Adaptive Noise Cancellation For Electronic Stethoscopes

Master Thesis

Jacob El-Omar

Aalborg University
Electronics & IT

Electronics and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Adaptive Noise Cancellation For Electronic Stethoscopes

Theme:
Master Thesis

Project Period:
Spring Semester 2025

Project Group:
924

Participant(s):
Jacob El-Omar

Supervisor(s):
Jan Østergaard

Copies: 1

Page Numbers: 150

Date of Completion:
May 28, 2025

Abstract:

This thesis investigates adaptive noise cancellation for improving heart and lung sound recordings in noisy environments, with a focus on resource-limited settings like rural India. Environmental and equipment noise often degrade auscultation quality, affecting diagnosis. To address this, adaptive filters (LMS, NLMS, RLS) and Independent Component Analysis (FastICA) were evaluated.

All methods met the target of improving SNR by at least 20 dB in synthetically mixed signals, confirming their theoretical effectiveness. RLS stood out among the filters, achieving up to 34 dB improvement and performing reliably across noise types using fixed hyperparameter.

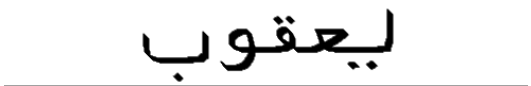
While FastICA showed strong results in controlled conditions, it was unreliable in real-world recordings due to timing and environmental sensitivity. In contrast, RLS with fixed hyperparameter proved practical and robust for real-life use.

These results highlight RLS-based adaptive filtering as a simple, effective solution for enhancing auscultation in noisy conditions, supporting broader deployment in frontline health settings.

Preface

This is Jacob El-Omar's master thesis specialising in Electronics Systems at AAU. The report has been made in the period: September 2024 - May 2025. The report deals with the design and construction of an Active/Adaptive Noise Cancellation For Electronic Stethoscopes. The report consists of an introduction, technical analysis, system design, results, discussion and conclusion. A formal thank you is extended to my supervisor, Jan Østergaard, for his support and guidance, as well as the company partners at Ai Health Highway India Pvt Ltd, particularly Alex Paul Kamson, for his involvement and contributions.

Aalborg University, May 28, 2025



Jacob El-Omar

Reading Guide

The project is divided into nine parts; introduction, technical analysis, requirement specification, system design, results, discussion, conclusion and appendix.

Source citation

In the report, sources are cited by putting numbers in brackets [ ] that refer to the same number in the bibliography on page 82. Appendix: Simulations and Test Reports is used for larger figures, simulations, tables, code and calculations used in the project. Tables and figures are numbered according to chapters, i.e. the first figure in chapter 1 has the number 1.1, etc. Figures and tables without source references are self-made.

Nomenclature

$f(t)$	Continuous function of t (continuous variable)
$x[n]$	Discrete sequence x at index n (discrete variable)
x	Scalar variable x
$\mathbf{x}$	Vector $\mathbf{x}$
$\mathbf{X}$	Matrix $\mathbf{X}$
x^*	Complex conjugate of scalar x
$\mathbf{X}^T$, $\mathbf{X}^H$	Transpose and Hermitian transpose of matrix $\mathbf{X}$
$\mathbf{X}^{-1}$	Inverse of matrix $\mathbf{X}$
x_i, X_{ij}	The ith element of vector $\mathbf{x}$ and the (i,j)th element of matrix $\mathbf{X}$
$\mathrm{tr}(\mathbf{X})$	Trace of matrix $\mathbf{X}$ (sum of diagonal elements)
$r_{xx}(l)$	Autocorrelation at lag l for the discrete sequence $\mathbf{x}$
$r_{xy}(l)$	Cross-correlation at lag l between the sequences $\mathbf{x}$ and $\mathbf{y}$
$\mathbf{R}_{xx} = \mathbb{E}\{\mathbf{x}\mathbf{x}^T\}$	Autocorrelation matrix of vector $\mathbf{x}$
$\mathbf{R}_{xy} = \mathbb{E}\{\mathbf{x}\mathbf{y}^T\}$	Cross-correlation matrix between vectors $\mathbf{x}$ and $\mathbf{y}$
S_{xx}	Power spectral density of $\mathbf{x}$
S_{xy}	Power spectral density of $\mathbf{x}$ and $\mathbf{y}$
$\mathbb{E}\{\cdot\}$	Expectation operator
$\hat{x}$, $\hat{\mathbf{x}}$, $\hat{\mathbf{X}}$	Estimate of scalar x, vector $\mathbf{x}$, and matrix $\mathbf{X}$

Contents

Introduction 1

This thesis explores the application of adaptive noise cancellation techniques to enhance the quality of heart and lung sound auscultation. Auscultation remains a cornerstone of clinical examination, providing valuable diagnostic information at minimal cost. However, its effectiveness is significantly compromised in noisy environments, which presents a particular challenge in resource-constrained settings.

1.1 Motivation

The primary motivation for this project stems from the need to support heart sound screening initiatives in rural India, where healthcare providers face significant challenges in obtaining clean heart sound recordings. In these settings, ambient noise from various sources, including conversations, equipment, and environmental factors, often contaminate the recordings, making accurate interpretation difficult. This issue is especially critical in primary care settings, where initial screening takes place without the benefit of soundproofed examination rooms [1, 2].

Improving signal quality through digital processing can potentially enhance diagnostic accuracy without requiring expensive infrastructure upgrades or specialized equipment. This approach aligns with the broader goal of making healthcare more accessible and effective in resource-limited regions, where cardiovascular diseases remain a significant health burden [3, 4]. The noise samples and project proposal were provided by company partners at Ai Health Highway India Pvt. Ltd.

1.2 State of the Art

Heart and lung sound enhancement has been approached through various signal processing techniques in recent literature. Traditional methods have used fixed filters,

wavelet transforms, and empirical mode decomposition to separate desired signals from noise [5]. More recent approaches have utilized adaptive filtering, which dynamically adjusts filter parameters based on the input signal characteristics.

Several studies have demonstrated the effectiveness of adaptive noise cancellation in heart sound analysis. Kumar et al. [6] implemented least mean squares (LMS) algorithms to reduce noise in phonocardiogram signals, while Messer et al. [7] investigated optimal filter designs for extracting heart sounds from lung sounds. Independent Component Analysis (ICA) has also emerged as a promising technique for separating mixed biomedical signals, with applications in separating maternal and fetal cardiac signals [8] and isolating heart sounds from respiratory noise [9].

In recent years, deep learning-based approaches such as convolutional neural networks and hybrid frameworks have also been explored for heart sound enhancement [4]. Despite these advancements, there is limited literature that compares adaptive filtering approaches and ICA techniques, specifically for heart sound enhancement across diverse noise environments usually they only test one noise type. Furthermore, the practical application of these methods in real-world clinical scenarios remains underexplored, particularly in the context of rural or resource-limited environments [1, 2].

1.3 Focus and Scope

This thesis focuses specifically on enhancing normal heart sounds, primarily due to the need to preserve signal fidelity in screening applications. In such contexts, it is essential that the denoising process does not introduce artifacts that could be mistaken for pathological sounds or obscure subtle clinical features in healthy signals, as this could lead to false negatives with serious consequences. Although this work focuses on normal heart sounds, testing on lung sounds will also be conducted briefly as an additional data type.

The thesis specifically explores two complementary approaches to noise cancellation:

1. Adaptive filtering using least mean squares (LMS), normalised least mean squares (NLMS), and recursive least-squares (RLS) algorithms
2. Independent Component Analysis (ICA) using the FastICA algorithm

The use of these methods was particularly appropriate for this application as the recording setup involved two microphones, providing the necessary inputs for both reference-based adaptive filtering and source separation via ICA. This dual-microphone configuration is relatively easy to implement in practical settings and does not significantly increase complexity or cost compared to single-microphone solutions.

It is important to note that the detailed analysis of abnormal heart sounds falls outside the scope of this thesis. This complex task requires specialised clinical expertise and will be addressed by the company's ongoing collaborations with medical professionals. Instead, this work focuses on providing the foundational signal processing capabilities that can deliver cleaner recordings for subsequent medical analysis.

Technical Analysis 2

This chapter explains the theory and concepts underlying various adaptive filter algorithms. The content is primarily based on Adaptive Filter Theory - Fifth Edition (2014) by Simon Haykin and Lecture Notes in Adaptive Filters - Second Edition (2014) by Jesper K. Nielsen and Søren H. Jensen [10, 11].

2.1 Wiener Filter

The Wiener filter is an essential adaptive filter, known for providing the best linear solution in stationary environments by minimising the mean-square error (MSE) [10]. When choosing a filter, you can select between finite or infinite impulse responses. Finite-duration impulse response (FIR) filters are commonly preferred for their stability, as they use only forward paths, whereas infinite-duration impulse response (IIR) filters, with feedback, can become unstable [10]. This project will use FIR filters and real values for the signals and filter coefficients. Figure 2.1 shows a filtering system that uses a Wiener filter.

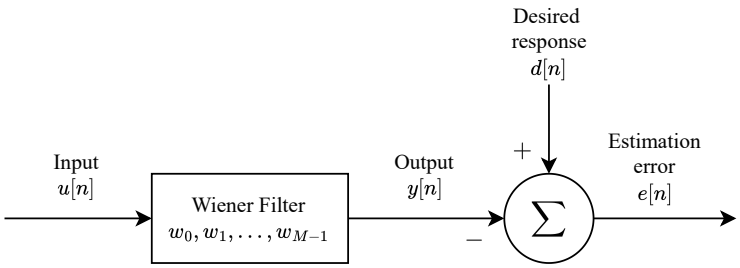


Figure 2.1: General block diagram of the Wiener filter - Modified from [10]

The Wiener Filter is an FIR filter with M taps, defined by the coefficients $w_0, w_1, \ldots, w_{M-1}$ [10, 11].

The estimation error $e[n]$ in figure 2.1 is defined as:

$$e[n] = d[n] - y[n] \tag{2.1}$$

$d[n]$ | Desired response

$y[n]$ | Output

The output is calculated as a linear convolution of the filter coefficients and the input sequence $u[n]$:

$$y[n] = \sum_{k=0}^{M-1} w_k u[n-k] \tag{2.2}$$

The $u[n]$ and $d[n]$ are assumed to be wide-sense stationary (WSS) stochastic processes with constant mean values, both set to zero [11]. This ensures that $e[n]$ represents signal variations without offsets, preventing misinterpretation of offsets as part of the signal.

To minimise the error, we define the cost function as the MSE at time n:

$$J(\mathbf{w}) = \mathbb{E}\left[e^2[n]\right] \tag{2.3}$$

Here, $\mathbb{E}$ represents the statistical expectation [10]. The minimisation of this cost function leads us to the principle of orthogonality.

2.1.1 Principle of Orthogonality

This principle ensures that, at optimality, the estimation error $e[n]$ is orthogonal to all components of the input sequence $u[n]$ [10]. This principle underpins the minimisation of the MSE and defines the conditions for an optimal filter.

To find the filter coefficients that minimise $J(\mathbf{w})$, we apply the k-th element of the gradient vector operator ∇_k with respect to the k-th coefficient w_k to the $J(\mathbf{w})$:

$$\nabla_k J(\mathbf{w}) = \frac{\partial J(\mathbf{w})}{\partial w_k}, \quad k = 0, 1, 2, \ldots \tag{2.4}$$

Expanding the partial derivative:

$$\begin{aligned}\nabla_k J(\mathbf{w}) &= \frac{\partial J(\mathbf{w})}{\partial w_k} \\ &= \frac{\partial \mathbb{E}[e^2[n]]}{\partial w_k} \\ &= 2\mathbb{E}\left[e[n] \cdot \frac{\partial e[n]}{\partial w_k}\right] \\ &= 2\mathbb{E}\left[e[n] \cdot \frac{\partial (d[n] - \sum_{k=0}^{M-1} w_k u[n-k])}{\partial w_k}\right] \\ &= -2\mathbb{E}[u[n-k]e[n]]\end{aligned} \tag{2.5}$$

For the cost function $J(\mathbf{w})$ to attain its minimum value, all the elements of the gradient vector $\nabla J(\mathbf{w})$ must be simultaneously equal to zero; that is,

$$\nabla_k J(\mathbf{w}) = 0, \quad 0 \leq k \leq M-1 \tag{2.6}$$

This leads to the Principle of Orthogonality [10]:

$$\mathbb{E}[u[n-k]e_o[n]] = 0 \tag{2.7}$$

Here, $e_o[n]$ is the optimal estimation error. Equation (2.7) ensures that the input signal and the error signal are uncorrelated, and thus the filter has minimised the MSE.

Further, at the optimal point, the filter output $y_o[n]$ and the estimation error $e_o[n]$ are orthogonal:

$$\mathbb{E}[y_o[n]e_o[n]] = \mathbb{E}\left[\sum_{k=0}^{M-1} w_{o,k} u[n-k]e_o[n]\right] = 0 \tag{2.8}$$

Here, $w_{o,k}$ represents the k-th filter coefficient in the optimal state. This implies that the filter output cannot be further improved by using the error signal. If $e_o[n] = 0$, the filter output $y_o[n]$ completely estimates the desired response $d[n]$.

2.1.2 Wiener-Hopf Equations

The Wiener-Hopf equations describe how to calculate the optimal filter coefficients [10]. Based on the orthogonality principle from equation (2.7):

$$\mathbb{E}\left[u[n-k]\left(d[n]-\sum_{i=0}^{M-1} w_{o,i}u[n-i]\right)\right]=0 \tag{2.9}$$

where $w_{o,i}$ is the ith coefficient in the impulse response of the optimum filter.

Rearranging gives:

$$\sum_{i=0}^{M-1} w_{o,i}\mathbb{E}\left[u[n-k]u[n-i]\right]=\mathbb{E}\left[u[n-k]d[n]\right] \tag{2.10}$$

where the left side represents the auto-correlation function of the filter input, and the right side represents the cross-correlation function between the filter input and the desired signal. For stochastic processes, the correlation functions depend only on the delay between them, allowing equation (2.10) to be rewritten as:

$$\sum_{i=0}^{M-1} w_{o,i}r[i-k]=p[-k],\quad 0\leq k\leq M-1 \tag{2.11}$$

The Wiener–Hopf equations can be written in compact matrix form [10]:

$$\mathbf{R}_{uu}\mathbf{w}_o=\mathbf{p}_{ud} \tag{2.12}$$

where

$\mathbf{R}_{uu}$	$M\times M$ symmetric correlation matrix of the input signal vector $\mathbf{u}[n]$
$\mathbf{w}_o$	Optimal filter weights vector
$\mathbf{p}_{ud}$	Cross-correlation vector between the input signal vector and the desired signal
$\mathbf{u}[n]$	$M\times 1$ tap-input vector

The correlation matrix $\mathbf{R}_{uu}$ is defined as [10]:

$$\mathbf{R}_{uu}=\mathbb{E}\left[\mathbf{u}[n]\mathbf{u}^T[n]\right]=\begin{bmatrix} r[0] & r[1] & \dots & r[M-1] \\ r[1] & r[0] & \dots & r[M-2] \\ \vdots & \vdots & \ddots & \vdots \\ r[M-1] & r[M-2] & \dots & r[0] \end{bmatrix} \tag{2.13}$$

The vectors $\mathbf{w}_o$ and $\mathbf{p}_{ud}$ are defined as:

$$\mathbf{w}_o=[w_{o,0},w_{o,1},\dots,w_{o,M-1}]^T \tag{2.14}$$

$$\mathbf{p}_{ud}=\mathbb{E}[\mathbf{u}[n]d[n]]=[p[0],p[-1],\dots,p[1-M]]^T \tag{2.15}$$

Assuming $\mathbf{R}_{uu}$ is non-singular, the optimal filter weights are given by [10]:

$$\mathbf{w}_o = \mathbf{R}_{uu}^{-1}\mathbf{p}_{ud} \tag{2.16}$$

This solution is valid if both $u[n]$ and $d[n]$ are wide-sense stationary (WSS) processes with known correlation and cross-correlation functions.

2.2 The Least Mean Squares (LMS) Algorithm

In practice, the information in $\mathbf{R}_{uu}$ and $\mathbf{p}_{ud}$ about the environment is often unavailable, making it impossible to directly compute the Wiener solution $\mathbf{w}_o$ [10]. To address this, adaptive filtering algorithms is used to adjust to statistical variations in unknown environments. These algorithms are in this project derived using two methods: Stochastic Gradient Descent (SGD) and Least Squares [10]. The first application of stochastic gradient descent is the the Least Mean-Square (LMS) Algorithm, introduced by Widrow and Hoff in 1960 [10]. Key features of the LMS algorithm include [10]:

- **Simplicity:** Its computational complexity scales linearly with the size of the finite-duration impulse response (FIR) filter.
- **No prior knowledge required:** Unlike the Wiener filter, it doesn't require statistical information about the environment.
- **Robustness:** It performs reliably even under unknown environmental disturbances.
- **No matrix inversion needed:** Unlike the Recursive Least Square (RLS) algorithm, it avoids computationally expensive correlation matrix inversion.

