ZAAWANSOWANE ALGORYTMY DSP

w zastosowaniach telekomunikacyjnych

Opracowanie: Grzegorz Szwoch

Politechnika Gdańska, Katedra Systemów Multimedialnych

Wstęp

Cztery algorytmy wybrane spośród bardziej zaawansowanych metod przetwarzania sygnałów stosowanych w telekomunikacji, przedstawione w praktycznych zastosowaniach:

- zmiana częstotliwości próbkowania,
  decymacja i interpolacja sygnału
- usuwanie zmiennych zakłóceń – filtry adaptacyjne
- detekcja częstotliwości – autokorelacja
- demodulacja sygnału – sygnał analityczny i transformator Hilberta

Zmiana częstotliwości próbkowania

PROBLEM #1: częstotliwość próbkowania 48 kHz
jest za mała dla naszych potrzeb.

- Nadpróbkowanie (*oversampling*)
  – przetwarzanie sygnału z większą częstotliwością próbkowania niż docelowa.
- W technice audio często stosuje się 4-krotne nadpróbkowanie, czyli fs = 192 kHz.
- Wada: więcej obliczeń, przy 4-krotnym nadpróbkowaniu DSP musi wykonać 4 razy więcej operacji w tym samym czasie.

Decymacja

- Na wyjściu potrzebujemy sygnał z *fs* = 48 kHz.
- Załóżmy *fs* = 192 kHz. Mamy 4 razy za dużo próbek.
- Weźmy tylko co czwartą próbkę. Mamy tyle próbek, ile potrzebujemy.
- Zanim pominiemy próbki, musimy zastosować filtr dolnoprzepustowy o częstotliwości granicznej równej docelowej cz. Nyquista (filtr decymacyjny).
  W naszym przypadku: *fg* = 24 kHz.
- Jeżeli nie zastosujemy filtru, wystąpi aliasing widma.

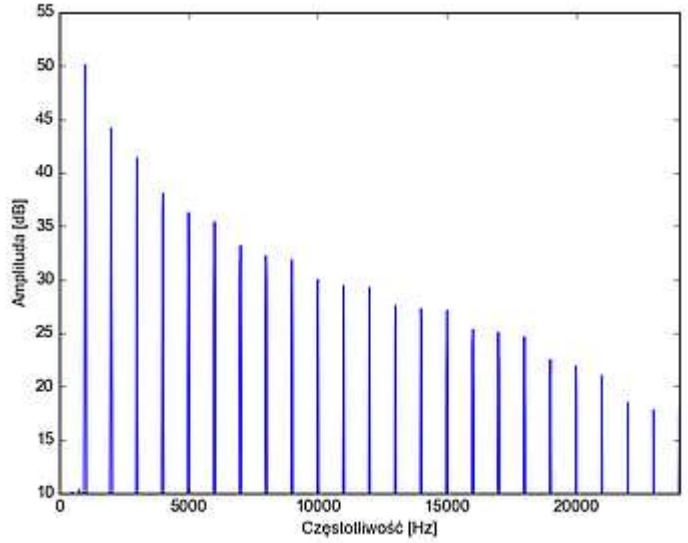
Decymacja

Decymacja z *fs1* do *fs2* = *fs1* / *D*:

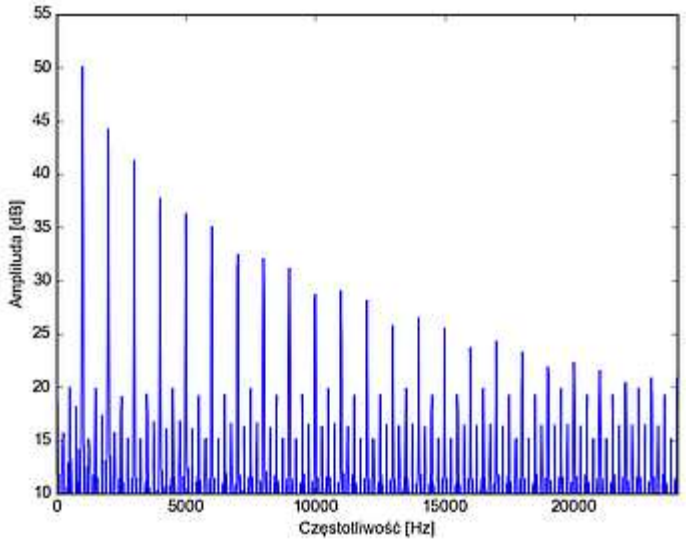
- najpierw filtr dolnoprzepustowy o *fg* = *fs2* / 2,
- następnie wzięcie co *D* próbki.

