POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

SP&L Homework 2 Adaptive interference mitigation on audio signals

Tesi di Laurea Magistrale in
Telecommunication Engineering

Author: **Hong CHEN**

Student ID: 11024836
Advisor: Prof. Name Surname
Co-advisors: Name Surname, Name Surname
Academic Year: 2025-07

Contents

1 | Adaptive interference mitigation on audio signals

1.1. Project Overview

The title describes a stereo(two-channel) audio signal $s(t) = [s_1, s_2]^T$. This signal is contaminated by noise $w(t)$, which is not directly independent but instead results from an external noise source $n(t)$. The noise $w(t)$ is generated by applying a 2 x 2 filter $h(t)$ to $n(t)$ through convolution. The mathematical model for the sampled version of the signal is:

$$s_{in}[u] = s[u] + w[u] \tag{1.1}$$

$$s_{in}[u] = [\ s_1[u] \quad s_2[u]\ ] + [\ w_1[u] \quad w_2[u]\ ] \tag{1.2}$$

Where:

$$w[u] = \mathbf{h} * n[u] \tag{1.3}$$

The provided data files represent the noise-interfered signals, denoted as Sin_x, where $x = a, b, c$, indicating increasing levels of noise difficulty. Additionally, a reference file, Sn_{ref_x}, is provided, where the interference signal s_{ref} is a noise-related reference signal measured from the environment ($e.g.$ an external noise signal measured through another microphone) indicating the source of the interference. It is generated by $n[u]$ through the unknown transfer function $G(z)$:

$$s_{ref}[u] = \mathbf{g} * n[u] \tag{1.4}$$

The goal of the numerical exercise is to process stereo audio signals that are contaminated with noise and to extract the clean or noise-mitigated signals. The performance of the noise mitigation is evaluated by listening to the processed signals through a stereo headset using the `Matlab` command `sound(`S_{est}`,` f_s`)`, where S_{est} is the estimated clean stereo signal and f_s is the sampling frequency.

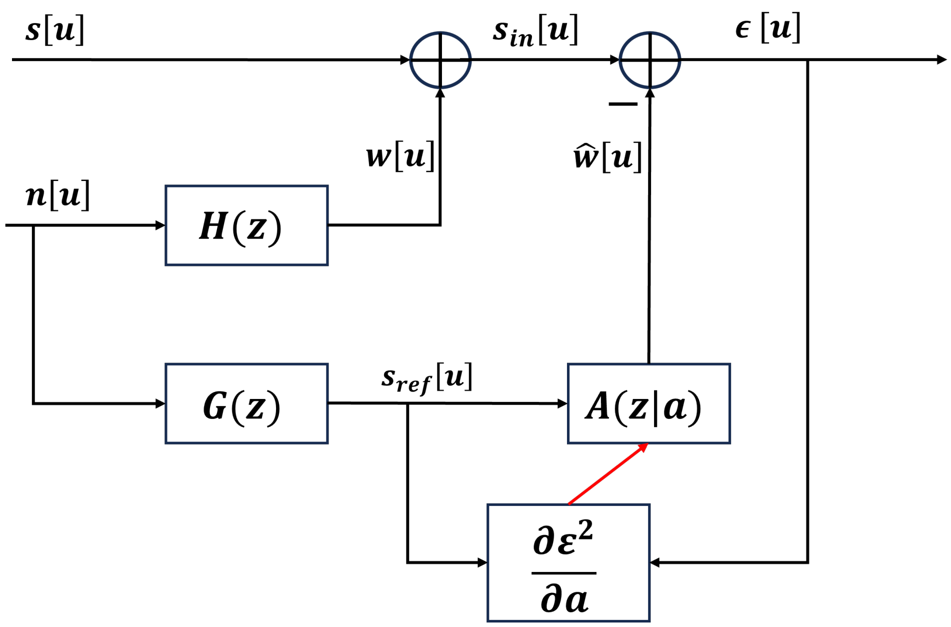

Figure 1.1: System Diagram

1.2. Filter Design

In order to generate an estimate $w[u]$ of the noise $\hat{w}[u]$, we design an adaptive filter $A(z|\mathbf{a})$ (FIR filter), the weight coefficients of which are represented by a vector $\mathbf{a}$, and the output is:

$$\hat{w}[u] = A(z|\mathbf{a}) * s_{ref}[u] \tag{1.5}$$

Based on S_{in} and the reference noise signal S_{ref} , and implemented using an adaptive filtering technique (LMS) to optimize the weighting coefficients of the filter $A(z|\mathbf{a})$ that is capable of maximizing noise suppression.
The final output error signal is:

$$\epsilon[u|\mathbf{a}] = s_{est}[u] = s_{in}[u] - \hat{w}[u] \tag{1.6}$$

Signal Reconstruction:
Applying the designed filter $A(z|\mathbf{a})$, the noise w affecting the original signal $s[u]$ is estimated, and the estimated signal $s_{est}[u]$ is obtained by subtracting the estimated noise $w[u]$ from the signal $s_{in}[u]$.

$$\mathbf{s}_{\text{est}}[u] = \mathbf{s}_{\text{in}}[u] - \mathbf{A} * \mathbf{s}_{\text{ref}}[u] \tag{1.7}$$

1.3. Optimization Method(LMS)

Adjust the filter parameter $\mathbf{a}$ to minimize the noise in the error signal $\epsilon[u|\mathbf{a}]$ while preserving the target signal $s[u]$ as much as possible. The LMS method achieves this by recursively updating the filter weights to gradually approximate the minimum mean square error (MSE).
Definition of cost function(Mean Square Error (MSE)):

$$\begin{aligned} J(\mathbf{a}) &= \mathbb{E}[(\epsilon[u|\mathbf{a}])^2] \\ &= \mathbb{E}[(s_{in}[u] - \hat{w}[u])^2] \\ &= \mathbb{E}[(s_{in}[u] - s_{ref}[u] \cdot \mathbf{a})^2] \\ &= \mathbb{E}[s_{in}^2[u] - 2 \cdot s_{in}[u] \cdot s_{ref}[u] \cdot \mathbf{a} + s_{ref}^2[u] \cdot \mathbf{a}^T \mathbf{a}] \end{aligned} \tag{1.8}$$

The gradient descent method is used to update the filter coefficients as follows:

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{a}} &= -2 \cdot \mathbb{E}[s_{in} \cdot s_{ref} - s_{ref}^2 \cdot \mathbf{a}] \\ &= -2 \cdot \mathbb{E}[s_{ref}(s_{in} - s_{ref} \cdot \mathbf{a})] \\ &= -2 \cdot \mathbb{E}[\epsilon \cdot s_{\text{ref}}] \end{aligned} \tag{1.9}$$

In the LMS algorithm, the expectation is not computed directly, but is approximated using the current observed values.

$$\frac{\partial J}{\partial \mathbf{a}} \approx -2 \cdot \epsilon[k] \cdot s_{ref}[k] \tag{1.10}$$

The weight update formula for the filter $\mathbf{A}$ is as follows:

$$\begin{aligned} \mathbf{a}[n+1] &= \mathbf{a}[n] - \frac{\mu_0}{1 + \alpha \cdot (\epsilon[n])^2} \cdot \frac{\partial J}{\partial \mathbf{a}} \\ &= \mathbf{a}[n] + \frac{2\mu_0}{1 + \alpha \cdot (\epsilon[n])^2} \cdot \epsilon[n] \cdot s_{ref}[n] \\ &= \mathbf{a}[n] + 2\mu[n] \cdot \epsilon[n] \cdot s_{ref}[n] \end{aligned} \tag{1.11}$$

1.3.1. Analysis of Step Size (μ) and Filter Order (L)

The filter order L determines how well the adaptive filter can model the impulse response of the interference path from the external noise source $n[u]$ to the recorded signal $s_{in}[u]$. A too-small L may not fully suppress the interference, while a too-large L increases computational cost and slows down convergence.

