

1. Filtr cyfrowy IIR (2+1 pkt)

W pliku `butter.mat` znajdują się z -zera, p -bieguny i k -współczynnik wzmocnienia analogowego filtra Butterwortha typu BP o częstotliwościach granicznych odpowiednio dolna 1189 i górna 1229 Hz.

Używając transformaty biliniowej wykonaj konwersję analogowego filtra $H(s)$ do postaci cyfrowej $H(z)$. Załóż, że częstotliwość próbkowania to $f_s = 16$ kHz.

Na pierwszym rysunku narysuj, charakterystykę amplitudowo-częstotliwościową filtra analogowego i cyfrowego. Zaznacz częstotliwości graniczne. Porównaj charakterystykę amplitudowo-częstotliwościową filtra cyfrowego z jego analogowym prototypem. Dlaczego częstotliwości graniczne nie są w tych samych miejscach?

Wygeneruj sygnał cyfrowy o czasie trwania 1 s, częstotliwości próbkowania $f_s = 16$ kHz, złożony z sumy dwóch harmonicznym o częstotliwościach odpowiednio: 1209 i 1272 Hz.

Wykonaj cyfrową filtrację sygnału za pomocą wyżej opisanego filtra. Filtr zaimplementuj (wykonaj) sam, bez użycia funkcji `filter(...)` lub podobnej. Porównaj oba sygnały w dziedzinie czasu i częstotliwości. Następnie użyj do filtracji funkcji `filter(...)` i porównaj czy otrzymany sygnał jest taki sam jak z własnej implementacji algorytmu filtracji.

Zadanie opcjonalne (+1 pkt): wykonaj korektę prototypu, tak aby częstotliwości graniczne wystąpiły w oczekiwanych miejscach. Wykorzystaj w tym celu technikę nazywaną: pre-warping (wzór (11.23) w [TZ]), tzn. zaprojektuj filtr analogowy na inną pulsację „analogową”, związaną z wymaganą pulsacją „cyfrową” wzorem:

$$\omega = \frac{2}{T} \operatorname{tg}\left(\frac{\Omega}{2}\right), \text{ gdzie } \omega = 2\pi f_a, \quad \Omega = 2\pi \frac{f_c}{f_s}, \quad T = \frac{1}{f_s}$$

Na jednym rysunku wyświetl charakterystyki amplitudowo-częstotliwościowe H :

- prototypu analogowego przed korekcją $H(s)$,
- filtra cyfrowego $H(z)$ powstałego metodą konwersji z $H(s)$,
- prototypu analogowego z korekcją pre-warping $H_w(s)$,
- filtra cyfrowego $H_w(z)$ powstałego metodą konwersji z $H_w(s)$.

2 Dekodowanie DTMF (1+3 pkt)

DTMF (*ang.* Dual Tone Multi Frequency) to nazwa systemów do sygnalizacji tonowej używanych w telefonach analogowych. Jest to archaiczny system, ale wciąż stosowany np. do wybierania opcji w automatycznym call-center.

Każdemu przyciskowi klawiatury odpowiada sygnał dźwiękowy składający się z sumy dwóch „tonów” (harmonicznym). Mapowanie znaku do częstotliwości składowych przedstawiono w poniższej tabeli.

	1209 Hz	1336 Hz	1477 Hz
697 Hz	1	2	3
770 Hz	4	5	6
852 Hz	7	8	9
941 Hz	*	0	#

I tak, przyciskając cyfrę „4” usłyszymy dźwięk złożony z tonów (częstotliwości) 1209 Hz i 770 Hz.

Celem ćwiczenia jest zdekodowanie „wystukanej” na klawiaturze sekwencji znaków na podstawie zaszumionego sygnału audio. Sekwencje `s0.wav...s9.wav` z pliku `lab06.zip` to zapisy audio 5-cio cyfrowych kodów PIN. Wybierz plik odpowiadający przedostatniej cyfrze Twojego numeru legitymacji studenckiej i rozkoduj go. Sygnał `s.wav` to sygnał wzorcowy składający się z sekwencji `[1,2,3,4,5,6,7,8,9,*,0,#]`.

