Projekt 1 - Usuwanie zakłóceń impulsowych z nagrań muzycznych

Michał Blicharz 184430, Maciek Dziewit 185755

Spis treści

1 Wstęp

Celem projektu była implementacja rozwiązania umożliwiającego usuwanie zakłóceń impulsowych z nagrań muzycznych. W tym celu zastosowano model autoregresyjny (AR) wykorzystywany do reprezentacji lokalnej dynamiki sygnału. Do wyznaczania parametrów modelu wykorzystany został algorytm ważonych najmniejszych kwadratów (EW-LS) Do detekcji zakłóceń wykorzystano wartość lokalnego odchylenia standardowego. Za zakłócenie uznawano próbki dla których błąd estymacji modelu autoregresyjnego był większy od trzykrotnej wartości odchylenia standardowego. Próbkę taką zastępowano interpolacją liniową sąsiednich próbek.

1.1 Przygotowanie sygnału wejściowego

W celu uodpornienia rozwiązania na występowanie ciszy w sygnale wejściowym (próbki o wartości zerowej) do sygnału wejściowego dodawany jest szum biały o bardzo małej wariancji. Zabieg ten zapobiega wybuchom algorytmu identyfikacji parametrów modelu.

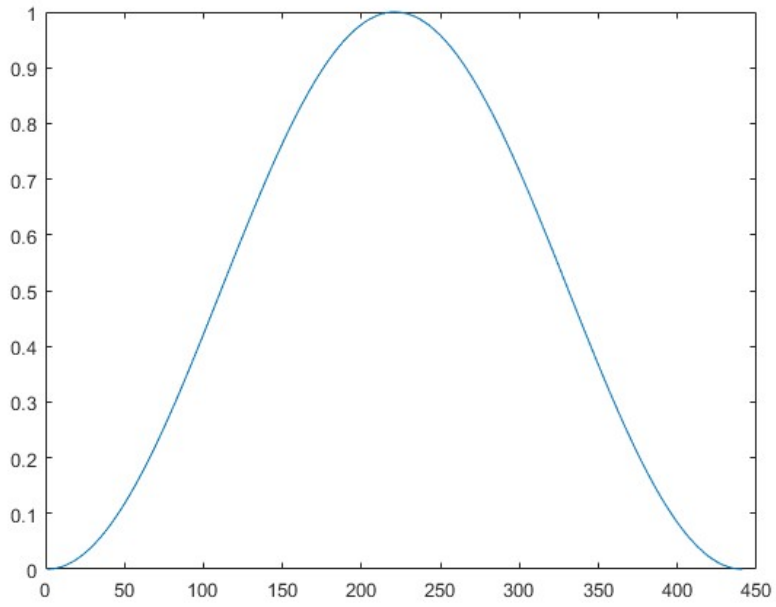
1.2 Model autoregresyjny

W celu estymacji lokalnej dynamiki wykorzystano model autoregresyjny 4 rzędu. Wzór modelu autoregresyjnego rzędu r opisuje następujące równanie: 1.

$$AR(r): \qquad y(t)\sum_{i=1}^{r} a_i y(t-i) + n(t) \tag{1}$$

Gdzie:

- r - rząd regresji
- a_i - współczynnik autoregresji dla próbki opóźnionej o i względem aktualnego czasu
- $n(t)$ - szum tworzący (o zerowej wartości średniej i znanej wariancji)



Rysunek 1: Przebieg okna zastosowanego w projekcie

1.3 Algorytm ważonych najmniejszych kwadratów (EW-LS)

W celu aktualizacji parametrów modelu AR wykorzystano algorytm ważony najmniejszych kwadratów. Równania opisujące ten algorytm zaprezentowano poniżej

$$\epsilon(t) = y(t) - \phi^T(t)\widehat{\theta}(t-1)$$

$$k(t) = \frac{\tilde{P}(t-1)\phi(t)}{\lambda + \phi^T(t)\tilde{P}(t-1)\phi(t)}$$

$$\widehat{\theta}(t) = \widehat{\theta}(t-1) + k(t)\epsilon(t)$$

$$P(t) = \frac{1}{\lambda}[P(t-1) - \frac{P(t-1)\phi(t)\phi^T(t)P(t-1)}{\lambda + \phi^T(t)p(t-1)\phi(t)}]$$