Przykład: generowanie sygnału piłokształtnego *f* = 1 kHz:

Nadpróbkowanie 4x i decymacja

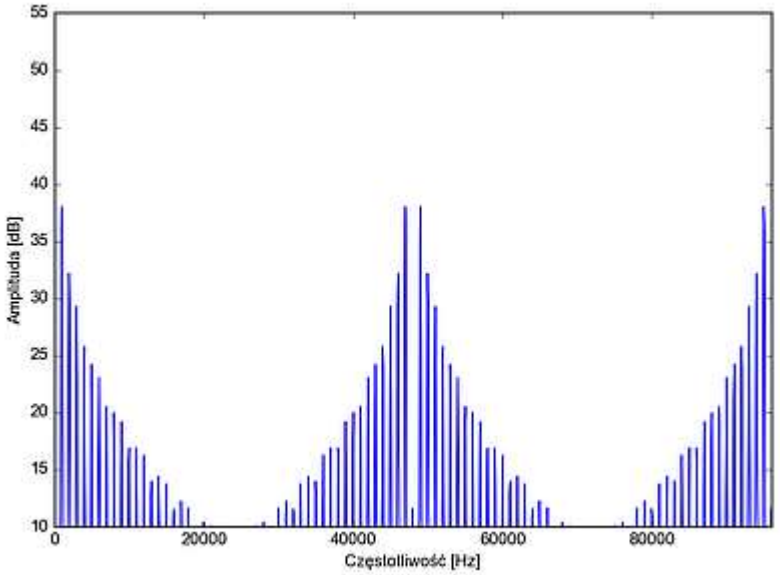


Bez nadpróbkowania



Interpolacja

- Mamy sygnał z *fs* = 48 kHz. Jak go nadpróbkować do 192 kHz?
- Wstawmy 3 zera pomiędzy każdą parę próbek.
- Mamy tyle próbek, ile potrzeba.
- Widmo: prawie dobrze…

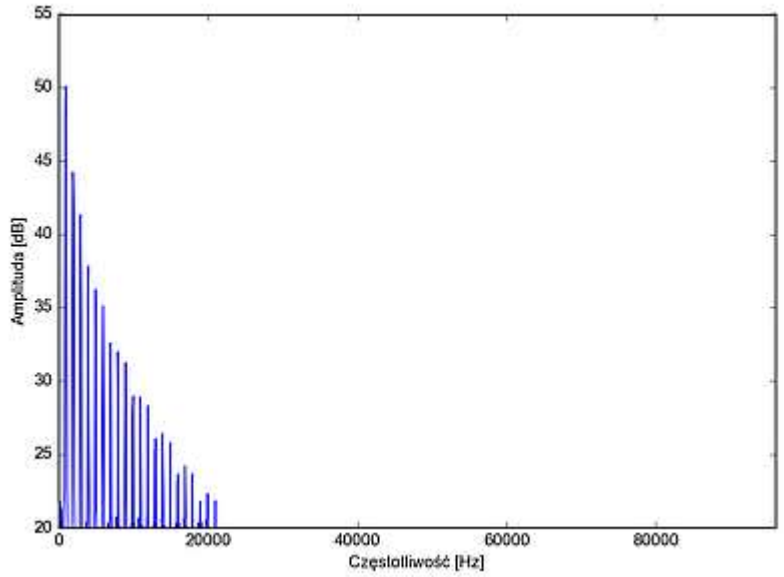


Interpolacja

Interpolacja z *fs1* do *fs2* = *fs1* * *L*:

- najpierw wstawienie (*L*-1) zer pomiędzy każdą parę próbek,
- następnie filtr dolnoprzepustowy o *fg* = *fs1* / 2.

Przykład: sygnał piłokształtny *f* = 1 kHz:



Przepróbkowanie

- Problem #1A: mamy sygnał o *fs1* = 44 100 Hz,
  a potrzebujemy *fs2* = 48 000 Hz.
- Przepróbkowanie (*resampling*)
  zmiana częstotliwości próbkowania (*L*/*D*) razy:
  - wstawienie (*L*–1) zer między każdą parę próbek,
  - filtr dolnoprzepustowy (wystarczy jeden),
  - wzięcie co *D* próbkę.
- *L* / *D* = 48000 / 44100 = 160 / 147
  a więc interpolacja *L* = 160 i decymacja *D* = 147

Funkcje z DSPLIB

- Filtr decymacyjny:

firdec	*Decimating FIR Filter*
Function	ushort oflag = firdec (DATA *x, DATA *h, DATA *r, DATA *dbuffer , ushort nh, ushort nx, ushort D)

