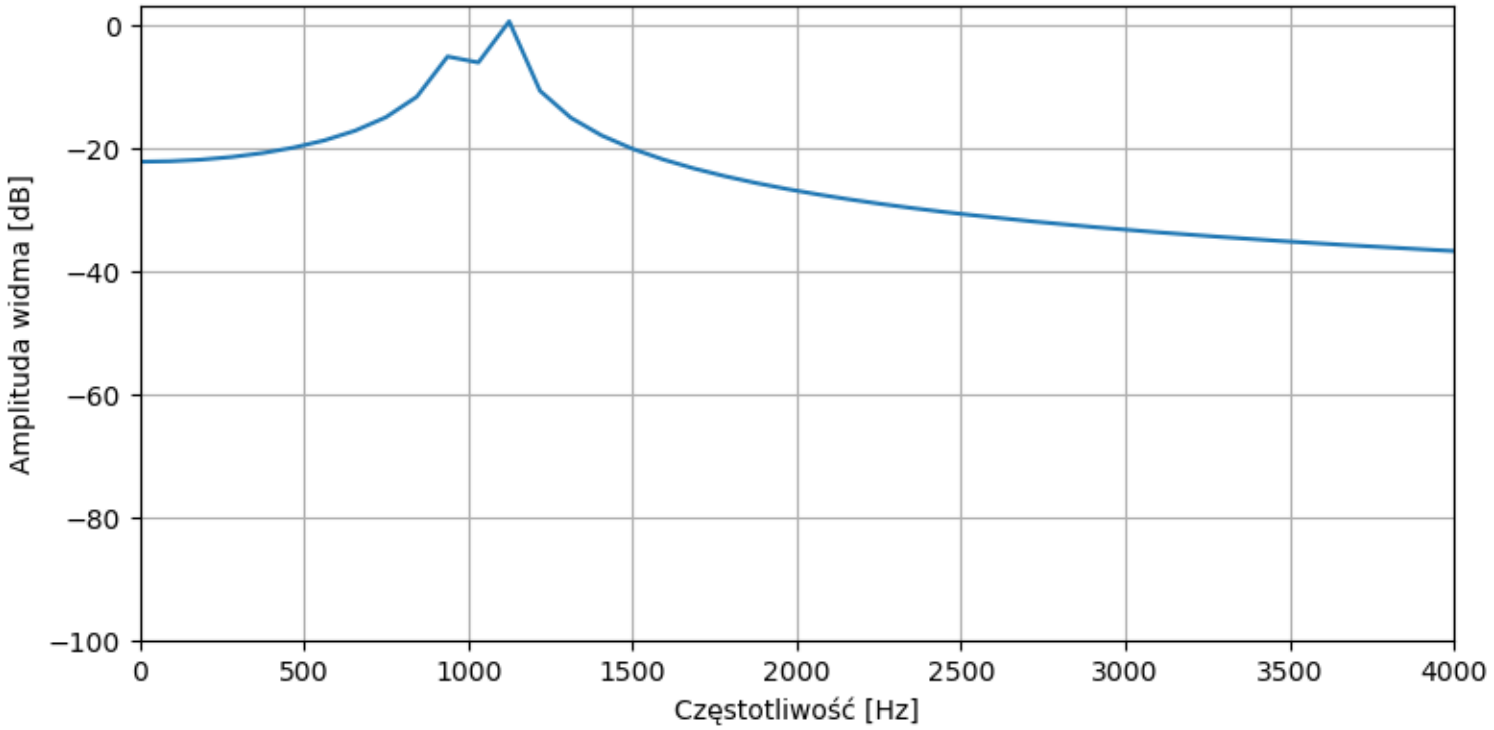


Energia widma „rozlewa się” na sąsiednie wartości. Nazywamy to przeciekami widmowymi (*spectral leakage*).

Przecieki widma

Przykład: suma sinusów 1000 Hz i 1100 Hz (*N* = 512).

Przecieki widma zacierają kształt składowych.

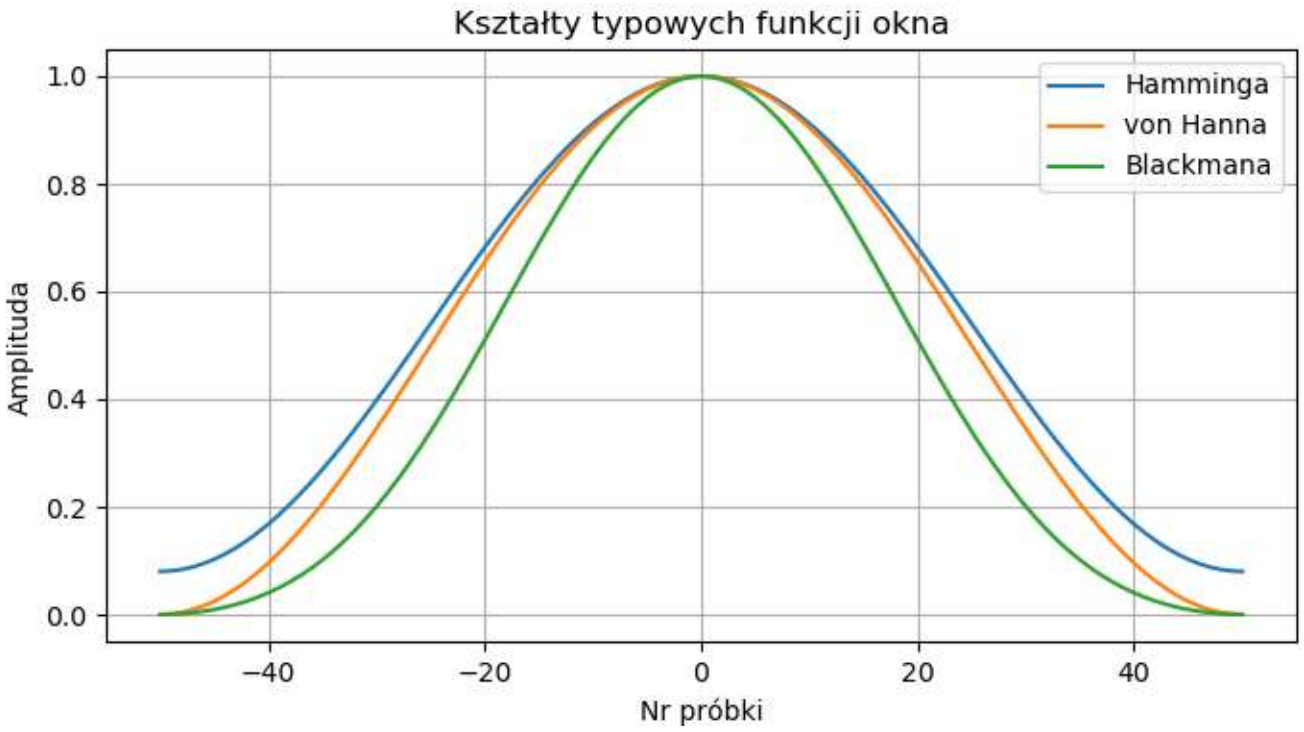


Funkcje okien czasowych

- Przecieki widma wynikają z nieciągłości przy zapętleniu okna analizy.
- Zmniejszenie przecieków widma uzyskujemy poprzez przemnożenie okna analizy przez funkcję okna czasowego.
- Funkcja okna tłumi skrajne wartości okna analizy.
- Skutek działania funkcji okna:
  - redukuje „rozlewanie się” amplitudy składowych na sąsiednie wartości widmowe,
  - ale jednocześnie poszerza główne prążki widmowe – efekt przecieku skupia się w wąskim zakresie.

