AKADEMIA GÓRNICZO-HUTNICZA

IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Elektrotechniki, Automatyki,

Informatyki i Elektroniki

Katedra Elektroniki

PRACA DYPLOMOWA

Inżynierska

Implementacja wybranej metody obiektywnej oceny jakości dźwięku lub mowy

Implementation of the chosen objective speech or audio quality assessment method

Krzysztof Czaja

Inżynieria Akustyczna

Opiekun pracy: Dr inż. Jakub Gałka

Kraków 2011/2012

Oświadczenie autora pracy

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i nie korzystałem ze źródeł innych niż wymienione w pracy.

….....

Podpis

Spis treści

1 Wstęp....5
1.1 Cel pracy....5
1.2 Założenia techniczne....5
2 Model pomiarowy....7
2.1 Informacje ogólne....7
2.2 Zarys modelu....9
2.3 Model psychoakustyczny....10
2.4 Model poznawczy....15
2.5 Parametry wyjściowe....16
3 Wykonanie....19
3.1 Informacje wstępne....19
3.2 Model ucha....24
3.3 Przetwarzanie wstępne....36
3.4 Parametry wyjściowe MOV....43
3.5 Ocena jakości....51
4 Weryfikacja....53
4.1 peaqb 1.0.beta....53
4.2 Test porównawczy....53
5 Podsumowanie....57
Słownik skrótów....59
Bibliografia....59
Załączniki....61

1 Wstęp

Jakość sygnału audio jest jednym z podstawowych czynników branych pod uwagę podczas projektowania cyfrowych systemów do transmisji radiowych, internetowych i telekomunikacyjnych transmisji strumieniowych, czy kodeków audio stosowanych do kompresji plików muzycznych. Jakość sygnału audio na charakter typowo subiektywny, dlatego najlepszym sposobem jej oceny są subiektywne „ślepe" testy odsłuchowe. Testy te są jednak kosztowne, czasochłonne i problematyczne, natomiast standardowe metody, jak na przykład: stosunek sygnału od szumu (SNR) czy zawartość harmonicznych (THD), nie oddają ocen subiektywnych. Aby umożliwić obiektywną ocenę jakości dającą wyniki zbliżone do testów subiektywnych International Telecommunication Union (ITU), stworzyło algorytm PEAQ (ang. Perceptual Evaluation of Audio Quality) do obiektywnej percepcyjnej oceny jakości sygnału audio. Praca ta zawiera opis metodyoraz jej implementacji programowej, jest oparta o rekomendację ITU-R BS.1387-1 [1]. Temat ten wybrałem ze względu na możliwość zastosowania tej metody zarówno do oceny różnego typu kodeków, jak i torów audio stosowanych do nagrań i postprodukcji studyjnej. Zdecydowałem się na ocenę jakości sygnału audio zamiast sygnału mowy, ze względu na bardziej ogólny charakter zagadnienia, natomiast, jako konkretną metodę wybrałem PEAQ, ponieważ jest połączeniem kilku metod, daje skorelowane wyniki z metodą subiektywną oraz posiada dokładną dokumentację i opis algorytmu.

1.1 Cel pracy

Celem pracy jest implementacja programowa wybranej rekomendacji ITU dotyczącej obiektywnej oceny jakości sygnału audio lub mowy oraz weryfikacja uzyskanego rozwiązania.

1.2 Założenia techniczne

Założeniem pracy jest napisanie programu, który przyjmuje dwa sygnały (referencyjny oraz testowany) i dokonuje oceny jakości sygnału testowanego. Program działa w trybie offline, jako sygnały wejściowe przyjmuje dwa pliki typu *wav* (mono lub stereo). Pierwszy plik jest sygnałem referencyjnym, natomiast drugi – testowanym.

Program porównuje sygnały zgodnie z algorytmem i w efekcie zwraca parametr zwany dalej stopniem obiektywnej różnicy ODG (ang. Objective Difference Grade). Oprócz wskaźnika ODG zwracany jest również wskaźnik zniekształceń DI (ang. Distortion Index) oraz parametry wyjściowe modelu MOV (ang. Model Output Variables), które będą opisane w dalszej części pracy, a które mogą posłużyć jako wskaźnik rodzaju zniekształceń mających największy wpływ na ogólny wynik.

2 Model pomiarowy

2.1 Informacje ogólne

Ponieważ zadaniem pomiaru obiektywnego jest częściowe zastąpienie pomiarów subiektywnych, dlatego na wstępie przedstawiono sposób przeprowadzania testów subiektywnych.

2.1.1 Pomiary subiektywne

Do subiektywnego pomiaru jakości audio są przygotowywane trzy sygnały oznaczone jako: A, B oraz C. Sygnał A jest zawsze znanym sygnałem referencyjnym, natomiast ukryty sygnał referencyjny oraz sygnał testowany są losowo oznaczane jako sygnały B oraz C. Zadaniem słuchacza jest ocena stopnia różnicy sygnału B w stosunku do sygnału A oraz sygnału C w stosunku do sygnału A. Słuchacz do oceny wykorzystuje skalę przedstawioną w tabeli (Tab. 2.1) [2].

Tab. 2.1. Skala subiektywnej oceny degradacji sygnałów audio.[2]

Degradacja	Ocena
niesłyszalna	5
słyszalna ale nie drażniąca	4
lekko drażniąca	3
drażniąca	2
bardzo drażniąca	1

Skala ta jest najczęściej nazywana skalą MOS (ang. Mean Opinion Score). Stosuje się ją nie tylko do określenia jakości dźwięku, czy mowy, ale także obrazu. Dokładne zalecenia do stosowania skali MOS można znaleźć w rekomendacji ITU-T P.800.

W wyniku pomiarów subiektywnych otrzymuje się stopień subiektywnej różnicy SDG (ang. Subjective Difference Grade), obliczany na podstawie ocen uzyskanych od słuchacza według wzoru 2.1.

$$SDG = O_T - O_R \tag{2.1}$$

gdzie:

SDG – stopień subiektywnej oceny

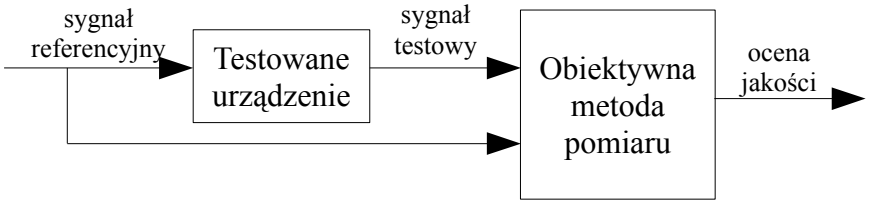
O_T – ocena sygnału testowanego

O_R – ocena sygnału referencyjnego

SDG powinno przyjmować wartości od – 4 do 0, gdzie 0 oznacza niesłyszalny stopień degradacji, a – 4 degradację bardzo drażniącą. Więcej informacji na temat przeprowadzania testów subiektywnych można znaleźć w rekomendacji |TU-R BS.1116 [2].

2.1.2 Pomiary obiektywne

Ogólną metodę obiektywnego pomiaru zastosowanego w tej pracy przedstawia rysunek 2.1. Na wejście testowanego urządzenia podaje się sygnał referencyjny, na wyjściu urządzenia otrzymuje się sygnał testowany, który jest porównywany z sygnałem referencyjnym zgodnie z zastosowaną metoda pomiaru.



Rys. 2.1. Zasada pomiarów obiektywnych. [1]

Zastosowana metoda pomiaru obiektywnego jest oparta o algorytm PEAQ. W wyniku otrzymujemy wskaźnik ODG, który odpowiada SDG w pomiarach subiektywnych. ODG również zawiera się w przedziale od – 4 do 0. Dokładność pomiarów jest na poziomie 0.1, jednak różnica na tym poziomie nie jest znacząca. Aby uzyskany wynik był poprawny sygnały powinny być zgodne ze sobą z dokładnością do 24 próbek.

2.1.3 Narząd słuchu

Aby zrozumieć złożoność problemu oraz zastosowane rozwiązania przydatnym będzie ogólne przybliżenie działania ludzkiego narządu słuchu.

Ucho składa się z trzech części: ucha zewnętrznego, środkowego oraz wewnętrznego.

Ucho zewnętrzne: fala dźwiękowa dociera do małżowiny usznej, a następnie przez kanał słuchowy do błony bębenkowej.

Ucho środkowe: błona bębenkowa pod wpływem padającej na nią fali dźwiękowej drga. Drgania te są wzmacniane przez trzy kosteczki słuchowe (młoteczek, kowadełko i strzemiączko) i przenoszone na okienko owalne.

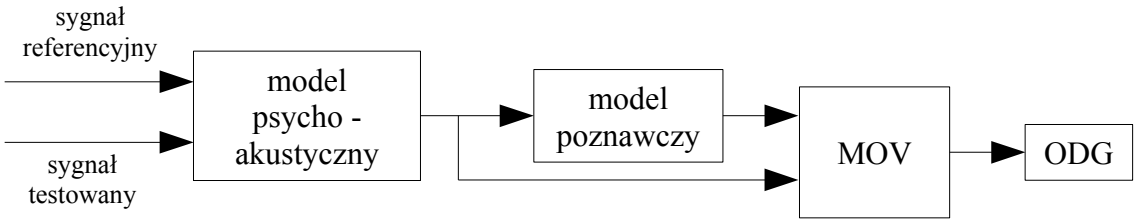
Ucho wewnętrzne (ślimak): drgania okienka owalnego, powodują zmiany ciśnienia płynu ślimakowego, które docierają do okienka okrągłego oraz powodują odkształcenie błony podstawnej ślimaka. Odkształcenia te są zależne od poziomu ciśnienia dźwięku, natomiast miejsce powstawania drgań zależy od jego częstotliwości. Drgania błony podstawnej powodują pobudzenie rzęsek organu Cortiego oraz wytworzenie potencjału, który powoduje pobudzenie komórek nerwowych. Dźwięk do neuronów jest przekazywany w formie impulsu. W zależności od częstotliwości dźwięku pobudzona zostaje inna grupa neuronów, natomiast częstość występowania impulsów przekazuje informację o głośności dźwięku.

Ucho jednak nie jest narządem doskonałym. Nie wszystkie częstotliwości są słyszane tak samo (ucho posiada własną charakterystykę częstotliwościową), jak również różnice w częstotliwościach są odbierane inaczej dla niskich i wysokich częstotliwości. Występują także zjawiska maskowania równoczesnego i maskowania nierównoczesnego. Dokładniejszy opis tych zjawisk znajduje się w dalszej części niniejszej pracy [3,4].

2.2 Zarys modelu

W obiektywnej metodzie pomiaru jakości dźwięku jako danymi wejściowymi są dwa sygnały: sygnał referencyjny oraz sygnał testowany. Sygnały te są cyfrowym odpowiednikiem fizycznych zmian ciśnienia odbieranych przez człowieka jako dźwięk. Aby możliwa była ocena jakości tych sygnałów, konieczne jest stworzenie dokładnego modelu działania ludzkiego narządu słuchu, jak również modelu poznawczego

reprezentującego sposób, w jaki mózg ocenia jakość usłyszanego dźwięku. Ogólny model systemu pomiarowego przedstawia schemat z rysunku 2.2.



Rys. 2.2. Ogólny model systemu. [1]

System pomiarowy składa się z modelu psychoakustycznego reprezentującego działanie ucha, oraz modelu poznawczego, który symuluje działanie mózgu, podczas oceny jakości dźwięku. Na wyjściu modelu otrzymuje się zestaw parametrów wyjściowych MOV, z których wyliczany jest wskaźnik ODG.

2.3 Model psychoakustyczny

Zadaniem modelu psychoakustycznego jest przetworzenie ramki sygnału czasowego w taki sposób, jak przetworzony zostałby dźwięk przez ucho człowieka. Na wyjściu modelu otrzymujemy reprezentację sygnału, który pojawiłby się na błonie podstawnej ślimaka. Ramka sygnału wejściowego jest poddawana dyskretnej transformacie Fouriera (DFT), sygnał jest korygowany charakterystyką ucha, a następnie liniowa skala częstotliwości widma jest zamieniana na skalę psychoakustyczną.

Maskowanie równoczesne jest modelowane na dwa różne sposoby w zależności od wyliczanego parametru MOV. Pierwszym z nich jest wyliczenie progu maskowania z sygnału referencyjnego, a następnie porównanie różnicy między sygnałem referencyjnym i testowanym z tym progiem (Rys. 2.3). Drugim sposobem jest wyliczenie wewnętrznych reprezentacji sygnałów i porównaniu ich ze sobą. Wewnętrzna reprezentacja sygnału jest współczynnikiem określającym informację, która dociera do mózgu, w celu porównania sygnałów (Rys. 2.4).

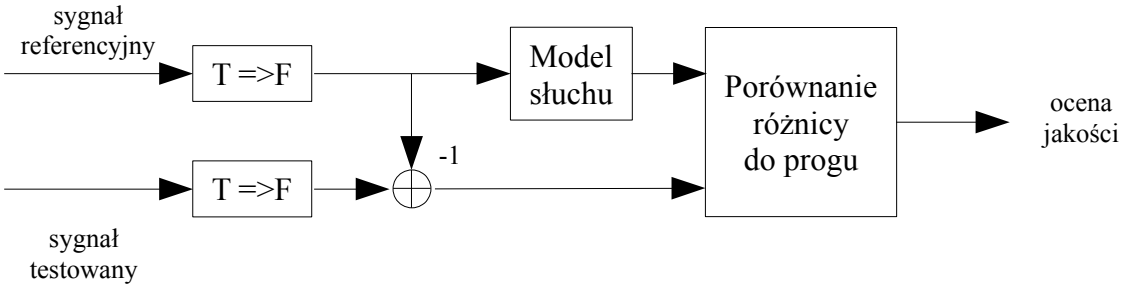
Próg maskowania odnosi się do progu słyszenia i jest wyliczany na podstawie zależnych od częstotliwości funkcji wagowej oraz pobudzenia. Maskowanie

nierównoczesne jest wykonywane przez wygładzanie sygnału w czasie.

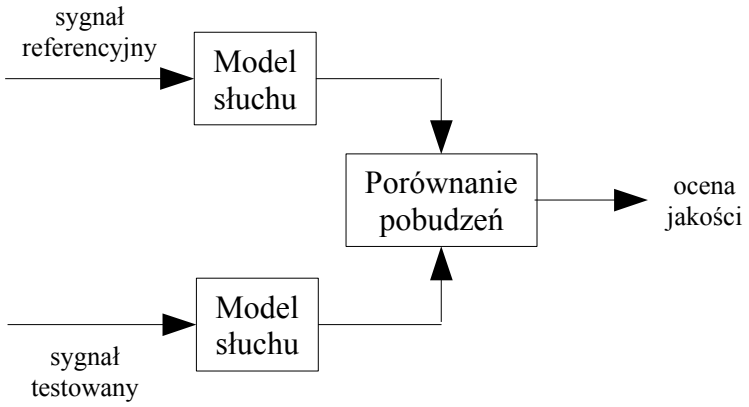
Głównym wyjściem modelu psychoakustycznego jest wzorzec pobudzenia oraz próg maskowania, jednak do niektórych obliczeń są wykorzystywane inne parametry wyliczane w poszczególnych etapach przetwarzania.

2.3.1 Charakterystyka ucha zewnętrznego i środkowego

Dźwięk docierający do słuchacza, jak opisano w podrozdziale 2.1.3, przechodzi najpierw przez ucho zewnętrzne i środkowe. Dźwięk, przechodząc przez te części ucha, ulega filtrowaniu pasmowo-przepustowemu. Do sygnału dodawany jest także szum spowodowany przepływem krwi. Amplituda tego szumu rośnie wraz ze zmniejszeniem częstotliwości. To właśnie charakterystyka ucha zewnętrznego i środkowego wraz z szumem ma największy wpływ na próg słyszalności. [3,4]



Rys. 2.3. Schemat wyliczania parametrów MOV na podstawie koncepcji progu maskowania. [1]



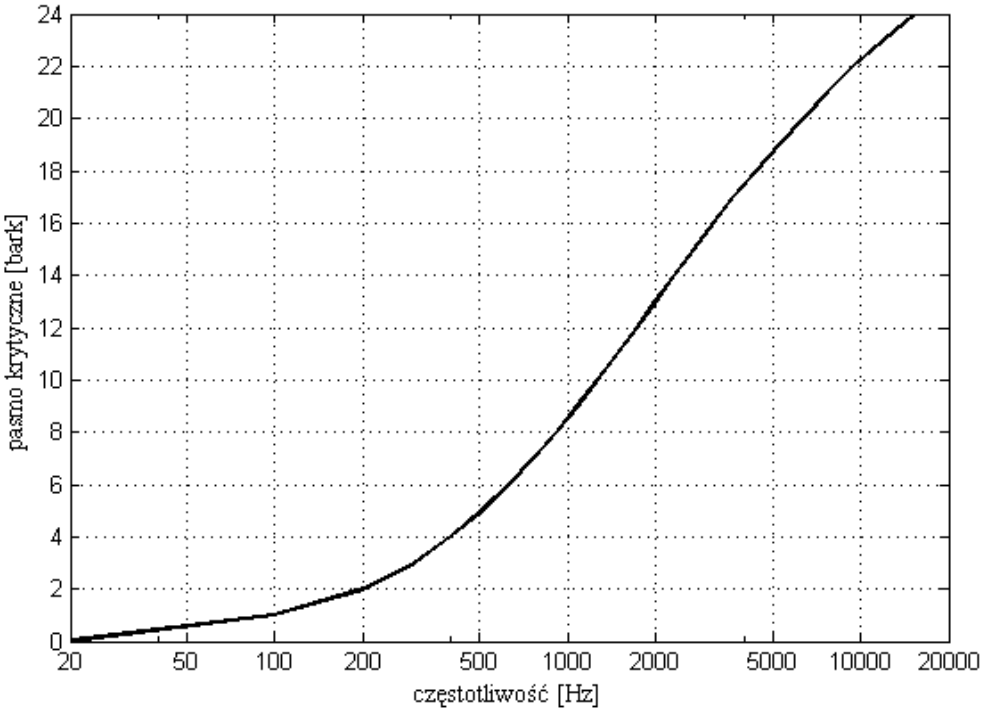
Rys. 2.4. Schemat wyliczania parametrów MOV na podstawie interpretacji wewnętrznej. [1]

2.3.2 Psychoakustyczna skala częstotliwości.

W podrozdziale 2.1.3 opisano sposób, w jaki sygnał pobudza błonę podstawną w ślimaku, oraz w jaki sposób jest odbierana informacja o poziomie ciśnienia akustycznego oraz częstotliwości dźwięku. Odbieranie wysokości dźwięku jest uzależnione od jego częstotliwości, jednak nie jest to zależność liniowa. Często przyjmuje się, że jest to zależność logarytmiczna, jednak nie jest to dostateczne przybliżenie w kontekście oceny jakości dźwięku. Istnieje kilka przybliżeń, jak na przykład: skala melowa, jednak w modelu przyjęto skalę Bark. Skala Bark dzieli słyszalny zakres częstotliwości na 24 niezachodzące na siebie pasma częstotliwości (Tab. 2.2). Rysunek 2.5 przedstawia zależność pomiędzy pasmem krytycznym w Barkach a częstotliwością. [3,5]

Tab. 2.2. Skala Bark. [5]

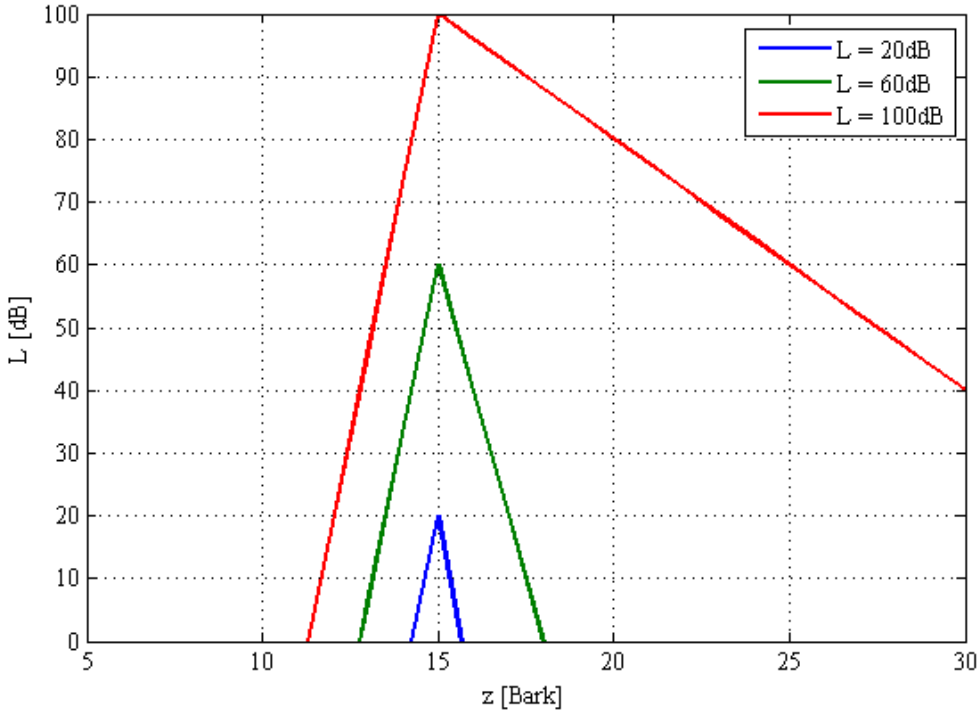
Bark	Częstotliwość środkowa	Częstotliwość odcięcia	Pasmo
		20	
1	50	100	80
2	150	200	100
3	250	300	100
4	350	400	100
5	450	510	110
6	570	630	120
7	700	770	140
8	840	920	150
9	1000	1080	160
10	1170	1270	190
11	1370	1480	210
12	1600	1720	240
13	1850	2000	280
14	2150	2320	320
15	2500	2700	380
16	2900	3150	450
17	3400	3700	550
18	4000	4400	700
19	4800	5300	900
20	5800	6400	1100
21	7000	7700	1300
22	8500	9500	1800
23	10500	12000	2500
24	13500	15500	3500



Rys. 2.5. Zależność pomiędzy pasmem krytycznym w Barkach a częstotliwością w Hz. [5]

2.3.3 Pobudzenie

Rzęski organu Cortiego reagują na pewne pasmo częstotliwości, które można określić przez charakterystykę filtru. Okazuje się, że przy wykorzystaniu skali Bark do określenia pasm częstotliwości, kształt charakterystyk jest praktycznie niezależny od częstotliwości środkowych filtrów. Nachylenie dolnej krzywej jest stałe i wynosi 27 dB/Bark. Nachylenie górnej krzywej jest natomiast zależne od poziomu sygnału L. Rysunek 2.6 przedstawia zaczerpniętą z rekomendacji [1] przykładową zależność nachylenia krzywych od poziomu sygnału. Dla sygnałów cichych nachylenie jest bardziej strome niż dla sygnałów głośnych. Jest to spowodowane mechanizmem sprzężenia zwrotnego, który występuje pomiędzy dwoma rodzajami rzęsek. Proces ten potrzebuje kilku milisekund do ustabilizowania się potencjału. Jednak pobudzenie spowodowane przez sygnał złożony nie jest funkcją liniową.



