**Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie**

Wydział Informatyki, Elektroniki i Telekomunikacji

Katedra Elektroniki

**Praca dyplomowa**
**magisterska**

# Eliminacja szumów i zakłóceń z sygnału mowy w czasie rzeczywistym na platformie typu embedded

*Imię i nazwisko:* Aleksander Strzeboński
*Kierunek studiów:* Elektronika i Telekomunikacja
*Typ studiów:* Stacjonarne
*Opiekun pracy:* dr hab. inż. Konrad Kowalczyk, prof. AGH

Kraków 2023

# Spis treści

# Rozdział 1

# Wstęp

## 1.1 Wprowadzenie

W poniższej pracy poruszona została tematyka eliminacji szumów i zakłóceń z sygnału mowy w czasie rzeczywistym na platformie embedded. Eliminacja szumów i zakłóceń jest częstym wyzwaniem, z którym mierzą się współczesne systemy przetwarzania sygnałów dźwiękowych. Niejednokrotnie odbierana mowa zakłócona jest przez szum lub inne źródło obecne w środowisku mówcy. Dzieje się tak szczególnie w miejscach publicznych takich jak ulice, restauracje, lotniska czy też biura. W takich środowiskach, mowa odbierana w mikrofonie może być bardzo niskiej jakości. W niewygłuszonych pomieszczeniach często występuje też pogłos, który dodatkowo zniekształca sygnał. Eliminacja pogłosu również jest bardzo ważna do uzyskania sygnału o wysokiej jakości. Algorytmy służące do zwiększania jakości sygnału używane są często do wstępnego przetwarzania sygnału mowy przed zastosowaniem algorytmu automatycznego rozpoznawania mowy (ang. Automatic Speech Recognition - ASR). Przygotowanie sygnału pozbawionego szumów i zakłóceń jest kluczowe do poprawnego rozpoznawania mowy. Ze względu na małe rozmiary i niski koszt, bardzo często sygnał mowy przetwarzany jest na platformach embedded. Taką platformę można umieścić w tej samej obudowie co mikrofon i dzięki temu przetwarzać sygnał od razu w trakcie jego odbierania, bez konieczności angażowania do tych celów zewnętrznego programu działającego na komputerze osobistym lub w chmurze.

Bardzo ważną kwestią jest dobór czujnika akustycznego. Eliminacji zakłóceń można dokonać za pomocą algorytmu działającego na sygnale odebranym z jednego mikrofonu [1]. Lepsze rezultaty można jednak uzyskać używając macierzy mikrofonowej. Taka macierz pozwala na wydobycie z sygnału jego zależności przestrzennych takich jak na przykład kierunek, z którego nadchodzi dany sygnał (ang. Direction of Arrival - DOA). Posiadając sygnał odebrany macierzą mikrofonową można też używać metod kształtowania wiązki (ang. beamforming). Algorytmy wykorzystujące macierz mikrofonową można podzielić na algorytmy "poinformowane", czyli takie, które znają geometrię macierzy mikrofonowej czyli dokładne rozmieszczenie mikrofonów w macierzy mikrofonowej oraz na algorytmy "ślepe", czyli takie, które nie znają geometrii macierzy mikrofonowej. W rzeczywistości poznanie dokładnej geometrii

macierzy mikrofonowej nie zawsze jest takie proste. Wymiary macierzy mikrofonowej cierpią też na pewną niedokładność związaną z samymi pomiarami lub też z procesem produkcyjnym, który sprawia, że te same modele macierzy mikrofonowej mogą się nieznacznie różnić między sobą. Takie niedokładności mogą znacząco obniżyć jakość działania algorytmów "poinformowanych". Algorytmy "ślepe"nie cierpią na te problemy. Z drugiej strony, należy jednak pamiętać, że mają one mniej informacji do wykorzystania, co sprawia, że, w szczególności w warunkach laboratoryjnych i symulacyjnych, dają nieco gorsze rezultaty niż algorytmy "poinformowane".

Istnieje wiele sposobów na eliminacje szumów i zakłóceń z sygnału mowy odebranej macierzą mikrofonową. Część metod polega na dekompozycji przestrzeni otrzymanego sygnału na podprzestrzeń sygnału źródłowego, podprzestrzenie sygnałów zakłócających i podprzestrzeń szumu. Jedną z takich metod jest transformata Karhunen-Loéve'a [2]. Podobne podejście można zaobserwować też w metodzie zgeneralizowanego rozkładu na wartości własne (ang. Generalized Eigenvalue Decomposition - GEVD) [3]. Kolejnym szeroko stosowanym podejściem jest używanie metod kształtowania wiązki takich jak optymalny wielokanałowy filtr Wienera (ang. Optimal Multi-channel Wiener filter - MWF) [4] czy wielokanałowy filtr Kalmana [5]. Większość z tych metod wymaga policzeniu krótko-czasowej transformaty Fouriera (ang. Short-Time Fourier Transform - STFT) i operowania osobno na każdej częstotliwości sygnału, przez co ignorowana jest korelacja między częstotliwościami.

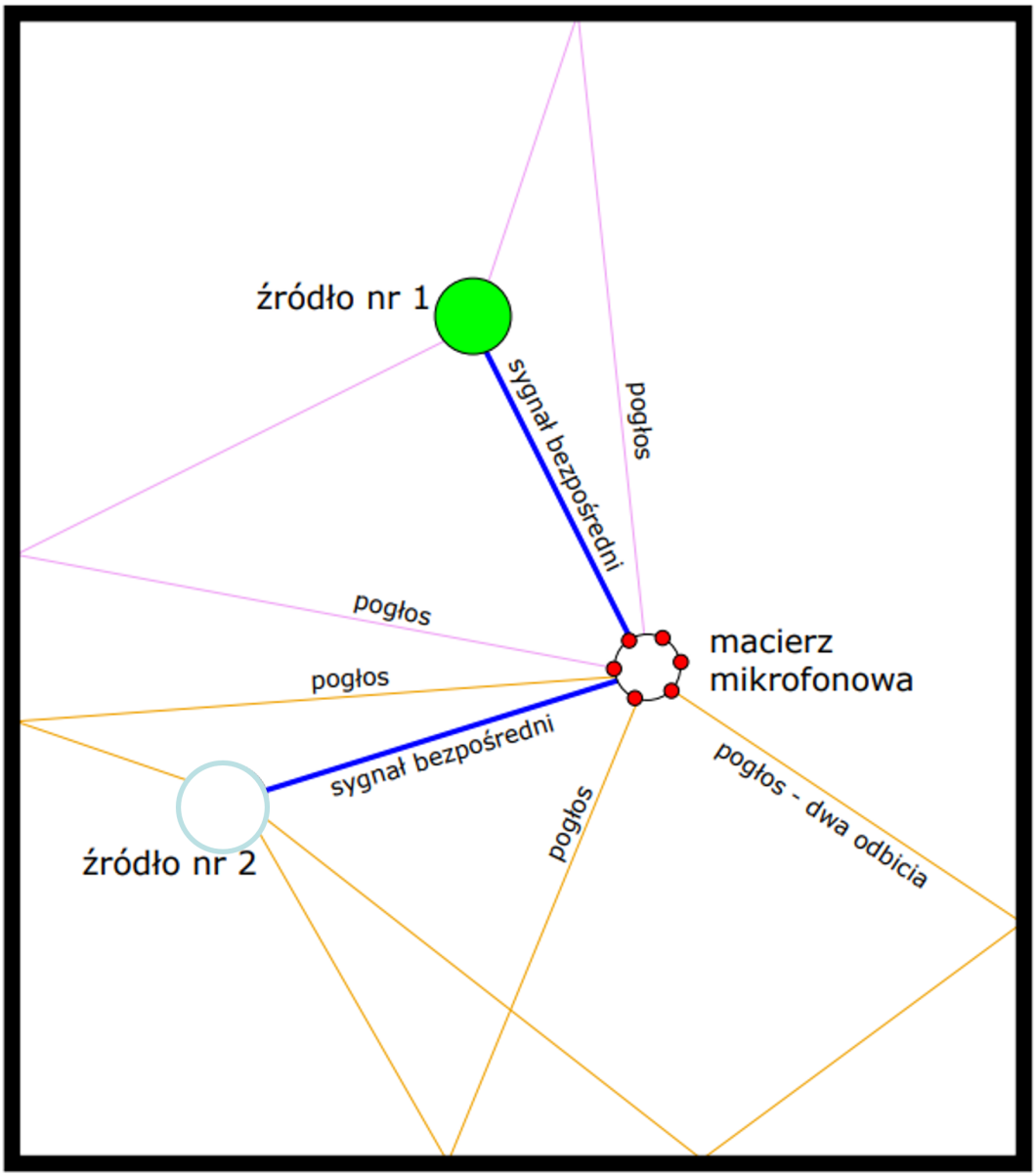
W tej pracy zdecydowano się na użycie metody bazującej na rozkładzie macierzy wielomianowej na wartości własne (ang. Polynomial Eigenvalue Decomposition - PEVD) [6]. Macierze wielomianowe potrafią jednocześnie przechowywać informacje o korelacji czasowej, przestrzennej oraz częstotliwościowej co stanowi pewną przewagę nad metodami używającymi STFT. Metody bazujące na PEVD używane są do wielu zadań z dziedziny przetwarzania sygnałów takich jak separacja źródeł, metody kształtowania wiązki [7], a także do dekodowanie sygnału w transmisji typu wiele wejść wiele wyjść (ang. Multiple Input Multiple Output - MIMO).

## 1.2 Cel

Celem pracy magisterskiej było stworzenie programu, który jest w stanie eliminować szumy i zakłócenia z sygnału mowy w czasie rzeczywistym działając na platformie embedded. Dodatkowo celem pracy było przetestowanie jakości i czasu działania programu w różnych warunkach zewnętrznych.

Na rysunku 1.1 symbolicznie przedstawiono problem, który należy rozwiązać. W pomieszczeniu znajdują się dwa lub więcej źródła sygnału, które jednocześnie nadają sygnał odbierany przez macierz mikrofonową. Jednym z tych źródeł sygnałów jest mówiący człowiek. Pozostałe źródła zakłócają mowę pierwszego źródła. Ponadto, sygnały ze wszystkich źródeł odbijają się od ścian i również trafiają do macierzy mikrofonowej jako pogłos. Dodatkowo, w sygnale odebranym przez macierz mikrofonową znajduje się też szum. Zadaniem programu jest takie przetworzenie za-

szumionego i zakłóconego sygnału odebranego przez macierz mikrofonową, aby na końcu otrzymać sygnał będący jak najwierniejszym odwzorowaniem sygnału bezpośredniego pochodzącego od pierwszego źródła.




Rysunek 1.1: Symboliczne przedstawienie problemu

## 1.3 Zakres wykonanej pracy

Pierwszym etapem wykonywanej pracy było zapoznanie się z literaturą dotyczącą różnych metod eliminacji szumów i zakłóceń z sygnału mowy. Następnie należało

wybrać metodę, której złożoność obliczeniowa pozwoli na wykorzystanie jej na platformie embedded w czasie rzeczywistym. Do implementacji metody wybrany został język programowania C++, a do weryfikacji poprawności implementacji wykorzystano język programowania Python. Kolejnym krokiem było stworzenie modelu pozwalającego w łatwy sposób, na bazie jedno-kanałowych nagrań dźwiękowych, generować sygnały odebrane w poszczególnych mikrofonach macierzy mikrofonowej, w różnych warunkach zewnętrznych takich jak rozmiar pokoju, położenie źródeł, położenie mikrofonów etc. Posłużono się w tym celu generatorem odpowiedzi impulsowej pomieszczenia (ang. Room Impulse Response - RIR) [8, 9]. Następnie, w obu wybranych językach programowania zaimplementowano algorytm rozkładu wielomianowej macierzy przestrzenno-czasowej korelacji na wartości własne (PEVD), którego główną częścią była implementacja algorytmu sekwencyjnej najlepszej rotacji drugiego rzędu (ang. second-order Sequential Best Rotation - SBR2) [10]. Kolejnym etapem było przeniesienie zaimplementowanego algorytmu na platformę embedded Raspberry Pi Zero W. Następnie dobrano parametry wewnętrzne algorytmu, tak aby jakość jego działania była jak najlepsza i równocześnie, żeby działał on w czasie rzeczywistym. W tym celu dokładnie zbadano zależności pomiędzy czasem a jakością działania programu dla różnych parametrów wewnętrznych algorytmu. Następnie, przetestowana została jakość działania algorytmu. W tym celu wykonano testy przy różnych warunkach zewnętrznych z użyciem różnych sygnałów źródłowych. Do ewaluacji wyników testów wykorzystano najpopularniejsze miary pokazujące jakość i zrozumiałość sygnałów mowy.