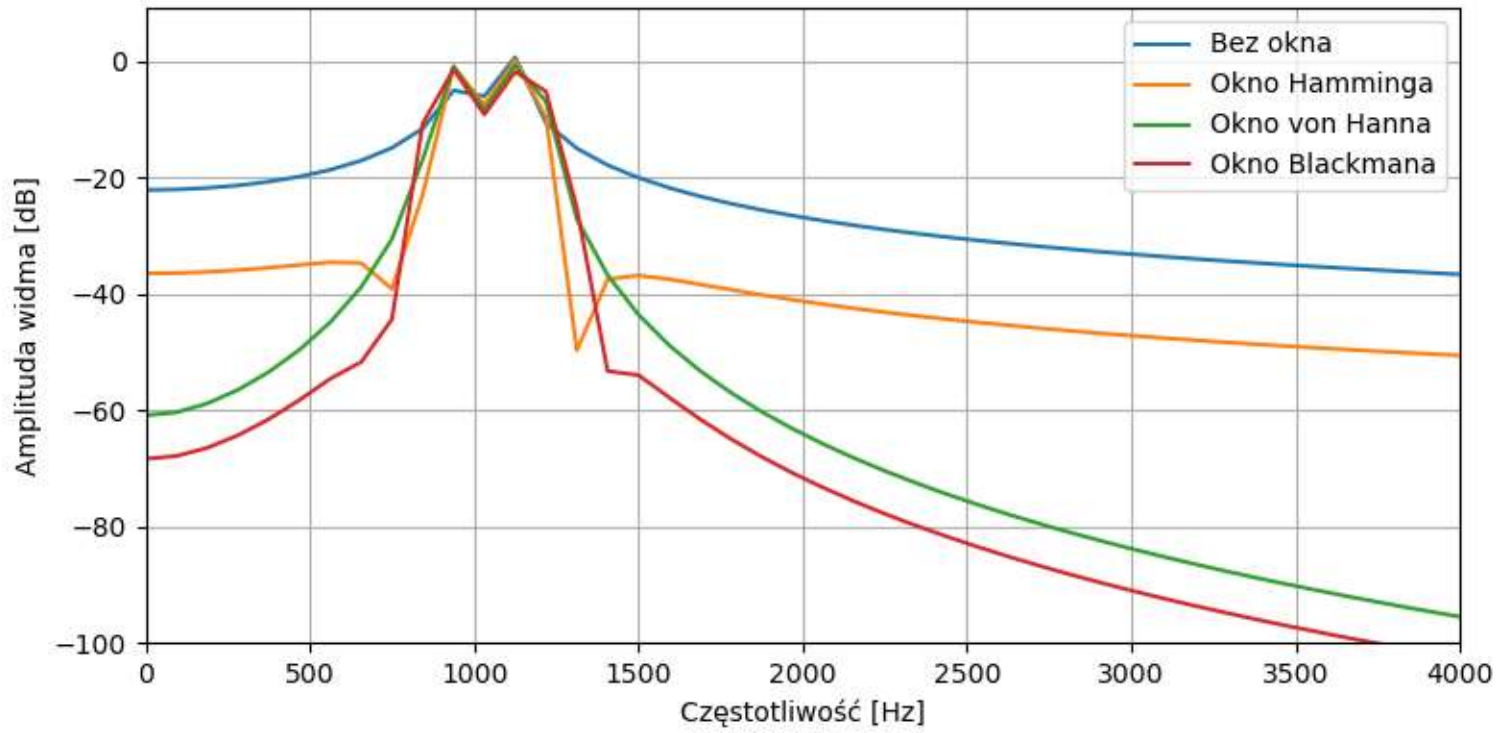
Funkcje okien czasowych

Najczęściej używane funkcje okna:



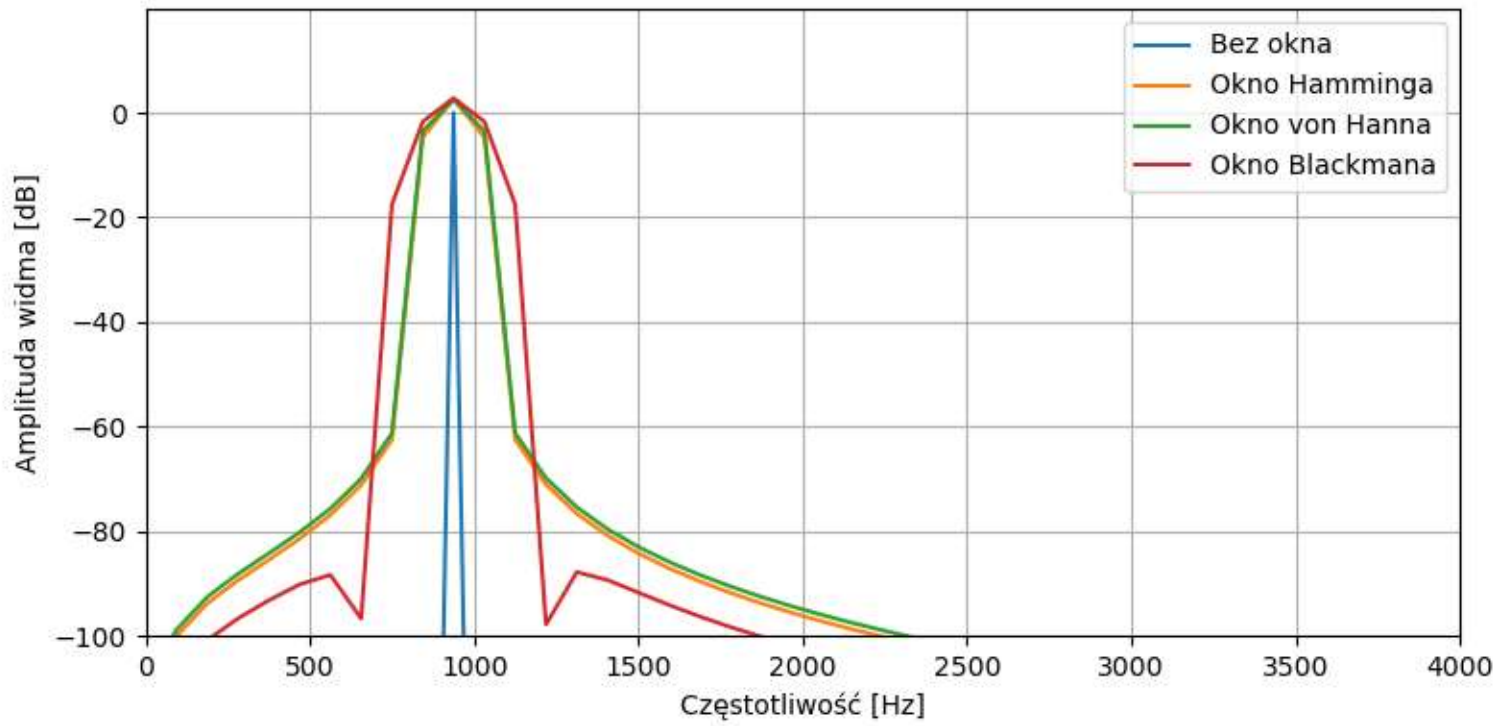
Wpływ okna czasowego

Zastosowanie funkcji okna w przypadku występowania przecieków widmowych – poprawa wyników analizy.



Wpływ okna analizy

Zastosowanie funkcji okna w przypadku braku przecieków widmowych – tutaj funkcje okna pogarszają wyniki.