Rys. 2.6. Zależność kształtu krzywych pobudzenia od poziomu sygnału. [1]

Po zaniku sygnału rzęski potrzebują czasu na powrót do pełnej wrażliwości na bodźce. Proces ten trwa nawet kilkaset milisekund i zależy od poziomu oraz czasu trwania sygnału. Informacje o głośnych sygnałach są szybciej przekazywane do mózgu niż informacje o sygnałach cichych, dlatego sygnał o dużym poziomie może zamaskować poprzedzający go słaby sygnał. [3]

2.3.4 Próg detekcji

Człowiek posiada trzy typy pamięci: długotrwałą, krótkotrwałą oraz sensoryczną (ultrakrótką). W kontekście oceny jakości najważniejszą rolę odgrywa pamięć sensoryczna. Zapamiętywanie informacji o sygnale jest najlepsze dla fragmentów sygnałów trwających mniej niż 5-8 sekund zależnie od słuchacza i słuchanego materiału. Poziom różnicy sygnałów, dla którego możliwe jest wykrycie różnicy między nimi jest uzależniony od ich ogólnego poziomu. Poziom ten nazywamy progiem detekcji. Dla sygnałów cichych próg ten jest większy niż dla sygnałów cichych. Przykładowo dla sygnału o poziomie 20 dB_{SPL} próg wynosi 0,75 dB, natomiast dla 80 dB_{SPL} – wynosi 0,2 dB. Dla różnicy sygnałów równej progowi detekcji

prawdopodobieństwo usłyszenia różnicy wynosi 50 %, natomiast w pobliżu tego progu zmienia się od 0 do 100%. [1,3]

2.3.5 Maskowanie

Możemy wyróżnić dwa rodzaje maskowania: maskowanie równoczesne oraz maskowanie nierównoczesne.

Maskowanie równoczesne

Maskowanie równoczesne występuje, gdy dwa sygnały: sygnał maskujący oraz sygnał maskowany, są prezentowane jednocześnie. Jeżeli występuje sygnał maskujący, to próg słyszenia się podnosi. Poziom maskowania jest zależny od rodzaju zarówno sygnału maskującego, jak i maskowanego. Jeżeli ton maskujemy szumem, to poziom maskowania jest praktycznie niezmienny, a próg maskowania wynosi około 5 dB. W przypadku odwrotnym (maskowanie szumu tonem), poziom maskowania zależy od częstotliwości tonu w przybliżeniu:

$$M = 15.5 + \frac{z}{Bark} \quad , \tag{2.2}$$

gdzie:

M – próg maskowania,

z – numer pasma krytycznego w Barkach.

Ogólnie w przypadku złożonym, gdy sygnał maskujący składa się sygnałów dodanych nieliniowo, próg maskowania jest wyższy niż próg powodowany każdym z nich osobno. [3,4]

Maskowanie nierównoczesne

Maskowanie nierównoczesne występuje, gdy sygnały maskujący i maskowany nie występują jednocześnie. Jeżeli sygnał maskowany wystąpi zaraz po sygnale maskującym próg maskowania jest zbliżony do progu maskowania jednoczesnego i maleje, aż do progu słyszenia. Czas, w którym próg ten maleje, jest zależny od czasu trwania sygnału maskującego i może wynosić od 5 ms (impuls Gaussa

trwający 0,05 ms) do ponad 150 ms (szum różowy trwający 1 s). Maskowanie wystąpi również jeżeli tuż po (około 5 ms) cichym dźwięku wystąpi sygnał głośny. Jeżeli poziom sygnału maskowanego jest tuż nad progiem maskowania, nie będzie on słyszany jako osobny sygnał, lecz jako zmiana brzmienia sygnału maskującego. [3,4]

2.3.6 Głośność i maskowanie częściowe

Głośność dźwięku jest zależna nie tylko od jego poziomu ciśnienia, ale również od jego częstotliwości i czasu trwania. Ogólnie głośność dźwięku złożonego jest mniejsza niż suma głośności poszczególnych jego składowych. W przypadku oceny jakości głośność niechcianych zniekształceń jest zmniejszona przez częściowe ich maskowanie spowodowane sygnałem referencyjnym. [3,4]

2.4 Model poznawczy

Model poznawczy przetwarza informacje, które dostarcza model psychoakustyczny. Dla pomiaru jakości najważniejszymi informacjami są różnice między sygnałem referencyjnym i sygnałem testowanym zarówno w dziedzinie częstotliwości, jak również w dziedzinie wysokości dźwięku (skala psychoakustyczna). W dziedzinie częstotliwości tymi informacjami są: szerokość pasma obu sygnałów oraz struktura harmonicznych ich różnicy, natomiast w dziedzinie wysokości dźwięku są to różnice w wielkościach oraz modulacjach obwiedni pobudzeń.

Do końcowego wyznaczenia ODG algorytm wykorzystuje 11 parametrów MOV. Parametry te są odpowiednio ważone, aby wynik był jak najbardziej zbliżony do SDG. Wagi parametrów uzyskano wykorzystując algorytm propagacji wstecznej błędu służący do uczenia sztucznych sieci neuronowych.

2.5 Parametry wyjściowe

Zastosowana metoda obiektywnego pomiaru jakości dźwięku korzysta z 11 parametrów wyjściowych MOV. Parametry te reprezentują różnego rodzaju zniekształcenia które mogą wystąpić w testowanym sygnale. Tabela 2.3 przedstawia ogólny opis parametrów MOV.

Tab. 2.3. Ogólny opis parametrów wyjściowych MOV. [1]

MOV	Opis
WinModDiff1	Zmiany w modulacji (obwiedni) pobudzeń (parametry związane z chropowatością)
AvgModDiff1	
AvgModDiff2	
RmsNoiseLoud	Głośność zniekształceń
BandwidthRef	Zniekształcenia liniowe
BandwidthTest	
RelDistFrames	Częstość występowania zniekształceń
TotalNMR	Odstęp zakłóceń od maski
MFPD	Prawdopodobieństwo wykrycia zniekształceń
ADB	
EHS	Struktura harmonicznych sygnału błędu

- *WinModDiff1* – okienkowana średnia różnic w modulacjach pobudzeń,
- *AvgModDiff1* – średnia różnica modulacji pobudzeń,
- *AvgModDiff2* – średnia różnica modulacji pobudzeń z naciskiem na zmiany modulacji w przypadku, gdy modulacja sygnału referencyjnego jest niewielka,
- *RmsNoiseLoud* – wartość skuteczna średniej głośności zniekształceń,
- *BandwidthRef* – średnia szerokość pasma sygnału referencyjnego,
- *BandwidthTest* – średnia szerokość pasma sygnału testowanego,
- *RelDistFrames* – względna liczba ramek, w których co najmniej jedno pasmo jest słyszalnie zniekształcone,
- *TotalNMR* – logarytm średniego stosunku zniekształceń od maski (NMR),
- *MFPD* – maksymalne prawdopodobieństwo usłtszenia różnicy między sygnałami po filtracji dolnoprzepustowej,
- *ADB* – liczba zniekształconych ramek,
- *EHS* – struktura harmonicznych sygnału błędu zdefiniowana jako maksymalna wartość ich funkcji autokorelacji.

3 Wykonanie

3.1 Informacje wstępne

3.1.1 Środowisko programistyczne

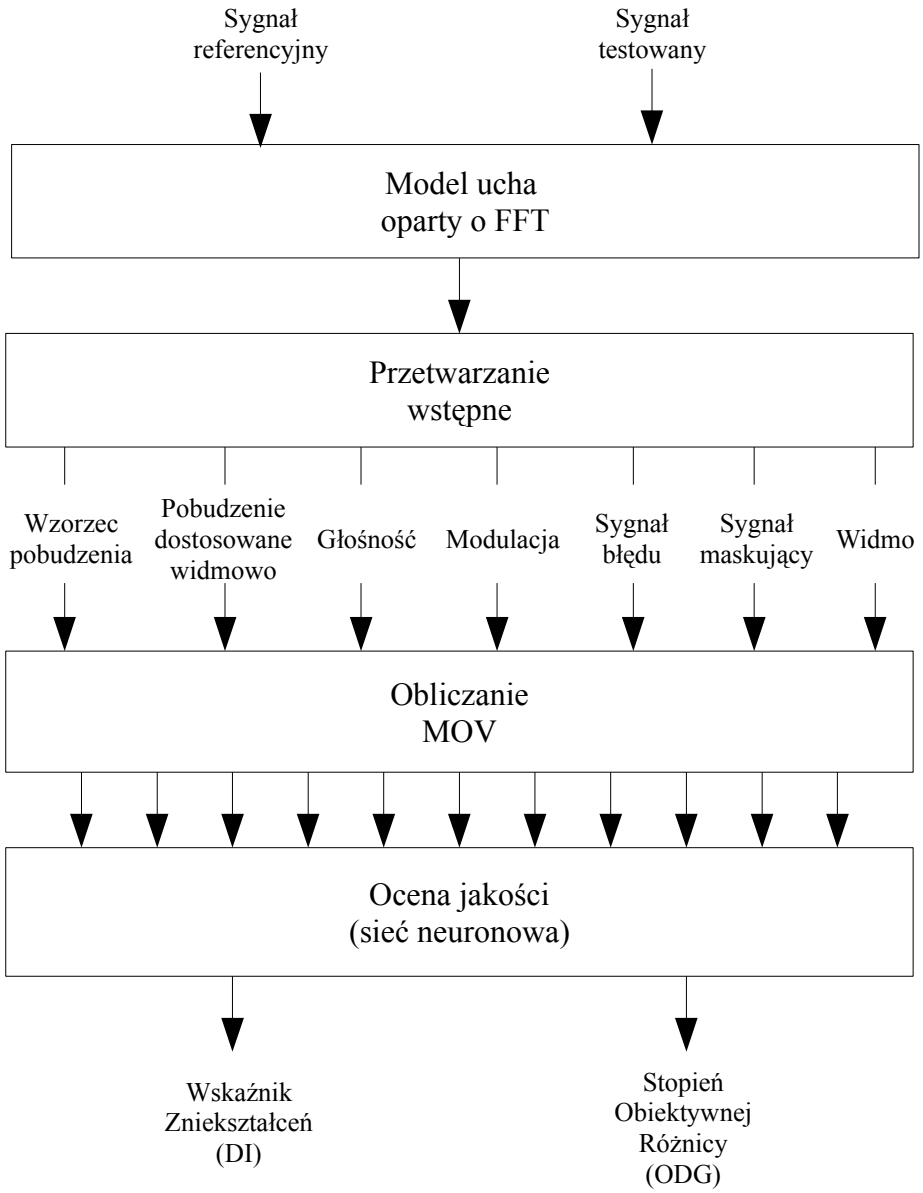
Do implementacji algorytmu wybrano pakiet do obliczeń inżynierskich Matlab. Pakiet ten posiada wiele udogodnień w przypadku pracy na wektorach czy sygnałach zespolonych, posiada również wiele wbudowanych funkcji, jak na przykład: szybką transformatę Fouriera. Zasadniczą różnicą w porównaniu do typowych języków programowania jest numeracja tablic rozpoczynająca się od „1” zamiast od „0”, co należało uwzględnić przy implementacji wzorów iteracyjnych podawanych przez rekomendację [1].

3.1.2 Etapy przetwarzania danych

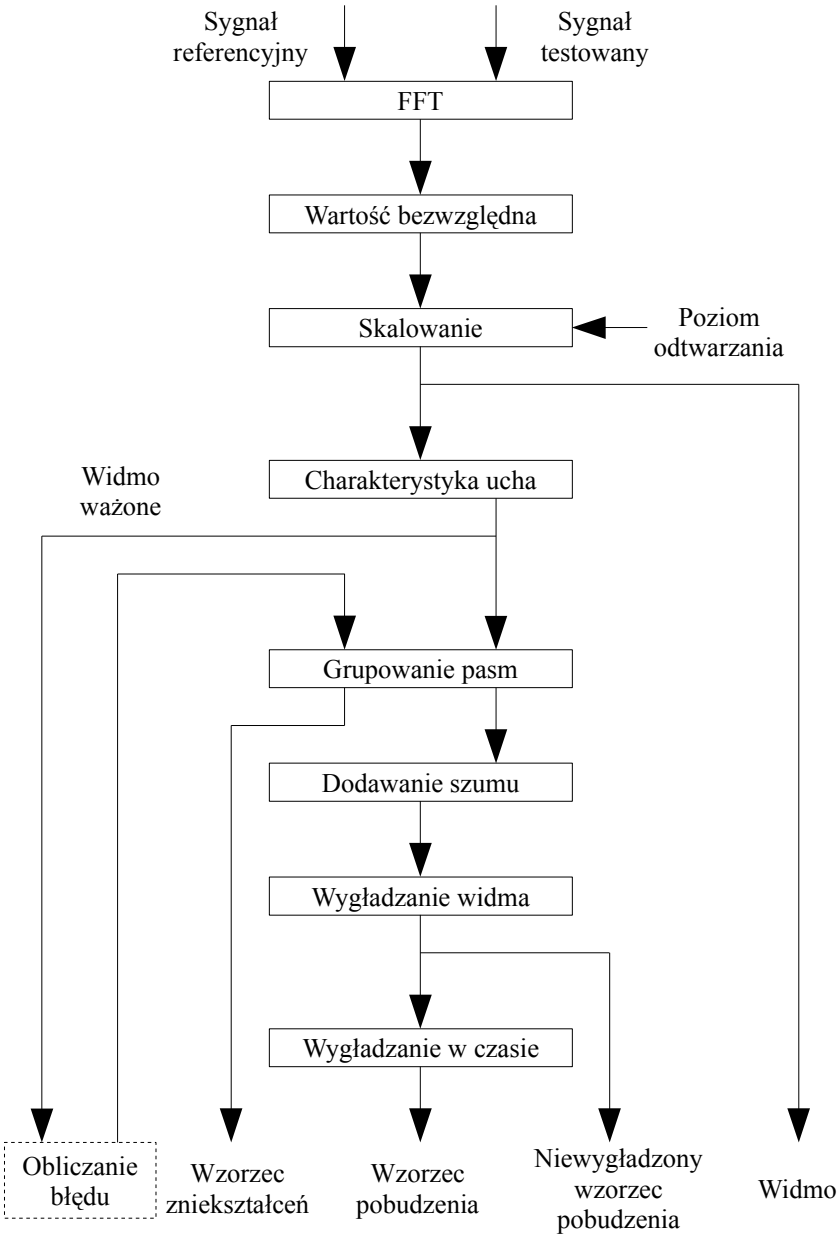
Na rysunku 3.1 przedstawiono ogólny schemat pomiarowy. Wejściem są dwa sygnały: sygnał referencyjny i sygnał testowy. Pomiar został podzielony na 4 etapy. Pierwszy to model ucha, z którego otrzymujemy wzorzec pobudzenia. Kolejnym jest wstępne przetwarzanie otrzymanego wzorca pobudzenia, z którego otrzymujemy wielkości koniecznych do wyliczenia parametrów wyjściowych MOV. Na podstawie wyliczonych wielkości MOV sieć neuronowa oblicza wskaźnik zniekształceń DI oraz stopień obiektywnej różnicy ODG.

3.1.2.1 Model ucha oparty o FFT

Model ucha oparty o FFT (Rys. 3.2) przyjmuje jako dane wejściowe dwa sygnały: sygnał referencyjny i sygnał testowany, oraz poziom odtwarzania jako parametr przy skalowaniu widma. Głównym wyjściem modelu jest pobudzenie, jednak do dalszych obliczeń konieczne są także: pobudzenie niewygładzone, widmo częstotliwościowe, widmo częstotliwościowe ważone oraz sygnał błędu. Implementacja programowa modelu jest przedstawiona w dalszej części niniejszej pracy.



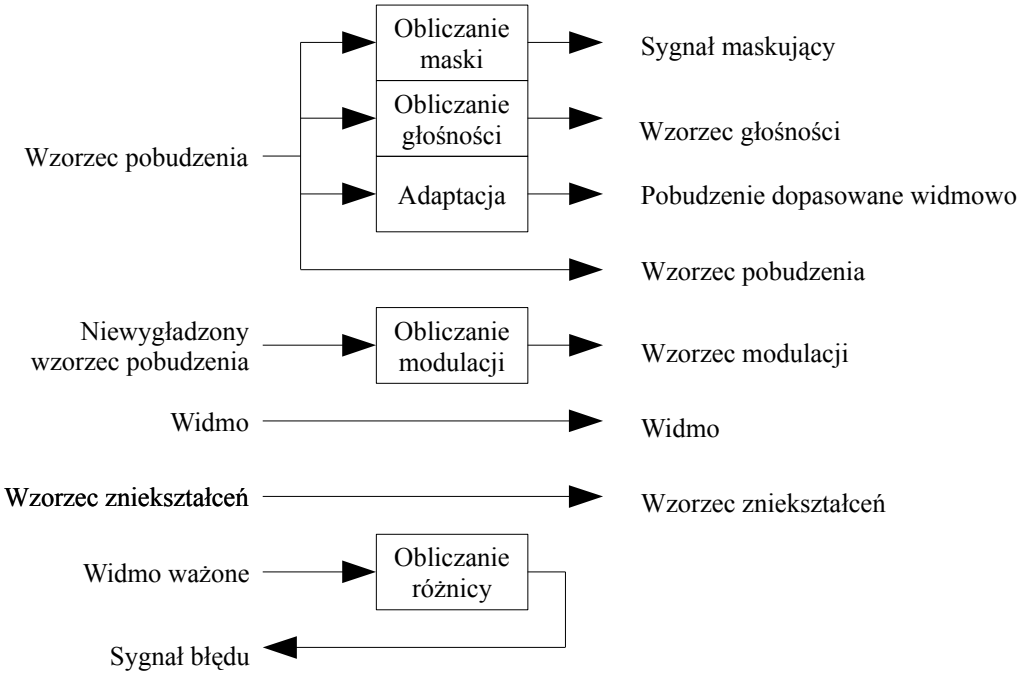
Rys. 3.1. Schemat pomiarowy. [1 s.32]



Rys. 3.2. Model ucha oparty o FFT. [1 s.34]

3.1.2.2 Przetwarzanie wstępne

Przetwarzanie wstępne służy do przygotowania danych do obliczenia parametrów MOV. Schemat przetwarzania przedstawiono na rysunku 3.3.



Rys. 3.3. Schemat przetwarzania wstępnego. [1 s.34]

3.1.2.3 Obliczanie parametrów wyjściowych MOV

Na tym etapie następuje zamiana wyliczonych do tej pory danych na współczynniki MOV dla każdej ramki sygnału. Dane są następnie odpowiednio uśredniane zarówno w dziedzinie częstotliwości, czasu, jak i ilości kanałów audio. W efekcie uzyskujemy 11 współczynników liczbowych.

3.1.2.4 Ocena jakości (sztuczna sieć neuronowa)

Ostateczna ocena jakości jest wyliczana na podstawie współczynników MOV za pomocą sztucznej sieci neuronowej o jednej ukrytej warstwie. Po zastosowaniu sieci otrzymujemy wskaźnik zniekształceń DI, który następnie jest przeliczany na stopień obiektywnej różnicy ODG.

3.1.3 Zarys programu

Zasadnicza część programu wykonywana jest w dwóch pętlach: jednej określającej numer ramki, drugiej – kanał analizowanego sygnału. Ogólny schemat pętli programowej przedstawiono poniżej:

```
% n – numer aktualnej ramki
% c – numer aktualnego kanału
% N – ilość ramek
% C – ilość kanałów (1 lub 2)

For n=1:N-1

   ramkowanie

   for c=1:C

     Model ucha
     Przetwarzanie wstępne
     Parametry MOV

   end
end

Uśrednianie MOV
Obliczenia DI oraz ODG
```

Pełny kod źródłowy programu znajduje się w załączniku nr 5.

Do programu przygotowano także prosty interfejs graficzny, przydatny w przypadku konieczności szybkiego porównania dwóch plików. Dokonano także kompilacji programu pakietem Matlab Compiler, aby możliwe było użycie programu bez konieczności uruchamiania pakietu Matlab.

3.2 Model ucha

Wejście modelu stanowią sygnały: referencyjny oraz testowany, próbkowane z częstotliwością 48 kHz i wyrównane względem siebie w czasie. Aby wyniki nie były przekłamane, sygnały te muszą sobie odpowiadać z dokładnością do 24 próbek. Sygnały te są cięte na ramki o długości 0.042 s (2048 próbek) z zakładką równą 50 %. Każda ramka jest okienkowana oknem Hanna, następnie wykonywana jest krótko - czasowa transformata Fouriera oraz skalowanie do poziomu odtwarzania. Widmo jest ważone charakterystyką ucha zewnętrznego i środkowego. Następnie grupuje się częstotliwości w odpowiednie pasma krytyczne, dodaje szum oraz wygładza w dziedzinie częstotliwości i dziedzinie czasu. Na koniec wyliczany jest próg maskowania.

3.2.1 Ramkowanie

Sygnał wejściowy jest cięty na ramki o długości 2048 próbek z zakładką równą 50 %. Rekomendacja [1] podaje następujący wzór:

$$t_n[k_t, n] = t[1024 \cdot n + k_t] \quad , \quad n = 0,1,2... \quad , \quad k_t = 0..2047 \tag{3.1}$$

gdzie:

t_n – ramka sygnału,

t – sygnał wejściowy,

n – numer ramki,

k_t – numer próbki w ramce.