Its goal is to find the weights $\mathbf{w}_o$ that minimise the cost function J, satisfying [10]:

$$J(\mathbf{w}_o) \leq J(\mathbf{w}), \quad \text{for all } \mathbf{w} \tag{2.17}$$

The weights are updated iteratively. At each step, the updated weight vector $\mathbf{w}[n+1]$ reduces the cost function J compared to the current weight vector $\mathbf{w}[n]$:

$$J(\mathbf{w}[n+1]) < J(\mathbf{w}[n]) \tag{2.18}$$

The cost function to minimise is the mean-squared error from equation (2.3). Since environmental statistics are not used, the statistical expectation operator $\mathbb{E}$ is removed. The resulting objective function is approximated instantaneously as:

$$J_s(\mathbf{w}[n]) \approx e[n]^2 \tag{2.19}$$

The subscript s in $J_s(\mathbf{w}[n])$ distinguishes it from the ensemble-average cost function J. Since the estimation error $e[n]$ is a sample of a stochastic process, $J_s(\mathbf{w}[n])$ is also a sample value of a stochastic process. As a result, the derivative of $J_s(\mathbf{w}[n])$ with respect to the k-th tap weight $w_k[n]$ is stochastic, consistent with the stochastic gradient descent method. The gradient of the cost function is calculated using ∇_k:

$$\begin{aligned} \nabla J_{s,k}[n] &= \frac{\partial J_s[n]}{\partial w_k[n]} \\ &= -2u[n-k]e[n], \quad k = 0, 1, \ldots, M-1 \end{aligned} \tag{2.20}$$

Unlike $\mathbb{E}[u[n-k]e[n]]$ from the ideal case, the term $u[n-k]e[n]$ in the simplified objective function causes gradient noise. As a result, the LMS algorithm does not converge to the exact Wiener solution w_o but instead to a "suboptimal" solution $\hat{w}$, which fluctuates around w_o [10].

The k-th weight of the LMS filter is updated as follows [10]:

$$\hat{w}_k[n+1] = \hat{w}_k[n] + \delta\hat{w}_k[n] \tag{2.21}$$

$$= \hat{w}_k[n] - \frac{1}{2}\mu\nabla J_{s,k}[n] \tag{2.22}$$

where

$\delta\hat{w}_k[n]$	A correction, applied in the opposite direction of the gradient of the cost function
μ	A small positive constant balancing convergence speed and stability.

Substituting equation (2.20) into equation (2.22) gives:

$$\hat{w}_k[n+1] = \hat{w}_k[n] + \mu u[n-k]e[n], \quad k = 0, 1, \ldots, M-1 \tag{2.23}$$

This can also be expressed in vector form [10]:

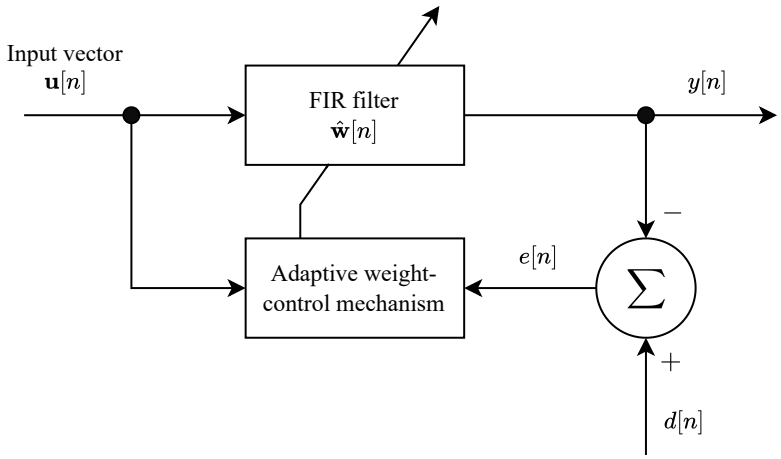
$$\hat{\mathbf{w}}[n+1] = \hat{\mathbf{w}}[n] + \mu\mathbf{u}[n]e[n]. \tag{2.24}$$

where $\hat{\mathbf{w}}[n]$ and $\mathbf{u}[n]$ are defined as:

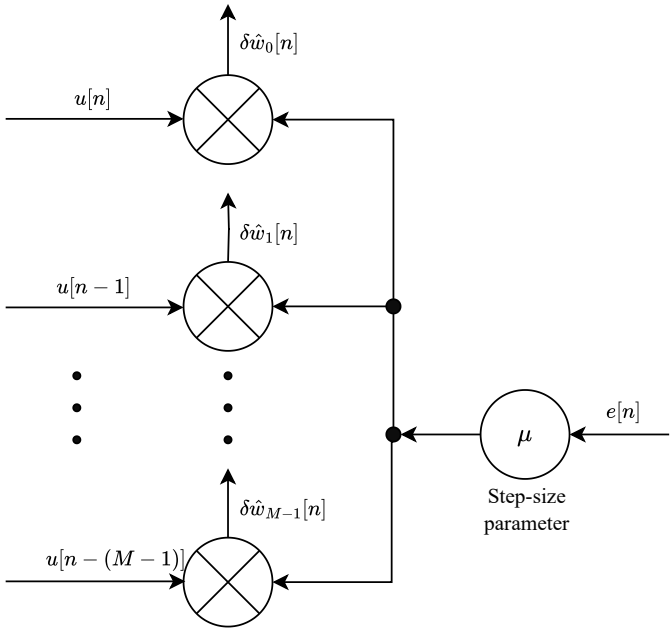
$$\hat{\mathbf{w}}[n] = [\hat{w}_0[n], \hat{w}_1[n], \ldots, \hat{w}_{M-1}[n]]^T \tag{2.25}$$

$$\mathbf{u}[n] = [u[n], u[n-1], \ldots, u[n-(M-1)]]^T \tag{2.26}$$

Figure 2.2 shows a block diagram of the LMS algorithm and its weight control mechanism.



(a) Adaptive FIR filter using the LMS algorithm



(b) Detailed weight-control mechanism

Figure 2.2: LMS algorithm and weight-control mechanism [10].

In Figure 2.2b, the weight adjustments $\delta\hat{\mathbf{w}}_k[n]$ are computed based on the input vector $\mathbf{u}[n]$ and the estimation error $e[n]$. These adjustments update the FIR filter weights in the next cycle $n+1$.

The LMS algorithm steps are summarised below [10]:

Algorithm 1 Least Mean-Square (LMS)

Initialise μ // Step-size parameter
Initialise M // Number of taps (i.e. filter length)
Initialise $\hat{\mathbf{w}}[0]$ // Initial tap-weight vector
for $n = 0, 1, 2, \ldots$ **do**
  $y[n] = \hat{\mathbf{w}}^T[n] \cdot \mathbf{u}[n]$ // Filter output
  $e[n] = d[n] - y[n]$ // Estimation error
  $\hat{\mathbf{w}}[n+1] = \hat{\mathbf{w}}[n] + \mu \cdot \mathbf{u}[n] \cdot e[n]$ // Update weights
end for

2.2.1 Stability of the LMS Algorithm

The stability of the LMS algorithm depends on the step-size parameter μ, constrained by [11]:

$$\lim_{n\to\infty} \Delta\mathbf{w}[n] \approx 0 \tag{2.27}$$

where $\Delta\mathbf{w}[n] = \mathbf{w}_o - \hat{\mathbf{w}}[n]$ is the weight-error vector with the optimal Wiener solution $\mathbf{w}_o$. To ensure convergence, μ must satisfy :

$$\lim_{n\to\infty} (1 - \mu\lambda_k)^n = 0, \quad k = 1, 2, \ldots, M, \tag{2.28}$$

where λ_k are the eigenvalues of the correlation matrix $\mathbf{R}_{uu}$. This requires:

$$-1 < 1 - \mu\lambda_k < 1, \quad \forall k \tag{2.29}$$

which leads to the step-size constraint, ensuring LMS stability:

$$0 < \mu < \frac{2}{\lambda_{\max}}, \tag{2.30}$$

where $\lambda_{\max}$ is the largest eigenvalue of $\mathbf{R}_{uu}$. Since the correlation matrix $\mathbf{R}_{uu}$ and its eigenvalues λ_k are unknown, its trace cannot be directly computed. However, an upper bound on the largest eigenvalue can be estimated based on the statistical properties of the input signal [11]:

$$\lambda_{\max} \leq \text{tr}(\mathbf{R}_{uu}) = \sum_{i=1}^{M} \lambda_i \approx Mr[0] = M\mathbb{E}[u[n]u[n]] = M\mathbb{E}[u^2[n]] \tag{2.31}$$

where $r[0]$ is the autocorrelation of the input at zero lag and $\mathbb{E}[u^2[n]]$ represents the expected power of the input signal. The trace provides an upper bound since the largest eigenvalue cannot exceed the sum of all eigenvalues. Using this approximation, the stability condition for μ can be reformulated as:

$$0 \leq \mu \leq \frac{2}{M\mathbb{E}[u^2[n]]}. \tag{2.32}$$

Since $\mathbb{E}[u^2[n]]$ is typically unknown, it can be estimated using an M-sample segment of the input signal:

$$\hat{\mathbb{E}}[u^2[n]] = \frac{1}{M}\mathbf{u}^T[n]\mathbf{u}[n] = \frac{1}{M}\|\mathbf{u}[n]\|^2 = \frac{1}{M}\sum_{i=0}^{M-1} u^2[n-i] \tag{2.33}$$

Simulation Setup

To illustrate the impact of μ on LMS convergence, Figure 2.3 depicts the evolution of $(1-\mu\lambda)^n$ over 50 adaptation cycles for different step-size values. The input signal is modeled as a unit-power white noise sequence with eigenvalues $\lambda_k = 1$. The LMS update equation is simulated using varying μ values, including cases where the algorithm remains stable $\mu = 0.1, 0.2, 0.5$ and an unstable scenario $\mu = 2$, highlighting divergence. The results show that smaller step sizes ensure stability but slow convergence, whereas larger values accelerate learning at the cost of potential instability.

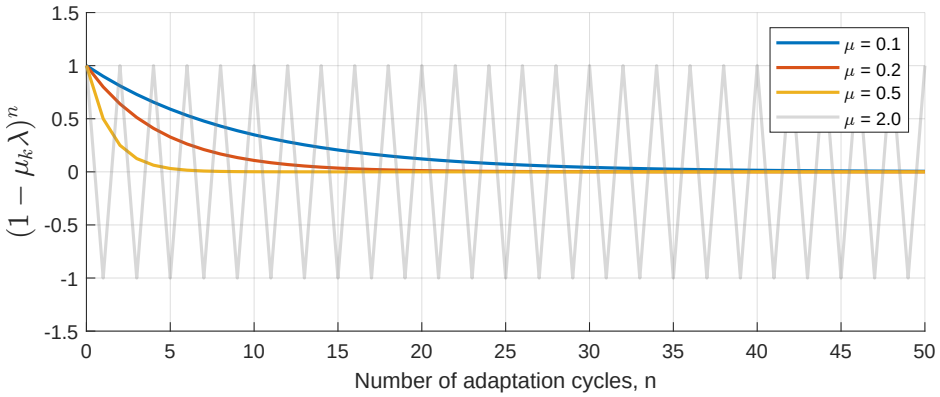


Figure 2.3: Evolution of $(1-\mu\lambda)^n$ over 50 adaptation cycles for varying step-size values and $\lambda_k = 1$.

With a large step-size parameter ($\mu = 0.5$), the equaliser reached steady state in about five

cycles. With a smaller step-size ($\mu = 0.1$), convergence slowed significantly. However, if $\mu > 1$, the system becomes unstable, causing divergence, where the error increases indefinitely and filter weights oscillate.

2.2.2 Computational Complexity of the LMS Algorithm

The computational complexity of the LMS algorithm estimates the computations needed per adaptation cycle. Table 2.1 outlines the cost for real-valued data, based on the steps in algorithm 1 [11, 12]:

Table 2.1: Computational cost per iteration for the LMS algorithm.

Term	$\times$	+ or −
$y[n] = \hat{\mathbf{w}}^T[n]\mathbf{u}[n]$	M	$M-1$
$e[n] = d[n] - y[n]$		1
$\mu e[n]$	1	
$\mu\mathbf{u}[n]e[n]$	M	
$\hat{\mathbf{w}}[n+1] = \hat{\mathbf{w}}[n] + \mu\mathbf{u}[n]e[n]$		M
Total	$2M+1$	$2M$

The algorithm requires $2M+1$ multiplications and $2M$ additions, or $4M+1$ operations per iteration. Its complexity is linear, $\mathcal{O}(M)$, meaning number of operations grow proportionally with the filter length, making the LMS algorithm efficient even for large filter sizes [12].

For comparison, $\mathcal{O}(1)$ implies number of operations remains the same regardless of the input size, $\mathcal{O}(M^2)$ implies number of operations increases with the square of the filter length but $\mathcal{O}(M)$ signifies linear growth in operations with filter length.

2.3 The Normalised Least Mean Squares (NLMS) Algorithm

The LMS algorithm has a limitation where weight adjustments depend on the input signal power. This dependence can cause poor convergence when the input power varies significantly, as adjustments are influenced by power levels rather than the actual signal content in $u[n]$ or the error signal $e[n]$. To address this, the normalised LMS (NLMS) algorithm is used.

2.3.1 The NLMS Algorithm

The NLMS algorithm normalises weight updates by the squared Euclidean norm of the input vector $\mathbf{u}[n]$ [10]. The weight update is expressed as:

$$\hat{\mathbf{w}}[n+1] = \hat{\mathbf{w}}[n] + \frac{\tilde{\mu}}{\|\mathbf{u}[n]\|^2}\mathbf{u}[n]e[n] \tag{2.34}$$

where

$\tilde{\mu}$	Adaptation constant
$\|\mathbf{u}[n]\|^2$	Estimated input power (instantaneous)

Normalization adjusts the step size based on input power, providing scale invariance and consistent convergence behavior [10].

The time-varying step size $\mu[n]$ is defined as:

$$\mu[n] = \frac{\tilde{\mu}}{\|\mathbf{u}[n]\|^2} \tag{2.35}$$

To prevent numerical issues in equation (2.34), such as division by zero when input elements are small, a Regularisation constant $\delta \gtrsim 0$ is added [10, 11]:

$$\mu[n] = \frac{\tilde{\mu}}{\delta + \|\mathbf{u}[n]\|^2} \tag{2.36}$$

The NLMS algorithm is detailed in Algorithm 2.

Algorithm 2 normalised LMS (NLMS)

Initialise $\tilde{\mu}$ // Adaptation constant
Initialise δ // Regularisation constant
Initialise M // Number of taps (i.e. filter length)
Initialise $\hat{\mathbf{w}}[0]$ // Initial tap-weight vector
for $n = 0, 1, 2, \ldots$ **do**
  $y[n] = \hat{\mathbf{w}}^T[n]\mathbf{u}[n]$ // Filter output
  $e[n] = d[n] - y[n]$ // Estimation error
  $\mu[n] = \tilde{\mu}/(\delta + \|u[n]\|^2)$ // Step-size parameter
  $\hat{\mathbf{w}}[n+1] = \hat{\mathbf{w}}[n] + \mu[n]\mathbf{u}[n]e[n]$ // Update weights
end for

2.3.2 Stability of the NLMS Algorithm

For the LMS algorithm, stability is ensured if [10]:

$$0 \leq \mu \leq \frac{2}{M\mathbb{E}[u^2[n]]} \tag{2.37}$$

Substituting $\mathbb{E}[u^2[n]] = \frac{1}{M}\|\mathbf{u}[n]\|^2$ from equation (2.33), the stability condition for NLMS becomes:

$$0 \leq \mu \leq \frac{2}{\|u[n]\|^2} \tag{2.38}$$

The time-varying step-size $\mu[n]$ is normalised by the input signal power:

$$\mu[n] = \frac{\tilde{\mu}}{\|u[n]\|^2} \tag{2.39}$$

To satisfy the stability condition, the adaptation constant $\tilde{\mu}$ must meet:

$$0 < \tilde{\mu} < 2 \tag{2.40}$$

A proper choice of $\tilde{\mu}$ ensures stable and reliable convergence of the NLMS algorithm [11, 12].

2.3.3 Computational Complexity of the NLMS Algorithm

The computational complexity of the NLMS algorithm is summarised in Table 2.2, based on real-valued data [11, 12].

Table 2.2: Computational cost per iteration for real-valued data in the NLMS algorithm.

Term	$\times$	+ or −	$\div$
$\|\mathbf{u}[n]\|^2$	M	$M-1$	
$y[n] = \hat{\mathbf{w}}^T[n]\mathbf{u}[n]$	M	$M-1$	
$e[n] = d[n] - y[n]$		1	
$e[n]\tilde{\mu}/(\delta\|\mathbf{u}[n]\|^2)$	1	1	1
$\mathbf{u}[n]e[n]\tilde{\mu}/(\delta + \|\mathbf{u}[n]\|^2)$	M		
$\mathbf{w}[n+1] = \mathbf{w}[n] + \mu[n]\mathbf{u}[n]e[n]$		M	
Total	$3M+1$	$3M$	1

In total, each iteration of the NLMS algorithm requires $3M + 1$ multiplications, $3M$ additions, and one division, summing to $6M+2$ operations. Its computational complexity is linear with respect to the filter length, $\mathcal{O}(M)$, making NLMS algorithm efficient even for large filter sizes.

2.4 The Recursive Least Squares (RLS) Algorithm

This section introduces the Recursive Least-Squares (RLS) algorithm, an adaptive method designed for fast convergence and high accuracy, ideal for rapidly changing environments.