Uwagi o funkcjach okna

- Nie ma „najlepszego” okna, ale okna Hamminga i von Hanna są najbardziej uniwersalne.
- Okno Blackmana przyda się gdy trzeba silnie stłumić przecieki, kosztem poszerzenia maksimów widmowych.
- Stosujemy funkcje okna gdy analizujemy widmo, np. szukamy maksimów.
- Nie stosujemy funkcji okna przy przetwarzaniu w dziedzinie częstotliwości, gdy później wracamy do dziedziny czasu.
- Normalizacja amplitudy widma przy stosowaniu okna *w*:

$$A[n] = \frac{2}{\sum w_i^2} \left| X[n] \right|$$

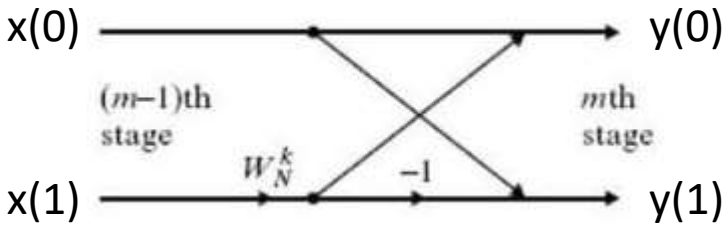
Obliczanie przekształcenia Fouriera

Są dwie metody obliczenia widma przez przekształcenie Fouriera:

1. Z definicji dyskretnego przekształcenia Fouriera (DFT):
   - działa dla każdego sygnału,
   - wymaga dużej liczby operacji mnożenia i dodawania.
2. Za pomocą algorytmu Cooleya-Tukeya, nazywanego szybkim przekształceniem Fouriera (FFT – *Fast Fourier Transform*):
   - zredukowana liczba operacji,
   - ograniczenia co do długości okna analizy,
   - metoda stosowana w procesorach sygnałowych.

FFT

Przekształcenie Fouriera dla dwóch próbek sygnału:



$$y(0) = x(0) + x(1) \cdot W_n^k$$

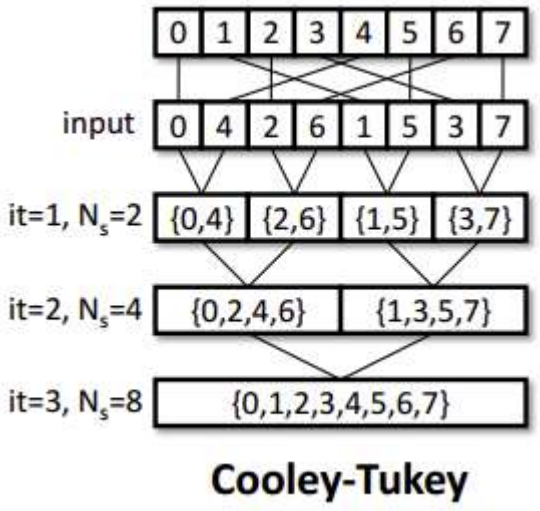
$$y(1) = x(0) - x(1) \cdot W_n^k$$

- Musimy wykonać: jedno mnożenie, jedno dodawanie, jedno odejmowanie.
- Taką strukturę nazywa się potocznie motylkiem (*butterfly*).
- Jest to składowy element FFT o podstawie 2 (*radix-2*).

FFT

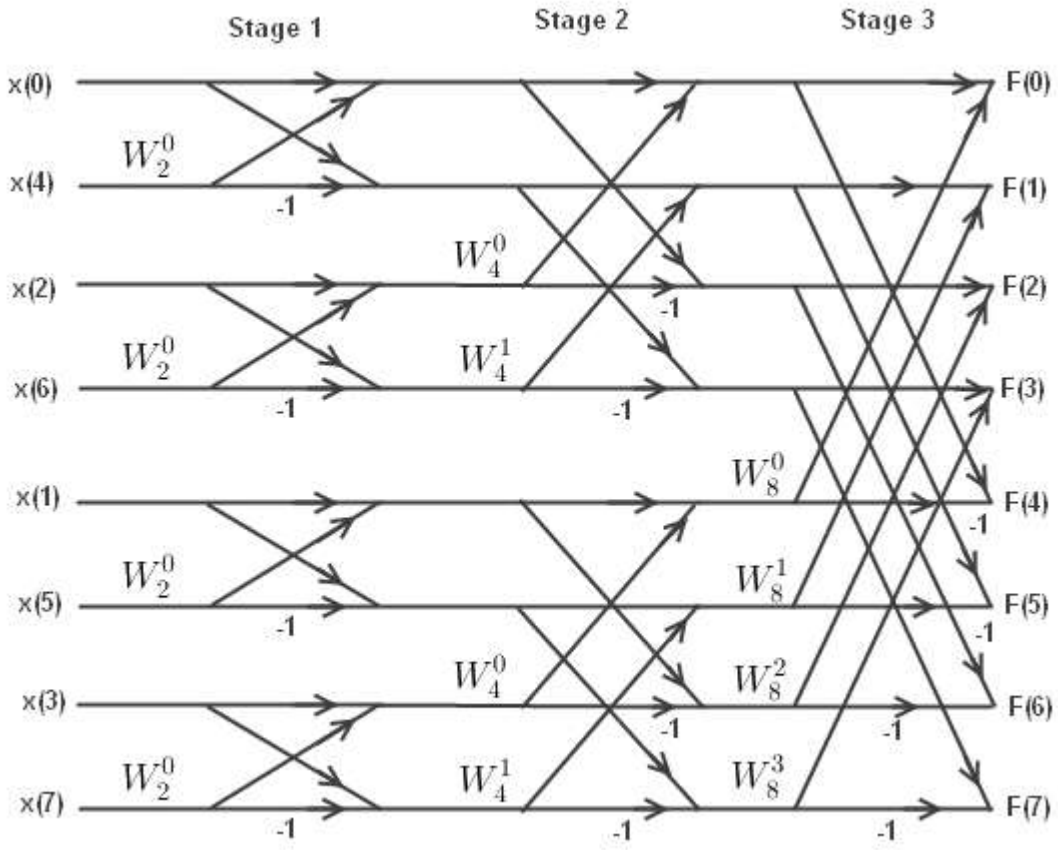
Algorytm Cooleya-Tukeya o podstawie 2:

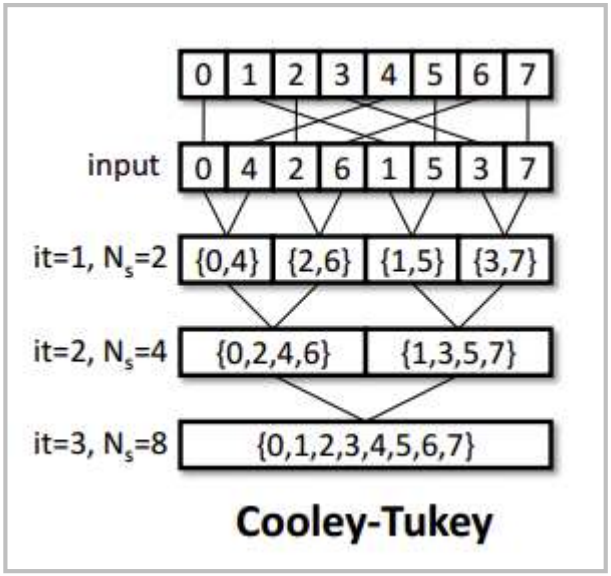
- Sygnał musi mieć długość 2^N.
- Dzielimy sygnał na dwie części, przydzielając próbki na zmianę do każdej części.
- Powtarzamy dla każdej części, aż dojdziemy do długości 2.
- Obliczamy „motylka” dla każdej części.
- Składamy wyniki w pary, obliczamy nowe „motylki”.
- Powtarzamy, aż złożymy całe widmo.