The step size μ controls the adaptation speed of the LMS algorithm. If μ is too small, convergence is slow. If μ is too large, the filter may oscillate or diverge.

$$0 < \mu < \frac{1}{\lambda_{max}} \tag{1.12}$$

where λ_{max} is the largest eigenvalue of the autocorrelation matrix of the input signal s_{ref}. In practice, an empirical bound using filter order L and signal power σ_x^2 is:

$$\mu < \frac{1}{3 \cdot L \cdot \sigma_x^2} \tag{1.13}$$

In this implementation, a dynamic step size is used:

$$\mu_k = \frac{\mu_0}{1 + \alpha \cdot \epsilon^2[k]} \tag{1.14}$$

This helps reduce the update strength when the error $\epsilon[k]$ becomes large, preventing sudden weight changes. Substituting into the LMS update equation gives:

$$\mathbf{a}[n+1] = \mathbf{a}[n] + \frac{2\mu_0}{1 + \alpha \cdot \epsilon^2[n]} \cdot \epsilon[n] \cdot \mathbf{s}_{ref}[n] \tag{1.15}$$

To select an appropriate step size μ, several candidate values were tested using the same input data. The performance was evaluated in terms of SNR and signal clarity.

1.4. Optimization Method (NLMS)

The NLMS algorithm is an improved version of the standard LMS algorithm. Its purpose is to iteratively update filter coefficients to minimize noise or interference in real-time.
In order to adaptively estimate the filter coefficients $\mathbf{a}$ and minimize the noise in the output signal $\epsilon[u|\mathbf{a}]$, we employ the Normalized Least Mean Squares (NLMS) algorithm, which improves the classical LMS by normalizing the update step with the energy of the input signal.

1.4.1. Cost Function and Gradient

We define the cost function as the instantaneous squared error:

$$J(\mathbf{a}) = \frac{1}{2} \cdot \epsilon^2[n] = \frac{1}{2} \cdot (s_{\text{in}}[n] - \hat{w}[n])^2 = \frac{1}{2} \cdot (s_{\text{in}}[n] - \mathbf{a}^T \cdot \mathbf{s}_{\text{ref}}[n])^2 \tag{1.16}$$

The gradient with respect to the filter coefficients is:

$$\frac{\partial J}{\partial \mathbf{a}} = -\epsilon[n] \cdot \mathbf{s}_{\text{ref}}[n] \tag{1.17}$$

1.4.2. NLMS Update Rule

Unlike LMS, NLMS normalizes the step size by the squared norm of the reference signal vector:

$$\mathbf{a}[n+1] = \mathbf{a}[n] + \frac{\mu}{\|\mathbf{s}_{\text{ref}}[n]\|^2 + \delta} \cdot \epsilon[n] \cdot \mathbf{s}_{\text{ref}}[n] \tag{1.18}$$

where:

- μ is the normalized step size, typically $0 < \mu < 2$,
- δ is a small constant to avoid division by zero,
- $\epsilon[n] = s_{\text{in}}[n] - \mathbf{a}^T[n] \cdot \mathbf{s}_{\text{ref}}[n]$ is the instantaneous error.

The larger the input signal energy $\rightarrow$ the smaller the step size $\rightarrow$ preventing overfitting.

1.5. Kalman-LMS Filtering

To improve adaptability and robustness under non-stationary or noisy conditions, we apply a Kalman-based adaptation scheme to recursively estimate the filter weights. This approach treats the filter coefficients as dynamic states, updated using a Bayesian framework.