Implementacja:

```
tn = t(:,1024*(n-1)+1 : 1024*(n+1));
```

3.2.2 FFT

Przechodząc do dziedziny częstotliwości sygnał najpierw jest okienkowany oknem Hanna:

$$h_w[k]=\frac{1}{2}\sqrt{\frac{8}{3}}\left[1-\cos\left(2\pi\frac{k}{N-1}\right)\right] \quad , \quad N=2048 \tag{3.2}$$

$$t_w[k_t,n]=h_w[k_t]\cdot t_n[k_t,n] \tag{3.3}$$

gdzie:

h_w – współczynniki okna,

t_w – sygnał po okienkowaniu,

N – długość ramki.

Następnie sygnał jest poddawany jest transformacie Fouriera:

$$F_f[k_f,n]=\frac{1}{2048}\sum_{k_t=0}^{2047} t_w[k_t,n]e^{-j\frac{2\pi}{2048}k_f k_t} \tag{3.4}$$

gdzie:

F_f – widmo ramki sygnału.

Na koniec wykonywane jest skalowania do poziomu odtwarzania *Lp* dla sygnału sinusoidalnego o pełnej skali (full scale).

$$F[k_f,n]=fac\cdot F_f[k_f,n] \tag{3.5}$$

$$fac=\frac{10^{\frac{Lp}{20}}}{Norm} \tag{3.6}$$

gdzie:

F – widmo wyskalowane,

fac – współczynnik normalizacji,

Norm – współczynnik normalizacji.

Współczynnik normalizacji Norm wyliczany jest w następujący sposób: 10 ramek przebiegu sinusoidalnego o częstotliwości 1019.5 Hz i poziomie 0 dB_{FS} jest okienkowany i poddawany transformacie Fouriera, a następnie jest wyliczana wartość bezwzględna z widma.

Jeżeli poziom dźwięku jest nieznany, zaleca się ustawienie go na poziom $Lp = 92$ dB.

Implementacja:

Funkcja – `FFTnorm`

Aby ograniczyć ilość wykonywanych mnożeń, skorzystano z liniowości transformaty Fouriera i współczynnik skalujący *fac* zintegrowano z oknem Hanna. Korzystając z tej samej własności pominięto pierwiastek w funkcji okna.

```
fac = 10^(Lp/20)/Norm;                                  %(3.6)
hw = fac.*hann(1024)';                                  %(3.2)
```

Współczynnik Norm wyliczono w następujący sposób:

```
fc = 1019.5;                    % częstotliwość sygnału normalizującego
fs = 48000;                     % częstotliwość próbkowania
t = 0:1024*9;                   % wektor czasu
s = sin(2*pi*fc*t/fs);          % sygnał normalizujący
hw = hann(1024);                % wektor okna Hanna
Norm = 0;
for i = 1:10
    S = abs(fft(s(1024*(i-1)+1:1024*(i+1)).*hw'));
    Norm = max(max(S), Norm);       % współczynnik skalujący
end
```

Obliczenia te są przeprowadzane tylko raz przy pierwszym wywołaniu funkcji. Wektor normalizującego okna Hanna jest przechowywany w pamięci jako zmienna `persistent`. Sygnał wejściowy jest okienkowany, wyliczana jest jego transformata Fouriera oraz jej moduł.

```
F = abs(fft(hw.*tn));                                   %(3.4)
```

3.2.3 Ucho zewnętrzne i środkowe

Charakterystykę częstotliwościową ucha zewnętrznego i ucha środkowego przedstawia funkcja (3.7):

$$\frac{W[k]}{dB} = -0.6 \cdot 3.64 \cdot \left(\frac{f[k]}{kHz}\right)^{-0.8} + 0.65 \cdot e^{-0.6 \cdot \left(\frac{f[k]}{kHz} - 3.3\right)^2} - 10^{-3} \cdot \left(\frac{f[k]}{kHz}\right)^{3.6} \tag{3.7}$$

$$\frac{f[k]}{kHz} = k \cdot 23.475 \cdot 10^{-3} \tag{3.8}$$

gdzie:

W – charakterystyka częstotliwościowa ucha,

f – kolejne częstotliwości FFT.

Nałożenie charakterystyki na widmo sygnału dokonujemy według wzoru (3.9).

$$F_e[k_f, n] = \left|F[k_f, n]\right| \cdot 10^{\frac{W[k_f]}{20}} \tag{3.9}$$

gdzie:

F_e – widmo ważone,

F – widmo wyskalowane (3.5).

Implementacja:

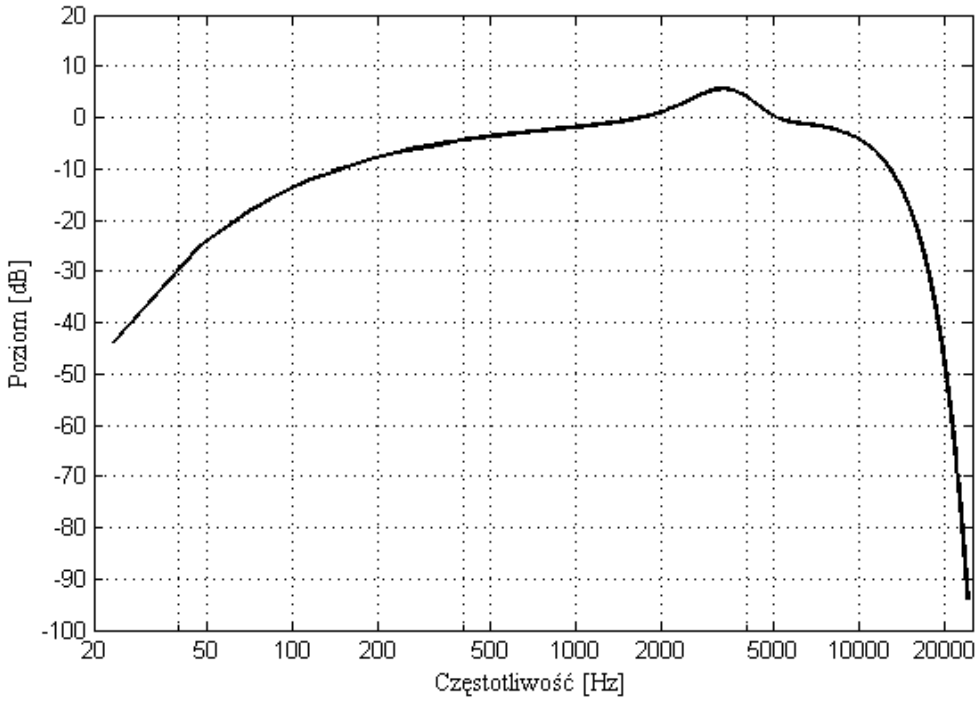
Funkcja – `OMEar`

Charakterystyka ucha jest wyliczana tylko raz i przechowywana w pamięci jako zmienna `persistent`. Ponieważ moduł widma był wyliczony w poprzednim etapie teraz wystarczy przemnożyć widmo przez charakterystykę ucha:

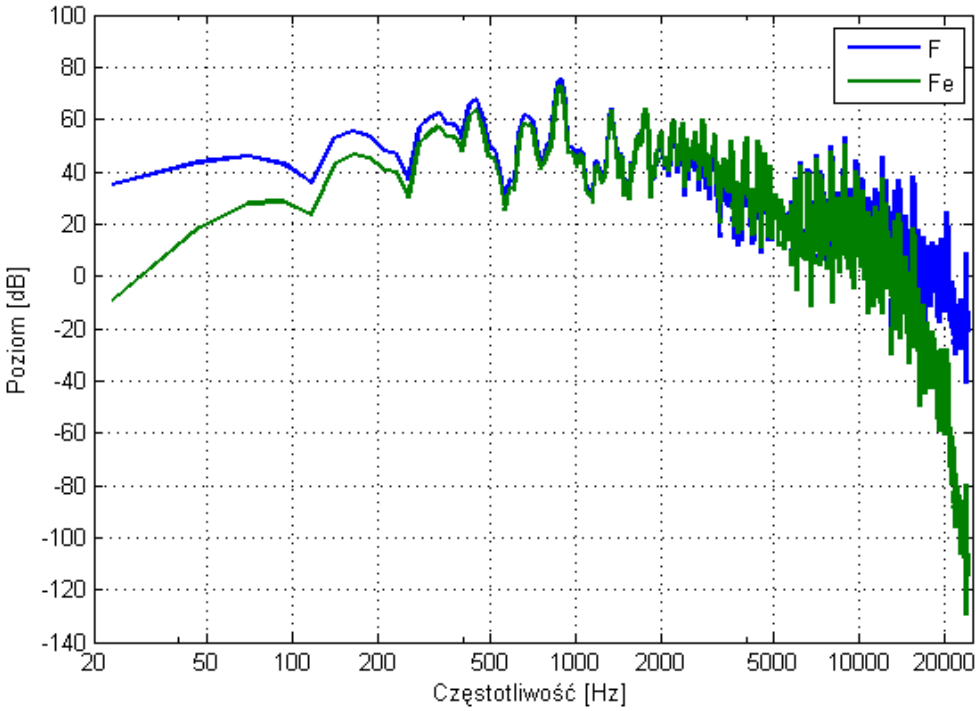
```
% fs częstotliwość próbkowania 48000

f = linspace(0,fs/2,1024)/1000;                    %(3.8)
W = -0.6*3.64*f.^(-0.8) + …
   6.5*exp(-0.6*(f-3.3).^2)-0.001*f.^(3.6);        %(3.7)
W = 10.^(W/20);
Fe = W.*F;                                         %(3.9)
```

Wpływ charakterystyki ucha (Rys. 3.4) na sygnał przedstawia rysunek 3.5.



Rys. 3.4. Charakterystyka częstotliwościowa ucha zewnętrznego i środkowego



Rys. 3.5. Wpływ charakterystyki ucha na widmo sygnału. F – widmo sygnału, Fe – widmo ważone charakterystyką ucha.

3.2.4 Grupowanie

Pasma krytyczne są wyliczane na podstawie przybliżenia:

$$\frac{z}{Bark} = 7 \cdot \operatorname{arsinh}\left(\frac{f/Hz}{650}\right) \tag{3.10}$$

gdzie:

z – reprezentacja częstotliwości w skali psychoakustycznej,

Bark – oznacza jednostkę skali.

Widmo, wyliczone przez 2048-punktową FFT, grupuje się do 109 pasm krytycznych. Pasma te nie odpowiadają dokładnie zależności ze wzoru (3.10). Dokładne tabele częstotliwości zaczerpnięte z rekomendacji [1] znajdują się w załączniku nr 1. Grupowanie odbywa się poprzez odpowiednie sumowanie energii dla określonych częstotliwości lub energii sygnału błędu.

$$Fsp[k_f, n] = \left|F_e[k_f, n]\right|^2 \tag{3.11}$$

$$Fsp[k_f, n] = \left|F_{noise}[k_f, n]\right|^2 \tag{3.12}$$

gdzie:

Fsp – reprezentacja energii sygnału,

F_e – widmo ważone (3.9),

F_{noise} – sygnał błędu (3.42).

Sposób grupowania rekomendacja przestawia w formie pseudokodu, który jest zamieszczony w załączniku nr 2. W efekcie dostajemy reprezentację energii sygnału w pasmach krytycznych oznaczoną dalej jako P_e.

Implementacja:

Funkcja – `CBGroup`

3.2.5 Szum wewnętrzny

Do odpowiednich pasm krytycznych dodawany jest szum zależny od częstotliwości.

$$P_{Thres}[k] = 10^{0.4 \cdot 0.364 \cdot \left(\frac{f_c[k]}{kHz}\right)^{-0.8}} \tag{3.13}$$

$$P_p[k,n] = P_e[k,n] + P_{Thres}[k] \tag{3.14}$$

gdzie:

Pthres – szum zależny od częstotliwości,

Pp – reprezentacja wysokości dźwięku,

fc – częstotliwość środkowa danego pasma krytycznego,

k – numer pasma krytycznego.

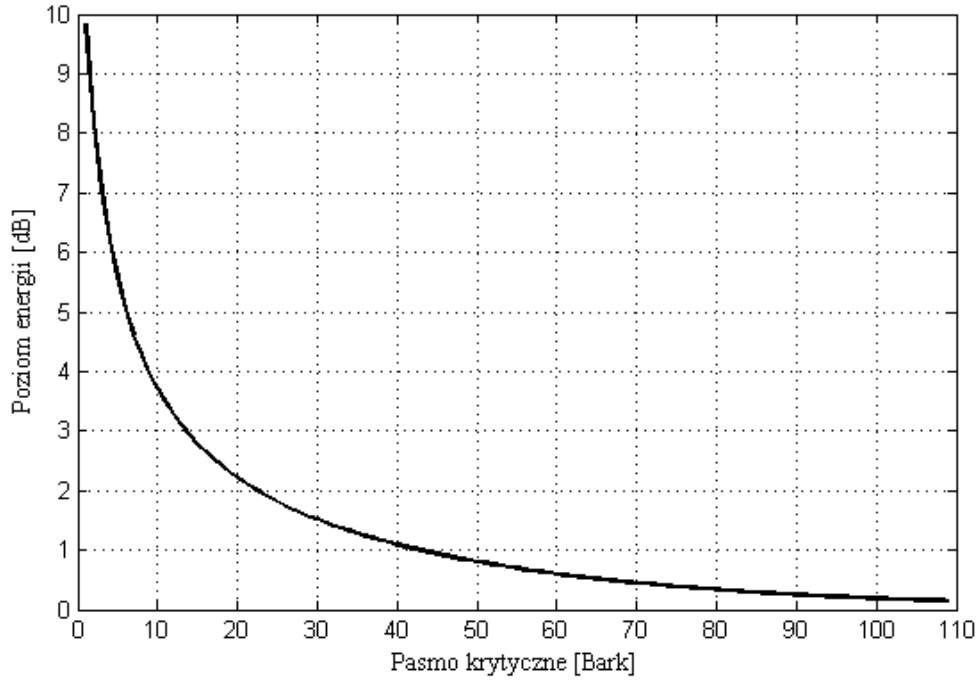
Implementacja:

Funkcja – `AddNoise`

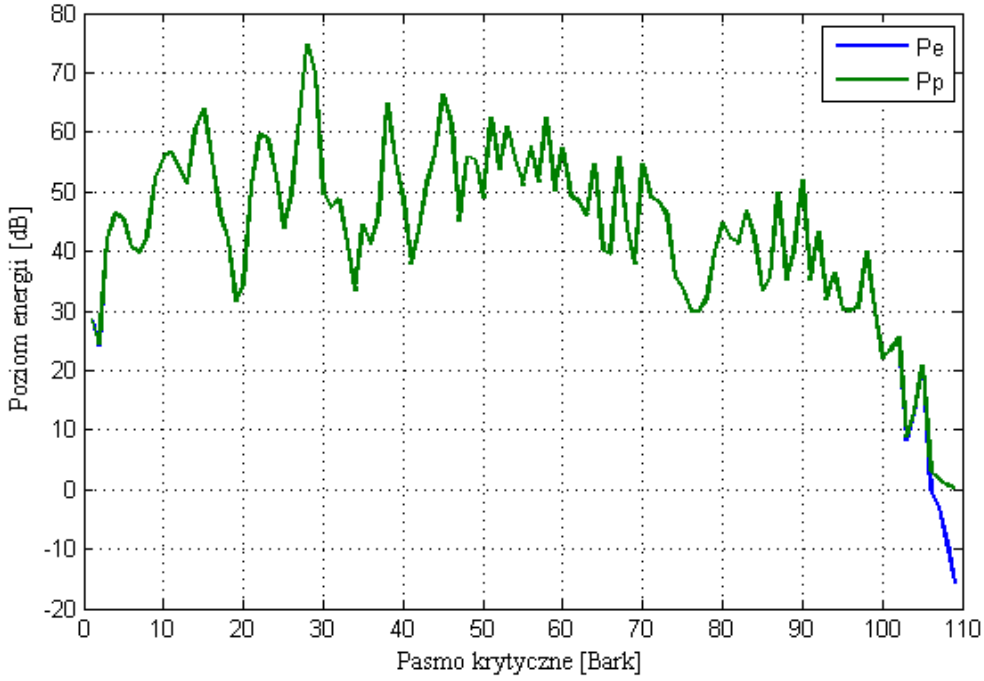
Implementacja jest bezpośrednim zastosowaniem wzorów z rekomendacji [1]. Również w tym przypadku parametr P_{Thres} jest wyliczany raz i przechowywany w pamięci.

```
Pthres = 10.^((0.4*0.364*(fc./1000).^(-0.8)));        %(3.13)
Pp = Pe + Pthres;                                     %(3.14)
```

Na rysunku 3.6 przedstawiono zależną od częstotliwości funkcję szumu wewnętrznego natomiast na rysunku 3.7 wpływ szumu wewnętrznego na ramkę sygnału.



Rys. 3.6. Zależność poziomu szumu od pasma krytycznego.



Rys. 3.7. Widmo w pasmach krytycznych. Pe – widmo, Fp – widmo z dodanym szumem wewnętrznym.

3.2.6 Wygładzanie w dziedzinie częstotliwości

Zachodzenie na siebie pasm krytycznych jest uzależnione od wielkości pobudzenia. Dolna krzywa wygładzania ma zawsze nachylenie równe 27 dB/Bark, natomiast nachylenie górnej uzależnione jest od częstotliwości oraz poziomu pobudzenia.

$$\frac{S_u[k, L[k,n]]}{dB/Bark} = -24 - \frac{230\text{Hz}}{f_c[k]} + 0.2 \cdot \frac{L[k,n]}{dB} \tag{3.15}$$

$$S_l[k, L[k,n]] = 27 \frac{dB}{Bark} \tag{3.16}$$

$$L[k,n] = 10 \cdot \log_{10}(P_p[k,n]) \tag{3.17}$$

gdzie:

L – poziom pobudzenia wyrażony w decybelach,

P_p – reprezentacja wysokości dźwięku (3.14),

S_u – górna krzywa wygładzania,

S_l – dolna krzywa wygładzania.

Funkcja wygładzania przyjmuje następującą postać:

$$E_2[k,n] = \frac{1}{Norm_{SP}[k]} \left(\sum_{j=0}^{Z-1} E_{line}[j,k,n]^{0.4} \right)^{\frac{1}{0.4}} , \tag{3.18}$$

gdzie:

E_2 – niewygładzony wzorzec pobudzenia.

Ze względu na duży stopień skomplikowania wzorów koniecznych do wyliczenia parametrów E_{line} oraz $Norm_{SP}$, wzory te zamieszczono w załączniku nr 2.

Implementacja:

Funkcja – `Spread`

W funkcji tej dokonano kilku przekształceń w stosunku do wzorów podanych

w rekomendacji. Ogólny sposób przekształceń również znajduje się w załączniku nr 2.

Efekt wygładzania częstotliwościowego można zaobserwować na rysunku 3.8.

3.2.7 Wygładzanie w dziedzinie czasu

Maskowanie niejednoczesne jest wykonywane przez zastosowanie filtru dolnoprzepustowego pierwszego rzędu. Stałe czasowe są wyliczane dla każdego pasma krytycznego.

$$\tau = \tau_{min} + \frac{100\text{Hz}}{f_c[k]} \cdot (\tau_{100} - \tau_{min}) \quad , \quad \begin{matrix} \tau_{100} = 0.030\text{s} \\ \tau_{min} = 0.008\text{s} \end{matrix} \tag{3.19}$$

gdzie:

τ – wektor stałych czasowych.

Filtrowanie wykonywane jest następująco:

$$E_f[k,n] = a \cdot E_f[k,n-1] + (1-a) \cdot E_2[k,n] \tag{3.20}$$

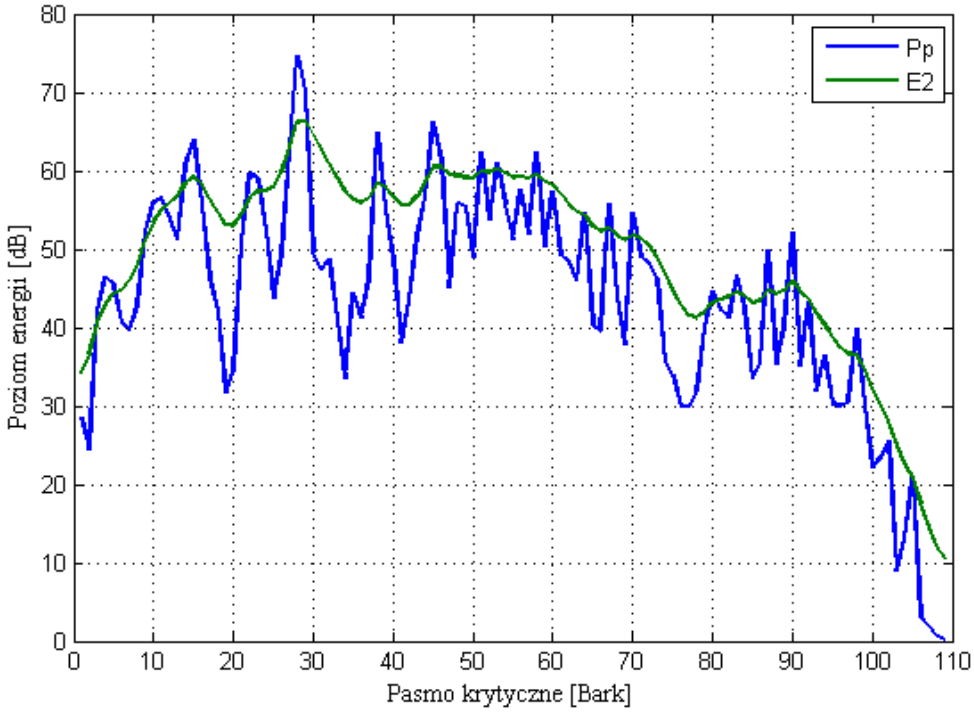
$$E[k,n] = max(E_f[k,n], E_2[k,n]) \tag{3.21}$$

gdzie:

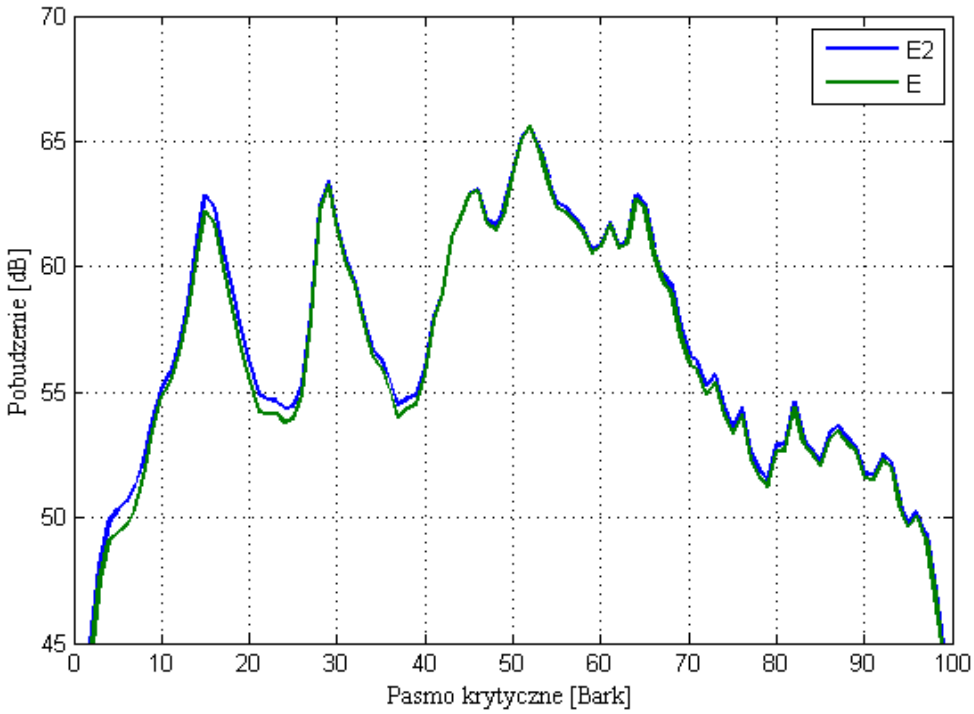
E – pobudzenie,

E_2 – pobudzenie (niewygładzone czasowo) (3.18).

$$a = e^{-\frac{4}{187.5} \cdot \frac{1}{\tau}} \tag{3.22}$$



Rys. 3.8. Efekt wygładzania w dziedzinie częstotliwości. Pp – widmo niewygładzone, E2 – widmo po wygładzeniu.