1.4 Detektor zakłóceń impulsowych

W celu detekcji zakłóceń wykorzystano detektor zakłóceń impulsowych oparty na lokalnym odchyleniu standardowym. Odchylenie standardowe wyliczane jest dla N poprzednich próbek na podstawie wartości ϵ ozyskanej w algorytmie EW-LS. Jeżeli błąd dla aktualnie próbki będzie większy od 3 krotności lokalnego

odchylenia standardowego, to dana próbka oznaczana jest jako zakłócenie. Próbka taka nie jest uwzględniana w obliczeniach algorytmu EE-LS, a jej wartość zostaje estymowana na podstawie sąsiednich próbek.

2 Program

Do rozwiązania problemu postawionego w projekcie posłużył program napisany w MatLabie. Zdecydowano się na wybór tego narzędzia zamiast pythona ze względu na szybkość jego działania. Kompilacja kodu w MatLabie była znacząco szybsza niż w Pythonie co przełożyło się na efektywniejszą pracę.

Główny kod składał się z kilku kluczowych elementów. Trzema najbardziej istotnymi były: Inicjalizacja parametrów EW-LS oraz modelu AR, główna pętla programu a także wyświetlanie wyników liczbowych i wykresów.

2.1 Inicjalizacja parametrów EWLS i modelu AR

W tej sekcji inicjalizowane są wszystkie kluczowe parametry potrzebne do działania algorytmu.

Parametry modelu AR

- **Rząd modelu AR:** $r = 4$ – określa liczbę wcześniejszych próbek wykorzystywanych do predykcji.
- **Czynnik zapominania:** $\lambda = 0.95$ – kontroluje wpływ starszych danych na estymację współczynników. Im mniejsza tym szybsze zapominanie.
- **Macierz kowariancji:** $P = 10000 \cdot I_r$ – początkowo ustawiona na dużą wartość, co oznacza dużą niepewność współczynników AR.
- **Wektor współczynników AR:** $\theta = [0, 0, 0, 0]^T$ – początkowo ustawiony na wartości zerowe.

Parametry detekcji błędów

- **Długość okna do obliczania lokalnego odchylenia standardowego:** std_window = 80.
- **Próg detekcji zakłóceń:** threshold_factor = 3 – próbka uznawana jest za zakłóconą, jeśli jej błąd predykcji przekracza trzykrotność lokalnego odchylenia standardowego.
- **Maksymalna liczba kolejnych błędnych próbek:** max_consecutive_errors = 4 – określa, ile błędnych próbek występujących bezpośrednio po sobie może zostać zastąpionych. Z danych do zadania wiadomo, że maksymalnie 4 próbki po sobie mogą być niepoprawne. Zatem 5 próbka po 4 błędnych musi być poprawna, dodano ograniczenie aby program nie uznał poprawnej próbki za błędną.
- **Liczba próbek na ustabilizowanie modelu:** startup_samples = 100 – przez pierwsze 100 próbek detekcja zakłóceń nie jest przeprowadzana. Przez pierwsze 100 próbek detekcja zakłóceń nie jest przeprowadzana,

ponieważ w początkowej fazie działania algorytmu estymator współczynników AR wymaga czasu na stabilizację. Podczas tej fazy początkowej wartości współczynników θ mogą być niestabilne, a błędy predykcji wysokie, co mogłoby prowadzić do fałszywego wykrywania zakłóceń. Pominięcie detekcji w pierwszych startup_samples próbkach pozwala algorytmowi na lepsze dostosowanie się do struktury sygnału i uniknięcie niepotrzebnej korekcji danych, która mogłaby pogorszyć jakość oczyszczonego sygnału.

Zmienne pomocnicze

Kod inicjalizuje również tablice do przechowywania wyników:

- θ_s – macierz zapisująca ewolucję współczynników modelu AR.
- $e_{\text{residuals}}$ – tablica przechowująca błędy predykcji dla każdej próbki.
- clean_signal – kopia oryginalnego sygnału, która będzie modyfikowana w celu usunięcia zakłóceń.