# Rozdział 2

# Analiza teoretyczna problemu

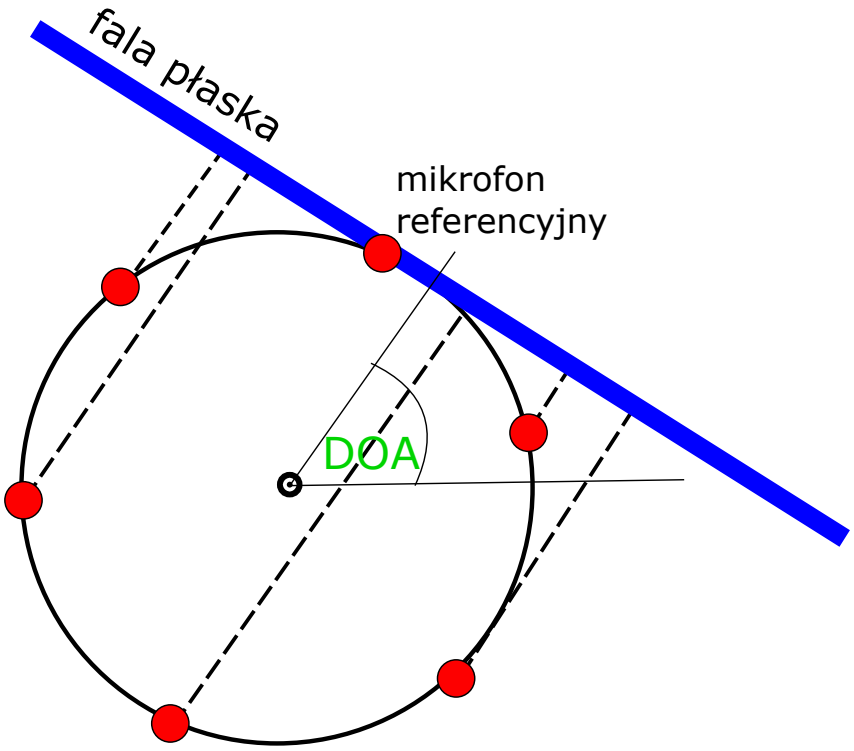
## 2.1 Model sygnału

Model sygnału jest zaczerpnięty z [10]. Zakłada się, że sygnał źródłowy, zakłócony przez $K-1$ sygnałów zakłócających oraz szum addytywny, odbierany jest przez $M$ mikrofonów w przestrzeni, która ze względu na swoją charakterystykę dokłada do sygnałów pogłos. Dodatkowo, założono, że mikrofony mają charakterystykę dookólną i umieszczone są na płaszczyźnie. Przy tych założeniach, sygnał odebrany w mikrofonie $m$ można przedstawić jako funkcja dyskretnej zmiennej czasu w następujący sposób:

$$x_m(n) = \sum_{k=1}^{K} s_k(n) * \mathbf{h}_{m,k} + v_m(n), \tag{2.1}$$

gdzie $x_m(n) \in \mathbb{R}$ to odebrany sygnał w mikrofonie $m$ w funkcji dyskretnej zmiennej czasowej $n$, $s_k(n) \in \mathbb{R}$ to nadany sygnał przez źródło $k$ w funkcji dyskretnej zmiennej czasowej $n$, $*$ oznacza splot, $\mathbf{h}_{m,k} \in \mathbb{R}^J$ oznacza odpowiedź impulsową filtru reprezentującego wielodrogowe zniekształcenie sygnału na drodze ze źródła $k$ do mikrofonu $m$, $J$ oznacza długość odpowiedzi impulsowej filtru, a $v_m(n) \in \mathbb{R}$ oznacza addytywny szum w mikrofonie $m$ w funkcji dyskretnej zmiennej czasowej $n$. Implementowany algorytm działa na ramce czasowej o zadanej długości. Dla danej ramki czasowej zakłada się, że filtry $\mathbf{h}_{m,k}$ są stacjonarne. Przyjmuje się, że sygnał ze źródła dla $k = 1$ jest sygnałem pożądanym, a sygnały z pozostałych źródeł są sygnałami zakłócającymi. W związku z tym, sygnałem, który chcemy otrzymać na wyjściu algorytmu, jest sygnał $s_1(n)$. Sygnały $x_m(n) \in \mathbb{R}$, biorąc pod uwagę liczbę mikrofonów równą $M$, możemy zapisać w postaci wektorowej $\mathbf{x}(n) \in \mathbb{R}^M$.

Na rysunku 2.1 można zaobserwować skąd biorą się filtry $\mathbf{h}_{m,k}$. Fala reprezentująca sygnał trafia pod pewnym kątem do macierzy mikrofonowej. Każdy mikrofon odbiera tę falę w innej chwili czasowej. Należy pamiętać, że propagacja sygnału jest wielodrogowa co oznacza, że takich fal, pochodzących od jednego źródła sygnału jest więcej, co przedstawione zostało na rysunku 1.1




Dystans między mikrofonem a mikrofonem referencyjnym dla danego DOA

Rysunek 2.1: Różnice w sygnale na poszczególnych mikrofonach macierzy mikrofonowej

## 2.2 Postawiony problem

Głównym celem tej pracy magisterskiej jest jak najlepsza estymacja pożądanego sygnału źródłowego, czyli $s_1(n)$. Liczba informacji jest bardzo ograniczona. Nieznana jest geometria macierzy mikrofonowej. Nie wiadomo też ile jest źródeł zakłócających. Zakłada się jednak, że sygnałów źródłowych jest mniej niż mikrofonów ($K < M$). Założono również, że moc pożądanego sygnału źródłowego $s_1(n)$ odebrana w macierzy mikrofonowej jest większa niż moc sygnałów zakłócających odebranych w macierzy mikrofonowej $s_k(n) : k \in \{2, ..., K\}$. Na wejściu, algorytm otrzymuje wielo źródłowy, pogłosowy, zaszumiony sygnał odebrany w $M$ mikrofonach tworzących macierz mikrofonową. Sygnał ten oznaczany jest jako $\mathbf{x}(n)$.

# Rozdział 3

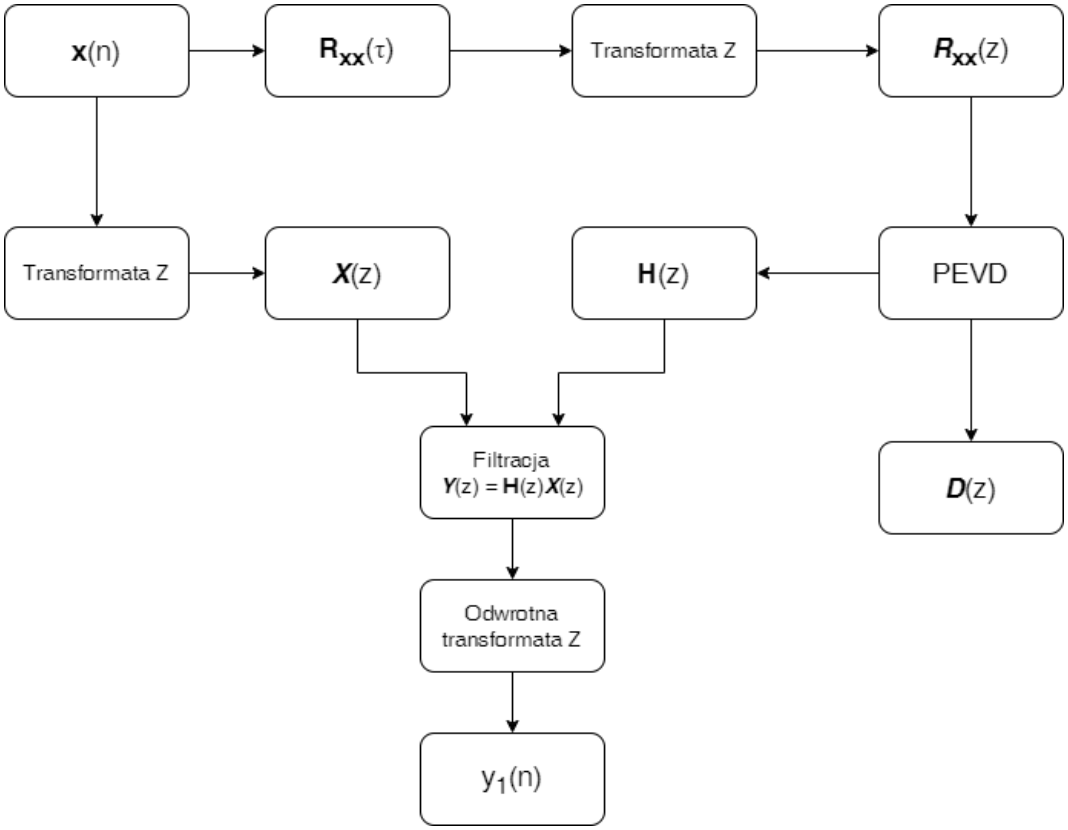
# Opracowane rozwiązanie problemu

## 3.1 Model rozwiązania

W celu rozwiązania problemu w głównej mierze użyty został algorytm opisany w publikacji [6]. Algorytm ten polega na rozkładzie na wartości własne wielomianowej macierzy (ang. Polynomial Eigen Value Decomposition - PEVD) przestrzenno-czasowej korelacji w dziedzinie transformacji Z. Więcej o transformacji Z można przeczytać tutaj: [11]. Po dokonaniu takiego rozkładu, możemy rozłożyć przestrzeń otrzymanego sygnału na podprzestrzenie sygnału pożądanego, sygnałów zakłócających oraz podprzestrzeń szumu i następnie przefiltrować sygnał otrzymany tak, żeby otrzymać sygnał pożądany. Można zatem wyróżnić następujące kroki potrzebne do rozwiązania problemu postawionego w sekcji 2.2:

- obliczenie wielomianowej macierzy przestrzenno-czasowej korelacji w dziedzinie transformacji Z,
- PEVD macierzy przestrzenno-czasowej korelacji przy pomocy algorytmu sekwencyjnej najlepszej rotacji drugiego rzędu (ang. second-order Sequential Best Rotation - SBR2) [10],
- obliczenie transformacji Z z sygnału wejściowego
- przefiltrowanie transformacji Z z sygnału wejściowego przy użyciu wektorów własnych reprezentujących podprzestrzeń sygnału pożądanego
- policzenie odwrotnej Z transformacji z przefiltrowanego sygnału w celu otrzymania sygnału pożądanego w dziedzinie czasu

Dla uogólnienia rozważań, w poniższych sekcjach sygnał wejściowy $\mathbf{x}(n)$ traktowany będzie jako sygnał zespolony. W postawionym problemie sygnał ten jest rzeczywisty. Schemat blokowy przetwarzania sygnału przedstawiony został na rysunku 3.1.




Rysunek 3.1: Schemat blokowy przetwarzania sygnału

## 3.2 Obliczenie wielomianowej macierzy przestrzenno-czasowej korelacji w dziedzinie transformacji Z

W dziedzinie czasu, wielomianowa macierz przestrzenno-czasowej korelacji z sygnału $\mathbf{x}(n) \in \mathbb{C}^M$ zdefiniowana jest w następujący sposób:

$$\mathbf{R_{xx}}(\tau) = \mathbb{E}\{\mathbf{x}(n)\mathbf{x}^H(n-\tau)\}, \tag{3.1}$$

gdzie $\mathbf{R_{xx}}(\tau) \in \mathbb{C}^{MxM}$ to macierz przestrzenno-czasowej korelacji z sygnału $\mathbf{x}(n)$ w funkcji zmiennej $\tau$ reprezentującej przesunięcie czasowe, $\mathbb{E}\{\cdot\}$ oznacza wartość oczekiwaną, a $[\cdot]^H$ oznacza sprzężenie Hermitowskie. Trzeba wziąć pod uwagę, że algorytm operuje na sygnale czasowym $\mathbf{x}(n)$ wziętym z pewnej ramki czasowej o długości $T+1$, czyli $n \in \{0, ..., T\}$. W związku z tym równanie 3.1 należy przybliżyć w następujący sposób:

$$\hat{\mathbf{R}}_{\mathbf{xx}}(\tau) \approx \frac{1}{T+1} \sum_{n=0}^{T} \mathbf{x}(n)\mathbf{x}^H(n-\tau). \tag{3.2}$$

W przypadku, gdy argument $n-\tau$ wykracza poza dziedzinę ramki czasowej $\{0, ..., T\}$, należy użyć wartości spoza ramki czasowej (np. $\mathbf{x}(-1)$). Jeżeli nie ma takiej możliwości, można, przy odpowiednio dużym $T$, przyjąć, że wartości funkcji $\mathbf{x}(n)$ dla n

spoza przedziału $\{0, ..., T\}$ wynoszą $\underline{\mathbf{0}}$, gdzie $\underline{\mathbf{0}}$ to wektor złożony z samych zer.

Z macierzy $\mathbf{R}_{\mathbf{xx}}(\tau)$ można policzyć transformację Z w następujący sposób:

$$\boldsymbol{\mathcal{R}}_{\mathbf{xx}}(z) = \sum_{\tau=-\infty}^{\infty} \mathbf{R}_{\mathbf{xx}}(\tau) z^{-\tau}, \tag{3.3}$$

gdzie $\boldsymbol{\mathcal{R}}_{\mathbf{xx}}(z) \in \mathbb{C}^{M\mathrm{x}M}$ to wielomianowa macierz przestrzenno-czasowej korelacji w dziedzinie transformacji Z reprezentowanej przez zmienną $z \in \mathbb{C}$. Macierz ta nazywana jest wielomianową, gdyż każdy wyraz tej macierzy jest wielomianem nieskończonego rzędu zmiennej $z$. Równoważnie, można patrzeć na $\boldsymbol{\mathcal{R}}_{\mathbf{xx}}(z)$ jak na wielomian nieskończonego rzędu zmiennej $z$, którego każdy współczynnik jest macierzą zespoloną o wymiarach $M\mathrm{x}M$. Warto zauważyć, że z definicji sygnału $\mathbf{x}(n)$ (2.1) wynika, że macierz $\boldsymbol{\mathcal{R}}_{\mathbf{xx}}(z)$ jest macierzą para-Hermitowską czyli $\boldsymbol{\mathcal{R}}^{H}_{\mathbf{xx}}(1/z) = \boldsymbol{\mathcal{R}}_{\mathbf{xx}}(z)$. W celu ograniczenia rzędu macierzy wielomianowej $\boldsymbol{\mathcal{R}}_{\mathbf{xx}}(z)$ i umożliwienia dalszych obliczeń ustala się, pewną wartość $W$, dla której można przyjąć następujące założenie dotyczące przestrzenno-czasowej korelacji:

$$\forall_{\tau>W \vee \tau<-W} \mathbf{R}_{\mathbf{xx}}(\tau) \approx \underline{\mathbf{0}}, \tag{3.4}$$

gdzie $\underline{\mathbf{0}}$ oznacza macierz zapełnioną samymi zerami. Jest to założenie bardzo istotne. Określa ono, że jeżeli pojedyncza, konkretna próbka sygnału źródłowego $s_k(n)$ po raz pierwszy obserwowana jest w macierzy mikrofonowej w chwili czasowej $t_0$, to po raz ostatni może być obserwowana w tej macierzy mikrofonowej w chwili czasowej $t_0+W$. Poprawna wartość $W$ zależy pośrednio od prędkości dźwięku, geometrii macierzy mikrofonowej oraz również od czasu pogłosu, a bezpośrednio od relacji pomiędzy filtrami $\mathbf{h}_{m,k}$ z równania 2.1, gdyż to te filtry określają jak bardzo poszczególne próbki są przesunięte w czasie względem siebie nawzajem. Z równań 3.2 oraz 3.4 wynika, że równanie 3.3 można przybliżyć do postaci

$$\hat{\boldsymbol{\mathcal{R}}}_{\mathbf{xx}}(z) \approx \sum_{\tau=-W}^{W} \hat{\mathbf{R}}_{\mathbf{xx}}(\tau) z^{-\tau}. \tag{3.5}$$