- Filtr interpolacyjny:

firinterp	*Interpolating FIR Filter*
Function	ushort oflag = firinterp (DATA *x, DATA *h, DATA *r, DATA *dbuffer , ushort nh, ushort nx, ushort I)

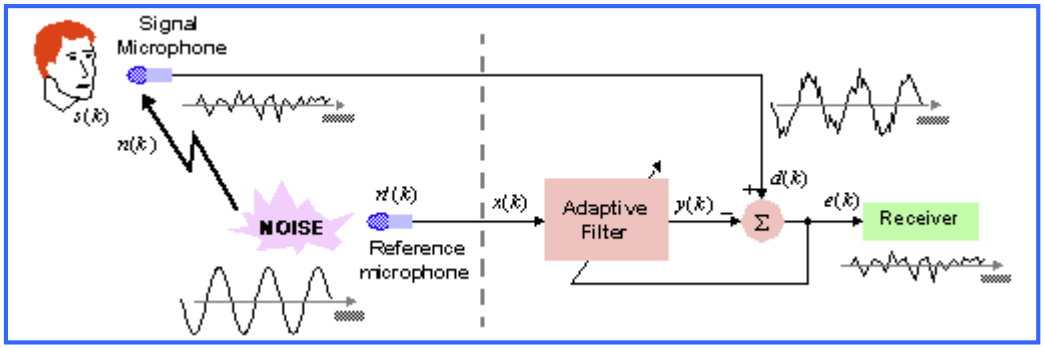
Usuwanie zakłóceń

PROBLEM #2

- Zestaw głośnomówiący w samochodzie.
- Mikrofon zbiera mowę oraz hałas z kabiny.
- Gdyby zakłócenia były stacjonarne, moglibyśmy zaprojektować filtr do usuwania zakłóceń.
- Zakłócenia są zmienne, więc to nie zadziała.
- Potrzebny jest filtr, który będzie adaptował swoją charakterystykę do zmiennego charakteru zakłóceń.

Filtry adaptacyjne

- Filtry adaptacyjne to takie filtry, których współczynniki są modyfikowane przez algorytm.
- Sygnał referencyjny: z głównego mikrofonu.
- Sygnał filtrowany: z drugiego mikrofonu, który zbiera tylko hałas, nie mowę.
- Sygnał błędu: różnica między wynikiem filtracji a sygnałem referencyjnym.
- Adaptacja współczynników filtru w taki sposób, aby zminimalizować błąd filtracji.



Algorytm LMS

- LMS – *Least Mean Squares* – metoda najmniejszych kwadratów.
- Założenie: średnia kwadratów wartości sygnału błędu powinna być jak najmniejsza.
- Im większa jest wartość błędu, tym bardziej należy zmodyfikować współczynniki filtru.
- Adaptacja współczynników:

$$w_k(n+1) = w_k(n) + \mu\, x(n-k)\, e(n)$$

(μ - krok adaptacji – szybkość zmian)

Filtry adaptacyjne w DSPLIB

dlms *Adaptive Delayed LMS Filter*

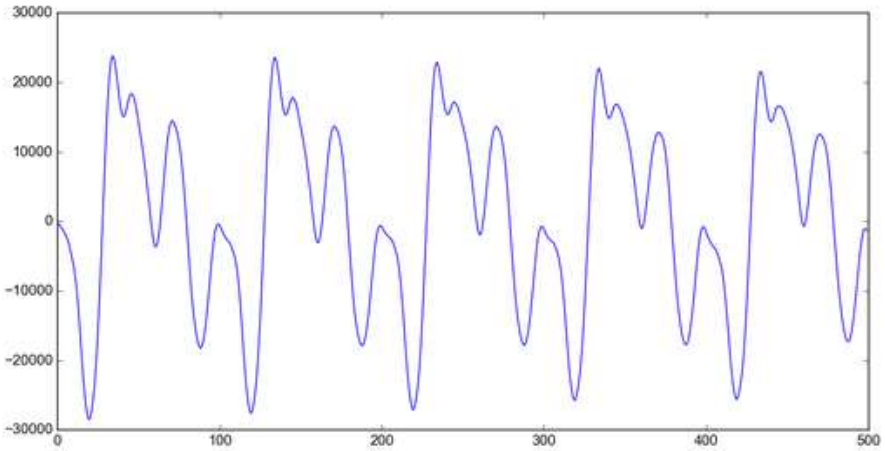
Function ushort oflag = dlms (DATA *x, DATA *h, DATA *r, DATA *des, DATA *dbuffer, DATA step, ushort nh, ushort nx)