Cooley-Tukey

FFT - przykład dla $N = 8 = 2^3$





FFT – współczynniki twiddle

- Współczynniki *W* mają postać:

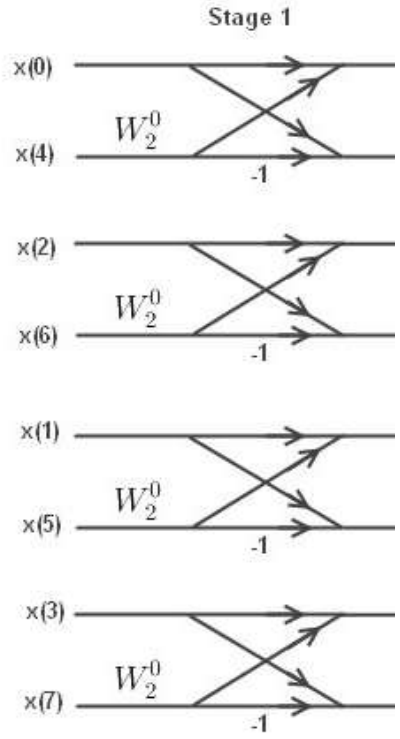
$$W_N^k = e^{-j2\pi k/N} = \cos\left(\frac{2\pi k}{N}\right) - j\sin\left(\frac{2\pi k}{N}\right)$$

- Są to *twiddle factors* („współczynniki skręcenia”).
- *N* jest rozmiarem transformaty na danym etapie.
- Dla *i*-tego etapu potrzeba 2^{i-1} współczynników.
- Współczynniki są stałe dla danego rozmiaru transformaty. Są one zatem obliczane wcześniej i zapisywane w tablicy.

Odwrócenie bitowe

- Aby przygotować pierwszy etap obliczania FFT, należy ustawić próbki we właściwej kolejności.
- Wystarczy odwrócić kolejność bitów w reprezentacji dwójkowej indeksów próbek – *bit reversal*.

0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7



Rozmiar transformaty

- Klasyczny algorytm FFT wymaga aby długość okna analizy była potęgą dwójki. Dobrze jest trzymać się tej konwencji.
- Typowe długości transformaty: 512, 1024, 2048.
- Jeżeli nie mamy wystarczającej liczby próbek, możemy uzupełnić je zerami do najbliższej potęgi dwójki. Nie należy robić tego bez potrzeby, tracimy czas na obliczanie zerowych próbek.

Współczesne implementacje FFT

Praktycznie stosowane biblioteki FFT, np. FFTW:

- Zaimplementowane algorytmy FFT o różnych podstawach (różnych rozmiarach składowego „motylka”), np.: 2, 3, 5, 7, 11, 13, 17, 19.
- Żądany rozmiar transformaty jest rozkładany na czynniki pierwsze, np.: $2016 = 2^5 \cdot 3^2 \cdot 7$.
- Sygnał jest dzielony na części i obliczane są cząstkowe FFT, w przykładzie: o podstawie 2, 3 i 7. Wyniki są składane razem (algorytm *split radix*).
- Jeżeli któraś z części jest dużą liczbą pierwszą, dla niej obliczane jest wolne DFT, czego należy unikać.

Porównanie złożoności: FFT i DFT

Dane z: Mark McKeown, FFT Implementation on the TMS320VC5505, TMS320C5505, and TMS320C5515 DSPs (SPRABB6A)

FFT Length	Direct DFT Computation		Radix-2 FFT	
	Complex Multiplications	Complex Additions	Complex Multiplications	Complex Additions
128	16,384	16,256	448	896
256	65,536	65,280	1,024	2,048
512	262,144	264,632	2,304	4,608
1024	1,048,576	1,047,552	5,120	10,240

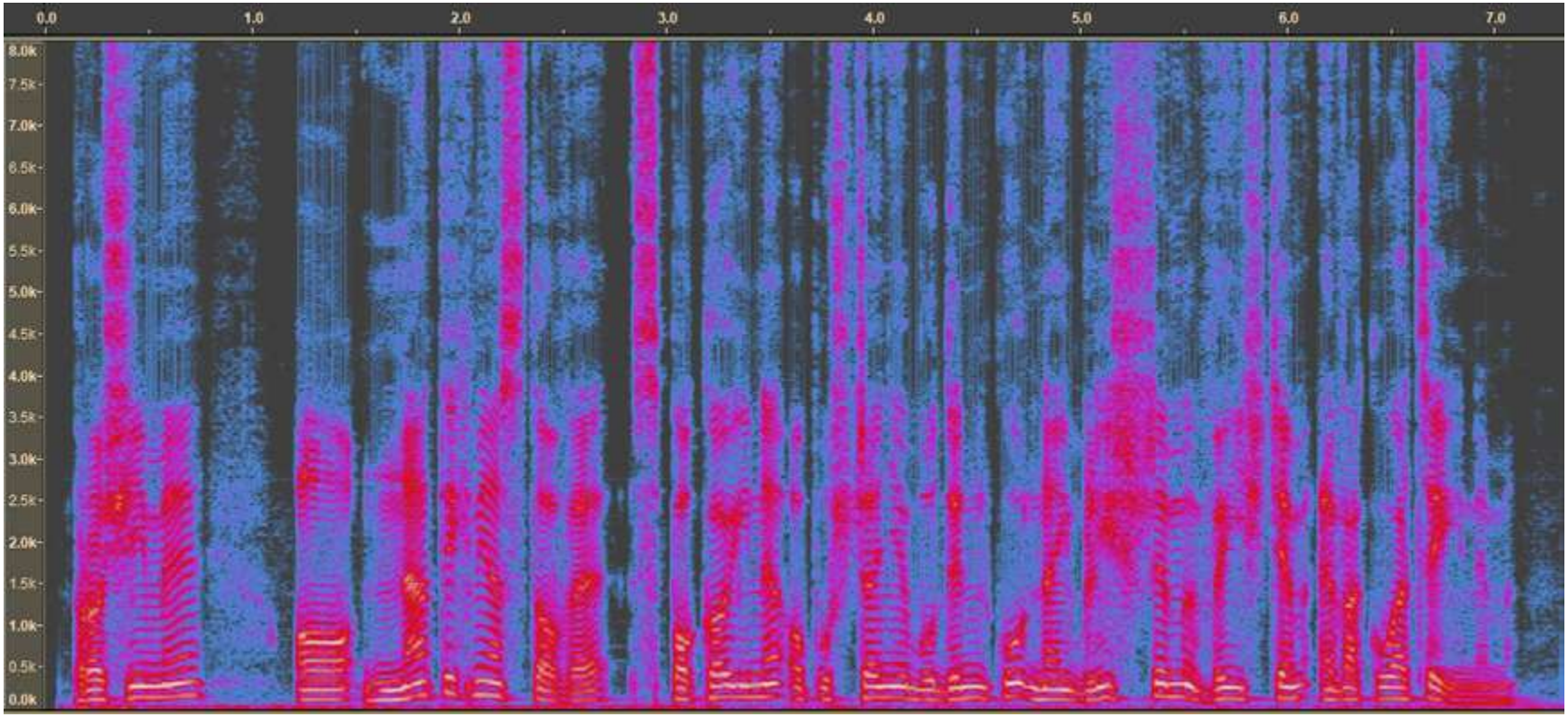
Dla *radix-2* z *N*=1024: ok. 5 tysięcy zespolonych mnożeń dla FFT, w porównaniu do ponad miliona mnożeń dla DFT (200× więcej).

Analiza sygnału ciągłego

- Przekształcenie Fouriera działa na blokach próbek.
- Analiza ciągłego sygnału wymaga podzielenia go na bloki (okna), dla każdego bloku wykonywane jest przekształcenie Fouriera.
- Nazywane jest to krótkookresowym przekształceniem Fouriera – STFT (*Short Term Fourier Transform*).
- Każde obliczone widmo „uśrednia” to co się dzieje wewnątrz danego bloku.
- Tracone są chwilowe zmiany sygnału wewnątrz bloku.