## 3.3 PEVD macierzy przestrzenno-czasowej korelacji przy pomocy algorytmu SBR2

Uzyskaną para-Hermitowską wielomianową macierz przestrzenno-czasowej korelacji w dziedzinie transformacji Z ($\hat{\boldsymbol{\mathcal{R}}}_{\mathbf{xx}}(z)$) należy w następnym kroku rozłożyć na wartości własne. Sprowadza się to do znalezienia macierzy spełniających następujące równania:

$$\mathbf{H}(z)\hat{\boldsymbol{\mathcal{R}}}_{\mathbf{xx}}(z)\tilde{\mathbf{H}}(z) = \hat{\boldsymbol{\mathcal{D}}}(z), \tag{3.6}$$

$$\mathbf{H}(z)\tilde{\mathbf{H}}(z) = \tilde{\mathbf{H}}(z)\mathbf{H}(z) = \mathbf{I}_M, \tag{3.7}$$

$$\hat{\boldsymbol{\mathcal{D}}}(z) \approx \mathrm{diag}\{\hat{d}_1(z), \hat{d}_2(z), ..., \hat{d}_M(z)\}, \tag{3.8}$$

gdzie, operator $\tilde{[\cdot]}(z)$ oznacza para-Hermitowskie sprzężenie tzn. $\tilde{\mathbf{H}}(z) := \mathbf{H}^H(1/z)$, $\mathbf{I}_M \in \mathbb{R}^{M\text{x}M}$ to macierz tożsamościowa, $\hat{\boldsymbol{\mathcal{D}}}(z) \in \mathbb{C}^{M\text{x}M}$ to w przybliżeniu wielomianowa macierz diagonalna, co zostało wyrażone w równaniu 3.8, $\mathbf{H}(z) \in \mathbb{C}^{M\text{x}M}$ to wielomianowa macierz diagonalizująca, co pokazuje równanie 3.6, oraz para-tożsamościowa, co pokazuje równanie 3.7. Wielomiany $\hat{d}_1, ..., \hat{d}_M$ powinny być ułożone w taki sposób, żeby współczynniki wielomianów $\hat{d}(z)$ stojące przy wyrazie $z^0$ ułożone były w kolejności malejącej. Kolejne współczynniki powiązane są z mocą kolejnych sygnałów źródłowych oraz szumu.

Istnieje kilka metod PEVD macierzy para-Hermitowskiej. Więcej można przeczytać o nich w tych publikacjach: [10, 12, 13]. W tej pracy zdecydowano się na metodę SBR2 [10]. Polega ona na sekwencyjnym diagonalizowaniu macierzy $\hat{\boldsymbol{\mathcal{R}}}_{\mathbf{xx}}(z)$ aż uznamy, że otrzymana macierz w sposób wystarczający przypomina macierz diagonalną. W związku z tym, wielomianowa, para-tożsamościowa macierz diagonalizująca przyjmuje postać:

$$\mathbf{H}(z) = \mathbf{G}_L(z)...\mathbf{G}_2(z)\mathbf{G}_1(z), \tag{3.9}$$

gdzie $L$ to liczba iteracji algorytmu, a $\mathbf{G}_l(z) \in \mathbb{C}^{M\text{x}M}$ to "elementarna", wielomianowa, para-tożsamościowa macierz diagonalizująca, która w każdej iteracji przyjmuje postać:

$$\mathbf{G}(z) = \mathbf{Q}^{(j,k)}(\theta, \phi)\mathbf{B}^{(k,t)}(z), \tag{3.10}$$

gdzie odpowiednie wartości $j, k, t, \theta, \phi$ dobierane są w każdej kolejnej iteracji. Macierz $\mathbf{Q}^{(j,k)}(\theta, \phi) \in \mathbb{C}^{M\text{x}M}$ odpowiedzialna jest za zespoloną elementarną rotację skalarną. Przyjmuje ona postać macierzy tożsamościowej z wyjątkiem podmacierzy $\hat{\mathbf{Q}}(\theta, \phi) \in \mathbb{C}^{2\text{x}2}$ zdefiniowanej na przecięciu rzędów $j$ oraz $k$ z kolumnami $j$ oraz $k$ w sposób następujący:

$$\hat{\mathbf{Q}}(\theta, \phi) = \begin{bmatrix} c & se^{i\phi} \\ -se^{-i\phi} & c \end{bmatrix}, \tag{3.11}$$

gdzie $c = \cos(\theta)$ oraz $s = \sin(\theta)$. Macierz wielomianowa $\mathbf{B}^{(k,t)}(z)$ reprezentuje elementarne opóźnienie o $t \in \mathbb{Z}$ próbek. Przyjmuje ona postać macierzy tożsamościowej z wyjątkiem elementu $(k, k)$, którego wartość wynosi $z^{-t}$ tzn.:

$$\mathbf{B}^{(k,t)}(z) = \begin{bmatrix} \mathbf{I}_{k-1} & 0 & 0 \\ 0 & z^{-t} & 0 \\ 0 & 0 & \mathbf{I}_{M-k} \end{bmatrix}, \tag{3.12}$$

gdzie $t \in \mathbb{Z}$.

Przyjmijmy, że algorytm ma za zadanie zdiagonalizować wielomianową macierz para-Hermitowską $\boldsymbol{\mathcal{R}}(z)$, która jest transformatą Z macierzy $\mathbf{R}(\tau)$ zgodnie z równaniem 3.5. W każdej kolejnej iteracji algorytm znajduje pozadiagonalny współczynnik w diagonalizowanej macierzy wielomianowej $\boldsymbol{\mathcal{R}}(z)$ o największej wartości bezwzględnej. Sprowadza się to do znalezienia takich $j, k, t \in \mathbb{Z}$, gdzie $j \neq k$, dla których wartość $|[\mathbf{R}(t)]_{jk}|$, gdzie $|.|$ oznacza wartość bezwzględną, a $[\mathbf{R}(t)]_{jk}$ oznacza element macierzy $\mathbf{R}(t)$ leżący w wierszu $j$ i kolumnie $k$, jest jak największa. Warto

zauważyć, że ze względu na to, że macierz $\boldsymbol{\mathcal{R}}(z)$ jest macierzą para-Hermitowską, poszukiwania można ograniczyć do górnej połówki diagonalizowanej macierzy, zatem zakłada się, że $j < k$. Po znalezieniu $j, k, t$ należy przeprowadzić elementarne opóźnienie przy użyciu macierzy $\mathbf{B}^{(k,t)}(z)$:

$$\boldsymbol{\mathcal{R}}'(z) = \mathbf{B}^{(k,t)}(z)\boldsymbol{\mathcal{R}}(z)\tilde{\mathbf{B}}^{(k,t)}(z). \tag{3.13}$$

Należy zauważyć, że w wyniku tej operacji największy pozadiagonalny współczynnik $[\mathbf{R}(t)]_{jk}$, który stał przy wyrazie $z^{-t}$ (zgodnie z równaniem 3.5) oraz jego para-Hermitowski odpowiednik $[\mathbf{R}(-t)]^*_{kj}$, gdzie $.^*$ oznacza sprzężenie, który stał przy wyrazie $z^t$ przesuwane są do macierzy stojącej przy wyrazie $z^0$, podczas gdy elementy diagonalne macierzy $\boldsymbol{\mathcal{R}}(z)$ pozostają nienaruszone. W efekcie można zapisać, że $[\mathbf{R}'(0)]_{jk} = [\mathbf{R}(t)]_{jk} = [\mathbf{R}(-t)]^*_{kj} = [\mathbf{R}'(0)]^*_{kj}$. Następnie, dobierane są parametry $\theta$ oraz $\phi$, tak żeby macierz $\mathbf{Q}^{(j,k)}(\theta, \phi)$ diagonalizowała macierz $\mathbf{R}'(0)$, która zawiera największe pozadiagonalne elementy macierzy $\boldsymbol{\mathcal{R}}(z)$. Robi się to poprzez przeniesienie tych elementów na główną diagonalię. Mówiąc dokładniej, parametry dobierane są tak żeby:

$$\begin{bmatrix} c & se^{i\phi} \\ -se^{-i\phi} & c \end{bmatrix} \begin{bmatrix} [\mathbf{R}'(0)]_{jj} & [\mathbf{R}'(0)]_{jk} \\ [\mathbf{R}'(0)]_{kj} & [\mathbf{R}'(0)]_{kk} \end{bmatrix} \begin{bmatrix} c & -se^{i\phi} \\ se^{-i\phi} & c \end{bmatrix} = \begin{bmatrix} [\mathbf{R}''(0)]_{jj} & 0 \\ 0 & [\mathbf{R}''(0)]_{kk} \end{bmatrix}. \tag{3.14}$$

Powyższy warunek spełniony jest, gdy:

$$\phi = \arg([\mathbf{R}'(0)]_{jk}) \tag{3.15}$$

oraz

$$\tan(2\theta) = \frac{2|[\mathbf{R}'(0)]_{jk}|}{[\mathbf{R}'(0)]_{jj} - [\mathbf{R}'(0)]_{kk}}. \tag{3.16}$$

Równanie 3.16 ma wiele rozwiązań. Żeby wartości bezwzględne wartości własnych macierzy po zdiagonalizowaniu uszeregowane były w kolejności malejącej, należy użyć cztero-ćwiartkowego arcus tangensa, czyli takiego, że:

$$\arctan(\frac{a}{b}) = \arg(ia + b). \tag{3.17}$$

Po obliczeniu wartości $\theta, \phi$ należy przeprowadzić rotację skalarną przy użyciu macierzy $\mathbf{Q}^{(j,k)}(\theta, \phi)$:

$$\boldsymbol{\mathcal{R}}''(z) = \mathbf{Q}^{(j,k)}(\theta, \phi)\boldsymbol{\mathcal{R}}'(z)\mathbf{Q}^{(j,k)^H}(\theta, \phi). \tag{3.18}$$

Warto zauważyć, że wpływa ona na wszystkie współczynniki wielomianu $\boldsymbol{\mathcal{R}}'(z)$. Z powyższych równań wynika, że:

$$\boldsymbol{\mathcal{R}}''(z) = \mathbf{G}(z)\boldsymbol{\mathcal{R}}(z)\tilde{\mathbf{G}}(z). \tag{3.19}$$

Równanie 3.19 definiuje jedną iterację algorytmu SBR2. Na koniec każdej iteracji należy podmienić $\boldsymbol{\mathcal{R}}(z) \leftarrow \boldsymbol{\mathcal{R}}''(z)$. Algorytm powtarza się przez $L$ iteracji. Poszukiwaną, diagonalizującą macierz $\mathbf{H}(z)$ wylicza się z równania 3.9. Można ją aktualizować na bieżąco po każdej iteracji. Należy zauważyć, że stopień wielomianu $\boldsymbol{\mathcal{R}}(z)$ rośnie wraz z każdą iteracją algorytmu. Tracąc nieco na dokładności, można temu zapobiec poprzez zerowanie współczynników macierzy stojących przy wyrazach $z^\tau$ dla każdego $\tau > W \vee \tau < -W$.

### 3.3.1 Uproszczenia PEVD

Należy zauważyć, że wszystkie współczynniki macierzy wielomianowej $\hat{\boldsymbol{\mathcal{R}}}_{\mathbf{xx}}(z)$ są liczbami rzeczywistymi. W związku z tym, $\phi$ z równania 3.15 może przyjmować tylko wartości ze zbioru $\{0, \pi\}$. Z tego wynika, że wszystkie wyrazy macierzy $\mathbf{Q}^{(j,k)}(\theta, \phi)$ są rzeczywiste i w konsekwencji wszystkie współczynniki macierzy wielomianowych $\mathbf{G}(z)$ oraz $\boldsymbol{\mathcal{R}}(z)$, w każdej kolejnej iteracji, a także wszystkie współczynniki macierzy diagonalizującej $\mathbf{H}(z)$, również są rzeczywiste.

## 3.4 Filtracja sygnału wejściowego

Pierwszym krokiem w filtracji jest policzenie transformacji Z z sygnału wejściowego $\mathbf{x}(n)$:

$$\boldsymbol{\mathcal{X}}(z) = \sum_{\tau=-\infty}^{\infty} \mathbf{x}(\tau) z^{-\tau}. \tag{3.20}$$

Następnie, sygnał $\boldsymbol{\mathcal{X}}(z)$ filtruje się przy użyciu macierzy $\mathbf{H}(z)$:

$$\boldsymbol{\mathcal{Y}}(z) = \mathbf{H}(z)\boldsymbol{\mathcal{X}}(z), \tag{3.21}$$

gdzie $\boldsymbol{\mathcal{Y}}(z)$ to transformacja z sygnału przefiltrowanego tzn.:

$$\boldsymbol{\mathcal{Y}}(z) = \sum_{\tau=0}^{T} \mathbf{y}(\tau) z^{-\tau}, \tag{3.22}$$

gdzie $\mathbf{y}(n) \in \mathbb{R}^M \wedge n \in \{0, 1, ..., T\}$ to przefiltrowany sygnał. W związku z tym, że niezerowe współczynniki wielomianu $\mathbf{H}(z)$ stoją tylko przy wyrazach $z^{\tau}$, dla których $\tau \leqslant W \wedge \tau \geqslant -W$ oraz dziedzina przefiltrowanego sygnału $\mathbf{y}(n)$ to $\{0, 1, ..., T\}$, sumę z równania 3.20 wystarczy policzyć jedynie w zakresie $\tau \in \{-W, -W+1, ...0, 1, ...T+W\}$. Podobnie jak wcześniej, w przypadku, gdy argument $\tau$ w równaniu 3.20 wykracza poza dziedzinę ramki czasowej $\{0, ..., T\}$, należy użyć wartości spoza ramki czasowej (np. $\mathbf{x}(-1)$). Jeżeli nie ma takiej możliwości, można, przy odpowiednio dużym $T$, przyjąć, że wartości funkcji $\mathbf{x}(n)$ dla $\tau$ spoza przedziału $\{0, ..., T\}$ wynoszą $\underline{\mathbf{0}}$, gdzie $\underline{\mathbf{0}}$ to wektor złożony z samych zer.

