Akademia Górniczo-Hutnicza
im. Stanisława Staszica
w Krakowie

Praca magisterska

Optymalizacja kodera mowy standardu LPC-10

Grzegorz Suder

Kierunek: Elektrotechnika
Specjalność: Inżynieria komputerowa w przemyśle

Nr albumu: 111848

Promotor
dr inż. Jarosław Bułat

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki

Kraków 2006

Oświadczenie autora

Ja, niżej podpisany Grzegorz Suder oświadczam, że praca ta została napisana samodzielnie i wykorzystywała (poza zdobytą na studiach wiedzą) jedynie wyniki prac zamieszczonych w spisie literatury.

.....
(Podpis autora)

Oświadczenie promotora

Oświadczam, że praca spełnia wymogi stawiane pracom magisterskim.

.....
(Podpis promotora)

Spis treści

Wstęp

Tematem przedstawionej pracy magisterskiej, jest optymalizacja kodera mowy standardu LPC–10 oraz wyznaczenie najlepszej książki kodowej dla kwantyzatora wektorowego współczynników LSP filtra traktu głosowego.

Gwałtowny rozwój technologii w ostatnich latach związanych z transmisją głosu takich jak komunikacja radiowa, sieci komórkowe czy też zdobywające coraz większą popularność rozmowy głosowe transmitowane za pomocą internetu (VoIP, ang. *Voice over Internet Protocol*), powodują wzrost wymagań co do redukcji przepływności bitowej strumienia danych skompresowanej mowy przy zachowaniu dobrej jakości przesyłanego dźwięku. Kompresji sygnału mowy poświęconych zostało wiele publikacji [5, 11, 16, 17], jednak w większości przypadków nie dotyczą one mowy polskiej. Brak jest również badań jakościowych przeprowadzonych pod tym kątem dla koderów o niskich przepływnościach bitowych jak i wygodnych w użyciu programowych implementacji algorytmu przetwarzania i kompresji ludzkiej mowy dla celów dydaktycznych.

Kodery mowy takie jak HVXC czy G.729, w znacznym stopniu bazują na algorytmie analizy–syntezy będącym jednym z punktów przedstawionej pracy. Podstawą leżącą u tej idei kompresji mowy jest zamodelowanie ludzkiego układu mowy, obejmującego szereg narządów. Bogata wiedza na temat wytwarzania mowy i jej percepcji, pozwala wyznaczyć szereg parametrów najistotniejszych z punktu widzenia kompresji, np. współczynników filtra traktu głosowego czy częstotliwości tonu podstawowego. Kwantyzacja wektorowa będąca jednym z przedmiotów badań tej pracy, może posłużyć nie tylko do zmniejszenia przepływności bitowej strumienia danych skompresowanej mowy ale i również do rozpoznawania mowy co otwiera szereg możliwości zastosowań zmodyfikowanego kodera mowy wraz z wyznaczonymi książkami kodowymi.

Zaprezentowana praca składa się z dwóch części: opisu teoretycznego poruszanych zagadnień obejmującego dwa pierwsze rozdziały, oraz pracy autorskiej. W części teoretycznej opisano algorytm działania kodera mowy standardu LPC–10, opis przekształcenia współczynników filtra traktu głosowego do postaci liniowych par spektralnych (LSP) oraz opis kwantyzacji skalarnej i wektorowej. Szczególny nacisk został położony na wyprowadzenie współczynników oraz omówieniu różnych metod kwantyzacji wektorowych wraz z algorytmami przeszukiwania książek kodowych.

Część autorska obejmuje implementację kodera mowy standardu LPC–10 w języku C++ wraz z zaproponowanymi ulepszeniami, zgromadzenie próbek dźwiękowych stanowiących zbiór treningowy, zaimplementowanie różnych metod kwantyzacji współczynników LSP oraz wyznaczenie dla nich najlepszej pod względem minimalizacji błędu średniokwadratowego (MSE) książki kodowej. Ostatni rozdział poświęcony jest

badaniom jakości sygnału mowy poddanemu schematowi kompresji według schematu analiza-synteza oraz kwantyzacji różnymi metodami. Głównym źródłem informacji o prezentowanych tutaj algorytmach była literatura poświęcona metodom kompresji mowy, w szczególności pozycje [1, 2, 5].

Ostatnim celem prezentowanej pracy jest gotowa do wykorzystania implementacja kodera mowy bazującego na standardzie LPC–10 wraz z przygotowaną książką kodową zoptymalizowaną pod kątem mowy polskiej, która mogłaby znaleźć zastosowanie w dziedzinach związanych z przetwarzaniem i cyfrową transmisją sygnału mowy. Mam nadzieję, że wyznaczone książki kodowe oraz wyniki badań jakości mowy, przyczynią się do polepszenia jakości jak i stopnia kompresji mowy oraz stanowić będą użyteczny materiał dydaktyczny.

Rozdział 1

Parametryczne kodery mowy

1.1. Wprowadzenie do przetwarzania i kompresji mowy

Podstawowym pojęciem związanym z przetwarzaniem danych w procesie komunikacji przez cyfrowe media takie jak internet, jest pojęcie kompresji danych – czyli takie przetworzenie sygnału wejściowego, by zajmował on mniejszą ilość bitów. Generalnie można dokonać podziału algorytmów kompresji na:

- kompresję bezstratną (np. kodowanie arytmetyczne, kodowanie Huffmana),
- kompresję stratną (np. kwantyzacja skalarna i wektorowa, kodowanie podpasmowe).

W przypadku kodowania danych audio dla telekomunikacji, stosowane są przede wszystkim algorytmy oparte na kompresji stratnej. Głównym tego powodem jest fakt, że transmisja danych ograniczona jest zazwyczaj do określonego zakresu częstotliwości (przedział częstotliwości skupiający przekaz odpowiadający artykułowanym słowom mieści się w zakresie $f_v \in (300, 3500)$ Hz [6]), co pozwala na wyeliminowanie części zbędnych informacji nie wpływających na przekaz. Nawet w tym zakresie częstotliwości, sygnał mowy charakteryzuje się nadmiarowością.

Przedstawiony w tej pracy algorytm LPC–10 jest oparty na schemacie analiza–synteza [2, 3, 4, 16]. Oznacza to, że przesyłane parametry nie są bezpośrednią reprezentacją kolejnych próbek sygnału wejściowego, ale zawierają informację o tym, jak odbiornik ma postąpić by odtworzyć sygnał wyjściowy. Metoda ta wymaga stworzenia i opisania modelu generacji analizowanego sygnału jak najbliższego rzeczywistemu.

Algorytmy kompresji mowy oparte na takim podejściu charakteryzują się większą kompresją przy tej samej jakości niż w innych koderach, ponieważ część informacji w postaci modelu jest już umieszczona zarówno po stronie kodera jak i dekodera, a przesyłane są tylko niezbędne informacje, tj.: współczynniki filtra traktu głosowego i wzmocnienie, informacja o dźwięczności/bezdźwięczności oraz okres pobudzenia dla dźwięcznych fragmentów mowy. Algorytm LPC–10 wykorzystuje model ściśle związany z fizycznym mechanizmem generacji mowy przez człowieka – tzw. *vocoder* (koder mowy, ang. *VOice CODER*). Dzięki stworzeniu cyfrowego modelu generacji mowy,

można zmniejszyć przepływność bitową, nawet bardziej niż w przypadku kompresji MP3. Wadą tego rozwiązania jest brak możliwości kompresji danych niepasujących do modelu, np.: muzyka czy EKG.

Schemat generacji mowy ludzkiej jest w istocie złożonym i skomplikowanym procesem angażującym wiele mechanizmów różnego rodzaju. Złożoność i ilość tych procesów uniemożliwiają stworzenie idealnego generatora mowy. Dodatkowo, opierając się na znajomości percepcji dźwiękowej przez człowieka, można dokonać pewnych uproszczeń w modelu generacji mowy, które nie wpływają na zrozumiałość przekazu (przykładem może tu być wykorzystanie informacji z ograniczonego pasma częstotliwości w zakresie ok. $f \in (0, 4000)$ Hz) a pozwolą zmniejszyć złożoność i zapotrzebowanie na pasmo zarówno kodera jak i dekodera.

Jakość generowanej mowy w prezentowanym koderze standardu LPC–10 jest niska ze względu na znaczne uproszczenia modelu generacji mowy, dlatego ten sposób kompresji stanowi zazwyczaj tylko punkt wyjścia dla bardziej zaawansowanych koderów mowy. W celu poprawienia jakości syntetyzowanego dźwięku, zostało opracowanych wiele rozwiązań. Podstawowym czynnikiem wpływającym na naturalność i jakość dźwięku, jest modelowanie sygnału pobudzenia będącego falą dźwiękową kształtowaną w torze oddechowym, oparta na rozróżnianiu większej ilości stanów jakie może przyjąć. W standardzie LPC–10 wyróżnia się dwa rodzaje pobudzeń: losowe oraz periodyczne, natomiast w koderach mowy takich jak MPEG–4 HVXC sygnał pobudzenia jest bardziej złożony i oparty o analizę widmową.

Bardziej zaawansowane metody analizy i syntezy mowy oferują tzw. *kodery hybrydowe*, będące w rzeczywistości techniką pośrednią między wokoderami a koderami falowymi. Główną różnicą między nimi jest podejście do kompresji sygnału pobudzenia, który w rzeczywistości nie ogranicza się tylko do dwóch stanów – dźwięczności i bezdźwięczności, ale uwzględnia również stany pośrednie. Kodery hybrydowe mają kilka charakterystycznych cech [4]:

- obwiednia widma traktu głosowego sygnału mowy może być wyznaczona algorytmami znanymi z kodera LPC–10,
- nie ma konieczności wyznaczania informacji o tym czy dana ramka sygnału jest dźwięczna czy bezdźwięczna ani konieczności wyznaczania częstotliwości tonu podstawowego,
- model generacji pobudzenia nie ma wpływu na jakość syntetyzowanej mowy.

Tego rodzaju schemat kompresji zapewnia dobra jakość dla stosunkowo niskich przepływności bitowych. Kosztem tego jest zwiększone zapotrzebowanie na moc obliczeniowa oraz zwiększone opóźnienie syntezy sygnału mowy w stosunku do sygnału nie poddanego kompresji. Najbardziej znanymi koderami hybrydowymi, są kodery bazujące na metodzie liniowego wzbudzania predykcyjnego CELP [3, 4]. Metoda ta została opracowana w 1984 roku przez Atala i Schroedera i charakteryzuje się tym, że zamiast wyznaczać parametry takie jak: dźwięczność/bezdźwięczność (ang. *voiced–unvoiced*, VUV) czy częstotliwość tonu podstawowego korzysta z książek kodowych z różnymi wektorami wzbudzeń, generujących sygnał najlepiej pasujący do fragmentu mowy, który jest kodowany. Przykładem implementacji kodera CELP, jest standard federalny

Stanów Zjednoczonych FS 1016 [3, 16] oraz koder standardu ITU–T G.728 o bardzo małym opóźnieniu wynoszącym ok. 2 ms [3].

Kolejna grupę algorytmów kompresji sygnału mowy, stanowią *kodery falowe* dostarczające sygnał o małym opóźnieniu i dobrej jakości. Ze względu na te cechy znalazły one szerokie zastosowanie w standardach kodowania danych audio. Podstawowymi metodami kodowania koderów falowych, są algorytmy:

- PCM (ang. *Pulse Code Modulation*) korzystający z kwantyzatora skalarnego zarówno równomiernego jak i nierównomiernego (opartego na charakterystykach A–law i μ–law),
- ADPCM (ang. *Adaptive Differential PCM*) dokonującego predykcji wartości aktualnej próbki na podstawie wartości poprzedniej próbki i dokonującego kodowania różnicowego,
- ATC (ang. *Adaptive Transform Coding*) będący połączeniem kilku metod kompresji: kwantyzatora wektorowego i TDHS (*Time Domain Harmonic Scaling*).

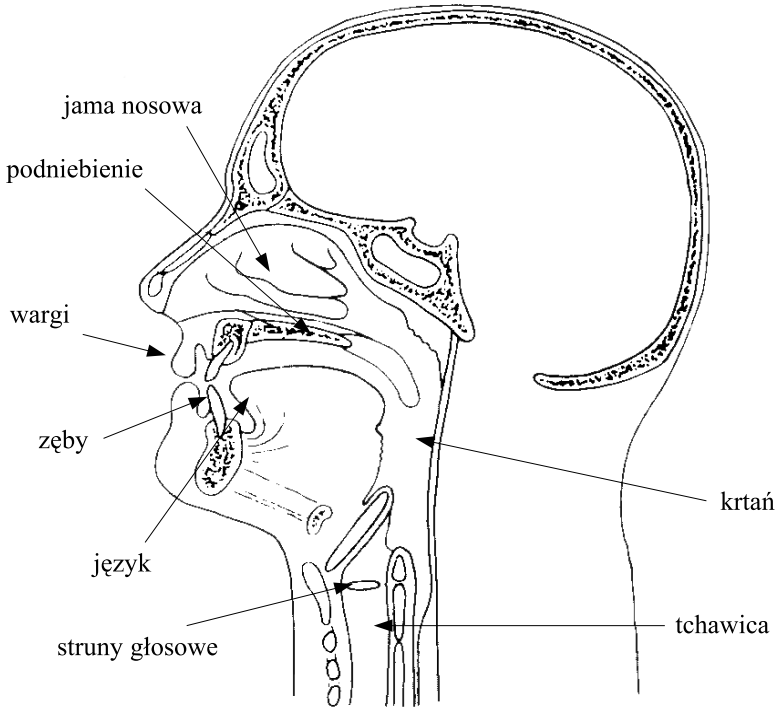
Podstawową wadą tego rodzaju kodowania jest stosunkowo wysoka przepływność bitowa, która w wielu przypadkach stanowi główne kryterium funkcjonalności kodowania. Jako przykład kodera falowego można przedstawić standard ITU–T G.711 oparty na algorytmie PCM z logarytmiczną funkcją kompresora o przepływności bitowej wynoszącej 64 kbit/s, oraz ITU–T G.721 oparty na algorytmie ADPCM 32 kbit/s. W tabeli 1.1 zostały zestawione podstawowe parametry najpowszechniej stosowanych koderów mowy.

lp.	Standard ITU–T	MOS	Przepływność bitowa [kbit/s]	MIPS	Rozmiar ramki [ms]
1	G.728 LD-CELP	4.0	16	30	0.625
2	G.711 PCM	4.3	64	0.2	0.125
3	G.722 ADPCM	4.2	32-64	7.8	0.125
4	FS-1016 CELP	3.7	4.8	19	30
5	FS-1015 LPC–10	2.5	2.4	20	22.5

Tabela 1.1. Zestawienie przykładowych koderów mowy wraz z najważniejszymi parametrami: subiektywną miarą jakości generowanego dźwięku (MOS), złożonością obliczeniową (MIPS) oraz wielkością przetwarzanej ramki.

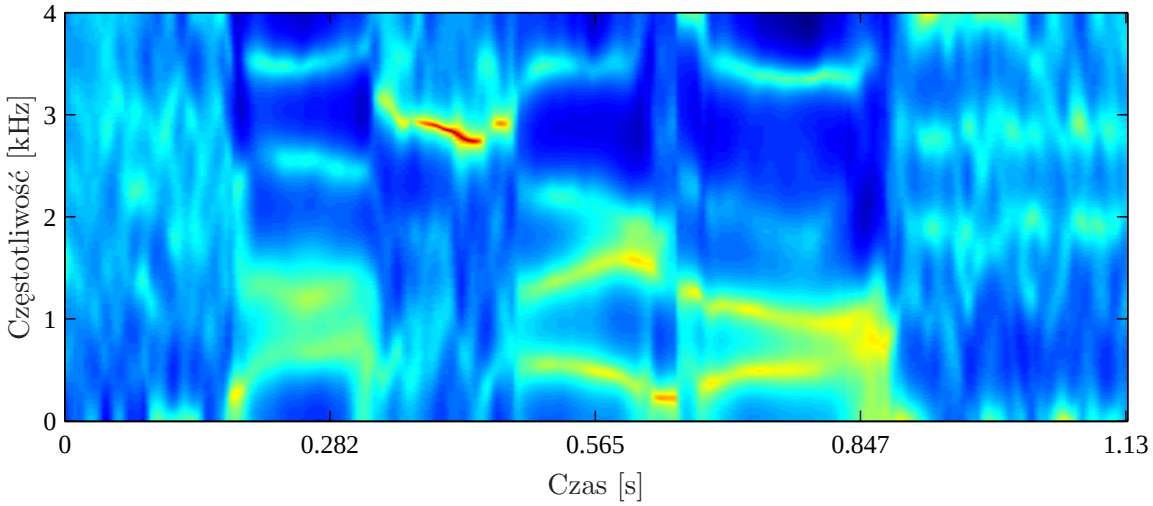
W procesie generacji ludzkiej mowy, biorą udział następujące elementy układy oddechowego: płuca, oskrzela i tchawica, krtań, struny głosowe (tzw. głośnia) i ostatecznie język, podniebienie, nos, zęby i wargi. Zasadniczo płuca, tchawica oraz część krtani poniżej strun głosowych spełniają tylko rolę źródła energii wytwarzając falę dźwiękową modulowaną w górnych warstwach układu głosowego. Wszystkie elementy znajdujące się powyżej tworzą *trakt głosowy* (będący rezonatorem) kształtujący falę dźwiękową generowaną przez struny głosowe. Powietrze przechodząc przez głośnie pobudza do drgania struny głosowe, co powoduje powstanie tzw. tonu podstawowego. (ang. *pitch*). W zależności od tego czy struny głosowe są otwarte na całej długości czy

też rytmicznie rozchylają się, formowana fala dźwiękowa ma postać bądź szumu bądź też ciągu impulsów. Zmiana częstotliwości tonu podstawowego jest ściśle związana z intonacją głosu. Największa zmienność występuje przy zakończeniu zdania (kropka, wykrzyknik, pytajnik). Dodatkowo na częstotliwość tonu podstawowego wpływają również takie czynniki jak płeć rozmówcy, jej wiek oraz cechy osobnicze. W zależności od tych parametrów częstotliwość tonu podstawowego może zmieniać się w zakresie 80–960 Hz [1, 6]. W przypadku kobiet zakres zmienności jest ok. dwa razy wyższy niż w przypadku mężczyzn i mieści się w zakresie ok. 160–960 Hz, podczas gdy dla mężczyzn jest to zakres ok. 80–480 Hz. Schematycznie trakt głosowy pokazany został na rysunku 1.1 z pominięciem elementów znajdujących się poniżej tchawicy, nie spełniających z punktu widzenia generacji mowy istotnej funkcji w procesie modelowania sygnału pobudzenia.



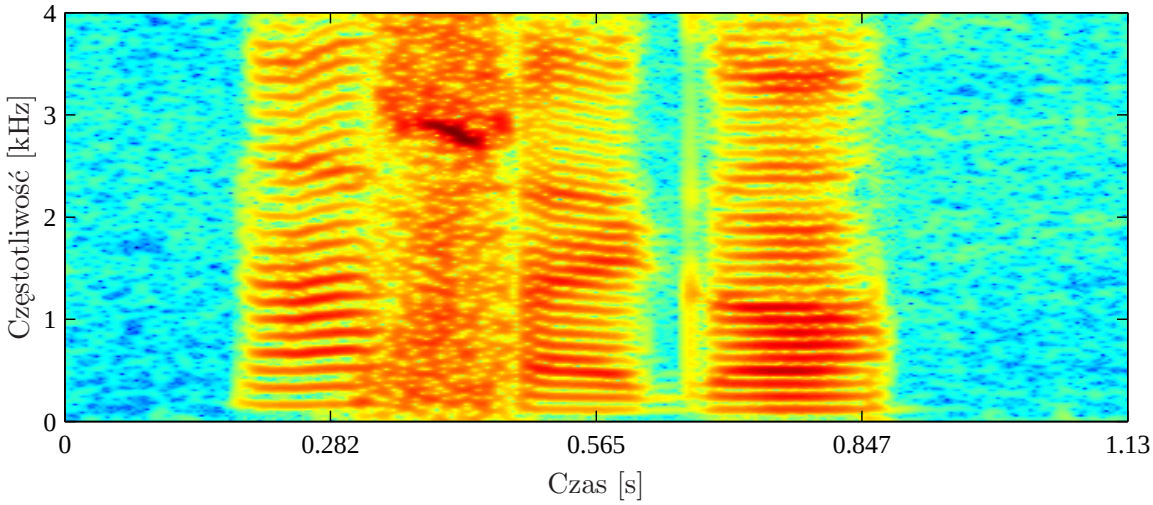
Rysunek 1.1. Schemat traktu głosowego.

Tor oddechowy kształtuje widmo sygnału krtaniowego powstającego przy przechodzeniu fali dźwiękowej przez struny głosowe. Dokonując estymacji parametrów opisujących trakt głosowy jako układ filtrujący sygnał pobudzenia, można wyznaczyć jego charakterystykę amplitudowo–częstotliwościową. Na rysunku 1.2, została pokazana zmiana tej charakterystyki w czasie dla kolejnych ramek mowy odpowiadających 20 ms, wyznaczona za pomocą funkcji `lpc(...)` i `freqz(...)` w programie `Matlab`.



Rysunek 1.2. Wykres czasowo–częstotliwościowy zmieniającej się charakterystyki amplitudowej traktu głosowego dla kolejnych ramek sygnału mowy na przykładzie słowa „naszego” wykonany przez autora.

Dla porównania na rysunku 1.3 została zaprezentowana krótkoczasowa transformata Fouriera tego samego sygnału wyznaczona funkcją `specgram(...)` programu `Matlab`:



Rysunek 1.3. Wykres modułu krótkoczasowej transformaty Fouriera słowa „naszego” wykonany przez autora.

Z rysunku 1.2 oraz 1.3 można zauważyć, że pomiędzy charakterystyką amplitudowo–częstotliwościową traktu głosowego a jego transformatą Fouriera istnieje

ścisły związek. Model generacji sygnału mowy można bowiem opisać jako filtrowanie pewnego sygnału pobudzającego filtr traktu głosowego. Najprostszy model generatora mowy składa się zatem z następujących elementów:

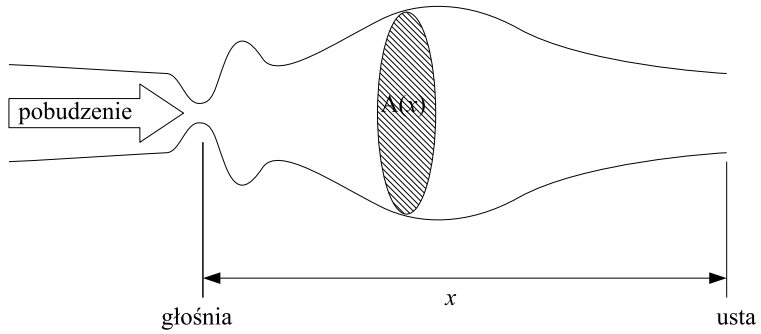
– sygnał pobudzenia, który w najprostszym przypadku można podzielić na dwa typy sygnałów: dla głosek dźwięcznych („a", „e", „i", „o", „u") i bezdźwięcznych („sz", „cz", „c", ...),
– układ rezonatorów (trakt głosowy) modelujący kształt widma sygnału pobudzenia.

Na rysunku 1.4 został przedstawiony schematycznie trakt głosowy. Rzeczywisty trakt głosowy dorosłego człowieka ma długość x około 16 cm, a przekrój $A(x)$ ma promień poniżej 2 cm [4]. Kształt fali dźwiękowej powstającej w trakcie głosowym opisany jest parą równań różniczkowych [11, 19]:

$$-\frac{\partial p(x,t)}{\partial x} = \rho \frac{\partial (u/A)}{\partial t} \tag{1.1}$$

$$-\frac{\partial u(x,t)}{\partial x} = \frac{1}{\rho c^2}\frac{\partial (p/A)}{\partial t} + \frac{\partial A}{\partial t} \tag{1.2}$$

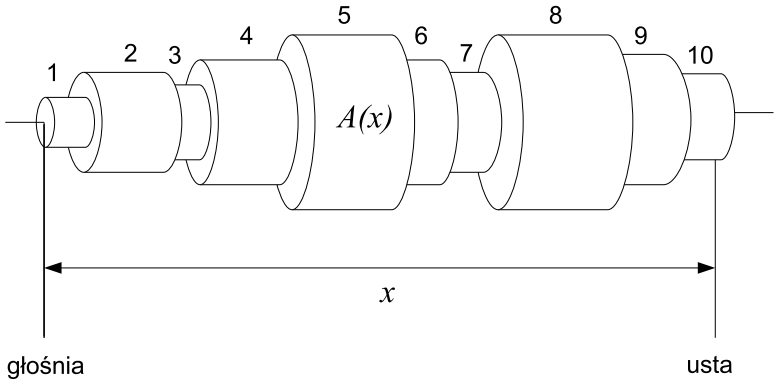
gdzie:
$p(x,t)$ – wariancja ciśnienia w tubie w chwili czasu t i w punkcie x,
$u(x,t)$ – wariancja przepływu w chwili czasu t i w punkcie x,
ρ – gęstość powietrza w tubie,
c – prędkość rozchodzenia się dźwięku,
$A(x,t)$ – funkcja przekroju poprzecznego tuby.



Rysunek 1.4. Schemat traktu głosowego.

Równania (1.1) i (1.2) rozwiązuje się wprowadzając szereg uproszczeń dotyczących traktu głosowego [4, 11]. Zazwyczaj dokonuje się podziału traktu głosowego na kilkanaście odcinków, które aproksymuje się tubami o długości Δx i przekroju kołowym o powierzchni $A(x,t)$ [4, 11]. W szczególności dla $\lim_{\Delta x \to 0}$ zdyskretyzowany model traktu głosowego będzie zgodny z założonym modelem rzeczywistego traktu głosowego. W

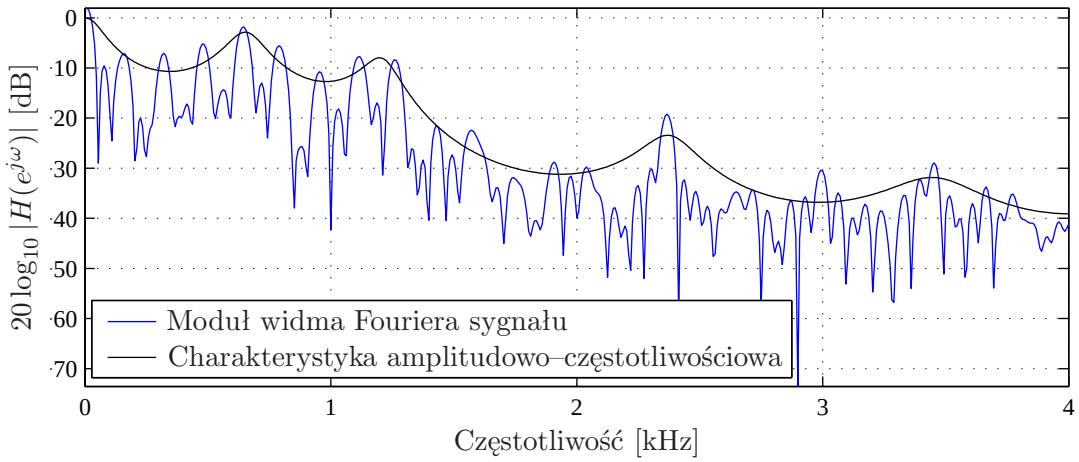
praktyce należy jednak przyjąć skończoną liczbę cylindrycznych przekrojów aproksymujących rzeczywisty trakt głosowy (rysunek 1.5). Tak przyjęty zdyskretyzowany model traktu głosowego wykazuje cechy wspólne z filtrami cyfrowymi [11].



Rysunek 1.5. Uproszczony schemat zdyskretyzowanego traktu głosowego.

Przyjmując, że dla dostatecznie krótkich fragmentów sygnału mowy (ramek mowy) trakt głosowy jest układem liniowo niezmiennym w czasie (ang. *Linear Time–Invariant*, LTI), można go opisać pewnym modelem transmitancji – filtrem IIR o k współczynnikach:

$$H(z) = \frac{G}{1 + \sum_{n=1}^{k} a_n z^{-n}} = \frac{G}{\prod_{n=1}^{k/2} (1 - p_n z^{-1})(1 - p_n^* z^{-1})} \tag{1.3}$$

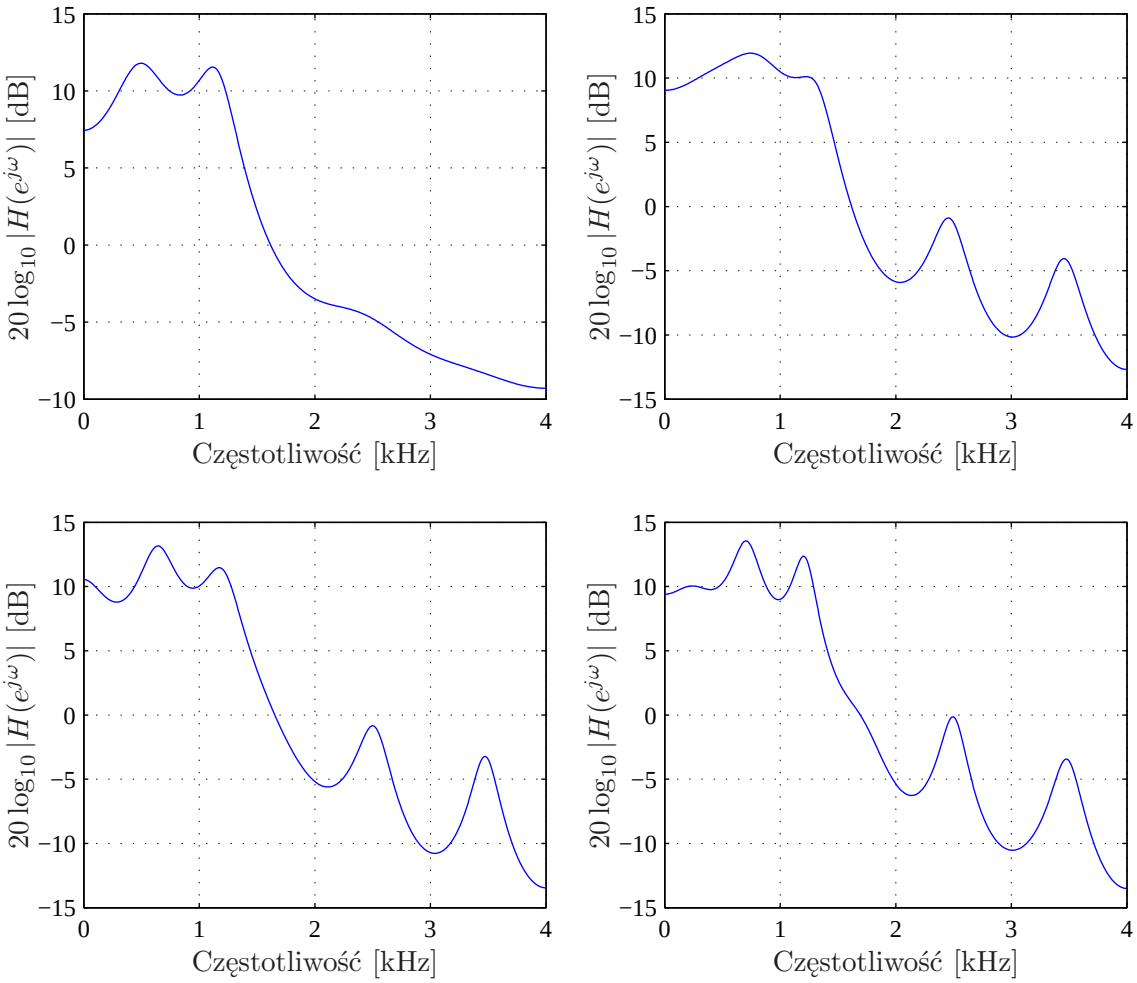


Rysunek 1.6. Porównanie modułu widma Fouriera sygnału z wyznaczoną na jego podstawie charakterystyką amplitudowo–częstotliwościową funkcją `lpc(...)`.

Na rysunku 1.6 została przedstawiona charakterystyka amplitudowo–częstotliwościowa powyższego filtra przy $k = 10$ na tle modułu widma Fouriera sygnału oryginalnego. Można zauważyć taki filtr oddaje dobrze charakterystykę sygnału nawet przy ograniczonej ilości współczynników.

W koderze standardu LPC–10, przyjmuje się ilość współczynników filtra traktu głosowego $k = 10$. Taka ilość jest wystarczająca do zachowania zrozumiałości przekazu. Zwiększenie tej wartości nie przynosi znacznego polepszenia jakości co zostało pokazane na rysunku 1.7. Dokładność estymacji współczynników filtra głosowego jest odpowiedzialna za zrozumiałość treści, jednak już samo uproszczenie modelu traktu głosowego wprowadza niedokładności spowodowane:

– brakiem płynnego przejścia pomiędzy kolejnymi przekrojami traktu głosowego,
– przekrojem odbiegającym od kołowego rzeczywistego traktu głosowego,
– brakiem uwzględnienia sprężystości ścian traktu głosowego.