- *x* – sygnał wejściowy
- *h* – współczynniki filtru
- *r* – bufor na sygnał po filtracji
- *des* – sygnał referencyjny
- *step* – krok adaptacji
- *nh* – liczba współczynników filtru,
- *nx* – liczba przetwarzanych próbek

Wyznaczanie wysokości dźwięku

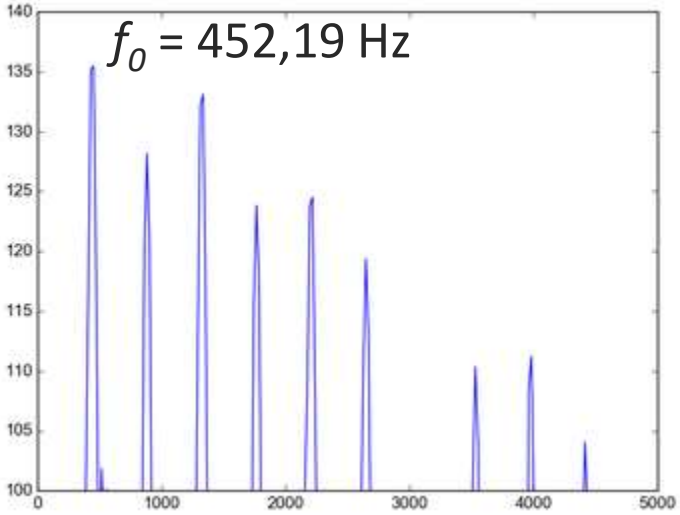
PROBLEM #3:

- Mamy nagrany dźwięk instrumentu muzycznego, np. trąbki. Chcemy znać jego wysokość na skali muzycznej.
- Dźwięki muzyczne są pseudo-okresowe i harmoniczne.
- Pseudo-okres – najkrótszy powtarzalny fragment.
- Częstotliwość podstawowa – odwrotność pseudo-okresu, wyznacza wysokość dźwięku na skali muzycznej, np. a^1.



Wysokość dźwięku przez FFT

- Podejście intuicyjne: liczymy FFT i szukamy pierwszego maksimum („prążka”).
- Metoda bardzo zawodna:
  - często znalezienie pierwszego prążka jest trudne (nie musi on być dominujący),
  - mała dokładność – zbyt mała rozdzielczość FFT
  - możemy obliczyć maksimum metodą opisaną na wcześniejszym wykładzie