2.4.1 The RLS Algorithm

The RLS algorithm minimises a least-squares cost function $J_2(\mathbf{w})$, which sums the squared errors over a data segment [10]:

$$J_2(\mathbf{w}) = \sum_{i=i_1}^{i_2} e^2[i] = \sum_{i=i_1}^{i_2} (d[i] - y[i])^2 = \sum_{i=i_1}^{i_2} \left(d[i] - \sum_{k=0}^{M-1} w_k u[i-k] \right)^2 \tag{2.41}$$

where

i_1 and i_2	Define the segment's length
$\mathbf{w}$	Filter weights

To prioritise recent data, the RLS algorithm uses a weighted cost function with a weighting factor $\beta[n,i]$ for $0 < \beta \leq 1$ [10]:

$$J_\beta(\hat{\mathbf{w}}[n]) = \sum_{i=1}^{n} \beta[n,i] e^2[i] = \sum_{i=1}^{n} \beta[n,i] \left(d[i] - \hat{\mathbf{w}}^T[n] u[i] \right)^2 \tag{2.42}$$

Where n is the length of the observable data and the current adaptation cycle. Figure 2.4 illustrates the FIR model used to determine the filter output and error.

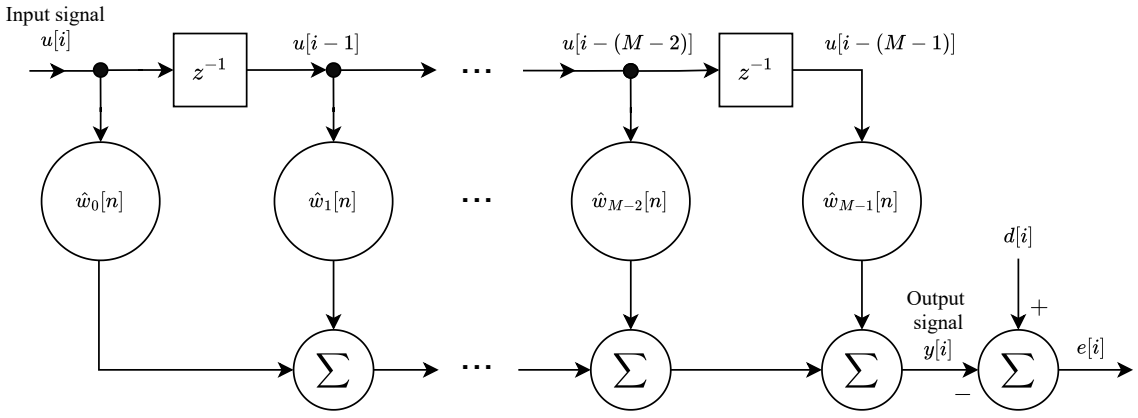


Figure 2.4: FIR filter structure where the input $u[i]$ is processed with weights $\hat{w}_k[n]$ to produce the output $y[i]$, which is compared to the desired signal $d[i]$ to compute the error $e[i]$ [10].

In equation (2.42), the filter weights are applied to both recent and past input measurements to improve performance. The weighting function $\beta[n,i]$ can take different forms, including the exponential function [10]:

$$\beta[n,i] = \lambda^{n-i}, \quad 0 < \lambda \leq 1, \quad i = 1,2,\ldots,n, \tag{2.43}$$

where λ is the forgetting factor which decreasing weight to older error values, as shown in figure 2.5.

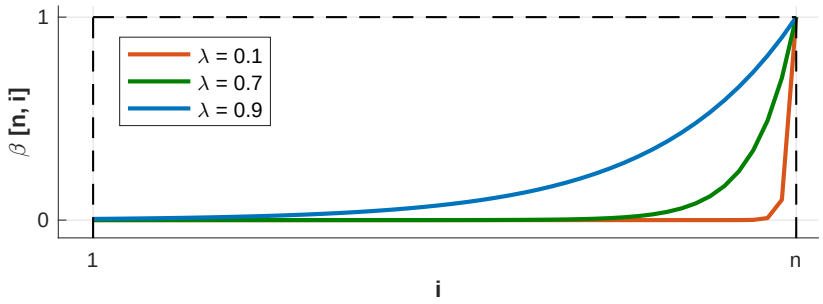


Figure 2.5: The weighting function $\beta[n,i] = \lambda^{n-i}$ for $\lambda = \{0.1, 0.7, 0.9\}$.

The sum in equation (2.42) gives the highest weight to the most recent measurements, with $\lambda^{n-n} = 1$. The algorithm's stability depends on the forgetting factor λ, and stability is ensured if λ meets the condition $0 < \lambda \leq 1$.

Initially, when n is small, the cost function may be affected by noise due to limited data. To address this, a Regularisation term is added, resulting in the following cost function:

$$J_\beta(\hat{\mathbf{w}}[n]) = \sum_{i=1}^{n} \lambda^{n-i} e^2[i] + \delta\lambda^n \|\hat{\mathbf{w}}[n]\|^2 \tag{2.44}$$

Here, δ is the Regularisation parameter, and its effect diminishes as n increases.

The optimal filter coefficients are determined using the cost function in equation 2.44. The gradient of this cost function with respect to the filter weights is calculated and set to zero, resulting in the following system of equations [10]:

$$\nabla J_\beta(\hat{\mathbf{w}}[n]) = 0 \tag{2.45}$$

$$\sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}[i]\mathbf{u}^T[i]\hat{\mathbf{w}}[n] + \delta\lambda^n\hat{\mathbf{w}}[n] = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}[i]d[i] \tag{2.46}$$

$$\mathbf{\Phi}[n]\hat{\mathbf{w}}[n] = \mathbf{z}[n] \tag{2.47}$$

where

$\mathbf{\Phi}[n]$	$M \times M$ time-average auto-correlation matrix of the filter input
$\mathbf{z}[n]$	$M \times 1$ time-average cross-correlation vector

The correlation matrix $\mathbf{\Phi}[n]$ is symmetric and positive definite, ensuring invertibility. The Regularisation term makes sure $\mathbf{\Phi}[n]$ is invertible from the start. To make the calculations recursive, the most recent data point is used to update $\mathbf{\Phi}[n]$ and $\mathbf{z}[n]$ [10]:

$$\begin{aligned}
\mathbf{\Phi}[n] &= \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}[i]\mathbf{u}^T[i] + \delta\lambda^n\mathbf{I} \\
&= \lambda\left[\sum_{i=1}^{n-1} \lambda^{n-1-i}\mathbf{u}[i]\mathbf{u}^T[i] + \delta\lambda^{n-1}\mathbf{I}\right] + \lambda^{n-n}\mathbf{u}[n]\mathbf{u}^T[n] \\
&= \lambda\left[\sum_{i=1}^{n-1} \lambda^{n-1-i}\mathbf{u}[i]\mathbf{u}^T[i] + \delta\lambda^{n-1}\mathbf{I}\right] + \mathbf{u}[n]\mathbf{u}^T[n] \\
&= \lambda\mathbf{\Phi}[n-1] + \mathbf{u}[n]\mathbf{u}^T[n].
\end{aligned} \tag{2.48}$$

This equation indicates that the present value of the correlation matrix $\mathbf{\Phi}[n]$ is computed by the old matrix $\mathbf{\Phi}[n-1]$ and the outer product of the newly arrived input sequence. The same approach for $\mathbf{z}[n]$ leads to [10]:

$$\begin{aligned}
\mathbf{z}[n] &= \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}[i]d[i] \\
&= \lambda\left[\sum_{i=1}^{n-1} \lambda^{n-1-i}\mathbf{u}[i]d[i]\right] + \mathbf{u}[n]d[n] \\
&= \lambda\mathbf{z}[n-1] + \mathbf{u}[n]d[n].
\end{aligned} \tag{2.49}$$

After calculating $\mathbf{\Phi}[n]$ and $\mathbf{z}[n]$, the filter weights are obtained by inverting the matrix $\mathbf{\Phi}[n]$ and solving the system. However, this can be computationally expensive, especially for large filter lengths. To address this, the Matrix Inversion Lemma is used.

2.4.2 The Matrix Inversion Lemma

The matrix inversion lemma, also called Woodbury's Identity, simplifies the computation of a matrix inverse [10, 13]. It applies to an $M \times M$ matrix $\mathbf{A}$ defined as:

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T \tag{2.50}$$

where

$\mathbf{B}$ and $\mathbf{D}$	Positive-definite matrices
$\mathbf{C}$	$M \times N$ matrix

The inverse of $\mathbf{A}$ is:

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^T\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{B} \tag{2.51}$$

The following relationships can be derived from equations 2.48 and 2.50:

$$\mathbf{A} \equiv \mathbf{\Phi}[n] \tag{2.52}$$

$$\mathbf{B}^{-1} \equiv \lambda\mathbf{\Phi}[n-1] \tag{2.53}$$

$$\mathbf{C} \equiv \mathbf{u}[n] \tag{2.54}$$

$$\mathbf{D} \equiv 1 \tag{2.55}$$

Using the lemma, the inverse of $\mathbf{\Phi}[n]$ is computed as:

$$\mathbf{\Phi}^{-1}[n] = \lambda^{-1}\mathbf{\Phi}^{-1}[n-1] - \lambda^{-1}\mathbf{\Phi}^{-1}[n-1]\mathbf{u}[n]\left(1 + \mathbf{u}^T[n]\lambda^{-1}\mathbf{\Phi}^{-1}[n-1]\mathbf{u}[n]\right)^{-1}\mathbf{u}^T[n]\lambda^{-1}\mathbf{\Phi}^{-1}[n-1] \tag{2.56}$$

or, equivalently:

$$\begin{aligned}\mathbf{\Phi}^{-1}[n] &= \lambda^{-1}\mathbf{\Phi}^{-1}[n-1] - \lambda^{-1}\frac{\lambda^{-1}\mathbf{\Phi}^{-1}[n-1]\mathbf{u}[n]}{1 + \lambda^{-1}\mathbf{u}^T[n]\mathbf{\Phi}^{-1}[n-1]\mathbf{u}[n]}\mathbf{u}^T[n]\mathbf{\Phi}^{-1}[n-1] \\ &= \lambda^{-1}\mathbf{\Phi}^{-1}[n-1] - \lambda^{-1}\mathbf{k}[n]\mathbf{u}^T[n]\mathbf{\Phi}^{-1}[n-1]\end{aligned} \tag{2.57}$$

Using equation (2.57), the inverse of the auto-correlation matrix $\mathbf{\Phi}[n]$ is not recalculated at every cycle. Instead, the matrix inversion lemma allows direct updates to $\mathbf{\Phi}^{-1}[n]$, avoiding costly computations. The filter weight update is expressed as:

$$\hat{\mathbf{w}}[n] = \mathbf{\Phi}^{-1}[n]\mathbf{z}[n] \tag{2.58}$$
$$= \hat{\mathbf{w}}[n-1] + \mathbf{k}[n](d[n] - \hat{\mathbf{w}}^T[n-1]\mathbf{u}[n]) \tag{2.59}$$
$$= \hat{\mathbf{w}}[n-1] + \mathbf{k}[n]\xi[n] \tag{2.60}$$

where

$\mathbf{k}[n]$	Gain vector, determines the weight adjustment
$\xi[n]$	Priori estimation error

This error is the difference between the desired signal and the filter output, computed before updating the coefficients, unlike LMS algorithms. Figure 2.6 shows the RLS algorithm's block diagram:

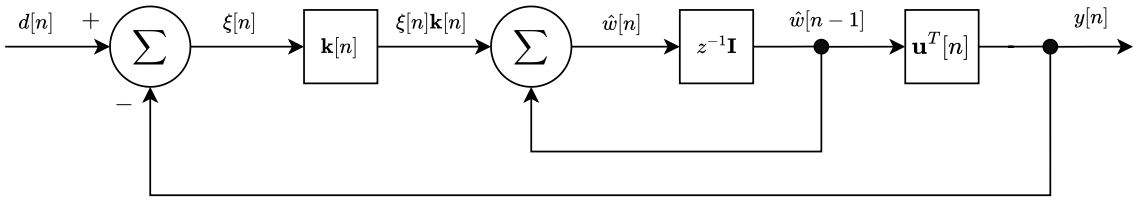


Figure 2.6: RLS algorithm block diagram with filtering and weight adaptation loops [10]. The update for $\mathbf{\Phi}^{-1}[n]$ is not shown.

An intermediate vector, $\boldsymbol{\pi}[n] = \mathbf{\Phi}^{-1}[n-1]\mathbf{u}[n]$, is introduced to simplify the gain vector, which is expressed as [10]:

$$\mathbf{k}[n] = \frac{\mathbf{\Phi}^{-1}[n-1]\mathbf{u}[n]}{\lambda + \mathbf{u}^T[n]\mathbf{\Phi}^{-1}[n-1]\mathbf{u}[n]} = \frac{\boldsymbol{\pi}[n]}{\lambda + \mathbf{u}^T[n]\boldsymbol{\pi}[n]} \tag{2.61}$$

The update for $\mathbf{\Phi}^{-1}[n]$ also uses $\boldsymbol{\pi}[n]$, as $\mathbf{\Phi}[n]$ is symmetric:

$$\mathbf{\Phi}^{-1}[n] = \frac{\mathbf{\Phi}^{-1}[n-1] - \mathbf{k}[n]\boldsymbol{\pi}^T[n]}{\lambda} \tag{2.62}$$

The initial inverse auto-correlation matrix $\mathbf{\Phi}^{-1}[0]$ is set as:

$$\mathbf{\Phi}[0] = \delta I \Rightarrow \mathbf{\Phi}^{-1}[0] = \delta^{-1}\mathbf{I} \tag{2.63}$$

This is used in the first adaptation cycle, where $\mathbf{\Phi}^{-1}[0]$ Initialises the process. Algorithm 3 summarises the RLS steps [10].

Algorithm 3 Recursive Least-Squares (RLS)

Initialise λ // Forgetting factor
Initialise δ // Regularisation constant
Initialise M // Number of taps (i.e. filter length)
Initialise $\hat{\mathbf{w}}[0]$ // Initial tap-weight vector
Initialise $\mathbf{\Phi}^{-1}[0] = \delta^{-1}\mathbf{I}$ // Initial correlation matrix
for $n = 1, 2, \ldots$ **do**
  $\boldsymbol{\pi}[n] = \mathbf{\Phi}^{-1}[n-1]\mathbf{u}[n]$ // Intermediate vector
  $\mathbf{k}[n] = \boldsymbol{\pi}[n]/(\lambda + u^T[n]\boldsymbol{\pi}[n])$ // Gain vector
  $y[n] = \hat{\mathbf{w}}^T[n-1]\mathbf{u}[n]$ // Filter output
  $\xi[n] = d[n] - y[n]$ // Priori error
  $\hat{\mathbf{w}}[n] = \hat{\mathbf{w}}[n-1] + \mathbf{k}[n]\xi[n]$ // Update weights
  $\mathbf{\Phi}^{-1}[n] = (\mathbf{\Phi}^{-1}[n-1] - \mathbf{k}[n]\boldsymbol{\pi}^T[n])/\lambda$ // Update correlation matrix
end for

2.4.3 Computational Complexity

The computational complexity of the RLS algorithm is summarised in Table 2.3 based on real-valued data [11, 12].

Table 2.3: Computational cost of the RLS algorithm for real-valued data.

Term	$\times$	+ or –	$\div$
$\boldsymbol{\pi}[n] = \mathbf{\Phi}^{-1}[n-1]\mathbf{u}[n]$	M^2	$M(M-1)$	
$\mathbf{k}[n] = \boldsymbol{\pi}[n]/(\lambda + \mathbf{u}^T[n]\boldsymbol{\pi}[n])$	M	M	1
$y[n] = \hat{\mathbf{w}}^T[n-1]\mathbf{u}[n]$	M	$M-1$	
$\xi[n] = d[n] - y[n]$		1	
$\hat{\mathbf{w}}[n] = \hat{\mathbf{w}}[n-1] + \mathbf{k}[n]\xi[n]$	M	M	
$\mathbf{\Phi}^{-1}[n] = (\mathbf{\Phi}^{-1}[n-1] - \mathbf{k}[n]\boldsymbol{\pi}^T[n])/\lambda$	M^2	M^2	M^2
Total	$2M^2 + 3M$	$2M^2 + 2M$	$M^2 + M$

In total, the RLS algorithm performs $5M^2 + 6M$ operations. Operations increases with the square of the filter length, $\mathcal{O}(M^2)$, which makes it the more computational demanding in comparison to the LMS and the NLMS algorithm. Figure 2.7 compares the complexities of LMS, NLMS, and RLS algorithms.

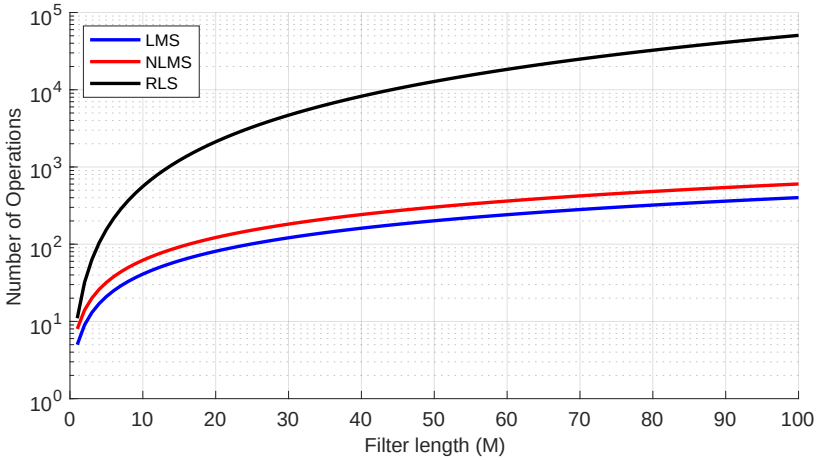


Figure 2.7: Computational cost of the LMS, NLMS and RLS algorithms.