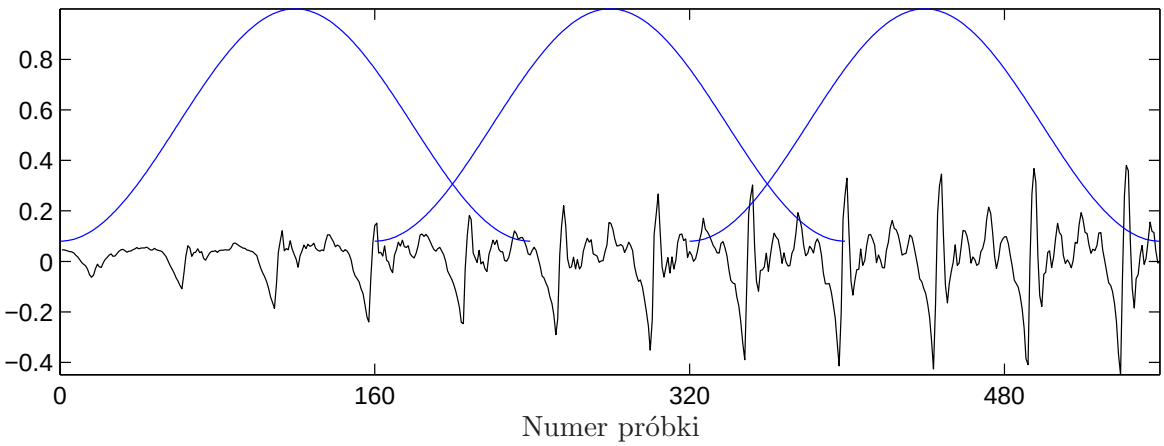


Rysunek 1.7. Wpływ ilości współczynników filtra traktu głosowego na kształt obwiedni widma tego filtra. Od lewej: 8 biegunów, 10 biegunów, 12 biegunów oraz 14 biegunów.

Wszystkie te czynniki wpływają na dokładność modelu i jakość dźwięku. Na rysunku 1.7 został przedstawiony wpływ ilości biegunów na kształt obwiedni widma traktu głosowego. Można z niego zauważyć, że dla 10, 12 i 14 biegunów wykresy są do siebie zbliżone. Dalsze zwiększanie ilości biegunów nie zmieni znacząco wykresów w stosunku dla ilości biegunów równej 10.

1.2. Koder mowy standardu LPC–10

Standard LPC–10 operuje na danych spróbkowanych z częstotliwością $f_{pr} = 8000$ Hz. W celu wykonania analizy, sygnał jest dzielony na fragmenty o długości 20 ms (tzw. ramki mowy). W tym celu sygnał wejściowy wymnażany jest z oknem Hamminga o długości 240 próbek, które jest przesuwane co 160 próbek (rysunek 1.8). Zatem w każdej ramce analizowanych jest 160 nowych próbek sygnału mowy i 80 próbek z poprzedniej ramki. Na podstawie tych próbek wyznaczane są parametry, które zostaną następnie przesłane do dekodera. Przy ramkach o długości 20 ms, parametry kodera LPC–10 wyznaczane są 50 razy na sekundę. Czas przetwarzania sygnału musi być zatem odpowiednio krótki by nie wprowadzać dodatkowych opóźnień ani nie pomijać ramek w procesie przetwarzania. Bardziej zaawansowane algorytmy np. wykrywania i usuwania nieciągłości informacji o tym czy ramka jest dźwięczna czy bezdźwięczna, aproksymacji współczynników filtra traktu głosowego pomiędzy ramkami, zwiększają dodatkowo opóźnienie kodera. Oprócz tego, należy jeszcze doliczyć opóźnienia spowodowane obecnością urządzeń pomiędzy koderem a dekoderem np. routery w sieci. W wielu przypadkach sumaryczne opóźnienie może być zbyt duże do zaakceptowania. Efektem jest utrudnienie bądź uniemożliwienie komunikacji głosowej pomiędzy nadawcą a odbiorcą.



Rysunek 1.8. Ilustracja segmentacji sygnału czasowego. Okno czasowe o długości 240 próbek i przesuwane o 160 próbek, jest wymnażane z sygnałem mowy dla danej ramki.

Standard algorytmu LPC–10 wymaga by wśród przesyłanych parametrów były współczynniki filtra traktu głosowego. Ponieważ filtr narządu mowy jest wrażliwy na

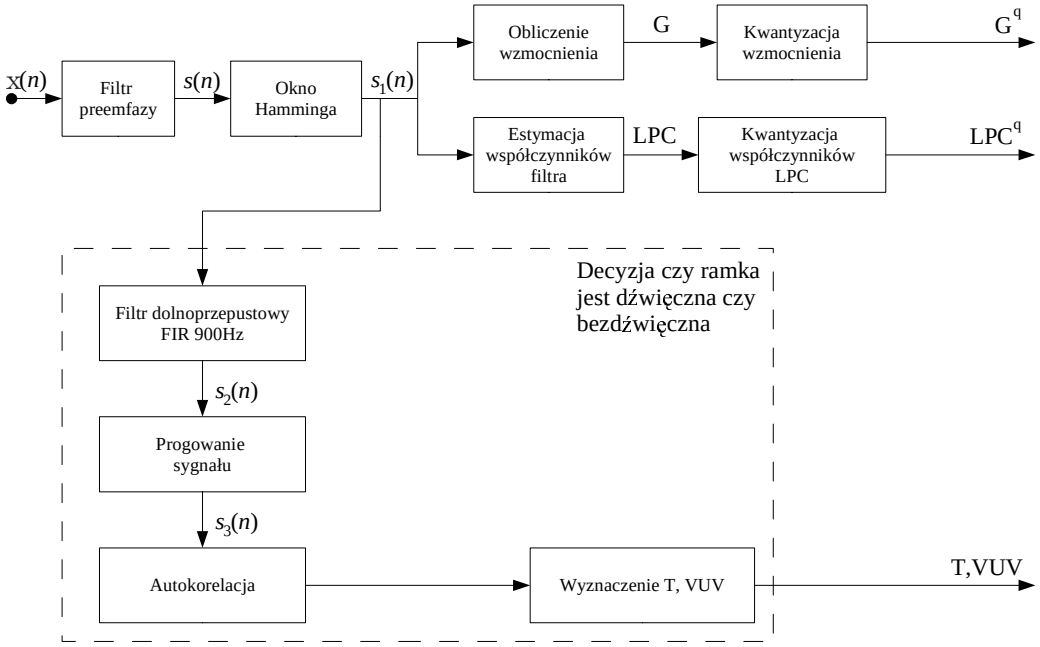
błędy kwantyzacji, współczynniki filtra przed zakodowaniem są przekształcane do innej postaci. Standard LPC–10 [3] określa by dokonać przekształcenia współczynników a_i zgodnie ze wzorem:

$$\gamma_i = \frac{1 + a_i}{1 - a_i} \tag{1.4}$$

Tak przekształcone współczynniki są kwantowane równomiernym kwantyzatorem skalarnym o wielkości książki kodowej zależnej od numeru współczynnika. Jak podaje [3], dla ramki dźwięcznej współczynniki $\gamma_1, \ldots, \gamma_4$ są kwantowane 5–bitowym kwantyzatorem, $\gamma_5, \ldots, \gamma_8$ 4–bitowym, γ_9 3–bitowym a γ_{10} 2–bitowym kwantyzatorem skalarnym. Dla ramki bezdźwięcznej kwantowane są tylko współczynniki $\gamma_1, \ldots, \gamma_4$, natomiast pozostałych 21 bitów wykorzystywanych jest do ochrony przed błędami. W tej pracy została zaimplementowana bardziej efektywna metoda przekształcania współczynników LPC do postaci LSP omówiona w następnym podrozdziale.

W niniejszej pracy magisterskiej, został zaimplementowany koder i dekoder mowy zgodny ze standardem LPC–10. Następnie wprowadzono pewne ulepszenia mające na celu poprawę jakości mowy kosztem nieznacznego zwiększenia przepływności bitowej. Poniżej został przedstawiony schemat blokowy kodera standardu LPC–10 w wersji podstawowej wraz z omówionymi poszczególnymi elementami.

1.2.1. Koder



Rysunek 1.9. Schemat blokowy kodera LPC–10.

W rzeczywistym trakcie głosowym, wielokrotności tonu podstawowego są tłumione z nachyleniem ok. 12 dB/oktawę przez rezonatory układu oddechowego [6]. W celu kompensacji tłumienia wyższych częstotliwości, na wejściu kodera znajduje się filtr preemfazy opisany równaniem czasowym:

$$s(n) = x(n) - 0.9375x(n-1) \tag{1.5}$$

gdzie $x(n)$ jest sygnałem wejściowym. Tak przefiltrowany sygnał wymnażany jest z oknem Hamminga o długości 240 próbek, a na podstawie otrzymanego fragmentu sygnału wyznaczane są współczynniki filtra traktu głosowego, jego wzmocnienie oraz wysokość tonu podstawowego.

Współczynniki i wzmocnienie filtra wyznaczane są przy wykorzystaniu modelu autoregresywnego (AR). Punktem wyjścia jest równanie 1.6, będące zmodyfikowanym równaniem Youle'a–Walkera [1, 3, 14]:

$$\begin{bmatrix} r(0) & r(1) & \dots & r(p-1) \\ r(1) & r(0) & \dots & r(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(p-1) & r(p-2) & \dots & r(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(p) \end{bmatrix} \tag{1.6}$$

gdzie $r(i)$ dla $i = 0, \dots, p$ jest estymatą funkcji autokorelacji sprogowanego sygnału określoną równaniem (1.9), p jest rzędem filtra (w koderze mowy standardu LPC–10 $p = 10$) natomiast $a_1, \dots, a_p$ są poszukiwanymi współczynnikami. Wzmocnienie filtra oblicza się ze wzoru:

$$G = \gamma\sqrt{\sigma^2_{min}} \tag{1.7}$$

gdzie γ jest odwrotnością pierwiastka współczynnika korekcyjnego dla okna czasowego. Wartości współczynników korygujących są stabelaryzowane. Stosowne dane można znaleźć np. w [7]. Dla okna Hamminga współczynnik korekcyjny wynosi 0.387, czemu odpowiada wartość $\gamma = 1.607$. Natomiast σ^2_{min} w (1.7) jest błędem predykcji obliczanym zgodnie z:

$$J_{min} = \sigma^2_{min} = r(0) + \sum_{j=1}^{p} a_j r(j), \tag{1.8}$$

$$r(k) = \frac{1}{N-p} \sum_{n=p}^{N-1} s_3(n) s_3(n+k) \tag{1.9}$$

gdzie $k = 0, \dots, N-1$ a $N = 240$ jest ilością analizowanych próbek.

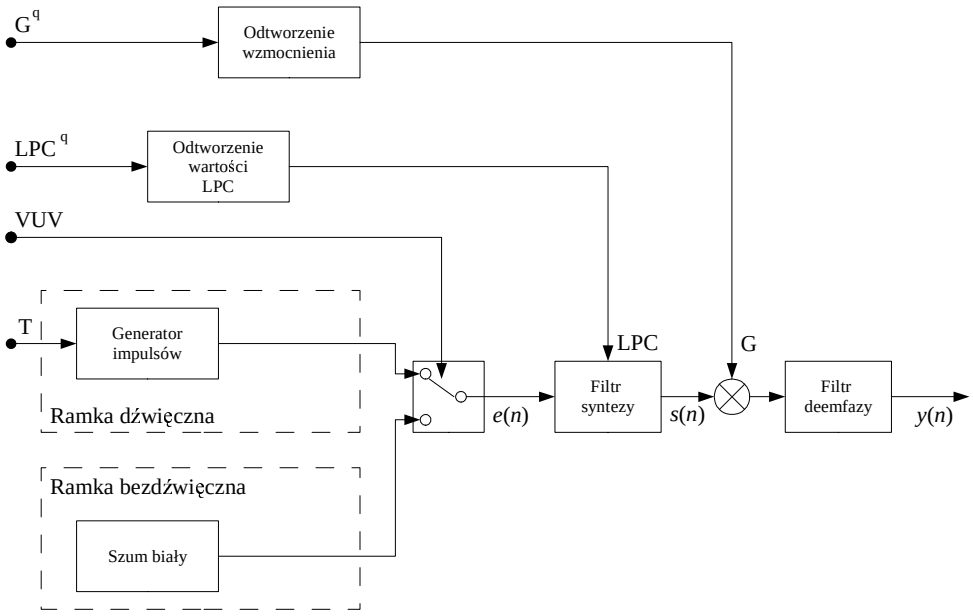
Wyznaczenie współczynników metodą bezpośrednią jest czasochłonnym zadaniem. W celu ich szybkiego obliczenia, wykorzystuje się algorytm Durbina–Levinsona [1, 3, 2, 4, 16] bądź algorytm Leroux–Gueguen [5, 16]. Drugi z algorytmów jest dedykowany dla procesorów stałoprzecinkowych w przypadku których daje lepsze wyniki niż algorytm Durbina–Levinsona. Jest on jednak mniej efektywny pod względem czasowym ze względu na konieczność przejścia z pośredniej postaci tzw. współczynników odbicia do postaci LPC (RS–to–LPC).

Wyznaczenie dźwięczności/bezdźwięczności, jest dokonywane przy pomocy metody autokorelacji. W celu lepszego uwydatnienia interesujących własności sygnału, przed wyznaczeniem częstotliwości tonu podstawowego sygnał jest filtrowany filtrem dolnoprzepustowym ograniczającym jego pasmo do ok. 900 Hz. Wyznaczając funkcję autokorelacji dla przefiltrowanego sygnału, a następnie wyszukując maksimum tej funkcji leżące w zakresie $T \in [20; 160]$ próbek (co odpowiada zakresowi częstotliwości tonu podstawowego wynoszącego 50–400 Hz) można dokonać oszacowania rodzaju pobudzenia. Jako próg, powyżej którego znalezione maksimum kwalifikuje się jako głoska dźwięczna, przyjmuje się wartość $(0.3 - 0.35)r(0)$, gdzie $r(0)$ jest wyznaczone z (1.9). Poniżej tej wartości ramka sygnału uznawana jest za bezdźwięczną. W celu poprawienia właściwości dyskryminacyjnych, przed wykonaniem autokorelacji wykonywane jest progowanie sygnału, czyli wykonanie operacji:

$$s_3(n) = \begin{cases} s_2(n) - P & \text{dla } s_2(n) \geqslant P \\ s_2(n) + P & \text{dla } s_2(n) \leqslant -P \\ 0 & \text{dla } s_2(n) \in (-P, P) \end{cases} \tag{1.10}$$

gdzie P jest wartością progową wyznaczoną na podstawie funkcji autokorelacji sygnału $s_2(n)$ [1, 16]. Ostatnim krokiem jest dokonanie kwantyzacji współczynników filtra i jego wzmocnienia (LPC^q,G^q). Kwantyzacja wspomnianych parametrów zostanie omówiona w kolejnych rozdziałach.

1.2.2. Dekoder



Rysunek 1.10. Schemat blokowy dekodera LPC–10.

Dekoder w pierwszej kolejności dokonuje operacji odwrotnej do kwantyzacji otrzymanych parametrów: LPC^q oraz G^q. W następnym kroku na podstawie informacji o dźwięczności i bezdźwięczności generowane jest pobudzenie. W pierwszym przypadku generator impulsów tworzy szereg impulsów Diraca o wysokości równej 1 w odstępach równych okresowi T, które stanowią pobudzenie dźwięczne. W modyfikacji algorytmu w celu poprawienia naturalności pobudzenia są one splatane z odpowiedzią impulsową pewnego filtra wyznaczoną w koderze co zostało opisane w rozdziale 3. Natomiast w przypadku pobudzenia bezdźwięcznego, generowany jest szum gaussowski.

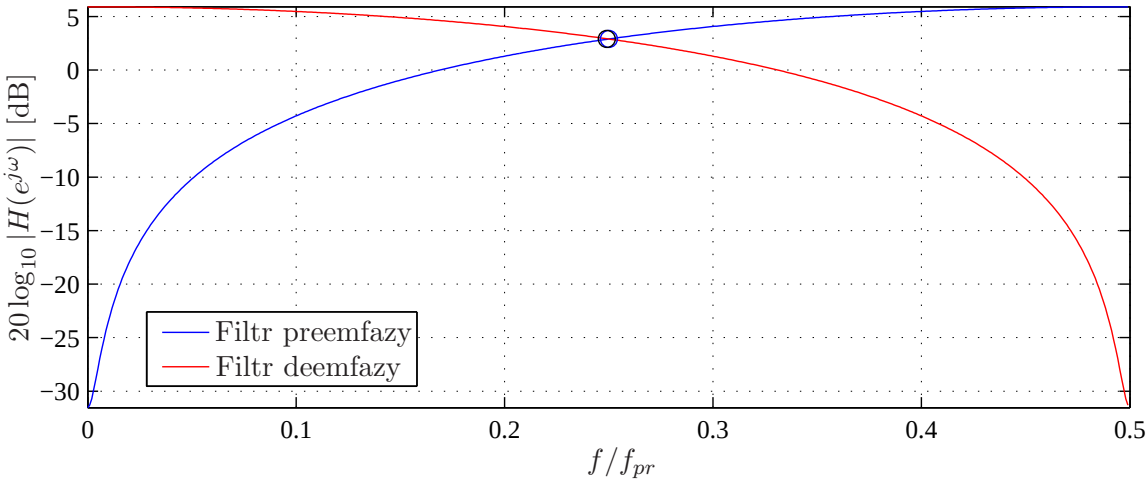
Wygenerowany sygnał pobudzenia jest filtrowany filtrem syntezy o odtworzonych 10 współczynnikach i wzmocnieniu G. Samo odtwarzanie współczynników LPC jest operacją odwrotną do wyznaczania współczynników LSP w koderze. Filtr syntezy o transmitancji (1.3) opisany jest równaniem czasowym:

$$s(n) = Ge(n) - \sum_{k=1}^{10} a_k s(n-k) \tag{1.11}$$

gdzie G jest wzmocnieniem filtra, a_k są jego współczynnikami wyznaczonymi np. algorytmem Durbina–Levinsona, natomiast $e(n)$ jest sygnałem pobudzenia. Tak zsyntetyzowany sygnał jest poddawany operacji filtracji filtrem deemfazy (rysunek 1.11) opisanym równaniem czasowym:

$$y(n) = s(n) + 0.9375y(n-1) \tag{1.12}$$

Filtr ten jest filtrem odwrotnym do filtra preemfazy i ma na celu stłumienie wyższych częstotliwości. Sygnał na wyjściu filtra jest deemfaazy jest odtworzonym sygnałem mowy o jakości zależnej od stopnia kwantyzacji współczynników i rozbudowania algorytmu LPC–10 (dokładności modelu).



Rysunek 1.11. Charakterystyki amplitudowo–częstotliwościowe filtra preemfazy oraz deemfazy, gdzie f_{pr} jest częstotliwością próbkowania wynoszącą 8 kHz.

Na rysunku 1.11 zostały przedstawione charakterystyki amplitudowo–częstotliwościowe filtrów preemfazy i deemfazy, o długości dwóch próbek. Widać, że częstotliwości odcięcia obydwu filtrów są zbliżone i znajdują się w okolicach $f = 1000$ Hz.

1.3. Algorytm wyznaczania współczynników LSP

Kwantyzacja współczynników LPC $\{a_1, a_2, \ldots, a_{10}\}$ nastręcza sporo trudności. W głównej mierze ze względu na duży zakres zmienności oraz możliwość zdestabilizowania filtra spowodowanego zaokrągleniami wartości współczynników dokonanymi podczas kwantyzacji. Z tego powodu zostały rozwinięte różne sposoby przekształcania współczynników filtra, tak by był on jak najmniej wrażliwy na błędy kwantyzacji. W przypadku koderów o niskiej przepływności bitowej, które są przedmiotem badań tej pracy, zdecydowanie najlepsze efekty daje reprezentacja współczynników filtra w postaci *liniowych par spektralnych* (ang. *Line Spectral Pair*, LSP). Metoda ta została zaproponowana w drugiej połowie lat 70 przez Fumitada Itakura [4, 16].

Dla filtra będącego modelem transmitancji:

$$A(z) = 1 + \sum_{n=1}^{N} a_n z^{-n} \tag{1.13}$$

przejście z LPC na reprezentację w postaci LSP, wymaga zwiększenia rzędu z N do $(N + 1)$ oraz wprowadzenia dwóch nowych wielomianów $P(z)$ i $Q(z)$ spełniających zależność:

$$A(z) = \frac{P(z) + Q(z)}{2} \tag{1.14}$$

Gdzie:

$$P(z) = A(z) + z^{-(P+1)} A(z^{-1}) = 1 + \sum_{n=1}^{N} (a_n + a_{N+1-n}) + z^{-(N+1)} \tag{1.15}$$

$$Q(z) = A(z) - z^{-(P+1)} A(z^{-1}) = 1 + \sum_{n=1}^{N} (a_n - a_{N+1-n}) - z^{-(N+1)} \tag{1.16}$$

Przekształcone w ten sposób równanie transmitancji filtra ma kilka własności:

- zera $P(z)$ i $Q(z)$ znajdują się na okręgu jednostkowym,
- zera $P(z)$ i $Q(z)$ przeplatają się ze sobą na przemian,
- po kwantyzacji współczynników opisujących ten filtr, zera wielomianów $P(z), Q(z)$ znajdują się nadal wewnątrz okręgu jednostkowego, więc układ jest minimalno-fazowy.

Zagadnienie przekształcenia współczynników LPC na LSP sprowadza się do wyznaczenia zer ww. wielomianów zwanych *Linear Spectral Frequencies* (LSF), przy założeniu, że znajdują się one wewnątrz okręgu jednostkowego (co zapewnia trzecia własność, która gwarantuje zarazem stabilność układu). Dla $N > 2$ prawdziwe są następujące własności [5]:

– -1 jest zerem $P(z)$,
– 1 jest zerem $Q(z)$.

Korzystając z powyższych własności można wykonać dzielenie wielomianów:

$$P'(z) = \frac{P(z)}{(1+z)} \tag{1.17}$$

$$Q'(z) = \frac{Q(z)}{(1-z)} \tag{1.18}$$

Przyrównując teraz:

$$P'(z) = \sum_{i=0}^{N} p_i z^{N-i} \tag{1.19}$$

$$Q'(z) = \sum_{i=0}^{N} q_i z^{N-i} \tag{1.20}$$

gdzie N jest rzędem filtra syntezy, można wyznaczyć współczynniki p_i i q_i wielomianów $P'(z)$ i $Q'(z)$. Ogólny wzór można zapisać w postaci:

$$\begin{aligned} &p_0 = 1; \\ &q_0 = 1; \\ &p_n = (a_n + a_{N-1+n}), n = 1, \ldots, N \\ &q_n = (a_n - a_{N-1+n}), n = 1, \ldots, N \end{aligned} \tag{1.21}$$

gdzie a_n, a_{N-1+n} są współczynnikami filtra (1.13). Wyciągając w równaniu 1.19 i 1.20 przed całość $A = \frac{1}{2}z^{-N/2}$ i grupując wyrazy po współczynnikach, można przekształcić wyrażenie do postaci:

$$P'(z) = A\left(p_0\left(\frac{z^{-\frac{N}{2}} + z^{\frac{N}{2}}}{2}\right) + p_1\left(\frac{z^{\frac{(N-2)}{2}} + z^{-\frac{(N-2)}{2}}}{2}\right) + \ldots + \frac{p_{N/2}}{2}\right) \tag{1.22}$$

$$Q'(z) = A\left(q_0\left(\frac{z^{-\frac{N}{2}} + z^{\frac{N}{2}}}{2}\right) + q_1\left(\frac{z^{\frac{(N-2)}{2}} + z^{-\frac{(N-2)}{2}}}{2}\right) + \ldots + \frac{q_{N/2}}{2}\right) \tag{1.23}$$

Podstawiając $z = e^{j\omega}$ i korzystając ze wzorów Eulera:

$$cos(\omega k) = \frac{e^{j\omega k} + e^{-j\omega k}}{2} \tag{1.24}$$

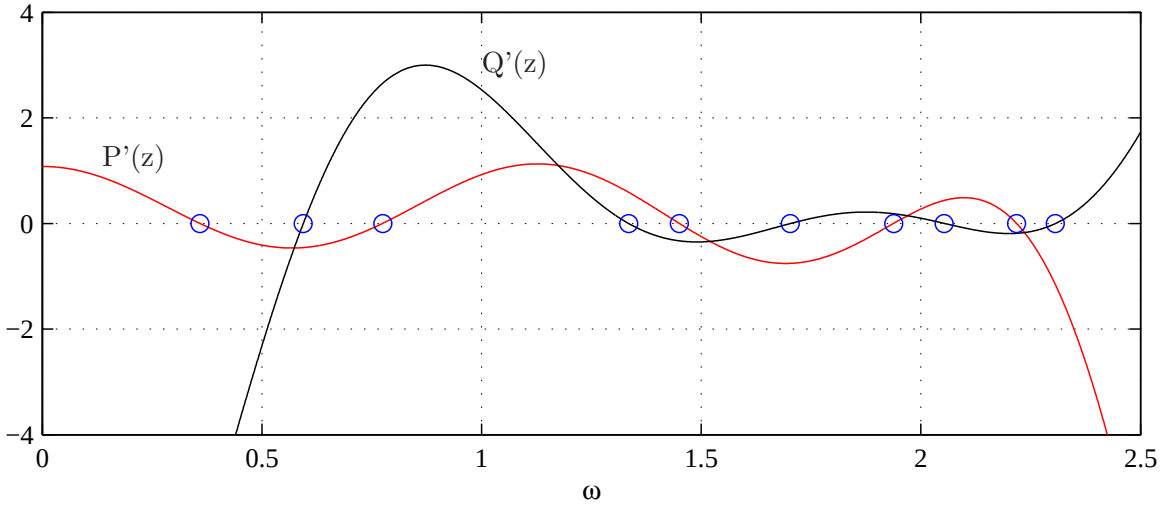
otrzymuje się ostateczną postać wielomianów:

$$P'(e^{j\omega}) = A\left(p_0 cos\left(\omega\frac{N}{2}\right) + p_1 cos\left(\omega\frac{N-2}{2}\right) + \ldots + \frac{1}{2}p_{N/2}\right) \tag{1.25}$$

$$Q'(e^{j\omega}) = A\left(q_0 cos\left(\omega\frac{N}{2}\right) + q_1 cos\left(\omega\frac{N-2}{2}\right) + \ldots + \frac{1}{2}q_{N/2}\right) \tag{1.26}$$

Dla otrzymanej postaci wielomianów 1.25 i 1.26 należy wyznaczyć miejsca zerowe, tzn. znaleźć takie ω_i dla których: $P'(e^{j\omega}) = 0$ i $Q'(e^{j\omega}) = 0$. W tym celu wykorzystywane są różne implementacje algorytmów poszukujących miejsc zerowych wielomianów.

Na rysunku 1.12 został przedstawiony przykład ilustrujący końcowy etap wyznaczania współczynników LSP.



Rysunek 1.12. Przykładowy wykres funkcji wielomianów $P'(z)$ i $Q'(z)$ z zaznaczonymi miejscami zerowymi.

W powyższym przykładzie szczególnie dobrze widoczna jest druga własność wielomianów $P(z)$ i $Q(z)$. Kolejne wartości zer występują na przemian, przeplatając się kolejno. Dla przykładu z rysunku, otrzymane wartości zer wynoszą:

$$\omega_1 = 0.359, \omega_2 = 0.594, \omega_3 = 0.775, \omega_4 = 1.335, \omega_5 = 1.450,$$

$$\omega_6 = 1.703, \omega_7 = 1.938, \omega_8 = 2.053, \omega_9 = 2.218, \omega_{10} = 2.306.$$

Oznaczając zera wielomianu $P'(z)$ jako θ_i, a zera wielomianu $Q'(z)$ jako γ_i można powiedzieć, że dla każdego otrzymanego przekształcenia wynik będzie postaci

$$0 < \gamma_1 < \theta_1 < \gamma_2 < \theta_2 < \ldots < \gamma_{P/2} < \theta_{P/2} < \pi \tag{1.27}$$

Wadą współczynników LSP jest ich złożoność obliczeniowa wpływająca na czas kodowania jak i dekodowania sygnału. Współczynniki LSP są również wrażliwe na kwantyzację w przypadku, gdy pierwiastki wielomianów $P(z)$ i $Q(z)$ znajdują się blisko siebie [16].

1.4. Przykład implementacji procedury LSP w języku C

W tabeli 1.2 został przedstawiony fragment funkcji dokonującej przekształcenia współczynników LPC do postaci liniowych par spektralnych (bez obliczania pierwiastków) zaczerpnięty ze źródeł weryfikacyjnych kodera mowy MPEG-4 HVXC (ang. *Harmonic Vecor eXcitation Coding*) (określanego w dalszej części jako MPEG-4). Przedstawione poniżej funkcje zostały wykorzystane w implementacji kodera mowy.

Tabela 1.2. Fragment kodu źródłowego z algorytmem przekształcającym współczynniki LPC do LSP znajdującego się w pliku `lpc2lsf.c`, zaczerpnięty ze standardu MPEG-4.

```
1  /* sprawdzenie ilosci wspolczynnikow */
2  odd = (np % \2 != 0)
3  if(odd) {
4      nb = (np + 1) / 2;
5      na = nb + 1;
6  }
7  else {
8      nb = np / 2 + 1;
9      na = nb;
10 }

12 /* obliczenie wspolczynnikow wielomianow P i Q */
13 fa[0] = 1.0;
14 for (i = 1, j = np; i < na; ++i, --j)
15     fa[i] = pc[i] + pc[j];
16 fb[0] = 1.0;
17 for (i = 1, j = np; i < nb; ++i, --j)
18     fb[i] = pc[i] - pc[j];

20 /* dzielenie wielomianow */
21 if (odd) {
22     for(i = 2; i < nb; ++i)
23     fb[i] = fb[i] + fb[i-2];
24 }
25 else {
26     for (i = 1; i < na; ++i) {
27         fa[i] = fa[i] - fa[i-1];
28         fb[i] = fb[i] + fb[i-1];
29     }
30 }
```

Wykorzystując własności LSP można wykonać również odwrotne przekształcenie (patrz tabela 1.3).

Tabela 1.3. Fragment kodu źródłowego z algorytmem przekształcającym współczynniki LSP do LPC znajdującego się w pliku `lsf2lpc.c`, zaczerpnięty ze standardu MPEG-4.

```
odd = order % 2;
for ( j=0; j<framel; j++) {
    xin1 = x[j];
    xin2 = x[j];
    for( i=0; i<(order>>1); i++) {
        n1 = i*4;
        n2 = n1+1;
        n3 = n2+1;
        n4 = n3+1;
        xout1 = -2. * cos(lsf[i*2+0]) * w[n1] + w[n2] + xin1;
        xout2 = -2. * cos(lsf[i*2+1]) * w[n3] + w[n4] + xin2;
        w[n2] = w[n1];
        w[n1] = xin1;
        w[n4] = w[n3];
        w[n3] = xin2;
        xin1 = xout1;
        xin2 = xout2;
    }
    /* dla rzedu filtra bedacego liczba nieparzysta */
    if(odd == 1) {
        n1 = i*4;
        n2 = n1+1;
        n4 = n2;
        xout1 = -2. * cos(lsf[i*2+0]) * w[n1] + w[n2] + xin1;
        w[n2] = w[n1];
        w[n1] = xin1;
    }

    xout1 = xin1 + w[n4+1];
    xout2 = xin2 - w[n4+2];
    x[j] =  0.5 * (xout1 + xout2);
    if(odd == 1) {
        w[n4+2] = w[n4+1];
        w[n4+1] = xin2;
    } else{
        w[n4+1] = xin1;
        w[n4+2] = xin2;
    }
}
```

Rozdział 2

Metody kwantyzacji sygnału wielowymiarowego

W tym rozdziale zostaną omówione metody reprezentacji wyznaczonych parametrów kodera mowy w takiej postaci, by można było w łatwy sposób zakodować dane. W większości przypadków ilość danych do transmisji jest o wiele większa od zadanej przepływności bitowej. Zadaniem kwantyzacji jest ograniczenie zbioru danych wejściowych tak, by otrzymać określoną przepływność bitową kosztem utraty części informacji. W zastosowanej implementacji kodera mowy kwantyzacji poddane zostały: wzmocnienie filtra G oraz jego współczynniki w postaci LSP. Dodatkowo koder mowy został rozszerzony o kwantyzator wektorowy omówiony w dalszej części. Zaprojektowane kwantyzatory zostały zbadane pod kątem wpływu na jakość odtwarzanego sygnału a wyniki badań omówione w rozdziale 4.

2.1. Kwantyzacja skalarna

Najprostszą metodą kompresji stratnej, jest kwantyzacja skalarna. Jest ona szeroko stosowana w procesie kompresji sygnałów jedno jak i wielowymiarowych. Najczęstszym przypadkiem stosowania tego rodzaju przetwarzania, jest operacja przekształcenia sygnału analogowego na postać cyfrową we wszelkiego rodzaju przetwornikach A/C. Sygnał analogowy poddawany jest kolejno operacjom: próbkowania, kwantyzacji oraz kodowania, w wyniku czego otrzymywany jest skończony zbiór wartości wyjściowych odpowiadający rzeczywistym danym. Można zatem powiedzieć, że kwantyzacja skalarna jest pewnego rodzaju ograniczeniem zbioru danych wejściowych. Jak zostanie pokazane w dalszej części, tego rodzaju kwantyzator można traktować jako szczególny przypadek wielowymiarowego kwantyzatora wektorowego.

2.1.1. Podstawowe definicje i własności

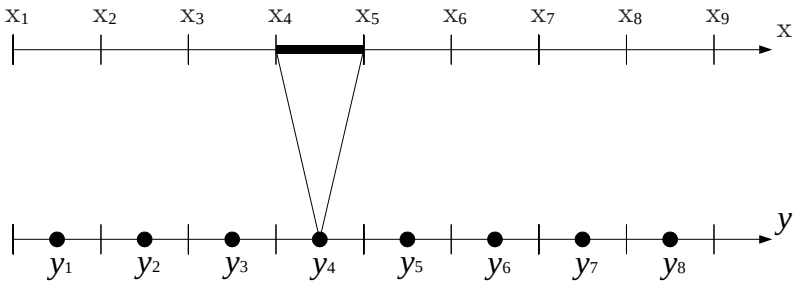
Kwantyzacja skalarna jest odwzorowaniem zbioru wartości $x \in R$ z zadanego przedziału w jego skończony N – punktowy podzbiór. Matematycznie kwantyzator skalarny Q można zdefiniować jako następujące odwzorowanie:

$$Q: \ R \rightarrow C \tag{2.1}$$

gdzie R jest zbiorem wartości wejściowych, natomiast

$$C \equiv \{y_1, y_2, \ldots, y_N\} \subset R \tag{2.2}$$

jest wyjściem kwantyzatora zwanym również *książką kodową* (ang. *codebook*) rozmiaru N. Na rysunku 2.1 została schematycznie przedstawiona zasada kwantyzacji.