Opisane powyżej zmienne są widoczne na rysunku 2 przedstawiający fragment kodu.

```
%% Główne parametry
% Inicjalizacja parametrów EWLS i modelu AR
r = 4;  % Rząd modelu AR
lambda = 0.95; % Czynnik zapominania
P = 10000 * eye(r); % Macierz kowariancji
theta = zeros(r, 1); % Współczynniki AR

% Parametry do wygładzania współczynników
max_theta_change = 0.05; % Maksymalna zmiana współczynnika między iteracjami

% Inicjalizacja zmiennych przechowujących wartości estymowane i błędy
theta_s = zeros(r, N);
e_residuals = zeros(1, N);

% Parametry detekcji błędów
std_window = 80; % Długość okna do obliczania lokalnego odchylenia std
threshold_factor = 3; % Maksymalne odchylenie próbki
max_consecutive_errors = 4; % Maksymalna liczba błędnych próbek z rzędu
startup_samples = 100; % Liczba próbek na ustabilizowanie estymacji

% EWLS i detekcja błędów
clean_signal = data; % Kopia sygnału do oczyszczenia
consecutive_error_count = 0; % Licznik kolejnych błędów
```

Rysunek 2: Fragment kodu z inicjalizacją parametrów

W tej części kodu zainicjalizowano kluczowe parametry algorytmu, w tym współczynniki modelu autoregresyjnego (AR), macierz kowariancji oraz parametry detekcji błędów, takie jak próg wykrywania zakłóceń i długość okna odchylenia standardowego

2.2 Główna pętla programu

Estymacja współczynników AR

W tej sekcji kodu realizowana jest estymacja współczynników modelu autoregresyjnego (AR) przy użyciu algorytmu rozszerzonych najmniejszych kwadratów (EWLS):

1. **Tworzenie wektora regresji** – Pobierane są r poprzednich próbek sygnału, które tworzą wektor regresji ϕ, wykorzystywany do przewidywania wartości bieżącej próbki.

2. **Obliczenie błędu predykcji** – Różnica między rzeczywistą wartością sygnału a prognozą modelu AR zapisywana jest w tablicy $e_{\text{residuals}}$.

3. **Aktualizacja współczynników AR** – Obliczany jest wektor zysków k, który dostosowuje współczynniki θ, a następnie wartości te są modyfikowane zgodnie z metodą EWLS:

$$k = \frac{P\phi}{\lambda + \phi^T P \phi}, \tag{2}$$

$$\theta_{\text{new}} = \theta + ke. \tag{3}$$

4. **Ograniczenie nagłych zmian współczynników** – Wprowadzone jest zabezpieczenie przed gwałtownymi zmianami parametrów, dzięki czemu ich modyfikacja nie przekracza zadanego progu max_theta_change.

5. **Aktualizacja macierzy kowariancji** P – Macierz kowariancji jest modyfikowana zgodnie z równaniem:

$$P = \frac{P - (k\phi^T P)}{\lambda}, \tag{4}$$

   co pozwala modelowi adaptować się do zmienności sygnału.

6. **Zapis współczynników do analizy** – Wartości współczynników θ są przechowywane w tablicy θ_s w celu późniejszej wizualizacji i analizy.

Opisane powyżej zmienne są widoczne na rysunku 3 przedstawiający fragment kodu.

```
%% Główna pętla i zapis
for t = r+1:N
    % Tworzenie wektora regresji (przeszłe wartości sygnału)
    phi = data(t-1:-1:t-r);

    % Błąd predykcji
    e = data(t) - ((phi') * theta);
    e_residuals(t) = e;

    % Aktualizacja współczynników AR za pomocą EWLS
    k = (P * phi) / (lambda + ((phi') * P * phi));
    theta_new = theta + k * e;

    % Ograniczenie nagłych zmian współczynników AR
    theta = theta + max(min(theta_new - theta, max_theta_change), -max_theta_change);

    % Aktualizacja macierzy kowariancji
    P = (P - (k * (phi') * P)) / lambda;

    % Zapisujemy współczynniki do analizy
    theta_s(:, t) = theta;
```

Rysunek 3: Fragment kodu z główną pętlą programu 1/2

Detekcja i korekcja zakłóceń

1. **Warunek rozpoczęcia detekcji** – Analiza błędów predykcji jest przeprowadzana dopiero po przekroczeniu liczby próbek startup_samples, aby umożliwić stabilizację modelu.