Funkcję $\mathbf{y}(n) \in \mathbb{R}^M \wedge n \in \{0, 1, ..., T\}$ można zapisać jako macierz $\mathbf{y} \in \mathbb{R}^{M \mathrm{x} (T+1)}$. Dzięki właściwościom PEVD, kolejne wiersze macierzy $\mathbf{y}$ o numerach od 1 do $K$ są odtworzonymi sygnałami źródłowymi $s_k(n)$ uszeregowanymi w kolejności malejącej pod względem mocy odtworzonego sygnału. Kolejne wiersze macierzy $\mathbf{y}$ o numerach od $K+1$ do $M$ reprezentują szum. W szczególności, pierwszy wiersz macierzy $\mathbf{y}$ to odtworzony, poszukiwany sygnał źródłowy $s_1(n)$ zgodnie z założeniem, że moc poszukiwanego sygnału źródłowego odebrana w macierzy mikrofonowej jest większa niż moc sygnałów zakłócających odebranych w mikrofonach. Gdyby moc ta była mniejsza, to odtworzony, poszukiwany sygnał źródłowy znajdowałby się w którymś z kolejnych rzędów macierzy $\mathbf{y}$. Zachowując jednak powyższe założenie można zapisać, że:

$$s_1(n) \approx y_1(n), \tag{3.23}$$

gdzie $n \in \{0, 1, ..., T\}$, $y_1(n)$ to pierwszy wyraz wektora w funkcji $\mathbf{y}(n)$, czyli pierwszy wiersz macierzy $\mathbf{y}$, a $s_1(n)$ to szukany sygnał źródłowy. Moc odtworzonych sygnałów powiązana jest z diagonalną macierzą $\hat{\boldsymbol{\mathcal{D}}}(z)$ z równania 3.8 w następujący sposób:

$$\mathbb{E}\{|y_k(n)|\} \approx [\hat{\mathbf{D}}(0)]_{kk}, \tag{3.24}$$

gdzie $\hat{\mathbf{D}}(\tau)$ to odwrotna transformata Z macierzy $\hat{\boldsymbol{\mathcal{D}}}(z)$, a $y_k(n)$ to k-ty wiersz macierzy $\mathbf{y}$. Warto zauważyć, że dla macierzy $\hat{\mathbf{D}}(\tau)$ zachodzi również:

$$[\hat{\mathbf{D}}(0)]_{11} \geqslant [\hat{\mathbf{D}}(0)]_{22} \geqslant ... \geqslant [\hat{\mathbf{D}}(0)]_{MM}. \tag{3.25}$$

# Rozdział 4

# Szczegóły implementacyjne

## 4.1 Generator sygnału

Aby wygenerować wieloźródłowe sygnały zawierające pogłos, niezbędne są oryginalne sygnały źródłowe oraz filtry (RIR), które są szczególnymi przypadkami filtrów o skończonej odpowiedzi impulsowej (ang. Finite Impulse Response - FIR).

Nieprzetworzone sygnały źródłowe zostały pobrane z bazy sygnałów mowy EBU SQAM CD [14].

Filtry RIR zostały wygenerowane za pomocą generatora [8], który jest dostępny jako biblioteka dla języka programowania Python. Należy wygenerować jeden filtr RIR dla każdej pary: źródło - mikrofon. Aby uzyskać odpowiednie filtry RIR, potrzebne są następujące informacje:

- wymiary pomieszczenia, w którym rozchodzi się dźwięk,
- czas trwania pogłosu, który zależy od współczynników odbicia od ścian pomieszczenia,
- maksymalny rząd odbić - symulowana, maksymalna ilość odbić sygnału od ścian pomieszczenia zanim dotrze on do mikrofonu,
- prędkość propagacji sygnału,
- współrzędne położenia źródeł dźwięku,
- współrzędne położenia mikrofonów w macierzy mikrofonowej,
- częstotliwość próbkowania. Aby uniknąć aliasingu, częstotliwość próbkowania musi spełniać warunek Nyquista:

  $$f_s \geqslant 2B, \tag{4.1}$$

  gdzie $f_s$ to częstotliwość próbkowania, a $B$ to szerokość pasma sygnału próbkowanego.

Sygnały źródłowe należy przefiltrować przy użyciu odpowiednich filtrów RIR, odpowiadających propagacji od każdego źródła dźwięku do każdego z mikrofonów, i w kolejnym kroku dodać odpowiednie, przefiltrowane sygnały dla każdego mikrofonu. Dodatkowo, należy wprowadzić własny szum mikrofonowy jako addytywny biały szum Gaussowski (ang. Additive White Gaussian Noise - AWGN) [11], który jest nieskorelowany pomiędzy mikrofonami. Powyższe kroki opisane zostały w równaniu 2.1.

## 4.2 Implementacja algorytmu

Mając gotowy generator sygnału można zająć się implementacją algorytmu eliminacji szumów i zakłóceń z sygnału mowy opisanego w rozdziale 3. Należy pamiętać, że algorytm ma działać w czasie rzeczywistym oraz ma być możliwy do zaimplementowania na platformie typu embedded.

### 4.2.1 Język programowania

Zdecydowano się na implementację algorytmu w języku C++. Jest to język bardzo powszechnie używany na urządzeniach embedded oraz w projektach działających w czasie rzeczywistym. Kod w języku C++ ma stosunkowo mały narzut w związku z czym wykonuje się stosunkowo szybko. Co niezwykle istotne dla ograniczonych pamięciowo urządzeń embedded, kod w języku C++ po zbudowaniu zajmuje relatywnie mało miejsca w pamięci, a w trakcie działania nie zużywa przesadnie dużo pamięci RAM. W celu przyspieszenia i ułatwienia obliczeń macierzowych, zdecydowano się na użycie biblioteki Eigen [15]. Jest to biblioteka dla języka C++ stworzona na potrzeby algebry liniowej, w szczególności do operacji na macierzach i wektorach. Jest to biblioteka szybka, szeroko używana i dobrze przetestowana. W celu szybszego działania m.in. unika ona dynamicznej alokacji pamięci oraz, kiedy jest to korzystne, rozwija ona pętle (ang. loop unrolling). Eigen posiada wsparcie dla wielu kompilatorów, co czyni kod łatwym do implementacji na urządzeniach embedded. Biblioteka Eigen może być kompilowana przy użyciu następujących kompilatorów:

- GCC,
- MSVC,
- Intel C++ Compiler,
- LLVM/CLang++,
- XCode 7,
- MinGW,
- QNX QCC.

W celach testowych, algorytm zaimplementowano również w języku Python z wykorzystaniem biblioteki Numpy. Dzięki temu łatwiej można było wyłapać błędy w trakcie implementacji, a także ułatwiło to późniejszą ewaluację rezultatów.

### 4.2.2 Obliczenie wielomianowej macierzy przestrzenno-czasowej korelacji w dziedzinie transformacji Z

Pierwszym krokiem algorytmu jest obliczenie wielomianowej macierzy przestrzenno-czasowej korelacji w dziedzinie transformacji Z. W tym celu należy policzyć najpierw macierz przestrzenno-czasowej korelacji z sygnału wejściowego w funkcji zmiennej $\tau$ z równania 3.2. W celu uniknięcia pętli iterującej po zmiennej $n$ i tym samym przyspieszenia działania algorytmu, można zapisać funkcję $\mathbf{x}(n)$ w macierzy $\mathbf{X} \in \mathbb{R}^{M\text{x}(T+1)}$. Policzenie $\hat{\mathbf{R}}_{\mathbf{xx}}(\tau)$ dla $\tau = 0$ wymaga policzenia $\sum_{n=0}^{T} \mathbf{x}(n)\mathbf{x}^H(n)$ co sprowadza się do zwykłego mnożenia dwóch macierzy: $\mathbf{X}\mathbf{X}^T$. Dla innych wartości $\tau$ należy odpowiednio przesunąć macierz $\mathbf{X}$. Dla wartości funkcji $\mathbf{x}(n)$ spoza dziedziny $\{0, ..., T\}$ przyjęto $\underline{\mathbf{0}}$. W celu przejścia z macierzą do dziedziny transformacji Z (3.5) należy zapisać kolejne wartości funkcji macierzowej $\hat{\mathbf{R}}_{\mathbf{xx}}(\tau)$ w przeciwnej kolejności, w tablicy, której każdy element reprezentuje współczynnik wielomianu zmiennej $z$ stojący przy odpowiednim wyrazie $z^{\tau}$. Można to zrobić już na etapie obliczania macierzy $\hat{\mathbf{R}}_{\mathbf{xx}}(\tau)$.

### 4.2.3 PEVD macierzy przestrzenno-czasowej korelacji przy pomocy algorytmu SBR2

Obliczona w poprzedniej podsekcji macierz $\hat{\mathbf{R}}_{\mathbf{xx}}(\tau)$ jest wejściem dla algorytmu SBR2. W publikacjach [6] oraz [10] macierz diagonalizowano iteracyjnie aż do momentu, gdy ustalony parametr reprezentujący jakość diagonalizacji osiągnie zadany arbitralnie poziom. Pozwala to, w teorii, zapewnić stałą jakość działania algorytmu. W praktyce nie jest to aż tak widoczne, gdyż jakość działania algorytmu w bardzo dużej mierze zależy od zależności między sygnałem źródłowym a sygnałami zakłócającymi. Minusem tej metody jest to, że algorytm działa wtedy w różnym czasie, gdyż liczba iteracji jest niedeterministyczna. Znacznie utrudnia to procesowanie danych w czasie rzeczywistym. Wprowadza to również dodatkowy narzut obliczeniowy związany z obliczaniem wspomnianej miary jakości diagonalizacji. Z tych powodów, w tej pracy, założono, że algorytm wykonywany jest zawsze przez z góry ustaloną liczbę $L$ iteracji. Metoda ta nie cierpi na powyżej wspomniane wady. Macierze $\mathbf{H}(z)$, oraz $\boldsymbol{\mathcal{R}}(z)$ aktualizowane są iteracyjnie zgodnie z równaniami 3.9, 3.13 i 3.18. Tak jak opisano w podsekcji 3.3.1, wszystkie obliczenia wykonywane są na liczbach czysto rzeczywistych.

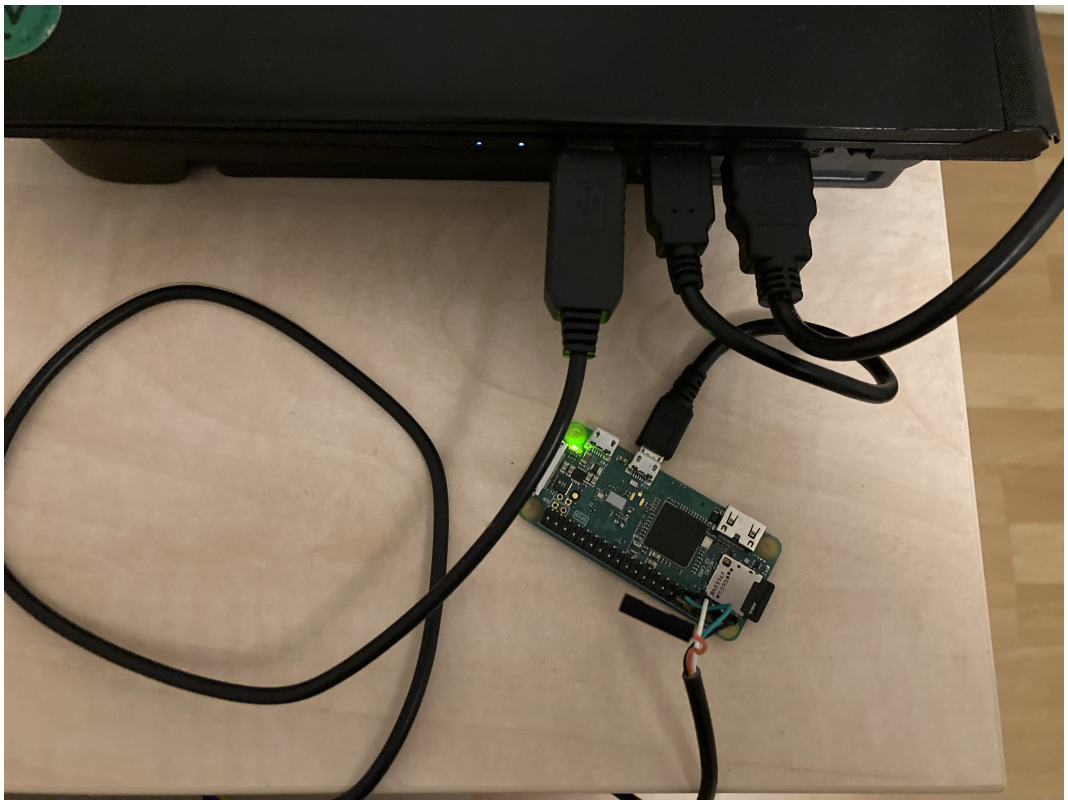
### 4.2.4 Filtracja sygnału wejściowego

Pierwszym etapem filtracji sygnału wejściowego jest przejście z sygnałem $\mathbf{x}(\tau)$ przechowywanym przez macierz $\mathbf{X}$ o wymiarach $M\text{x}(T+1)$ do dziedziny transformacji Z zgodnie z równaniem 3.20. Należy zauważyć, że jest to jedynie kwestia interpretacji. Wystarczy każdą kolejną kolumnę macierzy X traktować jako kolejny współczynnik wielomianu zmiennej $z$ stojący przy odpowiednim wyrazie $z^{-\tau}$. Macierz $\mathbf{H}(z)$ przechowywana jest jako tablica macierzy o wymiarach $M\text{x}M$, gdzie każdy kolejny element tablicy to kolejny współczynnik wielomianu zmiennej $z$. Mnożenie z równania 3.21 sprowadzono do sumy iloczynów kolejnych współczynników wielomianu $\mathbf{H}(z)$

przez wielomian $\boldsymbol{\mathcal{X}}(z)$. Należy zauważyć, że przy każdym takim iloczynie dostajemy macierz o wymiarach $M$x$(T+1)$, której każda kolumna jest kolejnym współczynnikiem wielomianu zmiennej $z$, przy czym wykładniki przy zmiennej $z$ maleją z każdą kolejną kolumną o 1 i kolumna nr 1 stoi przy wyrazie $z^{\tau}$, gdzie $z^{\tau}$ to wyraz przy którym stał współczynnik $\mathbf{H}(z)$ użyty w iloczynie. Należy również zauważyć, że z racji tego, że przefiltrowany sygnał poszukiwany znajduje się w całości w pierwszym wierszu macierzy wynikowej $\boldsymbol{\mathcal{Y}}(z)$, do obliczeń wystarczy brać tylko pierwszy wiersz macierzy $\mathbf{H}(z)$. Cała filtracja sprowadza się zatem do policzenia $2W+1$ mnożeń o postaci wektor razy macierz i sukcesywnego dodawania do siebie wektorów wynikowych odpowiednio przesuniętych zgodnie z tym, przy którym współczynniku wielomianu zmiennej $z$ stoi dany wynik iloczynu.