Rysunek 2.1. Odwzorowanie kwantyzatora skalarnego – grubą linią zaznaczono przedział wartości, któremu odpowiada pojedyncza wartość z książki kodowej.

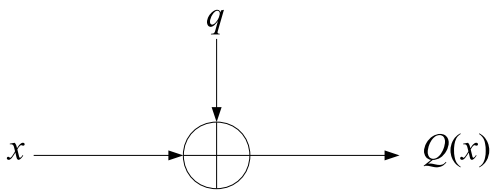
Kwantyzator jest zwany kwantyzatorem regularnym jeśli:

1. Każdy przedział R_i ma postać (x_{i-1}, x_i) z co najmniej jednym punktem ograniczającym.
2. $y_i \in (x_{i-1}, x_i)$.

gdzie wartości x_i nazywane są granicami decyzyjnymi (ang. *output levels*). Proces kwantyzacji skalarnej można zinterpretować jako przejście z ciągłego zbioru wartości wejściowych na podzbiór wartości wyjściowych. Różnica pomiędzy wejściem a wyjściem kwantyzatora, jest określana mianem błędu kwantyzacji i jest zdefiniowana jako:

$$q = x - Q(x) \tag{2.3}$$

Korzystając z definicji błędu kwantyzacji, i przekształcając ją można wprowadzić model szumu addytywnego (rysunek 2.2).



Rysunek 2.2. Model addytywnego szumu kwantyzatora.

Przestawiony na powyższym rysunku model błędu kwantyzacji jako szum addytywny

jest często stosowany ze względu na wygodę tego rodzaju reprezentacji. W rzeczywistości należy uwzględnić fakt, że szum zależy od sygnału wejściowego i nie może być traktowany jako dodatkowe źródło sygnału.

Dokonując pewnych założeń co statystycznej natury szumu kwantyzacji, można przypisać mu pewne właściwości:

1. Szum kwantyzacji ma rozkład równomierny,
2. Szum kwantyzacji jest szumem białym,
3. Szum kwantyzacji jest nieskorelowany z sygnałem wejściowym $E[xq] = E[x]E[q] = 0$.

O wiele częściej stosowaną miarą błędu dla danych wejściowych X, oraz kwantyzatora $Q = \{y_i, R_i; i = 1, 2, \ldots N\}$ jest *błąd średniokwadratowy* (ang. *mean square error*, MSE), wyrażony równaniem:

$$D = E[(X - Q(X))^2] = \sum_{i=1}^{N} \int_{R_i} (x - y_i)^2 f_X(x) dx \tag{2.4}$$

gdzie $f_X(x)$ jest zadanym rozkładem prawdopodobieństwa a N liczbą wartości wyjściowych kwantyzatora. Przedstawiona definicja błędu średniokwadratowego będzie w tej pracy punktem wyjścia do wyznaczania książek kodowych zarówno dla przypadku kwantyzatora skalarnego jak i wektorowego. Często spotykaną miarą jakości kwantyzatora jest SNR zdefiniowany jako stosunek mocy sygnału do mocy błędu kwantyzacji wyrażony w decybelach:

$$\mathrm{SNR} = 10 \log_{10} \frac{E(X^2)}{D} \tag{2.5}$$

2.1.2. Kwantyzator równomierny

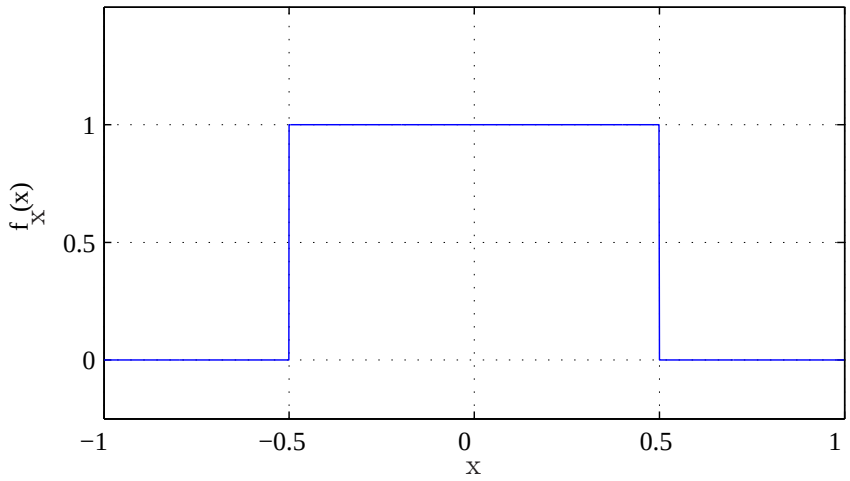
Najprostszą metodą kwantyzacji skalarnej jest zastosowanie kwantyzatora równomiernego o przedziałach równej wielkości $\Delta = x_{n+1} - x_n$, dla każdego $n = 1, \ldots, N-1$, gdzie N jest ilością przedziałów. Funkcja gęstości prawdopodobieństwa sygnału wejściowego o rozkładzie równomiernym dla takiego kwantyzatora została pokazana na rysunku 2.3. Z rysunku widać, że prawdopodobieństwo wystąpienia wartości z przedziału Δ jest takie samo dla $n = 1, \ldots, N-1$. Dla tego przypadku kwantyzacja wartości $x \in (x_{i-1}, x_i)$ odbywa się zgodnie zależnością:

$$Q(x) = \left\lfloor \frac{x}{\Delta} \right\rfloor \tag{2.6}$$

Maksymalny błąd kwantyzacji można zmniejszyć dwukrotnie, dokonując przesunięcia przedziałów wejścia (rysunek 2.4):

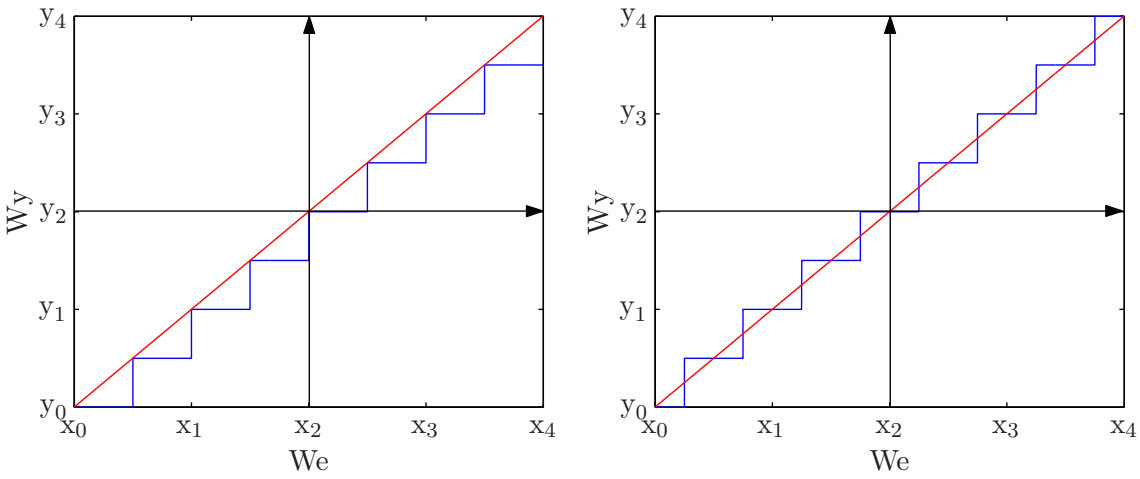
$$Q(x) = \left\lfloor \frac{x}{\Delta} + 0.5 \right\rfloor \tag{2.7}$$

Wartości wyjść kwantyzatora do jakich następuje przyporządkowywanie wejść, są rozmieszczone równomiernie będąc zarazem środkami przedziałów. Dlatego też ten rodzaj



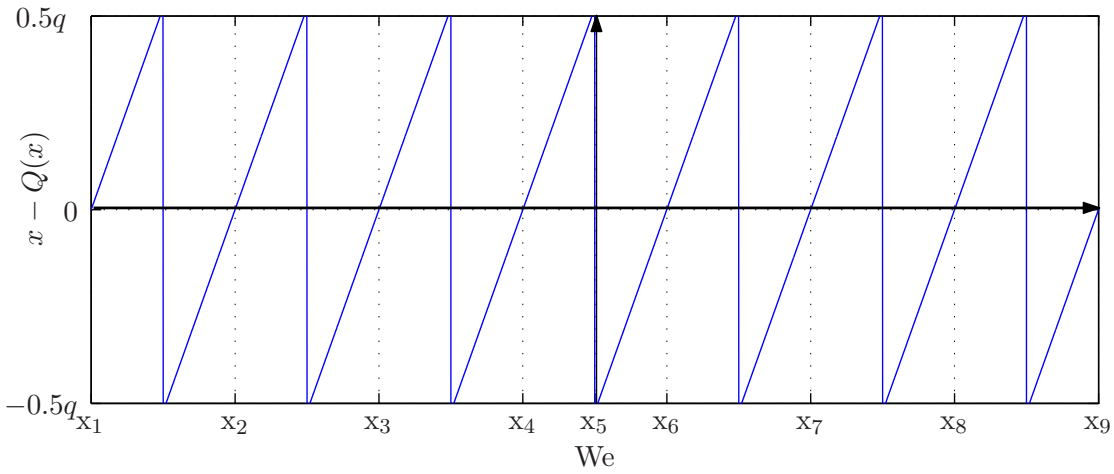
Rysunek 2.3. Przykładowa funkcja gęstości prawdopodobieństwa sygnału losowego o rozkładzie równomiernym.

odwzorowania jest liniowy. Na rysunku 2.4 została przedstawiona różnica pomiędzy kwantyzatorem skalarnym równomiernym bez przesunięcia i z przesunięciem. Można zauważyć, że dla drugiego przypadku maksymalny błąd kwantyzacji został zmniejszony z q do $0.5q$. Na rysunku 2.5 został przedstawiony błąd kwantyzacji dla danych wejściowych z zakresu $x_1, \ldots, x_9$.



Rysunek 2.4. Kwantyzator skalarny równomierny bez przesunięcia (lewy rysunek) oraz z przesunięciem (prawy rysunek).

Zakładając, że mamy do czynienia z rozkładem równomiernym sygnału wejściowego w przedziale $x \in [-x_{max}, x_{max}]$ można przyjąć, że funkcja gęstości prawdopodobień-



Rysunek 2.5. Błąd kwantyzacji między wejściem a wyjściem kwantyzatora równomiernego.

stwa tego sygnału jest określona zależnością $f_X(x) = \frac{1}{2x_{max}}$. Wówczas:

$$D = \sum_{i=1}^{M} \int_{y_{i-1}}^{y_i} (x_i - y_i)^2 \frac{1}{2x_{max}} dx \tag{2.8}$$

Wynikiem obliczenia całki (2.8), jest wartość błędu średniokwadratowego wynosząca $D = \frac{\Delta^2}{12}$ [2, 3], gdzie Δ jest wielkością kroku zdefiniowaną jako:

$$\Delta = \frac{2x_{max}}{N} \tag{2.9}$$

natomiast N jest ilością przedziałów.

Przedstawiony powyżej kwantyzator jest kwantyzatorem optymalnym pod względem błędu średniokwadratowego tylko dla danych o rozkładzie równomiernym. Na rysunku 2.6 zostały przedstawione histogramy dla każdego ze współczynników LSP. Zostały one wyznaczone na podstawie analizy statystycznej z danych wejściowych w programie `Matlab` za pomocą funkcji `histogram(...)`. Jako dane wejściowe posłużyły zebrane próbki dźwiękowe zamieszczone w Dodatku F. Na rysunku można zauważyć, że histogramy nie są opisane rozkładem równomiernym, a zatem opisany powyżej kwantyzator nie będzie efektywnym rozwiązaniem. Poszukując kwantyzatora dla danych wejściowych o nierównomiernym rozkładzie, można przyjąć założenie, że mamy do czynienia z kwantyzacją równomierną w przedziale największego prawdopodobieństwa. Ponieważ oczekiwanym rozkładem współczynników LSP jest rozkład Gaussa (rysunek 2.7), zatem można postąpić według następującego schematu:

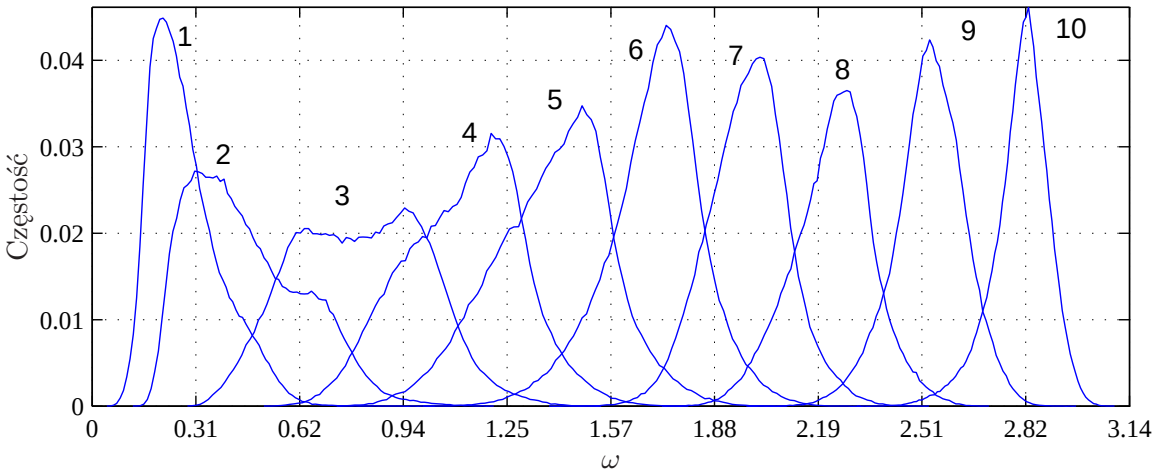
1. Wyznaczenie wartości średniej $\overline{x}$.
2. Wyznaczenie odchylenia standardowego σ_x.
3. Wyznaczenie wartości krańcowych przedziału: $x_{min} = \overline{x} - 2\sigma_x,\ x_{max} = \overline{x} + 2\sigma_x$.

4. Wyznaczenie wartości poddanej kwantyzacji dla danej przepływności bitowej zgodnie ze wzorem:

$$\widetilde{x} = K\frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.10}$$

gdzie $K = 2^N - 1$ – maksymalna wartość po kwantyzacji, N – ilość bitów na współczynnik. Operacja odwrotna do kwantyzacji jest dana wzorem:

$$x = \widetilde{x}\frac{x_{max} - x_{min}}{K} + x_{min} \tag{2.11}$$

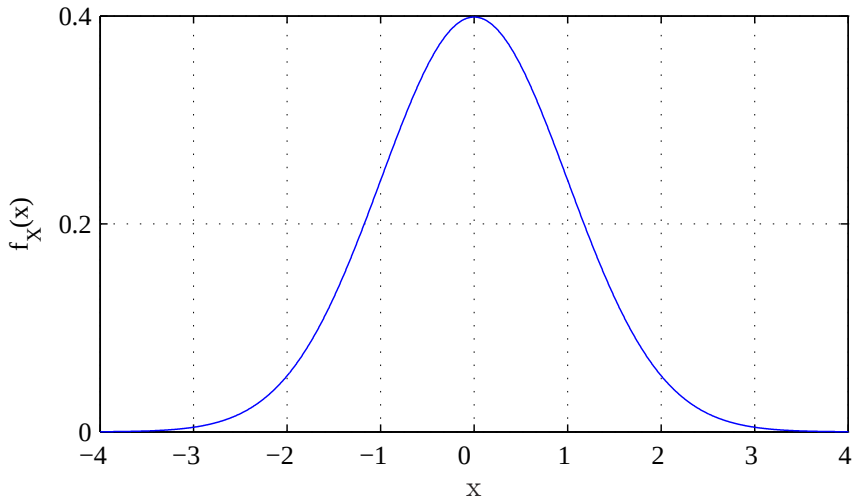


Rysunek 2.6. Rozkład gęstości prawdopodobieństwa kolejnych współczynników LSP, wyznaczony na podstawie danych statystycznych.

2.1.3. Kwantyzacja nierównomierna

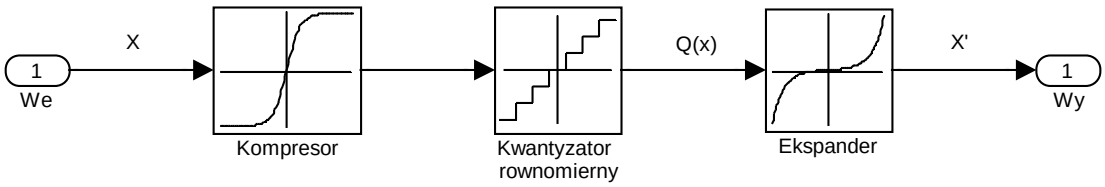
Dla rozkładów innych niż równomierny, opisany w rozdziale 2.1.2 kwantyzator skalarny jest najprostszym jednakże najmniej efektywnym rodzajem kwantyzatora skalarnego. Kwantyzator nierównomierny dzięki rozłożeniu wartości wyjść kwantyzatora w różnych odległościach od siebie, może zapewnić znacznie większy zakres dynamiki przy tej samej długości książki kodowej oraz takim samym błędzie średniokwadratowym. Na rysunku 2.7 został pokazany przykładowy rozkład gęstości prawdopodobieństwa dla kwantyzatora nierównomiernego.

Zagadnienie kwantyzacji nierównomiernej można również rozważać jako zastosowanie kwantyzatora równomiernego dla danych wejściowych przekształconych pewną funkcją. Jeśli dla obszarów o dużym prawdopodobieństwie wystąpienia wartości wejściowych przedziały kwantyzacji zostaną powiększone, a dla obszarów o małym prawdopodobieństwie wystąpienia wartości wejściowych przedziały zostaną zmniejszone, wówczas charakterystyka tak przekształconych danych odpowiadać będzie danym o rozkładzie równomiernym. Funkcja realizująca takie przekształcenie nazywana jest



Rysunek 2.7. Przykładowa funkcja gęstości gaussowskiego rozkładu prawdopodobieństwa.

kompresorem, natomiast funkcja odwrotna do niej i przywracająca pierwotną charakterystykę nazywana jest *ekspanderem*. Proces takiego przekształcenia został zaprezentowany na rysunku 2.8.



Rysunek 2.8. Schemat blokowy kwantyzatora nierównomiernego z wykorzystaniem kompresora i ekspandera.

Efekt zastosowania kompresora i ekspandera jest zatem taki sam jak z wykorzystaniem kwantyzatora nierównomiernego. W telekomunikacji najczęściej wykorzystywane są charakterystyki μ–law (Ameryka Północna oraz Japonia):

$$c_\mu(x) = sgn(x)\frac{ln\left(1+\frac{\mu|x|}{x_{max}}\right)}{\ln(1+\mu)}x_{max} \tag{2.12}$$

oraz A–law (Europa):

$$c_A(x) = \begin{cases} sgn(x)\frac{A|x|}{1+\ln(A)} & \text{dla } 0 \leqslant \frac{|x|}{x_{max}} < 1/A \\ sgn(x)\frac{1+\ln\left(\frac{A|x|}{x_{max}}\right)}{1+\ln(A)} & \text{dla } 1/A \leqslant \frac{|x|}{x_{max}} \leqslant 1 \end{cases} \tag{2.13}$$

gdzie A i μ są stałymi a x_{max} jest maksymalną wartością wejścia. Różnica pomiędzy kwantyzatorem opartym na A–law/μ–law a najlepszym pod względem błędu średniokwadratowego kwantyzatorem może sięgać ok. 4dB. Najlepszy kwantyzator może mieć jednak większy szum kwantyzacji, zwłaszcza gdy sygnał ma niską amplitudę, a jego zakres dynamiczny jest ograniczony do niewielkiego zakresy wejściowego.

Powyższe charakterystyki są opisane zaleceniem ITU–T jako standard funkcji kompresora (m.in. ITU–T G.711 oraz ITU–T G.726). W przypadku wyznaczania charakterystyki pierwotnej, stosuje się odwrotne funkcje $c_\mu^{-1}(x)$ i $c_A^{-1}(x)$ będące funkcjami ekspandera [3, 17].

Docelowo poszukiwany jest kwantyzator, który jest minimalizuje wyrażenie (2.4). Aby zapewnić minimalizację tego wyrażenia, taki kwantyzator powinien spełniać: warunek najbliższego sąsiada (ang. *nearest neighbour condition*) oraz warunek centroidu (ang. *centroid condition*). Pierwszy z nich spełnia dla zadanej książki kodowej C:

$$R_i \subset \{x : d(x, y_i) \leqslant d(x, y_j) \text{ dla każdego } j \neq i\} \tag{2.14}$$

Wówczas

$$Q(x) = y_i \text{ jeżeli } d(x, y_i) \leqslant d(x, y_j) \text{ dla każdego } j \neq i \tag{2.15}$$

Jeżeli powyższy warunek jest spełniony, to:

$$d(x, Q(x)) = \min_{y_i \in C} d(x, y_i) \tag{2.16}$$

Warunek najbliższego sąsiada jest warunkiem wystarczającym by dany kwantyzator był optymalny. Dowód można znaleźć w [2, 5].

Warunek centroidu jest zarazem warunkiem koniecznym i wystarczającym dla optymalnego kodera. Jego interpretacją jest środek masy dla danej komórki, który odpowiada wyjściu y_i z książki kodowej. Zmodyfikowana wersja warunku, zwana zgeneralizowanym warunkiem centroidu (ang. *generalized centroid condition*), pozwala na stosowanie dowolnej metryki odległości [2] i jest zdefiniowana jako:

$$cent(R) = \min_y{}^{-1} E(d(X, y)|X \in R) \tag{2.17}$$

Kwantyzator, który jest optymalny w podanym wcześniej sensie, ma następujące własności:

1. Jeżeli błąd kwantyzacji $q = x - Q(x)$, to $\sigma_q^2 = \sigma_x^2 - \sigma_{Q(x)}^2$.
2. Wartość oczekiwana nie ulega zmianie $E[Q(x)] = E[x]$.
3. Nie ma korelacji pomiędzy błędem kwantyzacji a zmienną, która jest kwantowana $E[(x - Q(x))Q(x)] = 0$.

Dowody powyższych własności można znaleźć w [2, 13].

Inne podejście wyznaczania książek kodowych zaproponował Stuart Lloyd. Zastosował on schemat generowania kwantyzatora na podstawie znajomości rozkładu prawdopodobieństwa wartości źródłowych, wyznaczonych na podstawie analizy statystycznej

danych. W zaproponowanym schemacie, dokonywana jest minimalizacja błędu średniokwadratowego przez obliczenie jego pochodnej po y_i i przyrównaniu do zera:

$$\frac{\partial}{\partial y_i}\left(\sum_{i=1}^{N}\int_{R_i}(x-y_i)^2 f_X(x)dx\right)=0, \tag{2.18}$$

$$\frac{\partial}{\partial y_i}\left(\sum_{i=1}^{N}\int_{R_i}\left(x^2 f_X(x)-xy_i f_X(x)+y_i^2 f_X(x)\right)dx\right)=0, \tag{2.19}$$

$$\int_{R_i}\left(-2x f_X(x)+2y_i f_X(x)\right)dx=0, \tag{2.20}$$

skąd

$$y_i=\frac{\int_{R_i} x f_X(x)dx}{\int_{R_i} f_X(x)dx}=\frac{\int_{b_{i-1}}^{b_i} x f_X(x)dx}{\int_{b_{i-1}}^{b_i} f_X(x)dx} \tag{2.21}$$

Rozwiązanie tego równania wykonuje się iteracyjnie uzyskując granice przedziałów kwantyzacji. Dla takiego kwantyzatora błąd średniokwadratowy określony jest wzorem:

$$D=\sigma_x^2-\sum_{i=1}^{M} y_i^2 P[b_{i-1}\leqslant X<b_i] \tag{2.22}$$

gdzie σ_x^2 jest wariancją danych wejściowych, a y_i są wektorami książki kodowej.

Pomimo zalet takich jak szybkość kwantyzacji i operacji odwrotnej do kwantyzacji oraz trywialnej implementacji, kwantyzacja skalarna charakteryzuje się stosunkową wysoką przepływnością bitową w przypadku danych wykazujących korelację pomiędzy kwantowanymi współczynnikami lub danych o rozkładzie prawdopodobieństwa o rozkładzie innym niż równomierny.

2.2. Kwantyzacja wektorowa

W przypadku danych, które można zgrupować w pewne bloki, podejście kwantyzatora skalarnego nie jest wystarczająco efektywne. Takim przypadkiem są współczynniki LPC/LSP, które można traktować jak pojedynczy 10 – wymiarowy wektor w każdym bloku i które są ze sobą skorelowane. Wykorzystując ten fakt można zastosować inne podejście do ich reprezentacji. Jako pierwszy wysunął taką koncepcję Shannon, kodując dane o coraz dłuższych blokach i uzyskując lepszą średnią bitową dla zadanego błędu średniokwadratowego.

2.2.1. Podstawowe definicje i własności

Kwantyzacja wektorowa w przestrzeni Euklidesowej R^M jest odwzorowaniem zbioru M – wymiarowych wektorów w skończony dzbiór N wektorów M – wymiarowych. Kwantyzator wektorowy można zdefiniować jako:

$$Q:\ R^M\rightarrow C \tag{2.23}$$

gdzie

$$C \equiv \{y_1, y_2, \ldots, y_N\} \subset R^M \tag{2.24}$$

Analogicznie jak dla przypadku kwantyzatora skalarnego, C nazywane jest książką kodową o rozmiarze N. Tak zdefiniowany kwantyzator ma rozdzielczość $r = \frac{\log_2(N)}{M}$, która określa ile bitów wymaganych jest do reprezentacji danego współczynnika.

Z każdym wektorem książki kodowej związane jest pojęcie *regionów* (bądź *komórek*) (ang. *Voronoi regions*). Dla $i = 1, 2, \ldots, N$ region zdefiniowany jest jako:

$$R_i = \{x \in R^M : Q(x) = y_i\} = Q^{-1}(y_i) \tag{2.25}$$

i spełnia zależność

$$\bigcup_i R_i = R^M \text{ oraz } R_i \bigcap R_j = \emptyset \text{ dla } i \neq j \tag{2.26}$$

Kwantyzator wektorowy jest nazywany regularnym, jeżeli:

1. każda komórka R_i jest figurą wypukłą,
2. $y_i \in R_i$ dla każdego i.

Podobnie jak w przypadku kwantyzatora skalarnego, można zdefiniować optymalny pod względem błędu średniokwadratowego kwantyzator wektorowy. Kwantyzator taki spełnia warunek najbliższego sąsiada:

$$R_i = \{x : d(x, y_i) \leqslant d(x, y_j)\} \text{ gdzie } i, j = 1, 2, \ldots, N \text{ oraz } i \neq j \tag{2.27}$$

W kwantyzacji wektorowej centroidem jest każdy niepusty zbiór wektorów $y_i \in R^M$ który minimalizuje błąd średniokwadratowy pomiędzy wektorami książki kodowej C a zbiorem wektorów treningowych $x \in R$:

$$cent(R_0) = \{y_0 : E[d(X, y_0)|X \in R] \leqslant E[d(X, y)|X \in R]\}, \forall_{y \in R^M} \tag{2.28}$$

Można wykazać [2, 5], że warunkiem optymalności kodera jest by centroid spełniał zależność:

$$y_i = cent(R_i) \tag{2.29}$$

Warunkiem koniecznym by kwantyzator $Q(x)$ był optymalny pod względem błędu (MSE), jest warunek zerowego prawdopodobieństwa granicznego (ang. *Zero Probability Boundary Condition*, ZPB):

$$P\left(\bigcup_{j=1}^{N} B_j\right) = 0 \tag{2.30}$$

Interpretacją powyższej definicji jest brak wystąpień punktów granicznych w optymalnej książce kodowej.

Dla ogólnego przypadku w kwantyzatorach zastosowanych np. w standardzie MPEG–4 HVXC, stosowana jest metryka odległości kwantyzatora wektorowego postaci:

$$D = d(x, y) = \sum_{i=1}^{M} w_i (x_i - y_i)^2 \tag{2.31}$$

gdzie M jest wymiarem wektora a w_i wagą współczynników LSP. W tej pracy wykorzystana została metryka odległości oparta o miarę błędu (MSE) postaci:

$$D = d(x, y) = \sum_{i=1}^{M} (x_i - y_i)^2 \tag{2.32}$$

Istotną cechą kwantyzatora wektorowego jest fakt, że nie tylko jest uogólnieniem przypadku kwantyzatora skalarnego na przestrzeń N – wymiarową, ale zapewnia uzyskanie najlepszej możliwej efektywności kodowania danych.

Wadą tego rodzaju kwantyzacji jest fakt, że nie istnieje optymalny pod względem czasowym algorytm tworzenia książki kodowej. W praktyce korzysta się z suboptymalnych algorytmów iteracyjnych np. omówionego w dalszej części algorytmu LBG. Wygenerowana książka kodowa musi być dostępna zarówno po stronie kodera i dekodera co zwiększa zapotrzebowania na zasoby pamięciowe. Dodatkowo w koderze proces wyszukiwania najbliższego sąsiada jest bardzo złożony. Samo odtworzenie danych po stronie dekodera jest trywialnym procesem wymagającym odczytania danych z tablicy o zadanym indeksie.

2.2.2. Standardowy algorytm LBG

Przedstawiony w poprzednim rozdziale algorytm Lloyda, można uogólnić na przypadek wielowymiarowy [2, 3], przy założeniu, że dostępny jest zbiór uczący (zwany zbiorem treningowym). Algorytm ten ze względu na stosunkowo łatwą możliwość implementacji znalazł szerokie zastosowanie w dziedzinie kompresji danych metodą słownikową. Schemat generowania książki kodowej dla danego zbioru wektorów treningowych można przedstawić następująco:

1. Inicjalizacja.
   - wybór zbioru wektorów treningowych $\{X_n\}_{n=1}^{N}$ gdzie N jest rozmiarem książki kodowej,
   - wyznaczenie początkowej postaci książki kodowej $C^k : \{Y_i^{(0)}\}_{i=1}^{M}$ gdzie M jest rozmiarem zbioru treningowego,
   - ustalenie parametrów: najmniejszej wartości błędu (MSE) $D^{(0)} = \infty$ (w praktyce przyjmowana jest bardzo duża wartość numeryczna, np. maksymalna wartość zakresu zmiennej typu `int`), $k = 0$ i wartości progu błędu ϵ.
2. Wyznaczenie komórek kwantyzacji.
   - przyporządkowanie zbioru wektorów treningowych X do najbardziej odpowiadających im (w sensie wykorzystywanej metryki np. błędu średniokwadratowego) wektorów książki kodowej Y:

$$R_i^{(k)} = \{X_n : d(X_n, Y_i) < d(X_n, Y_j) \forall_{i \neq j}\} \quad i = 1, 2, \ldots, M$$

   - wyznaczenie średniego błędu w danej iteracji $D^{(k)}$,
   - sprawdzenie warunku:

$$\frac{D^{(l-1)} - D^{(l)}}{D^{(l-1)}} \leqslant \epsilon$$

Jeżeli warunek został spełniony to znaczy, że zostało znalezione minimum lokalne o zadanym błędzie ϵ i następuje zakończenie algorytmu.

3. Uaktualnienie książki kodowej.
   - zastąpienie wszystkich wektorów ze starej książki kodowej $C^{(k-1)}$ nowymi wektorami książki kodowej $C^{(k)}$ będącymi środkami ciężkości w danym regionie $R_i^{(k)}$,
   - inkrementacja k o jeden i powrót do punktu 2.

Przedstawiony powyżej algorytm znany jest pod różnymi nazwami: uogólniony algorytm Lloyda (ang. *generalized Lloyd algorithm*, GLA), bądź algorytm Lindego–Buza–Graya (LBG) dla N – wymiarów. W szczególności dla $N = 1$ procedura wyznaczania obszarów kwantyzacji może być zastosowana do wyznaczenia książki kodowej kwantyzatora skalarnego nierównomiernego.