2. **Obliczenie lokalnego odchylenia standardowego** – Jeśli liczba dostępnych próbek przekracza długość okna std_window, obliczane jest lokalne odchylenie standardowe błędu predykcji:

$$\sigma_{\text{local}} = \text{std}(e_{\text{residuals}}(t - \text{std\_window} : t - 1)). \tag{5}$$

3. **Wykrywanie zakłóceń** – Jeśli wartość błędu predykcji przekracza określony próg $\text{threshold\_factor} \cdot \sigma_{\text{local}}$, licznik błędnych próbek zostaje zwiększony.

4. **Korekcja zakłóconych próbek**:

   - Jeśli liczba kolejnych zakłóceń nie przekracza limitu max_consecutive_errors, wartość zakłóconej próbki zostaje zastąpiona poprzednią wartością sygnału:

$$\text{clean\_signal}(t) = \text{clean\_signal}(t - 1). \tag{6}$$

   - Jeśli próbka jest poprawna, licznik błędów jest resetowany.
   - Dodatkowo, jeśli poprzednia próbka była błędna, stosowana jest interpolacja między sąsiednimi wartościami, co pomaga w płynnym przywróceniu sygnału:

$$\text{clean\_signal}(t) = 0.5 \cdot (\text{clean\_signal}(t - 1) + \text{clean\_signal}(t + 1)). \tag{7}$$

Opisana powyżej logika została zaimplentowana do programu w postaci kodu widocznego na rysunku 4.

```
    % **Detekcja zakłóceń dopiero po 100 próbkach**
    if t > startup_samples
        if t > std_window
            local_std = std(e_residuals(t-std_window:t-1)); % Lokalne odchylenie std błędu

            if abs(e) > threshold_factor * local_std
                consecutive_error_count = consecutive_error_count + 1;

                % Jeśli liczba błędnych próbek przekracza limit, pomijamy zastępowanie
                if consecutive_error_count <= max_consecutive_errors
                    clean_signal(t) = clean_signal(t-1); % Zastępujemy poprzednią wartością
                end
            else
                % Reset licznika jeśli próbka jest poprawna
                consecutive_error_count = 0;

                % Jeśli poprzednia próbka była błędna, zastosuj interpolację
                if t > 1 && e_residuals(t-1) > threshold_factor * local_std
                    clean_signal(t) = 0.5 * (clean_signal(t-1) + clean_signal(t+1)); % Interpolacja
                end
            end
        end
    end
end
```

Rysunek 4: Fragment kodu z główną pętlą programu 2/2

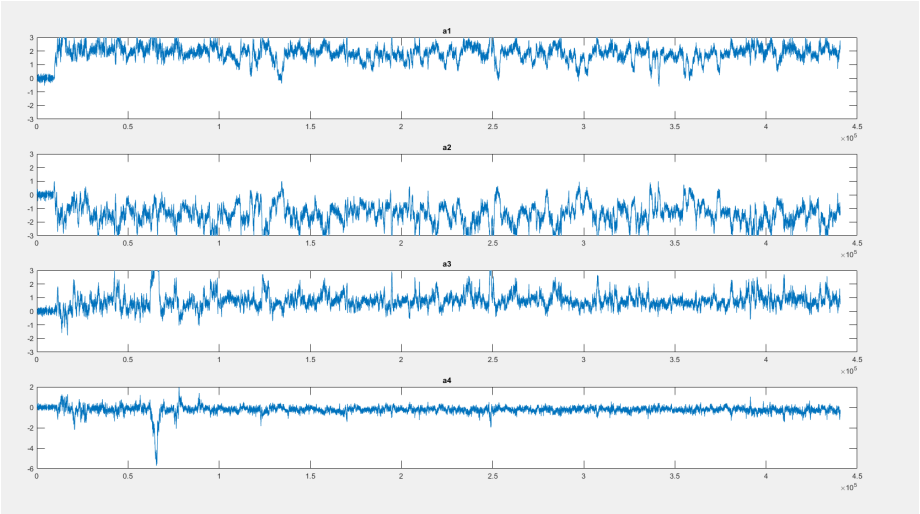
Główna pętla programu realizuje dwie kluczowe funkcje: estymację współczynników modelu autoregresyjnego (AR) oraz detekcję i korekcję zakłóceń. W pierwszej części pętli, przy użyciu algorytmu rozszerzonych najmniejszych kwadratów (EWLS), aktualizowane są współczynniki AR, co pozwala na bieżąco modelować strukturę sygnału i minimalizować błąd predykcji. Następnie, po początkowej fazie stabilizacji trwającej przez 100 próbek, przeprowadzana jest analiza błędów predykcji w celu wykrywania anomalii.