Rozkoduj sekwencje „ręcznie” patrząc na wykres czasowo-częstotliwościowy tego sygnału (funkcja `spectrogram(sX, 4096, 4096-512, [0:5:2000], fs)`).

Przefiltruj sygnał `sX` cyfrowym filtrem BP z ćwiczenia 1. Porównaj spektrogramy przed i po filtracji. Narysuj na jednym rysunku oba sygnały w dziedzinie czasu. Skompensuj opóźnienie sygnału wprowadzone przez filtrację.

Opcjonalne (+1 pkt): Zaprojektuj transformatę DtFT z algorytmem Goertzla nastrojoną na częstotliwości z powyższej tabeli (patrz [TZ]). Czy analiza wykonana w ten sposób jest łatwiejsza? Jeżeli tak to pod jakim względem.

Opcjonalne (+1 pkt): Zaprojektuj pasmowo-przepustowe filtry IIR nastrojone na częstotliwości harmoniczne z powyższej tabeli (użyj filtru IIR z jednym biegunem). Porównaj energie sygnałów na wyjściu wszystkich filtrów. Energia dwóch z nich powinna być zdecydowanie wyższa. Ta para odpowiada poszukiwanej cyfrze.

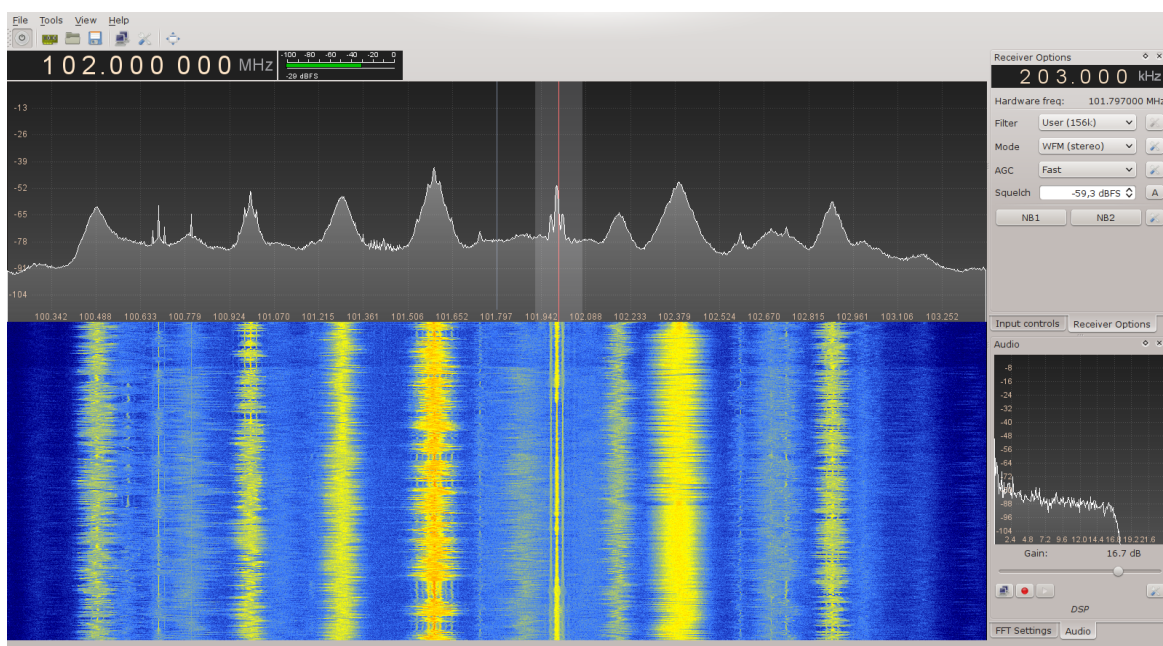
Opcjonalnie (+1 pkt), zaprojektuj algorytm decyzyjny, który w sposób automatyczny będzie rozpoznawał wprowadzany kod. Przetestuj go na wszystkich sekwencjach.