Figure 2.7 highlights the RLS algorithm's high computational cost.

2.5 Independent Component Analysis

Independent Component Analysis (ICA) is a method used to separate a mixture of signals into their original, independent sources. For example, if several people are speaking at the same time and their voices are recorded together, ICA can help isolate each person's voice from the mixture. It works by assuming that the original signals come from different sources and are statistically independent of one another. This technique is especially useful in situations where the signals of interest are mixed with unwanted noise even when no clean reference signal is available.

2.5.1 Mathematical Definition of ICA

Let $\mathbf{x} = (x_1, x_2, \ldots, x_m)^T$ represent an observed data vector, where each x_i is a linear mixture of n unknown, independent source signals. This mixing process can be modeled as [14]:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{2.64}$$

where

$\mathbf{x} = (x_1, x_2, \ldots, x_m)^T$	Observed mixed signal vector
$\mathbf{s} = (s_1, s_2, \ldots, s_n)^T$	Original independent source vector
$\mathbf{A}$	Unknown mixing matrix of size $m \times n$

The goal of ICA is to estimate a demixing matrix $\mathbf{W}$ that can separate the mixed signals and give us back signals that are as independent as possible:

$$\mathbf{y} = \mathbf{W}\mathbf{x} \tag{2.65}$$

where

$\mathbf{y}$	Estimated original source vector
$\mathbf{W}$	Demixing matrix to approximate $\mathbf{A}^{-1}$

The resulting vector $\mathbf{y}$ is an estimate of the original sources, although they may be scaled or appear in a different order [14].

2.5.2 Preprocessing for ICA

Before applying ICA, the observed signal must go through two preprocessing steps: **centering** and **whitening** [14].

Centering involves removing the mean from the observed data vector to ensure it has zero mean. This is done by subtracting the expected value (or sample mean, in practice) from $\mathbf{x}$:

$$\mathbf{x} \leftarrow \mathbf{x} - \mathbb{E}[\mathbf{x}] \tag{2.66}$$

The left arrow notation "$\leftarrow$" indicates that we're updating the vector $\mathbf{x}$ with the result of the right-hand operation.

Centering is important because it simplifies the ICA model, which assumes zero mean for both the observed mixtures and the source signals. Without centering, an additional bias term would be needed, complicating the separation. Additionally, centering is a prerequisite for the whitening step that follows.

Whitening transforms the centered data so that its components are uncorrelated and each has unit variance. This is achieved by computing the covariance matrix of the centered

data:

$$\mathbf{C}_x = \mathbb{E}[\mathbf{x}\mathbf{x}^T] \tag{2.67}$$

Since $\mathbf{C}_x$ is a real, symmetric, and positive semi-definite matrix, it can be decomposed via eigenvalue decomposition (EVD) as:

$$\mathbf{C}_x = \mathbf{E}\mathbf{D}\mathbf{E}^T \tag{2.68}$$

where

$\mathbf{E}$	Orthogonal matrix of eigenvectors (i.e., $\mathbf{E}^T = \mathbf{E}^{-1}$)
$\mathbf{D}$	Diagonal matrix of eigenvalues

The whitening matrix is defined as [14]:

$$\mathbf{V} = \mathbf{D}^{-1/2}\mathbf{E}^T \tag{2.69}$$

Applying this whitening matrix to the centered data yields the whitened vector:

$$\mathbf{z} = \mathbf{V}\mathbf{x} \tag{2.70}$$

The result is the vector $\mathbf{z}$, a whitened version of the observed data $\mathbf{x}$ with uncorrelated, standardised components. For simplicity, the complete whitening process is denoted by the function `whiten()`.

The next step involves applying ICA to the whitened signal $\mathbf{z}$ to extract statistically independent source components.

2.5.3 FastICA Algorithm

FastICA is an efficient algorithm that finds independent components by maximising non-Gaussianity. This approach is based on the central limit theorem, which suggests that mixtures of independent random variables tend to be more Gaussian than the original variables themselves.

Measuring Non-Gaussianity

A common measure of non-Gaussianity is kurtosis, defined for a zero-mean random variable y as [14]:

$$\text{kurt}(y) = \mathbb{E}[y^4] - 3(\mathbb{E}[y^2])^2 \tag{2.71}$$

For a Gaussian distribution, kurtosis equals zero. Deviations from zero indicate non-Gaussianity. However, since kurtosis is sensitive to outliers, a more robust alternative is the use of negentropy [14]. Negentropy is defined as the difference in entropy between a Gaussian random variable and the variable of interest. If only one nonquadratic function G is used, the approximation becomes [14]:

$$J(y) \approx (\mathbb{E}[G(y)] - \mathbb{E}[G(\nu)])^2 \tag{2.72}$$

where

ν	Standard Gaussian variable
$G(\cdot)$	Non-quadratic function

Commonly used approximations for G include [14]:

$$G_1(y) = \frac{1}{a_1} \log \cosh(a_1 y), \quad 1 \leq a_1 \leq 2 \tag{2.73}$$

$$G_2(y) = -\exp\left(-\frac{y^2}{2}\right) \tag{2.74}$$

Estimating a Single Independent Component

The FastICA algorithm begins by preprocessing the observed data $\mathbf{x}$, which includes centering and whitening. Let $\mathbf{z}$ denote the whitened data.

The estimation of a single independent component proceeds iteratively. First, a random vector $\mathbf{w}$ of unit norm is Initialised. Then, the algorithm updates this vector using a fixed-point iteration scheme [14]:

$$\mathbf{w} \leftarrow \mathbb{E}[\mathbf{z} g(\mathbf{w}^T \mathbf{z})] - \mathbb{E}[g'(\mathbf{w}^T \mathbf{z})]\mathbf{w} \tag{2.75}$$

The nonlinearity g is derived from G. For example, when using $G_1(y) = \log \cosh(y)$, the corresponding derivative and second derivative are [14]:

$$g(y) = \tanh(y) \tag{2.76}$$

$$g'(y) = 1 - \tanh^2(y) \tag{2.77}$$

After each update, the vector $\mathbf{w}$ is normalised:

$$\mathbf{w} \leftarrow \frac{\mathbf{w}}{\|\mathbf{w}\|} \tag{2.78}$$

The iteration continues until convergence, typically determined by a threshold ϵ, such that:

$$|\mathbf{w}^\top \mathbf{w}_{\text{old}}| > 1 - \epsilon \tag{2.79}$$

The resulting independent component is then obtained by projecting the whitened data:

$$y = \mathbf{w}^T \mathbf{z} \tag{2.80}$$

Algorithm 4 summarises the FastICA steps for a single component [14]:

Algorithm 4 FastICA for a single component

```
x ← x − E[x]                          // Center the data
z ← whiten(x)                         // Whiten the data
Initialise w                          // Random initial demixing vector
repeat
    w_old ← w                         // Store current vector
    w ← E[z g(w^T z)] − E[g'(w^T z)] w   // Fixed-point update
    w ← w/‖w‖                         // Normalise to unit norm
until |w^T w_old| > 1 − ε             // Check convergence
y = w^T z                             // Extract component
```

So far, this process has focused on estimating a single independent component. While it is possible to estimating multiple components by repeatedly running the algorithm with different initialisations, this approach tends to be unreliable due to potential redundancy between the components.

A more effective strategy leverages the fact that, in the whitened space, the independent components' corresponding vectors, $\mathbf{w}_i$, are orthogonal to one another. This orthogonality property allows for the use of orthogonalisation techniques to reliably estimate multiple independent components.

Estimating Multiple Independent Components

When estimating multiple components, one must ensure that each component corresponds to a distinct source. FastICA supports two strategies to avoid redundant estimation: deflationary and symmetric orthogonalisation. In the deflationary approach, components are estimated one at a time, with each new component being made orthogonal to the ones already found components using Gram-Schmidt-like

projections. In the symmetric approach, all components are estimated simultaneously and orthogonalised after each iteration.

Deflationary orthogonalisation is chosen in this project due to its ease of implementation.

The process begins with whitening, followed by iterative estimation of each independent component. After computing the weight vector $\mathbf{w}_p$ for the p-th component, orthogonality is enforced by subtracting its projections onto all earlier components [14]:

$$\mathbf{w}_p \leftarrow \mathbf{w}_p - \sum_{j=1}^{p-1} (\mathbf{w}_p^T \mathbf{w}_j)\mathbf{w}_j \tag{2.81}$$

This is followed by normalisation:

$$\mathbf{w}_p \leftarrow \frac{\mathbf{w}_p}{\|\mathbf{w}_p\|} \tag{2.82}$$

Once all m components are estimated, they are stacked into a demixing matrix $\mathbf{W}$, and the full set of independent components is obtained via:

$$\mathbf{y} = \mathbf{W}\mathbf{z} \tag{2.83}$$

Algorithm 5 outlines the full procedure for deflationary FastICA [14].

Algorithm 5 FastICA with deflationary orthogonalisation for multiple components

$\mathbf{x} \leftarrow \mathbf{x} - E[\mathbf{x}]$ // Center the data
$\mathbf{z} \leftarrow \text{whiten}(\mathbf{x})$ // Whiten the data
Choose m // Set number of components
$p \leftarrow 1$ // Initialise component counter
while $p \leq m$ **do**
  Initialise $\mathbf{w}_p$ // Initial demixing vector
  repeat
    $\mathbf{w}_p \leftarrow E[\mathbf{z}g(\mathbf{w}_p^T\mathbf{z})] - E[g'(\mathbf{w}_p^T\mathbf{z})]\mathbf{w}$ // Fixed-point update
    $\mathbf{w}_p \leftarrow \mathbf{w}_p - \sum_{j=1}^{p-1} (\mathbf{w}_p^T\mathbf{w}_j)\mathbf{w}_j$ // Deflationary orthogonalise
    $\mathbf{w}_p \leftarrow \mathbf{w}_p / \|\mathbf{w}_p\|$ // Normalise to unit norm
  until $\mathbf{w}_p$ has converged // Check convergence
  $p \leftarrow p + 1$ // Move to next component
end while
$\mathbf{y} = \mathbf{W}\mathbf{z}$ // Extract all components

Computational Complexity for FastICA

The computational complexity of the FastICA algorithm is summarised in Table 2.4 based on real-valued data for estimating m independent components from N observations of M-dimensional data [15].

Table 2.4: Computational cost of the FastICA algorithm for real-valued data.

Term	$\times$	+ or –	$\div$
Preprocessing:			
$\mathbf{x} \leftarrow \mathbf{x} - \mathbb{E}[\mathbf{x}]$		MN	
$\mathbf{C}_x = \mathbb{E}[\mathbf{x}\mathbf{x}^T]$	M^2N	$M^2(N-1)$	
$\mathbf{C}_x = \mathbf{E}\mathbf{D}\mathbf{E}^T$	$\mathcal{O}(M^3)$	$\mathcal{O}(M^3)$	
$\mathbf{V} = \mathbf{D}^{-1/2}\mathbf{E}^T$	M^2		M
$\mathbf{z} = \mathbf{V}\mathbf{x}$	M^2N	$M(M-1)N$	
Per iteration (for component p):			
$\mathbf{w} \leftarrow \mathbb{E}[\mathbf{z}g(\mathbf{w}^T\mathbf{z})] - \mathbb{E}[g'(\mathbf{w}^T\mathbf{z})]\mathbf{w}$	$2MN + M$	$2MN + M$	
$\mathbf{w}_p \leftarrow \mathbf{w}_p - \sum_{j=1}^{p-1}(\mathbf{w}_p^T\mathbf{w}_j)\mathbf{w}_j$	Mp	$M(p-1)$	
$\mathbf{w} \leftarrow \mathbf{w}/\|\mathbf{w}\|$	M	$M-1$	1
Preprocessing total	$2M^2N + M^2 + \mathcal{O}(M^3)$	$M^2(N-1) + M(M-1)N + \mathcal{O}(M^3)$	M
Per iteration total	$2MN + M + Mp + M$	$2MN + M + M(p-1) + M - 1$	1

FastICA's total complexity depends on the average iterations per component, K (usually 3–10 [15]), and the number of components, m. The preprocessing step takes about $\mathcal{O}(M^3 + M^2N)$ operations, mainly from eigenvalue decomposition ($\mathcal{O}(M^3)$) and covariance/whitening ($\mathcal{O}(M^2N)$) [16]. The iterative part adds roughly $\mathcal{O}(KmM^2 + KmMN)$ for all components, since each new component must be orthogonalised against previous ones. So, the overall complexity is

$$\mathcal{O}(M^3 + M^2N + KmM^2 + KmMN) \tag{2.84}$$

where for large M the eigenvalue step dominates, but for large N or many components m, the iteration costs matter more. With a small K, FastICA is efficient compared to gradient-based ICA [17, 18], but still more costly than simple adaptive filters [19].

Figure 2.8 compares the computational complexity of LMS, NLMS, RLS, and FastICA algorithms.

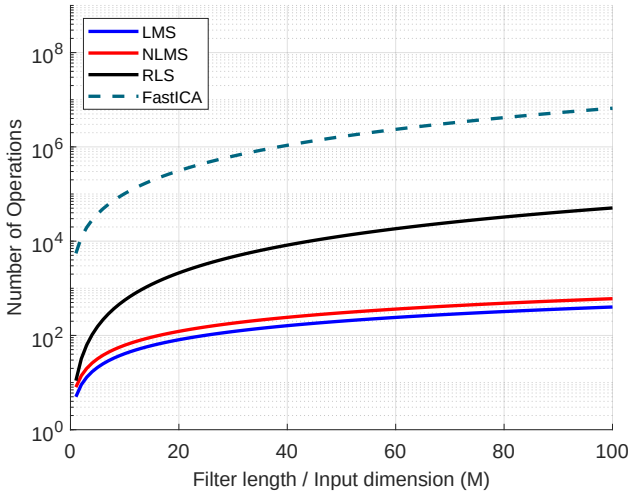


Figure 2.8: Computational cost of the LMS, NLMS, RLS, and FastICA algorithms across varying filter/input dimensions M.

The simulation evaluates these complexities over filter/input dimensions $M = 1$ to 100, fixing the number of observations to $N = 500$. For FastICA, the number of independent components is set to $m = 2$ with an average of $K = 5$ iterations per component. The complexity calculations incorporate key operations specific to each algorithm. All results are presented on a logarithmic scale to clearly illustrate the differences in computational cost.

2.6 Comparison Criteria

To evaluate the performance of the filtering algorithms used in this project, three criteria are considered: signal-to-noise ratio (SNR), convergence rate, and the Mel spectrogram. These metrics provide insight into how effectively each algorithm adapts to noise, improves signal quality, and enhances perceptual sound characteristics.

2.6.1 Signal-to-Noise Ratio (SNR)

The Signal-to-Noise Ratio (SNR) is a key metric used to evaluate the effectiveness of filtering algorithms in enhancing signal quality. It quantifies the ratio of the power of the original clean signal to the power of the residual noise remaining after filtering. The output SNR is defined as [10]:

$$\text{SNR}_{\text{out}} = 10\log_{10}\left(\frac{\sum_{n=1}^{N} s^2[n]}{\sum_{n=1}^{N}(s[n]-\hat{s}[n])^2}\right) \tag{2.85}$$

where

$s[n]$	Original clean signal
$\hat{s}[n]$	Estimated (filtered) signal
N	Total number of samples

A higher SNR value indicates better suppression of noise and improved fidelity of the recovered signal.

2.6.2 Convergence Rate

The convergence rate describes how quickly a filtering algorithm minimises the error between the desired signal and the filter output. It is typically evaluated using the Mean Square Error (MSE), which is defined as [10]:

$$\text{MSE}[n] = \mathbb{E}[|e[n]|^2], \quad e[n] = d[n] - \hat{s}[n] \tag{2.86}$$

where

$e[n]$	Error signal
$d[n]$	Desired signal

An algorithm that rapidly reduces the MSE to a low and steady value is said to converge quickly. In this project, convergence rate is visualised by plotting MSE versus the number of input samples. Filtering algorithms that quickly converge typically deliver faster results with better performance, although they may be more computationally intensive depending on the approach.

