Zastosowania procesorów sygnałowych

PRZEKSZTAŁCENIE FOURIERA

na procesorach sygnałowych

Opracowanie: Grzegorz Szwoch

Politechnika Gdańska, Katedra Systemów Multimedialnych

Przekształcenie Fouriera

- Dyskretne przekształcenie (transformacja) Fouriera (DFT):
  - przekształca reprezentację czasową sygnału – *N* próbek sygnału
  - w reprezentację częstotliwościową sygnału – *N* próbek widma.
- Transformata Fouriera dla sygnału określa widmo (spektrum) sygnału (przez analogię do widma światła). Traktuje sygnał jako sumę składowych widmowych - sinusów o różnych częstotliwościach i amplitudach.
- Odwrotne przekształcenie Fouriera (IDFT) – odtwarza reprezentację czasową z reprezentacji widmowej (DFT jest odwracalne).
- Przekształcenie Fouriera zakłada, że zbiór próbek poddawany przekształceniu jest okresem sygnału.

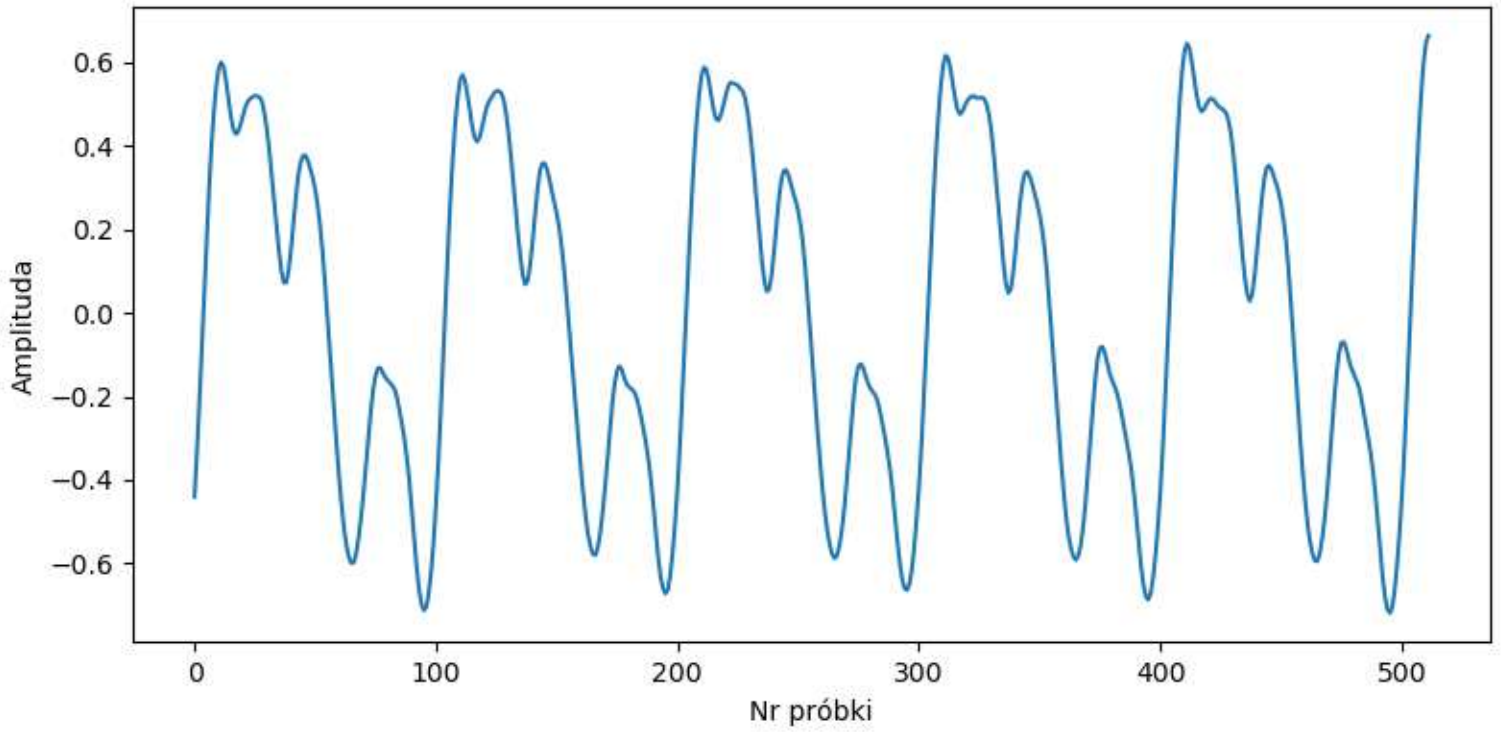
Zastosowania przekształcenia Fouriera

Do czego jest nam potrzebne przekształcenie Fouriera?

- Analiza częstotliwościowa (widmowa) – określenie składowych częstotliwościowych, z których zbudowany jest sygnał.
  - Wyznaczenie i zmierzenie dominujących składowych widmowych.
  - Przykład: mikrofalowy detektor prędkości pojazdów.
- Przetwarzanie sygnałów w dziedzinie częstotliwości.
  - Wiele operacji można wykonać znacznie szybciej w dziedzinie częstotliwości.
  - Przykład: filtracja FIR ciągłego sygnału za pomocą metody OLA, np. redukcja zakłóceń w sygnale, poprawa jakości dźwięku.
- Przekształcenie Fouriera jest (obok filtracji) podstawowym algorytmem wykonywanym na procesorach sygnałowych i częścią wielu złożonych algorytmów przetwarzania sygnałów.

Przykład wykorzystania przekształcenia Fouriera

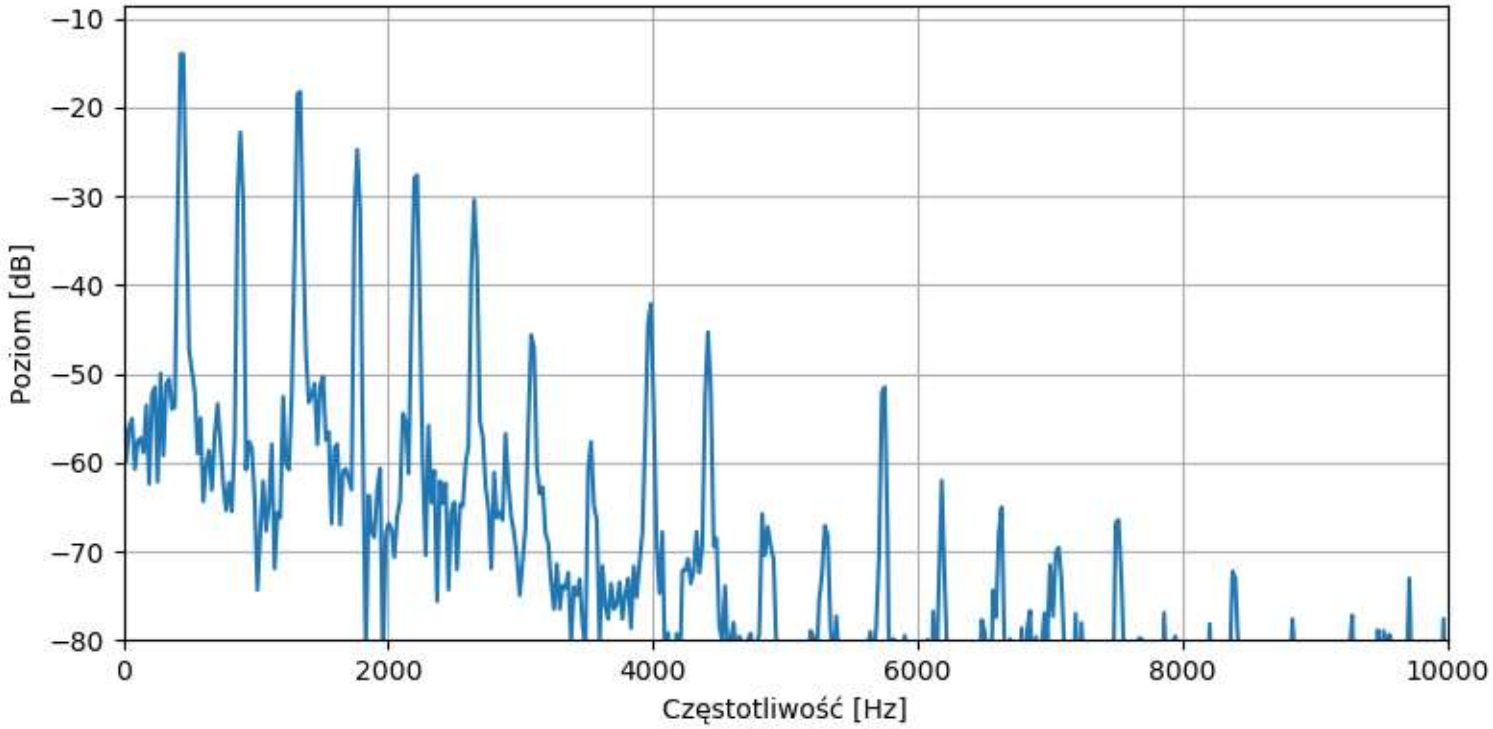
Nagranie dźwięku klarnetu.
Z wykresu czasowego nie dowiemy się z jakich składowych jest zbudowany sygnał.