2.2.3. Technika podziałów

W celu zmniejszenia złożoności przeszukiwania książki kodowej, można ją zbudować w oparciu o strukturę drzewa algorytmem zaproponowanym przez Linde, Buzo i Graya, zwanym techniką podziałów (ang. *splitting algorithm*) [2, 3]. Podstawową zaletą tego rodzaju algorytmu jest największa szybkość wyszukiwania obszaru kwantyzacji dla danych wejściowych. Wadą jest natomiast konieczność podwojenia rozmiaru książki kodowej, oraz fakt, że jej całkowity rozmiar musi być określony przez zależność:

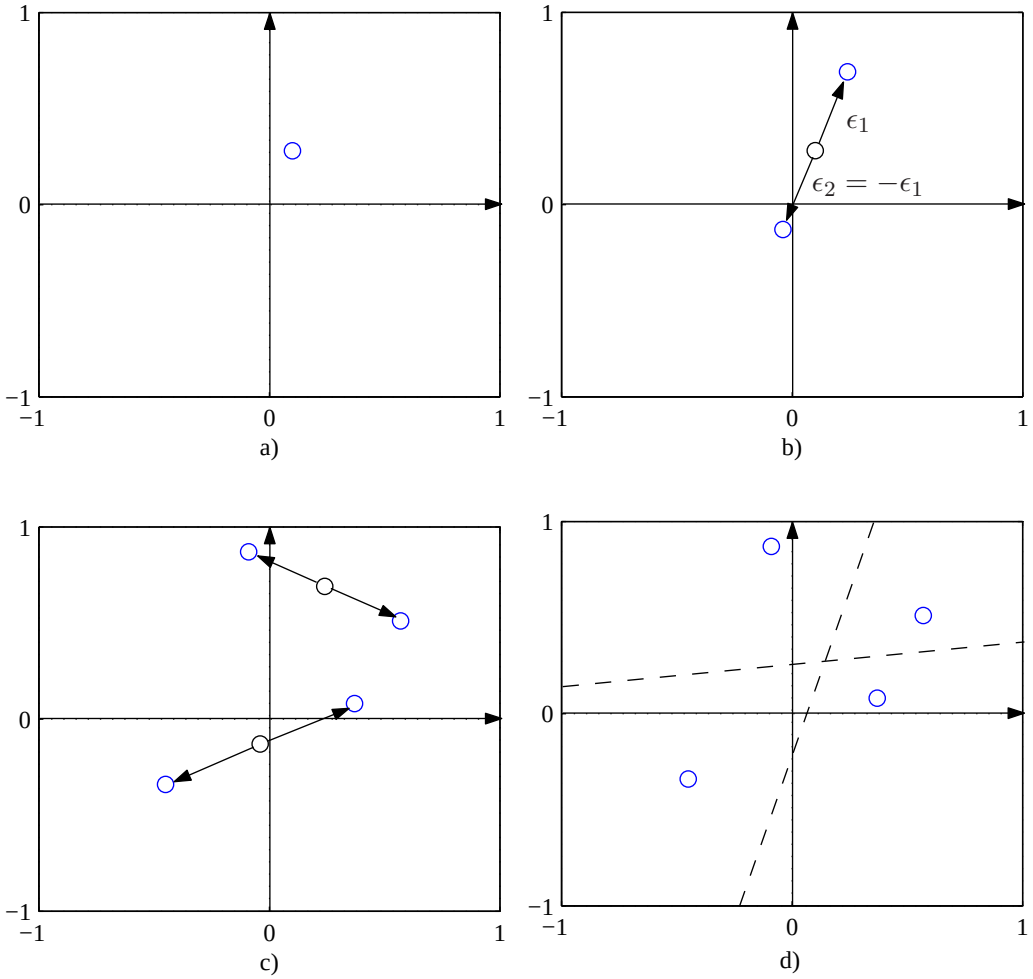
$$K = 2^{N+1} - 2 \tag{2.33}$$

gdzie N jest rzeczywistą ilością bitów wykorzystywaną przy kodowaniu współczynników poddanych kwantyzacji. Dodatkowo nie można użyć takiej metody z kwantyzatorem wykorzystującym pomiar odległości nie będący metryką np. MPEG–4 HVXC. Generacja takiej książki kodowej przebiega według następującego schematu:

1. Wyznaczenie centroida na podstawie wszystkich wektorów treningowych, będącego środkiem ciężkości całego zbioru.
2. Optymalizacja książki kodowej standardowym algorytmem LBG opisanym w podrozdziale 2.2.2.
3. Dwukrotne zwiększenie rozmiaru książki kodowej.
4. Wyznaczenie nowych centroidów poprzez dodanie wektorów zakłóceń ϵ_1 i ϵ_2 takich, że $\epsilon_1 = -\epsilon_2$ do każdego otrzymanego w punkcie 2 centroidu.
5. Jeśli ilość otrzymanych w wyniku podziału centroidów jest równa 2^N następuje wyjście z algorytmu. W przeciwnym wypadku następuje powrót do punktu 2.

Idea postępowania została przedstawiona na rysunku 2.9. Wychodząc od jednoelementowego słownika, w każdym etapie podwaja się jego wielkość dochodząc do założonej ilości poziomów. Dołączenie do bieżącego słownika otrzymanego w poprzednim etapie słownika zapewnia, że po podziale nowy słownik będzie przynajmniej tak samo dobry jak przed dokonaniem podziału. Metoda drzewa charakteryzuje się mniejszą efektywnością kompresji, głównie ze względu na fakt, że już na początkowych etapach decyzyjnych popełniony błąd co do wyboru poddrzewa może spowodować wybór

błędnego wektora docelowego. Dodatkowo na każdym poziomie przeszukuje się mniejszy podzbiór wektorów. Stosując modyfikacje algorytmu [10], można jednak zwiększyć efektywność przeszukiwania, dodatkowo zastosowanie takiej struktury pozwala na progresywną transmisję danych, gdzie każde poddrzewo przybliża coraz bardziej wektor docelowy.



Rysunek 2.9. Kolejne etapy podziałów: a) pierwszy wyznaczony centroid, b) wyznaczenie dwóch nowych centroidów przez zaburzenie wektorem losowym, c) zaburzenie każdego z nowych centroidów wektorem losowym, d) cztery centroidy otrzymane w wyniku zastosowania techniki podziału.

Sama technika podziałów może posłużyć do wyznaczenia zwykłej książki kodowej. Przy takim podejściu brany pod uwagę jest tylko ostatni poziom, który traktowany jest jak zwykła książka kodowa wygenerowana metodą LBG.

2.2.4. Metoda symulowanego wyżarzania

Aby wyeliminować konieczność wielokrotnego tworzenia książki kodowej w celu znalezienia globalnego minimum, można w każdej iteracji wprowadzić dodatkowe zakłócenie o pewnej wariancji i zerowej wartości średniej, które jest dodawane do każdego wyznaczonego centroidu. Tego typu optymalizacja jest znana pod nazwą *metody symulowanego wyżarzania* (ang. *stochastic relaxation*). Ogólny schemat postępowania można przedstawić następująco:

1. Inicjalizacja podstawowych parametrów tj. najmniejszej wartości błędu (MSE) $D^{(0)}$, ilości iteracji N po ilu ma się zakończyć wyżarzanie, $n = 0$ oraz błędu zbieżności do lokalnego minimum ϵ.
2. Wykonanie jednej iteracji algorytmu LBG omówionym w podrozdziale 2.2.2.
3. Obliczenie nowej wariancji zakłóceń zgodnie ze schematem schładzania. Jeżeli numer bieżącej iteracji $n > N$ wówczas $\sigma_n^2 = 0$ (zakończenie wyżarzania).
4. Wprowadzenie zaburzenia do zbioru wektorów książki kodowej.
5. Jeśli zmiana całkowitego MSE jest poniżej zadanego progu ϵ, wyjdź z algorytmu, w przeciwnym razie $n = n + 1$ i następuje powrót do punktu 2.

Wariancja zakłóceń w danej iteracji jest wyliczana w każdym kroku zgodnie z tzw. *schematem schładzania* (ang. *cooling schedule*):

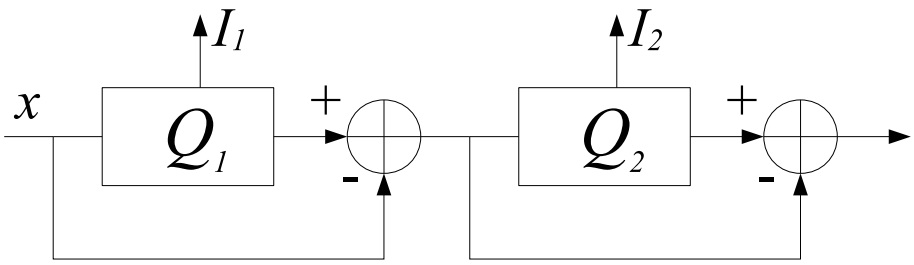
$$\sigma_n^2 = \sigma^2 \left(1 - \frac{n}{N}\right)^3 \tag{2.34}$$

gdzie:
σ^2 – wariancja początkowa
n – aktualny numer iteracji
N – całkowita ilość iteracji

Oprócz przedstawionych metod istnieją również i inne algorytmy generacji książek kodowych. Przykładem może być algorytm najbliższych sąsiadów (ang. *pairwise nearest neighbour*, PNN) [2, 3]. Złożoność obliczeniowa tego algorytmu dla rozmiarów książek kodowych będących przedmiotem zainteresowania tej pracy, wyklucza go jednak z praktycznych zastosowań sprawiając, że ma on znaczenie czysto akademickie.

2.2.5. Kwantyzatory wieloetapowe

W celu zredukowania złożoności obliczeniowej kodowania, można posłużyć się metodą wieloetapowego kwantyzatora wektorowego (ang. *multistage vector quantization*, MSVQ). W tym podejściu kilka koderów jest połączonych ze sobą kaskadowo w taki sposób, że wejściem kolejnego kodera jest różnica pomiędzy wartością wejściową w poprzednim etapie a wartością poddaną kwantyzacji. Dlatego też ten rodzaj kwantyzatora określany jest również jako *resztkowy kwantyzator wektorowy*. Idea MSVQ została



Rysunek 2.10. Schemat kodera MSVQ.

schematycznie przedstawiona na rysunku 2.10. Złożoność obliczeniowa jak i wielkość słownika dla takiego schematu, zostają zmniejszone z $\Pi_{i=1}^{M} N_i$ do $\sum_{i=1}^{M} N_i$, gdzie N_i – rozmiar książki kodowej dla danego etapu. Dla trzyetapowego kwantyzatora MSVQ, wyjścia każdego kwantyzatora można zapisać jako:

$$Y_1 = Q_1(X)$$

$$Y_2 = Q_2(Y_1) = Q_2(X - Q_1(X))$$

$$Y_3 = Q_3(Y_2) = Q_3(X - Q_1(X) - Q_2(X - Q_1(X)))$$

Po stronie dekodera odtworzona wartość będzie wynosiła:

$$X' = Y_1 + Y_2 + Y_3$$

Całkowity koszt obliczeniowy wynikający z zastosowania wieloetapowego kwantyzatora wektorowego wynosi:

$$MC_{MSVQ} = \sum_{i=1}^{N} 2^{r_i} \tag{2.35}$$

gdzie r_i to ilość bitów książki kodowej na i – tym poziomie, a N jest całkowitą ilością poziomów. Dla kwantyzatora wektorowego jednoetapowego o takiej samej wielkości książki kodowej koszt obliczeniowy wynosi:

$$MC_{VQ} = \prod_{i=1}^{N} 2^{r_i} \tag{2.36}$$

W tabeli 2.1 zostały zestawione wyniki porównania złożoności obliczeniowej przy przeszukiwaniu książki kodowej metodą siłową dla kwantyzatora jedno i dwuetapowego.

	Całkowita ilość działań	
Rozmiar książki kodowej [bit]	Jednoetapowy kwantyzator	Dwuetapowy kwantyzator
4	640	320
6	2560	640
8	10240	1280
10	40960	2560
12	163840	5120
14	655360	10240
16	2621440	20480
18	10485760	40960
20	41943040	81920

Tabela 2.1. Porównanie złożoności obliczeniowej przy zadanym rozmiarze książki kodowej dla kwantyzatora wektorowego jedno i dwuetapowego 10 – wymiarowego. Kwantyzator dwuetapowy ma w każdym etapie książki kodowe tej samej wielkości.

Z powyższej tabeli widać, że w przypadku książki kodowej o rozmiarze 2^{20} wektorów, ilość działań potrzebnych na przeszukanie książki kodowej metodą siłową kwantyzatora dwuetapowego jest 512 razy mniejsza niż w przypadku kwantyzatora jednoetapowego.

W tabeli 2.2 zostało przedstawione zestawienie kosztów pamięciowych dla przypadku jednopoziomowego i dwupoziomowego kwantyzatora wektorowego. Dodanie kolejnych poziomów spowodowałoby dalszą redukcję zapotrzebowania na pamięć.

	Zapotrzebowanie na pamięć	
Rozmiar książki kodowej [bit]	Jednoetapowy kwantyzator [słowa]	Dwuetapowy kwantyzator [słowa]
4	16	8
6	64	16
8	256	32
10	1024	64
12	4096	128
14	16384	256
16	65536	512
18	262144	1024
20	1048576	2048

Tabela 2.2. Porównanie wymagań pamięciowych przy zadanym rozmiarze książki kodowej dla kwantyzatora wektorowego jedno i dwuetapowego. Kwantyzator dwuetapowy ma w każdym etapie książki kodowe tej samej wielkości.

Z powyższego zestawienia widać, że jednopoziomowa 10–wymiarowa książka kodowa o rozmiarze 20 bitów zapisana na typie `float` wymaga 40 MB pamięci. Dla takich samych parametrów, książka dwupoziomowa wymaga natomiast tylko 80 kB.

Przedstawiona wyżej procedura znajduje zastosowanie dla dużych przepływności bitowych. Dla małych przepływności bitowych następuje znaczna utrata informacji co doprowadza do pogorszenia właściwości jakie oferuje taka struktura.

2.3. Algorytmy przeszukiwania książek kodowych

Osobnym zagadnieniem są algorytmy przeszukiwania książek kodowych. W najprostszym podejściu, można wykonać pełne przeszukanie wymagające w najgorszym przypadku $4NC$ działań, gdzie N jest wymiarem wektora, a C – wielkością książki kodowej. Metoda ta jest niezwykle czasochłonna zwłaszcza przy dużych książkach kodowych. Koszt obliczeniowy można zmniejszyć wykorzystując schemat częściowego obliczania odległości (ang. *partial distance search*, PDS). Ogólna idea opiera się na założeniu, że zamiast obliczać MSE dla wszystkich wektorów z książki kodowej, można obliczyć raz błąd średniokwadratowy pomiędzy wektorem wejściowym a pierwszym wektorem z książki kodowej, a dla następnych wektorów wyliczać częściowe sumy dla kolejnych składowych wektora. W przypadku gdy suma dla r składowych będzie większa od najmniejszej znalezionej wartości MSE, następuje przejście do następnego wektora. Poniżej został przedstawiony schemat postępowania [2]:

1. Inicjalizacja zmiennych: $i = 0, r = 1, D_r = 0$ gdzie: i jest indeksem wektora, r jest indeksem składowej wektora a D_r błędem (MSE) i – tego wektora o r sumach kwadratów różnic składowych.
2. Obliczenie wartości błędu (MSE) dla wektora o indeksie $i = 1$, $D = \sum_{i=1}^{R}(x_i - c_i)^2$ gdzie R jest wymiarem wektora, oraz inkrementacja wartości i o jeden.
3. Obliczenie $D_r = D_r + \sum_{i=1}^{r}(x_i - c_i)^2$.
4. Jeśli $D_r < D$ i $R < N$ (suma odległości r pierwszych elementów i – tego wektora jest mniejsza od błędu (MSE) D) to zwiększ r o jeden i wróć do punktu 2.
5. Jeśli $D_r < D$ i $r \geqslant N$ (został znaleziony wektor dla o mniejszej wartości MSE) wówczas $D^{(i)} = D_r$ oraz $i = i + 1$.
6. Jeśli $i < I$ gdzie I jest rozmiarem książki kodowej (nie przeszukano całej książki kodowej) to ustawienie $D_r = 0$ i powrót do punktu 2.

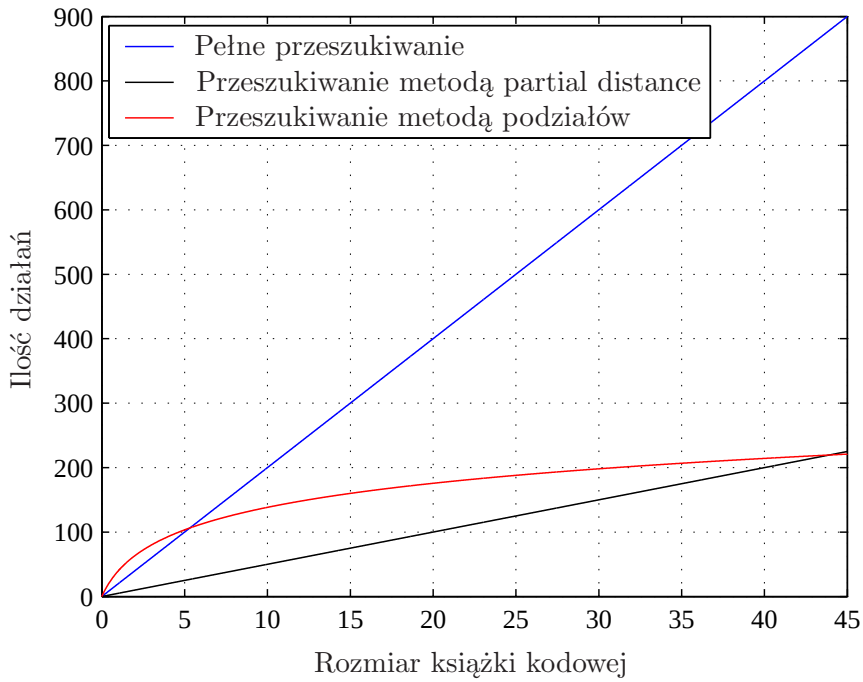
Postępowanie tą metodą pozwala zmniejszyć koszt obliczeniowy średnio czterokrotnie w stosunku do pełnego przeszukiwania. Ponadto nadaje się do przeszukiwania schematów kwantyzacji wektorowej z pomiarem, który nie spełnia warunku metryki (np. koder MPEG–4 HVXC). Ze względu na łatwość implementacji w tej pracy została wykorzystana właśnie ta metoda. Poniżej zamieszczona została funkcja do przeszukiwania książki kodowej napisana przez autora:

Tabela 2.3. Funkcja realizująca przeszukiwanie książki kodowej metodą częściowego obliczania odległości napisana przez autora.

```
1  int Partial_Distance(const vector *Vec, const vector *CBook, int CSize)
2  {
3      float D=0,D_part,tmp;
4      int ind=0;
5      D=Size(Vec)*MSE(Vec,&CBook[0]); // obliczenie bledu (MSE) dla pierwszego
6                                      // wektora
7      for(int j=1;j<CSize;j++)
8      {
9          D_part=0.0;
10         for(int i=0;i<Size(Vec);i++)
11         {
12             tmp=Vec->a[i]-CBook[j].a[i];
13             D_part+=tmp*tmp;
14             if(D_part>=D)
15             {
16                 break; // blad wiekszy od poprzedniego,mozna przejsc do nastepnego
17                        // wektora
18             }
19             if(i==Size(Vec)-1 && D_part<D)
20             {
21                 ind=j; // znaleziono wektor o mniejszym bledzie (MSE)
22                 D=D_part;
23             }
24         }
25     }
26     return(ind);
27 }
```

W przypadku zastosowania struktury książki kodowej w postaci drzewa, można wykorzystać dedykowany mu algorytm przeszukiwania. Zakładając, że jest to drzewo binarne dla którego na każdym poziomie porównuje się tylko dwa wektory, można zredukować koszt obliczeniowy do postaci logarytmicznej. W takim przypadku zysk obliczeniowy względem metody pełnego przeszukiwania zwiększa się wraz z wielkością książki kodowej. Na rysunku 2.11 zostały porównane koszty obliczeniowe dla ww. metod przeszukiwania książek kodowych. Z rysunku widać, że stosowanie książki kodowej opartej o strukturę drzewa daje największy zysk obliczeniowy dla dużych książek kodowych.

Zaprezentowane w tym rozdziale algorytmy kwantyzacji wektorowej nie wyczerpują tematu tworzenia i przeszukiwania optymalnej książki kodowej. Osobną grupę tworzą metody bazujące na kratowej kwantyzacji wektorowej, które pozwalają rozwiązać wymóg zapotrzebowania pamięci dla wyższych przepływności bitowych, klasyfikowana kwantyzacja wektorowa czy też adaptacyjna kwantyzacja wektorowa szczególnie przydatna dla źródeł danych zmieniających swoją charakterystykę w czasie [2, 3, 10]. Należy również wspomnieć o wariacjach na temat metod opartych na LBG, np. kwantyzator wektorowy typu skala–kształt (pośrednio zastosowany w implementacji algorytmu LBG) czy kwantyzator z usuniętą średnią (dedykowany głównie kompresji obrazów [2, 3]). Powyższe techniki stanowią zbyt obszerny i złożony dział nie będący przedmiotem zainteresowania prezentowanej pracy, z tego powodu nie zostały rozpatrzone pod kątem zastosowania w zaimplementowanym koderze mowy.



Rysunek 2.11. Ilustracja złożoności obliczeniowej algorytmu kwantyzacji wektorowej dla trzech metod przeszukiwania książki kodowej (dla wektora 10 – wymiarowego).

Rozdział 3

Implementacja programowa kodera mowy

W tym rozdziale omówiony zostanie zaprojektowany koder mowy oparty na standardzie LPC–10, a wzbogacony o dodatkowe algorytmy. Zmiany zaproponowano by poprawić jakość syntetyzowanego dźwięku poprzez ulepszenie syntezy sygnału pobudzenia. Zastosowane podejście podobne jest do schematu stosowanego w koderach mowy MELP oraz MPEG–4, które charakteryzują się dużo lepszą jakością generowanego sygnału niż koder standardu LPC–10. Zarówno sam algorytm rozszerzonego kodera mowy LPC–10 jak i generatory książek kodowych zostały napisane w języku C++ by przyspieszyć wykonywane obliczenia. Jako środowisko programistyczne został wybrany darmowy pakiet MinGW wraz z edytorem Dev–C++ pracujący w środowisku Windows. Ponieważ w żadnej części projektu nie zostały wykorzystane funkcje specyficzne dla systemu Windows, po niewielkich poprawkach zarówno koder jak i generatory książek kodowych można wykorzystać np. w środowisku Linux np. z użyciem kompilatora GCC.

Elementem na który w tej pracy został położony szczególny nacisk, jest rozszerzenie algorytmu LPC–10 o kwantyzację wektorową w celu zmniejszenia przepływności bitowej przy transmisji skompresowanej mowy oraz wyznaczanie książek kodowych dostosowanych do specyfiki polskiej mowy. Pierwszym krokiem w celu implementacji kwantyzacji wektorowej było zebranie odpowiedniej ilości próbek dźwiękowych stanowiących zbiór treningowy. Na podstawie zebranych danych wyznaczone zostały książki kodowe wykorzystane w procesie kwantyzacji przy wykorzystaniu algorytmów opisanych w rozdziale 2. W dalszej części rozdziału zostanie omówiona zaproponowana implementacja wraz z ulepszeniami.

3.1. Implementacja kodera LPC–10

Kod źródłowy algorytmu realizującego przetwarzanie sygnału składa się z funkcji realizujących zadania wyszczególnione w opisie kodera i dekodera w podrozdziale 1.2.1 oraz 1.2.2. Pełny opis dostępnych dla użytkownika interfejsów, został zamieszczony w Dodatku B wraz z przykładowym wykorzystaniem napisanych bibliotek.

Algorytm kodera i dekodera został napisany z wykorzystaniem możliwości programowania obiektowego, dodatkowo dodane zostały klasy pomocnicze mające na celu obsłużenie funkcji odpowiedzialnych za ustawianie podstawowych parametrów kodera (częstotliwość próbkowania, rząd filtra, długość zastosowanego okna) oraz wczytywanie z pliku parametrów z których koder korzysta. Wszystkie funkcje poza funkcjami przekształcającymi współczynniki LPC do LSP i odwrotnie (zaczerpniętymi z modelu weryfikacyjnego kodera MPEG–4) zostały napisane przez autora, w szczególności funkcja filtracji cyfrowej oparta na algorytmie buforów kołowych zgodnie z [1] (funkcja `filter::filterdo(...)`). Całkowity kod kodera i dekodera dokonującego przetwarzania i kompresji sygnału, zajmuje ok. 550 linii kodu.

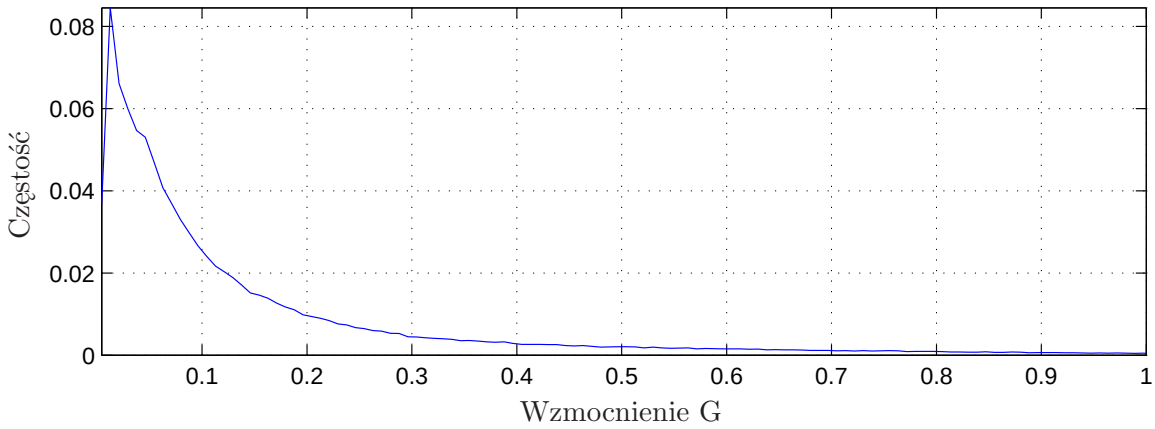
3.2. Wybór danych wejściowych

Podstawą do wyznaczenia książek kodowych był odpowiednio duży zbiór danych wejściowych o jak największym zróżnicowaniu. W tym celu zostały zebrane próbki mowy ludzkiej zgrane m.in. z radia oraz odbiornika telewizyjnego o całkowitej długości wynoszącej ok. 1.5 godziny. Na ich podstawie obliczone zostały współczynniki LSP, tworzące zbiór treningowy. Osobno zostały zebrane próbki mowy ludzkiej stanowiące zbiór testowy dla badań jakości kompresji sygnału mowy. Dane zostały zebrane tak by były jak najbardziej zbliżone do rzeczywistych warunków pracy kodera mowy, zarówno pod kątem różnej grupy lektorów (płeć i wiek), różnej głośności jak i obecności dodatkowych szumów obniżających jakość nagrania. Wszystkie próbki zostały zapisane w postaci plików wave o rozdzielczości 16 bitów przy częstotliwości próbkowania 8 kHz. Histogramy współczynników LSP dla wyznaczonego zbioru zostały przedstawione na rysunku 2.7. Można zauważyć, że poszczególne współczynniki dążą do rozkładu Gaussa co odpowiada charakterystyce zgodnej z oczekiwaniami. Zbiór treningowy został zapisany w postaci pliku binarnego pozbawionego nagłówka i gotowego do wczytania przez generator książki kodowej, którego kod źródłowy został zamieszczony w Dodatku F. Podobnie został wyznaczony zbiór treningowy dla wzmocnień filtra traktu głosowego. Zebrane i przetworzone dane pomiarowe zostały zamieszczone w plikach wyszczególnionych w tabeli 3.1 znajdujących się wraz ze źródłowymi próbkami dźwiękowymi w Dodatku F. Można zauważyć, że sam zbiór treningowy ze względu na swoje rozmiary nie nadaje się do bezpośredniego zastosowania w koderze mowy.

Nazwa	Rozmiar w bajtach	Wymiar wektora	Ilość wektorów	Opis
wektory.dat	11 223 040	10	280576	zbiór treningowy wykorzystany do generacji książek kodowych
lspa.dat	5 611 520	5	280576	zbiór treningowy dla książek kodowych drugiego poziomu
lspb.dat	5 611 520	5	280576	zbiór treningowy dla książek kodowych drugiego poziomu
G.dat	1 122 256	1	280576	zbiór treningowy wartości wzmocnień wykorzystany do generacji książek kodowych

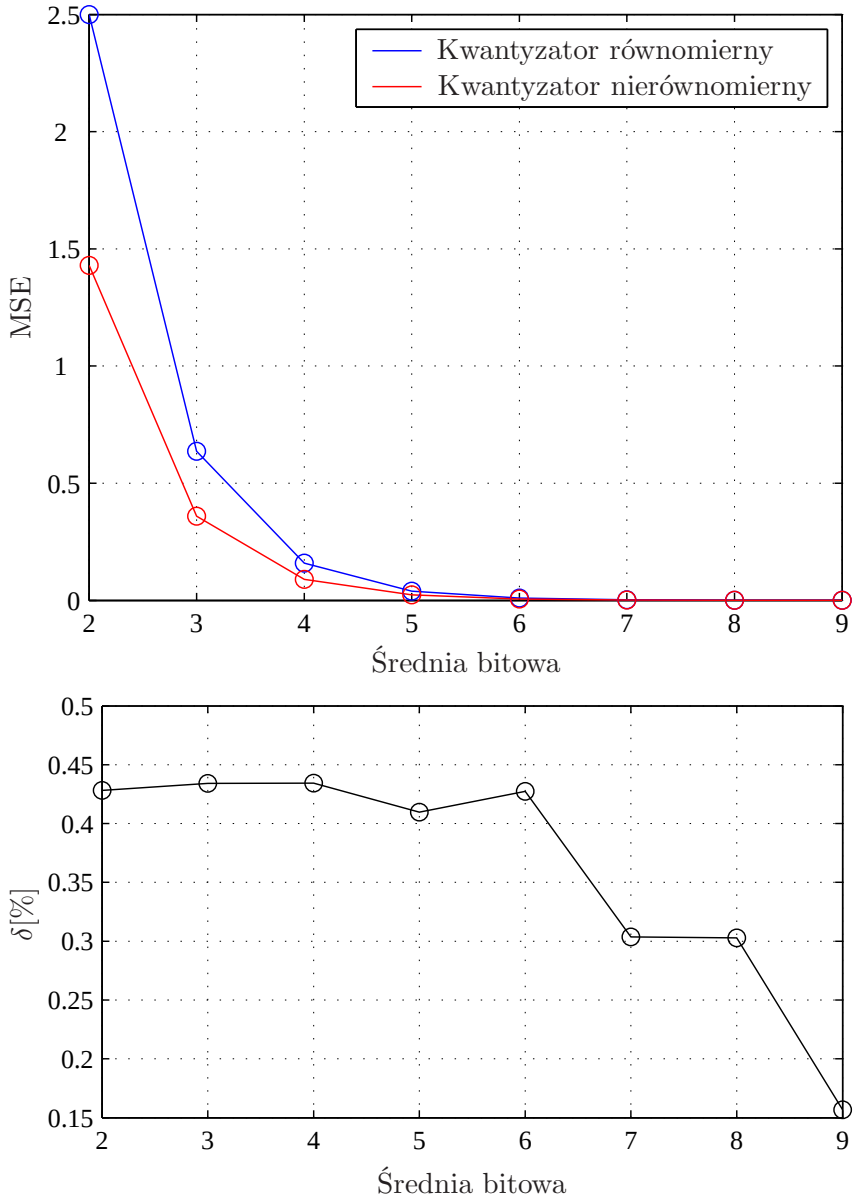
Tabela 3.1. Wykaz plików wyznaczonych podczas generowania optymalnych książek kodowych.

Rozkład gęstości prawdopodobieństwa wzmocnień filtra, został przedstawiony na rysunku 3.1. Na podstawie otrzymanego zbioru wzmocnień filtra G został wyznaczony przedział największego prawdopodobieństwa wynoszący $P \in [0, 0.6989]$. Otrzymany zbiór został wyznaczony przy pomocy zaimplementowanego w języku C++ algorytmu Durbina–Levinsona.



Rysunek 3.1. Rozkład gęstości prawdopodobieństwa dla wzmocnienia filtra traktu głosowego.

Z rysunku widać, że rozkład wzmocnień filtra traktu głosowego nie jest rozkładem gaussowskim jak można by tego oczekiwać, a bardziej przypomina rozkłady typu Chi–2 bądź F-Snedecora. Dla wyliczonego zakresu zmienności wzmocnień został wyznaczony i porównany błąd (MSE) dla kwantyzatora skalarnego równomiernego oraz nierównomiernego dla kilku różnych wielkości książek kodowych. Rezultaty przedstawiono na wykresie z rysunku 3.2.



Rysunek 3.2. Porównanie MSE dla wzmocnienia kwantyzatorów skalarnych: równomiernego i nierównomiernego (rysunek górny) oraz błędu względnego pomiędzy obydwoma wynikami (rysunek dolny).

Na przedstawionym rysunku widać, że zysk z zastosowania kwantyzacji skalarnej nierównomiernej daje błąd średniokwadratowy o ok. 43% mniejszy niż w przypadku kwantyzacji równomiernej. Można również zauważyć, że stosunek wartości MSE kwantyzatora równomiernego do nierównomiernego nie jest wartością stałą lecz zależy od średniej bitowej. Wraz ze wzrostem średniej bitowej przewaga kwantyzacji skalarnej nierównomiernej nad równomierną maleje.

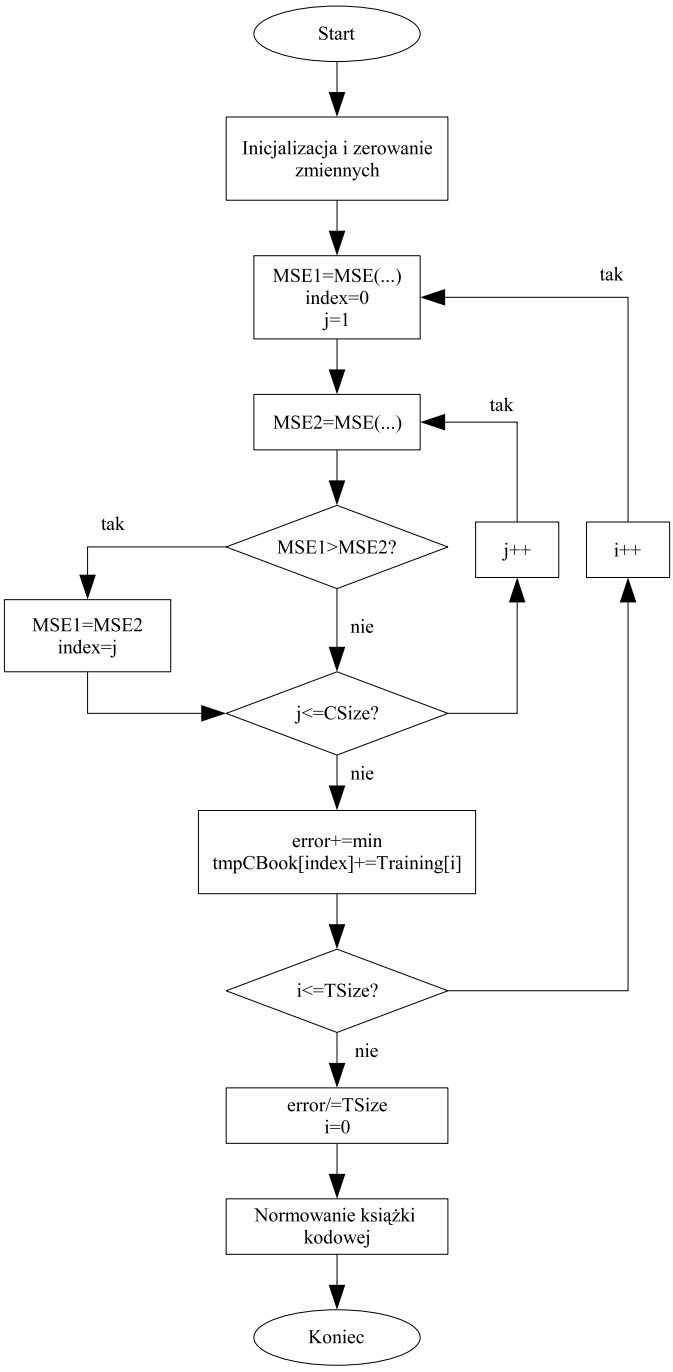
Z rysunku 3.2 można wywnioskować, że kwantyzator skalarny nierównomierny wprowadza mniejszy błąd średniokwadratowy niż kwantyzator równomierny przy takiej wielkości książki kodowej. Zwiększanie jej rozmiaru powoduje jednak, że różnica między obydwiema metodami kwantyzacji się zaciera. W szczególności, gdy rozmiar książki kodowej będzie dążył do rozmiaru zbioru treningowego, wówczas wartości MSE kwantyzatora równomiernego i nierównomiernego będą dążyć do zera.