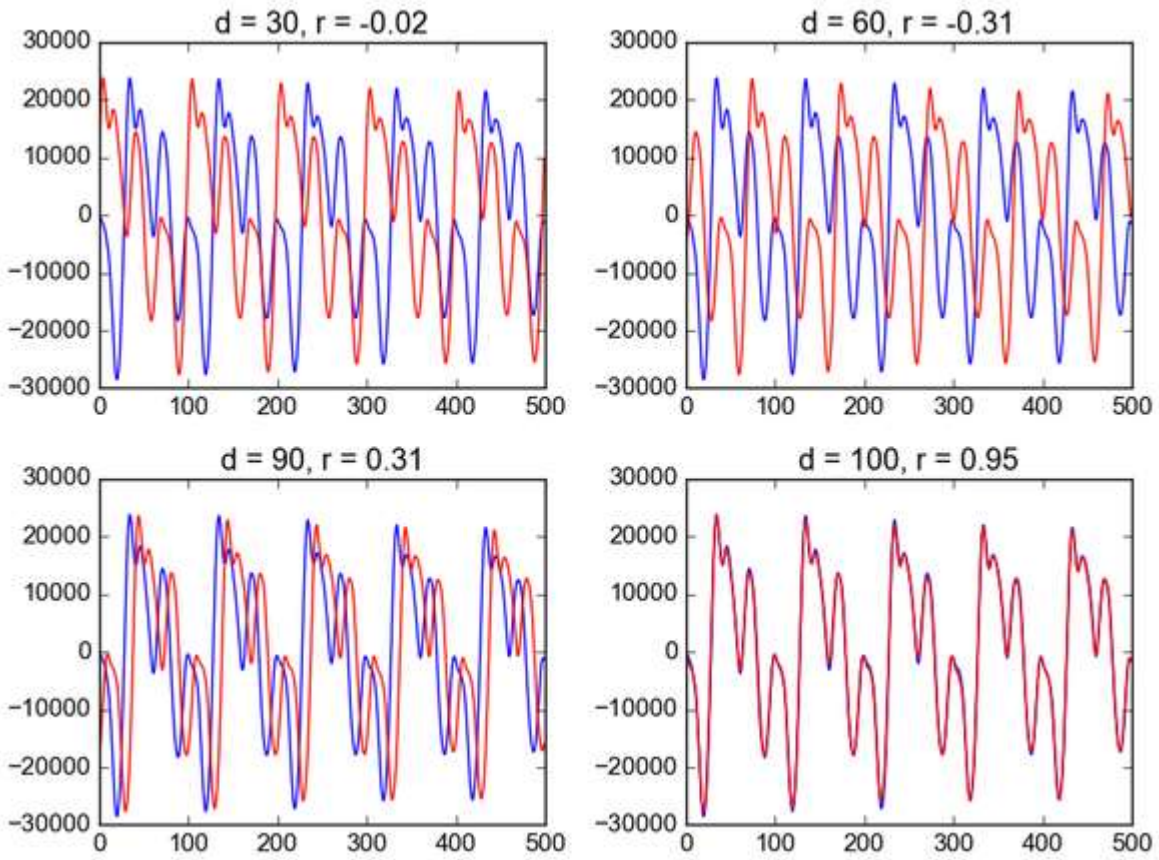


Autokorelacja

- Współczynnik autokorelacji: miara podobieństwa sygnału do jego kopii przesuniętej w czasie.
  Wartości od -1 do 1. Wartość 0: brak korelacji.
- Funkcja autokorelacji: wartości współczynnika dla różnych przesunięć (*lag*).
- Obliczamy współczynnik przesuwając sygnał kolejno o coraz większą liczbę próbek.
- Maksimum funkcji oznacza największe podobieństwo sygnałów dla danego przesunięcia.
- Czyli: pierwsze maksimum funkcji korelacji wyznaczy nam pseudo-okres sygnału, a więc i częstotliwość dźwięku.

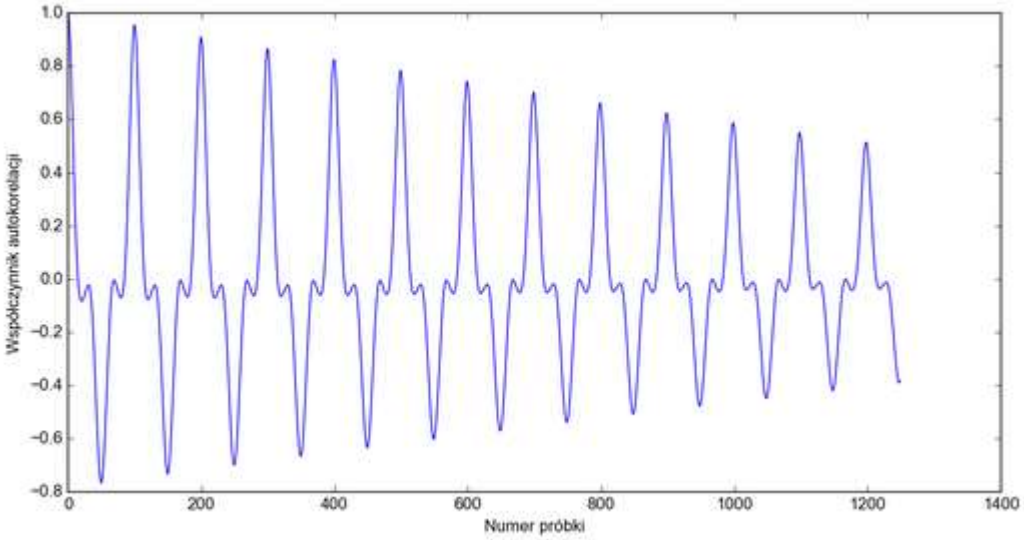
Autokorelacja

Współczynnik autokorelacji *r* dla różnych przesunięć *d*



Wysokość metodą autokorelacji

- Wykres funkcji autokorelacji wykazuje maksima dla kolejnych wielokrotności pseudo-okresu.
- Maksimum dla d = 100, czyli:
  f_0 = 44100 / 100 = 441 Hz



Autokorelacja

Inny przykład: wykrywanie obecności mowy
w zaszumionym sygnale.

- Sygnał mowy jest pseudo-okresowy: w funkcji autokorelacji będą obecne maksima.
- Szum jest nieskorelowany – brak maksimów.
- Bramkowanie sygnału: sprawdzenie ile wartości funkcji autokorelacji dla sygnału w buforze przekracza zadany próg.

Autokorelacja w DSPLIB

acorr	*Autocorrelation*
Function	ushort oflag = acorr (DATA *x, DATA *r, ushort nx, ushort nr, type)

Funkcję autokorelacji można również policzyć za pomocą FFT:

- obliczenie widma sygnału,
- obliczenie kwadratu modułu widma,
- obliczenie odwrotnej transformaty (IFFT).