Analiza sygnału ciągłego - STFT

Wynik STFT przedstawia się w formie spektrogramu:
czas (oś pozioma) – częstotliwość (pionowa) – amplituda (kolor)



Rozdzielczość czasowa STFT

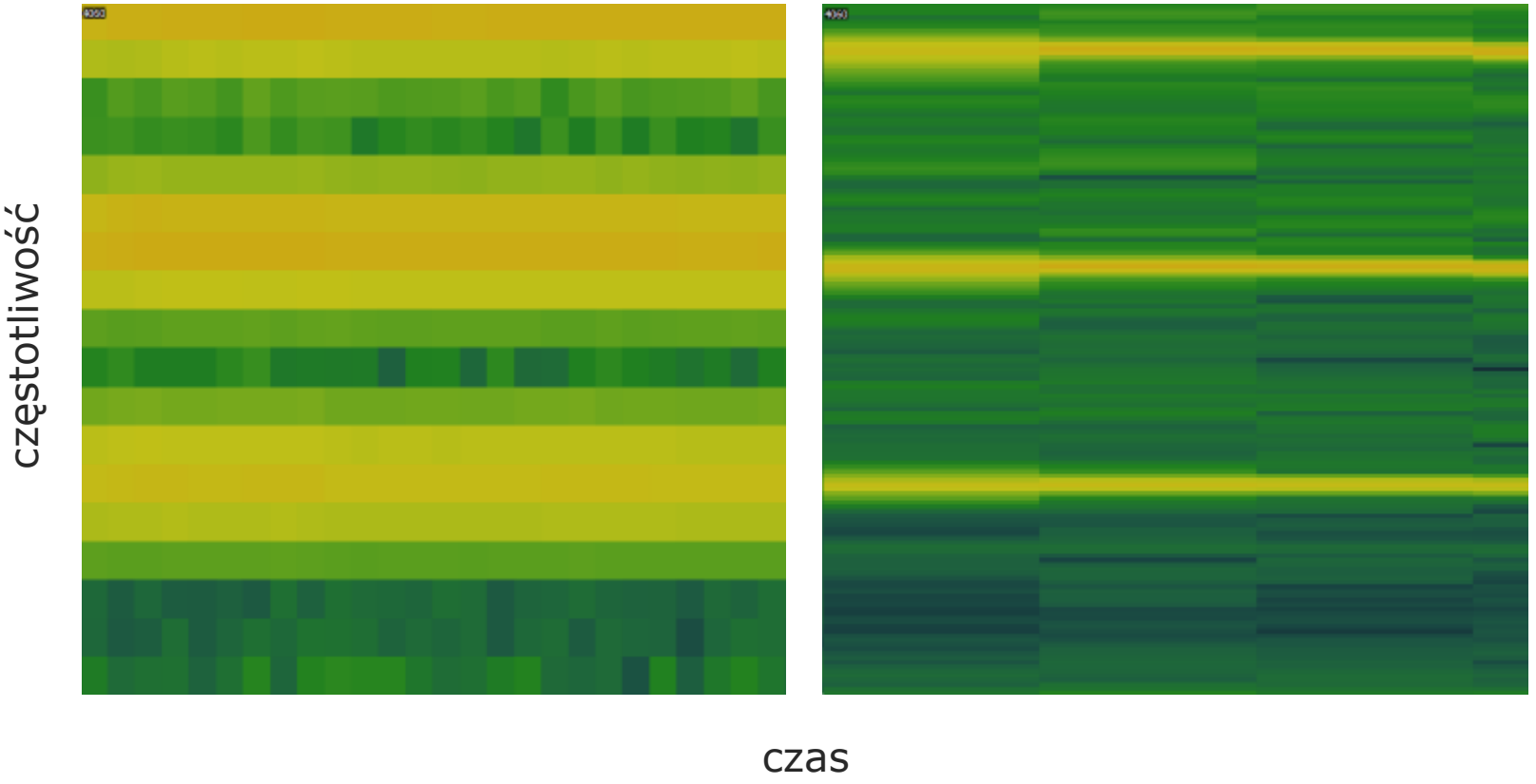
- Rozdzielczość czasowa wynika z długości okna:

$$dt = \frac{N}{f_s} = \frac{1}{df}$$

- Jest odwrotnością rozdzielczości częstotliwościowej.
- Znaczenie: minimalny odstęp czasowy między zdarzeniami w sygnale, które można rozróżnić w analizie STFT.
- Nie da się uzyskać jednocześnie dobrej rozdzielczości czasowej i częstotliwościowej w STFT.

Rozdzielczość czasowa STFT

Porównanie okien o długości 512 i 4096 próbek.



Zakladkowanie

- Zakładkowanie (*overlapping*) polega na tym, że okno analizy przesuwa się o mniej niż długość okna (niektóre próbki są analizowane więcej niż jeden raz).
- Cel zakładkowania:
  - zwiększenie (sztuczne) rozdzielczości czasowej,
  - zniwelowanie wpływu funkcji okna na widmo.
- Zwykle stosuje się następującą konwencję:
  - dla okien Hamminga i von Hanna: przesunięcie o ½ długości okna,
  - dla okna Blackmana: przesunięcie o ¼ długości okna.

FFT na procesorach sygnałowych

- Architektura i lista rozkazów procesorów sygnałowych umożliwiają szybkie wykonywanie FFT.
- Niektóre DSP mają specjalne koprocesory tylko do FFT.
- Producent DSP zwykle dostarcza zoptymalizowane procedury FFT napisane w asemblerze, z których powinniśmy korzystać.
- Często są to tylko implementacje radix-2.
- Można napisać samodzielnie algorytm FFT, ale trudno jest napisać działający szybciej niż już istniejący i przetestowany.

FFT na procesorze C5535

- Procesor stałoprzecinkowy C5535 (stosowany w projekcie) posiada koprocesor specjalnie do obliczania FFT (HWAFFT).
- Sprzętowa implementacja FFT i IFFT o rozmiarach: 8, 16, 32, 64, 128, 512, 1024.
- Operuje na sygnałach zespolonych. Przetwarzając sygnał rzeczywisty musimy wstawić zera za części urojone.
- Specjalna procedura do odwrotnego adresowania pamięci – wykonywanie odwracania bitowego.
- Współczynniki *twiddle* przechowywane w pamięci.
- Możliwość obliczania dwóch etapów FFT w jednym przebiegu.

FFT na procesorze C5535

- Funkcje do obliczania FFT i IFFT są dostępne z języka C.
- Wygodniej jest stosować funkcje z biblioteki DSPLIB.
  Dokumentacja: SPRU422J

Functions	Description
void cfft (DATA *x, ushort nx, type)	Radix-2 complex forward FFT – MACRO
void cfft32 (LDATA *x, ushort nx, type);	32-bit forward complex FFT
void cifft (DATA *x, ushort nx, type)	Radix-2 complex inverse FFT – MACRO
void cifft32 (LDATA *x, ushort nx, type);	32-bit inverse complex FFT
void cbrev (DATA *x, DATA *r, ushort n)	Complex bit-reverse function
void cbrev32 (LDATA *a, LDATA *r, ushort)	32-bit complex bit reverse
void rfft (DATA *x, ushort nx, type)	Radix-2 real forward FFT – MACRO
void rifft (DATA *x, ushort nx, type)	Radix-2 real inverse FFT – MACRO
void rfft32 (LDATA *x, ushort nx, type)	Forward 32-bit Real FFT (in-place)
void rifft32 (LDATA *x, ushort nx, type)	Inverse 32-bit Real FFT (in-place)

Zapis zespolonego widma

- Wartości widma są zespolone.
- Część rzeczywista i część zespolona wartości widmowych są zapisywane jako osobne liczby, jedna po drugiej:
  Re(0), Im(0), Re(1), Im(1), Re(2), Im(2), …
- Każda część zapisywana jako Q15 lub Q31.
- Funkcja *cfft* wymaga sygnału zespolonego.
  Jeżeli mamy sygnał rzeczywisty, musimy wstawić zera między wartości rzeczywiste.
- Do obliczenia IFFT musimy zapisać widmo w powyższej formie.