2.6.3 Mel Spectrogram

The Mel spectrogram is used to analyse the time-frequency characteristics of the signals, as it aligns more closely with human auditory perception. Humans perceive pitch and loudness on a logarithmic scale, being more sensitive to differences at lower frequencies than at higher ones [20]. The Mel scale accounts for this non-linear perception of pitch, while the Decibel scale reflects the logarithmic perception of loudness. Therefore, the Mel spectrogram provides a more perceptually relevant visual representation of sound,

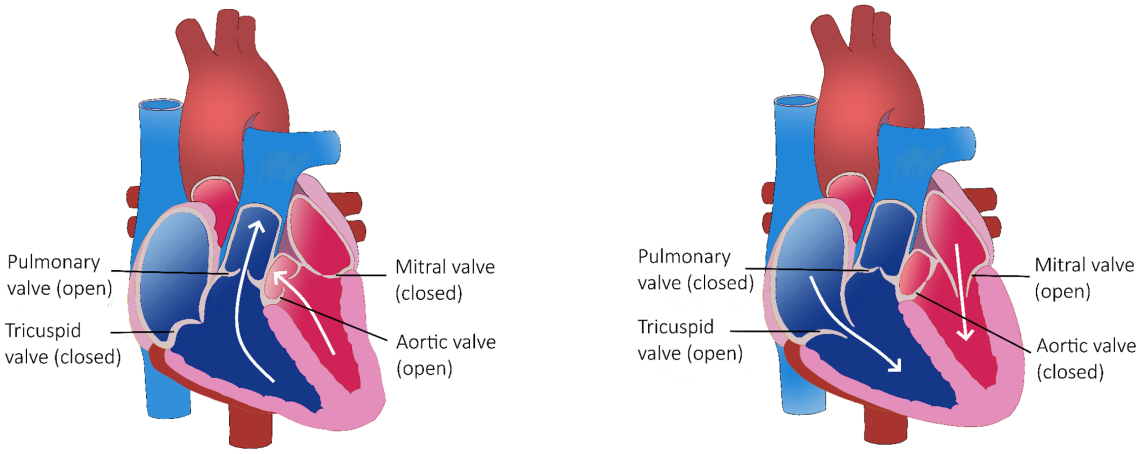
making it easier to evaluate improvements in signal quality. The Mel spectrogram is particularly useful for evaluating the effectiveness of filtering algorithms in preserving important perceptual features while reducing noise, providing a more intuitive way of assessing filtering performance.

2.7 Identifying Normal and Abnormal Heart and Lung Signals

The identification of normal and abnormal heart and lung signals plays a crucial role in diagnosing cardiovascular and pulmonary conditions. Heart and lung sounds result from physiological processes and can exhibit significant variations depending on pathological conditions. This section outlines the key characteristics of normal sounds, indicators of abnormalities, and methods used for their identification.

2.7.1 Heart Sounds

Normal heart sounds are primarily produced by the closure of the heart valves, specifically the atrioventricular valves. This results in the two fundamental heart sounds (FHS): S1 ("Lub") and S2 ("Dub") [21]. Each of these sounds corresponds to distinct valvular events, as illustrated in Figure 2.9:



(a) S1 sound is produced by the closure of the Mitral and Tricuspid valves.

(b) S2 sound is produced by the closure of the Aortic and Pulmonary valves.

Figure 2.9: Heart sounds and their corresponding valve closures [21, 22].

Further analysis of these sounds can be observed in the heart sound waveform shown in

Figure 2.10, which provides an example of the cyclical nature of the sounds.

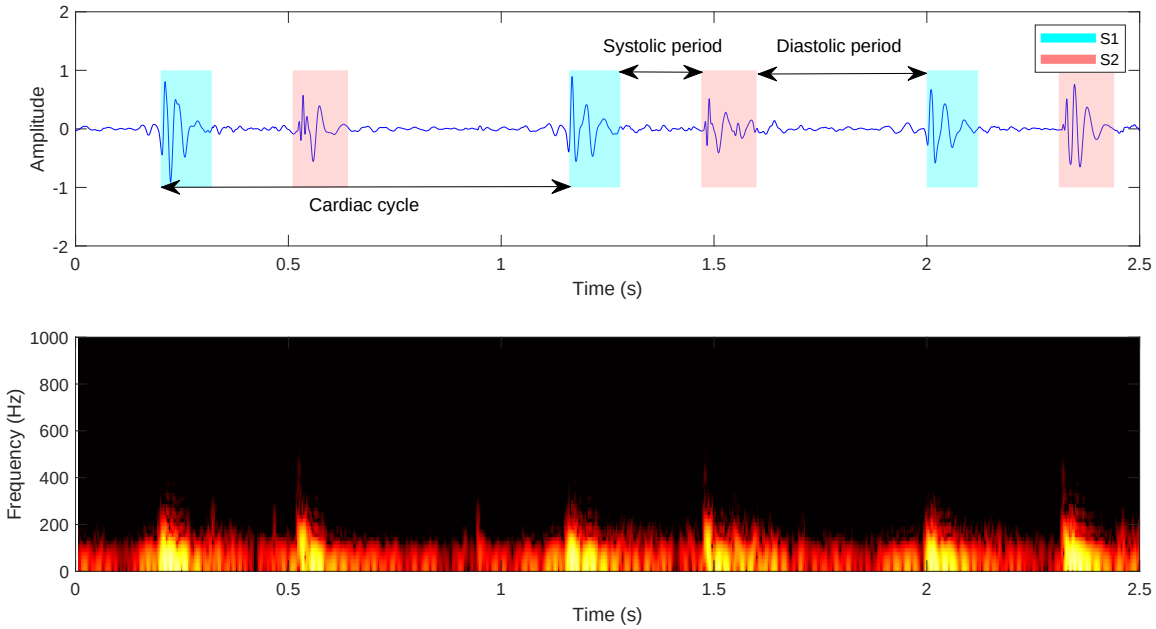


Figure 2.10: Example of normal heart sound waveform

In addition to the primary heart sounds (S1 and S2), S3 and S4 sounds may occur under specific conditions. The heart cycle consists of two main phases: Systole, during which the heart contracts and pumps blood (from S1 to S2), and Diastole, when the heart relaxes and fills with blood (from S2 back to the next S1). The heart cycle duration (HCD) is the total time for one complete cardiac cycle, which includes both systole (contraction) and diastole (relaxation):

$$\text{HCD} \propto \frac{1}{\text{HR}} \tag{2.87}$$

Thus, a higher heart rate (HR) results in a shorter heart cycle period, reducing the durations of both systole and diastole.

In addition to the typical durations of the primary heart sounds (S1: 120 ± 22 ms, S2: 92 ± 22 ms [23]) , the frequency ranges of various heart sounds provide important diagnostic information. Table 2.5 summarises the frequency ranges of common heart sounds, including S3, S4, and abnormal sounds associated with conditions like mitral stenosis, ejection murmurs, and regurgitation.

Table 2.5: Heart sound frequencies and associated ranges [23].

Heart sound	Frequency ranges (Hz)
S3 and S4	15-65
S1 and S2	20-200
Mitral stenosis	40-80
Ejection murmurs	200-400
Regurgitation	250-700

These frequency ranges are key in identifying specific cardiac conditions and can aid in differentiating between normal and abnormal heart sounds.

2.7.2 Lung Sounds

Normal lung sounds are primarily produced by inspiration, during which air flows into the lungs, and expiration, when air is expelled. These sounds can be categorised into two main types: Vesicular breath sounds and Tracheal/Bronchial sounds [22]. The process of breathing is illustrated in figure 2.11.

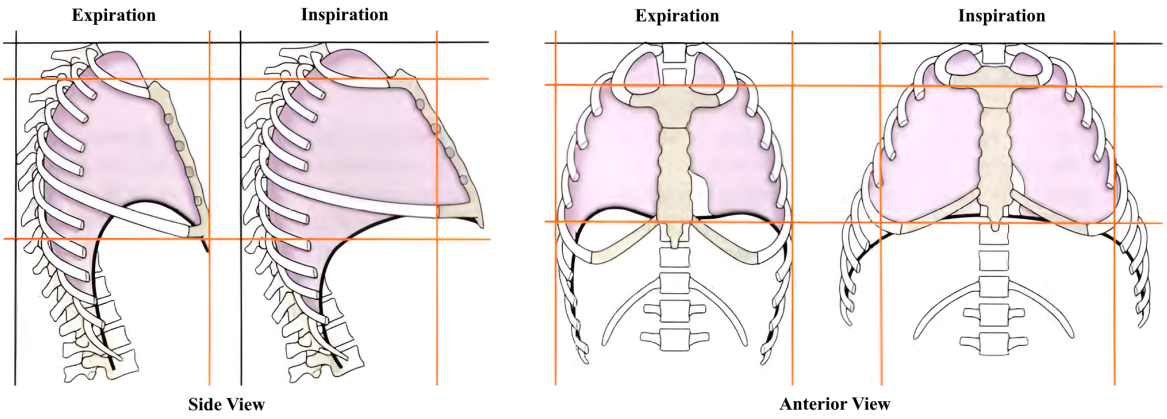


Figure 2.11: The variations in thoracic cavity and lung capacity during respiration [22].

Further analysis of these sounds can be observed in the lung sound waveform shown in Figure 2.12, which provides an example of the characteristic cyclical patterns during inspiration and expiration.

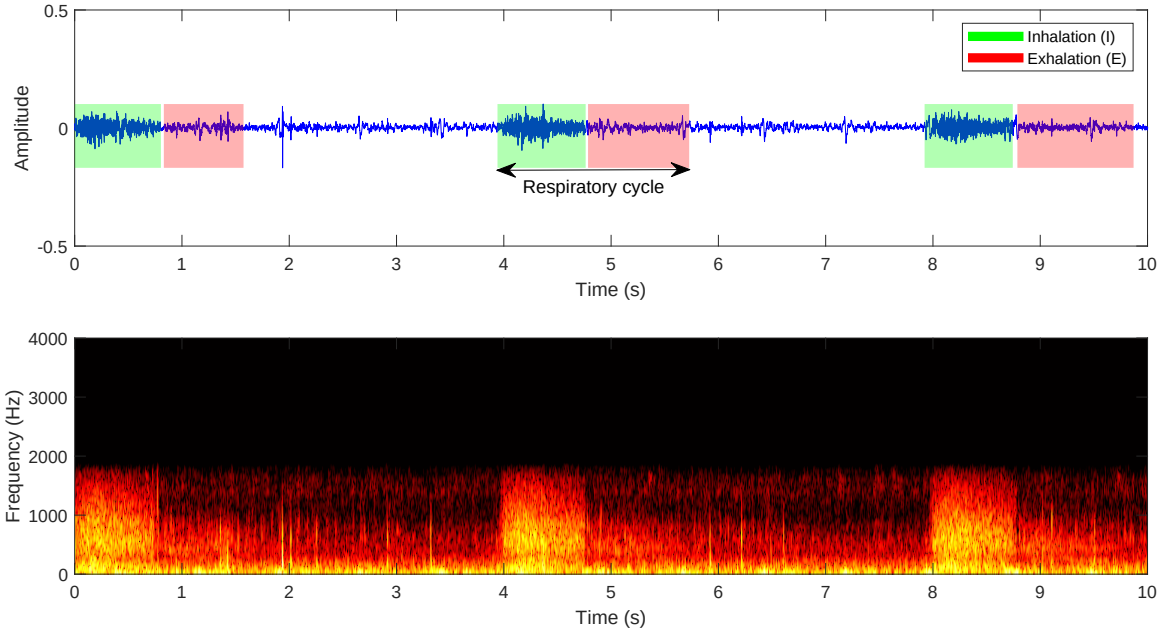


Figure 2.12: Example of normal lung sound waveform and its spectrogram.

In addition to these normal sounds, abnormal lung sounds such as wheezes, crackles, and rhonchi may occur which indicate respiratory distress or obstructions. The respiratory cycle duration (RCD) is the total time for one complete respiratory cycle, including both inspiration and expiration:

$$\text{RCD} \propto \frac{1}{\text{RR}} \tag{2.88}$$

Thus, a higher respiratory rate (RR) results in a shorter respiratory cycle period, reducing the durations of both inspiration and expiration.

In addition to the typical durations of vesicular and tracheal sounds, the frequency ranges of various lung sounds provide important diagnostic information. Table 2.6 summarises the frequency ranges of common lung sounds, including wheezes, crackles, and rhonchi.

Table 2.6: Lung sound frequencies and associated ranges [24].

Lung Sound	Frequency Range (Hz)
Normal Breath Sounds	100 – 2000
Wheezes	400 Hz – 2 kHz
Crackles	< 200
Rhonchi	< 300
Cough Sound	50 – 3000

These frequency ranges are key in identifying specific lung conditions and can aid in differentiating between normal and abnormal lung sounds.

2.7.3 Measurement and Auscultation Points

This section outlines key auscultation points for heart and lung sounds using a stethoscope. It highlights standard anatomical sites for accurate assessment, with figures illustrating the equipment and auscultation locations.

Stethoscope

The stethoscope is a vital medical instrument used for auscultation, enabling the detection of heart and lung sounds [25]. It consists of a chest piece with a diaphragm and a bell, tubing for sound transmission, and earpieces for listening. Figure 2.13 illustrates a standard stethoscope.

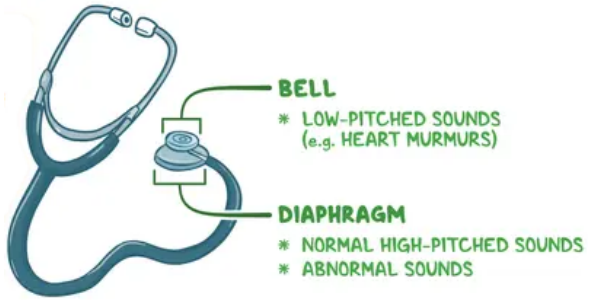


Figure 2.13: Standard stethoscope [25].

The stethoscope enables clinicians to assess heart function by auscultating specific anatomical locations where heart sounds are best heard. These auscultation points correspond to different heart valves and are essential for detecting abnormalities in cardiac activity.

Heart Auscultation Points

Auscultation is performed using a stethoscope at specific anatomical locations corresponding to heart valves. Figure 2.14 illustrates the standard auscultation sites for heart sounds.

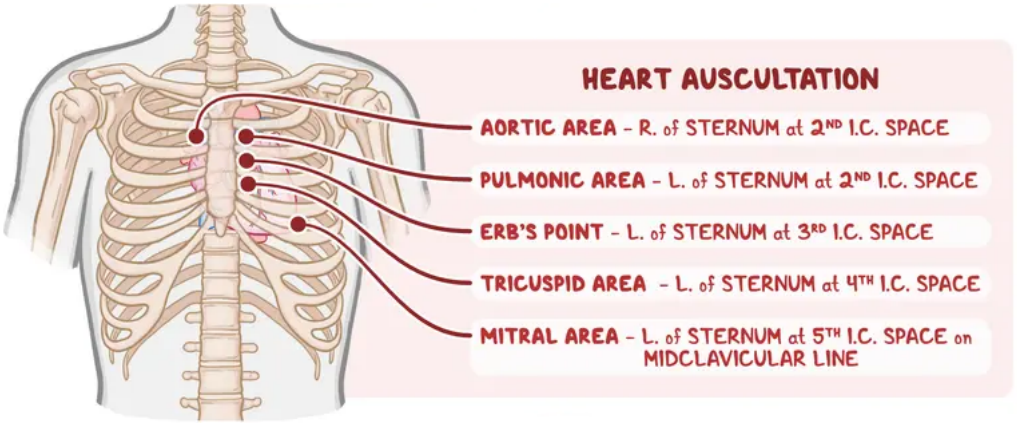


Figure 2.14: Standard auscultation sites for heart sounds [25].

Beyond cardiac assessment, the stethoscope is also used to evaluate lung sounds. Proper lung auscultation follows a systematic approach, allowing for the identification of normal breath sounds and potential respiratory conditions.

Lung Auscultation Points

Lung auscultation follows a *stepladder pattern*, comparing sounds on both sides of the chest [25]. Figure 2.15 illustrates the standard auscultation sites for heart sounds.

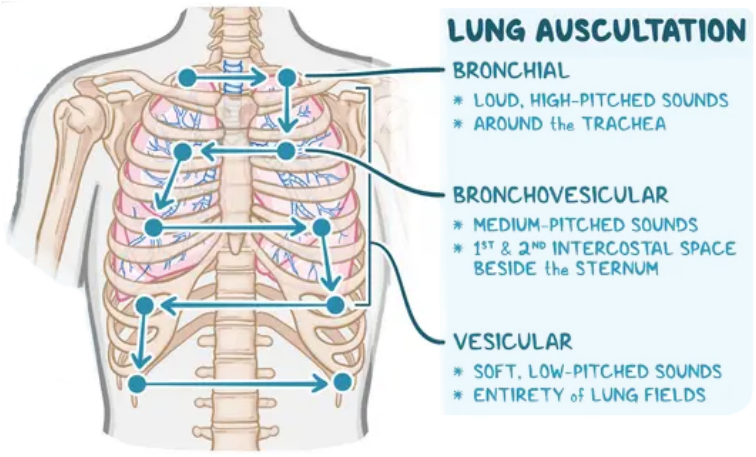


Figure 2.15: Standard auscultation sites for lung sounds [25].

Despite the effectiveness of auscultation, environmental noise and interference from body sounds can obscure critical details in heart and lung sounds. Traditional stethoscope-based assessments rely heavily on clinical experience, making them prone to variability and subjective interpretation. The integration of advanced signal processing techniques, such as adaptive noise cancellation, can enhance auscultation by improving the clarity of heart and lung sounds. Addressing these challenges is crucial for developing more reliable diagnostic tools.

2.8 Problem statement

The introduction and technical analysis lead to the following problem statement:

How can adaptive noise cancellation techniques be utilised to enhance the quality of heart and lung sound auscultation in clinical and non-clinical environments?

Requirement Specification 3

This chapter defines the requirements necessary to address the problem statement presented in section 2.8. These requirements serve as the foundation for the design, implementation, and evaluation of the signal enhancement system.

3.1 Objective

The objective of this project is to develop a system capable of improving the quality of heart and lung sound recordings through adaptive filtering or FastICA. The system must ensure that key biomedical features remain distinguishable and analysable in the presence of environmental and physiological noise. Effectiveness will be assessed through quantitative metrics. A key goal for the adaptive filter is to find a fixed set of general parameters i.e. step sizes, forgetting factors and filter length, that work for different noise types.