Analiza złożonego sygnału

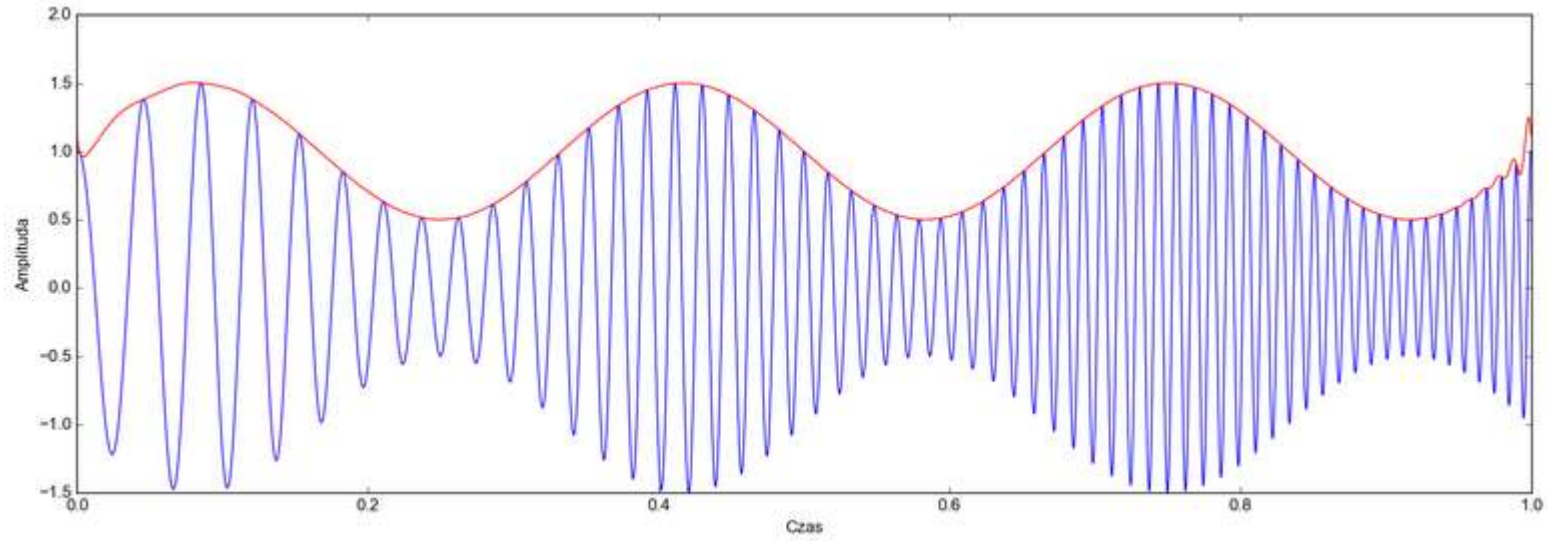
PROBLEM #4

- Mamy następujący złożony sygnał:
  - sygnał świergotowy (*chirp*) o liniowo narastającej częstotliwości od 20 do 100 Hz,
  - zmodulowany amplitudowo: amplituda sygnału przemnożona przez sygnał sinus f = 3 Hz.
- Dla dowolnej chwili chcemy znaleźć:
  - amplitudę obwiedni sygnału,
  - chwilową wartość częstotliwości.

Analiza złożonego sygnału

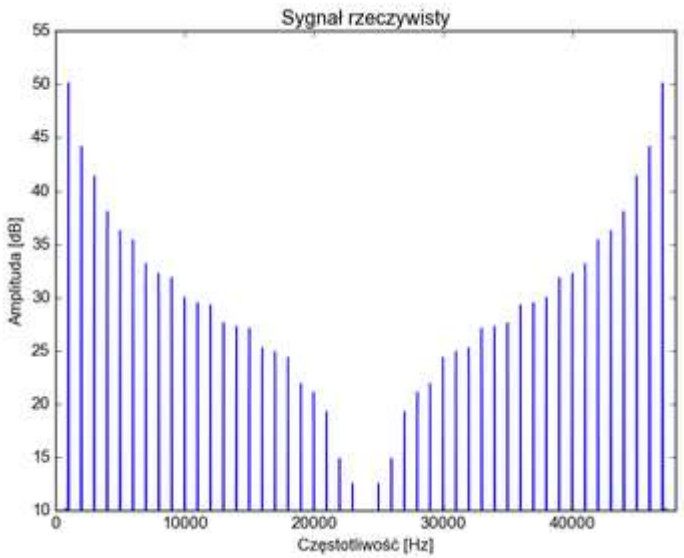
Tak wygląda nasz sygnał:

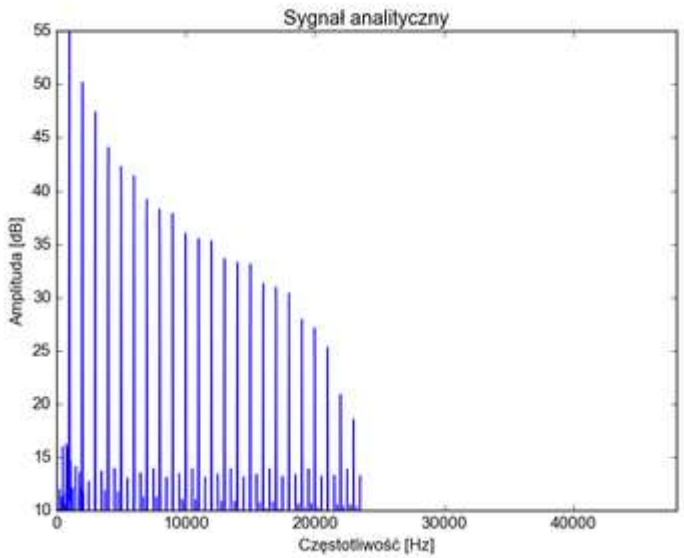
- kolor niebieski – wartości próbek sygnału,
- kolor czerwony – obwiednia amplitudy.



Sygnał analityczny

- Sygnał rzeczywisty: dwie kopie widma od 0 do *fs*.
- Wyzerujmy „drugą” kopię widma.
- Powstanie sygnał zespolony, nazywany sygnałem analitycznym.





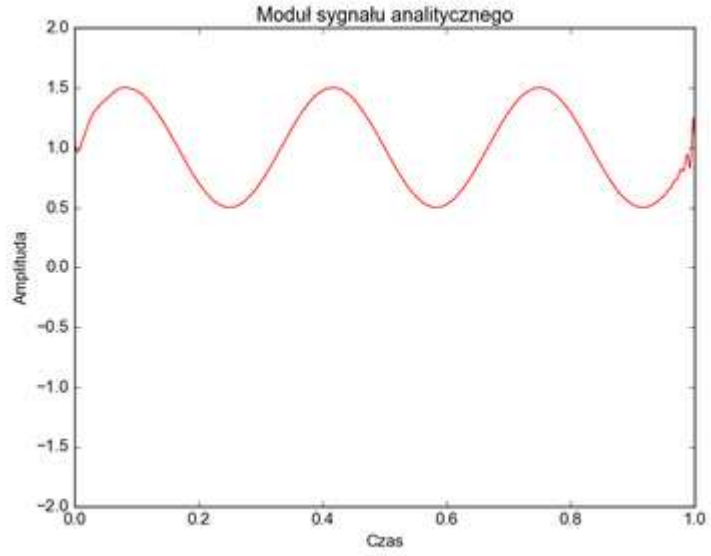
Obwiednia sygnału

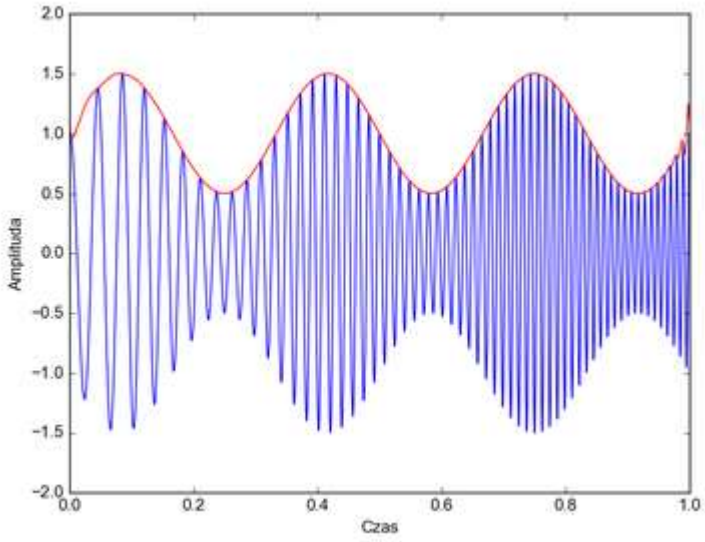
Co nam to dało:

- moduł sygnału analitycznego $r(n)$:

$$y(n) = \sqrt{\mathrm{Re}(r(n))^2 + \mathrm{Im}(r(n))^2}$$

jest obwiednią oryginalnego sygnału.