State Equation:

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \mathbf{v}[n], \quad \mathbf{v}[n] \sim \mathcal{N}(0, \mathbf{Q}) \tag{1.19}$$

Observation Equation:

$$d[n] = \mathbf{w}^\top[n]\mathbf{x}[n] + \eta[n], \quad \eta[n] \sim \mathcal{N}(0, R) \tag{1.20}$$

Kalman Filter Updates:

$$\mathbf{K}[n] = \frac{\mathbf{P}[n]\mathbf{x}[n]}{\lambda + \mathbf{x}^\top[n]\mathbf{P}[n]\mathbf{x}[n]} \tag{1.21}$$

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \mathbf{K}[n] \cdot e[n] \tag{1.22}$$

$$\mathbf{P}[n+1] = \frac{1}{\lambda}\left(\mathbf{I} - \mathbf{K}[n]\mathbf{x}^\top[n]\right)\mathbf{P}[n] \tag{1.23}$$

Where:

- $\mathbf{w}[n]$: adaptive filter weight vector,
- $\mathbf{x}[n]$: reference input vector,
- $d[n]$: observed noisy signal,
- $e[n] = d[n] - \mathbf{w}^\top[n]\mathbf{x}[n]$: a priori estimation error,
- $\mathbf{P}[n]$: error covariance matrix,
- $\mathbf{K}[n]$: Kalman gain,
- λ: forgetting factor ($0 < \lambda \leq 1$).

Initialization:

$$\mathbf{w}[0] = \mathbf{0}, \quad \mathbf{P}[0] = \delta \cdot \mathbf{I} \tag{1.24}$$

A larger δ reflects low confidence in initial weights and allows faster adaptation; a smaller δ assumes reliable initial estimates, leading to smoother updates.

Parameter Discussion and Tuning:

- **Kalman gain $\mathbf{K}[n]$** balances prediction and observation. A larger gain leads to faster updates.
- **Forgetting factor λ** controls memory: lower λ emphasizes recent data; higher λ favors stability.
- **Process noise $\mathbf{Q}$** adjusts filter agility: larger $\mathbf{Q}$ enables faster tracking of dynamic changes; smaller $\mathbf{Q}$ ensures smoother updates.
- **Observation noise R** reflects measurement trust: a large R reduces sensitivity to noisy inputs; a small R increases correction strength.

1.6. Result Analysis

1.6.1. Point a

In `point a`, the signal Sin_a is well-aligned with the reference $Sn_{ref\,a}$ (see figure 1.2). Applying NLMS followed by Kalman filtering effectively suppresses noise and enhances speech clarity (see Figure 1.3).