Rys. 3.9.Efekt wygładzania czasowego. E2 – widmo niewygładzone, E – widmo wygładzone.

Implementacja:

Funkcja – `TimeSpread`

Współczynniki *a* są wyliczane tylko raz i przechowywane w pamięci. W celu wykonania filtrowania w głównej części programu została stworzona zmienna do przechowywania danych z poprzedniej ramki sygnału. Zmienna ta jest przekazywana do funkcji jako parametr. Funkcja wykonuje obliczenia i zwraca nową wartość zmiennej pamiętającej, która znów zostanie przekazana do tej funkcji przy następnej iteracji.

```
[E, Ef] = TimeSpread(E2, Ef)                    % wywołanie funkcji

t = tmin + 100./fc*(t100-tmin);                 %(3.19)
a = exp(-4/187.5./t);                           %(3.22)

Ef = a.*Ef + (1-a).*E2;                         %(3.20)
E = max(Ef,E2);                                 %(3.21)
```

3.2.8 Próg maskowania

Maskowanie opisuje efekt, jaki zachodzi podczas, gdy cichy, ale wyraźny sygnał staje się niesłyszalny z powodu pojawienia się w jego sąsiedztwie sygnału głośniejszego. Próg maskowania wyliczamy według zależności:

$$M[k,n]=\frac{E[k,n]}{10^{\frac{m[k]}{10}}} \tag{3.23}$$

$$m[k]=\begin{cases} 3 & dla\ k\leqslant 48 \\ 0.25\cdot k\cdot res & dla\ k>48 \end{cases} \tag{3.24}$$

gdzie:

m – funkcja wagowa progu maskowania,

M – próg maskowania.

Implementacja:

Funkcja – `Mthres`

Implementacja jest wprost zastosowaniem powyższych wzorów. Funkcja wagowa *m* jest wyliczana raz i przechowywana w pamięci.

```
M(1:49) = 10^(3/10);                              %(3.24)
m(50:109) = 10.^(0.25.*0.25.*(49:108)./10);       %(3.24)

Ma = E./m;                                        %(3.23)
```

3.3 Przetwarzanie wstępne

3.3.1 Adaptacja poziomu i wzorca

Aby skompensować różnice poziomów i zniekształcenia liniowe pomiędzy sygnałem referencyjnym i sygnałem testowanym, dopasowano ich średnie poziomy do siebie.

Pierwszym etapem jest wygładzenie sygnałów filtrem dolnoprzepustowym pierwszego rzędu. Stałe czasowe filtrów wyliczamy na podstawie wzoru:

$$\tau = \tau_{min} + \frac{100\text{Hz}}{f_c[k]} \cdot (\tau_{100} - \tau_{min}) \quad , \quad \begin{matrix} \tau_{100} = 0.050\text{s} \\ \tau_{min} = 0.008\text{s} \end{matrix} \tag{3.25}$$

Dokonujemy filtrowania sygnałów:

$$P[k,n] = a \cdot P[k,n-1] + (1-a) \cdot E[k,n] \tag{3.26}$$

gdzie:

P – wzorzec,

E – wzorzec pobudzenia,

a – współczynniki wyliczane według wzoru (3.22).

Dla wyliczonych wzorców sygnałów: referencyjnego i testowanego, wyznacza się współczynnik korekcji poziomu:

, (3.27)

gdzie:

P_{Test} – wzorzec sygnału testowego,

P_{Ref} – wzorzec sygnału referencyjnego,

LevCorr – współczynnik korekcji.

Jeżeli współczynnik korekcji poziomu jest większy od 1, to sygnał referencyjny jest dzielony przez współczynnik korekcji, w przeciwnym wypadku sygnał testowy jest przez niego mnożony.

$$E_{L,Ref}[k,n]=E_{Ref}[k,n]/LevCorr[n] \qquad LevCorr[n]>1 \tag{3.28}$$

$$E_{L,Test}[k,n]=E_{Test}[k,n]\cdot LevCorr[n] \qquad LevCorr[n]\leqslant 1 \tag{3.29}$$

Współczynniki korekcji wzorca dla każdego z sygnałów są wyliczane przez porównanie obwiedni czasowych sygnałów.

$$R[k,n]=\frac{\sum_{i=0}^{n} a[k]^i\cdot E_{L,Test}[k,n-i]\cdot E_{L,Ref}[k,n-i]}{\sum_{i=0}^{n} a[k]^i\cdot E_{L,Ref}[k,n-i]\cdot E_{L,Ref}[k,n-i]} \tag{3.30}$$

Współczynniki *a* przyjęto takie same, jak wyliczone powyżej (3.22). W zależności od tego, jakie wartości przyjmuje współczynnik *R[k,n],* współczynniki korekcji sygnałów przyjmują następujące wartości:

$$\begin{array}{lll} R_{Test}[k,n]=\dfrac{1}{R[k,n]} & R_{Ref}[k,n]=1 & R[k,n]\geqslant 1 \\ R_{Test}[k,n]=1 & R_{Ref}[k,n]=R[k,n] & R[k,n]<1 \end{array} \tag{3.31}$$

Jeżeli mianownik równania 3.30 jest równy 0, a licznik większy od zera, to: $R_{Test}[k,n] = 0$ i $R_{Ref}[k,n] = 1$. Jeżeli licznik również jest równy 0, wartości $R_{test}[k,n]$ i $R_{ref}[k,n]$ przyjmujemy takie same, jak dla poprzedniej ramki sygnału. Jeżeli nie ma ramki poprzedniej (pierwsza ramka), współczynniki $R_{Test}[k,n]$ i $R_{ref}[k,n]$ ustawiamy na 1.

Współczynniki korekcji są uśredniane dla 8 sąsiadujących ze sobą pasm krytycznych, a następnie wygładzane w czasie z taką samą stałą czasową, jak w poprzednim przypadku.

$$\begin{aligned} PattCorr_{Test}[k,n] &= a \cdot PattCorr_{Test}[k,n-1] + (1-a) \cdot \frac{1}{M} \cdot \sum_{i=-M_1}^{M_2} R_{Test}[k+i,n] \\ PattCorr_{Ref}[k,n] &= a \cdot PattCorr_{Ref}[k,n-1] + (1-a) \cdot \frac{1}{M} \cdot \sum_{i=-M_1}^{M_2} R_{Ref}[k+i,n] \end{aligned} \tag{3.32}$$

gdzie:

M – szerokość okna częstotliwości w naszym przypadku równa 8,

M_1, M_2 – granice okna częstotliwości.

Na granicach szerokość okna ulega zmniejszeniu.

$$M_1 = min(M_1, k), \qquad M_2 = min(M_2, z-k-1), \qquad M = M_1 + M_2 + 1 \tag{3.33}$$

Pobudzenie dopasowane widmowo otrzymujemy przez przemnożenie pobudzenia dopasowanego poziomem przez współczynnik korekcji wzorca.

$$\begin{aligned} E_{P,Ref}[k,n] &= E_{L,Ref} \cdot PattCorr_{Ref}[k,n] \\ E_{P,Test}[k,n] &= E_{L,Test} \cdot PattCorr_{Test}[k,n] \end{aligned} \tag{3.34}$$

Implementacja:

Funkcja – `LPAdapt`

Mimo pozornej złożoności obliczeń implementacja w pakiecie Matlab jest dość prosta. Bardziej rozbudowane jest jedynie sprawdzanie warunków. Tak,jak w przypadku wygładzania w czasie wzorca pobudzenia, również tutaj wartości z poprzednich ramek są przekazywane do funkcji jako parametry. Dane są przechowywane w macierzach,

w których pierwszy wiersz jest sygnałem referencyjnym, drugi – sygnałem testowanym. Dla przykładu: `P(1,:)` oznacza wzorzec dla sygnału referencyjnego.

```
P(1,:) = a.*P(1,:) + (1-a).*E(1,:);                %(3.26)
P(2,:) = a.*P(2,:) + (1-a).*E(2,:);                %(3.26)

LevCorr = (sum(sqrt(P(2,:) .* P(1,:))) ./ sum(P(2,:)))^2;  %(3.36)

% sprawdzanie warunków i wpisywanie odpowiednich wartości

Rn = a.*Rn + EP(2,:).*EP(1,:);                     %(3.30) licznik
Rd = a.*Rd + EP(1,:).*EP(1,:);                     %(3.30) mianownik

% sprawdzanie warunków i wpisywanie odpowiednich wartości


for i = 1:z
    m1 = max(i - M1, 1);                           %(3.33)
    m2 = min(i + M2, z);                           %(3.33)
    s1 = sum(R(1,m1:m2));                          %(3.32) sumowanie
    s2 = sum(R(2,m1:m2));                          %(3.32) sumowanie
    PC(1,i) = a(i) * PC(1,i) + (1-a(i)) * s1 / (m2-m1+1);  %(3.32)
    PC(2,i) = a(i) * PC(2,i) + (1-a(i)) * s2 / (m2-m1+1);  %(3.32)
end

EP = EP.*PC;                                       %(3.34)
```

3.3.2 Modulacja

Z niewygładzonego wzorca pobudzenia wyznaczono przybliżoną wartość głośności przez podniesienie do potęgi 0.3, wyznaczono pochodną czasową i wygładzono w czasie, filtrując w taki sam sposób, jak w poprzednich przypadkach.

$$E_{der}[k,n]=a\cdot E_{der}[k,n-1]+(1-a)\cdot\frac{48000}{1024}\cdot\left|E_2[k,n]^{0.3}-E_2[k,n-1]^{0.3}\right| \quad (3.35)$$

gdzie:

E_2 – niewygładzony wzorzec pobudzenia,

E_{der} – wygładzona wartość pochodnej przybliżonej głośności.

$$E'[k,n]=a\cdot E'[k,n]+(1-a)\cdot E_2[k,n]^{0.3} \quad (3.36)$$

gdzie:

E' – wygładzona wartość przybliżonej głośności.

Modulację obwiedni wyznaczamy ze wzoru:

$$Mod[k,n]=\frac{E_{der}[k,N]}{1+E'[k,n]^{0.3}} \tag{3.37}$$

Implementacja:

Funkcja – `Modulation`

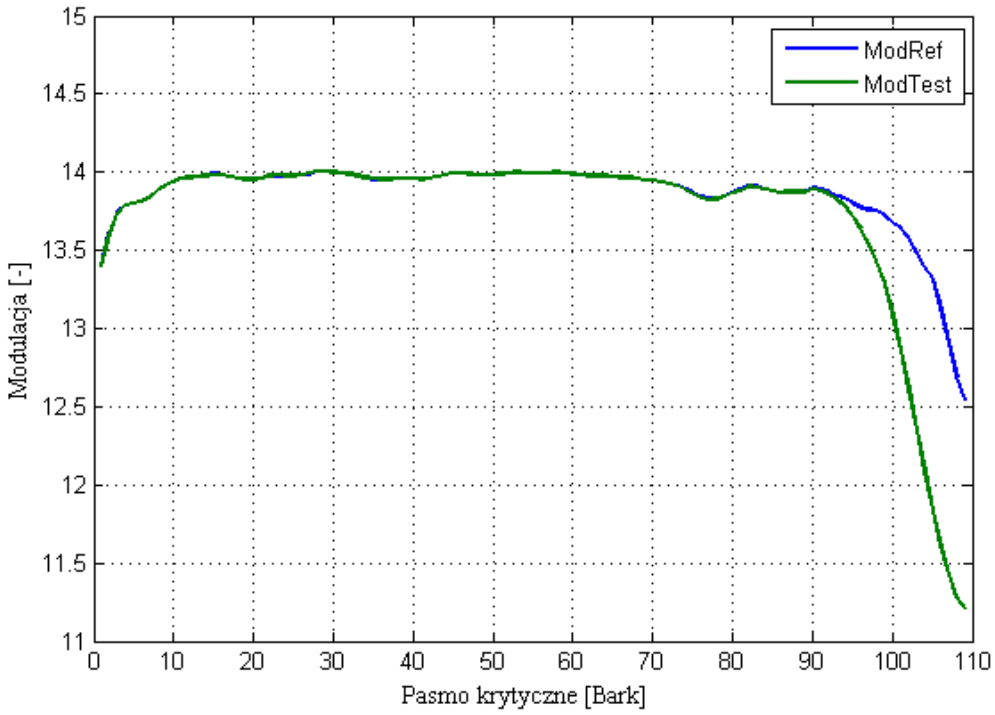
Ponieważ w tej funkcji również występuje filtrowanie, także do niej przekazywane są wartości poprzednich ramek w formie argumentów. Funkcja zwraca także parametr `ERavg` wykorzystywany później w obliczeniach różnicy modulacji.

```
Ed = 48000/1024 .* abs(E2.^0.3-E2b);            %(3.35)
Eder(1,:) = a.*Eder(1,:) + (1-a).*Ed(1,:);      %(3.35)
Eder(2,:) = a.*Eder(2,:) + (1-a).*Ed(2,:);      %(3.35)

E(1,:) = a.*E(1,:) + (1-a).*E2(1,:).^0.3;       %(3.36)
E(2,:) = a.*E(2,:) + (1-a).*E2(2,:).^0.3;       %(3.36)

M = Eder ./ (1 + E./0.3);                       %(3.37)
ERavg = E(1,:);
```

Wartości modulacji dla ramki sygnału referencyjnego i testowanego przedstawia rys. 3.10.



Rys. 3.10. Przebieg modulacji dla: ModRef – sygnału referencyjnego i ModTest – testowanego.

3.3.3 Głośność

Wzorzec głośności dla sygnału referencyjnego i sygnału testowanego wyliczono, korzystając ze wzoru:

$$N[k,n]=const\cdot\left(\frac{1}{s[k]}\frac{E_{Thres}[k]}{10^4}\right)^{0.23}\cdot\left[\left(1-s[k]+\frac{s[k]\cdot E[k,n]}{E_{Thres}[k]}\right)^{0.23}-1\right] \quad (3.38)$$

gdzie:

const – współczynnik skalujący 1.07664,

E_{Thres} – próg pobudzenia (3.39),

s – wskaźnik progu (3.40).

$$E_{Thres}[k]=10^{0.364\cdot\left(\frac{f_c[k]}{kHz}\right)^{-0.8}} \quad (3.39)$$

$$s[k]=10^{\frac{1}{10}\cdot\left[-2-2.05\cdot atan\left(\frac{f_c[k]}{4\text{kHz}}\right)-0.75\cdot atan\left(\left(\frac{f_c[k]}{1.6\text{kHz}}\right)^2\right)\right]} \quad (3.40)$$

Całkowitą głośność wyliczono, korzystając ze wzoru:

$$N_{total}[n]=\frac{24}{Z}\cdot\sum_{k=0}^{Z-1}max(N[k,n],0) \quad (3.41)$$

gdzie:

Z – ilość pasm krytycznych (109).

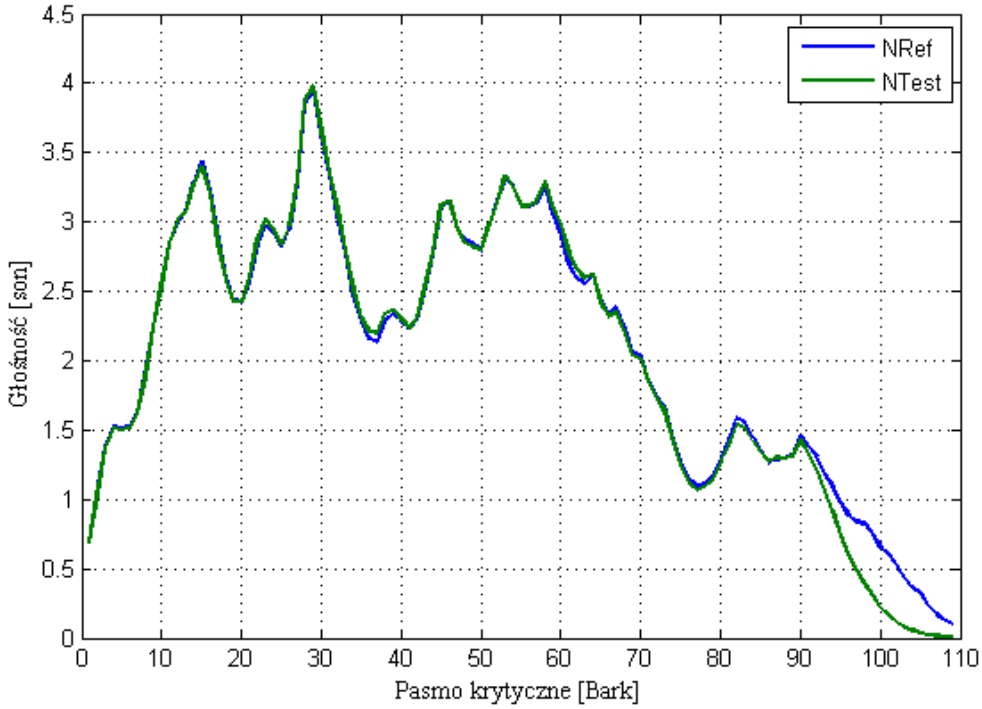
Implementacja:

Funkcja – `Loud`

Parametry `Et`, `s` i `Ets` są wyliczane raz i przechowywane w pamięci.

```
Etres = 10.^(0.364*(fc./1000).^(-0.8));              %(3.39)
s = 10.^((-2 - 2.05 .* atan(fc./4000) -  ...         %(3.40)
   0.75 .* atan((fc./1600).^2)) ./ 10);
Et = const * (Ethres./(s.*1e4)).^0.23;               %(3.38)
N = Et .* ((1 - s + s.*E./Ethres).^0.23 - 1);        %(3.38)
Ntot = (24/Z) * sum(max(N, 0));                      %(3.41)
```

Funkcję głośności dla sygnału referencyjnego i sygnału testowanego przedstawiono na rysunku 3.11.



Rys. 3.11. Przebieg głośności w dla: NRef – sygnału referencyjnego Ntest – sygnału testowanego.

3.3.4 Sygnał błędu

Sygnał błędu obliczony został jako różnica widm (ważonych charakterystyką ucha) sygnałów referencyjnego i testowanego.

$$F_{noise}[k_f, n] = \left| |F_{e,Ref}| - |F_{e,Test}| \right| \tag{3.42}$$

Sygnał błędu jest następnie grupowany w pasma krytyczne tak, jak opisano w podrozdziale 3.2.4. Po grupowaniu otrzymano wzorzec zniekształceń P_{noise}.

Implementacja:

```
Fnoise = abs(Fe(1,:) - Fe(2,:));                    %(3.42)
```

3.4 Parametry wyjściowe MOV

3.4.1 Różnica modulacji

Różnicę modulacji obwiedni chwilowych wyliczono korzystając z równania 3.43.

$$ModDiff[k,n] = w \cdot \frac{\left| Mod_{Test}[k,n] - Mod_{Ref}[k,n] \right|}{offset + Mod_{Ref}[k,n]} \tag{3.43}$$

$$\begin{array}{ll} w = 1 & Mod_{Test}[k,n] > Mod_{Ref}[k,n] \\ w = negWt & Mod_{Test}[k,n] < Mod_{Ref}[k,n] \end{array} \tag{3.44}$$

gdzie:

offset, *negWt* – patrz tabela 3.1.

Ogólną różnicę modulacji obwiedni dla ramki sygnału wyliczono, uśredniając różnice w poszczególnych pasmach krytycznych.

$$ModDiff[n] = \frac{100}{Z} \sum_{k=0}^{Z-1} ModDiff[k,n] \tag{3.45}$$

W późniejszych obliczeniach potrzebna również była funkcja wagowa (3.46), wyliczana na podstawie zmodyfikowanego wzorca pobudzeń (3.36) oraz szumu (3.13)

$$TempWt[n] = \frac{\sum_{k=0}^{Z-1} E'_{Ref}[k,n]}{E'_{Ref}[k,n] + levWt \cdot E_{Thers}[k]^{0.3}} \tag{3.46}$$

gdzie:

levWt – patrz tabela 3.1.