### 4.2.5 Warstwa sprzętowa

Program zbudowano, uruchomiono i testowano na platformie embedded Raspberry Pi Zero W [16]. Program zbudowano kompilatorem GCC z trzecim poziomem optymalizacji. Na platformie embedded zainstalowany był system operacyjny Raspbian 11 z jądrem systemu Linux 5.15.25+. Platforma Raspberry Pi Zero W posiada jednordzeniowy procesor o częstotliwości taktowania 1 GHz oraz blok pamięci RAM o wielkości 512 MB. Zbudowany program przed uruchomieniem ładowany jest do pamięci RAM. Na rysunku 4.1 zaprezentowano zdjęcie zrobione w trakcie testowania programu. Platforma zasilona była z laptopa. Zestawiono połączenie SSH pomiędzy laptopem a płytką, dzięki, któremu można było korzystać z komend terminala systemu operacyjnego Raspbian 11 działającego na płytce oraz przesyłać pliki pomiędzy płytką a laptopem w celu weryfikacji jakości działania programu. Wszystkie testy czasu działania programu przedstawione w tej pracy przeprowadzane były na wspomnianej platformie Raspberry Pi Zero W.



Rysunek 4.1: Testowanie programu

# Rozdział 5

# Dobór optymalnych parametrów algorytmu

Dobranie odpowiednich parametrów algorytmu jest kluczowe. Parametry te muszą być kompromisem pomiędzy szybkością a jakością działania algorytmu.

## 5.1 Miary ewaluacyjne

W celu obiektywnej oceny działania zaimplementowanego algorytmu przeprowadzono szereg eksperymentów. Jakość działania algorytmu została oceniona na podstawie miar porównujących sygnał idealny, który chcielibyśmy uzyskać, z sygnałem otrzymanym, który jest przybliżeniem sygnału idealnego. Zdecydowano się na skorzystanie z następujących miar:

- Liczony segmentami stosunek sygnału do szumu (ang. Segmental Signal to Noise Ratio - SegSNR) [17],
- Ważony częstotliwościowo liczony segmentami stosunek sygnału do szumu (ang. Frequency-weighted SegSNR - FwSegSNR) [17],
- Krótko-czasowa obiektywna zrozumiałość (ang. Short-Time Objective Intelligibility - STOI) [18] - miara bazująca na ludzkiej percepcji; określa ona jak dobrze sygnał mowy jest zrozumiały dla odbiorcy,
- Percepcyjna ewaluacja jakości mowy (ang. Perceptual Evaluation of Speech Quality - PESQ) [19] - miara bazująca na ludzkiej percepcji; określa ona jak dobra jest jakość odbieranego sygnału mowy,
- Spektralne zniekształcenie typu "Bark"(ang. Bark Spectral Distortion - BSD) [20] - miara określająca jak bardzo sygnał jest zniekształcony w stosunku do oryginału; im mniejsze wartości BSD tym lepiej; najbardziej przydaje się ona do określania jakości usuwania pogłosu.

W pracy bardzo często podawana jest różnica pomiędzy wartością danej miary obliczoną dla sygnału na wyjściu algorytmu, a wartością tej miary obliczoną dla sygnału na wejściu algorytmu, czyli dla nieprzetworzonego sygnału odebranego przez

jeden z mikrofonów. Dzięki temu można pokazać o ile algorytm polepszył sygnał. Uniezależnia to nieco przedstawianą jakość działania algorytmu od jakości sygnałów wejściowych. Taka różnica oznaczana jest symbolem $\Delta$. Dla przykładu, $\Delta\text{SegSNR} = \text{SegSNR}_{\text{out}} - \text{SegSNR}_{\text{in}}$, gdzie $\text{SegSNR}_{\text{out}}$ i $\text{SegSNR}_{\text{in}}$ oznaczają SegSNR policzony kolejno dla danych wyjściowych i wejściowych algorytmu. Dla miar SegSNR, FwSegSNR oraz BSD różnica ta podawana jest w decybelach.

## 5.2 Parametry zewnętrzne

Parametry zewnętrzne to takie, które zależą od środowiska, w którym robione są pomiary oraz od sprzętu, który odbiera sygnał. Algorytm nie ma wpływu na te parametry. Do parametrów zewnętrznych należą:

- Wymiary pomieszczenia, w którym rozchodzi się dźwięk. W tej pracy te wymiary to 10 m x 10 m x 3 m.
- Czas pogłosu.
- Maksymalny rząd odbić. Mówi on o tym po ilu maksymalnie odbiciach sygnał może dotrzeć od źródła do macierzy mikrofonowej. W tej pracy maksymalny rząd odbić ustawiony jest zawsze na nieskończoność. Należy pamiętać, że maksymalna liczba odbić ograniczona jest też w sposób niebezpośredni przez czas pogłosu.
- Prędkość rozchodzenia się sygnału. W tej pracy przyjęto przybliżoną prędkość dźwięku czyli 343 m/s.
- Liczba źródeł zakłócających $(K-1)$. Ważne, żeby liczba źródeł zakłócających oraz źródło pożądane były łącznie mniejsze niż liczba mikrofonów $(K < M)$.
- Położenie źródeł sygnału. W tej pracy założono, że źródła sygnału położone są na wysokości 1 m i w odległości 2 m od macierzy mikrofonowej.
- Specyfika odbieranych sygnałów. W tej pracy jest to sygnał mowy o pasmie częstotliwości $[0, 8]$ kHz.
- Częstotliwość próbkowania. W tej pracy częstotliwość próbkowania wynosiła 16 kHz, co, z racji tego, że maksymalna częstotliwość sygnału odbieranego wynosiła 8 kHz spełnia warunek Nyquista.
- Położenie macierzy mikrofonowej. W tej pracy macierz mikrofonowa jest umieszczona w odległości 5,5 m od dwóch ścian na wysokości 1 m.
- Maksymalna liczba mikrofonów macierzy mikrofonowej $(M_{\max})$. Algorytm może korzystać z mniejszej liczby mikrofonów niż maksymalna w celu przyspieszenia obliczeń.

- Kształt macierzy mikrofonowej. W tej pracy przyjmuje on formę okręgu o promieniu 10 cm, na którym mikrofony rozmieszczone są równomiernie. Aby uniknąć problemu aliasingu przestrzennego [21], odległość maksymalna pomiędzy mikrofonami w macierzy mikrofonowej powinna być mniejsza niż połowa długości najkrótszej fali w sygnale:

$$d_{\max} < \frac{\lambda_{\min}}{2}, \tag{5.1}$$

$$d_{\max} < \frac{v}{2f_{\max}}, \tag{5.2}$$

  gdzie $d_{\max}$ to odległość maksymalna pomiędzy mikrofonami w macierzy mikrofonowej, $\lambda_{\min}$ to długość najkrótszej fali w sygnale, a $f_{\max}$ to największa częstotliwość składowa w sygnale. W praktyce, bardziej preferowane są macierze mikrofonowe o nieco większych rozmiarach, ponieważ wtedy można zaobserwować bardziej zróżnicowane wartości sygnału w poszczególnych mikrofonach w danej chwili czasowej.

- Stosunki mocy sygnału pożądanego do mocy sygnałów zakłócających. Ważne, aby stosunki te były większe od 1, czyli żeby sygnał pożądany był głośniejszy niż sygnały zakłócające.

- Stosunek sygnału do szumu (ang. Signal to Noise Ratio - SNR) rozumiany jako stosunek mocy sumy sygnału pożądanego i sygnałów zakłócających do mocy szumu addytywnego.

## 5.3 Parametry wewnętrzne - dobór odpowiednich wartości

Parametry wewnętrzne to takie, które zależą od algorytmu. Bardzo ważny jest odpowiedni dobór parametrów wewnętrznych, żeby algorytm mógł działać w czasie rzeczywistym i jednocześnie osiągać jak najlepsze rezultaty. Dobór takich parametrów wiąże się zazwyczaj ze znalezieniem odpowiedniego kompromisu pomiędzy czasem wykonywania algorytmu a jakością jego działania. Można wyróżnić następujące parametry wewnętrzne algorytmu:

- Parametr $W$ odpowiedzialny za stopień wielomianu zmiennej $z$.

- Parametr $L$ oznaczający liczbę iteracji w algorytmie PEVD.

- Parametr $T$ oznaczający długość ramki czasowej, na której operuje algorytm.

- Parametr $M$ oznaczający liczbę mikrofonów, na których operuje algorytm. Z uwagi na ograniczenia sprzętowe liczba ta nie może przekraczać liczby mikrofonów w macierzy mikrofonowej ($M \leqslant M_{\max}$).

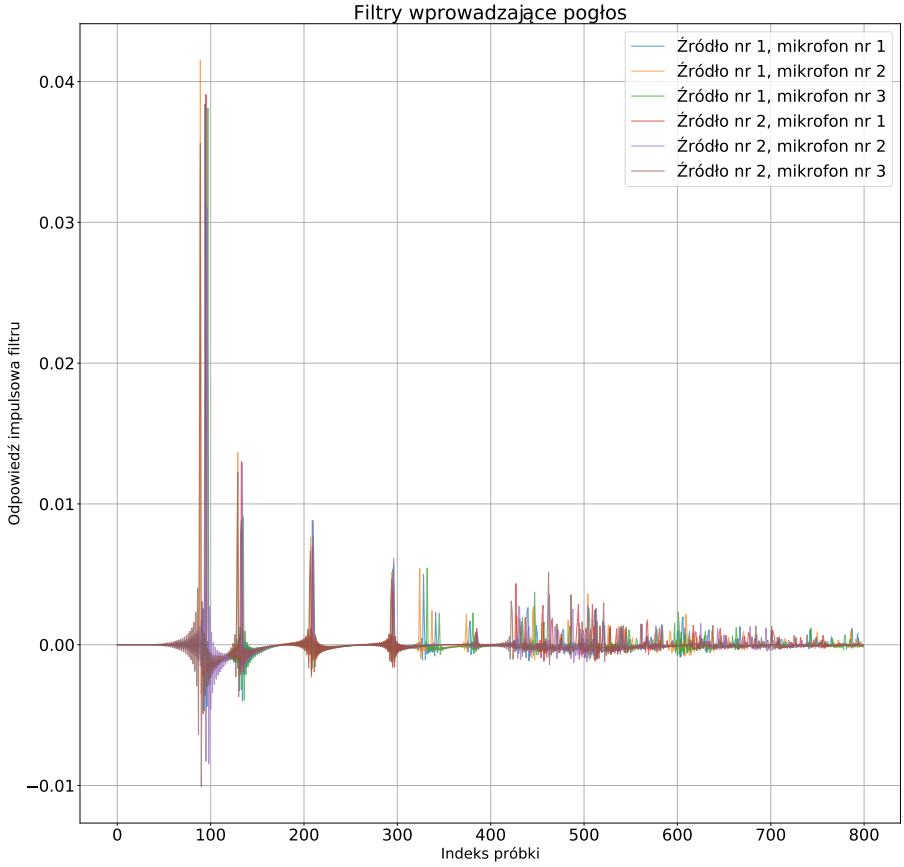
### 5.3.1 Parametr $W$

Parametr $W$ odpowiedzialny jest za stopień wielomianu zmiennej $z$. W teorii powinien być on tak duży jak daleko sięgać może korelacja czasowo-przestrzenna w sygnale odebranym zgodnie z równaniem 3.4. Z drugiej strony parametr ten znacząco wpływa na czas obliczeń. Należy pamiętać, że długości tablic zawierających kolejne współczynniki macierzy wielomianowych zmiennej $z$ $\hat{\boldsymbol{\mathcal{R}}}_{\mathbf{xx}}(z)$ oraz $\mathbf{H}(z)$ wynoszą $2W + 1$.

Na rysunku 5.1 przedstawiono odpowiedzi impulsowe filtrów odpowiedzialnych za wprowadzenie pogłosu - $\mathbf{h}_{m,k}$ z równania 2.1. Filtry wygenerowano dla czasu pogłosu ustawionego na wartość 0,2 s, dla liczby źródeł $K = 2$, dla liczby mikrofonów $M = 3$. Widać, że filtry te są stosunkowo długie. Na rysunku pokazano tylko początek tych filtrów. Maksymalna długość tych filtrów wynosi w teorii około 3300 próbek. Można to policzyć jako suma długości drogi bezpośredniej (około 100 próbek) i iloczynu czasu pogłosu i częstotliwości próbkowania ($0,2\text{s} \cdot 16\text{kHz}$). Sugerowałaby to, że wartość $W$ powinna wynosić około 3200, gdyż dany sygnał możemy obserwować najwcześniej w próbce o numerze ≈100 a najpóźniej w próbce o numerze ≈3300. Taka wartość sprawiłaby jednak, że obliczenia trwałyby bardzo długo. Na wykresie widać wyraźnie pierwsze kilka największych peaków, które odpowiedzialne są za drogę bezpośrednią oraz pierwsze odbicia sygnałów od ścian. Widać też, że im późniejsze odbicia tym mniejsza ich moc, co pokazuje, że odpowiednie zmniejszenie wartości $W$ może nie odbić się szczególnie szkodliwie na działaniu algorytmu, gdyż wartości funkcji korelacji przestrzenno-czasowej $\hat{\mathbf{R}}_{\mathbf{xx}}(\tau)$ z równania 3.2 dla dużych $\tau$ będą stosunkowo małe.

Na rysunku 5.2 przedstawiono wykres maksymalnych wartości bezwzględnych macierzy przestrzenno czasowej korelacji $\hat{\mathbf{R}}_{\mathbf{xx}}(\tau)$ w zależności od przesunięcia czasowego $\tau$. Czas pogłosu ustawiono podobnie jak wcześniej na 0,2 s, liczba mikrofonów to $M = 5$, a liczba źródeł to, podobnie jak wcześniej, $K = 2$. Widać, że duże wartości korelacji obserwowane są tylko dla małych bezwzględnych wartości $\tau$. Warto też zauważyć, że wykres jest symetryczny. Wynika to z tego, że macierz wielomianowa $\hat{\boldsymbol{\mathcal{R}}}_{\mathbf{xx}}(z)$ Jest macierzą para-Hermitowską, co jest mocno wykorzystywane przez algorytm.