FFT dla sygnału rzeczywistego

Funkcje *rfft* i *irfft* stosują pewien trik:

- traktują sygnał rzeczywisty jako zespolony, czyli część rzeczywista i urojona na zmianę,
- obliczają FFT o długości $N/2$,
- dokonują przekształceń tak, aby wynik był prawidłowy.

Z tego względu można obliczyć FFT o maksymalnej długości 2048 zamiast 1024 (jak dla *cfft*).

Szczegóły: Robert Matusiak, Implementing Fast Fourier Transform Algorithms of Real-Valued Sequences With the TMS320 DSP Platform (SPRA291)

UWAGA: wszystkie funkcje FFT z DSPLIB wykonują operacje „w miejscu”, tzn. nadpisują bufor wejściowy!

FFT dla sygnału rzeczywistego

- Widmo sygnału rzeczywistego jest symetryczne.
- Na podstawie *N* wartości sygnału rzeczywistego, funkcja *rfft* oblicza *N*/2 wartości zespolonych widma, zapisanych jako *N* liczb (część rzeczywista, część urojona).
- Pierwsze dwie wartości są rzeczywiste i reprezentują składową stałą oraz składową Nyquista.
- Reprezentacja widma:
  Re(0), Re(N/2), Re(1), Im(1), Re(2), Im(2), …

Przekroczenie zakresu

- Przy obliczaniu etapów FFT istnieje ryzyko przekroczenia zakresu liczb stałoprzecinkowych (*overflow*).
- Funkcje z DSPLIB posiadają dwa tryby pracy.
- Tryb SCALE:
  - po każdym etapie, wartości są dzielone przez 2
  - brak przepełnienia gdy wartości wejściowe < 1.
- Tryb NOSCALE:
  - brak skalowania,
  - brak przepełnienia tylko wtedy, gdy wartości wejściowe < (1/N), dla FFT o długości 2^N.

FFT sygnału rzeczywistego

- Dla sygnału rzeczywistego – funkcja *rfft*:

```
void rfft (DATA *x, ushort nx, type);
```

- Argumenty:
  - *x* – wskaźnik do bufora próbek (zostanie nadpisany przez wynik!), typ DATA jest równy short.
  - *nx* – rozmiar bufora (liczba próbek),
  - *type* – skalowanie, podajemy *SCALE*.

```
rfft(bufor, 2048, SCALE);
```

Konfiguracja projektu do obliczania FFT

Funkcje FFT z DSPLIB wymagają spełnienia następujących warunków (przykład dla *N* = 2048).

- Konfiguracja pamięci w pliku *.cmd*:

```
.fftcode        >  SARAM0
.data:twiddle   >  SARAM1, align(2048)
.input          >  DARAM0, align(4)
```

- Deklaracja bufora do obliczania FFT w kodzie C:

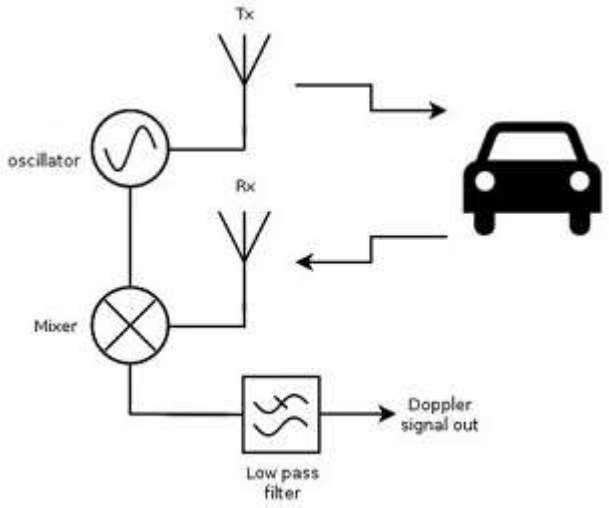
```
#define N 2048
#pragma DATA_SECTION (bufor_fft, ".input")
DATA bufor_fft[N];
```

- Nazwa ".input" jest przykładowa, można użyć dowolnej.

Praktyczny projekt – radar dopplerowski

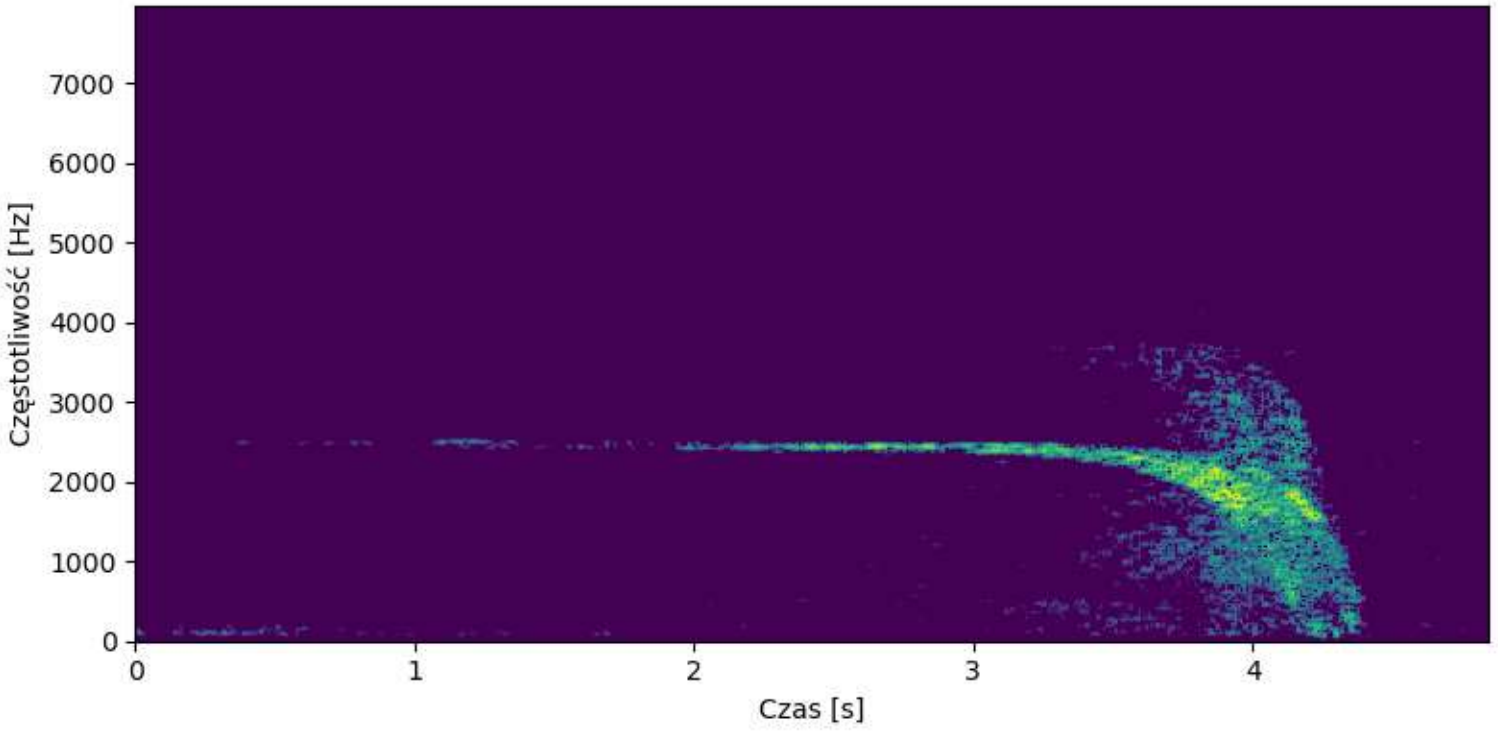
Zastosowanie DSP do analizy sygnału z mikrofalowego, Dopplerowskiego czujnika radarowego.