Przykład wykorzystania przekształcenia Fouriera

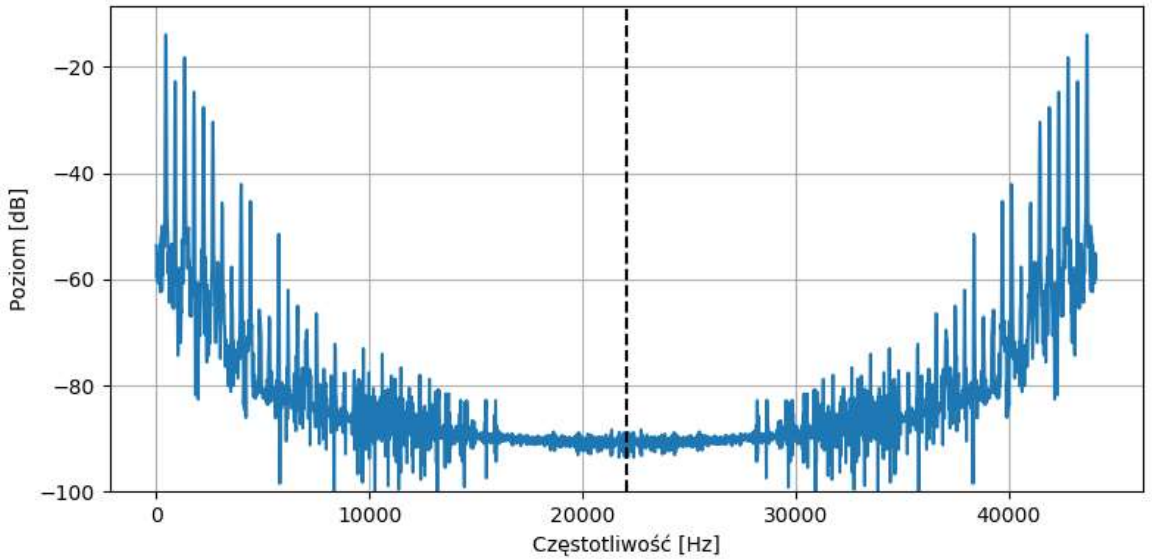
Wynik analizy częstotliwościowej dźwięku klarnetu (widmo).

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



Widmo sygnału rzeczywistego

- Zazwyczaj przekształcamy sygnały o wartościach rzeczywistych.
- *N* próbek sygnału przekształcamy w *N* zespolonych wartości widma:
  - wartość 1: składowa stała, suma wartości próbek (wartość rzeczywista)
  - wartości od 2 do $N/2$: składowe widma sygnału (zespolone)
  - wartość $N/2 + 1$: składowa Nyquista (rzeczywista), powinna być zerowa
  - wartości od $N/2 + 2$ do N: lustrzana kopia pierwszej połowy widma.
- Widmo z *N* próbek ma $(N/2 + 1)$ unikalnych wartości.
- Pozostałe wartości (od $N/2+2$) możemy pominąć dla sygnału o rzeczywistych wartościach.



Widmo amplitudowe i mocy

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

$$A(f) = |X(f)| = \sqrt{Re\big(X(f)\big)^2 + Im\big(X(f)\big)^2}$$

- Widmo mocy sygnału jest kwadratem modułu widma:

$$P(f) = |X(f)|^2 = Re\big(X(f)\big)^2 + Im\big(X(f)\big)^2$$

- Widma amplitudowe i mocy często wyrażamy 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*:

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

- Pierwsza wartość (składowa stała) jest rzeczywista i nie ma pary. Wartość ta podzielona przez *N* = wartość średnia sygnału w analizowanym okresie.
- Składowa Nyquista również jest rzeczywista i nie ma pary – powinna być podzielona przez *N*, ale w praktyce powinna być ona zerowa (jeżeli nie jest, wskazuje to na występowanie aliasingu widma).

Częstotliwości próbek widma

- Na podstawie *N* próbek sygnału obliczamy *N* próbek widma.
- Widmo pokrywa zakres częstotliwości od 0 do częstotliwości próbkowania f_S.
- Zatem *n*-ta wartość widma odpowiada częstotliwości:

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

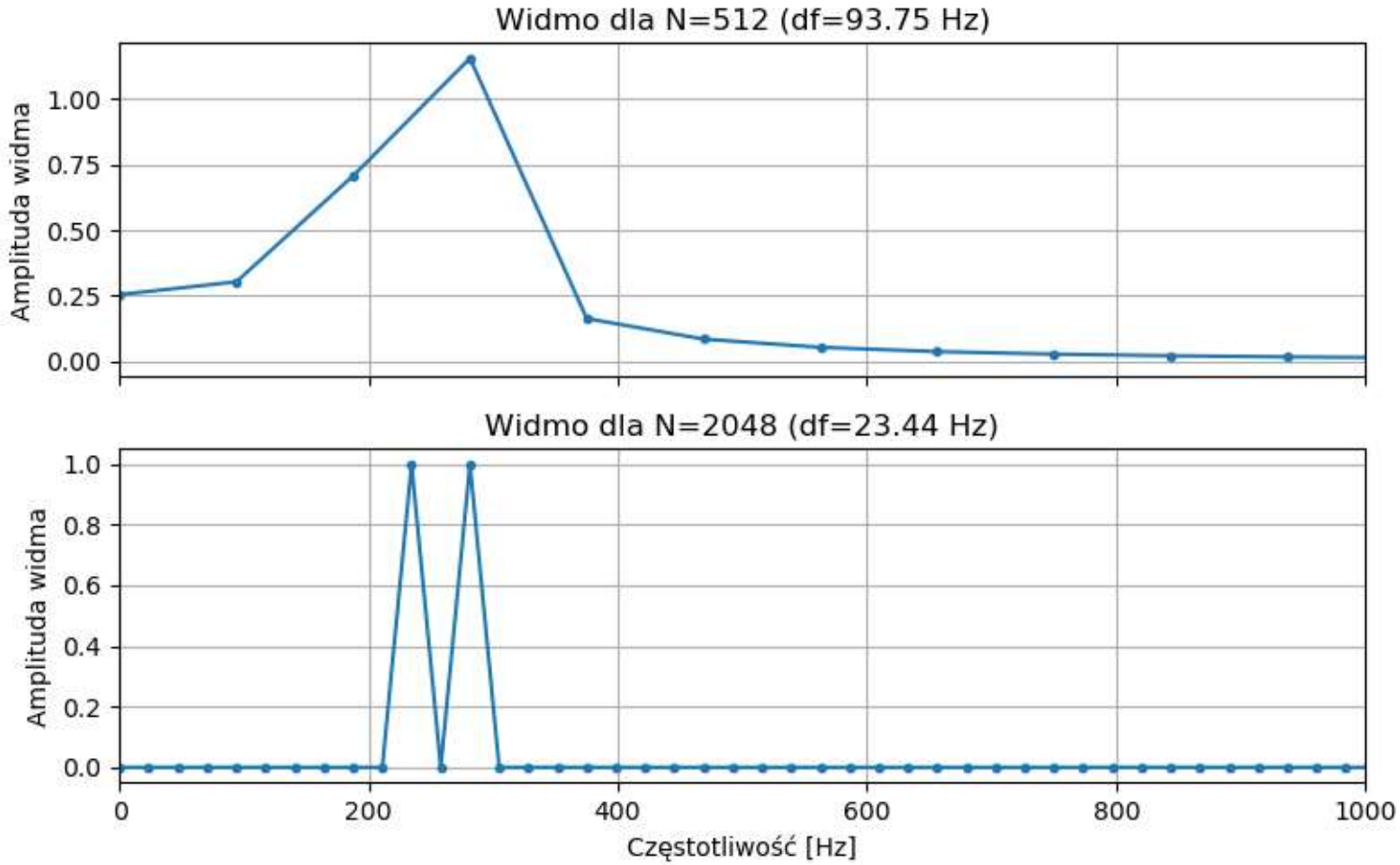
- Odstęp między próbkami widma wyznacza rozdzielczość częstotliwościową widma:

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

- Nie można rozróżnić dwóch składowych widmowych, jeżeli odstęp między nimi jest mniejszy niż *df.*
- Można sztucznie poprawić rozdzielczość uzupełniając sygnał zerami na końcu (*zero padding*). Nie dostajemy w ten sposób więcej danych (wartości są interpolowane), ale uzyskujemy lepszą rozdzielczość częstotliwościową.

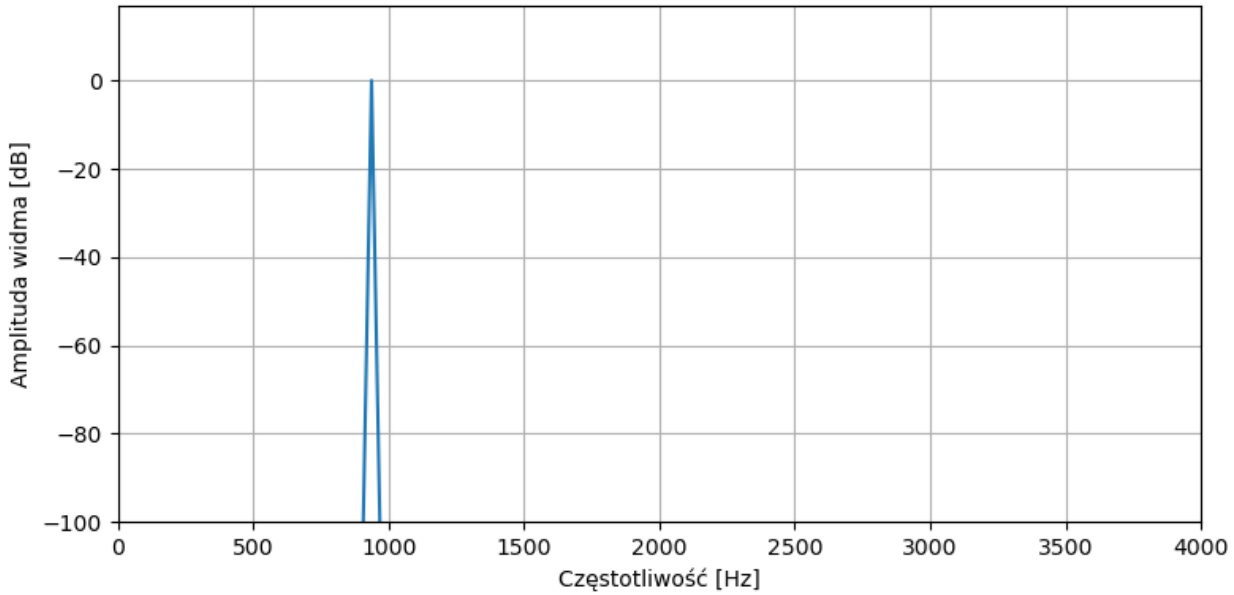
Rozdzielczość częstotliwościowa

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



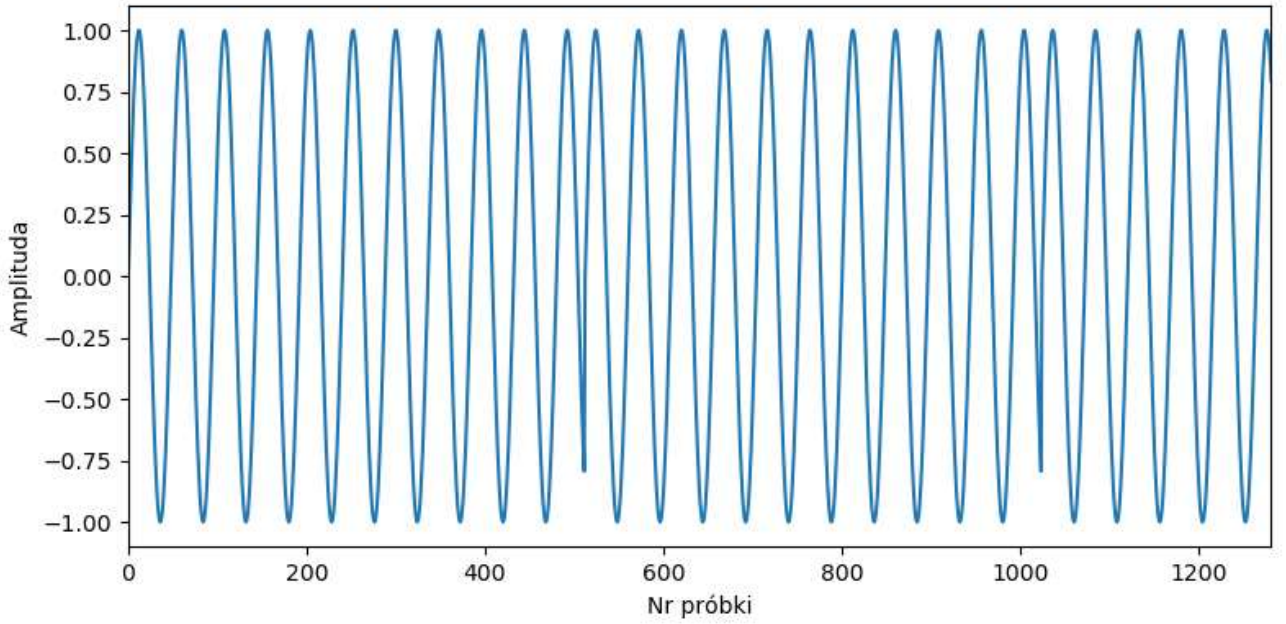
Okresowość sygnału

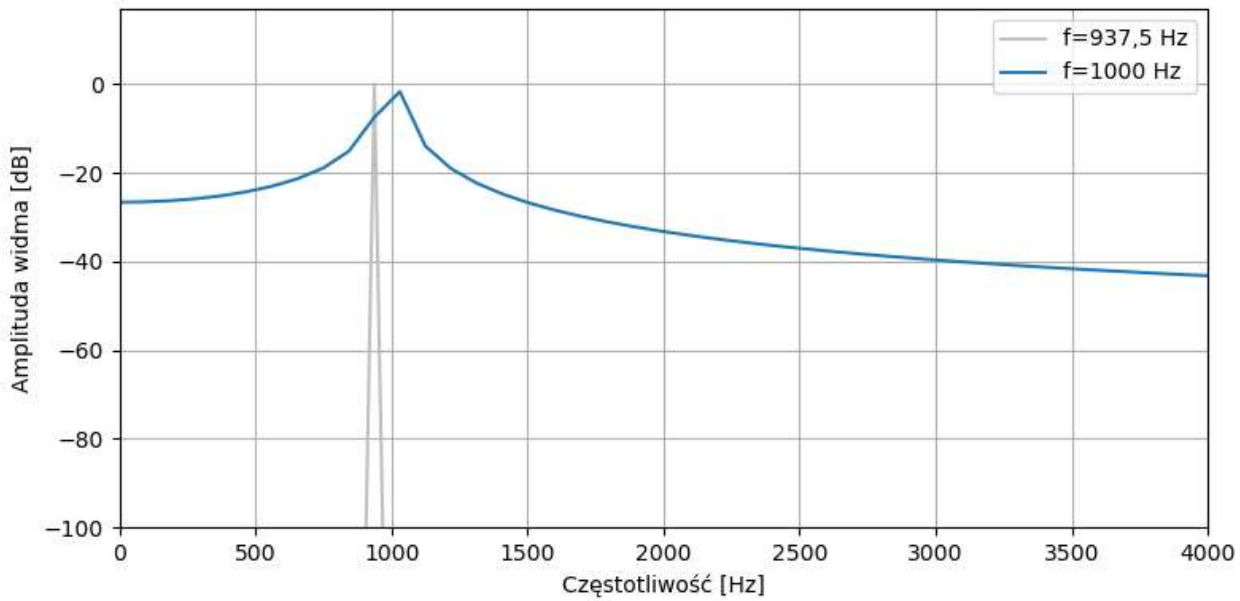
- Przekształcenie Fouriera zakłada, że zbiór próbek poddawanych przekształceniu jest dokładnie okresem analizowanego sygnału lub jego wielokrotnością.
- Przykład widma dla sytuacji gdy jest to prawdą (f = 937,5 Hz; N = 512):