Na rysunkach 5.3 oraz 5.4 można zobaczyć jaki wpływ ma parametr $W$ na jakość oraz czas działania algorytmu. W tych oraz kolejnych eksperymentach w tej pracy, o ile nie jest powiedziane inaczej, czas pogłosu ustawiono na 0,2 s, liczba mikrofonów to $M = 5$, liczba źródeł to $K = 2$, SNR to 20 dB, stosunek mocy sygnału pożądanego do sygnały zakłócającego to $\approx 11,11$, a liczba iteracji to $L = 15$. Czas trwania nagrania to 7 s, a ramka czasowa została ustawiona na cały czas trwania nagrania, czyli $T = 111998$. Na rysunku 5.3 pokazano jak zmienia się wartość $\Delta$FwSegSNR. Na rysunku 5.4 pokazano jak zmienia się wartość $-\Delta$BSD. BSD jest parametrem, który jest tym lepszy im jego wartość jest mniejsza, dlatego na wykresie wpływ algorytmu wyświetlony jest ze znakiem $-$, żeby większa wartość na wykresie odpowiadała większej jakości działania algorytmu. Można zauważyć, że jakość działania



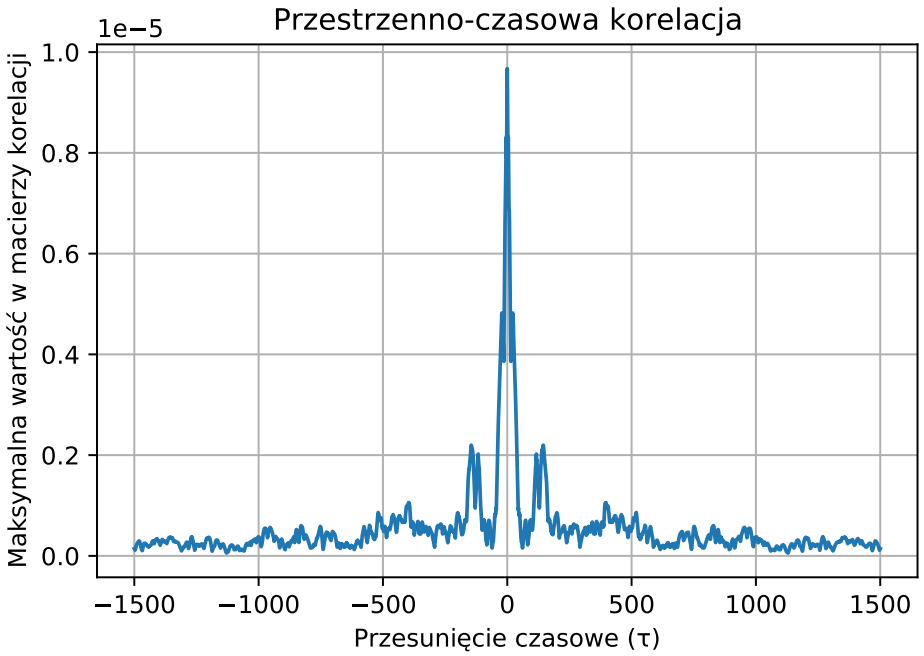

Rysunek 5.1: Wykresy odpowiedzi impulsowych filtrów wprowadzających pogłos

algorytmu od pewnego momentu się nie poprawia oraz, że czas działania algorytmu rośnie liniowo i mniej więcej dla $W = 30$ czas ten przekracza czas trwania nagrania, czyli algorytm przestaje działać w czasie rzeczywistym. Do dalszych eksperymentów wybrano więc wartość $W = 20$. Dla tej wartości zmierzono $\Delta\text{FwSegSNR} = 3,34\text{dB}$ oraz $\Delta\text{BSD} = -4,80\text{dB}$

### 5.3.2 Parametr $L$

Parametr oznacza liczbę iteracji algorytmu PEVD. Większa liczba iteracji prowadzi do lepszej diagonalizacji macierzy wielomianowej $\hat{\boldsymbol{\mathcal{R}}}_{\mathbf{xx}}(z)$. Z drugiej strony, im większa liczba iteracji, tym więcej czasu trwa działanie algorytmu.

Na rysunku 5.5 przedstawiono, w funkcji liczby iteracji $L$, pozadiagonalny współ-



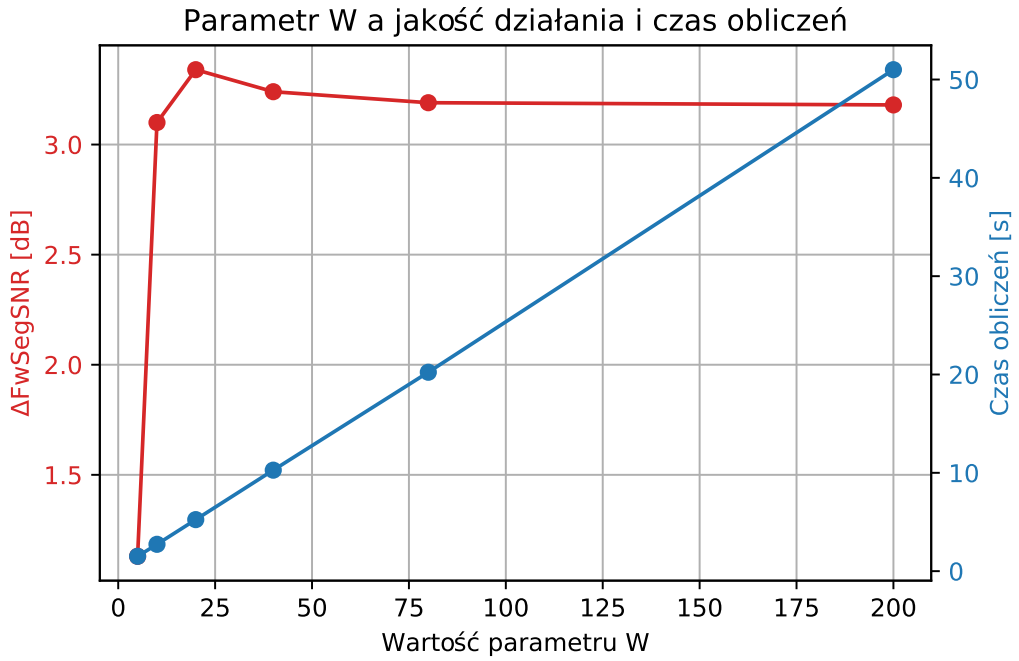

Rysunek 5.2: Wykres maksymalnych bezwzględnych wartości macierzy przestrzenno-czasowej korelacji $\hat{\mathbf{R}}_{\mathbf{xx}}(\tau)$ w zależności od przesunięcia czasowego $\tau$

czynnik w diagonalizowanej macierzy wielomianowej $\boldsymbol{\mathcal{R}}(z)$ o największej wartości bezwzględnej $|[\mathbf{R}(t)]_{jk}|$, który wyliczany jest w trakcie działania algorytmu PEVD w każdej iteracji. Eksperymenty przeprowadzono dla liczby źródeł $K = 3$, takiej samej mocy drugiego źródła zakłócającego co moc pierwszego źródła zakłócającego i SNR = 10dB. Na rysunku 5.6 można zobaczyć jak zmienia się miara $\Delta$FwSegSNR oraz czas wykonania programu w zależności od liczby iteracji ($L$) algorytmu PEVD. Można zauważyć, że jakość działania algorytmu do pewnego momentu rośnie. Widać też, że czas działania algorytmu zmienia się liniowo i bardzo nieznacznie. Algorytm przestaje działać w czasie rzeczywistym dopiero dla $L \approx 200$. Przy tak dobranych parametrach, szczególnie przy tak dużej wartości $T$, zmiana wartości $L$ ma bardzo mały wpływ na czas działania algorytmu. Do dalszych eksperymentów przyjęto $L = 15$.

### 5.3.3 Parametr $M$

Parametr $M$ oznacza liczbę mikrofonów. Im więcej mikrofonów tym więcej informacji o sygnale i potencjalnie lepsze wyniki. Wzrastająca liczba mikrofonów sprawia jednak, że rozmiary wszystkich macierzy, które używane są w algorytmie rosną. W szczególności rozmiary macierzy $\hat{\boldsymbol{\mathcal{R}}}_{\mathbf{xx}}(z)$ oraz $\mathbf{H}(z)$ rosną wprost proporcjonalnie do $M^2$.

Na rysunkach 5.7 oraz 5.8 można zobaczyć jaki wpływ ma parametr $M$ na jakość

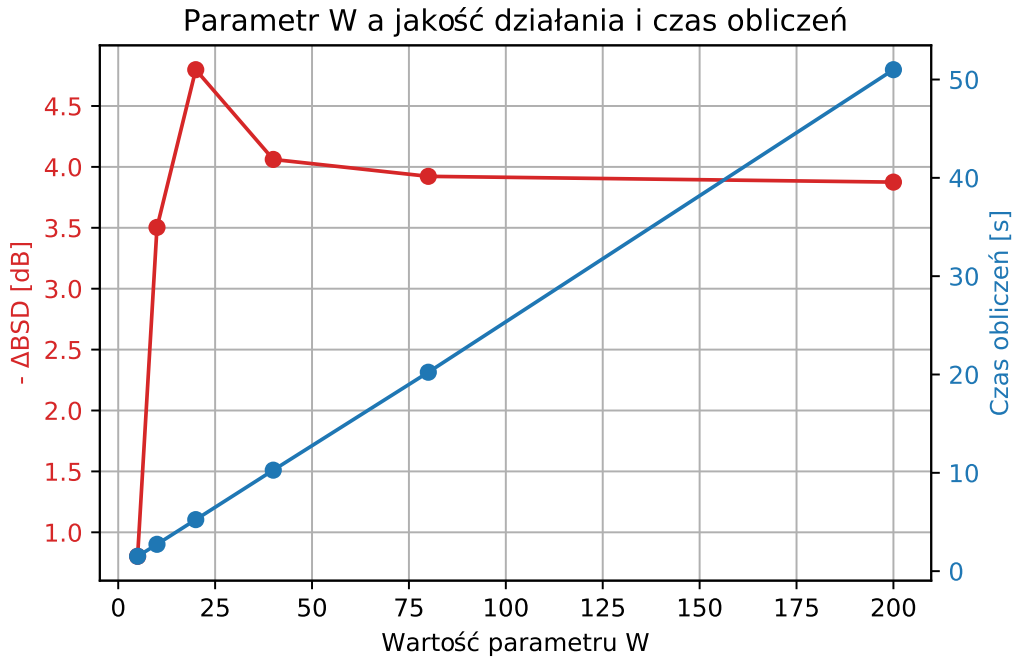



Rysunek 5.3: Parametr $W$ a miara ΔFwSegSNR i czas obliczeń

oraz czas działania algorytmu. Na rysunku 5.7 pokazano jak zmienia się wartość ΔFwSegSNR. Na rysunku 5.8 pokazano jak zmienia się wartość ΔPESQ. Widać, że czas działania zmienia się wraz z kwadratem liczby mikrofonów. Jakość działania algorytmu z początku poprawia się szybko a potem wolniej. Przy $M = 10$ algorytm nie działa już w czasie rzeczywistym. Do dalszych obliczeń przyjęto $M = 5$. Dla takiej wartości $M$ otrzymano $\Delta\text{PESQ} = 0,19$.

### 5.3.4 Parametr $T$

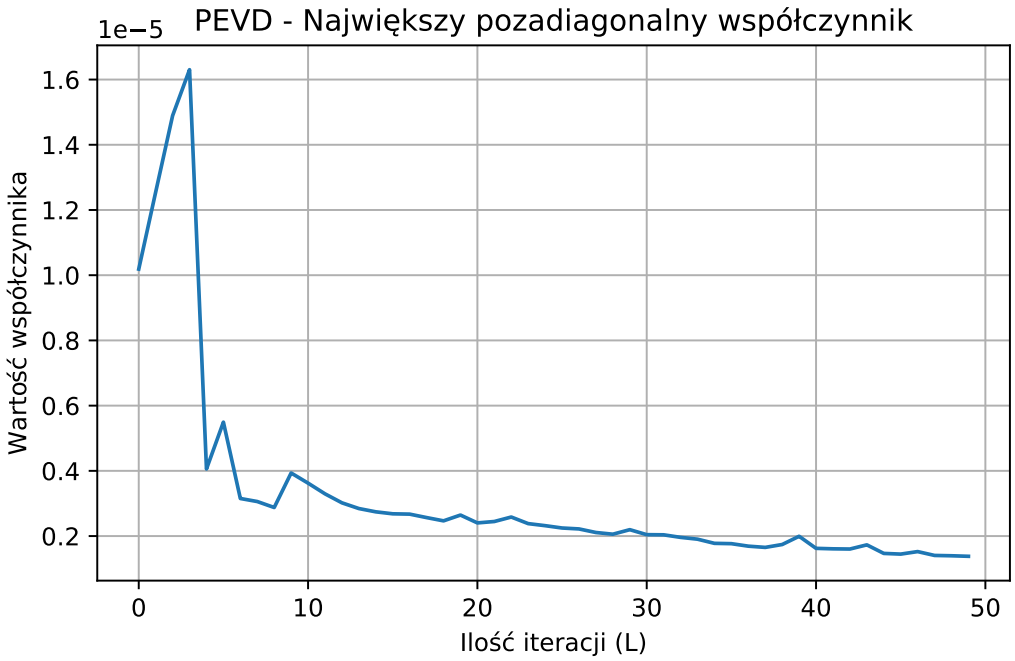
Parametr $T$ oznacza długość ramki czasowej, która przeliczana jest w algorytmie. Parametr ten musi być odpowiednio długi, tak aby przybliżenie wartości oczekiwanej z równania 3.1 poprzez równanie 3.2 było odpowiednio dokładne. Z drugiej strony, należy pamiętać, że w algorytmie założono, że pozycje źródeł i macierzy mikrofonowej są stacjonarne czyli nie zmieniają się w czasie. Oznacza to, że macierz przestrzenno-czasowej korelacji jest stała w czasie. W środowisku rzeczywistym nie jest to do końca prawdą i duże wartości parametru $T$ mogą zbyt bardzo uśredniać macierz przestrzenno-czasowej korelacji, podczas gdy mniejsze $T$ lepiej mogłoby oddać chwilowe zmiany w tej macierzy. Z perspektywy czasu wykonania algorytmu, większe $T$ sprawia, że sygnał dzieli się na mniejszą liczbę ramek i w związku z tym rzadziej trzeba przeliczać algorytm PEVD oraz macierze przestrzenno-czasowej korelacji. Z drugiej strony im większe $T$ tym dłużej trwa przeliczanie macierzy przestrzenno-czasowej korelacji i filtrowanie sygnału w każdej ramce. Warto pamiętać, że parametr $T$ wprowadza grupowe opóźnienie w przetwarzaniu sygnału - musimy naj-