3.2 Technical Requirement

The effectiveness of the system is evaluated using a single, measurable technical criterion:

Table 3.1: Technical Requirement

Index	Requirement	Target Value
T1	Minimum SNR improvement after signal enhancement.	> 20 dB

The technical requirement focuses solely on achieving a minimum SNR improvement because SNR is a widely accepted and quantifiable metric for assessing signal enhancement performance. This objective measurement provides a clear benchmark for system effectiveness across different noise conditions.

System Design 4

This chapter presents the design and implementation of the adaptive filtering system for noise cancellation in heart and lung sound recordings. The system is developed in multiple stages, beginning with simulations on synthetic data and progressing towards real-world testing. The purpose of this approach is to ensure a systematic evaluation of the filtering techniques before applying them to actual heart and lung sound recordings.

4.1 System Overview

Adaptive noise cancellation (ANC) is a signal processing method that suppresses unwanted noise by estimating it from the input and subtracting it, thereby isolating and recovering the clean underlying signal. In this project, ANC is applied to denoise heart and lung recordings. Figure 4.1 shows a block diagram of the ANC system.

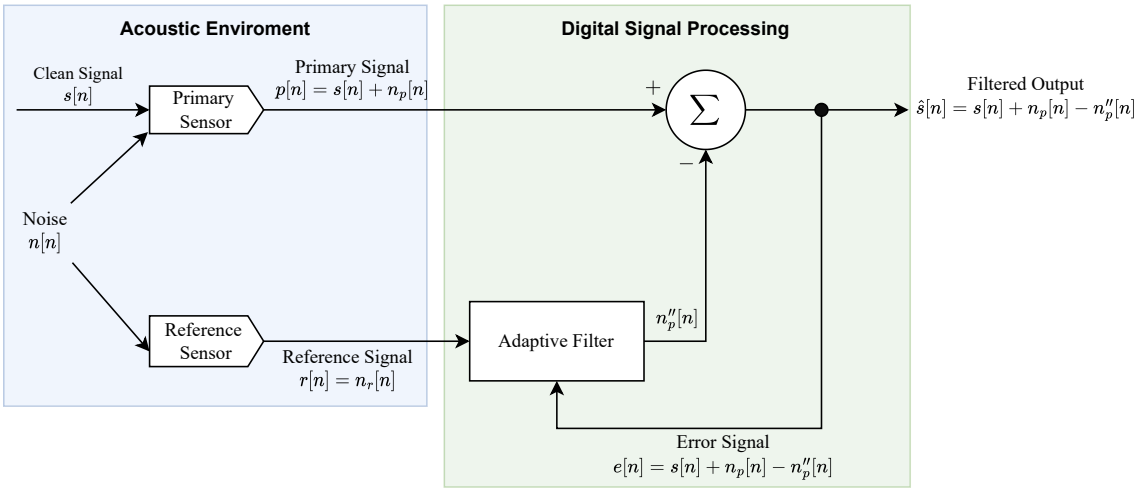


Figure 4.1: Block diagram of the ANC system. The system is divided into two domains: the Acoustic Environment, where signals are captured by sensors, and Digital Signal Processing, where adaptive filtering occurs to suppress noise.

The system consists of two main components: the **Acoustic Environment** and the **Digital Signal Processing** block.

- In the Acoustic Environment, the **Primary Sensor** captures the clean signal $s[n]$ (e.g., heart or lung sounds) along with noise $n_p[n]$, resulting in the primary signal $p[n] = s[n] + n_p[n]$ (also called the noisy signal). The **Reference Sensor** records only noise $n_r[n]$, producing the reference signal $r[n] = n_r[n]$.
- The reference signal is passed to the **Adaptive Filter**, which estimates the noise component $\hat{n}_p''[n]$ correlated with the primary signal. This estimate is subtracted from the primary signal to produce the **Filtered Output** $\hat{s}[n] = s[n] + n_p[n] - \hat{n}_p''[n]$.
- The same output is used internally as the **Error Signal** $e[n]$ for the adaptive algorithm to update its coefficients. Although the error signal and the filtered output are mathematically equivalent, the error signal serves a feedback role within the adaptive filtering process.

This system configuration enables real-time noise reduction and dynamic adaptation to changing noise conditions.

4.2 Grid Search for Parameter Optimisation

Grid search is a method for tuning hyperparameters by testing all possible combinations from a predefined range. For adaptive filtering (LMS, NLMS, RLS), this involves varying the filter length and either the step size (μ) or forgetting factor (λ) to find the best setup.

The tested parameter ranges are shown in Table 4.1.

Table 4.1: Parameter ranges used in the grid search.

Parameter	Range
Filter Length	[1, 2, 3, 4, 5, 6, 8, 10, 12, 16, 24, 32, 40, 60, 80, 100]
Step Size (μ)	[0.0001, 0.001, 0.002, 0.005, 0.0075, 0.01, 0.015, 0.02, 0.025, 0.03, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
Forgetting Factor (λ)	[0.92, 0.95, 0.97, 0.98, 0.985, 0.99, 0.995, 0.998, 0.9985, 0.999, 0.9992, 0.9995, 0.9997, 0.9999]

Filter lengths vary from 1 to 100, while step sizes vary from 0.0001 to 0.9 and forgetting factors vary from 0.92 to 0.9999. Each combination is evaluated by applying the filter and calculating the resulting SNR. The configuration with the highest SNR is selected as the optimal choice.

4.3 Sinus Wave

To establish a functional baseline, the filtering algorithms are first tested on synthetic data consisting of artificially generated signals with additive noise. The objective is to verify the basic functionality of the adaptive filters before applying them to real recordings.

4.3.1 System Setup

The simulated data includes a clean 10 Hz sine wave signal, which is corrupted by Gaussian noise with standard deviation of 0.5. The parameters for each filtering algorithm are tuned to evaluate their noise suppression capabilities. The system's performance is assessed by calculating the SNR before and after filtering. The tuning parameters used for this setup were set using the gridsearch method and can be seen in the table 4.2.

Table 4.2: Tuning parameters for the adaptive filters.

Parameter	Value
Sampling rate	8000 Hz
Filter length (LMS)	1
Filter length (NLMS)	1
Filter length (RLS)	1
Step size (LMS)	0.015
Step size (NLMS)	0.005
Forgetting factor (RLS)	0.9999

The MATLAB code for setup can be found in appendix A.

4.3.2 Performance Evaluation

The figure 4.2 displays a side-by-side comparison of the clean signal, primary signal, and error signals produced by the LMS, NLMS, and RLS filters.

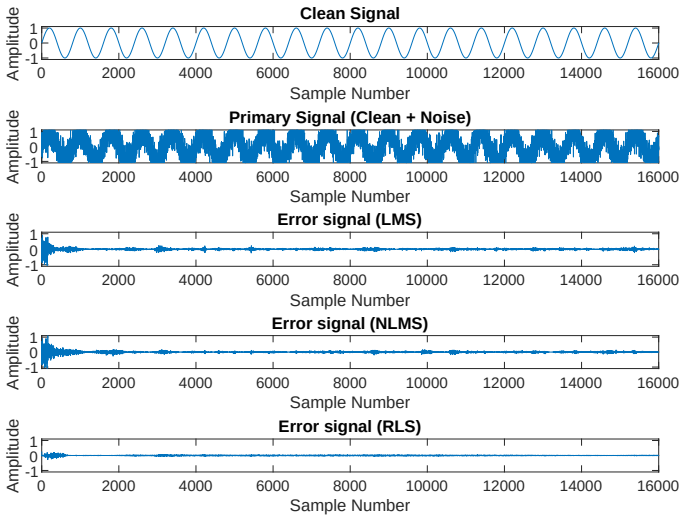


Figure 4.2: Error signals - Sinus Wave.

Figure 4.2 shows that during the first 500 samples, all filters exhibit a higher error amplitude but that the error amplitude gradually decrease approaching near-zero levels in later samples. The RLS filter yields the cleanest error signals.

Figure 4.3 illustrates how the output SNR varies with filter length across all three algorithms.

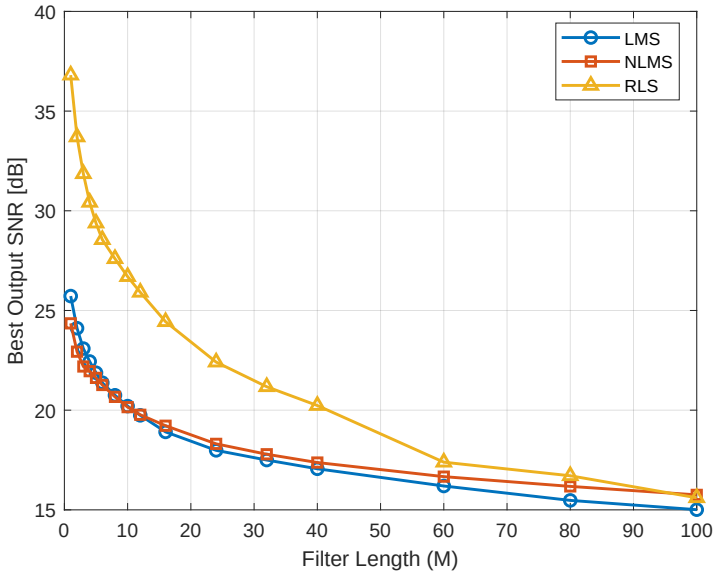


Figure 4.3: Output SNR vs Filter Length - Sinus Wave.

The output SNR as a function of filter length shows that as the filter length increases, the output SNR of the filtered signals decreases, appearing to follow an exponential trend. The highest output SNR is achieved with a filter length of 1 by the RLS algorithm at 38 dB, followed by LMS and NLMS reaching 27 dB and 26 dB respectively.

Figure 4.4 shows the mel spectrogram of the sinus wave.

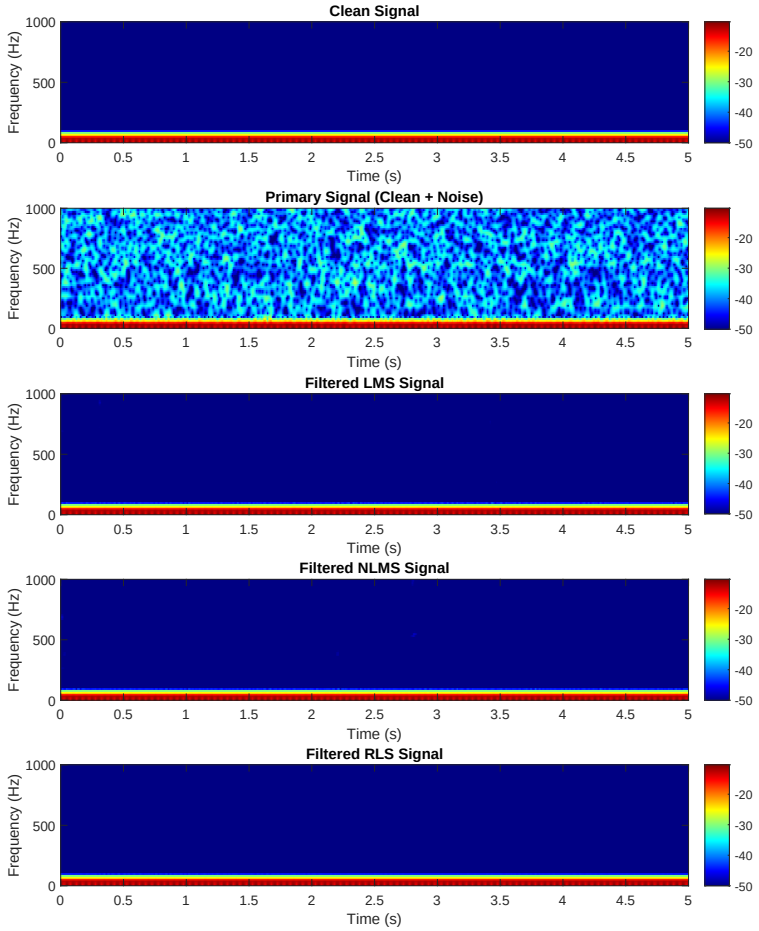


Figure 4.4: Mel Spectrogram - Sinus Wave.

The clean sinusoidal signal shows spectral content at very low frequencies, while the primary signal includes noise from added Gaussian noise. The filtered outputs using LMS, NLMS, and RLS recover the sinus wave leaving no visible noise behind.

Figure 4.5 shows the error convergence curves for the LMS, NLMS, and RLS algorithms

when applied to a noisy sinusoidal signal.

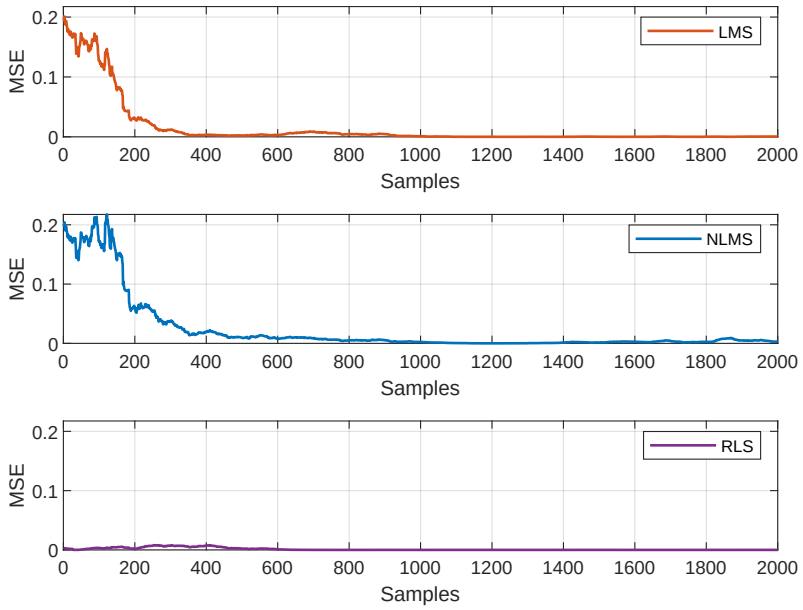


Figure 4.5: Error Convergence Curve - Sinus Wave.

The LMS filter converges fast, reaching steady-state after around 400 samples with a small residual error. The NLMS filter converges a bit slower than LMS. The RLS filter shows the fastest convergence, reducing the error significantly within the first sample and achieving the lowest steady-state error overall. All curves are smoothed with a moving average window to highlight general trends.

The SNR values before and after filtering for each algorithm are summarised in the table 4.3:

Table 4.3: Results after adaptive filtering for sinus wave.

Filtering Method	Output SNR (dB)	SNR Improvment (dB)
No Filtering	-1.7496	-
LMS	25.7225	27.4721
NLMS	24.3545	26.1041
RLS	36.8042	38.5539

The results demonstrate that the NLMS filter is the least effective filter whilst the LMS and

RLS filters achieves the highest SNR improvement.

4.4 Synthetic Heartbeat

Following the sinus wave evaluation, the system is tested using synthetic heartbeat sound generated using Matlab. The test setup includes controlled noise environments, allowing for a systematic analysis of filter performance.

Below is an illustration of the experimental setup used in real-life conditions:

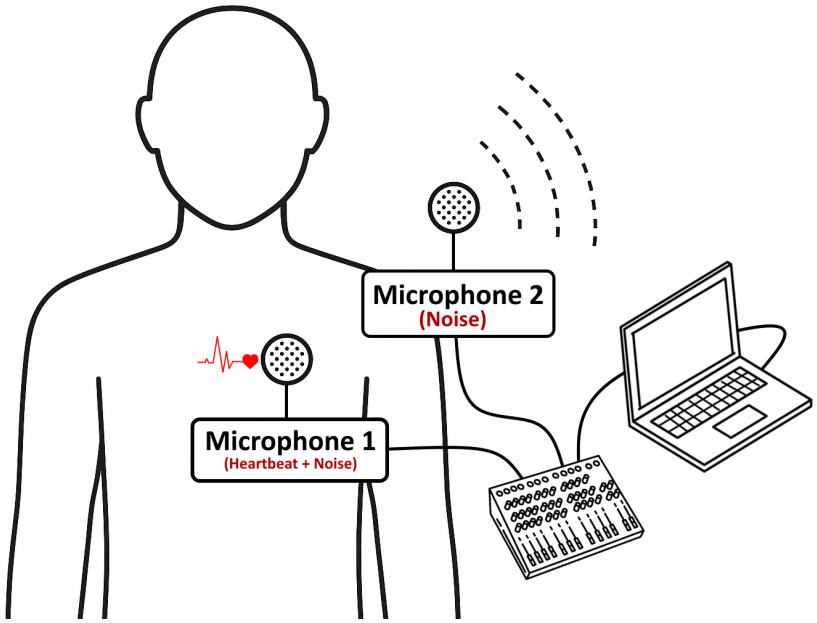


Figure 4.6: Experimental setup for heartbeat sound acquisition in noisy environments.

The required mixtures were combined digitally as follows:

$$\text{Primary} = \text{Clean Signal} + \text{Noise} \tag{4.1}$$

$$\text{Secondary} = \text{Noise} \tag{4.2}$$

These input mixture are meant to imitate what would happen if two microphones were employed: Microphone 1 being placed directly on the chest near the heart to capture both heartbeat and noise, while Microphone 2 is positioned slightly away from the body but still close to the chest, oriented toward the noise source to capture reference noise signals.

4.4.1 System Setup

The tuning parameters used for this setup were set using grid search and can be seen in the table 4.4.