Zadanie opcjonalnie. Plik `challenge.wav` zawiera sekwencje symboli DTMF o pogarszającej się jakości. Liczba symboli jest niejawną, długość symboli nie jest stała. Dodatkowe **20 pkt** otrzyma osoba, która zdekoduje najdłuższą poprawną sekwencję symboli. Na adres kwant@agh.edu.pl należy wysłać (do następnych zajęć) program dekodujący powyższą sekwencję. Liczba przyznanych punktów może zostać obniżona za „nieładny” program (np. strojenie algorytmu pod konkretny symbol). Zwycięzca jest tylko jeden!

3. Radio FM – dekodowanie (2+1 pkt)

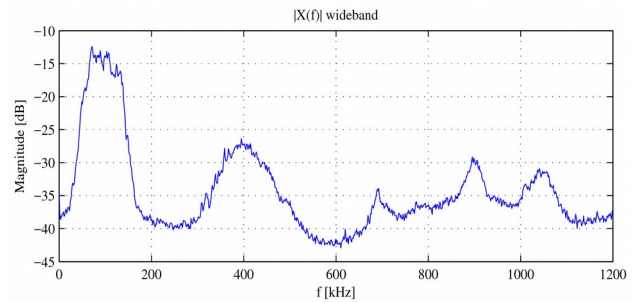
Założmy, że pojedyncza stacja analogowego radia FM znajduje się w paśmie $101\text{ MHz} \pm 100\text{ kHz}$ ($f_n=101\text{ MHz}$ to nośna sygnału). Aby efektywnie przetwarzać taki sygnał należy go przenieść do niższej częstotliwości. Dlatego część analogowa tunera cyfrowego wykonuje konwersję pasma, np. $[100\text{ MHz} \dots 103.2\text{ MHz}]$ do pasma $[0\text{ MHz} \dots 3.2\text{ MHz}]$ (mnożąc sygnał $x(t)$ oddzielnie przez $\cos(2f_0\pi t)$ oraz $-\sin(2f_0\pi t)$, $f_0=100\text{ MHz}$ otrzymujemy analogowe sygnały $y_c(t)$ i $y_s(t)$). Potem sygnały te są filtrowane analogowym filtrem dolnoprzepustowym o częstotliwości granicznej 3.2 MHz i próbkowane przetwornikiem A/C z $f_s=3.2\text{ MHz}$. Otrzymywane są w ten sposób dwie sekwencje próbek: $I(n)$ z $y_c(t)$ oraz $Q(n)$ z $y_s(t)$, które są dalej przetwarzane przez część cyfrową odbiornika radia FM. W paśmie $[0\text{ MHz} \dots 3.2\text{ MHz}]$ jest zawartych kilka stacji radiowych co widać na poniższym widmie sygnału.



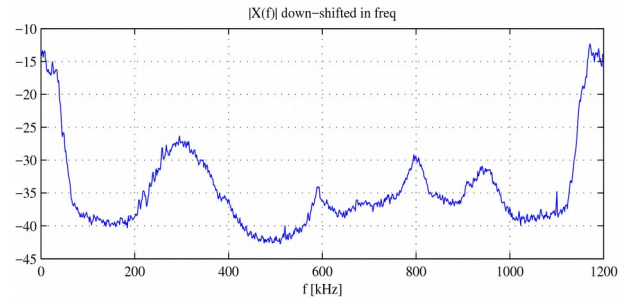
Dekodowanie sygnału FM polega na:

odfiltrowaniu pojedynczej stacji

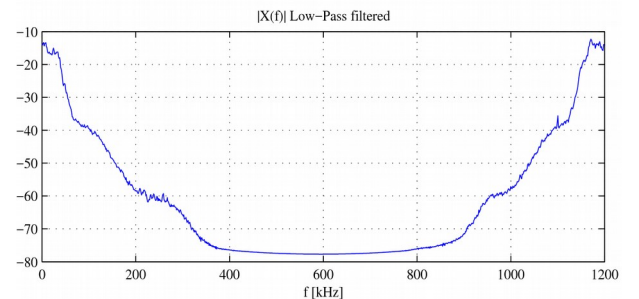
rysunek przedstawia widmo sygnału sprowadzonego do pasma podstawowego przez tuner