Tab. 3.1. Stałe do obliczeń parametrów MOV modulacji

MOV	negWt	offset	levWt
XxxModDiff1	1	1	100
XxxModDiff2	0,1	0,01	100

Implementacja:

Parametr `Et` jest wyliczany tylko przy pierwszym wywołaniu funkcji.

```
Et = 10.^(0.3.*(0.4*0.364*(fc./1000).^(-0.8)));       %(3.13)

for z=1:Z
    if (Mod(1,z) > Mod(2,z))                          %(3.44)
        num1 = Mod(1,z) - Mod(2,z);
        num2 = negWt2 * num1;
    else
        num1 = Mod(2,z) - Mod(1,z);
        num2 = num1;
    end
    MD1 = num1 / (offset1 + Mod(1,z));                %(3.43)
    MD2 = num2 / (offset2 + Mod(1,z));                %(3.43)
    s1 = s1 + MD1;
    s2 = s2 + MD2;
end
Mt1 = (100/Z)*s1;                                     %(3.45)
Mt2 = (100/Z)*s2;                                     %(3.45)
Wt = sum(ERavg./(ERavg + levWt.*Et));                 %(3.46)
```

WinModDiff1

Średnia okienkowana różnic w modulacjach wyliczana jest ze wzoru ogólnego:

$$WinX = \sqrt{\frac{1}{N-L+1} \cdot \sum_{n=L-1}^{N-1} \left(\frac{1}{L} \cdot \sum_{i=0}^{L-a} \sqrt{X[n-i]} \right)^4} \tag{3.47}$$

gdzie:

N – ilość ramek sygnału,

L – przybliżenie 100ms (4 ramki),

X – wartości uśredniane.

Przy uśrednianiu tego parametru pierwsze 0.5 s pomiaru jest pomijane.

AvgModDiff1* i *AvgModDiff2

Wartości te obliczane są jako ważone czasowo średnie z wartości różnic w modulacjach obwiedni dla każdej ramki sygnału. Różnica między *AvgModDiff1,* a *AvgModDiff2* polega na zastosowaniu innych stałych do obliczeń różnic modulacji (tab. 3.1). Sposób obliczenia średniej ważonej przedstawia wzór (3.48).

$$AvgX = \frac{\sum_{n=0}^{N-1} W[n] \cdot X[n]}{\sum_{n=0}^{N-1} W[n]} \tag{3.48}$$

gdzie:

W – wartości wag (3.46).

Tak samo, jak dla parametru *WinModDiff1,* pierwsze 0.5 s pomiaru jest pomijane przy uśrednianiu.

3.4.2 Głośność zniekształceń

Parametr ten określa zredukowaną głośność zniekształceń w obecności sygnału maskującego (referencyjnego).

$$NL[k,n] = \left(\frac{E_{PThres}}{s_{Test}} \right)^{0.23} \cdot \left[\left(1 + \frac{max\left(s_{PTest} \cdot E_{PTest} - s_{Ref} \cdot E_{PRef}, 0 \right)}{E_{Thres} + s_{Ref} \cdot E_{PRef} \cdot \beta} \right)^{0.23} - 1 \right] \tag{3.49}$$

$$s = ThresFac_0 \cdot Mod[k,n] + S_0 \tag{3.50}$$

$$\beta = \exp\left(-\alpha \cdot \frac{E_{PTest} - E_{PRef}}{E_{PRef}} \right) \tag{3.51}$$

gdzie:

NL – zredukowana głośność zniekształceń,

ThresFac, S_0, α – patrz tabela 3.2,

E_{Thres} – szum (3.13).

Uśrednianie częstotliwościowe wykonuje się według następującego wzoru:

$$NL_{tot}[n] = \frac{24}{Z} \sum_{k=0}^{Z-1} NL[k, n] \tag{3.52}$$

Tab. 3.2. Stałe do obliczeń parametrów MOV głośności zniekształceń.

MOV	α	ThresFac_0	S_0	Nl_{min}
XxxNoiseLoud	1,5	0,15	0,5	0

Implementacja:

Jak w poprzednim przypadku, parametr `Et` jest wyliczany tylko przy pierwszym wywołaniu funkcji.

```
Et = 10.^(0.4*0.364*(fc./1000).^(-0.8));              %(3.13)
sref  = TF0.*Mod(1,:) + S0;                           %(3.50)
stest = TF0.*Mod(2,:) + S0;                           %(3.50)
beta = exp(-alpha.*(EP(2,:)-EP(1,:))./EP(1,:));       %(3.51)
nom = max (stest.*EP(2,:) - sref.*EP(1,:), 0);        %(3.49)
dnom = Et + sref.*EP(1,:) .* beta;                    %(3.49)
NL = 24/Z*sum((Et./stest).^0.23.*...                  %(3.49)
   ((1+nom./dnom).^0.23-1));                          %(3.52)
```

RmsNoiseLoud

RmsNoiseLoud – średnia kwadratowa ze zredukowanych głośności zniekształceń.

$$RmsX = \sqrt{\frac{1}{N} \cdot \sum_{n=0}^{N-1} X[n]^2} \tag{3.53}$$

Chwilowe wartości głośności zniekształceń są brane pod uwagę dopiero po 50 ms, od chwili, gdy w którymkolwiek z kanałów (lewym lub prawym) zarówno sygnał referencyjny, jak i sygnał testowany przekroczą próg N_{Thres} = 0.1 sona. Dodatkowo przy uśrednianiu pomija się pierwsze 0.5 s pomiaru.

3.4.3 Szerokość pasma

Sposób obliczania szerokości pasma rekomendacja [1] przedstawia w formie pseudokodu. Pseudokod ten jest zamieszczony w załączniku nr 2. Ogólnie wyliczanie szerokości pasma odbywa się poprzez iteracyjne porównywanie poziomu dla danej częstotliwości widma z odpowiednim progiem zależnym od tego poziomu. Różnica pomiędzy pseudokodem a napisaną funkcją polega na tym, że porównywane są amplitudy, a nie poziomy (ograniczenie ilości wykonywanych działań) oraz od razu wprowadzony został warunek, że tylko ramki z szerokością większą niż 346 są brane pod uwagę.

BandwidthRef i BandwidthTest

Parametry *BandwidthRef* i *BandwidthTest* są czasową średnią arytmetyczna szerokości pasm wyliczonych dla ramek sygnału referencyjnego i testowanego.

3.4.4 Stosunek zniekształceń od maski

Stosunek zniekształceń od maski jest wyliczany na podstawie wzorca zniekształceń oraz progu maskowania:

$$NMR_{local}[n] = 10 \cdot \log_{10} \frac{1}{Z} \sum_{k=0}^{Z-1} \frac{P_{noise}[k,n]}{M[k,n]} \tag{3.54}$$

TotalNMR

Całkowity stosunek zniekształceń do progu maskowania wyliczono, uśredniając liniowo wartości w każdej ramki sygnału.

$$NMR_{tot} = 10 \cdot \log_{10} \frac{1}{N} \sum_{n} \left(\frac{1}{Z} \sum_{k=0}^{Z-1} \frac{P_{noise}[k,n]}{M[k,n]} \right) \tag{3.55}$$

Implementacja:

Ponieważ do uśredniania potrzebna jest reprezentacja energii sygnału, a nie jego poziom, pominięto na tym etapie logarytmowanie sygnału. Wyliczona została także wartość maksymalnego zniekształcenia, co umożliwiło pominięcie implementacji

kolejnego punktu jako osobnej funkcji.

```
NMRm = Pnoise./Ma;
NMRavg = sum(NMRm)./109;
NMRmax = max(NMRm);
```

3.4.5 Liczba znacząco zniekształconych ramek

RelDistFram

Parametr *RelDistFram* określa ilość ramek, w których w przynajmniej jednym paśmie wystąpi zniekształcenie przekraczające 1.5 dB. Liczba tych ramek jest odniesiona do całkowitej liczby ramek.

$$max\left(10\cdot\log_{10}\left(\frac{P_{noise}[k,n]}{M[k,n]}\right)\right)\geqslant 1.5\text{dB} \tag{3.56}$$

Implementacja:

Obliczenia parametru *RelDistFram* nie zostały zaimplementowane jako osobna funkcja, zostały zintegrowane z funkcją realizującą uśrednianie współczynników MOV.

3.4.6 Prawdopodobieństwo usłyszenia zniekształceń

Algorytm postępowania przy wyliczaniu prawdopodobieństwa usłyszenia zniekształceń został zamieszczony w załączniku nr 3. Po wykonaniu wszystkich kroków otrzymano dwa parametry potrzebne do dalszych obliczeń: całkowite prawdopodobieństwo usłyszenia zniekształceń dla ramki sygnału P_c oraz całkowitą liczbę stopni powyżej progu dla ramki sygnału Q_c.

MFPDB

MFPDB – maksymalne prawdopodobieństwo usłyszenia zniekształceń. Aby zredukować wpływ bardzo krótkich zniekształceń, wygładzono w czasie funkcję prawdopodobieństwa usłyszenia różnicy.

(3.57)

gdzie:

c_0 – współczynnik redukujący wpływ krótkich zniekształceń (0.9)

Maksymalne prawdopodobieństwo usłyszenia zniekształceń wyliczono na podstawie zależności (3.58).

$$PM_c[n] = max\left(PM_c[n-1] \cdot c_{1,} P'_c[n]\right) \tag{3.58}$$

gdzie:

c_1 – współczynnik uwzględniający „zapominanie” (0.99) lub (1)

Współczynnik c_1 uwzględnia proces „zapominania” zniekształceń, które miały miejsce wcześniej. Ponieważ model został zbudowany w oparciu o wyniki testów subiektywnych, w których możliwe było odsłuchanie wybranego fragmentu, współczynnik c_1 powinien mieć wartość 1 (brak efektu zapominania).

ADB

W celu określenia liczby zniekształconych ramek (*ADB*), zlicza się ramki, w których prawdopodobieństwo usłyszenia zniekształceń P_{bin} jest większe niż 0.5 (n_{dist}). Dla tych ramek wyznacza się całkowitą liczbę stopni powyżej progu.

$$Q_{\text{sum}} = \sum_n Q_c[n] \tag{3.59}$$

W zależności od ilości zniekształconych ramek oraz ilości stopni powyżej progu parametr *ADB* przyjmuje następujące wartości

$$\begin{aligned} &jeżeli \quad n_{dist} = 0 && ADB = 0 \\ &jeżeli \quad n_{dist} > 0 \;\; i \;\; Q_{\text{sum}} > 0 && ADB = \log_{10}\left(\frac{Q_{\text{sum}}}{n_{dist}}\right) \\ &jeżeli \quad n_{dist} > 0 \;\; i \;\; Q_{\text{sum}} = 0 && ADB = -0.5 \end{aligned} \tag{3.60}$$

3.4.7 Struktura harmonicznych sygnału błędu

EHS

Parametr *EHS* jest wyliczany przez zmierzenie maksymalnej wartości widma funkcji autokorelacji sygnału błędu. Korelację otrzymano poprzez obliczenie cosinusa kąta pomiędzy dwoma wektorami, zgodnie z równaniem 3.61.

$$C = \frac{\bar{F}_0 \cdot \bar{F}_t}{|\bar{F}_0| \cdot |\bar{F}_t|} \tag{3.61}$$

gdzie:

F_0 – wektor błędu,

F_t – przesunięty wektor błędu.

Wektor błędu jest różnicą logarytmów widm (ważonych charakterystyką ucha) między sygnałem referencyjnym i sygnałem testowanym. Długość korelacji wynosi 256 próbek. Pierwsza wartość korelacji obliczana jest przez nałożenie $F_t[0]$ z $F_0[0]$, a ostatnia przez nałożenie $F_t[0]$ z $F_0[255]$. Otrzymany wektor korelacji zokienkowano znormalizowanym oknem Hanna, usunięto składową stałą (poprzez odjęcie wartości średniej), wykonano FFT i podniesiono do kwadratu (otrzymując energię). Wartość parametru *EHS* to maksymalna wartość widma po pierwszej dolinie. Przy obliczaniach pominięto ramki o niskiej wartości energii (poniżej 8000[1]). Na koniec uśredniono otrzymane wartości i pomnożono przez 1000.

Implementacja:

```
D = log(F2(2,:)./F2(1,:));          %różnica logarytmów widm

F0=D(1:512);
Ft=[zeros(1,255) F0(1:256)];
C = zeros(1,256);
for i=1:256                         % obliczanie korelacji
    Cn=F0(1:255+i)*Ft(257-i:end)';
    Cd=norm(F0(1:255+i))*norm(Ft(257-i:end));
    C(i) = Cn/Cd;
end

C = C-sum(C)/256;                   % odjęcie składowej stałej
C = abs(fft(hw.*C)).^2;             % okienkowanie, fft
                                    % moduł i potęgowanie
```

1 Wartość odniesiona do 16 bitowego sygnału gdzie amplituda sygnału wynosi 32768.

```
  % wykrywanie maksymalnej wartości po pierwszej dolinie
cb = C(1);
cm = 0;
for n=2:128
    if C(n)>cb
        if C(n)>cm
            cm = C(n);
        end
    end
end
EHS = cm;
```

3.5 Ocena jakości

Do uzyskania parametru ODG zastosowano sztuczną sieć neuronową z jedną warstwą ukrytą. Funkcją aktywacji sieci jest funkcja sigmoidalna.

$$sig(x)=\frac{1}{1+e^{-x}} \tag{3.62}$$

Sieć posiada:

I – wejść,

J – węzłów warstwie ukrytej,

$a_{min}[i]$, $a_{max}[i]$ – zestaw współczynników skalujących,

$w_x[i]$ – zestaw wag wejściowych,

$w_y[i]$ – zestaw wag wyjściowych,

b_{min}, b_{max} – wyjściowe czynniki skalujące.

Tabele ze współczynnikami oraz wagami znajdują się w załączniku nr 1.

Wejścia są zamieniane na wskaźnik zniekształceń DI:

$$DI=w_y[J]+\sum_{j=0}^{J-1}\left(wy[j]\cdot sig\left(wx[I,j]+\sum_{i=0}^{I-1}w_x[i,j]\cdot\frac{x[i]-a_{min}[i]}{a_{max}[i]-a_{min}[i]}\right)\right) \tag{3.63}$$

Wskaźnik DI jest następnie zamieniany na ODG według wzoru:

$$ODG=b_{min}+(b_{max}-b_{min})\cdot sig(DI) \tag{3.64}$$

Parametr ODG może przyjmować niewielkie wartości dodatnie. Wynika to z faktu, że algorytm symuluje wyniki testów subiektywnych, w których taki wynik jest możliwy. Wynik taki można otrzymać, jeżeli słuchacz błędnie uzna sygnał testowany za sygnał referencyjny.

Implementacja:

Funkcja - `ArtNet`

```
MOVs = ((MOV-amin)./(amax-amin))';
s = wxb + sum(bsxfun(@times, wx, MOVs));
DI = wyb + sum(wy .* (1./(1+exp(-s))));
ODG = bmin + (bmax-bmin) * (1./(1+exp(-DI)));
```

4 Weryfikacja

Do sprawdzenia wyników działania programu opartego na tej rekomendacji służą pliki testowe. Niestety nie są one powszechnie dostępne, dlatego do weryfikacji użyto darmowego programu opartego na tym samym algorytmie i porównano wyniki uzyskane dla tych samych plików wejściowych. Program nosi nazwę *peaqb 1.0.beta*.

4.1 peaqb 1.0.beta

Program *peaqb 1.0.beta* został napisany dla systemów operacyjnych Linux. Działa w konsoli. Jako parametry przyjmuje ścieżki do plików testowych oraz poziomy odniesienia *Lp*. Program został napisany 2003 roku, a jego autorem jest Giuseppe Gottardi *'oveRet'*. Program został pobrany ze strony:

http://sourceforge.net/projects/peaqb/ .

4.2 Test porównawczy

Do testu zastosowano 30 losowo wybranych plików. Pliki referencyjne to wybrane 10-sekundowe fragmenty plików muzycznych, natomiast pliki testowane powstały poprzez kompresję pliku referencyjnego przy użyciu różnych kodeków audio. Tak powstały plik ponownie zdekodowano do pliku *wav.*

Nagranie trwające 10 s to około 500 ramek do przetworzenia. Wybrane zostały tak krótkie fragmenty, ponieważ program *peaqb 1.0.beta* przetwarza taką ilość ramek w czasie około 15 min, przy czym napisany program przetwarza je w czasie około 30 s.

W tabeli 4.1 zamieszczono wyniki porównania. Jak już wcześniej wspomniano, różnica parametru ODG na poziomie 0.1 jest nieistotna i w praktyce niezauważalna. Średnia różnica wyników między programami wynosi 0,02, maksymalna 0,06, a odchylenie standardowe 0,01.

Dodatkowo przeprowadzono test kodeków audio. Przetestowano 5 różnych kodeków, każdy w kilku wariantach jakości (stopnia kompresji), dla 15 różnych fragmentów muzyki. Łącznie 450 plików testowych. W tabeli 4.2 przedstawiono średnie wartości ODG dla każdego wariantu jakości kodeków.

Tab. 4.1.Wyniki weryfikacji programu.

np.	Kodek	Jakość	Nr pliku	Program 1	Program 2	Różnica
1	AAC	Q2	12	-2,77	-2,74	-0,03
2	AAC	Q3	4	-0,93	-0,92	-0,01
3	AAC	Q3	12	-1,35	-1,35	0,00
4	AAC	Q3	15	-0,79	-0,78	-0,01
5	AAC	Q4	3	-0,50	-0,50	0,00
6	AAC	Q4	12	-0,75	-0,74	-0,01
7	AAC	Q5	12	-0,19	-0,19	0,00
8	MP3	96	5	-0,93	-0,89	-0,04
9	MP3	128	2	-1,28	-1,26	-0,02
10	MP3	192	1	-0,24	-0,22	-0,02
11	MP3	128	6	-1,20	-1,19	-0,01
12	MP3	128	15	-0,68	-0,67	-0,01
13	MP3	192	13	-0,19	-0,19	0,00
14	MP3	96	7	-0,75	-0,73	-0,02
15	MP3	V0	6	-0,08	-0,08	0,00
16	MP3	V2	1	-0,09	-0,09	0,00
17	MP3	V5	4	-0,68	-0,67	-0,01
18	MP3	V7	5	-1,13	-1,09	-0,04
19	MP3	V7	5	-1,13	-1,09	-0,04
20	MP3	V7	13	-1,36	-1,35	-0,02
21	MP3	V9	9	-3,28	-3,28	-0,01
22	MPC	Q3	3	-2,14	-2,13	0,00
23	MPC	Q3	11	-1,74	-1,72	-0,01
24	OGG	Q0	3	-2,87	-2,81	-0,06
25	OGG	Q1	13	-2,94	-2,90	-0,04
26	OGG	Q2	9	-0,91	-0,89	-0,03
27	OGG	Q3	7	-0,75	-0,72	-0,04
28	OGG	Q3	14	-0,78	-0,77	-0,01
29	OGG	Q5	8	-0,39	-0,37	-0,02
30	OGG	Q7	8	-0,01	0,00	0,00
			Średnia			-0,02
			Odchylenie standardowe			0,01
			Maksymalna różnica			0,06

Tab. 4.2.Wyniki oceny jakości kodeków audio.

MP3 [CBR]			MP3 [VBR]		
bitrate [kbps]	**ODG**	**stopień kompresji**	**jakość**	**ODG**	**stopień kompresji**
320	0,03	0,210	V0	0,06	0,183
256	0,00	0,168	V2	-0,03	0,136
192	-0,07	0,126	V5	-0,91	0,089
128	-0,99	0,084	V7	-1,17	0,067
96	-1,08	0,063	V9	-3,06	0,046
48	-3,67	0,031			
OGG			**AAC**		
jakość	**ODG**	**stopień kompresji**	**jakość**	**ODG**	**stopień kompresji**
Q10	0,15	0,295	Q10	0,13	0,292
Q7	0,05	0,142	Q07	0,11	0,198
Q5	-0,31	0,103	Q05	0,00	0,121
Q2	-1,57	0,063	Q03	-1,20	0,060
Q0	-3,04	0,042	Q01	-3,41	0,019
MPC			**WMA [CBR]**		
jakość	**ODG**	**stopień kompresji**	**bitrate [kbps]**	**ODG**	**stopień kompresji**
Q10	0,11	0,222	192	-0,09	0,141
Q7	0,11	0,160	128	-0,69	0,095
Q5	0,00	0,120	96	-1,52	0,072
Q4	-0,52	0,092	64	-2,52	0,048
Q3	-1,61	0,067			

5 Podsumowanie

Porównanie otrzymanych wyników z wynikami z programu *peaqb 1.0.beta* wykazało jednoznacznie, że napisany program działa poprawnie. Różnica w wynikach nie generuje znaczących błędów pomiarowych, ponieważ jest ona na poziomie 0,02. Należy pamiętać, że przeprowadzana ocena jakości ma na celu przybliżenie wyników oceny subiektywnej, a różnica nawet na poziomie 0.1 ODG nie powoduje słyszalnych różnic w jakości dźwięku. Ciekawostką jest to, że napisany program zawsze pokazywał wyższy wynik niż program *peaqb 1.0.beta*. Trudno rozstrzygnąć, który program jest bardziej dokładny – w tym celu konieczne jest zastosowanie plików testowych, do których jednak dostęp jest ograniczony.

Wyniki testu kodeków audio zawarte w tabeli 4.2 wyraźnie pokazują, że w miarę zwiększania kompresji jakość sygnału maleje. Można zauważyć, że do stopnia kompresji na poziomie 1:8 dla większości kodeków utrata jakości jest niewielka.

Program można z powodzeniem stosować do oceny jakości kodeków audio. Ciekawym pomysłem byłoby użycie programu do porównania torów elektroakustycznych stosowanych w technice studyjnej. Głównym jednak zastosowaniem programu jest ocena wdrożenia nowych kodeków i systemów transmisji dźwięku.

Kolejnym etapem prac mogło by być zaimplementowanie wersji rozszerzonej programu opartej nie na modelu FFT, a na bankach filtrów. Wersja FFT została opracowana z myślą o pomiarze w czasie rzeczywistym. Oceniając program w przybliżeniu: czas potrzebny na obliczenie 10-sekundowego sygnału w pakiecie Matlab to ok 30 sekund. Biorąc pod uwagę fakt, że pakiet Matlab nie jest narzędziem do obliczeń w czasie rzeczywistym, sądzę, że bez większych problemów, można by stworzyć taki system działający w czasie rzeczywistym, oczywiście przy zastosowaniu odpowiedniego języka programowania.