- Nadajnik emituje falę elektromagnetyczną – sinus o częstotliwości 24,125 GHz.
- Odbiornik zbiera falę odbitą od obiektu.
- Na skutek efektu Dopplera, odbita fala ma inną częstotliwość niż nadana.
- Różnica częstotliwości jest proporcjonalna do prędkości obiektu.



Spektrogram sygnału z czujnika

Sygnał różnicowy – spektrogram dla przejazdu samochodu



Analiza sygnału

Schemat przetwarzania:

- podział sygnału na bloki o długości 2048 próbek, zakładkowanie: ½ długości okna,
- przychodzące próbki zapisywane w buforze kołowym,
- gdy mamy zapełniony bufor:
  - mnożymy przez okno Hamminga,
  - obliczamy FFT,
  - obliczamy widmo mocy (kwadrat modułu widma),
  - szukamy maksimów widmowych,
  - jeżeli znajdziemy – obliczamy prędkość pojazdu.

Buforowanie próbek sygnału

- Przychodzące próbki sygnału zapisujemy w buforze kołowym o rozmiarze 2048 próbek (liczb short typu Q15).
- Okno jest przesuwane o 1024 próbek. Gdy mamy zapisanych 1024 nowych próbek:
  - przechodzimy pętlą po próbkach w buforze kołowym, w kolejności od najstarszej do najmłodszej,
  - mnożymy próbkę przez odpowiednią wartość okna Hamminga (funkcja _*smpy*),
  - zapisujemy wynik mnożenia do bufora liniowego.

Funkcja okna

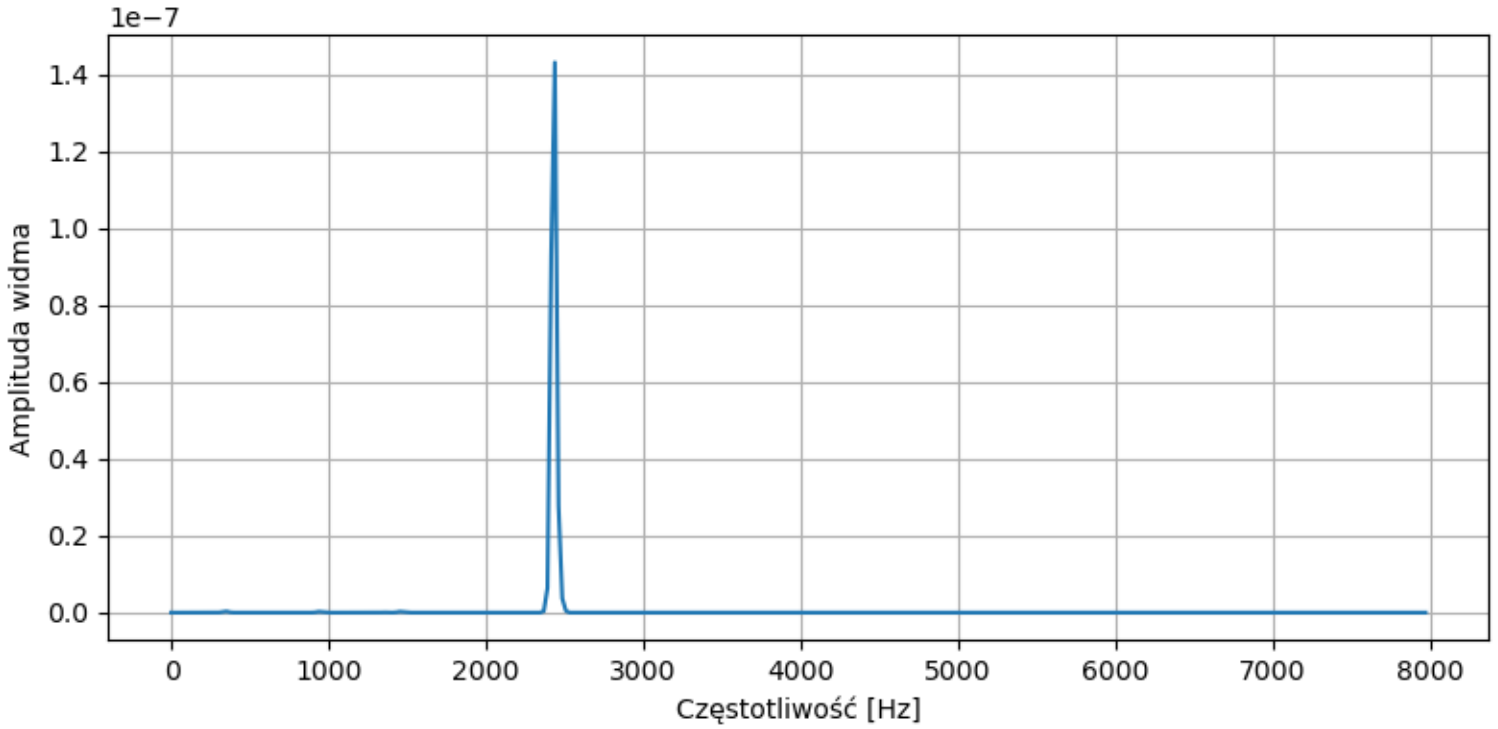
- Nie ma sensu obliczanie wartości okna Hamminga za każdym razem (choć jest to możliwe). Współczynniki są stałe dla ustalonej długości okna.
- Obliczamy wartości okna w zewnętrznym programie.
- Konwertujemy do formatu Q15.
- Zapisujemy je w stałej tablicy (*const short*) w kodzie C.
- UWAGA: maksymalna wartość okna wynosi 1. Nie można zapisać jedynki w Q15! Można przeskalować okno, mnożąc wartości zmiennoprzecinkowe przez 32767 zamiast przez 32768.

Obliczanie widma mocy

- Obliczamy widmo zespolone korzystając z funkcji *rfft*.
- Obliczamy widmo mocy:
  - przechodzimy pętlą po parach (Re, Im) liczb w wyniku *rfft*,
  - obliczamy kwadrat każdej liczby (*_smpy*),
  - dodajemy część rzeczywistą do urojonej,
  - zapisujemy w buforze (możemy użyć tego samego bufora).
- Gdybyśmy chcieli uzyskać widmo amplitudowe, trzeba byłoby obliczyć jeszcze pierwiastek z każdego wyniku (funkcja *sqrt_16* z DSPLIB).
- Jeśli potrzebujemy dokładnych wartości amplitudy, dzielimy wyniki przez 1024 (>>10).

Widmo dla pojedynczego bloku

Następnie analizujemy obliczone widmo, szukając maksimum.