Zakłócone próbki są identyfikowane na podstawie lokalnego odchylenia standardowego, a ich wartości korygowane poprzez zastąpienie poprzednią wartością lub interpolację, co pozwala na efektywne oczyszczenie sygnału.

2.3 Wyniki i wykresy

Ostatnią częścią programu było przedstawienie wyników na wykresach.

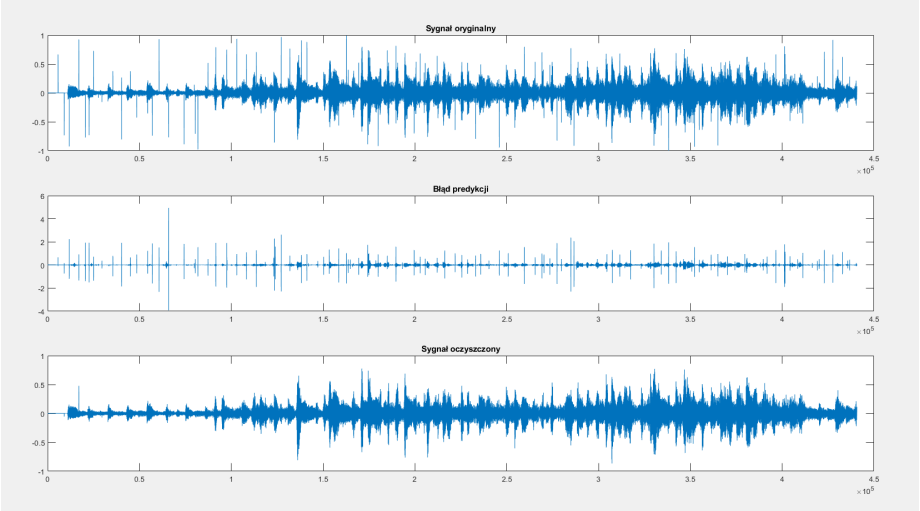
Na rysunku 5 przedstawiono wykresy parametrów modelu AR. Na wykresach widać częste lecz niewielkie oscylacje wartości poszczególnych parametrów. Oznacza to że model zmienia swoje parametry w sposób dynamiczny adaptując się do napotkanych warunków.



Rysunek 5: Wykres parametrów modelu AR

Na rysunku 6 przedstawiono wykresy: sygnału oryginalnego spróbkowanego z odpowiednią częstotliwością, błędów predykcji oraz sygnału oczyszczonego z szumów.

Porównując sygnał oryginalny z oczyszczonym wyraźnie widać, że piki symbolizujące szumy i zakłócenia zostały całkowicie usunięte. Można więc jasno stwierdzić, że program wykonuje swoje zadanie poprawnie.



Rysunek 6: Wykres parametrów modelu AR

3 Napotkane problemy i rozwiązania

3.1 Zerowe i nieskończone wartości parametrów modelu AR

Pierwszym napotkanym problemem były niepoprawne wartości parametrów modelu AR które pokazano na rysunku 7. Parametry modelu a_1, a_2 oraz a_3 były bliskie zeru natomiast parametr a_4 na początku uciekał do nieskończoności a potem również się zerował. Nie zgadzało się to z oczekiwanymi wynikami zatem w pierwszej kolejności sprawdzono czy w programie znajdują się jakieś błędy, które mogą być przyczyną takiego zachowania parametrów AR. Nie znaleziono żadnych błędów a dodatkowo program przetestowano na innych plikach z zaszumioną muzyką. Z testów wynikło, że program działał poprawnie na innych plikach, zatem problem nie leżał w kodzie a w samym pliku. Okazało się, że w pliku występuje dużo próbek zerowych, ponadto próbki te występują kolejno po sobie.