Słownik skrótów

DI	Distortion Index – wskaźnik zniekształceń
EHS	Error Harmonic Structure – Struktura harmonicznych sygnału błędu
ITU	International Telecommunication Union
ITU-R	International Telecommunication Union – Radiocommunication Sector
MFPD	Maximum of the Probability of Detection
MOV	Model Output Variable
NMR	Noise-to-Mask Ratio – stosunek zniekształceń od maski
ODG	Objective Difference Grade – stopień obiektywnej różnicy sygnałów
PEAQ	Perceptual Evaluation of Audio Quality – algorytm obiektywnego pomiaru jakości dźwięku
SDG	Subjective Difference Grade – stopień subiektywnej różnicy sygnałów
SNR	Signal-to-Noise Ratio – stosunek sygnału od szumu
THD	Total Harmonic Distortion – zawartość harmonicznych
wav	wave form audio format – format plików dźwiękowych

Bibliografia

[1] Rekomendacja ITU-R BS.1387-1, Method for objective measurements of perceived audio quality, 2001

[2] Rekomendacja ITU-R BS.1116-1, Methods for the subjective assessment of small impairments in audio systems including multichannel soundsystems. 1997

[3] Fastl H., Zwicker E., *Psycho-acoustics, Facts and Models*, Berlin, Springer, 2006

[4] Moore B. C., *Wprowadzenie do psychologii słyszenia.* Warszawa – Poznań, PWN, 1999

[5] Zwicker E., S*ubdivision of the Audible Frequency Range into Critical Bands.* The Journal of the Acoustical Society of America. 33 (2), 1961, 248

Załączniki

1. **Tabele**
2. **Pseudokody**
3. **Uzupełnienie wzorów**
4. **Interfejs graficzny**
5. **Kod źródłowy**

Zawartość DVD

Katalog PEAQ – zawiera funkcje zaimplementowane w pakiecie Matlab.

Katalog Sygnały – zawiera przykładowe sygnały testowe.

Program peaq.exe – skompilowana wersja programu.

Plik peaq.pdf – pełna treść pracy.

Załącznik 1

Tabele

Załącznik zawiera tabele o których mowa w rozdziałach 3.2.4 i 3.5 .

Tab. 1. Częstotliwości graniczne pasm krytycznych.

Nr Pasma	Częstotliwość dolna [Hz]	Częstotliwość środkowa [Hz]	Częstotliwość górna [Hz]	Szerokość pasma [Hz]
0	80,000	91,708	103,445	23,445
1	103,445	115,216	127,023	23,578
2	127,023	138,870	150,762	23,739
3	150,762	162,702	174,694	23,932
4	174,694	186,742	198,849	24,155
5	198,849	211,019	223,257	24,408
6	223,257	235,566	247,950	24,693
7	247,950	260,413	272,959	25,009
8	272,959	285,593	298,317	25,358
9	298,317	311,136	324,055	25,738
10	324,055	337,077	350,207	26,152
11	350,207	363,448	376,805	26,598
12	376,805	390,282	403,884	27,079
13	403,884	417,614	431,478	27,594
14	431,478	445,479	459,622	28,144
15	459,622	473,912	488,353	28,731
16	488,353	502,950	517,707	29,354
17	517,707	532,629	547,721	30,014
18	547,721	562,988	578,434	30,713
19	578,434	594,065	609,885	31,451
20	609,885	625,899	642,114	32,229
21	642,114	658,533	675,161	33,047
22	675,161	692,006	709,071	33,910
23	709,071	726,362	743,884	34,813
24	743,884	761,644	779,647	35,763
25	779,647	797,898	816,404	36,757
26	816,404	835,170	854,203	37,799
27	854,203	873,508	893,091	38,888
28	893,091	912,959	933,113	40,022
29	933,119	953,576	974,336	41,217
30	974,336	995,408	1016,797	42,461
31	1016,797	1038,511	1060,555	43,758
32	1060,555	1082,938	1105,666	45,111
33	1105,666	1128,746	1152,187	46,521
34	1152,187	1175,995	1200,178	47,991
35	1200,178	1224,744	1249,700	49,522
36	1249,700	1275,055	1300,816	51,116
37	1300,816	1326,992	1353,592	52,776
38	1353,592	1380,623	1408,094	54,502
39	1408,094	1436,014	1464,392	56,298
40	1464,392	1493,237	1522,559	58,167

Nr Pasma	Częstotliwość dolna [Hz]	Częstotliwość środkowa [Hz]	Częstotliwość górna [Hz]	Szerokość pasma [Hz]
41	1522,559	1552,366	1582,668	60,109
42	1582,668	1613,474	1644,795	62,127
43	1644,795	1676,641	1709,021	64,226
44	1709,021	1741,946	1775,427	66,406
45	1775,427	1809,474	1844,098	68,671
46	1844,098	1879,310	1915,121	71,023
47	1915,121	1951,543	1988,587	73,466
48	1988,587	2026,266	2064,590	76,003
49	2064,590	2103,573	2143,227	78,637
50	2143,227	2183,564	2224,597	81,370
51	2224,597	2266,340	2308,806	84,209
52	2308,806	2352,008	2395,959	87,153
53	2395,959	2440,675	2486,169	90,210
54	2486,169	2532,456	2579,551	93,382
55	2579,551	2627,468	2676,223	96,672
56	2676,223	2725,832	2776,309	100,086
57	2776,309	2827,672	2879,937	103,628
58	2879,937	2933,120	2987,238	107,301
59	2987,238	3042,309	3098,350	111,112
60	3098,350	3155,379	3213,415	115,065
61	3213,415	3272,475	3332,579	119,164
62	3332,579	3393,745	3455,993	123,414
63	3455,993	3519,344	3583,817	127,824
64	3583,817	3649,432	3716,212	132,395
65	3716,212	3784,176	3853,348	137,136
66	3853,817	3923,748	3995,399	141,582
67	3995,399	4068,324	4142,547	147,148
68	4142,547	4218,090	4294,979	152,432
69	4294,979	4373,237	4452,890	157,911
70	4452,890	4533,963	4643,482	190,592
71	4616,482	4700,473	4785,962	169,480
72	4785,962	4872,978	4961,548	175,586
73	4961,548	5051,700	5143,463	181,915
74	5143,463	5236,866	5331,939	188,476
75	5331,939	5428,712	5527,217	195,278
76	5527,217	5627,484	5729,545	202,328
77	5729,545	5833,434	5939,183	209,638
78	5939,183	6046,825	6156,396	217,213
79	6156,396	6267,931	6381,463	225,067
80	6381,463	6497,031	6614,671	233,208
81	6614,671	6734,420	6856,316	241,645
82	6856,316	6980,399	7106,708	250,392
83	7106,708	7235,284	7366,166	259,458
84	7366,166	7499,397	7635,020	268,854
85	7635,020	7773,077	7913,614	278,594
86	7913,614	8056,673	8202,302	288,688
87	8202,302	8350,547	8501,454	299,152
88	8501,454	8655,072	8811,450	309,996
89	8811,450	8970,639	9132,688	321,238
90	9132,688	9297,648	9465,574	332,886

Nr Pasma	Częstotliwość dolna [Hz]	Częstotliwość środkowa [Hz]	Częstotliwość górna [Hz]	Szerokość pasma [Hz]
91	9465,574	9636,520	9810,536	344,962
92	9810,536	9987,683	10168,013	357,477
93	10168,013	10351,586	10538,460	370,447
94	10538,460	10728,695	10922,351	383,891
95	10922,351	11119,490	11320,175	397,824
96	11320,175	11524,470	11732,438	412,263
97	11732,438	11944,149	12159,670	427,232
98	12159,670	12379,066	12602,412	442,742
99	12602,412	12829,775	13061,229	458,817
100	13061,229	13294,850	13536,710	475,481
101	13536,710	13780,887	14029,458	492,748
102	14029,458	14282,503	14540,103	510,645
103	14540,103	14802,338	15069,295	529,192
104	15069,295	15341,057	15617,710	548,415
105	15617,710	15899,345	16186,049	568,339
106	16186,049	16477,914	16775,035	588,986
107	16775,035	17077,504	17385,420	610,385
108	17385,420	17690,045	18000,000	614,580

Tab. 2. Współczynniki skalujące dla parametrów wejściowych sztucznej sieci neuronowej.

indeks [i]	MOV x[i]	a_{min} [i]	a_{max} [i]
0	BandwidthRef	393,916656	921
1	BandwidthTest	361,965332	881,131226
2	TotalNMR	-24,045116	16,21203
3	WinModDiff1	1,110661	107,137772
4	ADB	-0,206623	2,886017
5	EHS	0,074318	13,933351
6	AvgModDiff1	1,113683	63,257874
7	AvgModDiff2	0,950345	1145,018555
8	RmsNoiseLoud	0,029985	14,81974
9	MFPD	0,000101	1
10	RelDistFrames	0	1

Tab. 3. Wagi węzłów wejściowych sztucznej sieci neuronowej.

indeks [i]	MOV x[i]	węzeł 1 wx [i,0]	węzeł 2 wx [i,1]	węzeł 3 wx [i,2]
0	BandwidthRef	-0,502657	0,436333	1,219602
1	BandwidthTest	4,307481	3,246017	1,123743
2	TotalNMR	4,984241	-2,211189	-0,192096
3	WinModDiff1	0,051056	-1,762424	4,331315
4	ADB	2,32158	1,789971	-0,75456
5	EHS	-5,303901	-3,452257	-10,814982
6	AvgModDiff1	2,730991	-6,111805	1,519223
7	AvgModDiff2	0,62495	-1,331523	-5,955151
8	RmsNoiseLoud	3,102889	0,87126	-5,922878
9	MFPD	-1,051468	-0,939882	-0,142913
10	RelDistFrames	-1,804679	-0,50361	-0,620456
11	bias	-2,518254	0,654841	-2,207228

Tab. 4. Wagi węzłów wyjściowych sztucznej sieci neuronowej.

węzeł 1 wy [i,0]	węzeł 2 wy [i,1]	węzeł 3 wy [i,2]	bias wy [i,3]
-3,817048	4,107138	4,629582	-0,307594

Tab. 5. Współczynniki skalujące ODG.

	bmin	bmax
ODG	-3,98	0,22

Załącznik 2

Pseudokody

Załącznik zawiera pseudokody które wykorzystano podczas implementacji funkcji opisanych w podrozdziałach: 3.2.4 i 3.4.3

Pseudokod do podziału na pasma krytyczne dla podrozdziału 3.2.4 Grupowanie.

```
/* wejścia */
Fsp[ ]                          :     energia wejścia
/* wyjścia */
Pe[ ]                           :     energia zgrupowana
/* zmienne */
i                               :     numer pasma krytycznego
k                               :     numer pasma FFT
Z                               :     liczba pasm krytycznych
fl[]                            :     dolne granice pasm
fu[]                            :     górne granice pasm
Fres                            :     rozdzielczość


Fres = 48000/2048;
for(i=0; i<Z; i++)
{
   Pe[i]=0;
   for(k=0;k<1024;k++)
   {
     /* pasmo widma wewnątrz pasma krytycznego */
          if( (( k-0.5)*Fres >= fl[i])  && ((k+0.5)*Fres <= fu[i]))
     {
            Pe[i] += Fsp[k];
     }
     /* pasmo krytyczne wewnątrz pasma widma*/
          else if( (( k-0.5)*Fres < fl[i])  && ((k+0.5)*Fres > fu[i]))
     {
       Pe[i] += Fsp[k]*(fu[i]-fl[i])/Fres;
     }
     /* lewa granica */
     else if( ((k-0.5)*Fres < fl[i]) && ((k+0.5)*Fres > fl[i]))
     {
       Pe[i] += Fsp[k]*( (k+0.5)*Fres - fl[i])/Fres;
     }
     /* prawa granica */
     else if( ((k-0.5)*Fres < fu[i]) && ((k+0.5)*Fres > fu[i]);
     {
       Pe[i] += Fsp[k]*(fu[i]- (k-0.5)*Fres)/Fres;
     }
```

```
    /* pasmo widma poza pasmem krytycznym */
    else
    {
      Pe[i] += 0;
    }
  }

    /* ograniczenie wyniku */
    Pe[i]=max(Pe[i],0.000000000001);
}
```

Pseudokod do wyliczenia szerokości pasma dla podrozdziału 3.4.3 Szerokość pasma.

```
/* wejścia */
FLevRef[], FlevelTest[]       :     poziom widm dB
/* wyjścia */
BwRef, BwTest                 :     parametry wyjściowe
/* zmienne */
k                             :     indeks pasma FFT
ZeroThreshold                 :     próg pasma


ZeroThreshold = FLevelTst(921);
BwRef = BwTst = 0.0;
for(k=921;k<1024;k++)
{
  ZeroThreshold=max(ZeroThreshold,FLevelTst(k));
}

for (k = 920; k>=0; k--)
{
  if (FLevelRef[k] >= 10.0+ZeroThreshold)
  {
    BwRef = k+1;
    break;
  }
}

for (k = BwRef-1; k>=0; k--)
{
  if(FLeveltest[k] >= 5.0+ZeroThreshold)
  {
    BwTest=k+1;
    break;
  }
}
```

Załącznik 3

Uzupełnienie wzorów

Załącznik zawiera uzupełnienia wzorów dla podrozdziałów: 3.2.6 i 3.4.6 .

Uzupełnienie wzorów dla podrozdziału 3.2.6 Wygładzanie w dziedzinie częstotliwości.

$$E_{line}[j,k,n]=\left\{\begin{array}{ll}\dfrac{10^{\frac{L[j,n]}{10}}\cdot 10^{\frac{-res\cdot(j-k)\cdot s_l[j,L[j,n]]}{10}}}{\sum_{\mu=0}^{j-1}10^{\frac{-res\cdot(j-\mu)\cdot s_l[j,L[j,n]]}{10}}+\sum_{\mu=j}^{Z-1}10^{\frac{-res\cdot(\mu-j)\cdot s_u[j,L[j,n]]}{10}}} & dla\ k<j \\ \dfrac{10^{\frac{L[j,n]}{10}}\cdot 10^{\frac{-res\cdot(k-j)\cdot s_u[j,L[j,n]]}{10}}}{\sum_{\mu=0}^{j-1}10^{\frac{-res\cdot(j-\mu)\cdot s_l[j,L[j,n]]}{10}}+\sum_{\mu=j}^{Z-1}10^{\frac{-res\cdot(\mu-j)\cdot s_u[j,L[j,n]]}{10}}} & dla\ k\geqslant j\end{array}\right\} \quad (1)$$

$$Norm_{SP}[k]=\left(\sum_{j=0}^{Z-1}\widetilde{E_{line}}[j,k]^{0.4}\right)^{\frac{1}{0.4}} \quad (2)$$

$$\widetilde{E_{line}}[j,k,n]=\left\{\begin{array}{ll}\dfrac{10^{\frac{-res\cdot(j-k)\cdot s_l[j,0]}{10}}}{\sum_{\mu=0}^{j-1}10^{\frac{-res\cdot(j-\mu)\cdot s_l[j,0]}{10}}+\sum_{\mu=j}^{Z-1}10^{\frac{-res\cdot(\mu-j)\cdot s_u[j,0]}{10}}} & dla\ k<j \\ \dfrac{10^{\frac{-res\cdot(k-j)\cdot s_u[j,0]}{10}}}{\sum_{\mu=0}^{j-1}10^{\frac{-res\cdot(j-\mu)\cdot s_l[j,0]}{10}}+\sum_{\mu=j}^{Z-1}10^{\frac{-res\cdot(\mu-j)\cdot s_u[j,0]}{10}}} & dla\ k\geqslant j\end{array}\right\} \quad (3)$$

W implementacji dokonano kilku przekształceń w celu ograniczenia ilości obliczeń. Wzór przekształcono do ogólnej postaci:

$$E_{line}[j]=\begin{cases} E[j]\cdot S_l[k] & dla\ k<j \\ E[j]\cdot S_u[k] & dla\ k\geqslant j \end{cases} \tag{4}$$

gdzie:

$$S_l[k]=\left(10^{\frac{-27\cdot res}{10}}\right)^{0.4} \tag{5}$$

$$S_u[k]=\left(10^{\frac{\left(-24-\frac{230}{fc[j]}\right)\cdot res}{10}}\cdot P_p[j]^{0.2\cdot res}\right)^{0.4} \tag{6}$$

$$E[j]=\frac{P_p[j]}{S_{ld}[j]+S_{ud}[j]} \tag{7}$$

Wartości S_{ud} i S_{ld} są wyliczane poprzez wykorzystanie wzoru na sumę ciągu geometrycznego.

Uzupełnienie wzorów dla podrozdziału 3.4.6 Prawdopodobieństwo usłyszenie zniekształceń.

Algorytm postępowania w celu wyliczenia prawdopodobieństwa usłyszenia zniekształceń oraz liczby stopni powyżej progu dla podrozdziału 3.4.6 Prawdopodobieństwo usłyszenia zniekształceń.

Wzorce pobudzeń należy zlogarytmować:

$$E'[k,n]=10\cdot\log_{10}\left(E[k,n]\right) \tag{8}$$

Następnie wyznaczyć kolejno:

- średnie niesymetryczne pobudzenie,

$$L[k,n]=0.3\cdot max\left(E'_{Ref}[k,n],E'_{Test}[k,n]\right)+0.7\cdot E'_{Test}[k,n] \tag{9}$$

- efektywny stopień wykrywania *s* (przybliżenie najmniejszej słyszalnej różnicy), dla L > 0:

$$\begin{aligned} s[k,n] = 5.95072 \cdot \left(\frac{6.39468}{L[k,n]}\right)^{1..71332} - 0.198719 + ... \\ ... + 5.50197\mathrm{e}^{-2} \cdot L[k,n] - 1.02438\mathrm{e}^{-3} \cdot L[k,n]^2 + ... \\ ... + 5.05622\mathrm{e}^{-6} \cdot L[k,n]^3 + 9.01033\mathrm{e}^{-11} \cdot L[k,n]^4 \end{aligned} \quad , \tag{10}$$

dla L ≤ 0:

$$s[k,n] = 10^{30} \tag{11}$$

- różnicę poziomów pobudzeń,

$$e[k,n] = E'_{Ref}[k,n] - E'_{Test}[k,n] \tag{12}$$

Jeżeli *e* > 0 to stromość b (3.67) przyjmuje wartość 0.4 w przeciwnym wypadku 0.6. Zależność ta uwzględnia fakt, że różnica między sygnałami jest bardziej rażąca w przypadku gdy sygnał testowany jest głośniejszy od referencyjnego.

- współczynnik skalowania *a,*

$$a[k,n] = \frac{10^{\frac{\log_{10}(\log_{10}(2))}{b}}}{s[k,n]} \tag{13}$$

- prawdopodobieństwo usłyszenia zniekształceń,

$$p_c[k,n] = 1 - 10^{(-a[k,n] \cdot e[k,n]^b)} \tag{14}$$

Jeżeli współczynnik skalowania *a* jest równy *s* to prawdopodobieństwo że zniekształcenia zostaną usłyszane wynosi 0.5.

- całkowitą liczbę stopni powyżej progu detekcji,

$$q_c[k,n]=\frac{|\mathrm{INT}(e[k,n])|}{s[k,n]} \tag{15}$$

- prawdopodobieństwo usłyszenia zniekształceń dla sygnału stereo,

$$p_{bin}[k,n]=max(p_{\mathrm{left}}[k,n], p_{\mathrm{right}}[k,n]) \tag{16}$$

- liczę stopni powyżej progu detekcji dla sygnału stereo,

$$q_{bin}[k,n]=max(q_{\mathrm{left}}[k,n], q_{\mathrm{right}}[k,n]) \tag{17}$$

- całkowite prawdopodobieństwo usłyszenia zniekształceń dla ramki sygnału,

$$P_c[n]=1-\prod_k (1-p_c[k,n]) \tag{18}$$

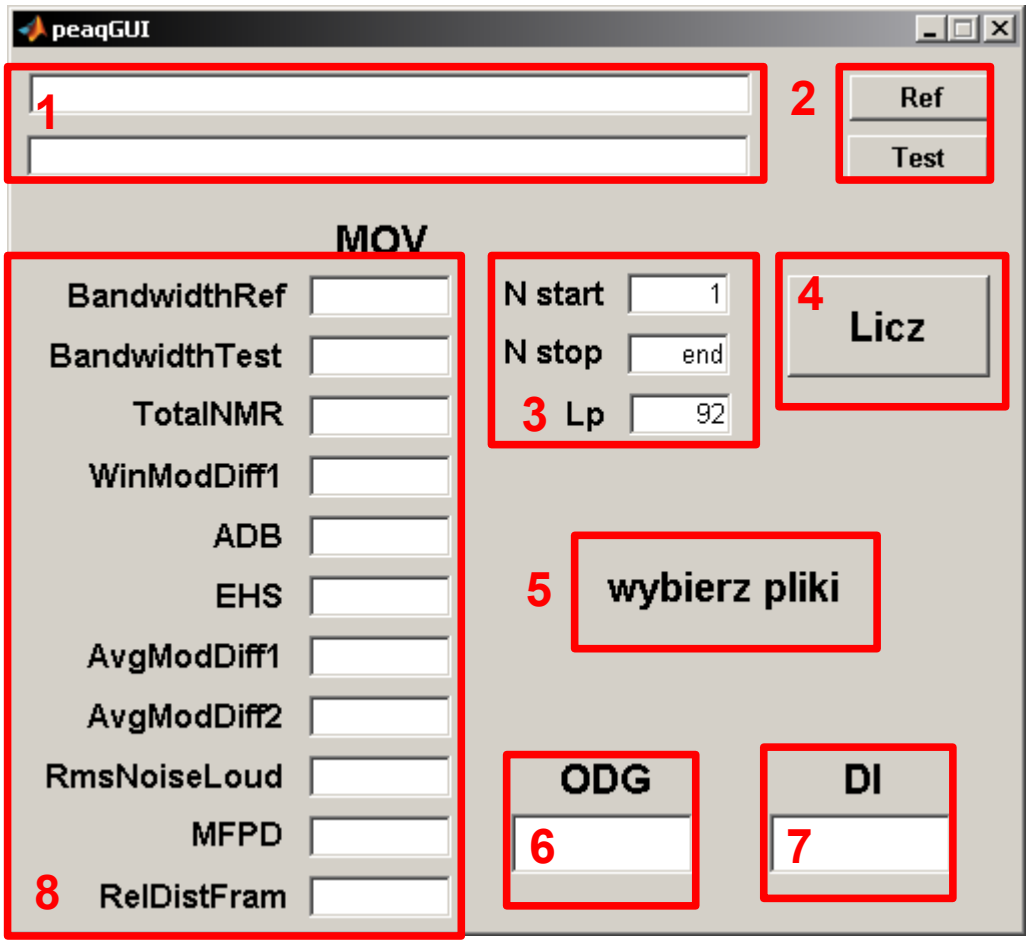
- całkowitą liczba stopni powyżej progu dla ramki sygnału.

$$Q_c=\sum_k q_c[k,n] \tag{19}$$

Załącznik 4

Interfejs graficzny

Prosty interfejs graficzny wykonano, aby możliwe było użycie programu bez konieczności uruchamiania całego pakietu Matlab.