Table 4.4: Tuning parameters for the adaptive filters.

Parameter	Value
Sampling rate	8000 Hz
Filter length (LMS)	1
Filter length (NLMS)	1
Filter length (RLS)	1
Step size (LMS)	0.01
Step size (NLMS)	0.03
Forgetting factor (RLS)	0.9999

The MATLAB code for this setup, including the simulation of signal and noise as well as the implementation of the filtering algorithms, can be found in appendix A.

4.4.2 Performance Evaluation

The figure 4.7 displays a side-by-side comparison of the clean signal, primary signal, and error signals produced by the LMS, NLMS, and RLS filters.

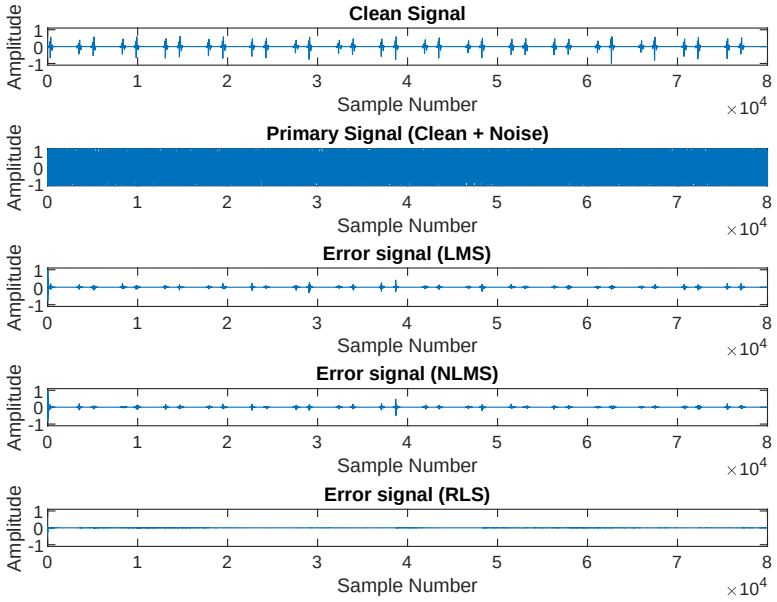


Figure 4.7: Error signals - Synthetic Heartbeat.

All filters significantly reduce the noise but both LMS and NLMS exhibit a slight periodic error pattern during the synthetic heartbeats, whereas RLS produces the cleanest result, demonstrating superior performance on the synthetic heartbeat signal.

Figure 4.8 illustrates the output SNR as a function of filter length across all three algorithms.

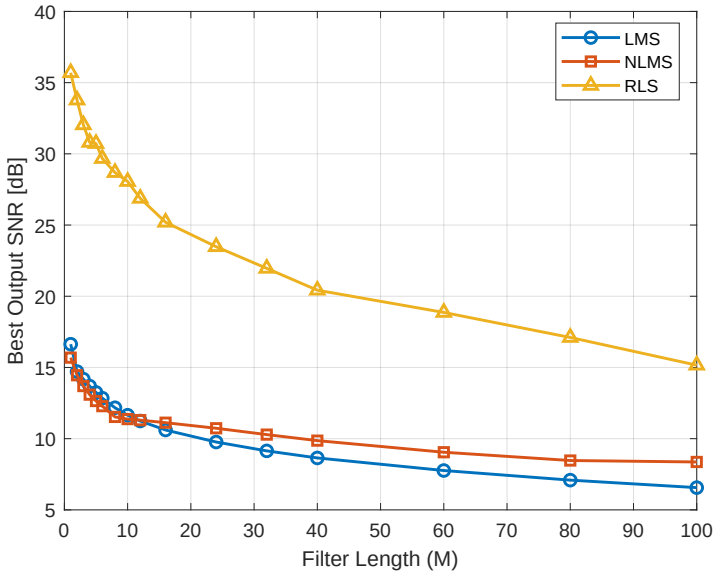


Figure 4.8: Output SNR vs Filter Length - Synthetic Heartbeat.

The output SNR appears to decrease with the increase of the filter length. The greatest output SNR is achieved with a filter length of 1 by the RLS algorithm at 35 dB, followed by LMS and NLMS, both reaching around 16 dB.

Figure 4.9 shows the mel spectrogram of the synthetic heartbeat.

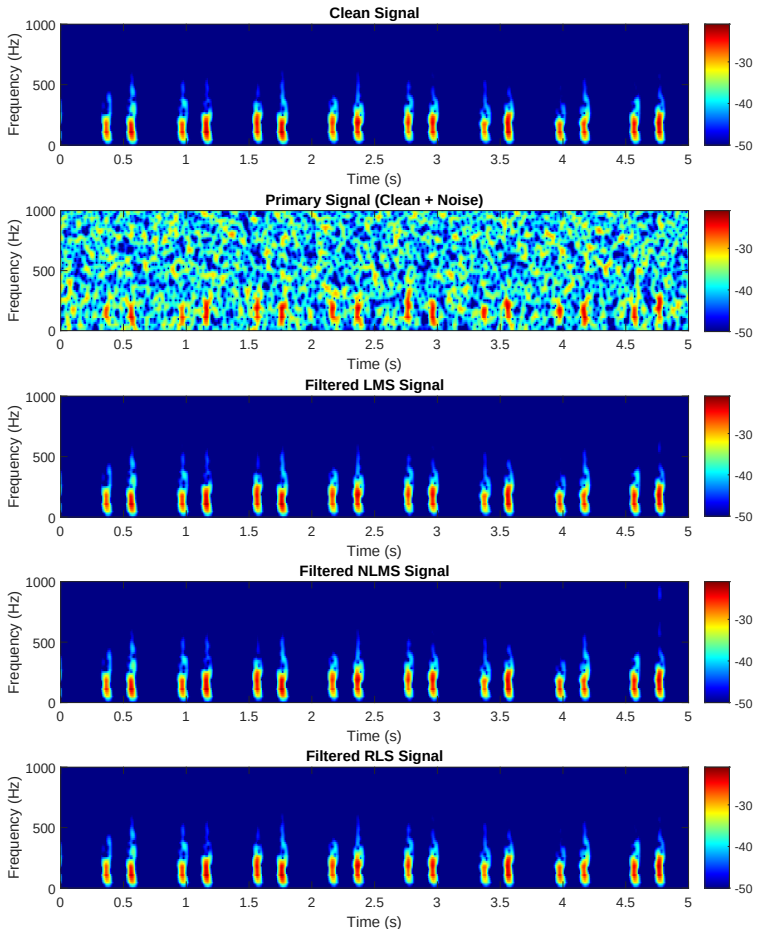


Figure 4.9: Mel Spectrogram - Synthetic Heartbeat.

The clean signal exhibits distinct, well-separated low-frequency heartbeat components. In the primary signal, added noise spreads across the entire frequency range, masking the heartbeat structure. Both the LMS and NLMS filters reduce the noise but leave a little residual artifact, particularly around the final heartbeat at 4.75 seconds above 500 Hz. In contrast, the RLS filter produces the cleanest mel spectrogram, effectively eliminating visible noise.

Figure 4.10 shows the MSE convergence for the LMS, NLMS, and RLS filters.

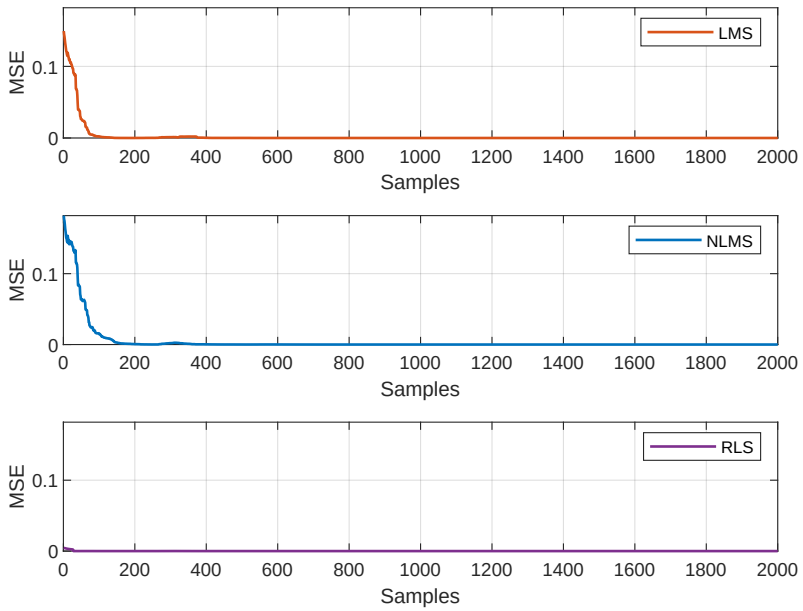


Figure 4.10: Error Convergence Curve - Synthetic Heartbeat.

The LMS and NLMS filters start with a high initial error, which rapidly decreases within the first 200 samples before gradually stabilising at a low MSE level. The RLS filter, in contrast, achieves near-zero error almost immediately with minimal fluctuations throughout, demonstrating its much faster convergence and higher precision.

The SNR values before and after filtering for each algorithm are summarised in the table 4.5:

Table 4.5: SNR improvement after adaptive filtering for Synthetic Heartbeat.

Filtering Method	Output SNR (dB)	SNR Improvment (dB)
No Filtering	-14.9449	-
LMS	16.6320	31.5769
NLMS	15.6893	30.6342
RLS	35.6963	50.6412

These controlled experiments facilitate the refinement of the adaptive filtering parameters before applying them to actual physiological recordings.

4.5 FastICA on Synthetic Heartbeat

This section evaluates the performance of the FastICA algorithm, specifically using deflation-based orthogonalisation, for estimating synthetic heartbeat from a controlled noise environment.

4.5.1 System Setup

In this experiment, the noise was created as square-modulated narrowband noise with a 2 Hz modulation rate, 50 Hz bandwidth centered at 500 Hz, and a sampling rate of 8 kHz. It was made by filtering white noise and turning it on and off with a square wave. Unlike normal Gaussian noise, which FastICA can't separate well, this modulated noise has a changing pattern that helps FastICA work by giving it the non-Gaussian structure it needs.

To simulate real-world sensor inputs, synthetic mixtures were digitally created by linearly mixing a clean heartbeat recording and the modulated noise at different ratios. This emulates the scenario where two sensors are placed at slightly different positions on the body, each capturing a distinct linear mixture of the original sources.

Below is an illustration of the experimental setup used in real-life conditions:

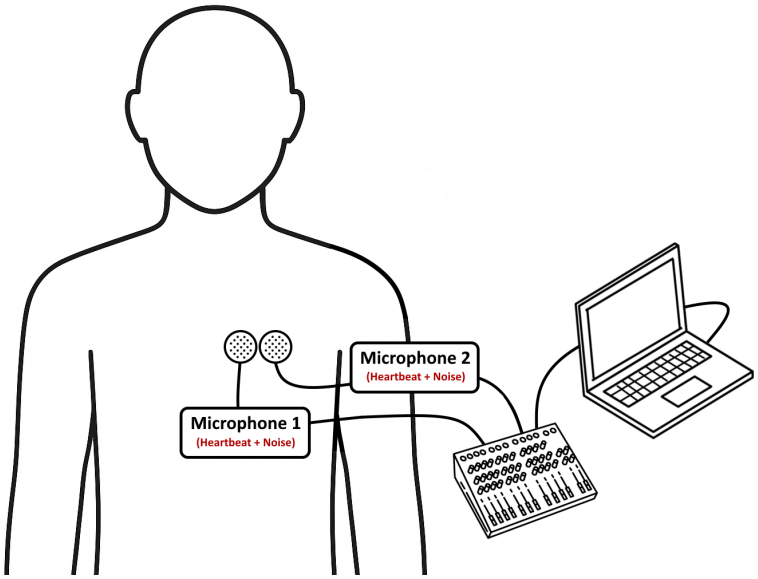


Figure 4.11: Experimental setup for heartbeat sound acquisition in noisy environments for the purpose of FastICA.

Let $\mathbf{s} = (s_1, s_2)^T$ represent the source vector, where s_1 is the clean heartbeat and s_2 is the noise component. The observed signal vector $\mathbf{x} = (x_1, x_2)^T$ is then constructed using a known mixing matrix $\mathbf{A} \in \mathbb{R}^{2\times 2}$: $\mathbf{x} = \mathbf{A}\mathbf{s}$ (4.3)

The specific mixing matrix used in this experiment is given in decibel (dB) scale as:

$$\mathbf{A}_{\text{dB}} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} -10\,\text{dB} & 0\,\text{dB} \\ -18\,\text{dB} & -5\,\text{dB} \end{bmatrix} \tag{4.4}$$

The linear scale values are obtained by converting from dB using the formula:

$$\mathbf{A} = 10^{\frac{\mathbf{A}_{\text{dB}}}{20}} = \begin{bmatrix} 10^{\frac{-10}{20}} & 10^{\frac{0}{20}} \\ 10^{\frac{-18}{20}} & 10^{\frac{-5}{20}} \end{bmatrix} = \begin{bmatrix} 0.3162 & 1 \\ 0.1259 & 0.5623 \end{bmatrix} \tag{4.5}$$

This corresponds to:

$$x_1 = A \cdot s_1 + B \cdot s_2 \tag{4.6}$$

$$x_2 = C \cdot s_1 + D \cdot s_2 \tag{4.7}$$

The signals $\mathbf{x}$ are then used as input to the FastICA algorithm, which attempts to estimate the original sources by learning a demixing matrix $\mathbf{W}$ such that:

$$\mathbf{y} = \mathbf{W}\mathbf{x} \tag{4.8}$$

where $\mathbf{y}$ is the estimated source signals (heartbeat and noise seperated).

This controlled approach allows for quantitative evaluation of the source separation performance, since the true sources are known and the clean heartbeat signal can be used as a reference for calculating signal-to-noise ratio (SNR) improvements.

4.5.2 Performance Evaluation

Figure A.27 shows a comparison between the clean signal, the noisy primary signal, and the error signal produced by the FastICA algorithm.

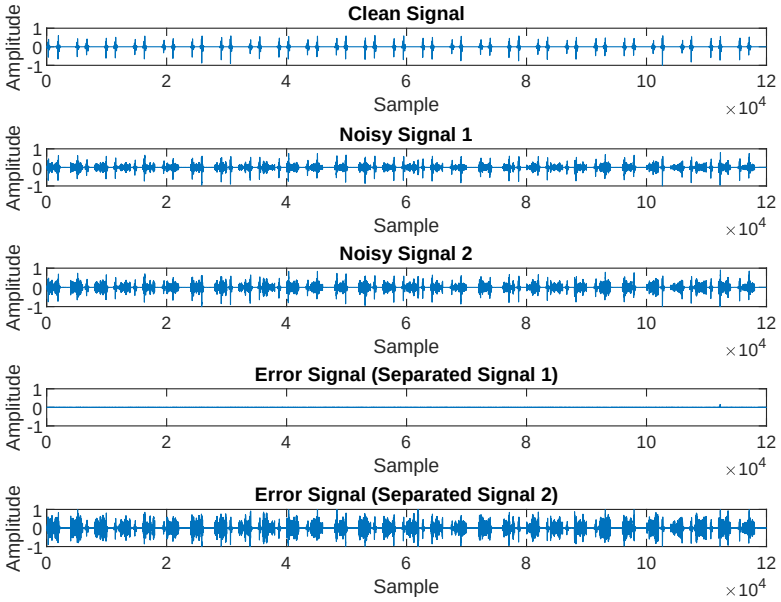


Figure 4.12: Error signals — Heartbeat signal corrupted by square modulated noise.

Separated Signal 1 represents the estimated heartbeat component with near-zero error plot. Separated Signal 2 represents the estimated noise component. FastICA successfully estimates the original source components from the mixed noisy inputs.

Figure 4.13 provides a visual analysis via Mel spectrograms, which help illustrate how much noise the FastICA algorithm removes across time and frequency.

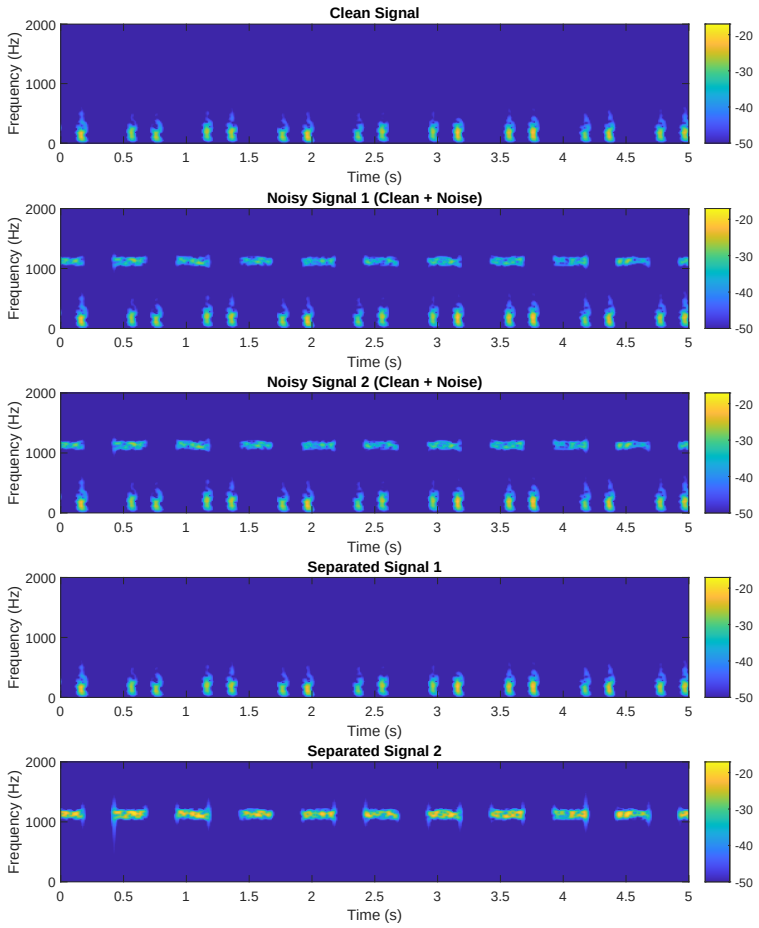


Figure 4.13: Mel spectrograms — Heartbeat signal in the presence of square modulated noise.