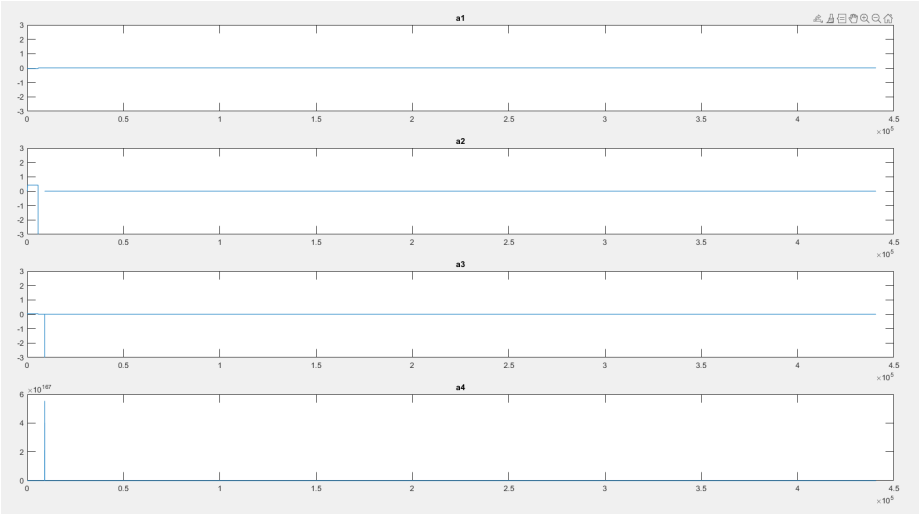
Kiedy próby przewidywania na podstawie zerowych próbek są błędne, błąd predykcji może rosnąć, ponieważ w algorytmie EWLS mamy proces aktualizacji współczynników na podstawie błędów predykcji. W przypadku zerowych próbek model stara się dostosować współczynniki do błędnych predykcji, co może prowadzić do ogromnych zmian w wartościach współczynników AR. Na rysunku 8 widać, że błąd predykcji w pierwszych chwilach osiąga ogromne wartości rzędu 10^{170}.

Gdy próbki sygnału są zerowe, współczynniki modelu AR takie jak a_1, a_2, a_3 mogą zostać obliczone na 0, ponieważ w zasadzie model "nie ma" żadnej informacji, która pozwoliłaby na dopasowanie zależności między próbkami. W wyniku tego, wartości tych współczynników pozostaną na to, ponieważ próby przewidywania w oparciu o zerowe dane nie pozwalają na uzyskanie znaczącej estymacji.

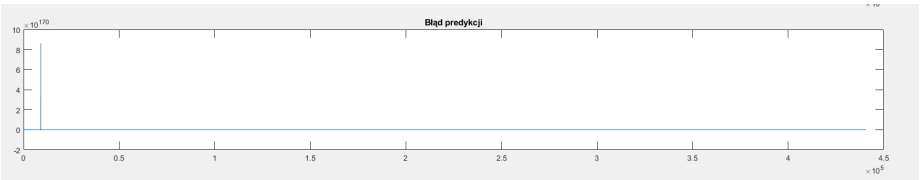
Rozwiązaniem tego problemu było lekkie zaszumienie tak aby wyeliminować próbki zerowe. Dokonano tego w sposób następujący:

```
1 data = data + (10^-6) * randn(size(data));
```

Dzięki temu udało się wyeliminować próbki zerowe, przetestowano program i zgodnie z oczekiwaniami otrzymano wyniki prezentowane we wcześniejszym rozdziale. Jedyną wadą tego rozwiązania jest to, że przez cały czas trwania utworu słychać niewielki szum, jednakże główne zakłócenia udało się usunąć.