Rys. 1. Interfejs graficzny.

1. Pola ścieżek dostępu dla pliku referencyjnego (u góry) i testowanego (u dołu).
2. Pola wyboru pliku referencyjnego i pliku testowanego.
3. Pole parametrów wejściowych: numer ramki początkowej, końcowej oraz poziom odsłuchu.
4. Przycisk licz powoduje rozpoczęcie obliczeń.
5. Pole dialogowe, informuje jeżeli wprowadzone parametry są błędne.

6. Uzyskany stopień obiektywnej różnicy ODG.
7. Uzyskana wartość zniekształceń DI.
8. Uzyskane parametry MOV.

Załącznik 5

Kod źródłowy

Lista plików funkcyjnych

1 Pliki główne
- 1.1 peaq.m
- 1.2 LoadData.m
- 1.3 Cbands.m

2 Model ucha
- 2.1 FFTnorm.m
- 2.2 OMEar.m
- 2.3 CBGroup.m
- 2.4 AddNoise.m
- 2.5 Spread.m
- 2.6 TimeSpread.m
- 2.7 Mthres.m

3 Przetwarzanie wstępne
- 3.1 LPAdapt.m
- 3.2 Modulation.m
- 3.3 Loud.m

4 Parametry wyjściowe MOV
- 4.1 MOV_mod.m
- 4.2 Nloud.m
- 4.3 MOV_Bwidth.m
- 4.4 MOV_NMR.m
- 4.5 MOV_Dprob.m
- 4.6 MOV_Eharm.m

5 Uśrednianie MOV i ocena jakości.
- 5.1 MOV_avg.m
- 5.2 ArtNet.m

Ze względu na dużą ilość kodu, wersja drukowana nie zawiera pełnego kodu źródłowego. Pełny kod znajduje się na płycie DVD

1 Pliki główne

1.1 peaq.m

```
function [ODG DI MOV] = peaq(ref, test, nstart, nstop, Lp)
% [ODG DI MOV] = peaq(ref, test, nstart, nstop)
%
% Funkcja służy do obiektywnej oceny jakości dźwięku.
%
% Argumenty funkcji
% string ref - sciezka do pliku referencyjnego (plik WAV)
% string test - sciezka do pliku testowanego (plik WAV)
% int nstart - numer ramki poczatkowej
% int nstop - numer ramki koncowej (jezeli nstop=0 to nstop=dlugosc
                                        pliku)
% double Lp - poziom ods³uchu SPL dla 1019.5Hz w dBFS
%
% Funkcja zwraca
% double ODG - warotsc wyliczonego ODG (obiektywna roznica -3.88 do
                                        0.22)
% double DI - wskaznik zniekształcen DI
% double MOV[] - parametry wyjsciowe MOV
%
% Uwagi:
% Sygnaly wejsciowe powinny sobie odpowiadac z dokladnoœcia do 24
                                        próbek
% Program oparty o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

addpath('sig', 'Init', 'Earmod', 'PreProcess', 'MOV');

if nargin < 3
    Lp = 92;
end
if nargin < 4
    nstop = 0;
end
if nargin < 5
    nstart = 1;
end

PAR = SetPAR(nstart, nstop, Lp);
MEM = SetMEM(PAR.Z);

[xR, xT, fnum] = LoadData(ref, test, PAR.fs);

PAR.C = size(xR,1);
if PAR.END == 0
    PAR.END = fnum;
end
PAR.N = PAR.END-PAR.STA+1;

for n = 1:PAR.STA:PAR.END-1
```

```
% Time processing
tR = xR(:,PAR.fram*(n-1)+1:PAR.fram*(n+1));
tT = xT(:,PAR.fram*(n-1)+1:PAR.fram*(n+1));

for c = 1:PAR.C
    %% FFT-based ear model

    % FFT
    F(1,:) = FFTnorm(tR(c,:), PAR.fs, PAR.Lp);
    F(2,:) = FFTnorm(tT(c,:), PAR.fs, PAR.Lp);

    % Outer and middle ear => Outer ear weighted FFT outputs
    Fe(1,:) = OMEar(F(1,:), PAR.fs);
    Fe(2,:) = OMEar(F(2,:), PAR.fs);

    % Grouping into critical bands
    Pe(1,:) = CBGroup(Fe(1,:), PAR.Z, PAR.fs, PAR.fram);
    Pe(2,:) = CBGroup(Fe(2,:), PAR.Z, PAR.fs, PAR.fram);

    % Adding internal noise => "Pitch patterns"
    Pp(1,:) = AddNoise(Pe(1,:));
    Pp(2,:) = AddNoise(Pe(2,:));

    % Spreading  => "Unsmeared excitation patterns"
    Norm = Spread(ones(1,109), ones(1,109), PAR.Z, PAR.res);
    E2(1,:) = Spread(Pp(1,:), Norm, PAR.Z, PAR.res);
    E2(2,:) = Spread(Pp(2,:), Norm, PAR.Z, PAR.res);

    % Time domain spreading => "Excitation patterns"
    [E(1,:), MEM(c).Ef(1,:)] = TimeSpread(E2(1,:),
                                   MEM(c).Ef(1,:));
    [E(2,:), MEM(c).Ef(2,:)] = TimeSpread(E2(2,:),
                                   MEM(c).Ef(2,:));

    % Masking threshold
    M(1,:) = MThres(E(1,:));

    %% Pre-processing of excitation patterns

    % Level and pattern adaptation => "Spectrally adapted
                                   patterns"
    [EP, MEM(c).P, MEM(c).Rn, MEM(c).Rd, MEM(c).PC] = ...
        LPAdapt(E, PAR.Z, MEM(c).P, MEM(c).Rn, MEM(c).Rd,
                                   MEM(c).PC);

    % Modulation
    [Mod, ERavg, MEM(c).E2b, MEM(c).Eder, MEM(c).E] = ...
        Modulation(E2, MEM(c).E2b, MEM(c).Eder, MEM(c).E);

    % Loudness
    MOVn.NRef(c,n)  = Loud(E(1,:), PAR.Z);
    MOVn.NTest(c,n)  = Loud(E(2,:), PAR.Z);

    % Calculation of the error signal
    Fnoise = abs(Fe(1,:) - Fe(2,:));
    Pnoise = CBGroup(Fnoise, PAR.Z, PAR.fs, PAR.fram);

    %% Calculation of Model Output Variables
```

```
        % Modulation differences
        [MOVn.Mt1(c,n) MOVn.Mt2(c,n) MOVn.Wt(c,n)] = ...
            MOV_mod(Mod, ERavg, PAR.Z);

        % Noise Loudness
        MOVn.NL(c,n) = MOV_Nloud(Mod, EP, PAR.Z);

        % Bandwidth
        [MOVn.BWRef(c,n) MOVn.BWTest(c,n)] = MOV_Bwidth(F);

        % Noise-to-mask ratios
        [MOVn.NMRavg(c,n) MOVn.NMRmax(c,n)] = MOV_NMR(Pnoise, M(1,:));

        % Probability of detection
        [MOVn.p(c,n,:) MOVn.q(c,n,:)] = MOV_Dprob(E);

        % Error harmonic structure
        MOVn.EHS(c,n) = MOV_Eharm(tR(c,:), tT(c,:), F, PAR.fram);

    end
end

MOV = MOV_avg(MOVn);

[ODG DI MOVs] = ArtNet(MOV);

function PAR = SetPAR(nstart, nstop, Lp)

PAR.fs = 48000;
PAR.A = 32768;
PAR.fram = 1024;
PAR.Lp = Lp;
PAR.Z = 109;
PAR.res = 0.25;
PAR.C = 0;

if nstart>=1
    PAR.STA = nstart;
else
    error('niewlaœciwa wartosc poczatkowa');
end
if nstop>nstart+1
    PAR.END = nstop+1;
elseif nstop == 0
    PAR.END = 0;
else
    error('niewlasciwa wartosc koncowa minimalna wartosc nstrat+2');
end

function MEM = SetMEM(Z)

MEM(1:2) = struct('Ef',zeros(2,Z),...
    'P',zeros(2,Z),...
    'Rn',zeros(1,Z),...
    'Rd',zeros(1,Z),...
    'PC',zeros(2,Z),...
    'E2b',zeros(2,Z),...
    'Eder',zeros(2,Z),...
    'E',zeros(2,Z));
```

1.2 LoadData.m

```
function [xR xT N] = LoadData(ref, test, fs)
% [xR xT N] = LoadData(ref, test, fs)
%
% Funkcja sluzy do przygotowania danych do dalszej obróbki.
%
% Argumenty funkcji
% string ref - sciezka do pliku referencyjnego (plik WAV)
% string test - sciezka do pliku testowanego (plik WAV)
% double fs - czestotliwosc probkowania
%
% Funkcja zwraca
% double xR[] - sygnal referencyjny (FS=48kHz)
% double xT[] - sygnal testowany (FS=48kHz)
% double N - ilosc ramek sygnalu
%
% Uwagi:
% Funkcja przygotowuje dane dla funkcji peaq.m
%
% Funkcja skladowa programu do pomiaru jakosci dzieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

[xR fR] = wavread(ref);
[xT fT] = wavread(test);

if fR~=fs
    xR = resample(xR,fs,fR);
end
if fT~=fs
    xT = resample(xT,fs,fT);
end
en = min(size(xR,1),size(xT,1));
xR = xR(1:en,:)';
xT = xT(1:en,:)';

N = floor(size(xR,2)/1024);
xR = xR(:,1:N*1024);
xT = xT(:,1:N*1024);
```

1.3 Cbands.m

```
function [fc fl fu] = CBands()
% [fc fl fu] = CBands()
%
% Funkcja sluzy do pozyskania wartosci czestotliwoœci pasm
                                        krytycznych.
%
% Argumenty funkcji
%
% Funkcja zwraca
```

```
% double fc[] - wartosci srodkowych czestotliwosci filtrów
% double fl[] - wartosci dolnych czestotliwosci filtrów
% double fu[] - wartosci górnych czestotliwosci filtrów
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwiêku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

fl = [  80.000,   103.445,   127.023,   150.762,   174.694, ...
       198.849,   223.257,   247.950,   272.959,   298.317, ...
       324.055,   350.207,   376.805,   403.884,   431.478, ...
       459.622,   488.353,   517.707,   547.721,   578.434, ...
       609.885,   642.114,   675.161,   709.071,   743.884, ...
       779.647,   816.404,   854.203,   893.091,   933.119, ...
       974.336,  1016.797,  1060.555,  1105.666,  1152.187, ...
      1200.178,  1249.700,  1300.816,  1353.592,  1408.094, ...
      1464.392,  1522.559,  1582.668,  1644.795,  1709.021, ...
      1775.427,  1844.098,  1915.121,  1988.587,  2064.590, ...
      2143.227,  2224.597,  2308.806,  2395.959,  2486.169, ...
      2579.551,  2676.223,  2776.309,  2879.937,  2987.238, ...
      3098.350,  3213.415,  3332.579,  3455.993,  3583.817, ...
      3716.212,  3853.817,  3995.399,  4142.547,  4294.979, ...
      4452.890,  4616.482,  4785.962,  4961.548,  5143.463, ...
      5331.939,  5527.217,  5729.545,  5939.183,  6156.396, ...
      6381.463,  6614.671,  6856.316,  7106.708,  7366.166, ...
      7635.020,  7913.614,  8202.302,  8501.454,  8811.450, ...
      9132.688,  9465.574,  9810.536, 10168.013, 10538.460, ...
     10922.351, 11320.175, 11732.438, 12159.670, 12602.412, ...
     13061.229, 13536.710, 14029.458, 14540.103, 15069.295, ...
     15617.710, 16186.049, 16775.035, 17385.420 ];

fc = [  91.708,   115.216,   138.870,   162.702,   186.742, ...
       211.019,   235.566,   260.413,   285.593,   311.136, ...
       337.077,   363.448,   390.282,   417.614,   445.479, ...
       473.912,   502.950,   532.629,   562.988,   594.065, ...
       625.899,   658.533,   692.006,   726.362,   761.644, ...
       797.898,   835.170,   873.508,   912.959,   953.576, ...
       995.408,  1038.511,  1082.938,  1128.746,  1175.995, ...
      1224.744,  1275.055,  1326.992,  1380.623,  1436.014, ...
      1493.237,  1552.366,  1613.474,  1676.641,  1741.946, ...
      1809.474,  1879.310,  1951.543,  2026.266,  2103.573, ...
      2183.564,  2266.340,  2352.008,  2440.675,  2532.456, ...
      2627.468,  2725.832,  2827.672,  2933.120,  3042.309, ...
      3155.379,  3272.475,  3393.745,  3519.344,  3649.432, ...
      3784.176,  3923.748,  4068.324,  4218.090,  4373.237, ...
      4533.963,  4700.473,  4872.978,  5051.700,  5236.866, ...
      5428.712,  5627.484,  5833.434,  6046.825,  6267.931, ...
      6497.031,  6734.420,  6980.399,  7235.284,  7499.397, ...
      7773.077,  8056.673,  8350.547,  8655.072,  8970.639, ...
      9297.648,  9636.520,  9987.683, 10351.586, 10728.695, ...
     11119.490, 11524.470, 11944.149, 12379.066, 12829.775, ...
     13294.850, 13780.887, 14282.503, 14802.338, 15341.057, ...
     15899.345, 16477.914, 17077.504, 17690.045 ];
```

```
fu = [ 103.445,   127.023,   150.762,   174.694,   198.849, ...
       223.257,   247.950,   272.959,   298.317,   324.055, ...
       350.207,   376.805,   403.884,   431.478,   459.622, ...
       488.353,   517.707,   547.721,   578.434,   609.885, ...
       642.114,   675.161,   709.071,   743.884,   779.647, ...
       816.404,   854.203,   893.091,   933.113,   974.336, ...
      1016.797,  1060.555,  1105.666,  1152.187,  1200.178, ...
      1249.700,  1300.816,  1353.592,  1408.094,  1464.392, ...
      1522.559,  1582.668,  1644.795,  1709.021,  1775.427, ...
      1844.098,  1915.121,  1988.587,  2064.590,  2143.227, ...
      2224.597,  2308.806,  2395.959,  2486.169,  2579.551, ...
      2676.223,  2776.309,  2879.937,  2987.238,  3098.350, ...
      3213.415,  3332.579,  3455.993,  3583.817,  3716.212, ...
      3853.348,  3995.399,  4142.547,  4294.979,  4452.890, ...
      4643.482,  4785.962,  4961.548,  5143.463,  5331.939, ...
      5527.217,  5729.545,  5939.183,  6156.396,  6381.463, ...
      6614.671,  6856.316,  7106.708,  7366.166,  7635.020, ...
      7913.614,  8202.302,  8501.454,  8811.450,  9132.688, ...
      9465.574,  9810.536, 10168.013, 10538.460, 10922.351, ...
     11320.175, 11732.438, 12159.670, 12602.412, 13061.229, ...
     13536.710, 14029.458, 14540.103, 15069.295, 15617.710, ...
     16186.049, 16775.035, 17385.420, 18000.000 ];
```

2 Model ucha

2.1 FFTnorm.m

```
function F = FFTnorm(tn, fs, Lp)
% F = FFTnorm(tn, fs, Lp)
%
% Funkcja s³u¿y do wyliczenia widma amplitudowego sygna³u wejœciowego.
%
% Argumenty funkcji
% double tn[] - sygna³ wejœciowy (ramka 2048 próbek czasowych)
% double fs - czêstotliwoœæ próbkowania
% double Lp - poziom odtwarzania (SPL dla 1019.5Hz w dBFS)
%
% Funkcja zwraca
% double F[] - widmo amplitudowe sygna³u wejœciowego
%
% U¿ywane funkcje:
% abs   isempty     sin
% fft   max
% hann  norm
%
% U¿ywane zmienne:
% double fac[]      double Norm[]
% int N             double s[]
% double hw[]       double t[]
%
% Uwagi:
% Funkcja sk³adowa programu do pomiaru jakoœci dŸwiêku opartego
% o rekomendacjê ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

persistent hw

if isempty(hw)
    N = size(tn,2);
    Norm = norm(N, fs);
    fac = 10^(Lp/20)/Norm;
    hw = fac.*hann(N)';
end

F = abs(fft(hw.*tn));
F = F(1:1025);

function Norm = norm(N, fs)

fc = 1019.5;

t = 0:N*9;
s = sin(2*pi*fc*t/fs);
hw = hann(N);
Norm = 0;
for i = 1:10
```

```
    S = abs(fft(s(1024*(i-1)+1:1024*(i+1)).*hw'));
    Norm
```

2.2 OMEar.m

```
function Fe = OMEar(F, fs)
% Fe = OMEar(F, fs)
%
% Funkcja sluzy do na widmo wejsciowe charakterystyki
% czestotliwoœæiowej ucha zewnetrznego i srodkowego
%
% Argumenty funkcji
% double F[] - widmo wejsciowe
% double fs - czestotliwosc próbkowania
%
% Funkcja zwraca
% double Fe[] - widmo skorygowane charakterysyke ucha zewnetrznego
%               i srodkowego
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%
persistent W

if isempty(W)
    fn = size(F,2);
    f = linspace(0,fs/2,fn)/1000;
    W = -0.6*3.64*f.^(-0.8) + 6.5*exp(-0.6*(f-3.3).^2) -
0.001*f.^(3.6);
    W = 10.^(W/20);
end

Fe = W.*F;
```

2.3 CBGroup.m

```
function Pe = CBGroup(Fe, Z, fs, N)
% Pe = CBGroup(Fe, Z, fs, N)
%
% Funkcja sluzy do podzia³u widma Fouriera na pasma krytyczne
%
% Argumenty funkcji
% double Fe[] - widmo wejsciowe
% int Z - ilosc pasm krytycznych
% double fs - czêstotliwoœæ probkowania
% double N - polowa dlugosci ramki
%
% Funkcja zwraca
% double Pe[] - reprezentacja wysokosci dziwku
%
```

```
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

persistent fl fu
Fres = fs/(2*N);

if(isempty(fl))
    [~, fl fu] = CBands();
end

Fsp = Fe.^2;
Pe = zeros(1,109);
for z = 1:Z
    for k=0:1024
        if (k-0.5)*Fres >= fl(z) && (k+0.5)*Fres<=fu(z)
            Pe(z) = Pe(z) + Fsp(k+1);
        elseif (k-0.5)*Fres < fl(z) && (k+0.5)*Fres > fu(z)
            Pe(z) = Pe(z) + Fsp(k+1)*(fu(z)-fl(z))/Fres;
        elseif (k-0.5)*Fres < fl(z) && (k+0.5)*Fres > fl(z)
            Pe(z) = Pe(z) + Fsp(k+1)*((k+0.5)*Fres - fl(z))/Fres;
        elseif (k-0.5)*Fres < fu(z) && (k+0.5)*Fres > fu(z)
            Pe(z) = Pe(z) + Fsp(k+1)*((fu(z)-(k-0.5)*Fres))/Fres;
        end
    end
    Pe(z) = max(Pe(z),1e-12);
end
```

2.4 AddNoise.m

```
function Pp = AddNoise(Pe)
% Pp = AddNoise(Pe)
%
% Funkcja dodaje szum do widma w pasmach krytycznych
%
% Argumenty funkcji
% double Pe[] - widmo wejsciowe (w pasmach krytycznych)
%
% Funkcja zwraca
% double Pp[] - reprezentacja wysokosci dziwku
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%
```

```
persistent Pthres

if isempty(Pthres)
    fc = CBands();
    Pthres = 10.^((0.4*0.364*(fc./1000).^(-0.8)));
end

Pp = Pe + Pthres;
```

2.5 Spread.m

```
function E2 = Spread(Pp, Norm, Z, res)
% E2 = Spread(Pp, Norm, Z, res)
%
% Funkcja wygladza reprezentacje wysokosci w dziedzinie
% czestotliwosci
%
% Argumenty funkcji
% double Pp[] - reprezentacja wysokosci dzwieku
% double Norm[] - Wspolczynnik skalujacy
% int Z - wilosc pasm krytycznych
% double res - rozdzielczosc pasm
%
% Funkcja zwraca
% double E2[] - wzorzec pobudzenia niewygladzony czasowo
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

persistent fc

if isempty(fc)
    fc = CBands();
end

Su = zeros(1,Z);
E = zeros(1,Z);
Sln = 10^(-27 * res/10);
for j=1:Z
   Sun = 10^((-24-230/fc(j))*res/10) * Pp(j)^(0.2*res);
   Sld = (1-Sln^j)/(1-Sln);
   Sud = (1-Sun^(Z-(j-1)))/(1-Sun);
   El = Pp(j)/(Sld+Sud-1);
   Su(j) = Sun^0.4;
   E(j) = El^0.4;
end

Sl = Sln^0.4;
Eline = zeros(1,Z);
Eline(Z) = E(Z);
for j=Z-1:-1:1
```

```
    Eline(j) = Sl*Eline(j+1) + E(j);
end
for j=1:Z-1
    a = E(j);
    b = Su(j);
    for i=j+1:Z
        a = a * b;
        Eline(i) = Eline(i) + a;
    end
end
E2 = Eline.^(1/0.4)./Norm;
```