Rysunek 5.4: Parametr $W$ a miara $-\Delta$BSD i czas obliczeń

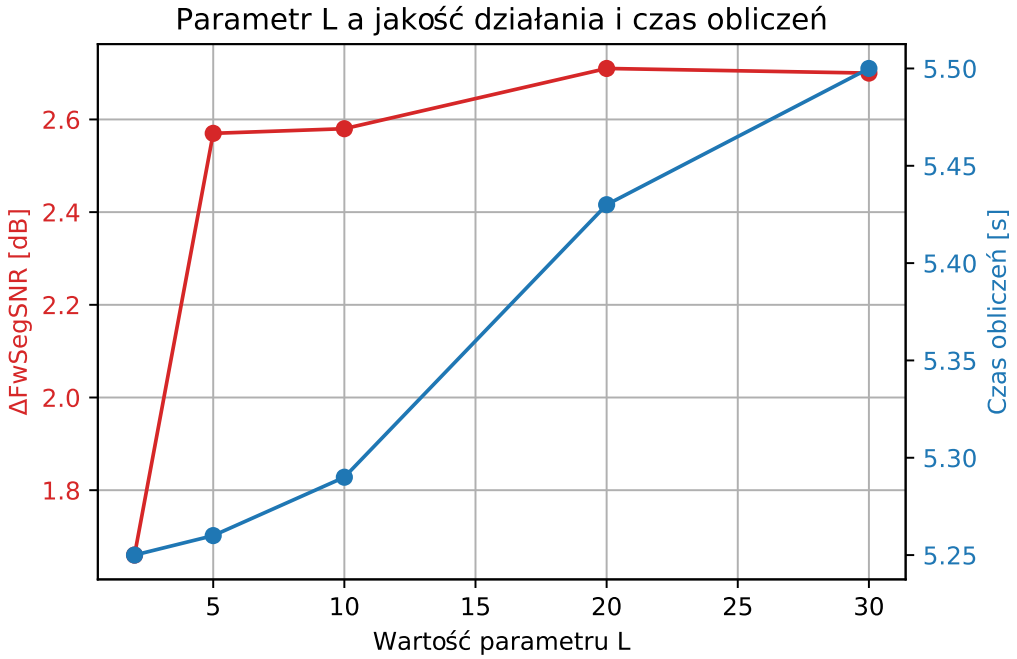
pierw zebrać $T$ próbek, żeby przeliczyć z nich PEVD i przefiltrować sygnał. Macierz przestrzenno-czasowej korelacji można przeliczać na bieżąco w trakcie odbierania próbek. Im większe $T$ tym większe opóźnienie grupowe.

Na rysunkach 5.9 oraz 5.10 można zobaczyć jaki wpływ ma parametr $T$ na jakość oraz czas działania algorytmu. Warto przypomnieć, że te oraz wcześniejsze eksperymenty przeprowadzane są na nagraniu o długości 7 sekund. Wartości $T$ przeliczone zostały na sekundy poprzez podzielenie długości ramki czasowej przez częstotliwość próbkowania czyli 16 kHz. Na przykład, dla $T \approx 2s$ program podzielił sygnał na cztery ramki i dla każdej ramki osobno przeprowadzał PEVD. Na rysunku 5.9 pokazano jak zmienia się wartość $\Delta$FwSegSNR. Na rysunku 5.10 pokazano jak zmieniają się wartości $\Delta$PESQ oraz $\Delta$STOI. Widać, że dla dużych $T$, jakość działania algorytmu jest bardzo podobna. Gdy $T$ zaczyna robić się stosunkowo małe, jakość działania zaczyna spadać, gdyż macierz przestrzenno-czasowej korelacji nie jest aż tak dobrze uśredniona. Czas wykonania algorytmu praktycznie się nie zmienia. Dzieje się tak, dlatego że przy liczbie iteracji PEVD $L = 15$, ta część algorytmu wykonywana jest dużo szybciej niż liczenie macierzy przestrzenno-czasowej korelacji oraz filtrowanie sygnału. Czas liczenia macierzy przestrzenno-czasowej korelacji i filtrowania zwiększa się liniowo wraz ze wzrostem $T$. Z drugiej strony, liczba ramek czasowych maleje liniowo wraz ze wzrostem $T$. Sumarycznie czas wykonania algorytmu się nie zmienia. Algorytm działa w czasie rzeczywistym niezależnie od wartości parametru $T$. Do dalszych obliczeń przyjęto $T = 111998$ rezygnując tym samym z mniejszego

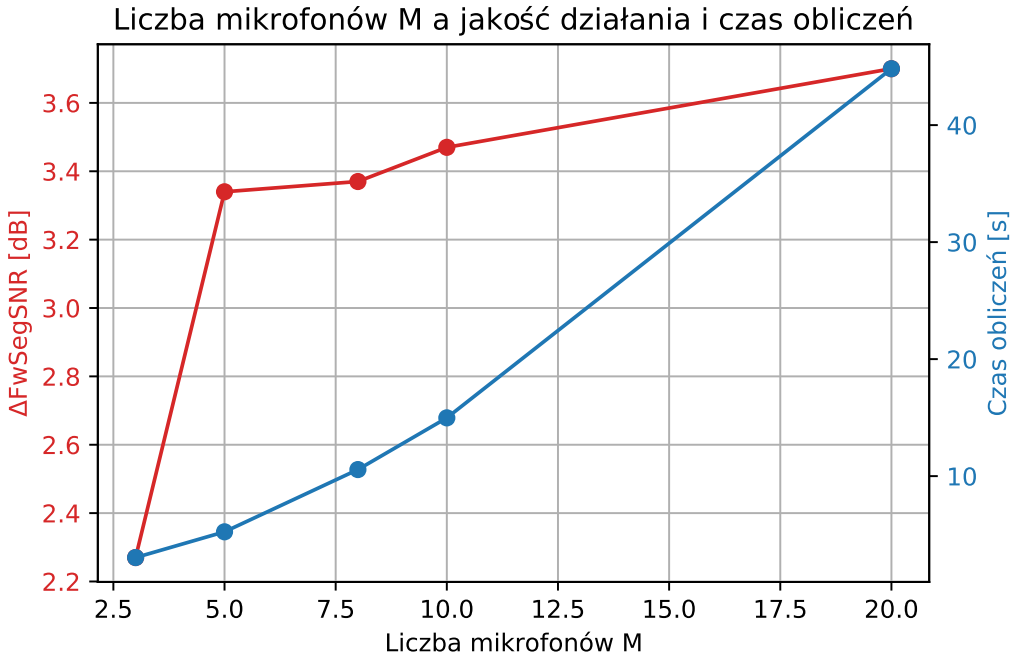



Rysunek 5.5: Największy pozadiagonalny współczynnik $|[\mathbf{R}(t)]_{jk}|$ a ilość iteracji $L$

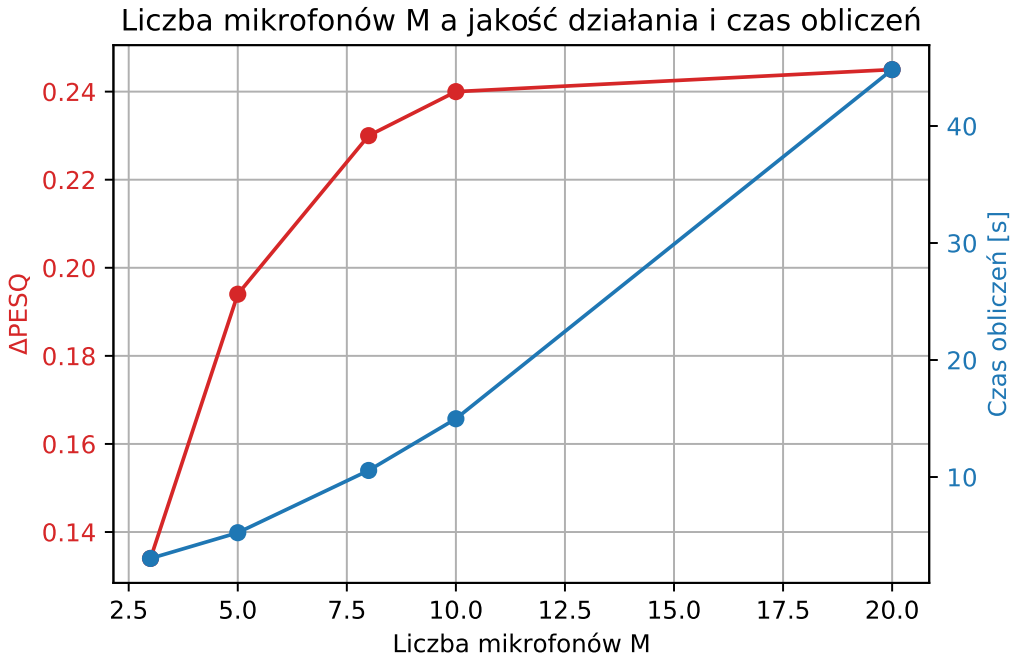
opóźnienia grupowego na rzecz wyższej jakości działania. Dla takiej wartości $T$ otrzymano $\Delta\text{STOI} = 0.087$. W tym eksperymencie położenie źródeł i mikrofonów było stałe. W innym scenariuszu korzystniejsze byłoby wybranie mniejszej wartości $T$.



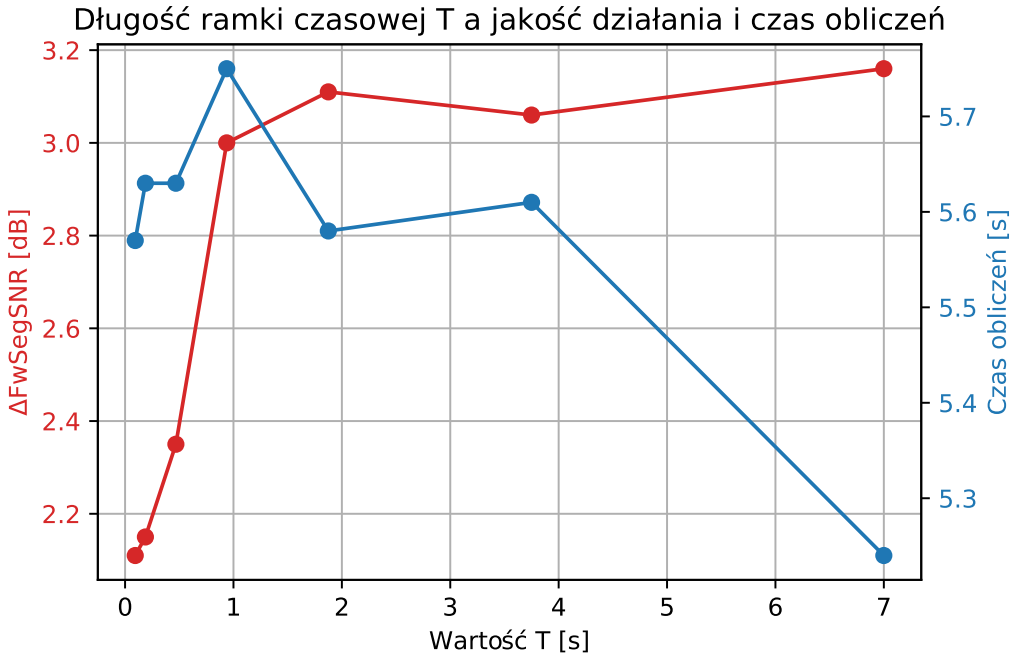

Rysunek 5.6: Parametr $L$ a miara ΔFwSegSNR i czas obliczeń



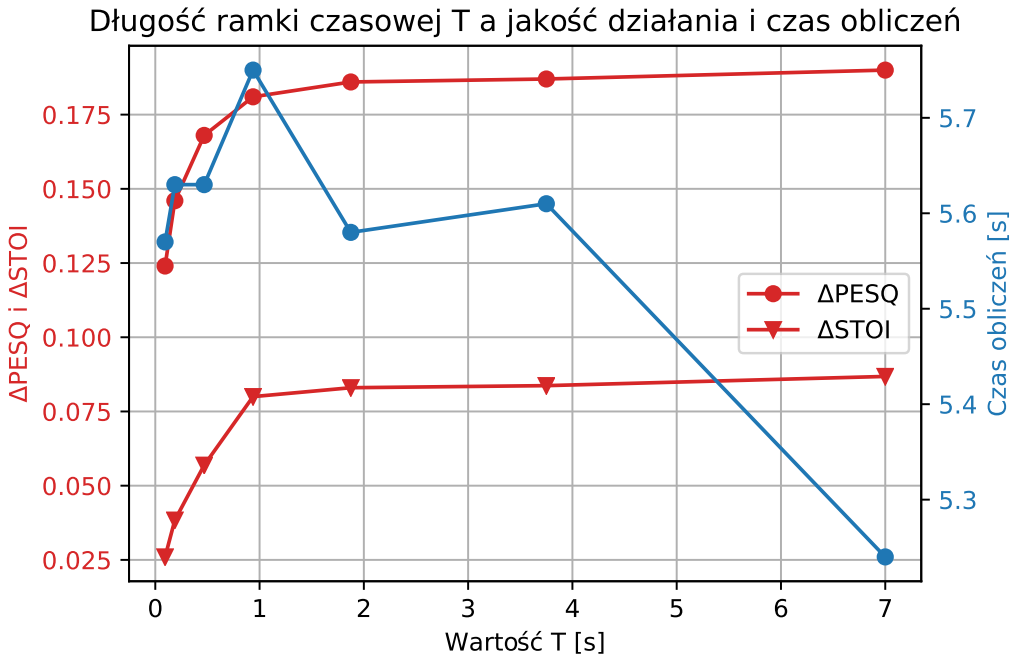

Rysunek 5.7: Liczba mikrofonów $M$ a miara ΔFwSegSNR i czas obliczeń




Rysunek 5.8: Liczba mikrofonów $M$ a miara ΔPESQ i czas obliczeń




Rysunek 5.9: Długość ramki czasowej $T$ przeliczona na sekundy a miara ΔFwSegSNR i czas obliczeń