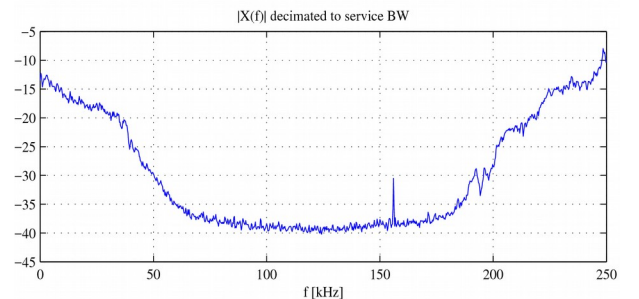
przesunięcie (ponowne!) widma sygnału **wideband_signal** z częstotliwości 0.1 MHz (odpowiednik 100.37 MHz) do 0 Hz



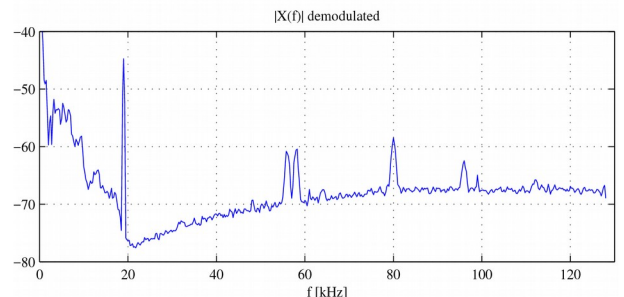
zastosowaniu **filtru LP** o szerokości pasma np. 80 kHz na sygnale **wideband_signal_filtered** co powoduje usunięcie pozostałych stacji radiowych z sygnału



zmiany częstotliwości próbkowania z 3.2 MHz na 160 kHz (pozostawienia co 20-tej próbki) – otrzymujemy w ten sposób sygnał **x**,

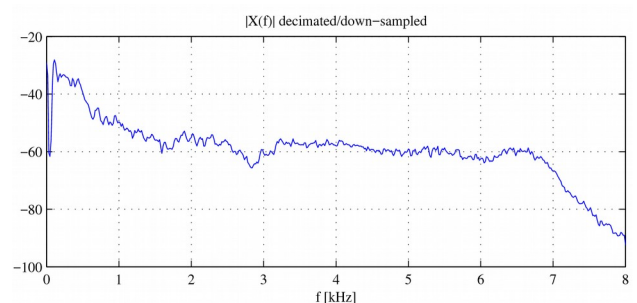


demodulacji FM sygnału **x** do sygnału **y**, w ten sposób uzyskujemy sygnał „hybrydowy”, doskonale na nim widać część mono, pilot 19 kHz, sygnał stereo i RDS