3.3. Implementacja generatorów książek kodowych

W trakcie prac nad wyznaczaniem optymalnych książek kodowych, zostały napisane w języku C++ algorytmy odpowiedzialne za generację książek kodowych na podstawie zadanego zbioru treningowego. Standardowy algorytm LBG został zaimplementowany zgodnie z opisem podanym w podrozdziale 2.2. Całe wywołanie algorytmu zostało rozbite na pomniejsze funkcje by zminimalizować ryzyko powstawania błędów. Dodatkową zaletą rozbicia programu na pomniejsze funkcje, była możliwość wykorzystania części przygotowanych rozwiązań w implementacjach innych algorytmów. Podstawą algorytmu jest funkcja `LBG(...)` mająca na celu wykonanie jednej iteracji algorytmu i wyznaczenie książki kodowej o pewnym błędzie średniokwadratowym. Schemat blokowy tej funkcji został pokazany na rysunku 3.3, gdzie:

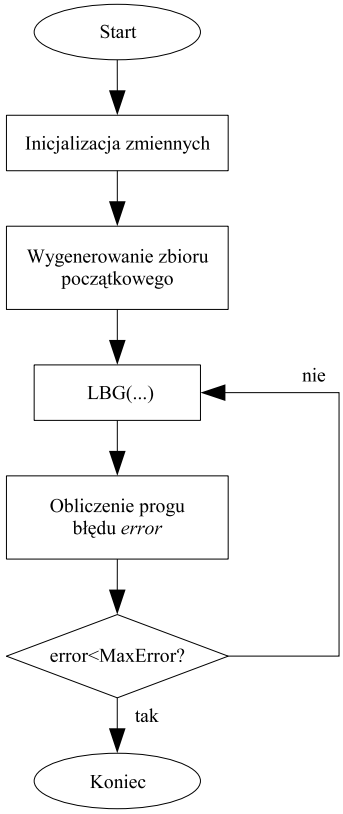
- `j,i` są indeksami inkrementowanymi w pętlach for,
- `CSize` jest zadanym rozmiarem książki kodowej postaci 2^N gdzie N jest ilością bitów,
- `TSize` jest rozmiarem zbioru treningowego będącego ilością wektorów treningowych,
- `error` jest błędem średniokwadratowym książki kodowej,
- `Training` jest wskaźnikiem do zbioru treningowego,
- `tmpCBook` jest zmienną pomocniczą,
- `index` zawierają informację o indeksie do wektora z książki kodowej najbliższego szukanemu wektorowi. treningowemu

Wybór początkowych wektorów książki kodowej został dokonany przez wygenerowanie niepowtarzających się indeksów do zbioru treningowego generatorem liczb pseudolosowych (funkcje `szukaj(...)` oraz `randomLBG(...)`). Cała metoda postępowania została zawarta w funkcji `standardLBG(...)` zwracającej całkowity błąd (MSE) dla wygenerowanej książki kodowej. Schemat blokowy dla tej funkcji został przedstawiony na rysunku 3.4. Szczegółowe informacje o powyższych funkcjach zostały zamieszczone w Dodatku A.



Rysunek 3.3. Schemat blokowy funkcji wykonującej jedną iterację algorytmem LBG.

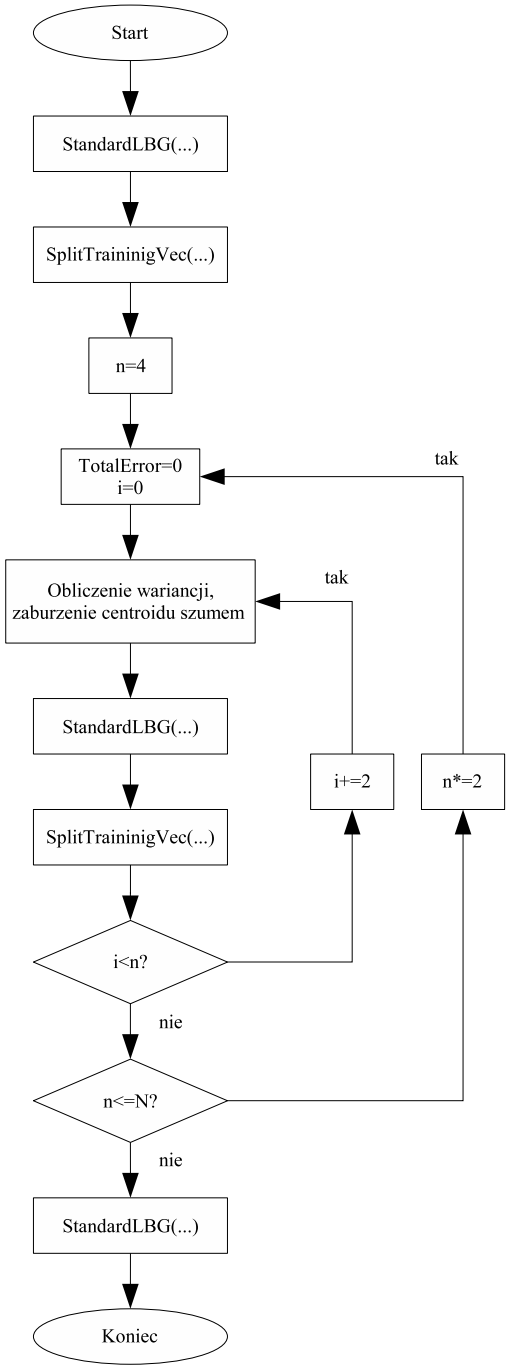
Najwięcej trudności, sprawił algorytm oparty na technice podziałów zaimplementowany w funkcji `splitLBG(...)`. Jego implementacja opisana w podrozdziale 2.2.3 wymagała wykorzystania dynamicznej alokacji pamięci dla każdego nowego podzbio-



Rysunek 3.4. Schemat blokowy funkcji wyznaczającej książkę kodową metodą standardowego LBG o zadanym błędzie zbieżności oznaczonym jako `MaxError`.

ru na danym poziomie drzewa. Pomimo rozbicia algorytmu na wiele pomniejszych funkcji, największa trudność przy pisaniu kodu generatora była związana z podziałem poszczególnych centroidów oraz koniecznością przechowywania niezbędnych informacji związanych z danym poziomem podziału, takich jak tablica `bracket` (informacje o ilości wektorów w każdym centroidzie). W tej części programu wykorzystano funkcję `LBG(...)` ze standardowego algorytmu LBG. Schemat blokowy dla metody generacji techniką podziałów został przedstawiony na rysunku 3.5, gdzie:

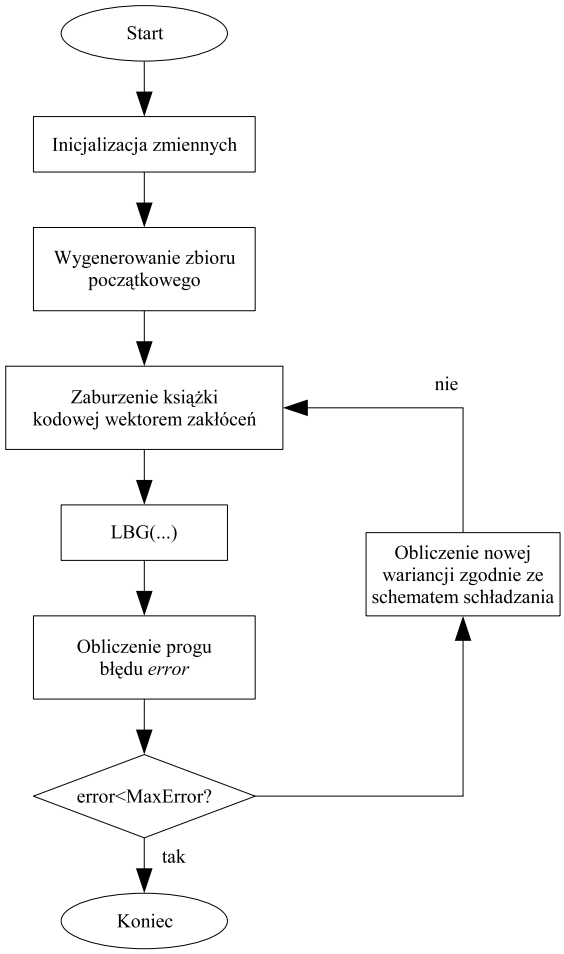
- `N` jest rozmiarem książki kodowej na ostatnim poziomie,
- `i,n` są zmiennymi inkrementowanymi w pętlach `for`,
- `TotalError` jest całkowitym błędem (MSE) zwracanym przy wychodzeniu z funkcji `splitLBG(...)`.



Rysunek 3.5. Schemat blokowy funkcji wyznaczającej książkę kodową metodą techniki podziałów.

Algorytm symulowanego wyżarzania wymagał zmodyfikowania funkcji `LBG(...)` tak by uwzględniała wpływ wektora zaburzeń zakłócającego każdy z centroidów.

W tym celu została wykorzystana dodatkowa zmienna `ITERACJE` typu `int`, która była jednym z parametrów w równaniu schematu schładzania. Jej interpretacją jest szybkość schładzania układu. Zgodnie z [5] została przyjęta wartość zmiennej `ITERACJE=400`. Przyjęcie większych wartości nie wpływało znacząco na poprawę wyniku natomiast znacznie wydłużały czas potrzebny na wygenerowanie książki kodowej. Schemat blokowy zastosowanego algorytmu generacji bazującym na symulowanym wyżarzaniu przedstawiony został na rysunku 3.6.



Rysunek 3.6. Schemat blokowy funkcji wyznaczającej książkę kodową metodą symulowanego wyżarzania.

Omówione powyżej implementacje algorytmów generatora książki kodowej mają ok. 350 linii kodu źródłowego. Przykładowa aplikacja tworząca książki kodowe jak i kody źródłowe została zamieszczone w Dodatku F.

3.4. Wyznaczanie książki kodowej pierwszego poziomu

Na podstawie otrzymanego zbioru treningowego, zostały wyznaczone optymalne książki kodowe dla kwantyzatora skalarnego i wektorowego z wykorzystaniem różnych algorytmów. W przypadku kwantyzatora skalarnego, punktem wyjścia było wyznaczenie granic przedziałów wartości kolejnych współczynników LSP. Obliczone wartości zostały zamieszczone w tabeli 3.2. Wyznaczenie przedziału zmienności współczynników LSP zostało wykonane procedurą przedstawioną w podrozdziale 2.1.

Współczynnik	a_{Min}	a_{Max}
a_1	0.070	0.496
a_2	0.120	0.809
a_3	0.413	1.212
a_4	0.769	1.481
a_5	1.039	1.753
a_6	1.411	1.998
a_7	1.743	2.230
a_8	1.999	2.488
a_9	2.323	2.749
a_{10}	2.641	2.982

Tabela 3.2. Granice przedziałów zmienności dla kolejnych współczynników LSP oznaczonych jako a_i.

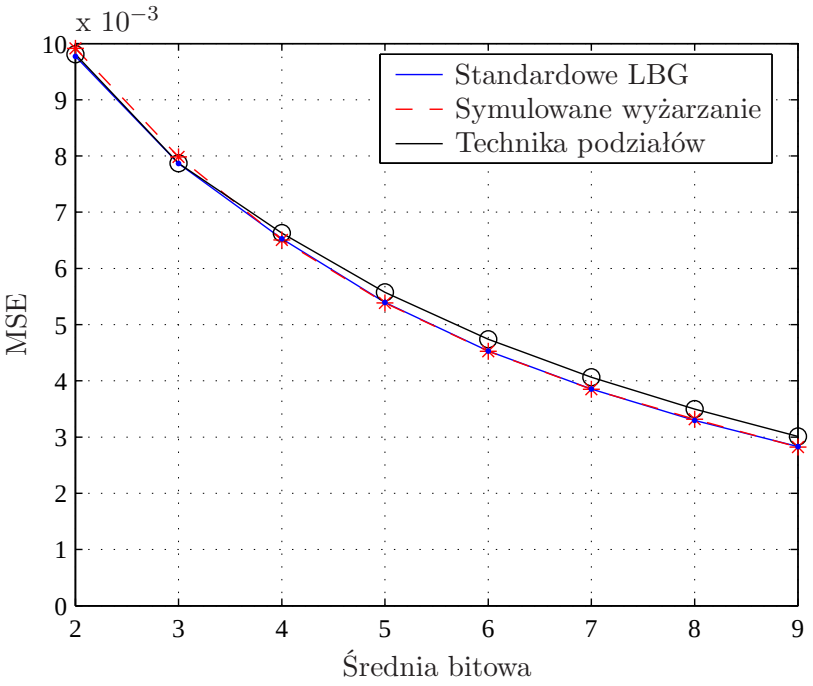
W podobny sposób jak dla przedstawionej kwantyzacji wzmocnień filtra traktu głosowego, zostało wykonane porównanie za pomocą miary MSE, kwantyzatorów skalarnych równomiernych i nierównomiernych. W przypadku kwantyzatora nierównomiernego, najlepszy kwantyzator został wyznaczony metodą LBG dla przypadku jednowymiarowego. Wyniki zostały przedstawione w tabeli 3.3, przy czym podstawą do wyznaczenia wartości błędu (MSE) było przydzielenie bitów dla każdego współczynnika LSP zgodnie ze specyfikacją standardu LPC–10. Wyznaczone granice przedziałów z tabeli 3.2 zostały wykorzystane do porównania błędów średniokwadratowych podczas kwantyzacji współczynników LPC kwantyzatorami: równomiernym i nierównomiernym.

Na podstawie wyników z tabeli 3.3 można zauważyć, że kwantyzator skalarny nierównomierny daje średnio o 15% mniejszy błąd dla wyznaczonych współczynników LSP niż kwantyzator skalarny nierównomierny.

		Błąd średniokwadratowy	
Współczynnik	Bitów na wsp	Kwantyzator równomierny	Kwantyzator nierównomierny
a_1	5	$0.148 \cdot 10^{-4}$	$0.123 \cdot 10^{-4}$
a_2	5	$0.385 \cdot 10^{-4}$	$0.311 \cdot 10^{-4}$
a_3	5	$0.518 \cdot 10^{-4}$	$0.483 \cdot 10^{-4}$
a_4	5	$0.412 \cdot 10^{-4}$	$0.382 \cdot 10^{-4}$
a_5	4	$1.660 \cdot 10^{-4}$	$1.482 \cdot 10^{-4}$
a_6	4	$1.122 \cdot 10^{-4}$	$0.975 \cdot 10^{-4}$
a_7	4	$0.771 \cdot 10^{-4}$	$0.687 \cdot 10^{-4}$
a_8	4	$0.778 \cdot 10^{-4}$	$0.689 \cdot 10^{-4}$
a_9	3	$2.361 \cdot 10^{-4}$	$2.069 \cdot 10^{-4}$
a_{10}	2	$6.080 \cdot 10^{-4}$	$4.897 \cdot 10^{-4}$

Tabela 3.3. Porównanie wartości MSE kwantyzatorów skalarnych równomiernych i nierównomiernych dla książek kodowych o rozmiarach zgodnych ze standardem LPC–10. Współczynniki a_i są współczynnikami filtra traktu głosowego w postaci LSP.

Punktem wyjścia do ustalenia efektywności metod generacji książek kodowych przedstawionych w podrozdziale 2.2, było wykonanie ok. 1000 generacji książek o różnych długościach metodami bazującymi na algorytmie LBG, a następnie porównanie z jedną generacją metodą symulowanego wyżarzania (rysunek 3.7).



Rysunek 3.7. Porównanie wartości błędu (MSE) dla badanych metod generacji książek kodowych.

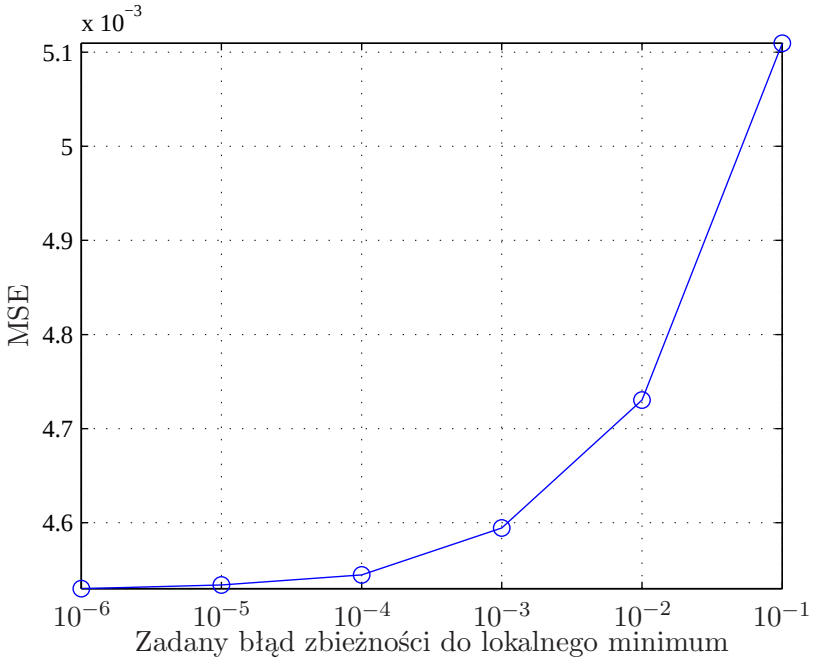
Z rysunku 3.7 widać wyraźnie przewagę standardowego LBG nad techniką podziałów. Różnica jakości między książkami generowanymi tymi metodami wynosi ok. 6%. Algorytm symulowanego wyżarzania daje błąd średniokwadratowy zbliżony do najlepszej wartości MSE uzyskanej z wielu iteracji standardowym algorytmem LBG, jednak zbieżność do lokalnego minimum osiągnięta zostaje już po pierwszych generacjach.

Drugim parametrem wpływającym na jakość wygenerowanych książek kodowych jest próg błędu zbieżności do lokalnego minimum, zdefiniowany jako:

$$\epsilon \geqslant \frac{D^{(l-1)} - D^{(l)}}{D^{(l-1)}} \tag{3.1}$$

gdzie $D^{(l-1)}$ jest wartością MSE z poprzedniej iteracji, a $D^{(l)}$ wartością MSE z bieżącej. Dla kilku różnych wartości ϵ zostały wyznaczone (przy zadanym rozmiarze książki kodowej) błędy średniokwadratowe obliczone za pomocą algorytmu LBG.

Zależność wartości błędu (MSE) od błędu zbieżności do lokalnego minimum została przedstawiona na rysunku 3.8. Książka kodowa dla której wyznaczona została charakterystyka, miała rozmiar odpowiadający 6 bitom.



Rysunek 3.8. Zależność MSE od zadanego progu błędu zbieżności dla lokalnego minimum, dla książki kodowej o rozmiarze $N = 64$.

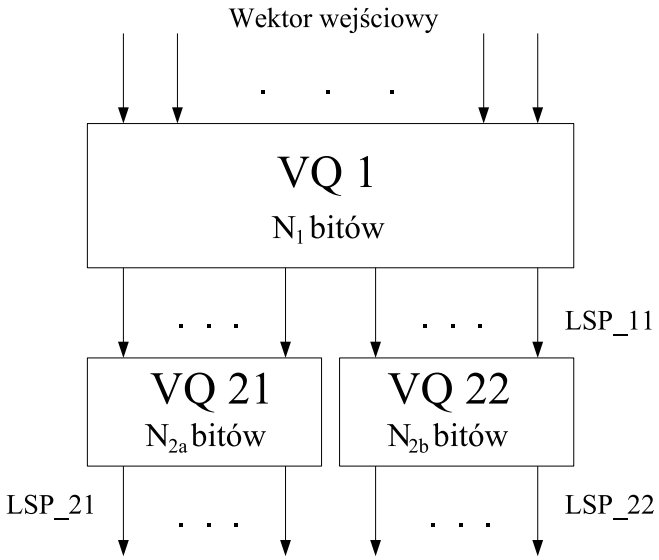
Na podstawie wyników przedstawionych na rysunku 3.8 można przyjąć, że już dla błędu zbieżności wynoszącego $\epsilon = 10^{-3}$ dalsze zmniejszanie jego wartości nie wpłynie znacząco na całkowite MSE książki kodowej (zostanie osiągnięty stan ustalony), zwiększy natomiast niepotrzebnie czas obliczeń.

W ogólnym przypadku wybór metody inicjalizacji i generowania książki kodowej nie powinien mieć znaczenia [3], jednak wyniki przeprowadzonych badań świadczą o

tym, że najmniej efektywnym algorytmem pod względem minimalizacji błędu średniokwadratowego dla rozpatrywanego kwantyzatora wektorowego współczynników LSP jest technika podziałów.

3.5. Wyznaczenie książki kodowej drugiego poziomu

Ze względu na wzrost złożoności obliczeniowej, w rozpatrywanej implementacji wyznaczone zostały książki kodowe dwupoziomowe. W przypadku generacji książek metodą MSVQ (podobnie jak w koderze mowy MPEG–4), w pierwszej kolejności została wyznaczona książka kodowa rozmiaru N_1. Następnie zostały wyznaczone dwa zbiory treningowe z wykorzystaniem otrzymanej książki kodowej. Jeden zawierał wektory różnic pomiędzy współczynnikami poddanymi kwantyzacji a współczynnikami wejściowymi o indeksach $i = 1, 2, 3, 4, 5$, natomiast drugi zawierał wektory różnic o indeksach $i = 6, 7, 8, 9, 10$. Dla obydwu zbiorów zostały wyznaczone książki kodowe drugiego poziomu o wielkościach N_{2a} i N_{2b}.



Rysunek 3.9. Schemat wyznaczania poszczególnych książek kodowych z tabeli 3.4. LSP_11 jest książką kodową pierwszego poziomu, a LSP_21, LSP_22 książkami kodowymi drugiego poziomu.

Na rysunku 3.9 został przedstawiony schemat postępowania przy wyznaczaniu dwupoziomowych książek kodowych. Błąd kwantyzacji na wyjściu kwantyzatora VQ1 jest dzielony na dwa wektory, które są poddawane dalszej kwantyzacji w VQ21 i VQ22.

Otrzymane wartości błędów (MSE) dla kilku przykładowych książek kodowych zostały przedstawione w tabeli 3.4.

	Pierwszy poziom		Drugi poziom		
lp.	N_1 [bit]	MSE_1	N_{2a} [bit]	N_{2b} [bit]	MSE_2
1	14	$1.455 \cdot 10^{-3}$	14	13	$0.078 \cdot 10^{-3}$
2	10	$2.518 \cdot 10^{-3}$	10	10	$0.310 \cdot 10^{-3}$
3	5	$5.379 \cdot 10^{-3}$	7	5	$1.861 \cdot 10^{-3}$
4	5	$5.379 \cdot 10^{-3}$	6	4	$2.222 \cdot 10^{-3}$
5	4	$6.491 \cdot 10^{-3}$	3	3	$4.186 \cdot 10^{-3}$

Tabela 3.4. Porównanie wartości MSE dla różnych książek kodowych, gdzie: N_i jest rozmiarem książki kodowej a MSE_i jest jej błędem średniokwadratowym dla i – tego poziomu.

Książki kodowe z drugiego wiersza tabeli 3.4 odpowiadają standardowi MPEG–4 dla niskich przepływności bitowych. Z przedstawionych danych widać, że dla takiej samej przepływności bitowej kwantyzator wielopoziomowy wprowadza o wiele większy błąd średniokwadratowy. Dla książki kodowej o rozmiarze 10 bitów, kwantyzator dwupoziomowy z wiersza 5 wprowadza całkowity błąd średniokwadratowy $MSE = 10.677 \cdot 10^{-3}$. W porównaniu z pierwszym poziomem kwantyzatora z wiersza 2 (który można traktować jako niezależny kwantyzator o książce kodowej rozmiaru 10 bitów) jest to wynik ok. 4-krotnie gorszy.

Zbadany został również wpływ wielkości książek kodowych drugiego poziomu na błąd (MSE) przy $N_1 = 5$ bitów oraz $N_{2a}+N_{2b} = 10$ bitów. Wyniki zostały zamieszczone w tabeli 3.5. Można zauważyć, że najmniejsza wartość MSE występuje gdy jest tylko jedna książka kodowa – innymi słowy sama obecność książki kodowej dla drugiego zbioru pięciu wektorów powoduje gwałtowny wzrost błędu średniokwadratowego. Dla takiego przypadku następuje wzrost wartości MSE o 66%. Z zamieszczonych danych widać również, że w obecności obydwu książek kodowych najmniejszą wartość MSE otrzymuje się dla dla $N_{2a} = 2$ bity i $N_{2b} = 3$, bądź $N_{2a} = 3$ bity i $N_{2b} = 2$ bity (wiersz 3 i 4 tabeli 3.5).

Dla potrzeb pracy magisterskiej książki kodowe zostały wygenerowane w oparciu o schemat kwantyzacji dwupoziomowej – w trakcie przeprowadzanych badań jakości, powyższe wyniki zostały zweryfikowane na podstawie badań jakości metodą obiektywną i subiektywną.

	LSP_21		LSP_22		
lp.	MSE_1	N_{2a} [bit]	MSE_2	N_{2b} [bit]	MSE
1	0	0	$1.324 \cdot 10^{-3}$	5	$0.662 \cdot 10^{-3}$
2	$2.532 \cdot 10^{-3}$	1	$1.523 \cdot 10^{-3}$	4	$2.027 \cdot 10^{-3}$
3	$2.155 \cdot 10^{-3}$	2	$1.765 \cdot 10^{-3}$	3	$1.960 \cdot 10^{-3}$
4	$1.817 \cdot 10^{-3}$	3	$2.087 \cdot 10^{-3}$	2	$1.952 \cdot 10^{-3}$
5	$1.565 \cdot 10^{-3}$	4	$2.453 \cdot 10^{-3}$	1	$2.009 \cdot 10^{-3}$
6	$1.354 \cdot 10^{-3}$	5	0	0	$0.677 \cdot 10^{-3}$

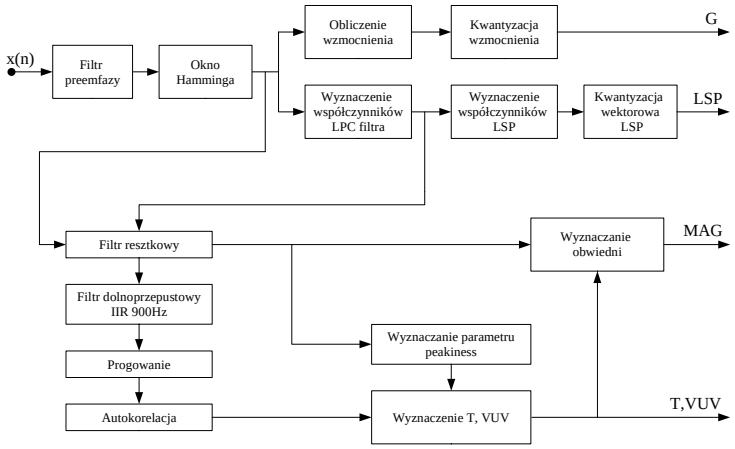
Tabela 3.5. Zależność całkowitego błędu drugiego poziomu od wielkości książek kodowych LSP_21 i LSP_22 dla $N_{2a} + N_{2b} = 10$ bit. N_i jest rozmiarem książki kodowej LSP_2i, MSE_i jej błędem średniokwadratowym, a MSE całkowitym błędem średniokwadratowym.

3.6. Zaproponowane ulepszenia kodera LPC–10

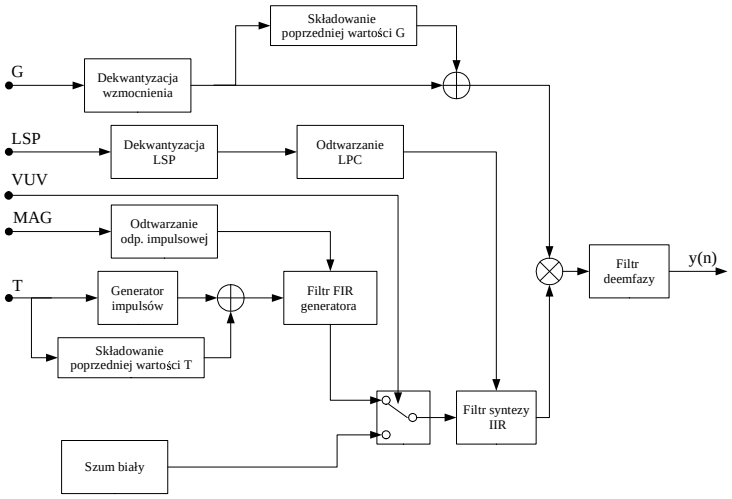
Na rysunkach 3.10 i 3.11 zostały przedstawione schematy blokowe rozszerzonego kodera i dekodera mowy standardu LPC–10. Najważniejszą modyfikacją zaimplementowanego algorytmu kodera mowy standardu LPC–10, było wykorzystanie przekształcenia współczynników LPC do postaci LSP, oraz ich kwantyzacja kwantyzatorem wektorowym. Przekształcanie współczynników LPC do postaci LSP i LSP na LPC zostało omówione w rozdziale 2. Zaprezentowany koder mowy wykorzystuje do konwersji funkcje ze źródeł modelu weryfikacyjnego kodera MPEG–4. Sam proces kwantyzacji wektorowej został zaimplementowany w postaci funkcji przeszukujących książki kodowe metodą *partial disance search* (definicja napisanej funkcji została zamieszczona w tabeli 2.3). Dodatkowo przygotowana została funkcja przeszukująca drzewo binarne w algorytmie podziałów (tabela 3.6)

Tabela 3.6. Funkcja przeszukująca binarne drzewo kodowe utworzone metodą techniki podziałów.

```
1  unsigned int coder::vector_quantisation( const vector *Vec,
2  const vector *CodeBook, int N )
3  {
4      float min1, min0;
5      int index=2;
6      for(int n=2; n<=N/2; n*=2 )
7      {
8          min0=MSE(Vec,&CodeBook[index-2]);
9          min1=MSE(Vec,&CodeBook[index-1]);
10         if((min0-min1)>0.0) index+=1;
11         index*=2;
12     }
13     return(index-2);
14 }
```



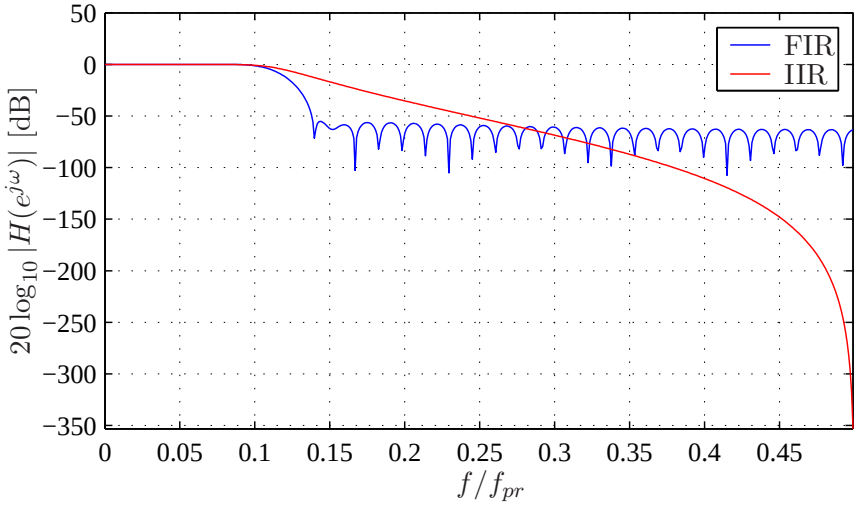
Rysunek 3.10. Schemat blokowy zaimplementowanego kodera mowy standardu LPC–10.



Rysunek 3.11. Schemat blokowy zaimplementowanego dekodera mowy standardu LPC–10.

Ponieważ sygnał pobudzenia ma istotny wpływ na jakość syntetyzowanego sygnału mowy, zaimplementowane zostały różne metody mające na celu ulepszenie algorytmu generacji pobudzenia. W najprostszym przypadku kodera mowy LPC–10 wyróżniane są dwa stany: dźwięczny i bezdźwięczny. Kluczowe znaczenie dla poprawnej syntezy sygnału w dekoderze stanowi poprawna detekcja stanu danej ramki (dźwięczna lub bezdźwięczna).

W przedstawionym koderze mowy filtr dolnoprzepustowy FIR o częstotliwości odcięcia $f_g = 900$ Hz i 65 współczynnikach, został zastąpiony filtrem IIR o siedmiu współczynnikach a_i i b_i w celu redukcji złożoności obliczeniowej. Na rysunku 3.12 zostały przedstawione charakterystyki amplitudowe dla obydwu filtrów.