Rysunek 5.10: Długość ramki czasowej $T$ przeliczona na sekundy a miary ΔPESQ, ΔSTOI oraz czas obliczeń

# Rozdział 6

# Weryfikacja eksperymentalna oraz jej rezultaty

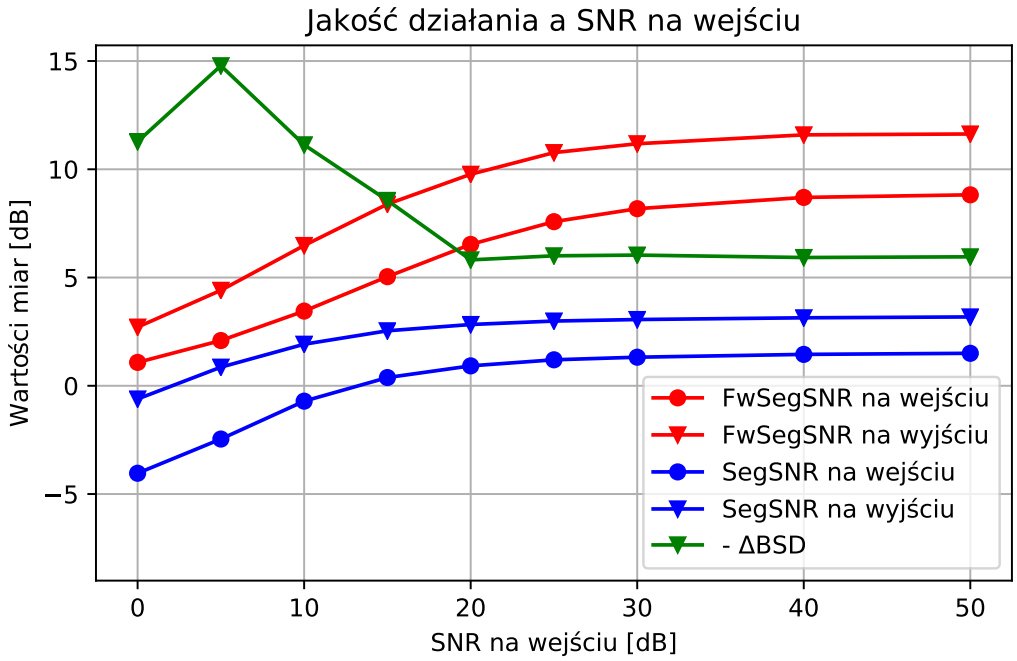
Do ostatecznej weryfikacji eksperymentalnej użyto parametrów wewnętrznych ustalonych w poprzednim rozdziale tzn. $W = 20, L = 15, M = 5, T = 111998$. Dla takich wartości, 7-sekundowy sygnał przetwarzany jest na platformie Raspberry Pi Zero W w czasie 5,24 s. Parametry zewnętrzne opisane zostały w poprzednim rozdziale.

## 6.1 Jakość działania a SNR na wejściu

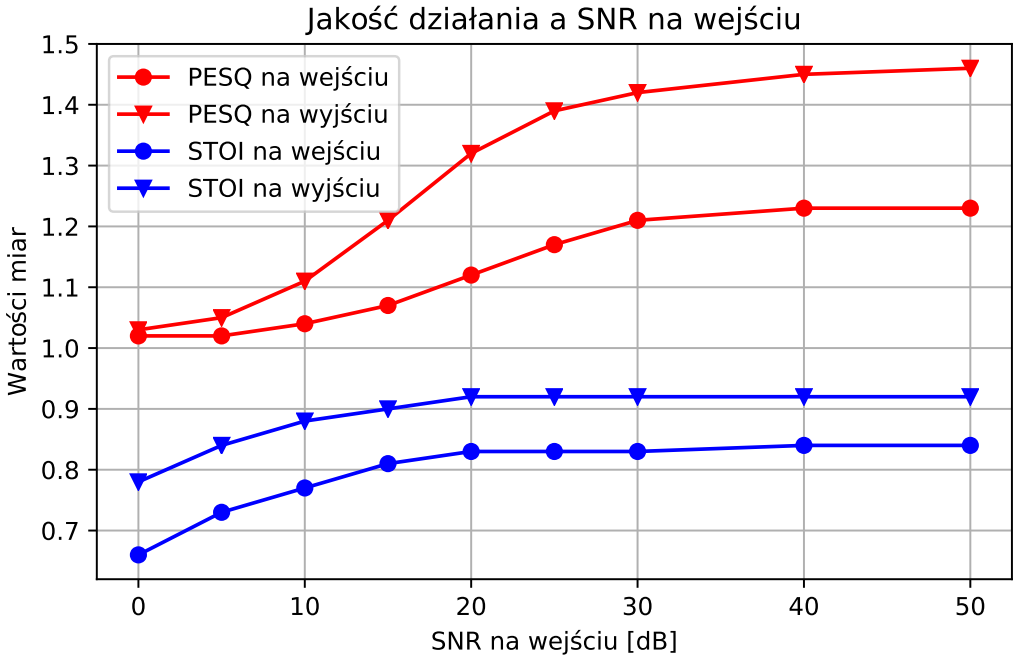
Na rysunkach 6.1 oraz 6.2 przedstawiono jak zmieniają się miary oceniające jakość działania algorytmu wraz ze zmianą SNR na wejściu. Widać, że algorytm z podobną jakością zwiększa parametry FwSegSNR i SegSNR niezależnie od ich wartości wejściowych. Algorytm najlepiej radzi sobie z usuwaniem pogłosu , gdy SNR wejściowe jest niskie o czym świadczy wykres miary $\Delta$BSD. Algorytm najlepiej radzi sobie ze zwiększaniem jakości mowy (PESQ) dla większych wartości SNR na wejściu. Zrozumiałość mowy (STOI) zwiększana jest o podobną wartość niezależnie od wejściowej wartości STOI.

## 6.2 Jakość działania a liczba źródeł zakłócających

W tabeli 6.1 przedstawiono w sposób liczbowy miary jakości działania algorytmu dla SNR na wejściu wynoszącego 20 dB oraz, zarówno jednego źródła zakłócającego (łączna liczba źródeł $K = 2$), jak i dwóch źródeł zakłócających ($K = 3$). Widać, że algorytm radzi sobie dobrze w obu przypadkach, przy czym lepiej wypada, gdy




Rysunek 6.1: Miary FwSegSNR, SegSNR i BSD a SNR na wejściu




Rysunek 6.2: Miary PESQ i STOI na SNR wejściu

źródło zakłócające jest jedno.

Tabela 6.1: Miary jakości

| Miary jakości działania algorytmu dla różnych ilości źródeł | | | | | |
|---|---|---|---|---|---|
| Łączna liczba źródeł ($K$) | FwSegSNR [dB] | SegSNR [dB] | BSD | PESQ | STOI |
| | Na wejściu | | | | |
| $K = 2$ | 6,48 | 0,91 | $5{,}34 \cdot 10^5$ | 1,12 | 0,82 |
| $K = 3$ | 5,42 | -0,56 | $23{,}99 \cdot 10^5$ | 1,09 | 0,77 |
| | Na wyjściu | | | | |
| $K = 2$ | 9,76 | 2,83 | $1{,}28 \cdot 10^5$ | 1,32 | 0,91 |
| $K = 3$ | 8,32 | 1,16 | $7{,}49 \cdot 10^5$ | 1,20 | 0,87 |
| | Poprawa | | | | |
| $K = 2$ | 3,28 | 1,92 | -6,20 dB | 0,20 | 0,09 |
| $K = 3$ | 2,90 | 1,72 | -5,06 dB | 0,11 | 0,10 |

# Rozdział 7

# Podsumowanie

W pracy zaimplementowano na platformie embedded - Raspberry Pi Zero W program pozwalający na eliminacje szumów i zakłóceń z sygnału mowy w czasie rzeczywistym. Do uzyskania zamierzonego efektu użyto metody PEVD rozkładu przestrzeni sygnału odebranego na przestrzenie sygnałów źródłowych i przestrzeń szumu, która korzysta z metody diagonalizacji macierzy wielomianowej SBR2.

W ramach ewaluacji poprawności działania zaimplementowanego programu przeprowadzono testy, w których badane były zarówno poszczególne etapy algorytmu, jak i czas oraz jakość działania programu jako całość. Uzyskano rezultaty świadczące o wysokiej jakości eliminacji szumów i zakłóceń z sygnału mowy przez program przy jednoczesnym działaniu programu w czasie rzeczywistym.

W przyszłości należałoby zastanowić się nad sposobem na zminimalizowanie opóźnienia grupowego wprowadzanego przez program. Można w tym celu obliczać wielomianową macierz przestrzenno-czasowej korelacji z wcześniejszych próbek sygnału a filtr wynikający z jej diagonalizacji stosować na próbkach późniejszych. Takie rozwiązanie spotęgowałoby jednak problem niestacjonarności tej macierzy w czasie. Można w większych szczegółach rozważyć to zagadnienie i próbować znaleźć pewien kompromis między dużym opóźnieniem grupowym a problemami wynikającymi z niestacjonarności. Należałoby też rozważyć dalsze przyspieszenie działania algorytmu. Można w tym celu użyć nowszej, wielordzeniowej wersji Raspberry Pi i przeprowadzać część obliczeń w sposób równoległy. Przyspieszenie algorytmu pozwoliłoby na zwiększenie liczby mikrofonów w programie, co z kolei poprawiłoby jakość jego działania. Można też spróbować zaimplementować metodę kształtowania wiązki jako część wstępną działającą przed algorytmem PEVD. Znacząco poprawiłoby to jakość działania programu, jednak zaimplementowanie tego w czasie rzeczywistym mogłoby być bardzo trudne.

# Bibliografia


[1] Tuomas Virtanen. "Monaural Sound Source Separation by Nonnegative Matrix Factorization With Temporal Continuity and Sparseness Criteria". W: *IEEE Transactions on Audio, Speech, and Language Processing* 15.3 (2007), s. 1066–1074. DOI: 10.1109/TASL.2006.885253.

[2] Y. Ephraim i H.L. Van Trees. "A signal subspace approach for speech enhancement". W: *IEEE Transactions on Speech and Audio Processing* 3.4 (1995), s. 251–266. DOI: 10.1109/89.397090.

[3] Yi Hu i Philipos C. Loizou. "A subspace approach for enhancing speech corrupted by colored noise". W: *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*. T. 1. 2002, s. I-573-I–576. DOI: 10.1109/ICASSP.2002.5743782.

[4] Jingdong Chen i in. "New insights into the noise reduction Wiener filter". W: *IEEE Transactions on Audio, Speech, and Language Processing* 14.4 (2006), s. 1218–1234. DOI: 10.1109/TSA.2005.860851.

[5] Thomas Dietzen i in. "Integrated Sidelobe Cancellation and Linear Prediction Kalman Filter for Joint Multi-Microphone Speech Dereverberation, Interfering Speech Cancellation, and Noise Reduction". W: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020), s. 740–754. DOI: 10.1109/TASLP.2020.2966869.

[6] Vincent W. Neo, Christine Evers i Patrick A. Naylor. "Enhancement of Noisy Reverberant Speech Using Polynomial Matrix Eigenvalue Decomposition". W: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), s. 3255–3266. DOI: 10.1109/TASLP.2021.3120630.

[7] S. Weiss i in. "MVDR broadband beamforming using polynomial matrix techniques". W: *2015 23rd European Signal Processing Conference (EUSIPCO)*. 2015, s. 839–843. DOI: 10.1109/EUSIPCO.2015.7362501.

[8] Emanuël Habets. *Room Impulse Response Generator*. Wrz. 2010. URL: https://www.researchgate.net/publication/259991276_Room_Impulse_Response_Generator.

[9] Jont Allen i David Berkley. "Image method for efficiently simulating small-room acoustics". W: *The Journal of the Acoustical Society of America* 65 (kw. 1979), s. 943–950. DOI: 10.1121/1.382599.

[10] John G. McWhirter i in. "An EVD Algorithm for Para-Hermitian Polynomial Matrices". W: *IEEE Transactions on Signal Processing* 55.5 (2007), s. 2158–2169. DOI: 10.1109/TSP.2007.893222.

[11] Tomasz P. Zieliński. *Cyfrowe przetwarzanie sygnałów od teorii do zastosowań*. 2005.

[12] Soydan Redif, Stephan Weiss i John G. McWhirter. "Sequential Matrix Diagonalization Algorithms for Polynomial EVD of Parahermitian Matrices". W: *IEEE Transactions on Signal Processing* 63.1 (2015), s. 81–89. DOI: 10.1109/TSP.2014.2367460.

[13] Soydan Redif, Stephan Weiss i John G. McWhirter. "An approximate polynomial matrix eigenvalue decomposition algorithm for para-Hermitian matrices". W: *2011 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. 2011, s. 421–425. DOI: 10.1109/ISSPIT.2011.6151599.

[14] *Sound quality assessment material recordings for subjective tests*. URL: https://tech.ebu.ch/publications/sqamcd.

[15] *Eigen - strona główna biblioteki*. URL: https://eigen.tuxfamily.org/index.php?title=Main_Page.

[16] *Raspberry Pi Zero W*. URL: https://www.raspberrypi.com/products/raspberry-pi-zero-w/.

[17] Yi Hu i Philipos C. Loizou. "Evaluation of Objective Quality Measures for Speech Enhancement". W: *IEEE Transactions on Audio, Speech, and Language Processing* 16.1 (2008), s. 229–238. DOI: 10.1109/TASL.2007.911054.

[18] Cees H. Taal i in. "An Algorithm for Intelligibility Prediction of Time–Frequency Weighted Noisy Speech". W: *IEEE Transactions on Audio, Speech, and Language Processing* 19.7 (2011), s. 2125–2136. DOI: 10.1109/TASL.2011.2114881.

[19] A.W. Rix i in. "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs". W: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*. T. 2. Maj 2001, 749–752 vol.2. DOI: 10.1109/ICASSP.2001.941023.

[20] S. Wang, A. Sekey i A. Gersho. "An objective measure for predicting subjective quality of speech coders". W: *IEEE Journal on Selected Areas in Communications* 10.5 (1992), s. 819–829. DOI: 10.1109/49.138987.

[21] I.A. McCowan. *Robust Speech Recognition Using Microphone Arrays*. Queensland University of Technology, Brisbane, 2001.