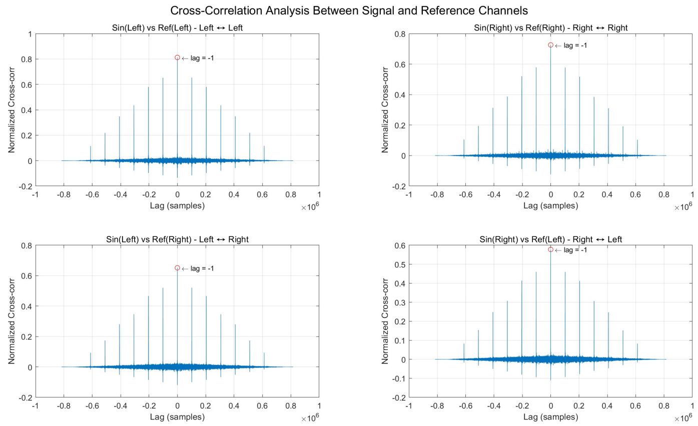

Figure 1.2: Point a: Cross Correlation Between Signal and Ref

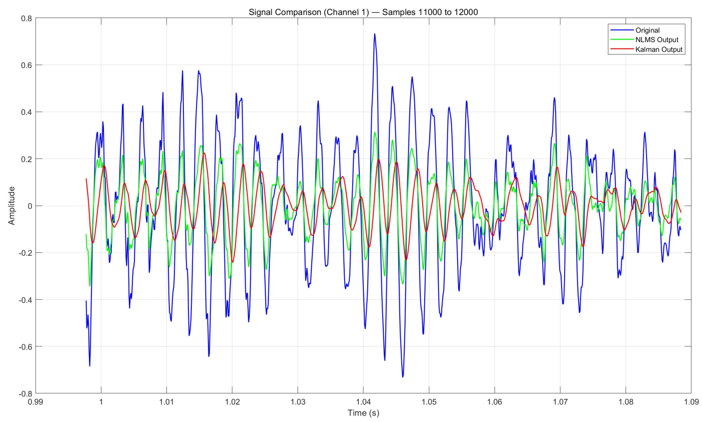

Figure 1.3: Point a: Signal Comparison

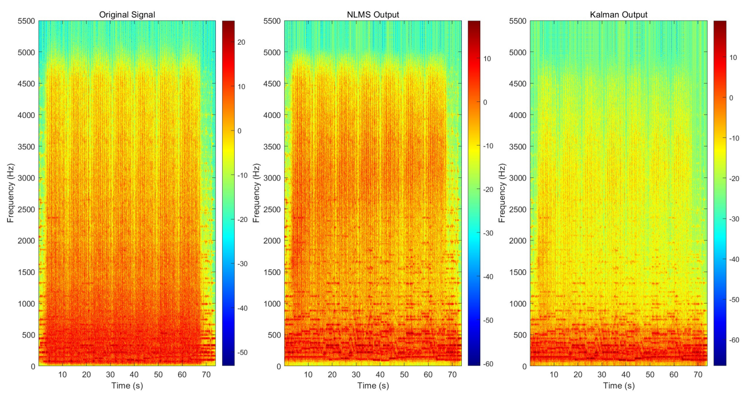

Figure 1.4: Point a: Signal spectrum

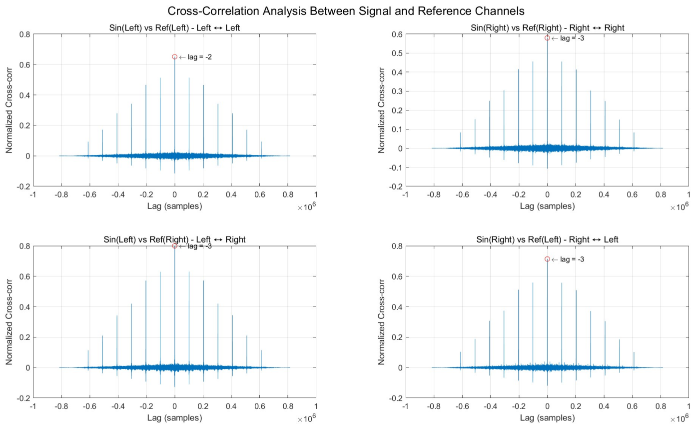

Figure 1.5: Point b: Cross Correlation Between Signal and Ref

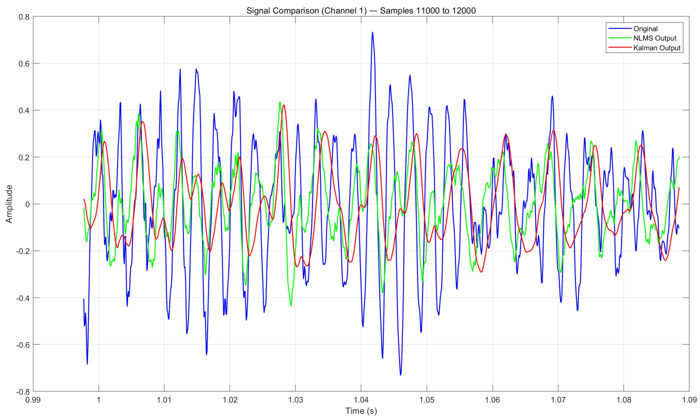

Figure 1.6: Point b: Signal Comparison

1.6.2. Point b

Problem b follows the same processing logic as `Point a`, but the stereo channels in Sin_b are swapped. We confirmed this using cross-correlation analysis between the noisy signal and the reference channels. It was observed that: $Sin_b(:,1)$ is most correlated with $Sn_ref_b(:,2)$; $Sin_b(:,2)$ is most correlated with $Sn_ref_b(:,1)$ (see figure 1.5).
And since Sin_b in `point b` shows a -3-sample delay with the reference signal (see cross-correlation analysis), the reference was time-aligned before processing.