Okno analizy i przecieki widma

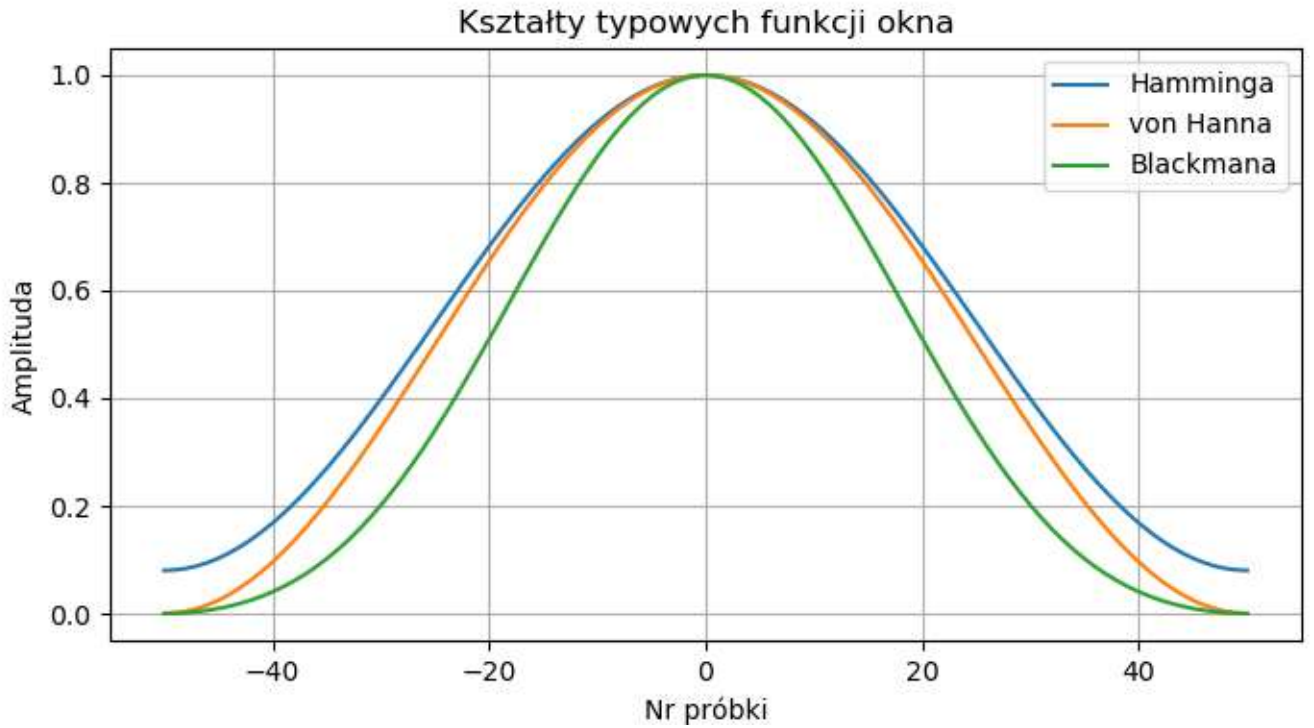
- W praktyce wycinamy fragment dłuższego sygnału za pomocą okna analizy.
- Uzyskujemy transformatę przy założeniu, że okno to jest okresem sygnału.
- Ponieważ nie jest to (zazwyczaj) prawdą, powstaje efekt przecieków widma
  - amplituda składowej „rozlewa się” na kilka sąsiednich wartości.