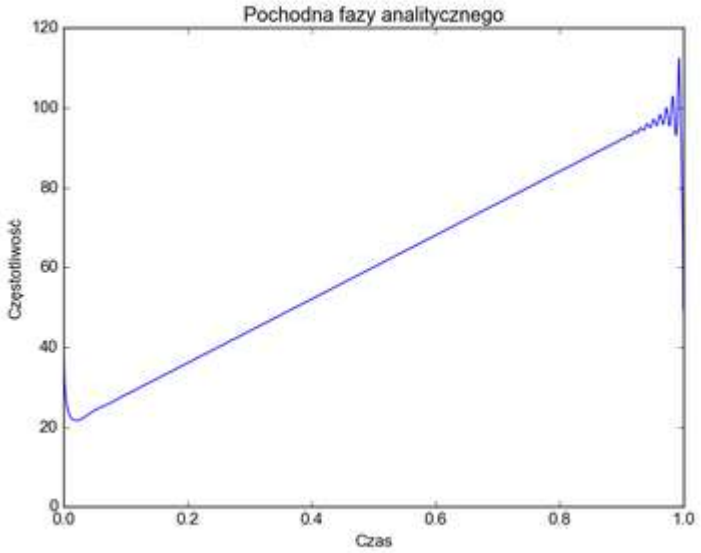
Częstotliwość chwilowa

- faza chwilowa sygnału analitycznego:

$$\varphi(n) = \arctan\left(\frac{\mathrm{Im}(r(n))}{\mathrm{Re}(r(n))}\right)$$

- częstotliwość chwilowa
  = pochodna fazy:

$$f(n) = \frac{f_s}{2\pi}\left(\varphi(n) - \varphi(n-1)\right)$$



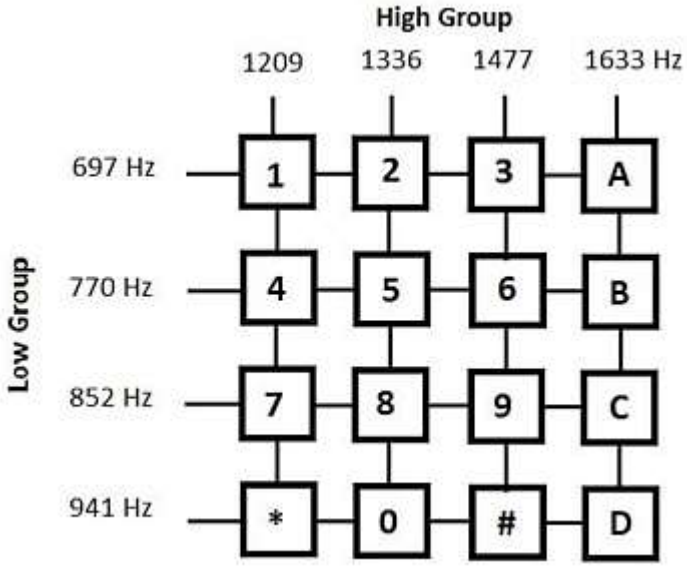
Transformator Hilberta w DSPLIB

- Algorytm przetwarzający sygnał rzeczywisty w analityczny nazywa się transformatorem Hilberta.
- Może to być filtr FIR lub można wykonywać operacje na widmie (FFT).
- W bibliotece DSPLIB:

hilb16	*FIR Hilbert Transformer*
Function	ushort oflag = hilb16 (DATA *x, DATA *h, DATA *r, DATA *dbuffer, ushort nx, ushort nh)

Bonus - DTMF

- DTMF – *Dual-Tone Multi Frequency*
- Metoda kodowania cyfr w telekomunikacji.
- Każda cyfra jest reprezentowana przez dwuton - dwa sinusy, dwie spośród 8 częstotliwości.
- Np. cyfra 6:
  770 Hz + 1477 Hz
- Zastosowanie
  – np. wybieranie tonowe w telekomunikacji.



Bonus - DTMF

„Zadanie domowe” – do samodzielnego zastanowienia się. Jak zrobić detektor DTMF za pomocą procesora sygnałowego?

- Detekcja początku i końca dwutonu:
  - jak wykryć cyfrę, ale jej nie powtarzać?
- Detekcja częstotliwości obu tonów:
  - filtry? FIR czy IIR?
  - FFT?
  - może inna metoda?

Podobny, ale trudniejszy problem:
jak zrobić detektor sygnałów nadawanych alfabetem Morse’a?