1.6.3. Point c

Given a noisy audio signal `Sin_c(:,1)` containing both music and additive noise, and a reference noise signal `Sn_ref_c` with correlated but not identical noise, the objective is

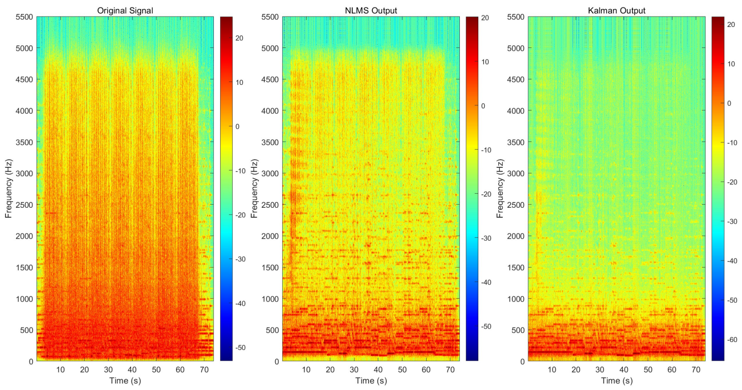

Figure 1.7: Point b: Spectrum Comparision

to suppress the noise and recover a cleaner version of the music.

We model the observed signal as:

$$s_{\text{in}}(i) = s(i) + n(i) \tag{1.25}$$

where $s(i)$ is the clean signal and $n(i)$ is the noise component. The reference input $\mathbf{x}(i)$ is assumed to be correlated with $n(i)$ but uncorrelated with $s(i)$.

We aim to learn a filter $\mathbf{W}$ such that:

$$\mathbf{W}^\top \mathbf{x}(i) \approx n(i) \tag{1.26}$$

Thus, the error to minimize becomes:

$$\min_{\mathbf{W}} \sum_i \left[s_{\text{in}}(i) - \mathbf{W}^\top \mathbf{x}(i) \right]^2 \tag{1.27}$$

Finally, the estimated clean signal is obtained by subtracting the estimated noise:

$$\hat{s}(i) = s_{\text{in}}(i) - \mathbf{W}^\top \mathbf{x}(i) \tag{1.28}$$

To improve performance, the two channels `Sn_ref_c(:,1)` and `Sn_ref_c(:,2)` are concatenated into a unified reference vector, enriching the information available to the adaptive filter.

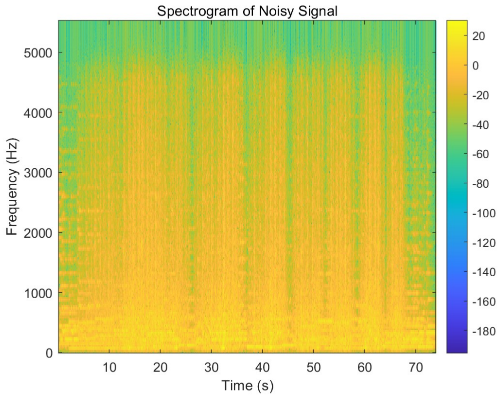

Figure 1.8: Point c: Spectrograms of noisy signal

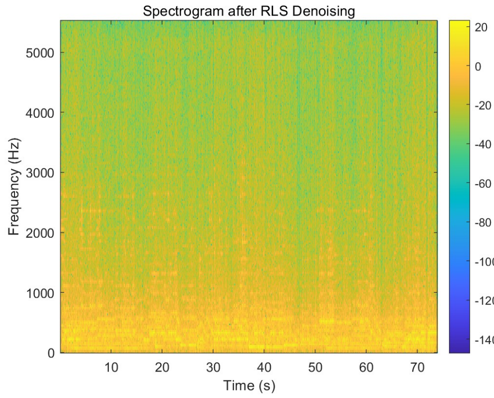

Figure 1.9: Point c: Spectrograms of clean signal

1.6.4. Point d: Reference-Based Denoising

Objective: The core goal of this task is to extract the clean audio signal from a recording corrupted by time-varying sinusoidal interference.

Interference Characteristics: The noise consists of strong sinusoids whose frequencies vary over time. In the time-frequency domain (spectrogram), these appear as bright diagonal or curved streaks.