Rysunek 7: Wykres parametrów modelu AR

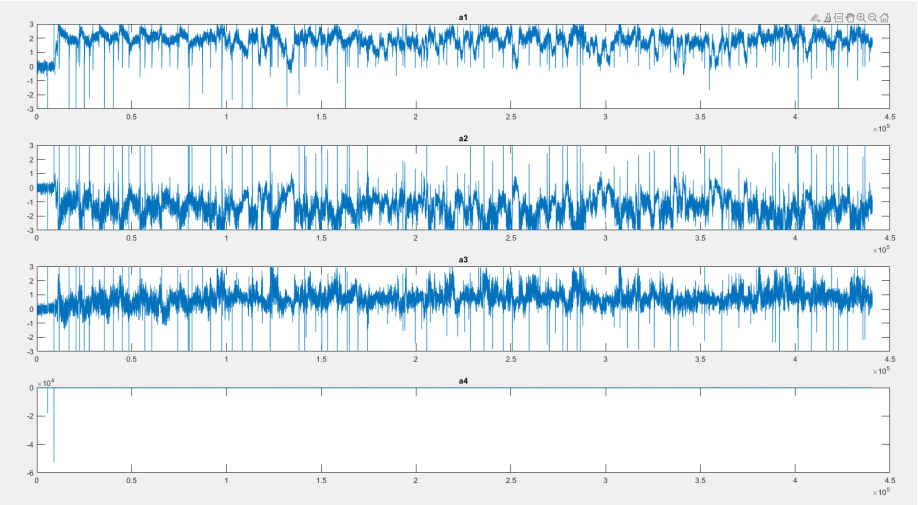


Rysunek 8: Błąd predykcji uciekający do nieskończoności

3.2 Piki parametrów AR

Drugim mniej znaczącym i łatwiejszym do naprawienia problemem było wystąpienie pików przy parametrach modelu AR. (Widoczne na rysunku 9) Piki zaobserwowano po naprawie pierwszego problemu związanego z zerowymi wartościami próbek. Parametry modelu AR zaczęły działać poprawnie, jednakże pojawiły się nieoczekiwane piki. W celu ich eliminacji nałożono limit na maksymalną zmianę wartości theta w każdym kroku co rozwiązało problem.

```
1 max_theta_change = 0.05;
```



Rysunek 9: Piki parametrów modelu AR

4 Wnioski

Środowisko MATLAB okazało się trafnym wyborem ze względu na szybkość działania i łatwość implementacji algorytmów numerycznych. W porównaniu z Pythonem, MATLAB zapewnił szybszą kompilację kodu i efektywniejszą pracę nad projektem.

Efektywność modelu autoregresyjnego: Model AR okazał się skutecznym narzędziem do estymacji lokalnej dynamiki sygnału muzycznego. Dzięki zastosowaniu algorytmu EW-LS, możliwe było dynamiczne dostosowywanie współczynników modelu do zmieniających się warunków sygnału, co pozwoliło na skuteczną detekcję i usuwanie zakłóceń impulsowych.

Zastosowanie interpolacji liniowej do korekcji zakłóconych próbek pozwoliło na płynne przywrócenie sygnału. W przypadkach, gdy liczba kolejnych zakłóconych próbek nie przekraczała ustalonego limitu, zastąpienie ich wartościami sąsiednimi okazało się wystarczające do uzyskania zadowalającej jakości sygnału.

Jednym z głównych problemów napotkanych podczas realizacji projektu było występowanie dużej liczby zerowych próbek w sygnale wejściowym. Powodowało to niestabilność współczynników modelu AR, a w skrajnych przypadkach prowadziło do ich ucieczki do nieskończoności. Rozwiązaniem tego problemu było dodanie niewielkiego szumu białego do sygnału, co pozwoliło na uniknięcie zerowych próbek i stabilizację działania algorytmu. Ograniczenie nagłych zmian współczynników: Wprowadzenie limitu na maksymalną zmianę wartości współczynników modelu AR w każdym kroku iteracyjnym pozwoliło na uniknięcie niepożądanych pików w ich wartościach. Dzięki temu model działał bardziej stabilnie, a jakość oczyszczonego sygnału uległa poprawie.