2.6 TimeSpread.m

```
function [E, Ef] = TimeSpread(E2, Ef)
% [E, Ef] = TimeSpread(E2, Ef)
%
% Funkcja wygladza reprezentacje wysokosci w dziedzinie
% czasu
%
% Argumenty funkcji
% double E2[] - wzorzec pobudzenia niewyglaczony czasowo
% double Ef[] - wartoœæ E2 z poprzedniej ramki
%
% Funkcja zwraca
% double E[] - wzorzec pobudzenia
% double Ef[] - wartoœæ E2 dla kolejnej ramki
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

persistent a

if isempty (a)
    fc = CBands();
    t100 = 0.030;
    tmin = 0.008;
    t = tmin + 100./fc*(t100-tmin);
    a = exp(-4/187.5./t);
end

Ef = a.*Ef + (1-a).*E2;
E = max(Ef,E2);
```

2.7 Mthres.m

```
function Ma = MThres(E)
% Ma = MThres(E)
%
```

```
% Funkcja wylicza prog maskowania
%
% Argumenty funkcji
% double E[] - wzorzec pobudzenia
%
% Funkcja zwraca
% double Ma[] - prog maskowania
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

persistent m

if isempty(m)
    m(1:49) = 10^(3/10);
    m(50:109) = 10.^(0.25.*0.25.*(49:108)./10);
end

Ma = E./m;
```

3 Przetwarzanie wstępne

3.1 LPAdapt.m

```
function [EP, P, Rn, Rd, PC] = LPAdapt(E, Z, P, Rn, Rd, PC)
% [Mod, ERavg, E2b, Eder, E] = Modulation(E2, E2b, Eder, E)
%
% Funkcja dostosowuje poziom glosnosci i wzorca
%
% Argumenty funkcji
% double E[] - wzorzec pobudzenia
% int Z - ilosc pasm krytycznych
% double P[] - wartosc wzorca dla poprzedniej ramki
% double Rn[] - wartosc licznika dla poprzedniej ramki
% double Rd[] - wartosc mianownika dla poprzedniej ramki
% double PC[] - wzorzec korekcji dla poprzedniej ramki
%
% Funkcja zwraca
% double EP[] - wzorzec dopasowany widmowo
% double P[] - wartosc wzorca dla kolejnej ramki
% double Rn[] - wartosc licznika dla kolejnej ramki
% double Rd[] - wartosc mianownika dla kolejnej ramki
% double PC[] - wzorzec korekcji dla kolejnej ramki
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

%% Level and pattern adaptation

persistent a M1 M2

if isempty(a)
    M1 = 3;
    M2 = 4;
    fc = CBands();
    t100 = 0.050;
    tmin = 0.008;
    t = tmin + 100./fc*(t100-tmin);
    a = exp(-4/187.5./t);
end

%% Filtering
P(1,:) = a.*P(1,:) + (1-a).*E(1,:);
P(2,:) = a.*P(2,:) + (1-a).*E(2,:);

%% Level adaptation
LevCorr = (sum(sqrt(P(2,:) .* P(1,:))) ./ sum(P(2,:)))^2;

EP = zeros (2, Z);
R = zeros (2, Z);
for z = 1:Z
```

```
    if (LevCorr > 1)
        EP(1,z) = E(1,z) / LevCorr;
        EP(2,z) = E(2,z);
    else
        EP(1,z) = E(1,z);
        EP(2,z) = E(2,z) * LevCorr;
    end
end

%% Pattern adaptation
Rn = a.*Rn + EP(2,:).*EP(1,:);
Rd = a.*Rd + EP(1,:).*EP(1,:);

for z = 1:Z
    if Rd(z) < 0 || Rn(z) < 0
        error ('>>> PQadap: Rd or Rn is < 0');
    end
    if Rd(z) == 0 && Rn(z) > 0
        R(1,z) = 1;
        R(2,z) = 0;
    elseif Rd(z) == 0 && Rn(z) == 0
        if z > 1
            R(1,z) = R(1,z-1);
            R(2,z) = R(2,z-1);
        else
            R(1,z) = 1;
            R(2,z) = 1;
        end
    elseif Rn(z) >= Rd(z)
        R(1,z) = 1;
        R(2,z) = Rd(z) / Rn(z);
    else
        R(1,z) = Rn(z) / Rd(z);
        R(2,z) = 1;
    end
end

for z = 1:Z
    m1 = max(z - M1, 1);
    m2 = min(z + M2, Z);
    s1 = sum(R(1,m1:m2));
    s2 = sum(R(2,m1:m2));
    PC(1,z) = a(z) * PC(1,z) + (1-a(z)) * s1 / (m2-m1+1);
    PC(2,z) = a(z) * PC(2,z) + (1-a(z)) * s2 / (m2-m1+1);
end

EP = EP.*PC;
```

3.2 Modulation.m

```
function [Mod, ERavg, E2b, Eder, E] = Modulation(E2, E2b, Eder, E)
% [EP, P, Rn, Rd, PC] = LPAdapt(E, Z, P, Rn, Rd, PC)
%
% Funkcja wylicza wartosc modulacji obwiedni
%
% Argumenty funkcji
% double E2[] - wzorzec pobudzenia niewygladzony czasowo
% double E2b[] - wartosc wzorca dla poprzedniej ramki
% double Eder[] - wartosc posrednia dla poprzedniej ramki
```

```
% double E[] - wartosc posrednia dla poprzedniej ramki
%
% Funkcja zwraca
% double Mod[] - wzorzec modulacji obwiedni
% double E2b[] - wartosc wzorca dla nastepnej ramki
% double Eder[] - wartosc posrednia dla nastepnej ramki
% double E[] - wartosc posrednia dla nastepnej ramki
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

persistent a

if isempty(a)
    fc = CBands();
    t100 = 0.050;
    tmin = 0.008;
    t = tmin + 100./fc*(t100-tmin);
    a = exp(-4/187.5./t);
end

Ed = 48000/1024 .* abs(E2.^0.3-E2b);
Eder(1,:) = a.*Eder(1,:) + (1-a).*Ed(1,:);
Eder(2,:) = a.*Eder(2,:) + (1-a).*Ed(2,:);
E(1,:) = a.*E(1,:) + (1-a).*E2(1,:).^0.3;
E(2,:) = a.*E(2,:) + (1-a).*E2(2,:).^0.3;
E2b = E2.^0.3;
Mod = Eder ./ (1 + E./0.3);
ERavg = E(1,:);
```

3.3 Loud.m

```
function Ntot = Loud(E, Z)
% Ntot = Loud(E, Z)
%
% Funkcja wylicza wzorzec glosnosci
%
% Argumenty funkcji
% double E[] - wzorzec pobudzenia
% int Z = ilosc pasm krytycznych
%
% Funkcja zwraca
% double Ntot[] - oglolna glosnosc
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
```

```
% Ostatnia modyfikacja:
% 9.01.2012
%

persistent s Ethres Et

if isempty(s)
    fc = CBands();
    const = 1.07664;
    Ethres = 10.^(0.364*(fc./1000).^(-0.8));
    s = 10.^((-2 - 2.05.*atan(fc./4000) -
0.75.*atan((fc./1600).^2)) ./ 10);
    Et = const * (Ethres./(s.*1e4)).^0.23;
end

N = Et .* ((1 - s + s.*E./Ethres).^0.23 - 1);
Ntot = (24/Z) * sum(max(N, 0));
```

4 Parametry wyjściowe MOV

4.1 MOV_mod.m

```
function [Mt1 Mt2 Wt] = MOV_mod(Mod, ERavg, Z)
% [Mt1 Mt2 Wt] = MOV_mod(Mod, ERavg, Z)
%
% Funkcja wylicza roznice w modulacjach obwiedni
%
% Argumenty funkcji
% double Mod - wzorzec modulacji obwiedni
% double ERavg - wartosc z funkcji Modulation
% int Z = ilosc pasm krytycznych
%
% Funkcja zwraca
% double Mt1 - roznica modulacji obwiedni 1
% double Mt2 - roznica modulacji obwiedni 2
% double Wt - wspolczynnik wagowy
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

persistent Et

if isempty(Et)
    fc = CBands();
    Et = 10.^(0.3.*(0.4*0.364*(fc./1000).^(-0.8)));
end

% Parameters
negWt2 = 0.1;
offset1 = 1.0;
offset2 = 0.01;
levWt = 100;

s1 = 0;
s2 = 0;
for z=1:Z
    if (Mod(1,z) > Mod(2,z))
        num1 = Mod(1,z) - Mod(2,z);
        num2 = negWt2 * num1;
    else
        num1 = Mod(2,z) - Mod(1,z);
        num2 = num1;
    end
    MD1 = num1 / (offset1 + Mod(1,z));
    MD2 = num2 / (offset2 + Mod(1,z));
    s1 = s1 + MD1;
    s2 = s2 + MD2;
end
Mt1 = (100/Z)*s1;
```

```
Mt2 = (100/Z)*s2;
Wt = sum(ERavg./(ERavg + levWt.*Et));
```

4.2 Nloud.m

```
function NL = MOV_Nloud(Mod, EP, Z)
% NL = MOV_Nloud(Mod, EP, Z)
%
% Funkcja wylicza glosnosc zniekształcen
%
% Argumenty funkcji
% double Mod[] - wzorzec modulacji obwiedni
% double EP[] - wzorzec dopasowany widmowo
% int Z = ilosc pasm krytycznych
%
% Funkcja zwraca
% double NL - glosnosc zniekształcen
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

persistent Et

if isempty(Et)
    fc = CBands();
    Et = 10.^(0.4*0.364*(fc./1000).^(-0.8));
end

% Parameters
alpha = 1.5;
TF0 = 0.15;
S0 = 0.5;

sref  = TF0.*Mod(1,:) + S0;
stest = TF0.*Mod(2,:) + S0;
beta = exp(-alpha.*(EP(2,:)-EP(1,:)) ./ EP(1,:));
nom = max (stest.*EP(2,:) - sref.*EP(1,:), 0);
dnom = Et + sref.*EP(1,:) .* beta;

NL = 24/Z*sum((Et./stest).^0.23 .* ((1 + nom./dnom).^0.23 - 1));

NL = max(NL, 0);
```

4.3 MOV_Bwidth.m

```
function [BWRef BWTest] = MOV_Bwidth(F)
% [BWRef BWTest] = MOV_Bwidth(F)
%
% Funkcja wylicza szerokosc pasm sygnalow
```

```
%
% Argumenty funkcji
% double F[] - widmo z uwzglednieniem charakterystyki
%              ucha zewnetrznego i srodkowego
%
% Funkcja zwraca
% int BWRef - szerokosc pasma sygnalu referencyjnego
% int BWTest - szerokosc pasma sygnalu testowanego
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

F2 = F.^2;
Xth = F2(2,922);
Xth = max(Xth,max(F2(2,922:1024)));

BWRef = 0;
for k = 921:-1:348
    if F2(1,k) >= 10*Xth
        BWRef = k;
        break;
    end
end

BWTest = 0;
for k = BWRef:-1:1
    if F2(2,k) >= 10^0.5*Xth
        BWTest = k;
        break;
    end
end
```

4.4 MOV_NMR.m

```
function [NMRavg NMRmax] = MOV_NMR(Pnoise, Ma, Z)
% [NMRavg NMRmax] = MOV_NMR(Pnoise, Ma)
%
% Funkcja wylicza stosunek zniekształcen do maski
%
% Argumenty funkcji
% double Pnoise[] - szum
% double Ma[] - prog maskowania
% int Z - liczba pasm krytycznych
%
% Funkcja zwraca
% double NMRavg - stosunek NMR usredniony
% double NMRmax - stosunek NMR maksymalny
%
% Uwagi:
```

```
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

NMRm = Pnoise./Ma;
NMRavg = sum(NMRm)./Z;
NMRmax = max(NMRm);
```

4.5 MOV_Dprob.m

```
function [p q] = MOV_Dprob(E)
% [p q] = MOV_Dprob(E)
%
% Funkcja wylicza prawdopodobienstwo detekcji
%
% Argumenty funkcji
% double E[] - wzorzec pobudzenia
%
% Funkcja zwraca
% double p[] - prawdopodobienstwo wykrycia
% double q[] - liczba stopni powyzej progu detekcji
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

persistent c g d1 d2 bP bM

if isempty(c)
    c = [-0.198719 0.0550197 -0.00102438 5.05622e-6 9.01033e-11];
    d1 = 5.95072;
    d2 = 6.39468;
    g = 1.71332;
    bP = 4;
    bM = 6;
end

z = length (E);

p = zeros (1, z);
q = zeros (1, z);

for i = 1:z
    EdBR = 10 * log10 (E(1,i));
    EdBT = 10 * log10 (E(2,i));
    edB = EdBR - EdBT;
```

```
    if (edB > 0)
        L = 0.3 * EdBR + 0.7 * EdBT;
        b = bP;
    else
        L = EdBT;
        b = bM;
    end
    if (L > 0)
        s = d1*(d2/L)^g + c(5)*L^4 + c(4)*L^3 + c(3)*L^2 + c(2)*L +
c(1);
    else
        s = 1e30;
    end
    p(i) = 1 - 0.5^((edB / s)^b);        % Detection probability
    q(i) = abs (fix(edB)) / s;           % Steps above threshold
end
```

4.6 MOV_Eharm.m

```
function EHS = MOV_Eharm(tR, tT, F, N)
% EHS = MOV_Eharm(tR, tT, F, N)
%
% Funkcja wylicza strukture harmonicznych sygnalu bledu
%
% Argumenty funkcji
% double tR[] - ramka sygnalu referencyjnego
% double tR[] - ramka sygnalu testowanego
% double F[] - widmo
% int N - ilosc ramek
%
% Funkcja zwraca
% double EHS - struktura harmonicznych sygnalu bledu
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

F2 = F.^2;
persistent hw EnThr

if isempty(hw)
    hw = (1/256)*sqrt(8/3)*hann(256)';
    EnThr = 8000/(16^2/2)^2;
end

EnRef = sum(tR(N:end).^2);
EnTest = sum(tT(N:end).^2);

if EnRef<EnThr && EnTest<EnThr
    EHS = 0;
    return;
end
```

```
D = log(F2(2,:)./F2(1,:));

F0=D(1:512);
Ft=[zeros(1,255) F0(1:256)];
C = zeros(1,256);

for i=1:256
    Cn=F0(1:255+i)*Ft(257-i:end)';
    Cd=norm(F0(1:255+i))*norm(Ft(257-i:end));
    C(i) = Cn/Cd;
end

C = C-sum(C)/256;
C = abs(fft(hw.*C)).^2;

cb = C(1);
cm = 0;
for n=2:128
    if C(n)>cb
        if C(n)>cm
            cm = C(n);
        end
    end
end
EHS = cm;
```

5 Uśrednianie MOV i ocena jakości.

5.1 MOV_avg.m

```
function MOV = MOV_avg(MOVn)
% MOV = MOV_avg(MOVn)
%
% Funkcja usrednia w czasie i wzgledem kanalow
% parametry MOV
%
% Argumenty funkcji
% double MOVn[] - macierz z parametrami MOV
% dla wszystkich ramek
%
% Funkcja zwraca
% double MOV[] - wektor usrednionych parametrow MOV
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

fram_num = size(MOVn.NRef,2);
Cn = size(MOVn.BWRef,1);

% BWRef
[MOV(1) MOV(2)] = BW_avg(MOVn.BWRef, MOVn.BWTest, Cn);

% NMRB RDF
[MOV(3) MOV(11)] = NMR_avg(MOVn.NMRavg, MOVn.NMRmax, fram_num, Cn);

% WinModDiff1B, AvgModDiff1B, AvgModDiff2B
[MOV(4) MOV(7) MOV(8)] = MD_avg(MOVn.Mt1, MOVn.Mt2, MOVn.Wt, fram_num,
Cn);

% RmsNoiseLoudB
[MOV(9)] = NL_avg(MOVn.NRef, MOVn.NTest, MOVn.NL, fram_num, Cn);

% ADBB, MFPDB
[MOV(5) MOV(10)] = PD_avg(MOVn.p, MOVn.q, fram_num, Cn);

% EHSB
[MOV(6)] = EHSB_avg(MOVn.EHS, Cn);

% functions
function [BWRef_avg BWTest_avg] = BW_avg(BWRef, BWTest, Cn)

BWRef_avg = 0;
BWTest_avg = 0;

for i=1:Cn
    BWRef_n0 = nonzeros(BWRef(i,:));
    nr = size(BWRef_n0,1);
```

```
    BWTest_n0 = nonzeros(BWTest(i,:));
    nt = size(BWTest_n0,1);
    BWRef_avg = BWRef_avg + sum(BWRef_n0)/nr;
    BWTest_avg = BWTest_avg + sum(BWTest_n0)/nt;
end

BWRef_avg = BWRef_avg/Cn;
BWTest_avg = BWTest_avg/Cn;

function [TNMR RDF] = NMR_avg(NMRavg, NMRmax, fram_num, Cn)

TNMR = 0;
for i=1:Cn
    TNMR = TNMR + 10*log10(sum(NMRavg(i,:))/fram_num);
end
TNMR = TNMR/Cn;

Thr = 10^(1.5 / 10);
RDF = 0;

for i=1:Cn
    n=0;
    for j=1:fram_num
        if NMRmax(i,j) >= Thr;
            n = n+1;
        end
    end
    RDF = RDF + n/fram_num;
end
RDF = RDF/Cn;

function [Win1B Avg1B Avg2B] = MD_avg(Mt1B, Mt2B, Wt, fram_num, Cn)

del = ceil(0.5*48000/1024);
L = 4;
Win1B = 0;
for i=1:Cn
    S = 0;
    for j=del+L:fram_num
        K = 0;
        for k=0:L-1
            K = K + sqrt(Mt1B(i,j-k));
        end
        S = S + (K/L)^4;
    end
    Win1B = Win1B + sqrt(S/(fram_num-del-L+1));
end
Win1B = Win1B/Cn;

del = del+1;
Avg1B = 0;
for i=1:Cn
    Avg1B = Avg1B +
sum(Wt(i,del:end).*Mt1B(i,del:end))/sum(Wt(i,del:end));
end
Avg1B = Avg1B/Cn;

Avg2B = 0;
for i=1:Cn
    Avg2B = Avg2B +
```

```
sum(Wt(i,del:end).*Mt2B(i,del:end))/sum(Wt(i,del:end));
end
Avg2B = Avg2B/Cn;

function [RmsNLB] = NL_avg(NRef, NTest, NL, fram_num, Cn)

N50 = 3;
del = ceil(0.5*48000/1024)+1;

Thr = 0.1;
iT(1:2) = fram_num;
for i=1:Cn
    for j=1:fram_num
        if NRef(i,j) > Thr && NTest(i,j) > Thr
            iT(i) = j;
            break
        end
    end
end
IT = min(iT);

del = max(IT+N50, del);

RmsNLB = 0;
for i=1:Cn
    RmsNLB = RmsNLB + sqrt(sum(NL(i,del:end).^2)./(fram_num-del+1));
end
RmsNLB = RmsNLB/Cn;

function [ADBB MFPDB] = PD_avg(p, q, fram_num, Cn)

if Cn == 2
    pbin = max(p(1,:,:), p(2,:,:));
    qbin = max(q(1,:,:), q(2,:,:));
else
    pbin = p;
    qbin = q;
end
Pc = 1 - prod(1 - pbin,3);
Qc = sum(qbin,3);

c0 = 0.9;
c1 = 1;
Phc = 0;
PMc = 0;
Qsum = 0;
nd = 0;

for i=1:fram_num
    Phc = c0 * Phc + (1 - c0) * Pc(i);
    PMc = max (PMc * c1, Phc);

    if Pc(i) > 0.5
        nd = nd + 1;
        Qsum = Qsum + Qc(i);
    end
end

if (nd == 0)
    ADBB = 0;
```

```
elseif (Qsum > 0)
    ADBB = log10 (Qsum / nd);
else
    ADBB = -0.5;
end

MFPDB = PMc;

function [EHSB] = EHSB_avg(EHS, Cn)

EHSB = 0;
for i = 1:Cn
    EHSi = nonzeros(EHS(i,:));
    ni = size(EHSi,1);
    EHSB = EHSB + sum(EHSi)/ni;
end
EHSB = 1000*EHSB/Cn;
```

5.2 ArtNet.m

```
function [ODG DI MOVs] = ArtNet(MOV)
% MOV = MOV_avg(MOVn)
%
% Funkcja oblicza ODG oraz DI
%
% Argumenty funkcji
% double MOV[] - usrednione parametry MOV
%
% Funkcja zwraca
% double ODG - obiektywny stopien roznicy ODG
% double DI - wskaznik zniekształcen
% double MOVs[] - wyskalowane wartosci MOV
%
% Uwagi:
% Funkcja skladowa programu do pomiaru jakosci dzwieku opartego
% o rekomendacje ITU-R BS.1387-1
%
% Autor:
% Krzysztof Czaja <czaja.kf@gmail.com>
%
% Ostatnia modyfikacja:
% 9.01.2012
%

amin = [ 393.916656   361.965332    -24.045116...
         1.110661    -0.206623       0.074318...
         1.113683     0.950345       0.029985...
         0.000101     0];

amax = [ 921          881.131226     16.212030...
         107.137772   2.886017       13.933351...
         63.257874    1145.018555    14.819740...
         1            1];

wx = [  -0.502657,     0.436333,     1.219602;...
         4.307481,     3.246017,     1.123743;...
         4.984241,    -2.211189,    -0.192096;...
         0.051056,    -1.762424,     4.331315;...
         2.321580,     1.789971,    -0.754560;...
```

```
        -5.303901,    -3.452257,    -10.814982;...
         2.730991,    -6.111805,     1.519223;...
         0.624950,    -1.331523,    -5.955151;...
         3.102889,     0.871260,    -5.922878;...
        -1.051468,    -0.939882,    -0.142913;...
        -1.804679,    -0.503610,    -0.620456];

wxb = [ -2.518254,     0.654841,    -2.207228 ];

wy = [  -3.817048,     4.107138,     4.629582 ];

wyb =   -0.307594;

bmin = -3.98;
bmax = 0.22;

MOVs = ((MOV-amin)./(amax-amin))';

s = wxb + sum(bsxfun(@times, wx, MOVs));

DI = wyb + sum(wy .* (1./(1+exp(-s))));

ODG = bmin + (bmax-bmin) * (1./(1+exp(-DI)));
```