Rysunek 3.12. Charakterystyki amplitudowo–częstotliwościowe filtrów FIR i IIR, gdzie f_{pr} jest częstotliwością próbkowania sygnału wynoszącą 8 kHz.

Jako prototyp filtra IIR został wybrany filtr Butterwortha ze względu na płaskie pasmo przenoszenia, wyznaczony standardową funkcją `butter(...)` programu `Matlab`. W tabeli 3.7 zostały zamieszczone współczynniki zaimplementowanego filtra dolnoprzepustowego. Zaprojektowany filtr ma mniejsze nachylenie niż filtr FIR o 65 współczynnikach, jednak w praktycznym zastosowaniu wyznaczanie częstotliwości tonu podstawowego dla sygnałów przefiltrowanych obydwoma filtrami daje identyczny efekt. Oznacza to, że zaproponowany filtr IIR spełnia założenia. W wyniku powyższej modyfikacji ilość potrzebnych obliczeń w procesie filtracji została zredukowana ok. 4.5-krotnie.

a	b
$0.619 \cdot 10^{-3}$	$1000 \cdot 10^{-3}$
$3.718 \cdot 10^{-3}$	$-3278.718 \cdot 10^{-3}$
$9.297 \cdot 10^{-3}$	$4859.545 \cdot 10^{-3}$
$12.396 \cdot 10^{-3}$	$-4041.615 \cdot 10^{-3}$
$9.297 \cdot 10^{-3}$	$1967.245 \cdot 10^{-3}$
$3.718 \cdot 10^{-3}$	$-527.142 \cdot 10^{-3}$
$0.619 \cdot 10^{-3}$	$60.444 \cdot 10^{-3}$

Tabela 3.7. Współczynniki zaimplementowanego filtra IIR.

W celu poprawy wykrywania informacji o dźwięczności/bezdźwięczności dla danej ramki, wykorzystana została metodę filtra resztkowego. W istocie jest to filtr odwrotny do filtra traktu głosowego (1.3):

$$G(z) = \frac{1}{H(z)} \tag{3.2}$$

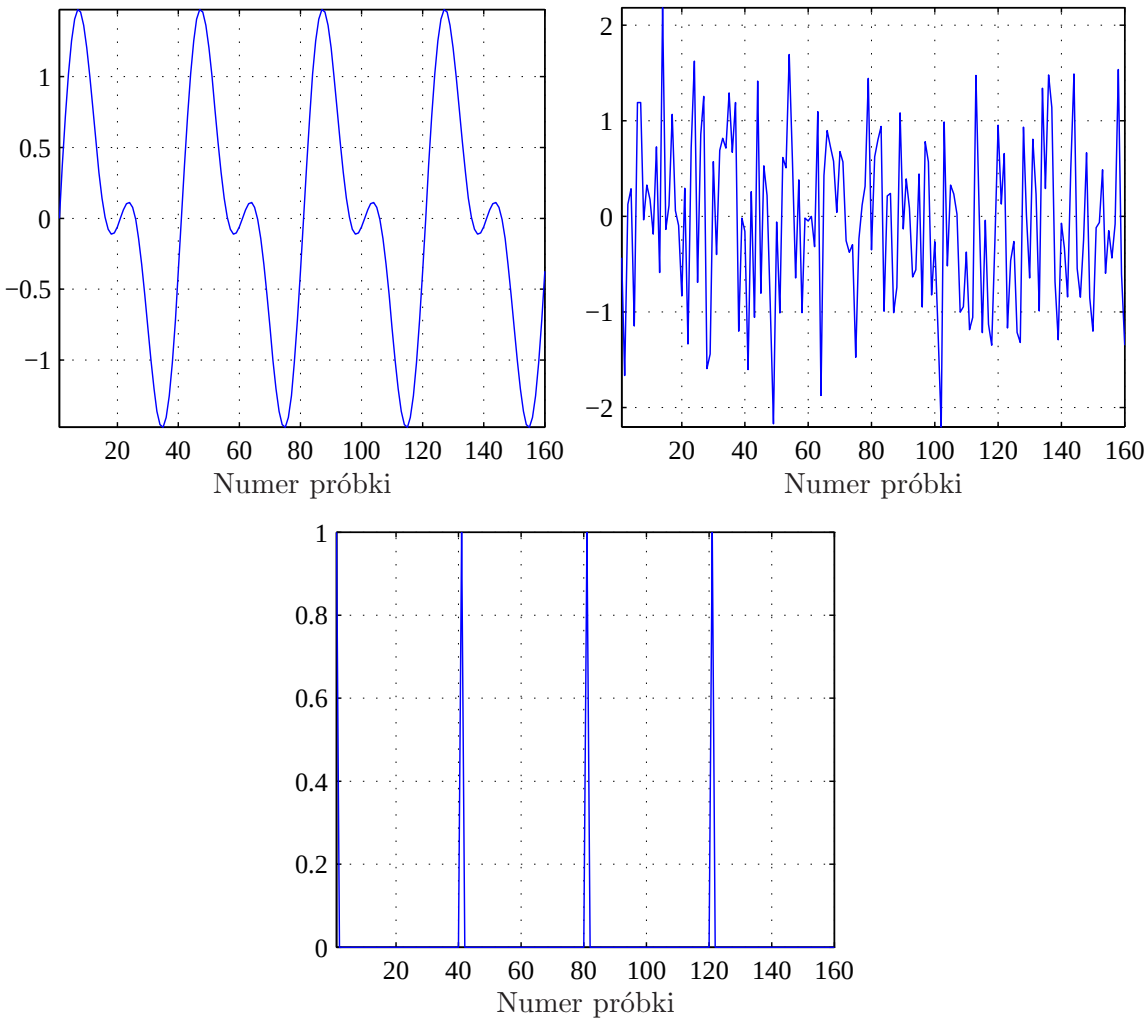
Przepuszczając sygnał wejściowy przez taki filtr, otrzymuje się błąd predykcji, spowodowany niezgodnością otrzymanego modelu. Sygnał resztkowy może być traktowany jako idealne pobudzenie filtra syntezy. Oznacza to, że filtrując sygnał filtrem $H(z)$ otrzymujemy idealny sygnał mowy tj. taki w którym nie ma żadnych zniekształceń. Sygnał resztkowy ma dużo mniejszą dynamikę niż sygnał oryginalny. W chwilach gdy filtr traktu głosowego jest pobudzany powietrzem wyrzucanym rytmicznie przez struny głosowe, w sygnale resztkowym pojawiają się okresowo impulsy, na podstawie których można wywnioskować charakter pobudzenia - dźwięczny bądź bezdźwięczny [1]. Bardziej złożone kodery mowy kompresują sygnał resztkowy za pomocą kodowania wielopasmowego, np. w koderach MBE (ang. *Multiband Excitation Coder*) [3] czy koderach mowy GSM [1].

W standardzie LPC–10 sygnał wejściowy jest poddawany autokorelacji mającej na celu oszacowanie czy sygnał jest dźwięczny czy bezdźwięczny oraz wyznaczenia ewentualnego tonu podstawowego. W przedstawionej tu modyfikacji kodera standardu LPC–10, autokorelacji poddawany jest sygnał resztkowy. Sprowadzenie modelu generacji mowy tylko do dwóch stanów może spowodować błędną ocenę parametru dźwięczny/bezdźwięczny (VUV) np. jeśli ramka trafi na przejście z ramki dźwięcznej na bezdźwięczną. Zmysł percepcji słuchowej człowieka odbiera tego rodzaju przekłamania szczególnie wyraźnie, zwłaszcza jeśli w wyniku błędnej detekcji pomiędzy dwiema ramkami dźwięcznymi znajdzie się ramka bezdźwięczna (bądź w odwrotnym przypadku: między ramkami bezdźwięcznymi znajdzie się ramka dźwięczna). Najprostszą metodą korekcji jest opóźnienie wysyłania danych o jedną ramkę i po wyznaczeniu następnej ramki sprawdzenie i ewentualne skorygowanie informacji o tonie podstawowym otrzymanym podczas analizy poprzednich danych. Stosowane są również metody wspomagające podjęcie decyzji o dźwięczności/bezdźwięczności bezpośrednio dla ana-

lizowanej ramki. Jedną z takich metod jest wyznaczanie parametru p (ang. *peakiness*) zdefiniowanego jako [5]:

$$p = \frac{\sqrt{\frac{1}{N}\sum_{n=0}^{N-1} e^2(n)}}{\frac{1}{N}\sum_{n=0}^{N-1} |e(n)|} \tag{3.3}$$

gdzie $e(n)$ jest sygnałem resztkowym, a N długością sygnału wynoszącą 240 próbek. Parametr ten pozwala na wykrywanie impulsów pojawiających się w sygnale resztkowym.



Rysunek 3.13. Porównanie wartości *peakiness* dla różnego rodzaju sygnałów: fragment sygnału okresowego $p = 1.16$ (górny lewy rysunek), szum gaussowski $p = 1.23$ (prawy górny rysunek), seria impulsów Diraca $p = 6.32$ (dolny rysunek).

Na rysunku 3.13 zostały przedstawione trzy rodzaje sygnałów i odpowiadające im

wartości parametru p. Widać, że im sygnał jest bardziej podobny do ciągu impulsów Diraca, tym większą wartość przyjmuje parametr p. W szczególności dąży do maksimum gdy ilość impulsów w danej ramce maleje [5]. Można zatem jego właściwości wykorzystać dwojako:

– do wykrywania dźwięczności/bezdźwięczności wykorzystując założenie, że sygnał dźwięczny jest w przybliżeniu ciągiem impulsów Diraca,
– do wykrywania przejścia między ramkami, gdy ilość impulsów gwałtownie maleje.

Ogólnie przyjętą wartością progową, poniżej której sygnał uznawany jest za bezdźwięczny (bądź w przypadku koderów o przepływnościach bitowych powyżej 4 kbit/s za prawdopodobnie bezdźwięczny), jest wartość $p = 1.3 \div 1.34$ [5]. Podczas przeprowadzanych testów implementacji kodera LPC–10, wartość $p = 1.34$ została uznana za wystarczającą. Przyjęcie mniejszej wartości powodowało wyraźne przekłamania częstotliwości tonu podstawowego.

W praktyce stosowane są dodatkowo również inne metody np. analiza ilości przejść sygnału pobudzenia przez zero w danej ramce. Ramka bezdźwięczna będzie posiadała stosunkowo dużą ilość przejść przez zero, podczas gdy ramka dźwięczna (a w szczególności dla pobudzenia szeregiem impulsów) charakteryzować się będzie małą ilością przejść przez zero.

Aby złagodzić wpływ gwałtownych zmian parametrów: okresowości T oraz wzmocnienia G przy przejściu między ramkami, w dekoderze została zaimplementowana interpolacja liniowa tych współczynników:

$$T_i = (1 - \alpha_i)T_{i-1} + \alpha_i T_i \tag{3.4}$$

$$G_i = (1 - \alpha_i)G_{i-1} + \alpha_i G_i \tag{3.5}$$

gdzie $i \in [0, 159]$ i oznacza numer aktualnej próbki, natomiast $\alpha_i = \frac{i}{160}$. Wartość T zaokrąglana jest do najbliższej wartości całkowitej. Do przechowywania informacji o poprzednim okresie, została wykorzystana dwuelementowa tablica danych typu `int`, natomiast dla wzmocnienia filtra z poprzedniej ramki dwuelementowa tablica typu `float`. Poniżej został zamieszczony fragment funkcji dekodera realizujący interpolację liniową:

Tabela 3.8. Fragment kodu źródłowego implementacji kodera LPC–10 realizującego interpolację liniową parametru G.

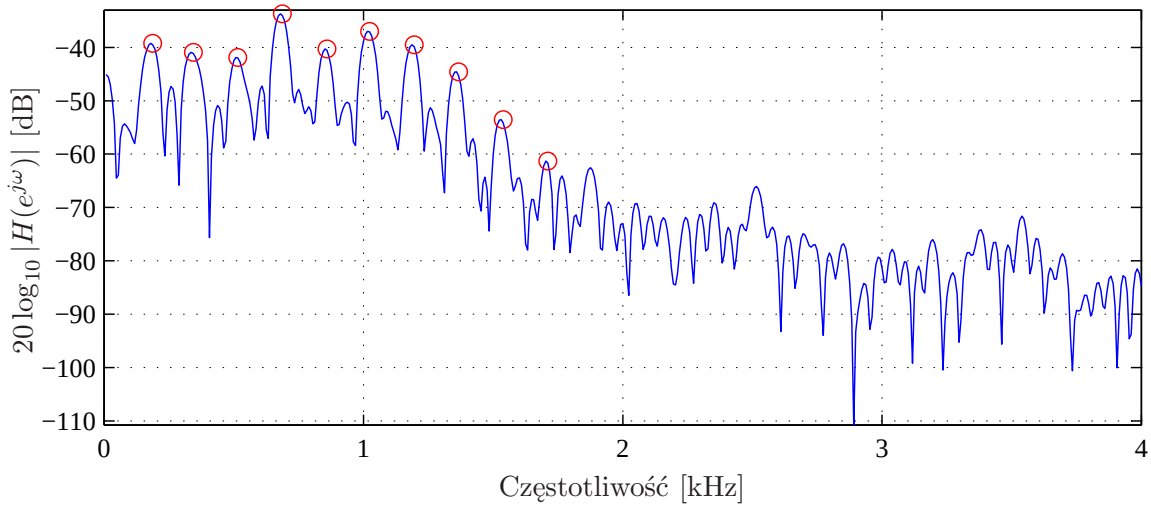
```
1  /* filtr syntezy */
2  a_fir[0]=1;
3  mvpIIR->filterdo(a_fir,y,err,tmp);
4  for(int i=0;i<160;i++)
5  {
6      if(G2[pitch_index]*G!=0)
7          tmp[i]*=(1-i/160.0)*G2[pitch_index]+(i/160.0)*G;
8      else
9          tmp[i]*=G;
10 }
```

W celu poprawienia jakości generowanego sygnału resztkowego, zaimplementowany model generacji mowy został rozbudowany o algorytm wyznaczania obwiedni widma sygnału pobudzenia, zaczerpnięty ze standardu MELP (ang. *Mixed Excitation Linear Prediction*) [5]. Podstawą idei poprawy jakości generowanego sygnału mowy jest fakt, że ramka dźwięczna nie składa się z prostych impulsów Diraca, ale raczej z ich splotu z pewną odpowiedzią impulsową. Interpretacją w dziedzinie częstotliwości jest fakt, że wielokrotności częstotliwości tonu podstawowego nie mają stałej amplitudy (rysunek 3.14)

Tabela 3.9. Fragment kodu źródłowego implementacji kodera LPC–10 realizującego interpolację liniową parametru *pitch*.

```
1 float Tx=BS->T;
2 for(int i=T_prev;i<160;i+=Tx)
3 {
4       if(T2[pitch_index]*BS->T!=0)
5           Tx=round((1-i/160.0)*T2[pitch_index]+(i/160.0)*(BS->T));
6       T_prev=(BS->T)-160+i;
7       err[i]=1.0;
8 }
```

Ponieważ ton podstawowy w standardzie LPC–10 zawiera się w przedziale $T \in [20, 160]$ próbek (co odpowiada zakresowi częstotliwości $f_T \in [50, 400]$ Hz), zatem wyznaczenie 10 pierwszych wartości wielokrotności częstotliwości tonu podstawowego wystarczy w zupełności aby pokryć zakres częstotliwości $f \in (500, 4000)$ Hz.



Rysunek 3.14. Widmo dźwięcznej ramki sygnału mowy z naniesionymi dziesięcioma wielokrotnościami częstotliwości tonu podstawowego.

Poszukiwanie obwiedni widma odbywa się przez pobieranie wartości amplitud wyznaczonych za pomocą FFT w punktach $f_T, 2f_T, \ldots, 10f_T$, gdzie F_T jest częstotliwością tonu podstawowego. W rzeczywistości wielokrotności częstotliwości tonu pod-

stawowego f_T nie znajdują się w równych odległościach. Dlatego też podczas wyznaczania ich amplitud widmowych należy przeszukać pewien zakres częstotliwości $\texttt{round}(Mi/pitch) \pm 5$, gdzie M jest długością sygnału z którego obliczone zostało widmo za pomocą FFT, a $i = 1, \ldots, 10$. Wyznaczona w ten sposób obwiednia może zostać poddana kwantyzacji i przesłana do dekodera. Poniżej został zamieszczony fragment funkcji wyznaczającej obwiednię widma, napisany dla potrzeb zaimplementowanego kodera mowy.

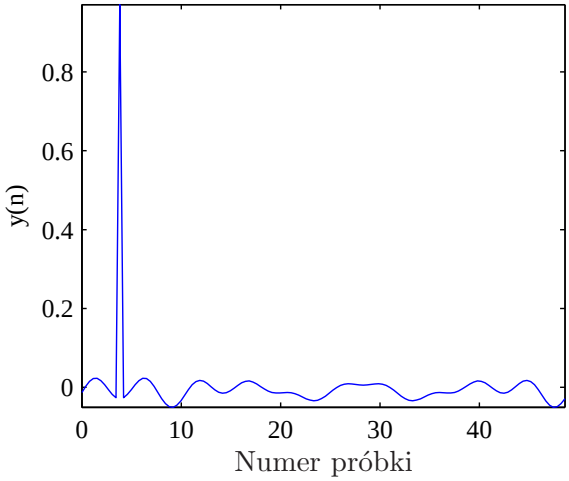
Tabela 3.10. Fragment kodu źródłowego wyznaczającego obwiednię widma dla ciągu impulsów.

```
1  float k0=512.0f/(float)pitch;
2  zesp *Z;
3  Z=moj.fft(input);
4  float tmp,peak=0.0,sum=0.0;
5  int freq=0;
6  for( int m=1; m<=10; m++ )
7  {
8       freq=(int)round(m*k0);
9       if(freq>255) magn[m-1]=peak;
10      else
11      {
12           peak=0;
13           tmp=abs(Z[freq]);
14           int i;
15           for(i=freq-5;i<freq+5;i++)
16           {
17                if(i>255) break;
18                tmp=abs(Z[(int)round(i)]);
19                if(tmp>peak) peak=tmp;
20           }
21           magn[m-1]=peak;
22      }
23      sum+=magn[m-1];
24 }
25 if(sum==0.0f)
26 for(int i=1;i<=10;i++) magn[i-1]=1.0f;
```

W dekoderze po odtworzeniu obwiedni widma, proces generacji sygnału pobudzenia dźwięcznego jest wykonywany zgodnie z przedstawioną poniżej procedurą:

1. Usunięcie składowej stałej.
2. Unormowanie wartości amplitud widma do $P_{mag} = \sqrt{\frac{1}{10}\sum_i mag(i)}$, gdzie P_{mag} jest wartością skuteczną wyznaczonego widma, a $mag(i)$ są unormowanymi wartościami modułów widma sygnału [5].
3. Dodanie $(T-20)$ zer aby długość odpowiedzi impulsowej była równa okresowi tonu podstawowego T.
4. Dodanie odbitych 10 współczynników widma na koniec sygnału w celu zachowania symetrii widma.
5. Obliczenie odwrotnej dyskretnej transformaty Fouriera.
6. Przesunięcie kołowe o 10 próbek.
7. Splot szeregu impulsów Diraca z otrzymaną odpowiedzią impulsową.

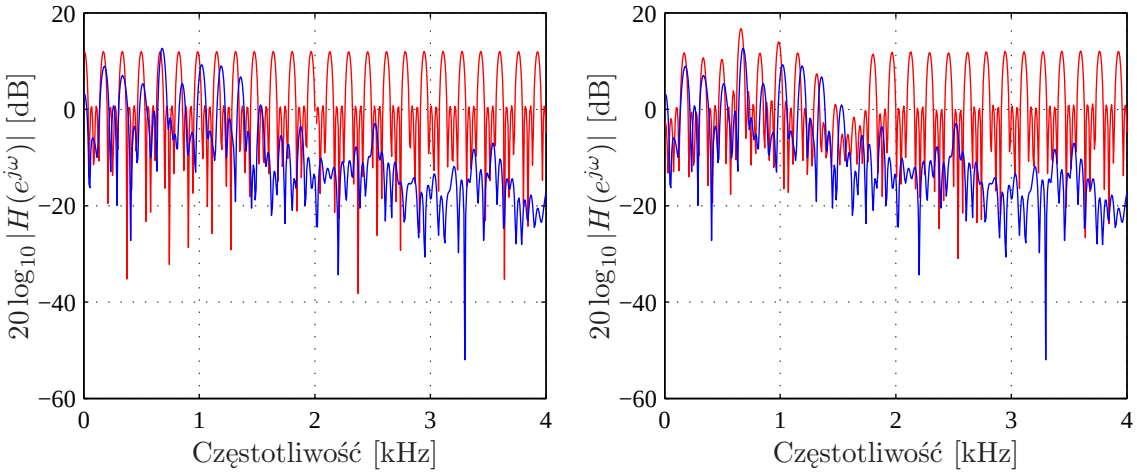
Przykładowa odpowiedź impulsowa wyznaczona powyższą metodą z 10-punktowej obwiedni widma sygnału resztkowego, jest przedstawiona na rysunku 3.15.



Rysunek 3.15. Odpowiedź impulsowa wyznaczona na podstawie 10–punktowej obwiedni widma po przesunięciu kołowym.

Na podstawie wyznaczonej odpowiedzi impulsowej został wygenerowany sygnał, którego widmo zostało przedstawione na rysunku (3.16). W porównaniu do widma sygnału będącego szeregiem impulsów Diraca widać, że sygnał w interesującym z punktu widzenia kodera mowy zakresie częstotliwości $f \in (0, 1000)$ Hz jest bardziej zbliżony do widma sygnału resztkowego (sygnału idealnego). Efektem dodania informacji o widmie obwiedni sygnału resztkowego jest zatem polepszenie naturalności i jakości sygnału mowy po rekonstrukcji dla ramek dźwięcznych. Kosztem tego jest zwiększenie zapotrzebowania na przepływność bitową. W przeprowadzonych na potrzeby tej pracy badaniach, został zbadany wpływ splotu odpowiedzi impulsowej z sygnałem pobudzenia na jakość sygnału na wyjściu kodera. Wyniki omówione w rozdziale 4 potwierdzają, że jakość syntetyzowanej w ten sposób mowy jest wyraźnie lepsza od pobudzenia zwykłym ciągiem impulsów jednostkowych.

Cały koder został napisany z wykorzystaniem możliwości programowania obiektowego, dodatkowo dodane zostały klasy pomocnicze mające na celu obsłużenie funkcji odpowiedzialnych za ustawianie podstawowych parametrów kodera (częstotliwość próbkowania, rząd filtra traktu głosowego, długość zastosowanego okna) i wczytywanie parametrów z których koder korzysta. Kody źródłowe kodera i dekodera, zostały zamieszczone w Dodatku F wraz z opisem plików nagłówkowych. Wszystkie funkcje poza funkcjami przekształcającymi współczynniki LPC do LSP i LSP do LPC (zaczerpniętymi ze standardu MPEG–4) zostały napisane przez autora, w szczególności funkcje filtrów cyfrowych oraz klasa odpowiedzialna za obliczanie FFT metodą radix-2 [1].



Rysunek 3.16. Porównanie widm ramki sygnału dźwięcznego odtworzonego przez zsyntetyzowanie ciągów impulsów Diraca (rysunek lewy) oraz splot ciągu impulsów Diraca z wyznaczoną odpowiedzią impulsową (rysunek prawy) z widmem sygnału resztkowego.

Tak zaimplementowany koder o przepływności bitowej wynoszącej 2.4 kbit/s wykonuje operacje kodowania z szybkością ok. 200 ramek na sekundę na procesorze AMD Athlon 1.6 GHz. Oznacza to, że opóźnienie wprowadzane przez koder mowy jest w granicach 5 ms. Analiza kodu wykazała, że najwięcej czasu procesora zajmuje procedura przeszukiwania książki kodowej oraz wyznaczanie funkcji autokorelacji. Optymalizacja tych funkcji pod kątem szybkości mogłaby zmniejszyć opóźnienie co najmniej o połowę. Dla porównania dekoder pracuje z szybkością ok. 2600 ramek na sekundę co odpowiada opóźnieniu 0.3 ms. Biorąc pod uwagę fakt, że opóźnienie związane z samą specyfiką schematu analiza synteza wynosi 40 ms (dodatkowe 20 ms związane z ciągłością parametru VUV), można uznać, że otrzymany wynik jest bardzo dobry.

Rozdział 4

Testy porównawcze jakości kompresji mowy

Niniejszy rozdział poświęcony jest oszacowaniu jakości kompresji sygnału mowy oraz wpływu na tą jakość książek kodowych, których wyznaczenie zostało opisane w poprzednim rozdziale. Przeprowadzone badania miały na celu określenie efektywności i jakości zastosowanych kwantyzatorów oraz wpływu modyfikacji kodera LPC–10 na jakość generowanego dźwięku. Omówione zostaną metody oceny sygnału mowy na podstawie subiektywnych miar jakości. Szczególny nacisk zostanie postawiony na omówienie subiektywnych miar jakości opartych na testach odsłuchowych. Omówione zostaną również metody pomiarów jakości metodami obiektywnymi wykonanych w celu weryfikacji otrzymanych wyników. Zebrane dane posłużą do porównania możliwości zastosowanych metod pomiarowych.

4.1. Miary jakości mowy

W celu określenia jakości wyjściowego sygnału mowy przyjmowane są kryteria takie jak:

- zrozumiałość zsyntetyzowanej mowy,
- naturalność czyli brak artefaktów tj. echo, przerywania czy szum wpływające negatywnie na odbiór,
- możliwość zidentyfikowania rozmówcy na podstawie „barwy" jego głosu.

Wyznaczenie wskaźnika jakości może opierać się zarówno na ocenie subiektywnej jak i obiektywnej. Wyznaczenie współczynnika jakości musi odbyć się z uwzględnieniem szeregu warunków co do sygnału. W szczególności są to:

- występowanie szumu tła (tło dźwiękowe nie będące mową np. muzyka),
- różny poziom głośności sygnału,
- zróżnicowana grupa rozmówców.

Często są przedstawiane dodatkowe kryteria związane z samą transmisją sygnału np. gubienie pakietów, opóźnienia wprowadzane przez koder oraz urządzenia transmisyjne (np. routery) czy też przepustowość łącza. W tej pracy jakość kodera mowy nie

będzie badana pod tym kątem. Wyznacznikiem jakości kompresji sygnału mowy oraz wpływu na nią wygenerowanych książek kodowych, będą kryteria zrozumiałości oraz naturalności zsyntetyzowanej mowy.

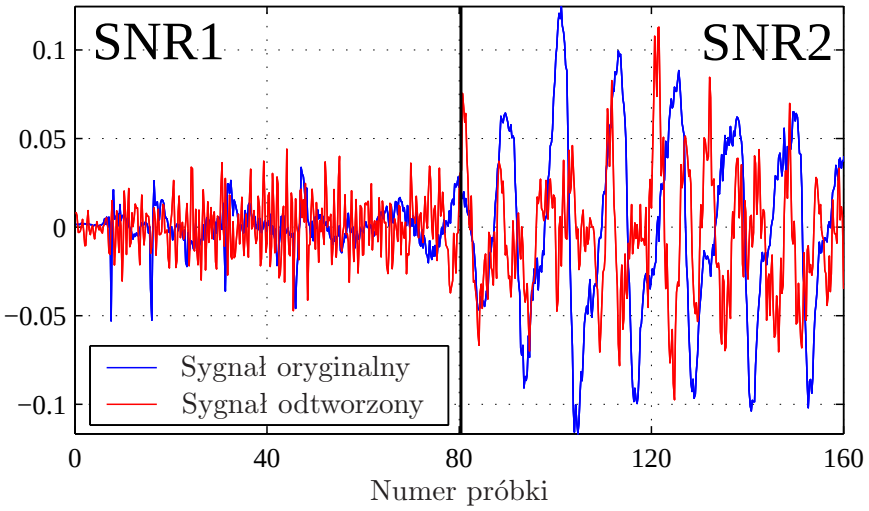
4.1.1. Obiektywna miara jakości

Nie istnieje uniwersalna obiektywna miara jakości taka jak np. PSNR (ang. *peak signal-to-noise ratio*) w analizie obrazów [3]. Stosowanie jako miary jakości SNR zdefiniowanego równaniem (2.5) w przypadku sygnału będącego ludzką mową, jest ograniczone przez właściwości wynikające z samej definicji SNR. Niedopasowania pomiędzy kolejnymi ramkami czy przesunięcia fazy (na które ucho ludzkie jest praktycznie niewrażliwe) wpływają na wartość oceny, która może się znacznie różnić od oceny związanej z percepcją dźwięku [6]. Pewnym udoskonaleniem jest uśrednianie wartości SNR z pewnego dłuższego okresu czasu (ang. *Segmental Signal-to-Noise Ratio*) zdefiniowane jako:

$$\mathrm{SSNR} = \frac{1}{N}\sum_{m=1}^{N}\mathrm{SNR}(m) \tag{4.1}$$

gdzie $\mathrm{SNR}(m)$ jest wartością SNR z kolejnej ramki odpowiadającej 20 ms, a N jest całkowitą ilością ramek w sygnale mowy, którego jakość jest badana.

SSNR jako wskaźnik jakości sygnału mowy, jest znacznie bliższy subiektywnej mierze jakości, ponieważ uwzględnia wpływ ramek mowy o niskiej energii. Przykładem mogą być ramki ciszy, które zakłócone pogarszają jakość sygnału.



Rysunek 4.1. Przykład przedstawiający przewagę SSNR nad zwykłym SNR. SNR jest wyznaczone dla próbek $i = 0, \ldots, 160$, SNR1 dla $i = 0, \ldots, 80$, SNR2 dla $i = 80, \ldots, 160$, a SSNR=(SNR1+SNR2)/2.

Na rysunku 4.1 została pokazana zaleta stosowania miary SSNR w porównaniu do SNR. Po podzieleniu sygnału na dwie ramki o różnym wzmocnieniu, zostały wyliczone wartości: SNR1=0.325 dB i SNR2=0.680 dB. Po uśrednieniu otrzymana została

wartość SSNR=0.503 dB. Natomiast wartość SNR wyznaczona dla całego sygnału wyniosła 0.653 dB i była bardzo bliska wartości SNR2. Widać, że wartości sygnału o niskiej energii dla próbek $i = 0, \ldots, 80$ praktycznie nie wpływają na wartość SNR wyznaczonego z (2.5).

Ponieważ informacja o fazie nie jest pożądana przy pomiarze jakości metodą obiektywną, zaproponowana została metoda obliczenia SSNR w oparciu moduł widma sygnału mowy. Tak określony wskaźnik jakości można zdefiniować jako:

$$\mathrm{SSNR}_{FFT} = \frac{1}{N}\sum_{m=1}^{N}\mathrm{SNR}_{FFT}(m) \tag{4.2}$$

gdzie:

$$\mathrm{SNR}_{FFT} = 10\log_{10}\left(\frac{\sum_{n}|FFT(x(n))|^2}{\sum_{n}\left(|FFT(x(n))| - |FFT(y(n))|\right)^2}\right) \tag{4.3}$$

Alternatywą dla miar jakości opartych na SNR, są miary oparte na metrykach bazujących na odległościach pomiędzy współczynnikami LPC sygnałów, takich jak logarytm ilorazu wiarygodności znany również jako funkcja wiarygodności (ang. *likelihood ratio*)[4, 16, 20]. Tego rodzaju miary odległości są ściśle związane z wyznaczanymi współczynnikami LPC. Przyjmując, że dla sygnału $x(n)$ będącego sygnałem wejściowym, a_x oznaczają współczynniki LPC wyznaczone np. metodą Durbina–Levinsona, a dla sygnału $y(n)$ będącego odtworzonym sygnałem po operacji kwantyzacji i odwrotnej do kwantyzacji, a_y oznaczają współczynniki LPC sygnału odtworzonego, można zapisać jako:

$$D_{LR}(n) = \frac{a_y^T R_x a_y}{a_x^T R_x a_x} \tag{4.4}$$

bądź

$$D_{LR}(n) = \frac{a_x^T R_y a_x}{a_y^T R_y a_y} \tag{4.5}$$

gdzie R_x, R_y są macierzami autokorelacji (1.6) dla sygnału wejściowego $x(n)$ i wyjściowego $y(n)$. Tak zdefiniowane wartości $D_{LR1}(n)$ i $D_{LR2}(n)$ są funkcjami wiarygodności. Miara jakości będąca logarytmem ilorazu wiarygodności (ang. *log–likelihood ratio*, LDR), zdefiniowana jest jako

$$LDR = 10\log_{10}(\bar{D}_{LR}) \tag{4.6}$$

gdzie $\bar{D}_{LR}$ jest wartością średnią $D_{LR}(n)$ ze wszystkich ramek.

Inną obiektywną miarą jakości opartą na współczynnikach LPC, jest miara odległości cepstralnych (ang. *cepstral distance*, CD). Bazuje ona na obliczaniu odległości pomiędzy obwiedniami widma oryginalnego i odtworzonego sygnału mowy. Dla zadanej ramki sygnału, miarę odległości można przedstawić jako [4, 16]:

$$D_{cd}(n) = D_B\sqrt{2\sum_{i=1}^{p}(c_i^{(x)} - c_i^{(y)})^2} \tag{4.7}$$

gdzie p jest rzędem filtra traktu głosowego, $D_B = \frac{10}{\ln(10)}$ jest współczynnikiem skalującym, a $c_i^{(x)}, c_i^{(y)}$ są współczynnikami cepstralnymi LPC odpowiednio sygnału: $x(n)$ i $y(n)$ zdefiniowanymi jako:

$$c_1 = -a_1 \tag{4.8}$$

$$c_i = -a_i - \sum_{k=1}^{i-1} \frac{i-k}{i} c_{i-k} a_k \tag{4.9}$$

W równaniu (4.9) a_i są współczynnikami filtra traktu głosowego oraz $i = 2, \ldots, p$. Na podstawie wartości ze wszystkich ramek, dane są uśredniane dla całego sygnału, dając tym samym wskaźnik średniej odległości cepstralnej będący miarą jakości $CD = \bar{D}_{cd}(n)$. Napisany na potrzeby tego rodzaju pomiarów w programu `Matlab` skrypt, został zamieszczony w Dodatku D. W porównaniu do miar opartych na SNR, tego rodzaju metryka lepiej odpowiada mierzonym wartością jakości subiektywnej opartych na MOS [4]. W przeciwieństwie do miar opartych na SNR, miary wykorzystujące współczynniki LPC mają tym lepszą jakość im wartość wskaźnika jakości jest mniejsza.