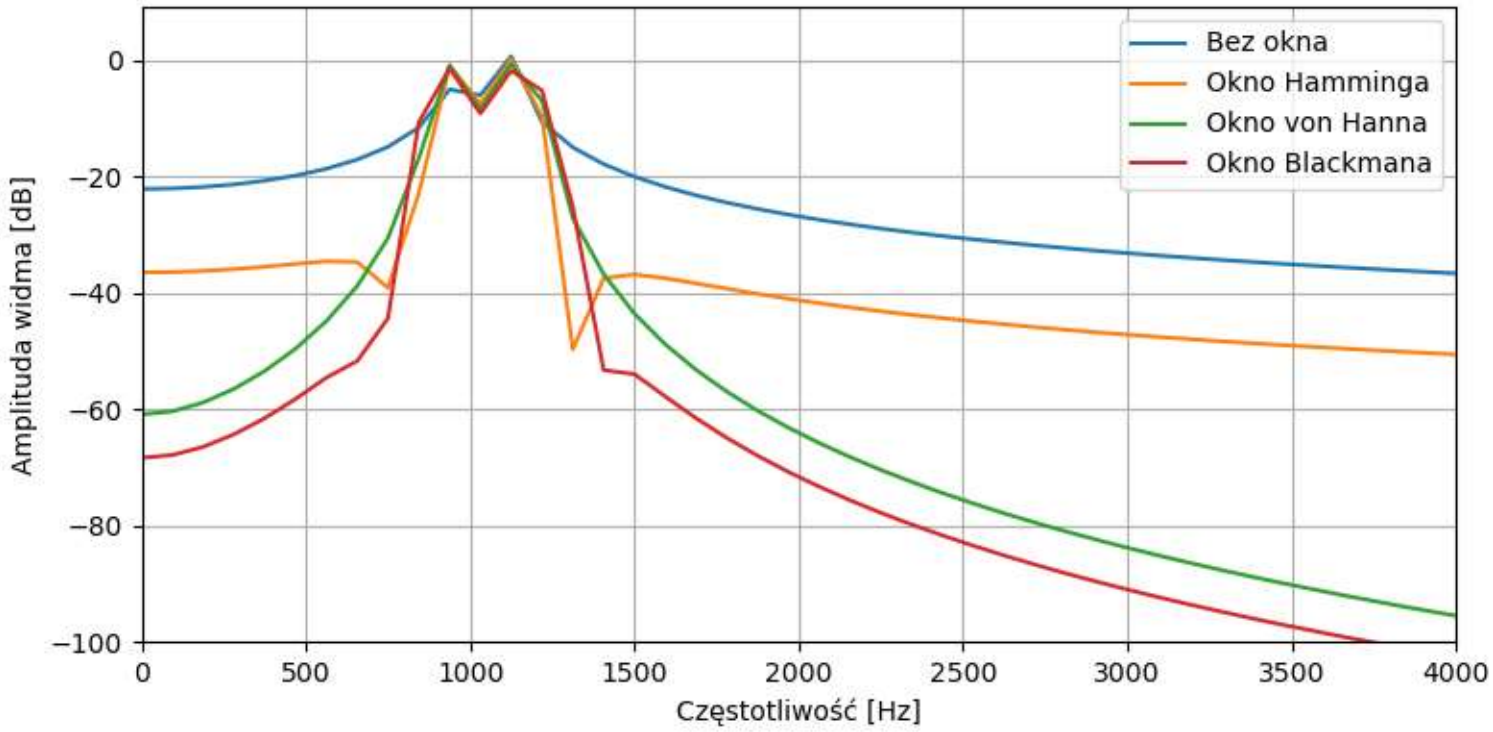
Funkcje okien czasowych

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



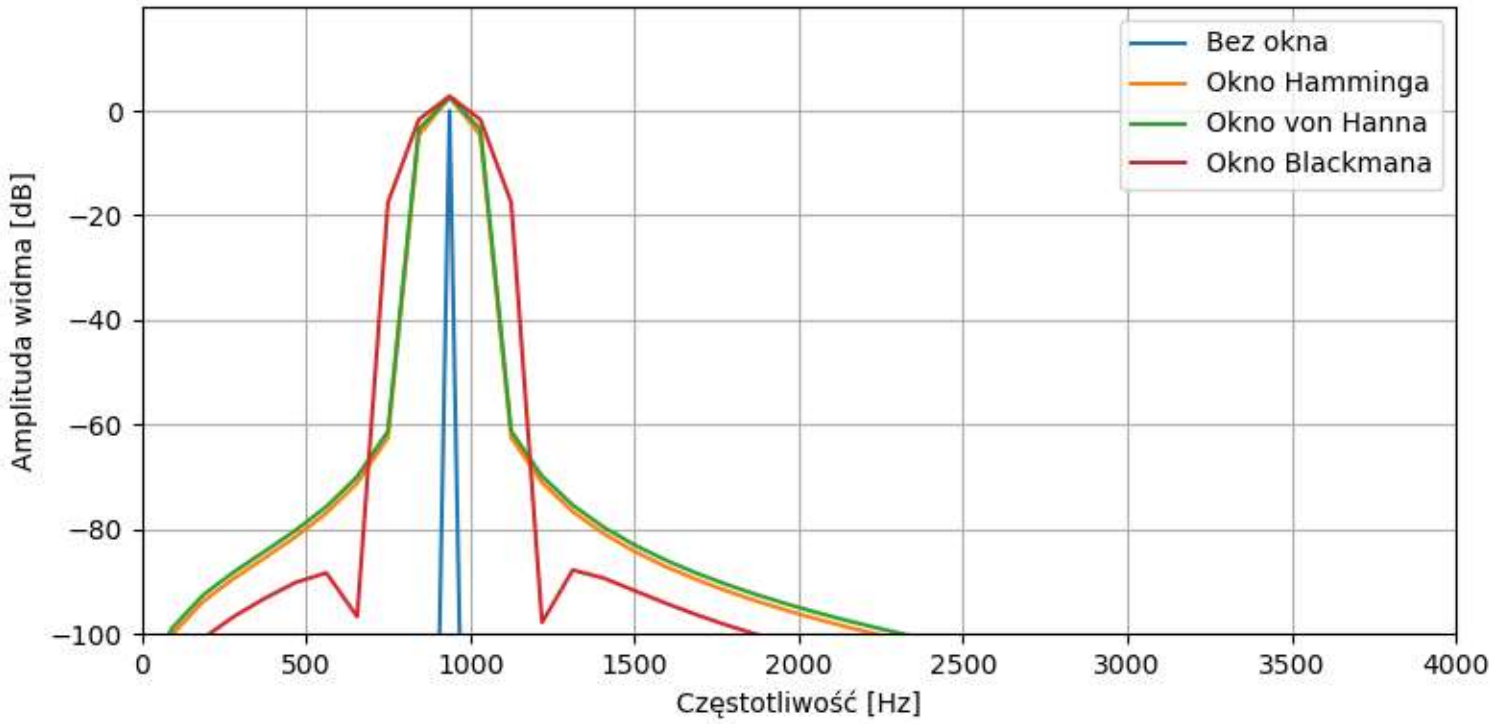
Wpływ okna czasowego

Zastosowanie funkcji okna w przypadku występowania przecieków widmowych – stłumienie przecieków, poprawa dokładności analizy widmowej.



Wpływ okna analizy

Zastosowanie funkcji okna w przypadku braku przecieków widmowych – tutaj funkcje okna pogarszają wyniki, poszerzając „pik” widma.



Uwagi o funkcjach okna

- Nie ma „najlepszego” okna, ale okna Hamminga i von Hanna są najbardziej uniwersalne i najczęściej stosowane w praktyce.
- 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*:

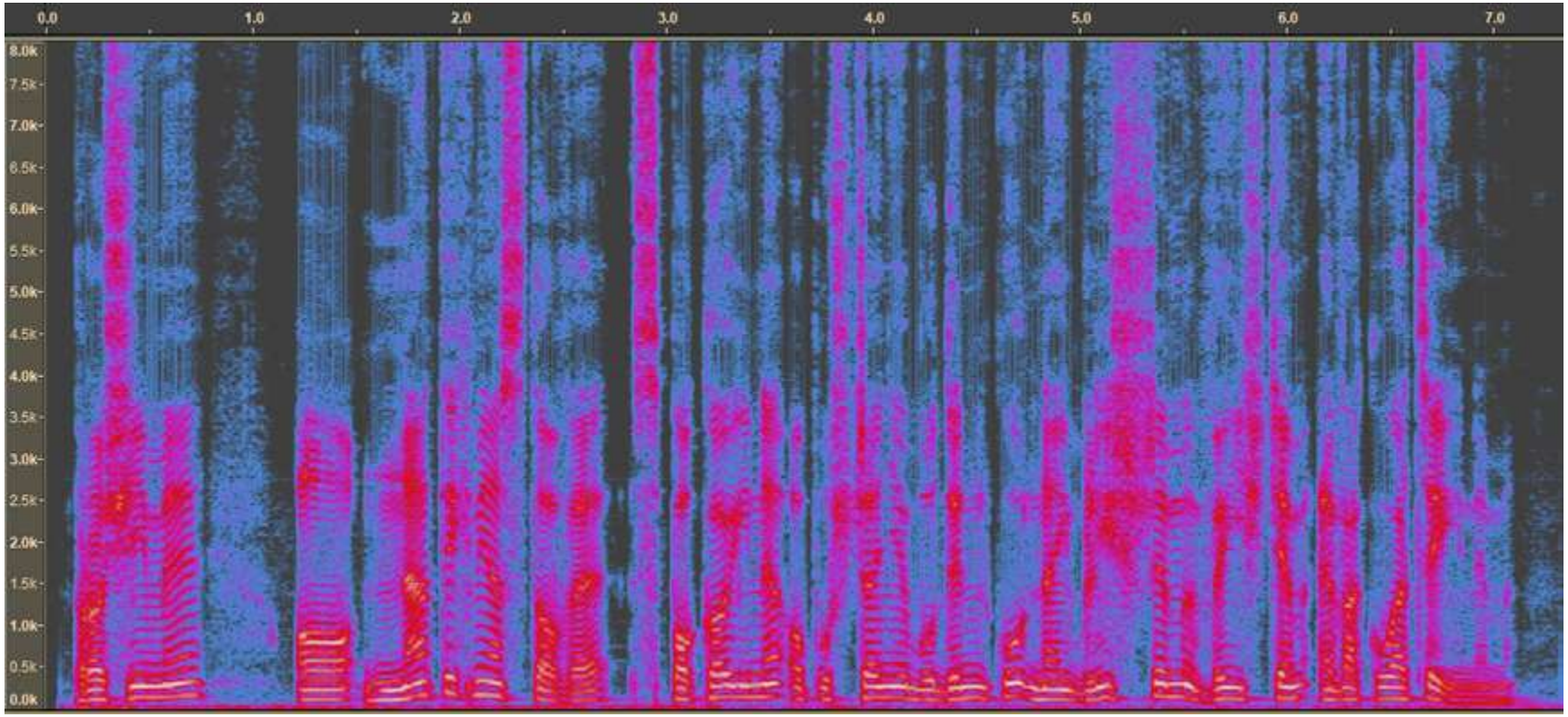
$$A[n] = \frac{2}{\sum w_i} |X[n]|$$

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

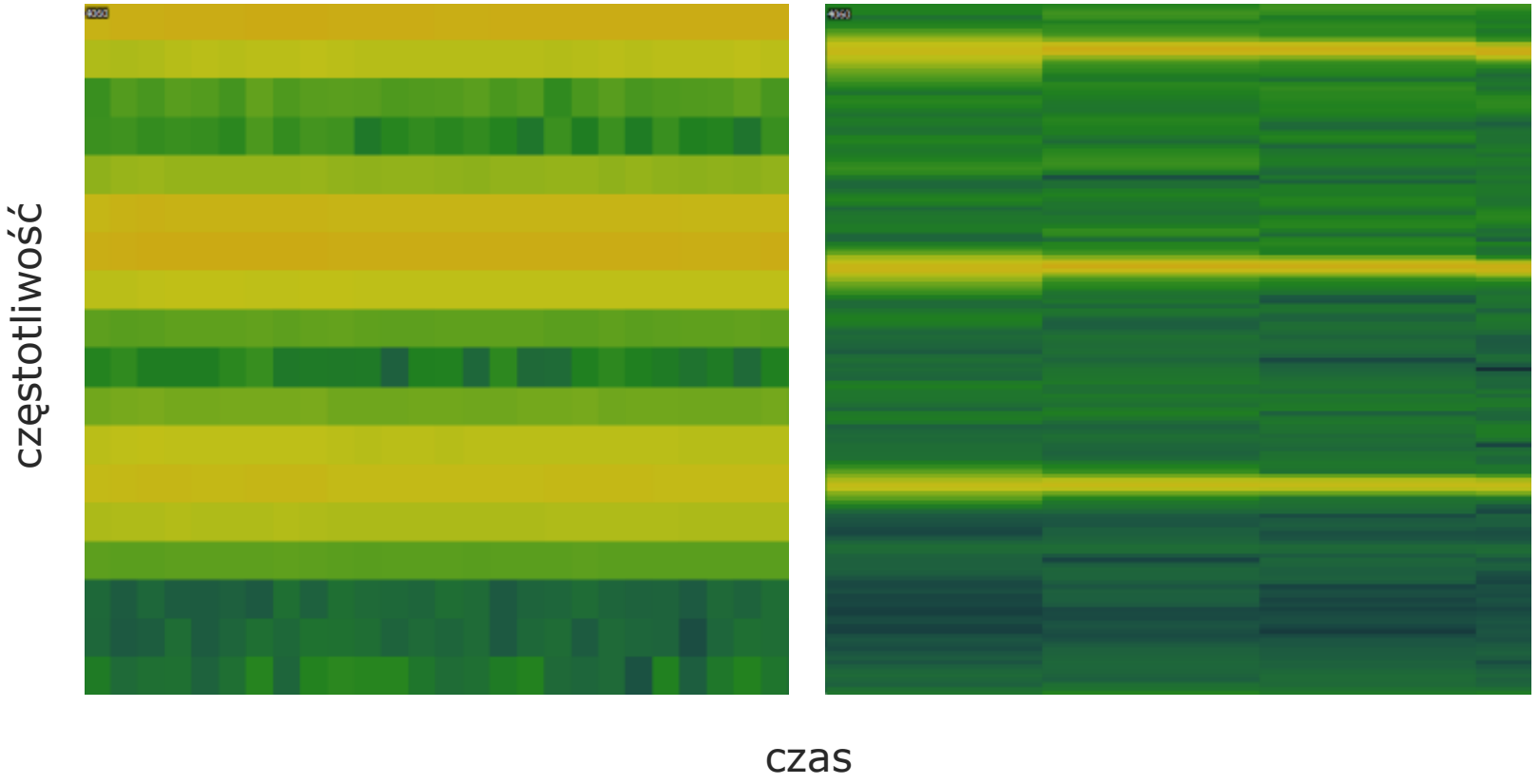
- Rozdzielczość czasowa wynika z długości okna:

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

- Interpretacja: minimalny odstęp czasowy między zdarzeniami w sygnale, które można rozróżnić w analizie STFT.
- Czyli:
  - rozmiar okna musi być duży, aby uzyskać dobrą rozdzielczość częstotliwościową,
  - rozmiar okna musi być mały, aby uzyskać dobrą rozdzielczość czasową.
- Nie da się tego pogodzić w analizie STFT. Musimy wybrać właściwy kompromis, zależnie od charakteru sygnału i tego, co chcemy uzyskać.

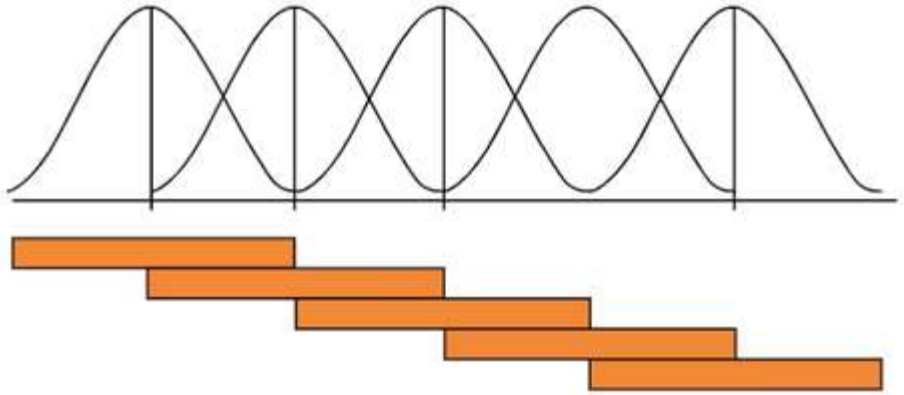
Rozdzielczość czasowa STFT

Porównanie okien o długości 512 i 4096 próbek.



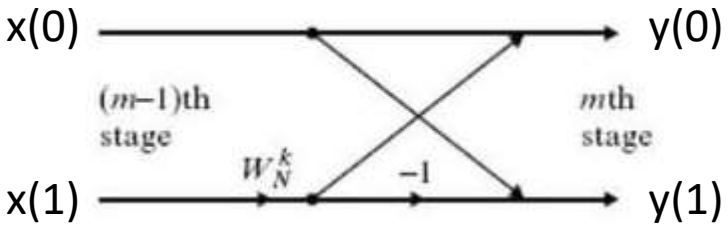
Zakładkowanie

- Zakładkowanie (*overlapping*) polega na tym, że okno analizy przesuwa się o mniej niż długość okna (niektóre próbki są analizowane więcej niż jeden raz).
- Efekt: zniwelowanie wpływu funkcji okna na widmo (tłumienia skrajnych wartości) oraz poprawienie rozdzielczości czasowej.
- Zwykle dla okna Hamminga i von Hanna stosuje się przesunięcie o ½ długości okna, dla okna Blackmana: o ¼ długości okna.



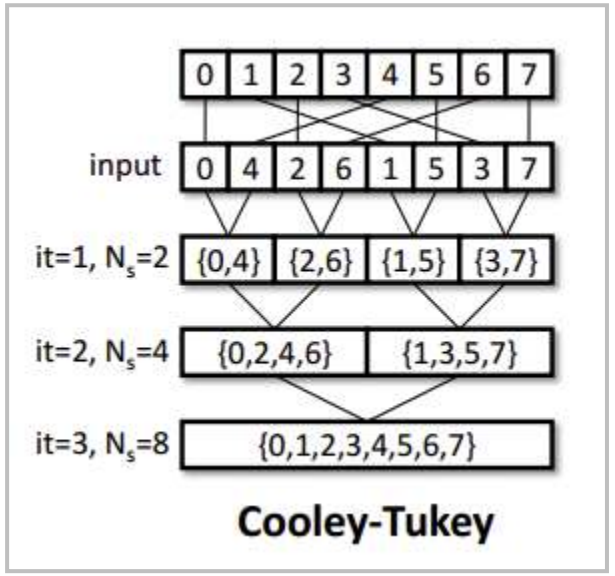
FFT

- Procesory sygnałowe obliczają transformatę za pomocą szybkiego przekształcenia Fouriera – FFT (*Fast Fourier Transform*), algorytm Cooleya-Tukeya.
- Zwykle implementują one algorytm *radix-2*, w którym rozmiar transformaty musi być potęgą dwójki (np. 512, 1024, 2048).
- Algorytm dzieli blok próbek sukcesywnie na pół, aż dochodzi do transformaty dwóch elementów („motylka”), po czym składa wynik transformaty.
- Algorytmy na „zwykłych procesorach” zazwyczaj obsługują dowolny rozmiar transformaty, ale najszybciej działają dla potęg dwójki.



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

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



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 operacji mnożenia liczb zespolonych dla FFT, w porównaniu do ponad miliona operacji dla DFT (200× więcej).