Wyszukiwanie maksimum

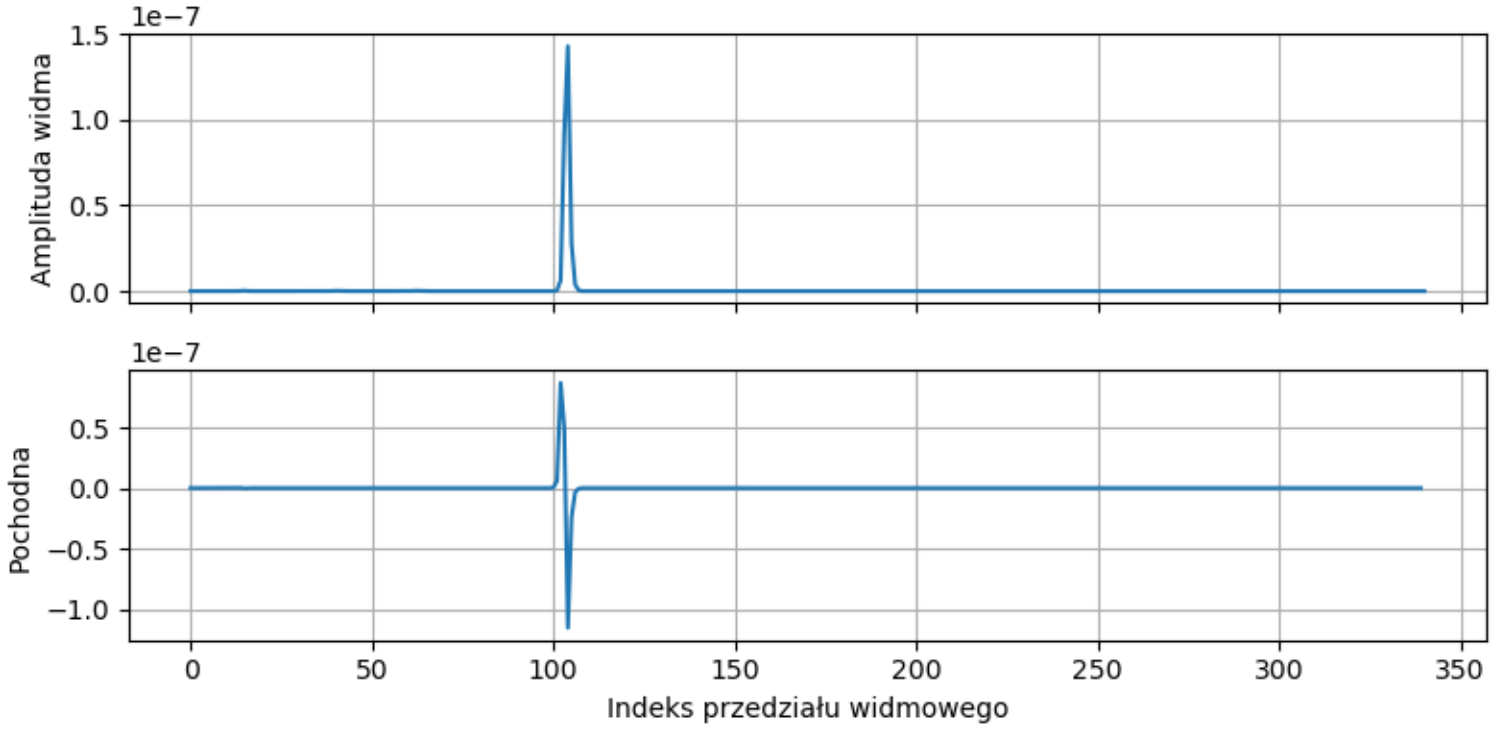
Istnieje wiele metod szukania maksimum. Jedna z prostszych:

- obliczamy pochodną widma, czyli od danej próbki widma odejmujemy poprzednią,
- jeżeli pochodna zmienia znak z dodatniej na ujemną, oznacza to maksimum,
- dodatkowo musimy założyć minimalny próg amplitudy widmowej, aby wyeliminować wpływ szumu,
- można zadać dodatkowe parametry, np. maksymalną szerokość „prążka” (eliminacja szerokich maksimów).

Wyszukiwanie maksimum

Wykres widma i jego pochodnej.

Znalezione maksimum dla indeksu 104.



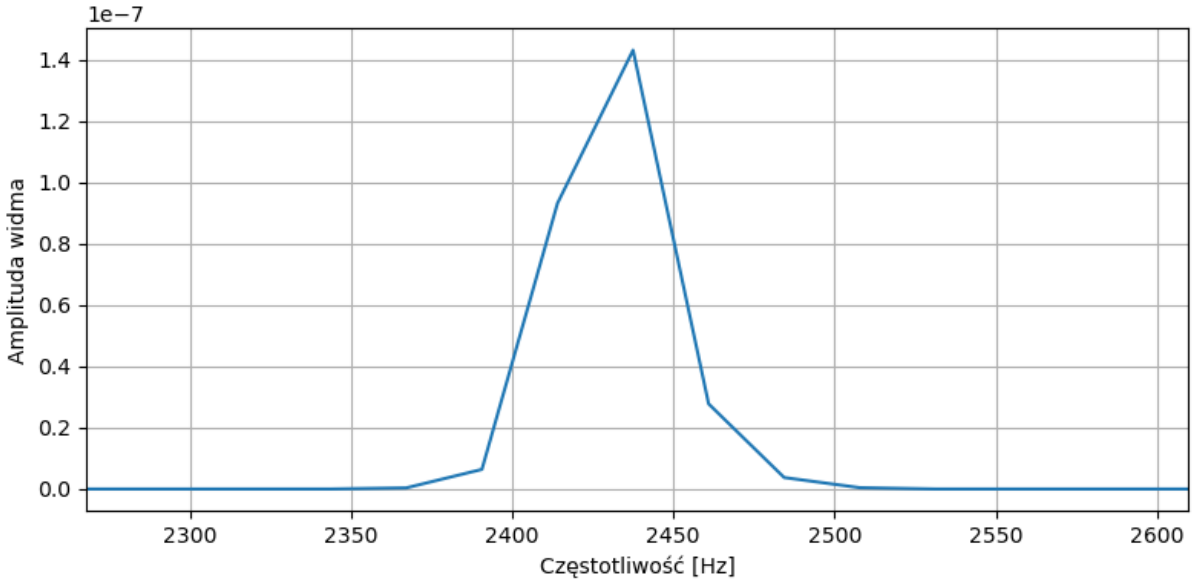
Obliczenie prędkości

Pozostało obliczenie prędkości obiektu.

- Zależność między częstotliwością a prędkością:
  $v \approx 0{,}02234 \cdot f$ (wynika ze wzoru Dopplera)
- Zależność między indeksem widma a częstotliwością, zakładając $f_S = 48$ kHz: $f = 23{,}4375 \cdot n$
- Zatem: $v \approx 0{,}52425 \cdot n$ [km/h]
- W zapisie Q15: $v \approx 17179 \cdot n$
- Obliczamy: 104 * 17179 = 1786616
- W przeliczeniu na format zmiennoprzecinkowy:
  1786616 / 32768 ≈ 54,52 km/h

Dokładność obliczeń

- Pamiętajmy, że dokładność obliczenia częstotliwości wynika z rozdzielczości częstotliwościowej. Dla *N* = 2048 mamy maksymalny błąd 11,72 Hz, czyli ok. 0,26 km/h.
- Znaleziona próbka widma o największej amplitudzie niekoniecznie jest maksimum ciągłego widma.



Dokładniejsze wyszukiwanie maksimum

- Możemy dokładniej obliczyć częstotliwość maksimum, chociaż jest to algorytm raczej dla zmiennoprzecinkowych procesorów, ponieważ wymaga dzielenia.
- Dowolne trzy punkty wyznaczają parabolę.
- Dopasowujemy parabolę do znalezionej maksymalnej próbki i do jej dwóch sąsiadów. Wartości tych próbek kolejno: *a*, *b*, *c*.
- Dokładne położenie maksimum:

$$m = n + \frac{1}{2}\frac{a-c}{a-2b+c}$$

- W naszym przykładzie: *m* = 103,8; *v* = 54,42 (było 54,52).

Źródło: https://ccrma.stanford.edu/~jos/sasp/Quadratic_Interpolation_Spectral_Peaks.html

Dokładniejsze wyszukiwanie maksimum

Wynik dopasowania paraboli i znalezione maksimum (×):