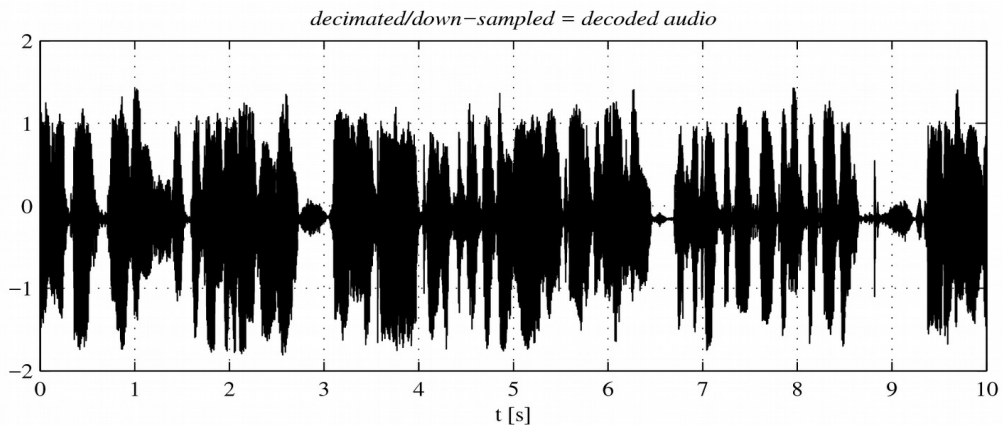


pozostawienie tylko sygnału mono znajdującego się w paśmie 0-16 kHz:

- filtracja LP (filtr o częstotliwości granicznej 16 kHz),
- zmiany częstotliwości próbkowania z 160 kHz do 32 kHz (pozostawienie co 5-tej próbki) – otrzymujemy sygnał **ym**,
- de-emfazy (słabego tłumienia wyższych częstotliwości).



W wyniku tych operacji uzyskujemy monofoniczny sygnał audio:



Poniżej przedstawiono kod programu cyfrowej części odbiornika radia FM. Pełny program znajduje się w pliku `decoder_fm.m`. Program działa „zgrubnie” (specjalnie!), należy go poprawić :). Zadania:

- narysuj charakterystyki czasowo-częstotliwościowe i widma gęstości mocy oryginalnego sygnału oraz sygnału w kolejnych punktach programu; do wyznaczania widma gęstości mocy użyj funkcji: `psd(spectrum.welch('Hamming',1024), wideband_signal(1:M), 'Fs', fs);`
- odszukaj częstotliwości, w których znajdują się stacje radiowe („górkę” na widmie gęstości mocy sygnału `wideband_signal`), spróbuj dekodować inne stacje,
- podmień w (1) istniejący filtr na cyfrowy IIR typu Butterworth LP rzędu 4 o częstotliwości granicznej 80 kHz,
- dodaj w (2) filtr antyaliasingowy: LP o częstotliwości granicznej 16 kHz, sprawdź na poprawnie działającym dekodерze, jaki ma wpływ pominięcie filtra antyaliasingowego,
- **opcjonalnie** (+1 pkt): zaprojektuj filtr de-emfazy (pkt 5) o płaskiej charakterystyce do 2.1 kHz i opadaniu 20 dB/dekadę powyżej tej częstotliwości (cyfrowy Butterworth LP),
 - narysuj charakterystykę amplitudowo-częstotliwościową zaprojektowanego filtra i docelowego filtra,
 - zaprojektuj filtr pre-emfazy (odwrotny do de-emfazy, ten który jest w nadajniku), porównaj charakterystyki obu filtrów, wykonaj filtrację filtrem pre-emfazy, następnie de-emfazy i sprawdź jak te operacje wpłynęły na sygnał.

```
% IQ --> complex
wideband_signal = s(1:2:end) + sqrt(-1)*s(2:2:end);

% Extract carrier of selected service, then shift in frequency the selected service to the baseband
wideband_signal_shifted = wideband_signal .* exp(-sqrt(-1)*2*pi*fc/fs*[0:N-1]);

% Filter out the service from the wide-band signal (1)
b=???; a=???;
wideband_signal_filtered = filter( b, a, wideband_signal_shifted );

% Down-sample to service bandwidth - bwSERV = new sampling rate
x = wideband_signal_filtered( 1 : fs/bwSERV : end );

% FM demodulation
dx = x(2:end).*conj(x(1:end-1));
y = atan2( imag(dx), real(dx) );

% Decimate to audio signal bandwidth bwAUDIO (2)
y = .... % antialiasing filter
ym = y(bwSERV/bwAUDIO); % decimate (1/5)

% De-emfaza, flat characteristics to 2.1 kHz, then falling 20 dB/decade
% (...)

% Listen to the final result
ym = ym-mean(ym); ym = ym/(1.001*max(abs(ym)));
soundsc( ym, bwAUDIO);
```