4.1.2. Subiektywna miara jakości

Mean Opinion Score (znany również jako ang. *Absolute Category Rating*, ACR) jest ogólnie przyjętym standardem określania jakości dźwięku (zwłaszcza w telefonii VoIP), zatwierdzonym przez ITU (ang. *International Telecommunication Union*) jako zalecenie P.800.1 (03/03) [16]. Metoda MOS będąca subiektywną miarą jakości mowy, bazuje na opinii pewnej grupy osób poddanej testom odsłuchowym. Uczestnicy testów mają przydzielić prezentowanym próbkom dźwiękowym oceny w pięciostopniowej skali: znakomita (5), dobra (4), średnia (3), słaba (2), zła (1). Na podstawie zebranych ocen obliczana jest średnia będąca subiektywną miarą jakości.

Innym podejściem wyznaczenia subiektywnej miary jakości, jest *Degradation Mean Opinion Score* (DMOS) znany również jako *Degradation Category Rate* (DCR). Podczas testów porównywane są ze sobą dwie próbki dźwiękowe - oryginalna i zsyntetyzowana, a jako wynik podawany jest stopień pogorszenia jakości w skali 1-5.

Modyfikacja powyższej metody nazywana *Comparision Category Rate* (CCR) a polegająca na niesprecyzowaniu badanej osobie który z porównywanych dźwięków jest oryginalny a który zsyntetyzowany, pozwala na eliminację błędnej oceny zasugerowanej jakością sygnału źródłowego. Stosowana w tym przypadku jest sześciostopniowa skala jakości: (3) o wiele lepsza jakość pierwszej próbki, (2) lepsza jakość, (1) niewiele lepsza jakość, (0) brak różnicy między obydwoma próbkami, (-1) niewiele gorsza jakość, (-2) gorsza jakość, (-3) o wiele gorsza jakość. Nadmienić należy, że wykonanie testów mających na celu wyznaczenie współczynnika MOS jest dosyć uciążliwe i czasochłonne. Wymaga zaangażowania znacznej grupy osób (minimum 16 w większości testów [5]) wybranej spośród całej docelowej grupy odbiorców.

W ostatnich latach rozwijane są bardziej zaawansowane metody pomiaru obiektywnego współczynnika jakości, takie jak *Perceptual Speech Quality Measure* (PSQM, ITU–T rec. P.861) czy *Perceptual Evaluation of Speech Quality* (PESQ, ITU–T rec. P.862). Pomimo swojej efektywności ich zastosowanie rozciąga się poza sam pomiar

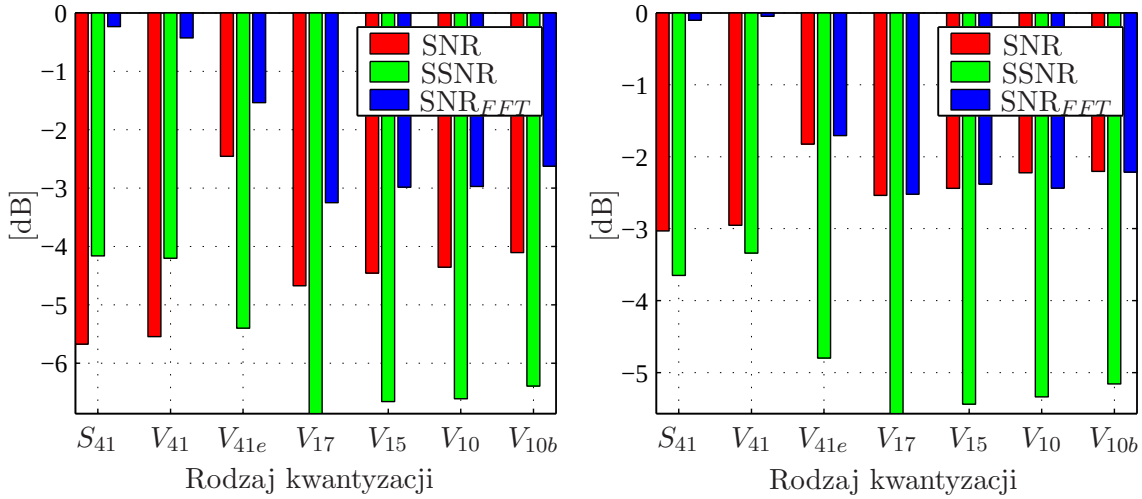
jakości mowy i uwzględnia wpływ zakłóceń w trakcie transmisji – z tego powodu nie są rozważane w tej pracy.

4.2. Porównanie jakości kompresji mowy dla różnych książek kodowych

Dla 14 różnych próbek dźwiękowych (siedem fragmentów mowy kobiecej i siedem fragmentów mowy męskiej), zostały przeprowadzone testy mające na celu wyznaczenie zarówno obiektywnego jak i subiektywnego współczynnika jakości. Wyniki zostały zamieszczone w tabeli oraz naniesione na wykres zależności subiektywnego wskaźnika jakości mowy w funkcji obiektywnego wskaźnika jakości mowy. W celu wyznaczenia wartości SNR i SSNR został wykorzystany kod napisany w języku skryptowym programu `Matlab`, zamieszczony w Dodatku D. Poniżej zostały zamieszczone wyniki badań otrzymane dla różnych wielkości książek kodowych kwantyzatorów: skalarnego oraz wektorowego jedno i dwupoziomowego. Zawierają one zarówno wyniki będące obiektywnymi wskaźnikami jakości otrzymanymi ze wzorów (2.5), (4.1) oraz (4.2) jak i subiektywnymi wskaźnikami jakości.

4.2.1. Obiektywna jakość wyznaczonych książek kodowych

W celu określenia jakości wyznaczonych dla kilku średnich bitowych książek kodowych, zostały wyznaczone wartości SNR, SSNR oraz SSNR_{FFT} względem sygnału wejściowego będącego plikiem wave nie poddanym schematowi kompresji metodą analiza–synteza. Na rysunku 4.2 zostało przedstawione porównanie SNR, SSNR i SSNR_{FFT} w formie wykresu dla próbek mowy kobiecej oraz męskiej.



Rysunek 4.2. Obiektywna ocena jakości bazująca na SNR, SSNR i SSNR_{FFT} względem sygnału wejściowego dla sygnału mowy kobiecej (rysunek lewy) oraz męskiej (rysunek prawy).

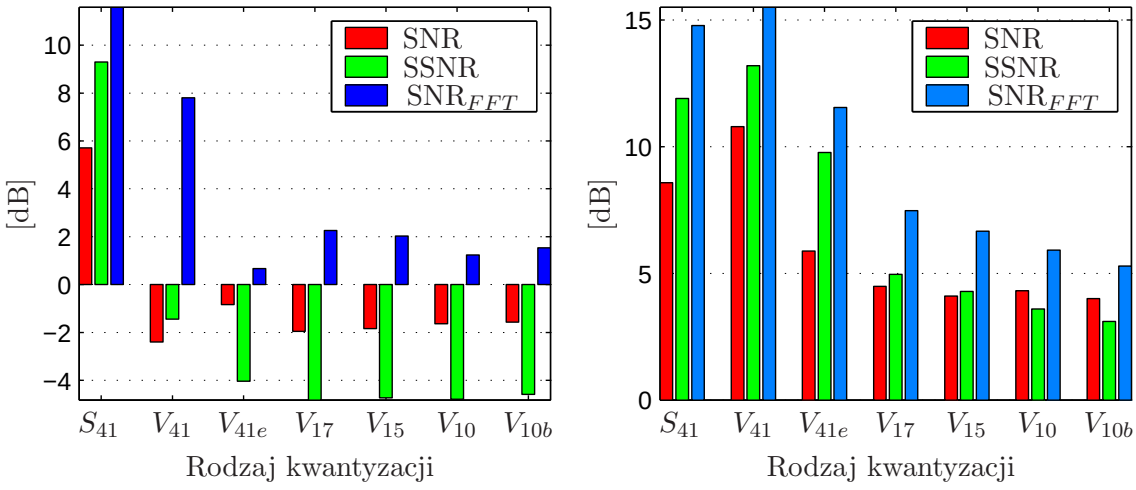
Kolejne indeksy na przedstawianych wykresach dla pomiarów jakości metodą obiektywną, odpowiadają kwantyzatorom:

Oznaczenie	Opis kwantyzatora
S_{41}	kwantyzator skalarny 41–bitowy
V_{41}	kwantyzator wektorowy 41–bitowy dwupoziomowy
V_{41e}	kwantyzator wektorowy 41–bitowy z obwiednią widma
V_{17}	kwantyzator wektorowy 17–bitowy dwupoziomowy
V_{15}	kwantyzator wektorowy 15–bitowy dwupoziomowy
V_{10b}	kwantyzator wektorowy 10–bitowy jednopoziomowy
V_{10}	kwantyzator wektorowy 10–bitowy dwupoziomowy

Tabela 4.1. Opis zastosowanych kwantyzatorów w pomiarach jakości.

Kwantyzatory opisane w tabeli C.1 zostały wybrane tak, by przepływność bitowa kodera mowy była jak najmniejsza przy zachowaniu zrozumiałości treści. W tym celu zaproponowane zostały książki kodowe o coraz mniejszym rozmiarze i różnej ilości poziomów.

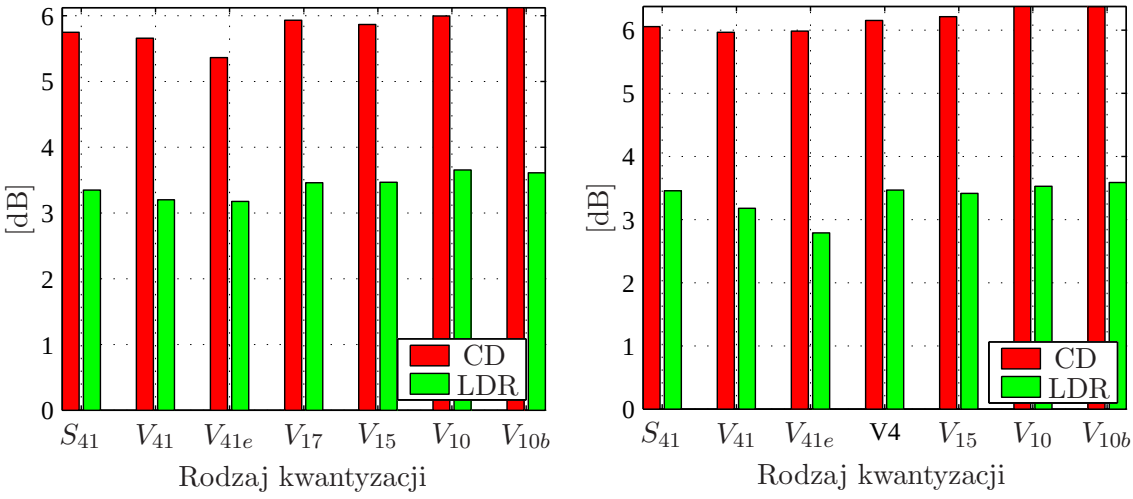
Porównanie sygnału wyjściowego dekodera z nieprzetworzonym sygnałem wejściowym nie jest dobrym wyznacznikiem jakości wygenerowanych książek kodowych, ze względu na uproszczenia generacji sygnału resztkowego, które znacząco wpływają na kształt zsyntetyzowanego sygnału mowy. By zminimalizować wpływ pobudzenia na wynik pomiarów, sygnał skompresowany został porównany z sygnałem poddanym schematowi analiza–synteza bez kwantyzacji współczynników.



Rysunek 4.3. Obiektywna ocena jakości bazująca na SNR, SSNR i SSNR$_{FFT}$ względem sygnału poddanego analizie i syntezie bez kwantyzacji dla sygnału mowy kobiecej (rysunek lewy) oraz męskiej (rysunek prawy).

Na rysunku 4.3 zostały przedstawione wyniki pomiarów jakości metodami opartymi na SNR. Można je interpretować jako wpływ kwantyzacji na jakość syntetyzowanej mowy. Z wykresów widać, że kwantyzator wektorowy o rozmiarze książki kodowej wynoszącym 41 bitów, ma zbliżoną jakość do kwantyzatora skalarnego o takim samym rozmiarze książki kodowej. Dla sygnału mowy męskiej, kwantyzator wektorowy wykazuje wzrost jakości o średnio 1.5 dB w stosunku do sygnału poddanego kwantyzacji skalarnej. Przewagą tego rodzaju kwantyzacji jest fakt, że zmniejszenie wielkości książki kodowej kwantyzatora skalarnego przy zachowanej zrozumiałości przekazu nie jest już możliwe w przeciwieństwie do kwantyzatora wektorowego. Tak jak można się było tego spodziewać, najgorsze rezultaty daje dwupoziomowa książka kodowa o rozmiarze 10 bitów. Interesujący wydaje się być fakt, że pomimo iż całkowity błąd średniokwadratowy dla 10-bitowej książki kodowej jednopoziomowej był czterokrotnie mniejszy niż w przypadku książki dwupoziomowej, to pomiary jakości oparte na SNR praktycznie nie wykazują przewagi jakościowej książki jednopoziomowej nad dwupoziomową. Na podstawie dokonanych pomiarów widać również, że przyjęta miara jakości oparta na (4.2) nie wnosi żadnych istotnych informacji o jakości dźwięku w stosunku do (2.5) i (4.1).

W celu porównania różnych miar jakości, przeprowadzone zostały również badania współczynników jakości bazujących na ilorazie wiarygodności oraz odległościach cepstralnych opisanych w podrozdziale 4.1.1. Wyznaczanie miar jakości tymi metodami było stosunkowo prostym zadaniem przy wykorzystaniu standardowych funkcji programu `Matlab`. Ponieważ przy pomiarze jakości nie ma czynnika krytycznego którym jest czas, zatem interpreter poleceń języka `Matlab` nadał się do tego celu bardziej niż aplikacja napisana w języku C/C+, nie ograniczając przy tym uniwersalności zastosowania zaimplementowanych algorytmów.



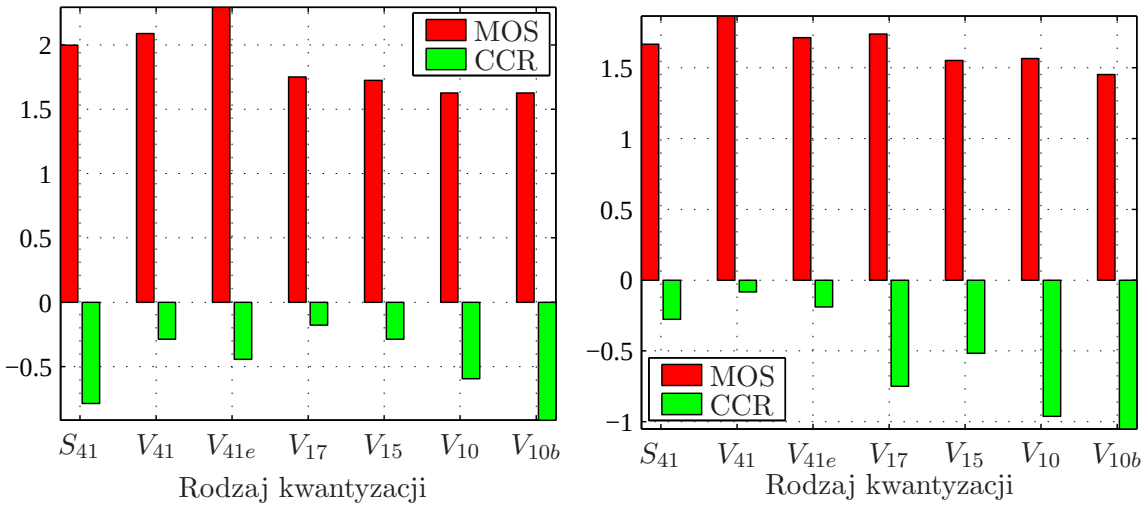
Rysunek 4.4. Porównanie metod pomiaru jakości metodą obiektywną na podstawie metryk *cepstral distance* oraz *log–likelihood ratio* dla sygnału mowy kobiecej (rysunek lewy) oraz męskiej (rysunek prawy).

Na rysunku 4.4 zostały wykreślone wskaźniki jakości LDR i CD dla różnych rozmiarów książek kodowych. Z przedstawionych na rysunku wykresów widać, że wyniki są zgodne z przewidywaniami. Szczególnie widoczna jest przewaga jakości sygnału V_{41e} poddanego kwantyzacji kwantyzatorem wektorowym o książce kodowej długości 41 bitów nad innymi kwantyzatorami, dla mowy kobiecej składającej się z dużej ilości ramek dźwięcznych. W przypadku mowy męskiej jakość jest porównywalna do jakości analogicznego kwantyzatora wektorowego bez obwiedni widma. Podobnie jak w przypadku miar jakości opartych na SNR, praktycznie nie widać różnic pomiędzy 10–bitowymi książkami kodowymi: jedno i dwupoziomowymi.

4.2.2. Subiektywna jakość wyznaczonych książek kodowych

W celu wyznaczenia wskaźnika jakości wygenerowanych książek kodowych, zostały wykonane testy odsłuchowe plików dźwiękowych przetworzonych koderem LPC–10 i poddanych kwantyzacji dla różnych przepływności bitowych. Punktem odniesienia było wyznaczenie wartości MOS dla fragmentów mowy poddanych schematowi analiza-synteza bez kwantyzacji współczynników, które w przeprowadzanych testach powinny mieć najlepszą jakość z badanych próbek. Grupa osób oceniających jakość dźwięku została wybrana tak, by odpowiadała jak najszerszej grupie docelowych odbiorców.

Na rysunku 4.5 zostały wykreślone wyniki pomiarów jakości MOS i CCR będące wartościami średnimi wyników przeprowadzonych testów odsłuchowych. Na rysunku widoczna jest przewaga kwantyzacji wektorowej sygnału V_{41} nad kwantyzacją skalarną sygnału S_{41} przy takim samym rozmiarze książki kodowej wynoszącym 41 bitów. Pomimo, że zastosowany został tylko dwupoziomowy kwantyzator wektorowy, to całkowita poprawa jakości wynosi ok. 0.2 punkta w skali MOS.



Rysunek 4.5. Porównanie jakości mowy wyznaczonej na podstawie wartości MOS i CCR dla mowy kobiecej (rysunek lewy) i męskiej (rysunek prawy).

Zastosowanie dodatkowej informacji o obwiedni widma pobudzenia, poprawiło znacznie jakość sygnału oznaczonego na rysunku 4.5 jako V_{41e} względem sygnału V_{41} o ok. 0.3 punkta w skali MOS. Widoczne jest to szczególnie w przypadku sygnału mowy kobiecej zawierającego więcej ramek dźwięcznych. W przypadku sygnału mowy męskiej z przewagą ramek bezdźwięcznych nie ma poprawy jakości. Dla praktycznych zastosowań można przyjąć, że książka kodowa oparta na 41–bitowym kwantyzatorze wektorowym dwupoziomowym wprowadza najmniejszy błąd średniokwadratowy sygnału w procesie kwantyzacji i odwrotnym do kwantyzacji. Wykorzystanie książki kodowej rozmiaru 41 bitów z obwiednią widma pobudzenia wymagało by stworzenia książki kodowej jej dedykowanej.

Na podstawie wykonanych pomiarów zostały wykreślone zależności pomiędzy otrzymanymi subiektywnymi a obiektywnymi miarami jakości mowy. Na rysunkach 4.6 oraz 4.7 zostały przedstawione wykresy zależności pomiędzy obiektywnymi miarami jakości mowy opartymi na SNR, SSNR i SSNR_{FFT} a wartościami MOS i CCR. Z rysunków 4.6 oraz 4.7 widać, że pomiędzy miarami obiektywnymi opartymi na SNR, a miarami subiektywnymi opartymi na współczynnikach MOS i CCR, praktycznie nie ma żadnej korelacji.

W przypadku porównania miar *cepstral distance* i *log–likelihood* z miarami MOS (rysunek 4.8 oraz rysunek 4.9) oraz CCR (rysunek 4.10 i rysunek 4.11), można zauważyć pewną zależność pomiędzy współczynnikami obiektywnymi a subiektywnymi. Jak podaje [4] pomiędzy współczynnikami MOS a *cepstral distance* istnieje ścisły związek. W zakresie wartości $CD \in [5, 6]$ otrzymane wartości pasują do krzywej regresji wyznaczonej doświadczalnie dla różnego rodzaju koderów. Potwierdza to słuszność zastosowanych metod pomiaru bazujących na analizie LPC jak i poprawność dokonanych pomiarów. Korelacja współczynników CD z wartościami CCR oraz LDR z MOS i CCR sugeruje, że można doświadczalnie wyznaczyć zależność również i dla tych miar jakości. Takie podejście wymagałoby jednak dokonania pomiarów dla szerszego przepływności bitowej powyżej 4 kbit/s nie będących przedmiotem zainteresowania tej pracy.

Na podstawie otrzymanych wyników wyznaczone zostały proste regresji dla wartości wskaźnika jakości MOS w funkcji CD oraz LDR. Ponieważ dla przepływności bitowych poniżej 4 kbit/s zależność jest zbliżona do liniowej [4], zatem wyznaczone proste regresji były postaci:

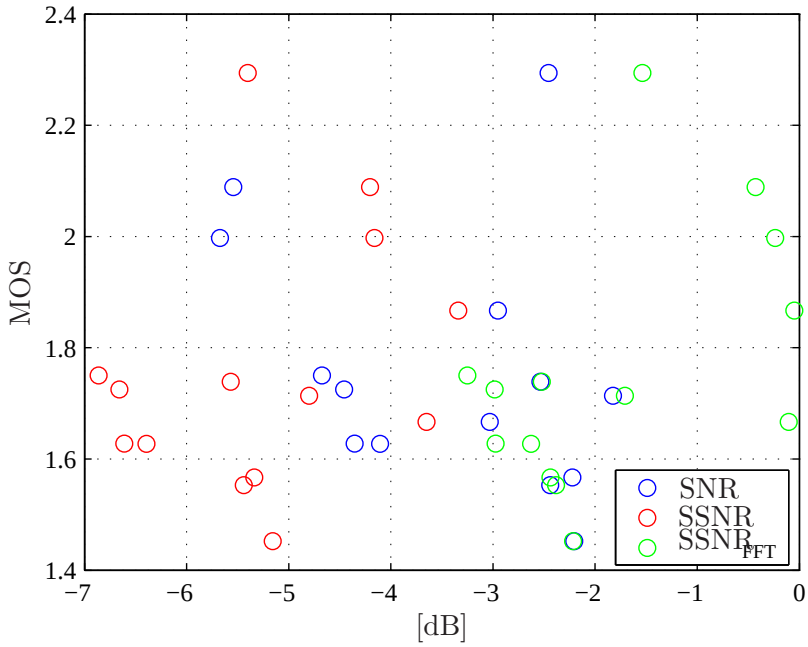
$$y = A \cdot x + B \tag{4.10}$$

Dla funkcji $MOS = f(CD)$ współczynniki prostej regresji wynoszą odpowiednio:

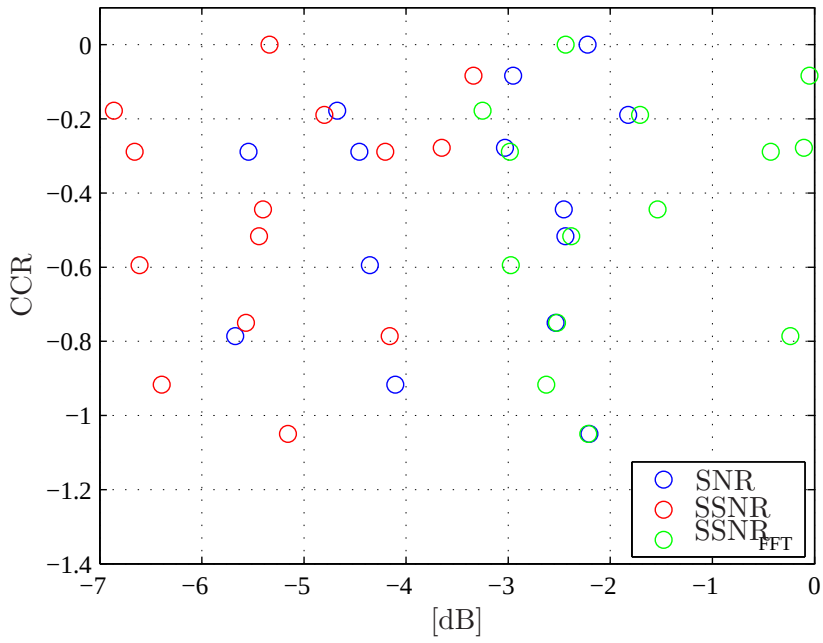
$$A = -0.108, B = 2.409$$

natomiast dla funkcji $MOS = f(LDR)$ wynoszą one:

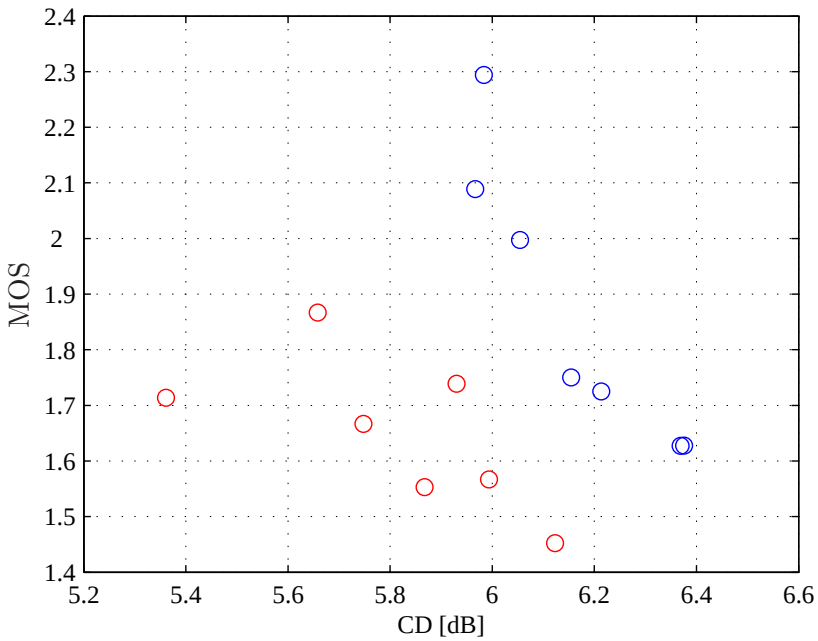
$$A = -1.083, B = 4.123$$



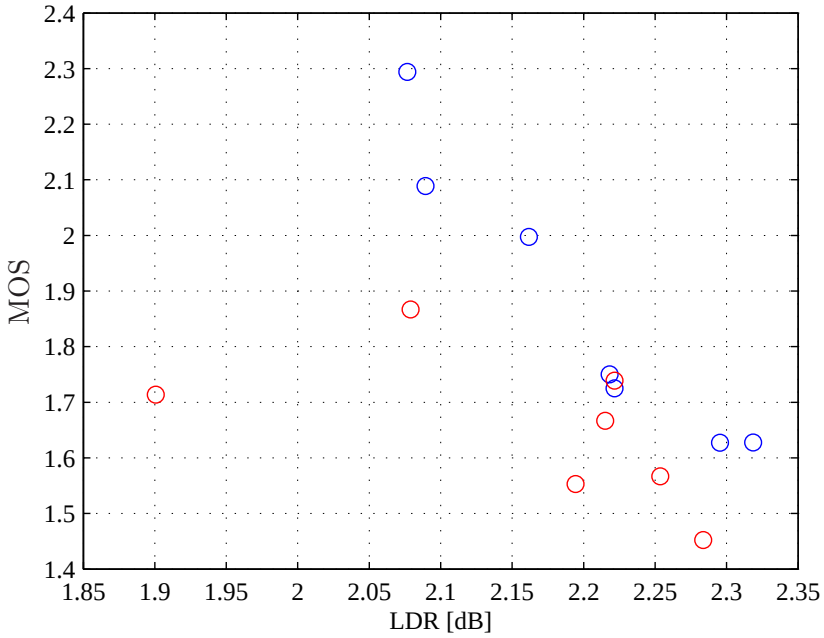
Rysunek 4.6. Wartości MOS w funkcji miar: SNR, SSNR oraz SSNR_{FFT}.



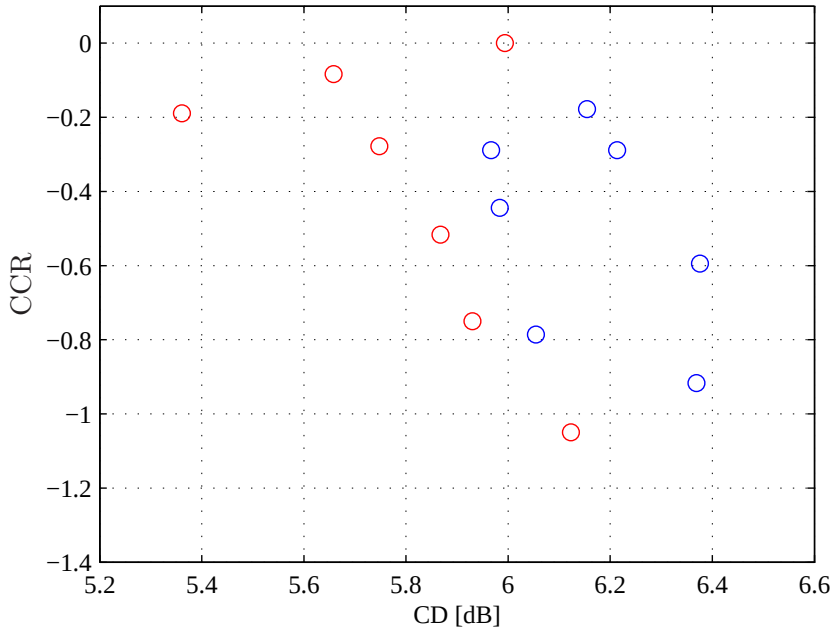
Rysunek 4.7. Wartości CCR w funkcji miar: SNR, SSNR oraz SSNR_{FFT}.



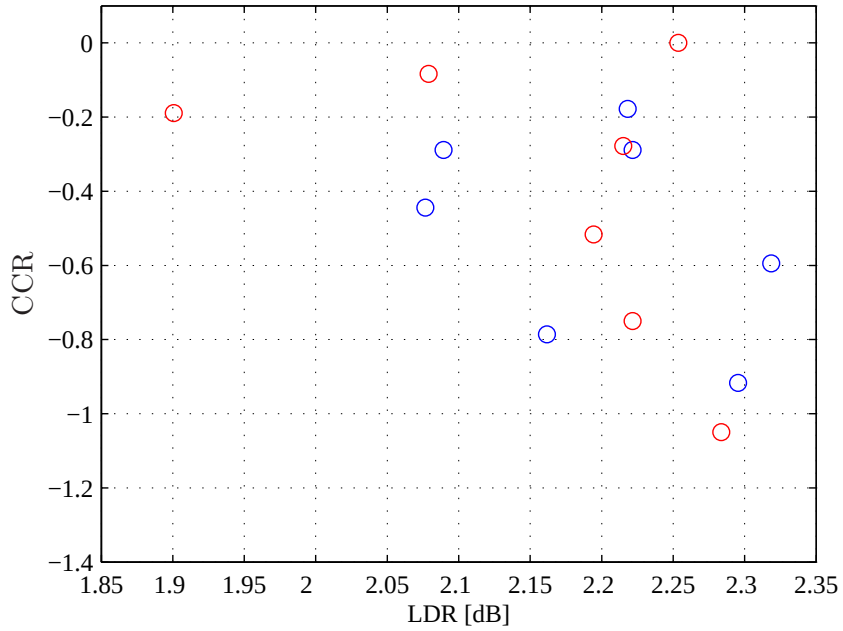
Rysunek 4.8. Wartości MOS w funkcji miary *cepstral distance* (CD). Kolorem niebieskim oznaczono wyniki dla sygnału mowy kobiecej, a czerwonym męskiej.



Rysunek 4.9. Wartości MOS w funkcji miary *log–likelihood ratio* (LDR). Kolorem niebieskim oznaczono wyniki dla sygnału mowy kobiecej, a czerwonym męskiej.



Rysunek 4.10. Wartości CCR w funkcji miary *cepstral distance* (CD). Kolorem niebieskim oznaczono wyniki dla sygnału mowy kobiecej, a czerwonym męskiej.



Rysunek 4.11. Wartości CCR w funkcji miary *log–likelihood ratio* (LDR). Kolorem niebieskim oznaczono wyniki dla sygnału mowy kobiecej, a czerwonym męskiej.