Reference Noise Signals: An additional file `Sn_ref_d1.mat` provides two reference noise channels in the variable `sn_ref_matrix`, which are partially correlated with the interference in `Sin_d`.

Key Challenges:

- The interference is *non-stationary*, with frequency content that evolves over time.
- Effective denoising requires the joint use of **both** reference noise channels to capture the full interference structure.

After RLS Denoising

In the post-filtered spectrogram, the prominent sinusoidal trace that previously drifted over time is significantly attenuated or even entirely removed.

As a result, the underlying musical or speech structure becomes more clearly visible, with a more natural energy distribution across frequencies.

The background appears less cluttered, and the residual noise is more consistent with the expected spectral profile of clean audio.

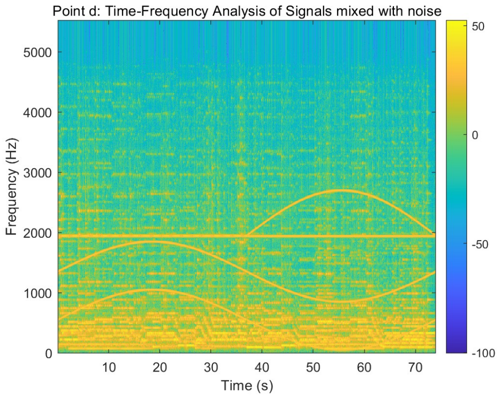

Figure 1.10: Point d: With Noise

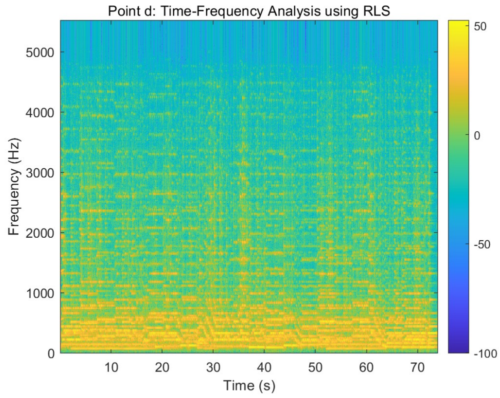

Figure 1.11: Point d: RLS De-noised

1.6.5. Point e

Objective: To remove a slowly time-varying sinusoidal interference from the noisy signal `Sin_e`. After applying RLS, the spectrogram of the output signal `s_est_e` shows that the slanted bright line corresponding to the interference is significantly attenuated or removed, indicating effective denoising.

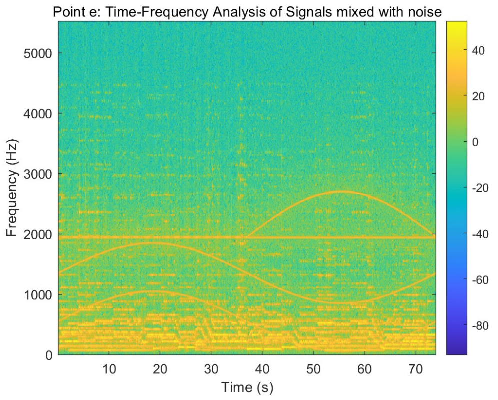

Figure 1.12: Point e: With Noise

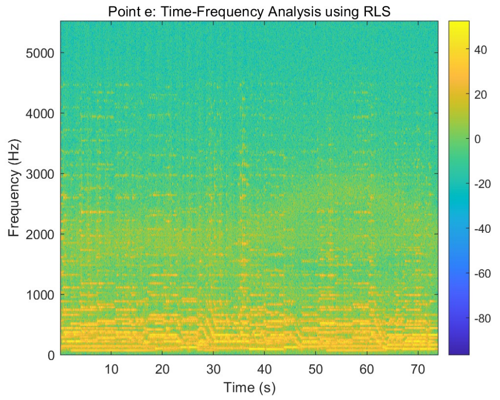

Figure 1.13: Point e: RLS De-noised

As in point d, the simulation output demonstrates effective removal of the interfering sinusoidal lines.