FFT na procesorach sygnałowych

- Architektura i lista rozkazów procesorów sygnałowych umożliwiają szybkie obliczanie FFT.
- Niektóre procesory sygnałowe mają specjalne koprocesory tylko do FFT.
- Procesor: C5535: sprzętowe przyspieszenie FFT (HWAFFT), FFT i IFFT o rozmiarach: 8, 16, 32, 64, 128, 512, 1024 (dla sygnałów rzeczywistych również 2048).
- Producent procesora zwykle dostarcza zoptymalizowane procedury FFT napisane w asemblerze, z których powinniśmy korzystać.
- Często są to tylko implementacje radix-2, a więc rozmiar transformaty musi być potęgą dwójki.
- Można napisać samodzielnie algorytm FFT, ale trudno jest napisać działający szybciej niż już istniejący i przetestowany.

FFT na procesorze C5535

- Instrukcje procesora do obliczania FFT i IFFT są dostępne z poziomu języka C.
- Wygodniej jest stosować funkcje z biblioteki *DSPLIB* dla wartości rzeczywistych (*rfft*) i zespolonych (*cfft*).
  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), Re(3), Im(3), …
- Każda część zapisywana jako Q15 (*short*) lub Q31 (*long*).
- Do obliczenia IFFT musimy zapisać widmo w powyższej formie.
- Funkcja *cfft* wymaga sygnału zespolonego. Jeżeli mamy sygnał rzeczywisty, musimy wstawić zera między wartości rzeczywiste.
- Funkcja *rfft* działa dla sygnału rzeczywistego. Zapis widma:
  Re(0), Re(Nyquist), Re(1), Im(1), Re(2), Im(2), Re(3), Im(3), …
- Rozmiar transformaty: 8, 16, 32, 64, 128, 512, 1024, (tylko *rfft*) 2048.

FFT sygnału rzeczywistego za pomocą DSPLIB

- Obliczenie FFT dla sygnału rzeczywistego za pomocą *DSPLIB*: funkcja *rfft*:

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

- Argumenty:
  - *x*: wskaźnik do tablicy zawierającej próbki sygnału – zostanie ona nadpisana przez widmo (!); typ DATA jest aliasem *short* (liczby 16-bitowe).
  - *nx*: liczba próbek w tablicy,
  - *type* – zwykle podajemy stałą *SCALE*, która dzieli wartości przez 2 po każdym etapie (co zmniejsza rozdzielczość wartości widma); stała *NOSCALE* wyłącza skalowanie, ale znacznie zwiększa to ryzyko przepełnienia zakresu.

```
rfft(tablica, 1024, 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 tablicy *dane* (nazwa przykładowa) do obliczania FFT w kodzie C:

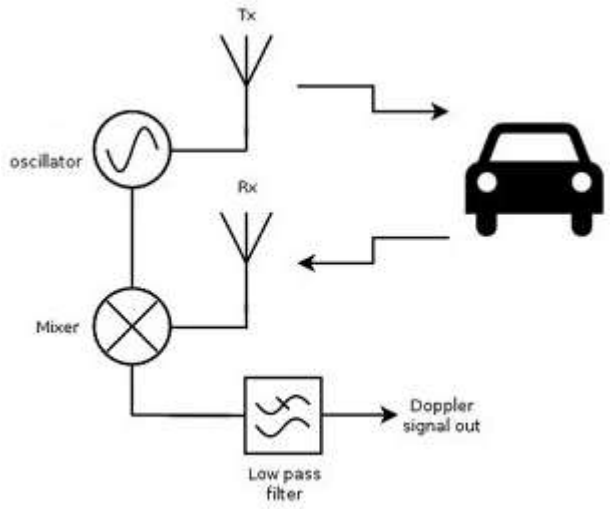
```
#pragma DATA_SECTION (dane, ".input")
DATA dane[2048];
```

- Nazwa sekcji *.input* jest przykładowa, można użyć dowolnej. Pozostałe dwie sekcje są zdefiniowane w kodzie procedury FFT.

Praktyczny projekt – radar dopplerowski

Zastosowanie procesora sygnałowego do analizy sygnału z mikrofalowego, dopplerowskiego czujnika radarowego.

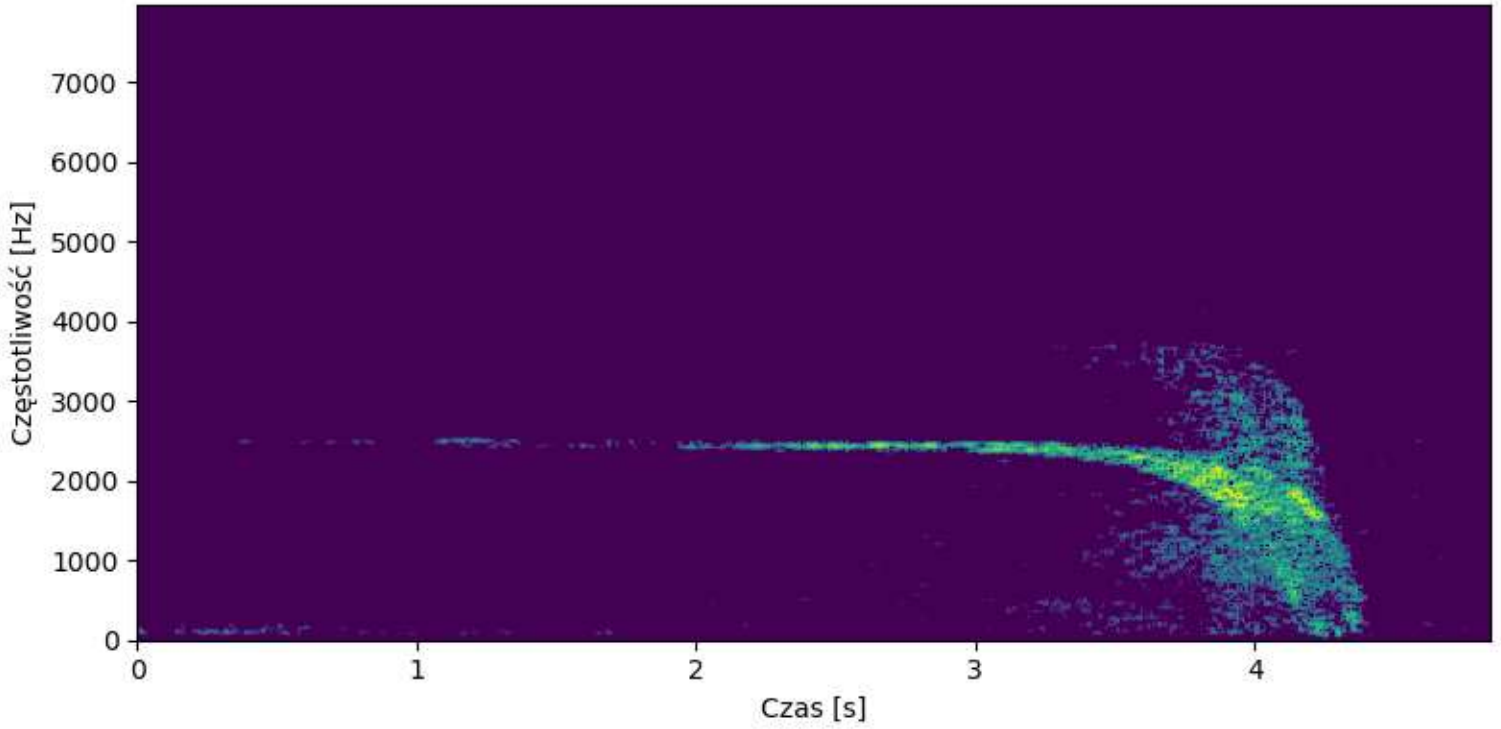
- Nadajnik emituje falę elektromagnetyczną – sinus o częstotliwości 24,125 GHz.
- Odbiornik zbiera falę odbitą od obiektu.
- Na skutek efektu Dopplera, odbita fala ma inną częstotliwość niż nadana.
- Różnica częstotliwości jest proporcjonalna do prędkości obiektu.
- Sygnał różnicowy z miksera czujnika leży w paśmie akustycznym.



Spektrogram sygnału z czujnika

Sygnał różnicowy – spektrogram dla przejazdu samochodu.

Nasze zadanie: wyodrębnić dominującą składową widma, znaleźć jej częstotliwość i przeliczyć na prędkość pojazdu.



Schemat przetwarzania sygnału

- Przychodzące próbki sygnału zapisujemy w buforze kołowym o rozmiarze 2048 próbek (liczby *short* w formacie Q15).
- Okno analizy jest przesuwane o 1024 próbek (zakładkowanie ½ długości okna).
- 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,
  - obliczamy FFT,
  - obliczamy widmo mocy (kwadrat modułu widma),
  - szukamy maksimów widmowych,
  - jeżeli je znajdziemy – obliczamy prędkość pojazdu.

Funkcja okna

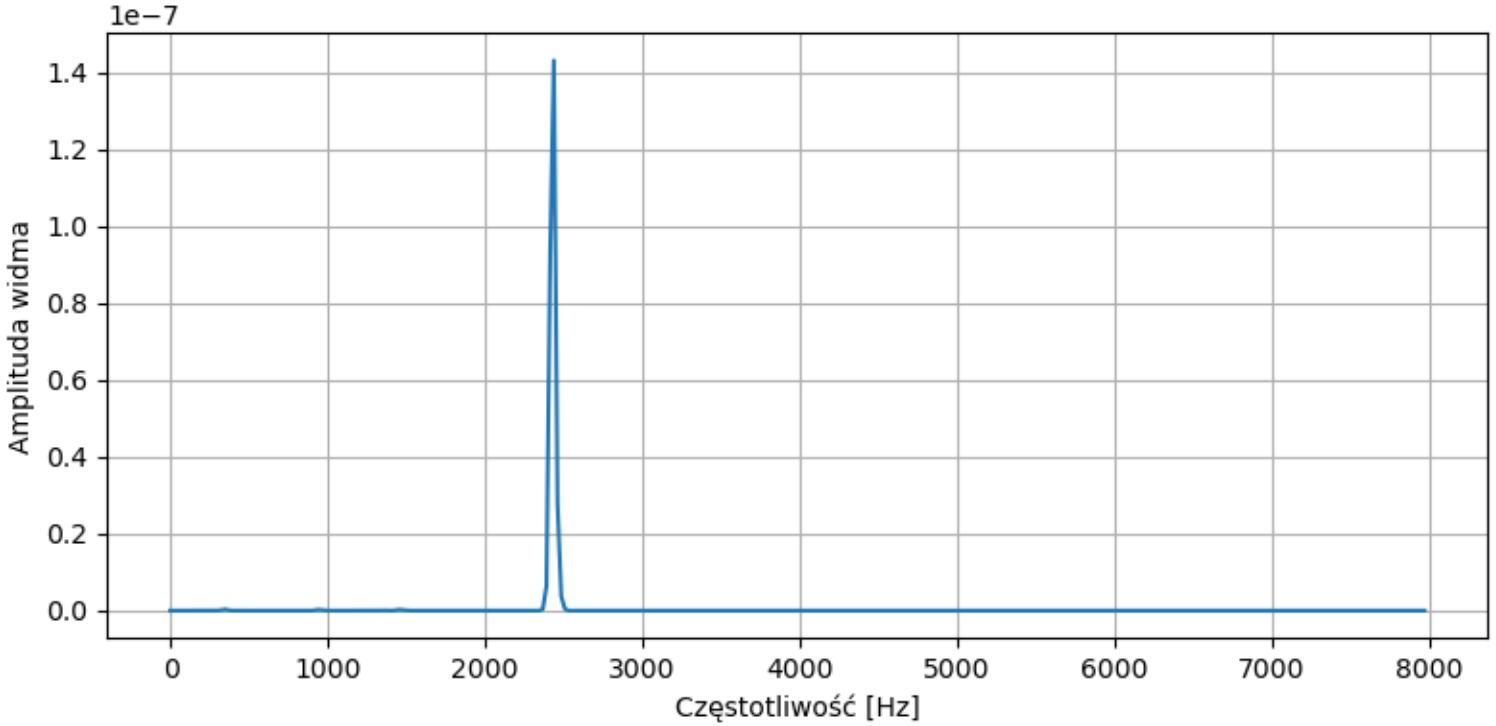
- Nie ma sensu obliczanie wartości okna Hamminga za każdym razem (choć jest to możliwe). Współczynniki są stałe dla ustalonej długości okna.
- Obliczamy wartości okna w zewnętrznym programie, konwertujemy do formatu Q15 i zapisujemy je w stałej tablicy (*const short*) w kodzie C.
- Uwaga: maksymalna wartość okna wynosi 1. Nie można zapisać jedynki w Q15! Należy przeskalować wartości zmiennoprzecinkowe aby nie było przepełnienia.

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.

Widmo dla pojedynczego bloku

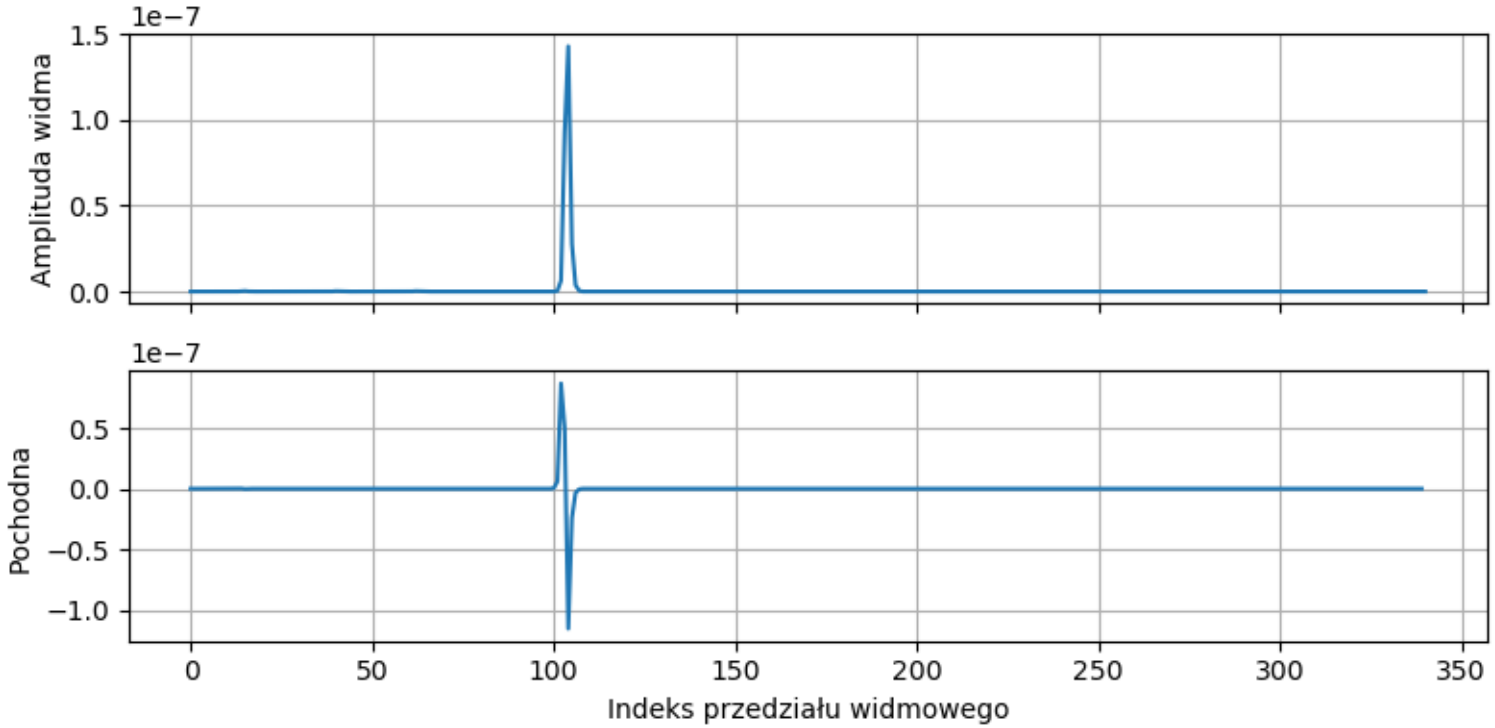
Następnie analizujemy obliczone widmo, szukając maksimum.



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).



Obliczenie prędkości

Znajdujemy maksimum widma dla wartości n = 104.
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 (dokładne obliczenia: 54,541)

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 (×):