Wnioski końcowe

W przedstawionej pracy zaimplementowany został koder mowy standardu LPC–10 wraz z modyfikacjami mającymi na celu polepszenie jakości generowanej mowy. Omówione algorytmy generacji książek kodowych zostały zaimplementowane w języku C++. Posłużyły one do stworzenia kwantyzatorów wektorowych, które zostały zbadane pod kątem wpływu na jakość syntetyzowanej mowy. Podczas przeprowadzanych badań potwierdzono, że jakość kwantyzatora wektorowego jest lepsza niż kwantyzatora skalarnego przy tej samej średniej bitowej. Na podstawie wygenerowanych książek kodowych można stwierdzić, że dla przepływności bitowej 1250 bit/s (odpowiadającej zaprojektowanej książce kodowej o rozmiarze 10 bitów), jakość dźwięku wynosi ok. 1.5 stopnia w skali MOS. Oznacza to, że dla książek kodowych o rozmiarze 10 bitów osiągnięty został próg zrozumiałości. Jednocześnie zaproponowany został kwantyzator wektorowy 41-bitowy, który zachowuje przepływność bitową standardu LPC–10. Dla tego kwantyzatora jakość sygnału wzrosła średnio o 0.25 punkta w skali MOS w stosunku do standardu LPC–10. Potwierdzono również dalszy wzrost jakości o ok. 0.18 punkta w skali MOS przy zastosowaniu splotu sygnału pobudzenia z obwiednią widma sygnału resztkowego. Dodatkowo, na podstawie przeprowadzonych badań, wyznaczono zależność pomiędzy subiektywną miarą jakości MOS a miarami obiektywnymi opartymi na wyznaczaniu współczynników LPC. Oznacza to, że dla kodera o przepływności bitowej poniżej 4 kbit/s można uniknąć czasochłonnych badań jakości i wyznaczając średnie wartości funkcji ilorazu wiarygodności bądź odległości cepstralnych, a następnie dokonać przeliczenia na odpowiadające im wartości MOS. W przypadku miary CD wartość odchylenia standardowego od prostej regresji wynosi $\sigma_{CD} = 0.273$, natomiast w przypadku LDR wynosi $\sigma_{LDR} = 0.112$.

Wynikiem praktycznej realizacji omawianych w pracy zagadnień, jest programowa implementacja zmodyfikowanego kodera mowy LPC–10 napisana w postaci gotowych do użycia klas w języku C++. Konstrukcja programu została tak pomyślana, by można było dokonywać łatwych modyfikacji zastępując niektóre elementy składowe programu. Analiza budowy programu może posłużyć nie tylko do zapoznania się z zasadami kompresji mowy ale i zagadnieniami cyfrowego przetwarzania sygnału. Przygotowane klasy i funkcje odpowiedzialne m.in. za filtrację cyfrową czy wyznaczanie FFT stanowią przykład praktycznego zastosowania zagadnień poznanych w trakcie studiów a sama implementacja zarówno kodera jak i generatora książek kodowych daje pojęcie o złożoności procesów zachodzących we wszelkich dziedzinach związanych z telekomunikacją. Ponieważ w implementacji nie zostały wykorzystane specyficzne funkcje

systemu Windows, koder można z powodzeniem przenieść na dowolne inne środowisko dysponujące kompilatorem C++ np. GCC pod systemem operacyjnym Linux.

Zaimplementowane różne metody generacji książek kodowych, pozwalają na przygotowanie szerokiej gamy kwantyzatorów. W trakcie przygotowywania kodu generatorów, głównym problemem okazała się czasochłonność zastosowanych metod generacji książek kodowych. Wygenerowanie około 1000 książek kodowych metodą standardowego LBG jak i techniką podziałów w celu znalezienia najbardziej optymalnej książki kodowej w sensie globalnym, zajmowało od kilkunastu do kilkudziesięciu godzin obliczeń na komputerze wyposażonym w procesor AMD Athlon o częstotliwości taktowania zegara 1.6 GHz. Metoda symulowanego wyżarzania wymagała co prawda tylko jednego wywołania funkcji wyznaczającej książkę kodową, niemniej czas potrzebny na wygenerowanie książki kodowej był zbliżony. Dodatkowo, algorytm LBG uzupełniony metodą symulowanego wyżarzania ma charakter sekwencyjny więc nie można go rozproszyć wykonując równolegle obliczenia na wielu komputerach.

Zaprezentowana implementacja jak i wygenerowane książki kodowe mogą posłużyć zarówno do polepszenia jakości transmitowanego dźwięku w rozmowach głosowych dla przepływności bitowej poniżej 4 kbit/s, jak i do rozpoznawania dźwięku. Szczególnie ta druga własność jest pożądana z punktu widzenia automatyki oraz wszelkich dziedzin, których przedmiotem zainteresowania jest sterowanie za pomocą głosu. Przedstawiona implementacja kodera mowy stanowić może punkt wyjścia do rozszerzenia standardu LPC–10 o kolejne algorytmy. Jako przykład można wymienić kwantyzację współczynników $mag(i)$, implementację innych niż autokorelacja algorytmów wykrywania dźwięczności/bezdźwięczności ramki bądź uwzględnienie większej ilości rodzajów pobudzenia.

Bibliografia

[1] T. P. Zieliński *Od teorii do cyfrowego przetwarzania sygnałów* AGH ('02)

[2] A. Gersho, R.M. Gray: *Vector quantization and signal compression* KAP ('92)

[3] K. Sayood *Kompresja danych - wprowadzenie*, Wydawnictwo RM ('02)

[4] S. Furui *Digital speech processing, synthesis and recognition* Wyd. 2 ('01)

[5] Wai C. Chu *Speech coding algorithms: Foundation and Evolution of Standardized Coders* A John Wiley & Sons, Inc., Publication ('03)

[6] R. Tadeusiewicz *Sygnał mowy* Wydaw. Komunikacji i Łączności ('88)

[7] J. Gajda *Statystyczna analiza danych pomiarowych* Wyd. 1 ('02)

[8] B. Eckel *Thinking in C++* Wydawnictwo HELION ('02)

[9] J. Brzózka, L. Dorobczyński *Programowanie w Matlab* Wydawnictwo MIKOM ('98)

[10] A.Przelaskowski *Kompresja danych: podstawy, metody bezstratne, kodery obrazów* Wydawnictwo BTC, Warszawa ('05)

[11] L.R. Rabiner, R.W. Schafer *Digital processing of speech signals* ('78)

[12] R. G. Lyons *Understanding Digital Speech Processing* A Prentice Hall PTR Publication ('01)

[13] Władysław Skarbek *Metody reprezentacji obrazów cyfrowych* Akademicka Oficyna Wydawnicza PLJ, Warszawa ('93)

[14] Andrzej Czyżewski *Dźwiek cyfrowy. Wybrane zagadnienia teoretyczne, technologia, zastosowania* Akademicka Oficyna Wydawnicza EXIT, Warszawa ('98)

[15] Steven E. Levinson *Mathematical models for speech technology* A John Wiley & Sons, Inc., Publication ('05)

[16] Panaos E. Papamichalis *Practical approaches to speech coding* Prentince-Hall Inc ('87)

[17] A.M. Kondoz *Digital Speech: Coding for Low Bit Rate Communication Systems* A John Wiley & sons ('94)

[18] Alan V. Oppenheim Alan S. Willsky *Signals and systems* Prentice Hall ('97)

[19] Stephen E. Levinson *Mathematical Models for Speech Technology* A John Wiley & Sons, Inc., Publication ('05)

[20] Siegmund Brandt *Analiza danych - metody statystyczne* Wydawnictwo Naukowe PWN, Warszawa ('98)

Dodatek A

Opis funkcji generujących książki kodowe

Wszystkie załączone do tej pracy kody źródłowe napisane przez autora, są oparte na licencji GNU GPL. Poniżej został zamieszczony spis funkcji jakie zostały wykorzystane w algorytmach generatorów książek kodowych.
Plik nagłówkowy: `lbg.h`

void randomLBG(const vector *Training, vector *CBook, int TSize, int CSize):

Funkcja inicjalizująca początkową książkę kodową poprzez wylosowanie określonej ilości wektorów ze zbioru treningowego.

bool szukaj(int b,int *tab,int N):

Funkcja pomocnicza do funkcji `LBGrandomLBG(...)`, mająca na celu wyeliminowanie powtarzających się wektorów. Zwracana wartość informuje czy wylosowany wektor już istnieje w książce kodowej.

float LBG(const vector *Training, vector *CodeBook, int TSize, int CSize):

Funkcja wykonuje jedną optymalizację wygenerowanej książki kodowej algorytmem LBG a następnie zwraca całkowity błąd średniokwadratowy.

float standardLBG(const vector *TrainingSet, vector *CodeBook, int TSize, int CSize, float MaxError):

Funkcja wykonująca optymalizacje algorytmem LBG do momentu, gdy względna zmiana zniekształcenia zwracanego przez funkcję `LBG(...)` nie będzie mniejsza od zadanego progu `MaxError`. Zwraca błąd średniokwadratowy dla tej wartości. Jest to najbardziej podstawowy przypadek generowania książki kodowej.

float splitLBG(vector *, vector *, int, int, float):

Funkcja wyznacza książkę kodową przez technikę podziałów, wychodząc od centroida będącego środkiem ciężkości dla wszystkich wektorów treningowych. Po

wyznaczeniu ostatniej warstwy wektorów książki kodowej wykonywane jest jedno przeszukiwanie książki kodowej metodą LBG w celu wyznaczenia wartości błędu MSE. Otrzymana wartość jest zwracana przez funkcję.

unsigned int searchVector(const vector *Vec, const vector *CodeBook, int N):

Funkcja wyszukująca w książce kodowej o strukturze drzewa najbliższy wektor do zadanego. Zwraca indeks do wektora z książki kodowej odpowiadającemu szukanemu.

float relaxationLBG(const vector *TrainingSet, vector *CodeBook, int TSize, int CSize,float MaxError):

Funkcja wyznacza książkę kodową na podstawie algorytmu symulowanego wyżarzania. Wektory zakłóceń są dodawane do każdego centroidu aż do przekroczenia ilości iteracji zdefiniowanych przez zmienną `ITERACJE`. Wynikiem działania funkcji jest całkowity błąd średniokwadratowy.

Dodatek B

Opis klas i funkcji kodera oraz dekodera LPC-10

B.1. LPC10ExternalData

Klasa `LPC10ExternalDate` odpowiedzialna jest za wczytywanie książek kodowych z plików binarnych dla różnych metod kwantyzacji. Dla potrzeb kodera, dane są zapisywane bez żadnych nagłówków zawierających informacje o pliku.
Plik nagłówkowy: `lpc10_externaldata.h`

bool ReadScalarCoeff(char *name):

Wczytuje współczynniki minimalne i maksymalne dla każdego ze współczynników LSP dla kwantyzacji równomiernej. Funkcja przy wyjściu zwraca wartość `true`, w przypadku gdy plik ze współczynnikami nie istnieje bądź alokacja pamięci się nie powiedzie zwracana jest wartość `false`.

bool ReadResidualCoeff(char *name):

Wczytuje z pliku współczynniki LSP dla filtra resztkowego typu FIR bądź IIR. Funkcja przy wyjściu zwraca wartość `true`, w przypadku gdy plik ze współczynnikami LSP nie istnieje bądź alokacja pamięci się nie powiedzie zwracana jest wartość `false`.

bool ReadGCoeff(char *name):

Wczytuje książkę kodową wzmocnień filtra. Funkcja przy wyjściu zwraca wartość `true`, w przypadku gdy plik z książką kodową wzmocnień nie istnieje bądź alokacja pamięci się nie powiedzie zwracana jest wartość `false`.

bool ReadLSPScalarQuantLevels(char *name)

Wczytuje książkę kodową współczynników LSP dla kwantyzacji nierównomiernej. Funkcja przy wyjściu zwraca wartość `true`, w przypadku gdy książka kodowa nie istnieje bądź alokacja pamięci się nie powiedzie zwracana jest wartość `false`.

float* pGetResidualCoeff() const:

Zwraca wskaźnik do współczynników LSP filtra resztkowego.

float* pGetScalarCoeff() const:

Zwraca wskaźnik do tablicy z wartościami minimalnymi i maksymalnymi współczynników LSP dla kwantyzacji równomiernej.

float* pGetGCoeff() const:

Zwraca wskaźnik do książki kodowej wzmocnień.

float *pGetLSPQuant() const:

Zwraca wskaźnik do książki kodowej ze współczynnikami LSP dla kwantyzacji nierównomiernej.

B.2. LPC10Params

Klasa `LPC10Params` odpowiedzialna jest za ustawienie podstawowych parametrów: długości okna, częstotliwości próbkowania, rzędu filtra traktu głosowego, ilości wczytywanych próbek na ramkę oraz ilość próbek użytych do wyznaczania FFT. Domyślnie konstruktor ustawia powyższe parametry zgodnie z definicją pliku nagłówkowego `lpc10_defines.h`.
Plik nagłówkowy: `lpc10_params.h`

void SetWindowParams(int length):

Długość okna czasowego.

void SetFrequency(int freq):

Ustawienie informacji o zastosowanej częstotliwości próbkowania.

void SetFilterOrder(int ord):

Ustawienie rzędu filtra traktu głosowego.

void SetSampleNumber(int number):

Ustawienie ilości próbek wczytywanych w jednej ramce.

void SetFFTLength(int number):

Ustawienie ilości próbek przy obliczaniu FFT.

int GetWindowLength() const:

Zwraca długość zastosowanego okna czasowgo.

int GetFrequency() const:

Zwraca częstotliwość próbkowania w Hz.

int GetSampleNumber() const:

Zwraca ilość próbek wczytywanych w danej ramce.

int GetFilterOrder() const:

Zwraca rząd filtra traktu głosowego.

int GetFFTLength() const:

Zwraca ilość próbek dla ilu będzie wyznaczane widmo FFT.

B.3. Koder

Właściwą klasa kodera mowy standardu LPC–10 jest klasa `coder`. Zawiera ona dostępne z zewnątrz podstawowe interfejsy obsługujące procedurę kodowania. Ponieważ za opcje konfiguracyjne odpowiedzialne są klasy omówione powyżej, dla użytkownika zostały dostarczone tylko dwie metody.
Plik nagłówkowy: `lpc10_codec.h`

bool Create(const LPC10Params &params, const LPC10ExternalData &data):

Funkcja jako parametry przyjmuje referencję do obiektu klasy `LPC10Params` zawierający podstawowe informacje o parametrach pracy kodera. Drugi parametr będący referencją do obiektu klasy `LPC10ExternalData` zajmuje się przekazaniem wskaźników do obszaru pamięci w którym znajdują się książki kodowe. Do klasy `coder` przekazywane są referencje by nie tworzyć kopii książek kodowych w pamięci w przypadku równoczesnego wykorzystywania kodera i dekodera. Przy wyjściu z funkcji zwracana jest wartość `true`, natomiast w przypadku gdy alokacja pamięci się nie powiedzie zwracaną wartością jest `false`.

void code(const float *z,BITSTREAM *BS):

Jest to właściwa funkcja kodująca jedną ramkę sygnału mowy. Jako parametry wejściowe przyjmuje ona wskaźnik do ciągu próbek dźwiękowych zapisanych na typie `float` i unormowanych do zakresu $x \in [-1, 1]$ oraz wskaźnik do struktury `BITSTREAM` przechowującej informacje o wszystkich kodowanych parametrach.

B.4. Dekoder

Podobnie jak opisana wyżej klasa, za dekodowanie sygnału odpowiedzialna jest klasa `decoder`. Interfejs dostępny z zewnątrz również został ograniczony do dwóch funkcji opisanych poniżej.
Plik nagłówkowy: `lpc10_codec.h`

bool Create(const LPC10Params &params, const LPC10ExternalData &data):

Analogicznie jak funkcja `Create` w klasie `coder`, zajmuje się inicjalizacją podstawowych parametrów dekodera. Jako parametr przyjmuje referencje do obiektów klasy `LPC10Params`, która przechowuje informacje o podstawowych parametrach dekodowania, oraz referencję do klasy `LPC10ExternalData` odpowiedzialnej za dostęp do książek kodowych. Przy wyjściu z funkcji zwracana jest wartość `true`, natomiast w przypadku gdy alokacja pamięci się nie powiedzie zwracaną wartością jest `false`.

void decode(BITSTREAM *BS, float *out):

Jest to funkcja odpowiedzialna za zdekodowanie danych znajdujących się pod wskaźnikiem do struktury `BITSTREAM`. Wynikiem jest zsyntetyzowany sygnały mowy w postaci 160 próbek zapisanych na typie `float`, w obszarze pamięci pod wskaźnikiem `*out`.

Tabela B.1. Definicja pliku źródłowego `lpc10_defines.h`.

```
1 #define LPC10_DEFAULTFILTERORDER        10
2 #define LPC10_DEFAULTWINDOWLENGTH      240
3 #define LPC10_DEFAULTSAMPLENUMBER      160
4 #define LPC10_DEFAULTFFTLENGTH         256
5 #define LPC10_DEFAULTFREQUENCY        8000
```

Tabela B.2. Deklaracja pliku `lpc10_params.cpp`.

```
1 #include "lpc10_defines.h"
2 #include "lpc10_params.h"

4 LPC10Params::LPC10Params()
5 {
6     order     = LPC10_DEFAULTFILTERORDER;
7     window    = LPC10_DEFAULTWINDOWLENGTH;
8     sample    = LPC10_DEFAULTSAMPLENUMBER;
9     frequency = LPC10_DEFAULTFREQUENCY;
10    fourier   = LPC10_DEFAULTFFTLENGTH;
11 }
```

Tabela B.3. Przykład implementacji kodera mowy za pomocą dostępnych klas.

```
1  //Przyklad wykorzystania implementacji kodera mowy
2  //Praca dyplomowa ,,Optymalizacja kodera mowy LPC-10''
3  //AGH Grzegorz Suder 2006

5  vector *cbook,*lpc21,*lpc22,*lsp3;
6  Import_Cbook(&cbook,&lpc21,&lpc22,&lsp3); //wczytywanie ksiazki kodowej
7                                            //pod podane wskazniki
8  coder cod(cbook,lpc21,lpc22,lsp3); //inicjalizacja kodera}
9  decoder dec(cbook,lpc21,lpc22,lsp3); //inicjalizacja dekodera}
10 LPC10Params params;
11 LPC10ExternalData data;
12 data.ReadScalarCoeff("c:\\scalarcoeff.dat");
13 data.ReadResidualCoeff("c:\\fir.dat");
14 data.ReadGCoeff("c:\\cbooks\\G\\G64.dat");
15 data.ReadLSPScalarQuantLevels("c:\\cbooks\\lsp_all.dat");

17 cod.Create(params,data);
18 dec.Create(params,data);
19 float *u; //wskaznik do probek dzwiekowych
20 int c;
21 int N; //dlugosc sygnalu np. na podstawie wielkosci pliku

23 // petla kodujaca i dekodujaca
24 for(int i=1;i<N;i++)
25 {
26     T_prev=BS2->VUV;
27         T2=BS2->T;
28         tmp=BS2;
29         BS2=BS1;
30         BS1=tmp;
31         u=Input.read(); //wczytanie ramki
32         c=cod.code(u,BS1); //zakodowanie ramki do BS1
33         if(BS1->VUV==T_prev && BS2->VUV!=BS1->VUV)
34               BS2->T=(BS1->T+T2)/2; //usuniecie nieciaglosci w VUV}
35         BS2->T_prev=BS1->T;
36         c=dec.decode(BS2,out); //zdekodowanie ramki sygnalu
37 }
```

Dodatek C

Tabele

W poniższych tabelach znajdują się wyniki pomiarów jakości MOS i CCR. Zamieszczone wyniki zostały zebrane podczas testów odsłuchowych na wybranej grupie osób. Kolejne kolumny tabel oznaczają:

Oznaczenie	Sygnał mowy
We	przetworzony koderem bez kwantyzacji
S_{41}	kwantyzator skalarny 41–bitowy
V_{41}	kwantyzator wektorowy 41–bitowy dwupoziomowy
V_{41e}	kwantyzator wektorowy 41–bitowy z obwiednią widma
V_{17}	kwantyzator wektorowy 17–bitowy dwupoziomowy
V_{15}	kwantyzator wektorowy 15–bitowy dwupoziomowy
V_{10b}	kwantyzator wektorowy 10–bitowy jednopoziomowy
V_{10}	kwantyzator wektorowy 10–bitowy dwupoziomowy

Tabela C.1. Opis zastoswanych metod kwantyzacji próbek dźwiękowych przy pomiarach jakości MOS i CCR.

We	S_{41}	V_{41}	V_{41e}	V_{17}	V_{15}	V_{10}	V_{10b}
2.00	2.00	3.00	2.00	2.00	1.00	1.00	1.00
2.50	2.50	3.00	2.20	1.60	2.00	1.80	1.60
2.50	2.25	1.00	2.00	1.00	1.75	1.50	1.25
2.70	2.40	3.00	2.60	2.00	2.40	2.20	2.00
2.00	1.90	1.90	2.00	2.00	1.85	1.80	2.00
3.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
1.20	1.10	1.50	1.10	1.00	1.00	1.00	1.00
3.00	2.00	2.50	2.00	1.00	1.00	1.00	1.00
1.50	1.30	1.20	1.30	1.10	1.20	1.10	1.00
2.00	2.00	3.00	3.00	2.00	2.00	2.00	2.00
3.00	2.50	2.80	3.00	1.00	2.00	1.50	1.20
2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
2.00	2.00	1.50	1.75	1.50	1.50	1.50	1.50
2.30	2.00	2.60	2.15	1.55	1.80	1.60	1.45
3.20	3.00	2.30	2.50	2.30	2.00	2.30	2.30
2.00	2.00	3.00	2.00	3.00	2.00	2.00	2.00
2.00	1.00	3.00	2.00	2.00	2.00	1.00	2.00
3.00	2.00	4.00	2.00	2.00	2.00	2.00	2.00

Tabela C.2. Wyniki pomiarów jakości metodą subiektywną MOS dla sygnału mowy kobiecej

We	S_{41}	V_{41}	V_{41e}	V_{17}	V_{15}	V_{10}	V_{10b}
1.00	1.00	2.00	2.00	2.00	1.00	1.00	1.00
3.00	2.50	2.30	2.10	1.60	1.80	2.00	1.80
2.00	2.00	1.25	1.50	1.50	1.50	1.50	1.50
1.50	1.00	2.10	1.90	3.00	2.30	2.40	2.00
1.85	1.80	1.80	1.80	1.80	1.80	1.80	1.78
2.00	2.00	3.00	2.00	2.00	2.00	2.00	1.00
1.00	1.20	1.10	1.10	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.50	1.00	1.00	1.00
1.00	1.00	1.10	1.00	1.10	1.00	1.00	1.00
2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
2.30	1.20	3.00	2.80	2.50	1.50	1.50	1.20
2.70	2.70	2.70	2.00	2.00	2.00	2.00	2.00
2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
1.75	1.50	1.25	1.25	1.25	1.25	1.25	1.25
1.70	1.50	1.40	1.35	1.20	1.30	1.25	1.20
1.50	1.60	1.60	1.50	1.40	1.50	1.50	1.40
3.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
2.00	2.00	2.00	2.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	2.00	1.00	1.00	1.00

Tabela C.3. Wyniki pomiarów jakości metodą subiektywną MOS dla sygnału mowy męskiej

S_{41}	V_{41}	V_{41e}	V_{17}	V_{15}	V_{10}	V_{10b}
0.00	0.00	0.00	1.00	0.00	0.00	1.00
-1.00	0.00	-1.00	0.00	1.00	0.00	0.00
-2.00	1.00	0.00	-1.00	0.00	1.00	-1.00
-1.50	1.00	2.00	0.00	-1.00	2.00	-2.00
-1.00	0.00	-3.00	2.00	1.00	0.00	2.00
-1.00	0.00	2.00	0.00	0.00	0.00	-2.00
-1.00	1.00	2.00	1.00	-1.00	1.00	1.00
1.00	-1.00	1.00	0.00	-1.00	0.00	1.00
-1.00	-1.00	2.00	1.00	-1.00	0.00	-2.00
-1.00	1.00	1.00	0.00	1.00	0.00	0.00
0.00	0.00	1.00	1.00	1.00	0.00	2.00
-2.00	0.00	0.55	2.20	0.00	-1.50	-3.00
0.00	0.00	1.00	-1.00	0.00	0.00	-1.00
-1.00	1.00	0.00	1.00	-1.00	-2.00	2.00
-1.65	0.00	2.15	-0.50	0.00	0.00	-1.70
-1.00	0.00	0.00	-0.50	0.00	0.00	-0.50
0.00	1.00	0.00	-1.00	1.00	2.00	1.00
0.00	1.00	0.00	0.00	0.00	-1.00	0.00
2.00	0.00	1.00	1.00	0.00	-1.00	-1.00

Tabela C.4. Tabela z wynikami pomiarów jakości współczynnika CCR dla mowy kobiecej

S_{41}	V_{41}	V_{41e}	V_{17}	V_{15}	V_{10}	V_{10b}
1.00	-1.00	0.00	0.00	1.00	0.00	1.00
0.00	2.00	2.00	-1.00	-1.00	1.00	-1.00
0.00	-2.00	2.00	-1.00	-2.00	1.00	1.00
1.50	-1.50	2.00	0.00	-1.00	0.00	1.00
0.00	-1.00	1.00	-2.00	2.00	0.00	0.00
0.00	2.00	-1.00	-1.00	1.00	-1.00	0.00
2.00	-3.00	0.00	-2.00	2.00	-2.00	2.00
1.00	-2.00	0.00	-1.50	2.00	0.00	1.00
-1.00	-2.00	0.00	-2.00	-1.00	0.0	0.00
-1.00	0.00	-1.00	0.00	-1.00	1.00	1.00
0.00	0.00	0.00	0.00	-1.00	-1.00	0.00
2.50	-1.00	-3.00	-2.80	-3.00	0.00	0.50
1.00	-2.00	-1.00	0.00	-2.00	-1.00	0.00
2.00	-2.00	0.00	-3.00	2.00	0.00	1.00
1.50	-2.00	1.40	-0.60	-2.50	-2.20	1.80
2.00	-2.00	0.00	0.00	-1.50	-2.00	0.00
0.00	1.00	1.00	-1.00	0.00	1.00	0.00
1.00	0.00	0.00	-1.00	-2.00	0.00	0.00
0.00	-1.00	-1.00	0.00	2.00	0.00	0.00

Tabela C.5. Tabela z wynikami pomiarów jakości współczynnika CCR dla mowy męskiej

Dodatek D

Skrypty języka Matlab

Przedstawione w tym załączniku skrypty służą do oceny jakości sygnału metodami obiektywnymi. Wywołanie każdej m–funkcji sprowadza się do podania dwóch parametrów, którymi są: scieżka do oryginalnego pliku oraz ścieżka do pliku poddanego schematowi analiza–synteza wraz z kwantyzacją i dekwantyzacją. Skrypty zostały napisane w Matlabie 6.5 ale ponieważ nie używają specyficznych funkcji dla tej wersji programu, mogą z powodzeniem zostać wykorzystane w starszych wersjach środowiska `Matlab`.

Tabela D.1. Funkcja w języku Matlab służąca do wyznaczania oceny jakości dźwięku mowy metodą wyznaczania miary *log-likelihood ratio*.

```
% funkcja obliczajaca wartosc cepstral distance
% Praca dyplomowa ,,Optymalizacja kodera mowy LPC-10''
% AGH Grzegorz Suder 2006
% LIKELIHOOD.m

function [DLR]=LIKELIHOOD(wejscie,wyjscie)
      x = wavread(wejscie);
      y = wavread(wyjscie);
  shift=160; % dlugosc ramki

  N_org=length(x); % dlugosc sygnalu wejsciowego
  N_por=length(y); % dlugosc sygnalu porownywanego

  % zapewnienie tej samej dlugosci obydwu sygnalow
  if N_org>N_por
      N=N_por;
      x=x(1:N_por);
  end

  if N_org<N_por
      N=N_org;
      y=y(1:N_org);
  end

  r = [];
      OUT = [];

      for len = 1:shift:N-shift
              we_x = x(len:shift+len-1);
              we_y = y(len:shift+len-1);
              we_x = we_x.*hamming(shift);
              we_y = we_y.*hamming(shift);
              we_x = we_x-mean(we_x);
              we_y = we_y-mean(we_y);
              N = length(we_x);

              [a_x g_x] = lpc(we_x,10); % wyznaczenie wspolczynnikow filtra
              [a_y g_y] = lpc(we_y,10);

              a_x = real(a_x)'; % usuniecie wartosci zespolonych
              a_y = real(a_y)';

              r_x = xcorr(we_x,'biased')';
              r_y = xcorr(we_y,'biased')';

              r_x = r_x(N:2*N-1);
              r_y = r_y(N:2*N-1);

              % wyznaczenie macierzy autokorelacji
              for m = 1:11
                      R_x(m,:) = [r_x(m:-1:2) r_x(1:12-m)];
                      R_y(m,:) = [r_y(m:-1:2) r_y(1:12-m)];
              end
              dlr = (a_y'*R_x*a_y)/(a_x'*R_x*a_x); % obliczenie wartosci DLR

              if isnan(dlr) == 1 % usuniecie wszelkich wartosci nieliczbowych
                      dlr = 1;
              end
              OUT = [OUT dlr];
      end
      DLR = 10*log10(mean(OUT));
```

Tabela D.2. Funkcja w języku Matlab służąca do wyznaczania oceny jakości dźwięku mowy metodą wyznaczania miary *cepstral distance*.

```
% funkcja obliczajaca wartosc cepstral distance
% Praca dyplomowa ,,Optymalizacja kodera mowy LPC-10''
% AGH Grzegorz Suder 2006
% CEPSTRAL.m

function [CD]=CEPSTRAL(wejscie,wyjscie)
      x = wavread(wejscie);
      y = wavread(wyjscie);
  shift=160; % dlugosc ramki

  N_org=length(x); % dlugosc sygnalu wejsciowego
  N_por=length(y); % dlugosc sygnalu porownywanego

  % zapewnienie tej samej dlugosci obydwu sygnalow
  if N_org>N_por
      N=N_por;
      x=x(1:N_por);
  end

  if N_org<N_por
      N=N_org;
      y=y(1:N_org);
  end

      N = length(x);
      r = [];
      OUT = [];
      for len = 1:160:N-shift
            we_x = x(1+len:shift+len);
            we_y = y(1+len:shift+len);
            we_x = we_x.*hamming(shift);
            we_y = we_y.*hamming(shift);
            we_x = we_x-mean(we_x);
            we_y = we_y-mean(we_y);
            N = length(we_x);
            [a_x g_x] = lpc(we_x,10); % wyznaczenie wspolczynnikow filtra
            [a_y g_y] = lpc(we_y,10);

            c_x(1) = 0; % inicjalizacja poczatkowych wartosci wsp. cepstralnych
            c_y(1) = 0;
            c_x(2) = -a_x(2);
            c_y(2) = -a_y(2);

            % obliczenie wartoci c_x i c_y
            for i = 2:10
                  tmp1 = 0;
                  tmp2 = 0;
                  for k = 1:i-1
                        tmp1 = tmp1+((i-k)/i)*c_x(i-k+1)*a_x(k+1);
                        tmp2 = tmp2+((i-k)/i)*c_y(i-k+1)*a_y(k+1);
                  end
                  c_x(i+1) = -a_x(i+1)-tmp1;
                  c_y(i+1) = -a_y(i+1)-tmp2;
            end

            tmp = sum((c_x-c_y).^2); % obliczenie CD

            if isnan(tmp) == 1
                  tmp = 0;
            end
            OUT = [OUT (10/log(10))*sqrt(2*tmp)];
      end

      OUT = real(OUT);
      CD = mean(OUT);
```

Tabela D.3. Funkcja w języku Matlab służąca do wyznaczania oceny jakości dźwięku mowy metodami: SNR, SSNR oraz $SSNR_{FFT}$.

```
% funkcja obliczajaca wartosc cepstral distance
% Praca dyplomowa ,,Optymalizacja kodera mowy LPC-10''
% AGH Grzegorz Suder 2006
% SNR.m

function [SNR, SSNR, SSNR_FFT]=WYZNACZ_SNR(wejscie,wyjscie)
      x = wavread(wejscie);
      y = wavread(wyjscie);
  shift=160; % dlugosc ramki

  N_org = length(x);
      N_por = length(y);

  % zapewnienie tej samej dlugosci obydwu sygnalow
  if N_org>N_por
      N=N_por;
      x=x(1:N_por);
  end

  if N_org<N_por
      N=N_org;
      y=y(1:N_org);
  end

      % obliczenie zwyklego SNR
      SNR = 10*log10(sum(x.^2)/sum((x-y).^2));

      fft_shift = 96; % ilosc zer jaka nalezy dodac do sygnalu by dlugosc ramki
                      % wyniosla 2^N
      SSNR = 0;
      SSNR_FFT = 0;
      dl = 0;

      for i = 1:shift:N-shift
              org = x(i:i+shift-1);
              por = y(i:i+shift-1);
              org_fft = abs(fft([org' zeros(1,fft_shift)])); % dodanie zer dla FFT
              por_fft = abs(fft([por' zeros(1,fft_shift)]));
              roz = sum((org-por).^2);
              roz_fft = sum((org_fft-por_fft).^2);
              suma = sum(org.^2);
              suma_fft = sum(org_fft.^2);
              if suma == 0
                      suma = 1;
              end
              if suma_fft == 0
                      suma_fft = 1
              end
              SSNR = SSNR+10*log10(suma/roz); % obliczenie SSNR
              SSNR_FFT = SSNR_FFT+10*log10(suma_fft/roz_fft); % obliczenie SSNR_FFT
              dl = dl+1;
      end

      % unormowanie optrzymanych wartosci SSNR i SSNR_FFT do ilosci ramek
      SSNR = SSNR/dl;
      SSNR_FFT = SSNR_FFT/dl;
```

Dodatek E

Opis zawartości CDROM-u

- Wave – zebrane próbki dźwiękowe
  - (...)
  - Pliki_testowe – pliki wykorzystane do testów MOS i CCR
- Ksiazki_kodowe – wygenerowane książki kodowe różnych rozmiarów
  - (...)
- Wektory_treningowe – zbiór wektorów treningowych do tworzenia książek
  - (...) kodowych
- Pliki_zrodlowe – programy w języku C++
  - Koder_mowy – kod źródłowy zaimplementowanego kodera mowy
    - (...)
  - Generator – kod źródłowy generatora książek kodych
    - (...)
- Kod_wynikowy – kod wynikowy kodera mowy i generatora książek
  - (...) kodowych
- Matlab – skrypty w języku Matlab
  - (...)
- Contents.txt – opis zawartości CDROM-u

Dodatek F

CDROM