The clean signal shows distinct heartbeat patterns below 200 Hz, while the noisy signals exhibit energy—particularly around 1000 Hz. Separated Signal 1 successfully recovers the heartbeat, while Separated Signal 2 captures the noise.

Table 4.6 summarises the SNR improvements for FastICA.

Table 4.6: SNR improvement — Heartbeat signal with Noise.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-0.72	-
FastICA (Separated Signal 1)	37.94	38.66
FastICA (Separated Signal 2)	-6.74	-6.02

This controlled experiment confirms the correct implementation of FastICA, ensuring its reliability before applying it to real physiological data.

Results 5

The results show the performance of the adaptive noise cancellation system under realistic and challenging conditions. The objective is to verify that the system meets the functional and technical requirements outlined in Chapter 3. While initial testing was conducted using synthetic signals to validate baseline functionality, this stage focuses on real-world applicability by analysing real heartbeat recordings contaminated with different types of environmental and physiological noise.

5.1 Adaptive Filtering of Public Heart Sounds

To assess the effectiveness of the system in practical scenarios, normal heartbeat recordings are sourced from publicly available databases, while noise samples were provided by industry partners at Ai Health Highway India Pvt Ltd [24, 5]. These recordings simulate various clinical and pre-hospital conditions in which noise significantly degrades signal quality. The aim is to evaluate how well the LMS, NLMS, and RLS adaptive filters perform in estimating clean heartbeat signals from these noisy recordings.

The evaluation focuses on four representative noise types commonly encountered in real clinical and field settings:

1. **Artifacts** — Transient noises caused by patient movement, clothing friction, or stethoscope manipulation.
2. **Ambulance & Traffic** — Sirens, engine noise, and road vibrations typical in pre-hospital emergency response environments.
3. **Conversation** — Overlapping speech or staff communication during auscultation procedures.

4. **Hospital Ambient Noises** — Background sounds in a hospital ward such as equipment beeps, footsteps, and ventilation systems.

Each noise type is evaluated individually to understand its specific impact on the adaptive filter performance. The following section presents detailed results for the Artifacts noise type, including parameter configurations, spectrogram analysis, error convergence behavior, and quantitative SNR metrics. Results for the remaining noise types are summarised in Table 5.3, with full details provided in the appendix.

5.1.1 Artifacts

This test case evaluates the system's ability to denoise a heartbeat signal contaminated with typical body movement artifacts, such as stethoscope handling noise and clothing friction. Table 5.1 outlines the adaptive filter parameters optimised for this scenario using grid search.

Table 5.1: Tuning parameters for adaptive filters — Artifacts.

Parameter	**Value**
Sampling rate	16 kHz
Filter length (LMS)	12
Filter length (NLMS)	8
Filter length (RLS)	12
Step size (LMS)	0.001
Step size (NLMS)	0.001
Forgetting factor (RLS)	0.9999

Figure 5.1 shows a comparison between the clean signal, the noisy primary signal, and the error signals produced by the LMS, NLMS, and RLS filters.

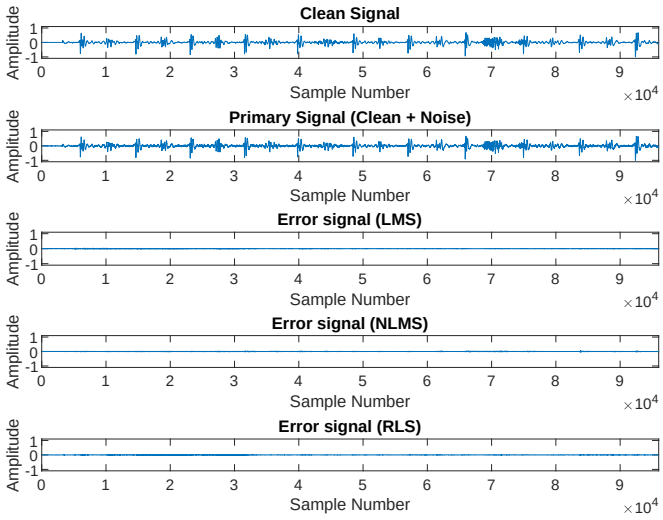


Figure 5.1: Error signals - Heartbeat signal corrupted with Artifacts.

All three filters substantially reduce the artifact noise showing near zero error signal plots.

Figure 5.2 illustrates how the output SNR varies with filter length across all three algorithms.

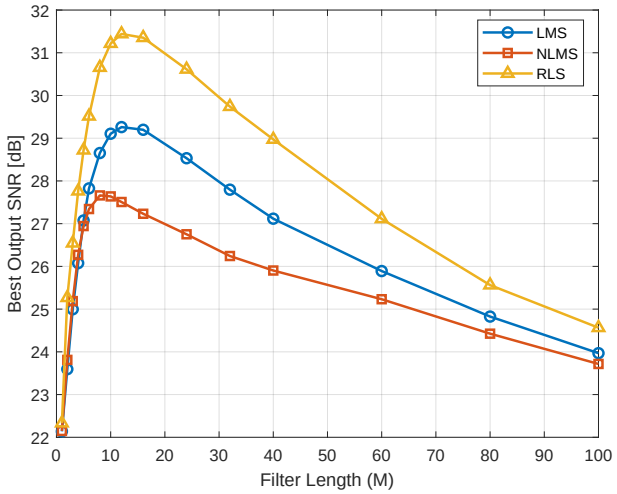


Figure 5.2: Output SNR vs Filter Length - Heartbeat signal corrupted with Artifacts.

The RLS filter exhibits the best overall performance, achieving the highest SNR values across most filter lengths. Its peak performance occurs at a filter length of approximately

10, where it reaches about 31.5 dB, indicating optimal efficiency with shorter filter configurations. Beyond this peak, the SNR gradually declines as the filter length increases, dropping to roughly 24.5 dB at a length of 100.

The LMS filter follows a similar trend in its SNR curve but attains a lower maximum SNR of around 29 dB at a filter length near 10.

Finally, the NLMS filter also shows a comparable shape in performance but peaks at a significantly lower SNR of about 26 dB around a filter length of 9, making it the least effective among the three.

The spectrogram in figure 5.3 provides a time-frequency view of how effectively each algorithm removes noise from the signal.

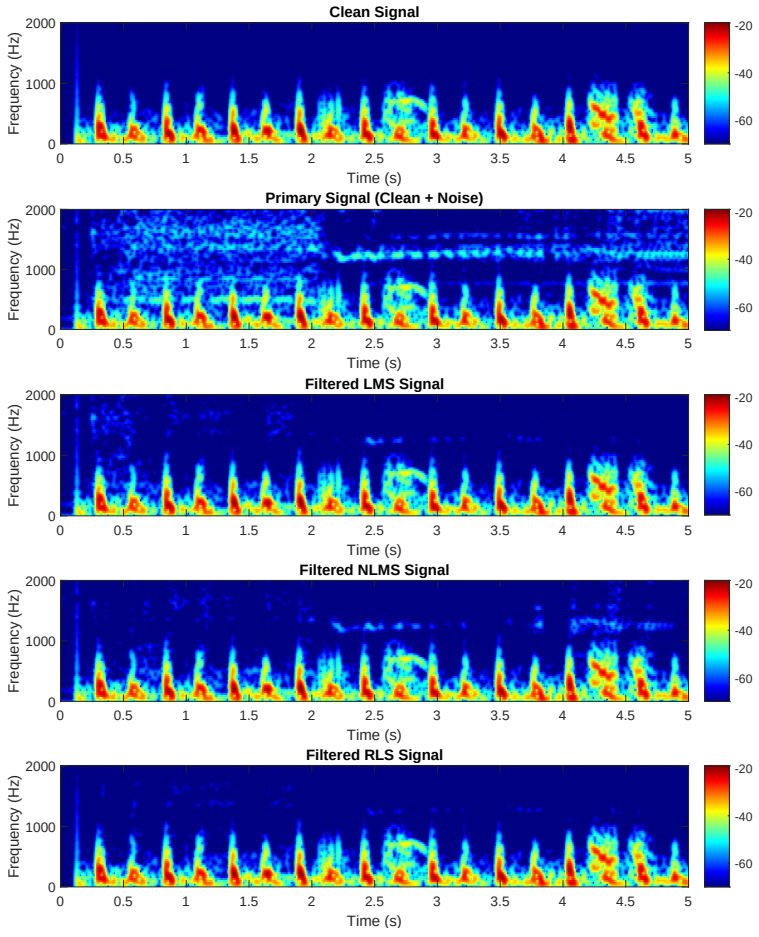


Figure 5.3: Mel Spectrogram - Heartbeat signal corrupted with Artifacts.

The clean signal exhibits periodic low-frequency components characteristic of heartbeats. The primary signal, contaminated by artifacts, shows substantial spectral energy across a wide frequency range. All three adaptive filters reduce much of the mid- to high-frequency noise. The RLS filter produces the cleanest spectrogram, leaving minimal noise above 1000 Hz.

Figure 5.4 presents the convergence of MSE over number of samples, demonstrating how quickly and effectively each algorithm adapts.

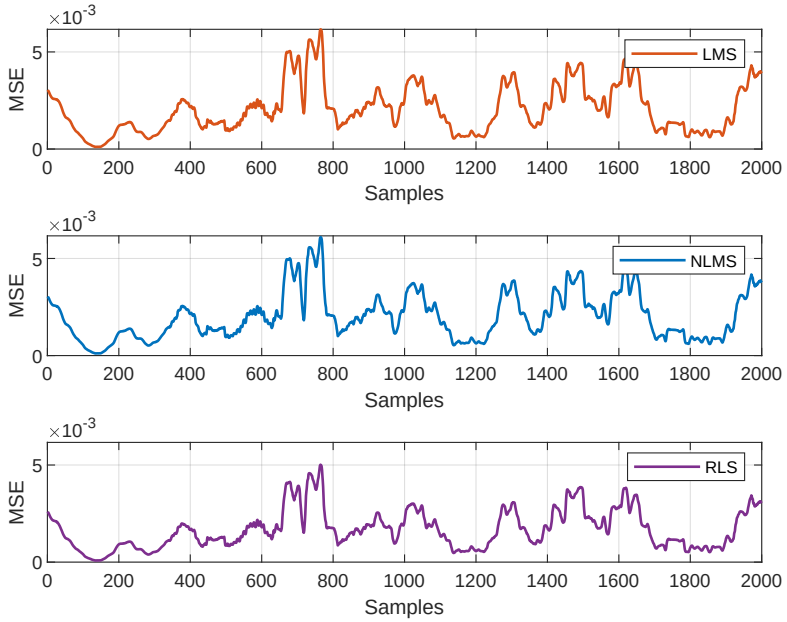


Figure 5.4: Error Convergence Curve - Heartbeat with Artifacts.

From a practical standpoint, the MSE values remain mostly below 5×10^{-3} across the entire duration—indicating that all three algorithms achieve excellent performance. To better illustrate the scale of the fluctuations across the samples, a zoomed-out view is provided in figure 5.5.

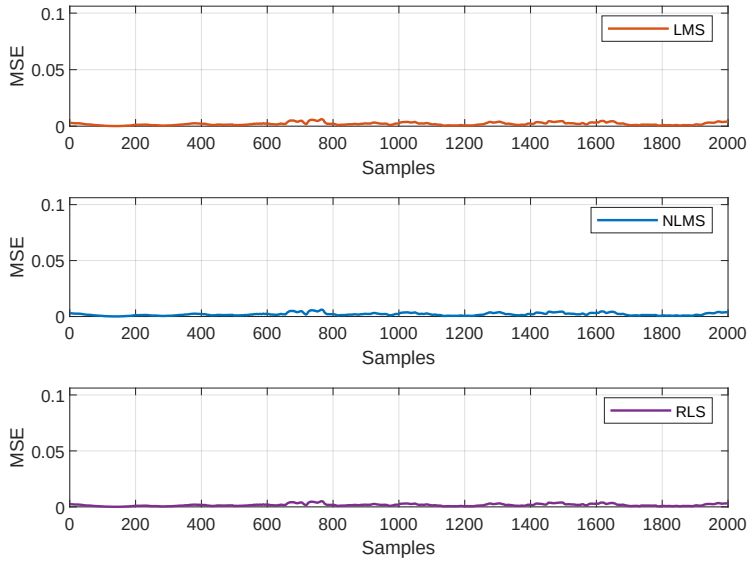


Figure 5.5: Zoomed-out error convergence curves for heartbeat signal with artifacts.

As shown, the error curves appear as nearly flat lines approaching zero, clearly demonstrating rapid convergence to a steady state. Although the RLS algorithm shows slightly better performance, all three methods effectively suppress the artifacts from the heartbeat signal.

Table 5.2 shows the SNR improvements for each method.

Table 5.2: SNR improvement — Heartbeat with Artifacts.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-3.87	-
LMS	29.26	33.13
NLMS	27.66	31.53
RLS	31.44	35.31

Based on the results in table 5.2, the RLS filter achieves the highest output SNR and SNR improvement, making it the most effective method for artifact removal in this case.

5.1.2 Performance Across Different Noise Types

While the detailed analysis of the Artifacts noise type demonstrates the effectiveness of adaptive filtering in removing patient movement and equipment handling noise, it is

important to examine how the system performs across all noise environments. Table 5.3 provides a comprehensive summary of SNR improvements achieved by each algorithm across all four noise types.

Table 5.3: Performance summary across all noise types (best performer in bold).

Noise Type	Input SNR (dB)	Output SNR (dB)		
		LMS	NLMS	RLS
Artifacts	-3.87	29.26	27.66	**31.44**
Ambulance & Traffic	-4.64	19.75	18.32	**30.33**
Conversation	-0.03	37.86	37.85	**44.21**
Hospital Ambient	-0.26	29.44	**30.46**	30.27

As evident from Table 5.3, the adaptive filtering system effectively enhances heartbeat signals across all noise environments tested. The RLS algorithm consistently outperforms both LMS and NLMS in three out of four noise scenarios, with particular strength in handling conversation noise where it achieves an exceptional 44.21 dB output SNR. For hospital ambient noise, the NLMS filter slightly outperforms the others, achieving 30.46 dB output SNR.

The most challenging noise environment appears to be the ambulance and traffic scenario, where input SNR is lowest at -4.64 dB. Even in this demanding situation, the RLS filter achieves a remarkable 30.33 dB output SNR, demonstrating the system's robustness in emergency transport conditions.

These results confirm that adaptive filtering provides an effective solution for heart sound denoising across diverse clinical environments. The detailed analysis for the remaining noise types can be found in Appendix A.2.

5.2 General Adaptive Filter

This section examines a general-purpose adaptive filter configuration with fixed parameters for heart sound denoising across diverse noise environments. The objective is to establish a single robust parameter set achieving consistent SNR improvement of at least 20 dB across all tested noise scenarios without requiring type-specific tuning.

5.2.1 Fixed Hyperparameter Selection

Performance analysis across both experimental and public heart sound datasets yielded the following optimal general parameter configuration:

Table 5.4: General adaptive filter parameters applicable across noise types.

Parameter	Value
Sampling rate	16 kHz
Filter length (LMS)	8
Filter length (NLMS)	8
Filter length (RLS)	8
Step size (LMS)	0.1
Step size (NLMS)	0.01
Forgetting factor (RLS)	0.9999

These parameters represent a balance between computational efficiency and consistent performance across varied noise characteristics. The moderate filter length of 8 provided optimal results across most scenarios, while the selected step sizes and forgetting factor demonstrated stability across different signal-to-noise ratios.

5.2.2 Performance on Experimental Recordings

Table 5.5 presents a comparison of fixed versus tuned filter performance across all noise types in the experimental heart sound recordings:

Table 5.5: Fixed vs. tuned parameters performance (Output SNR in dB) on experimental recordings.

Noise Type	Input SNR (dB)	LMS		NLMS		RLS	
		Fixed	Tuned	Fixed	Tuned	Fixed	Tuned
Ambient Noise	-8.82	5.10	13.98	6.96	14.02	25.56	26.23
Baby Crying	-8.90	20.22	22.84	18.12	23.53	44.91	46.26
Drilling	-0.23	22.09	26.15	21.74	27.43	36.18	41.43
Hammering	-1.46	21.92	25.63	19.74	26.91	43.26	45.58
Speech	-9.30	9.55	15.28	12.58	17.64	36.45	38.37
Average		15.78	20.78	15.83	21.91	37.27	39.57

The experimental results reveal that the RLS algorithm using fixed parameters consistently achieves SNR improvements exceeding 25 dB across all noise types, including the

challenging ambient noise and speech scenarios. LMS and NLMS algorithms show more significant performance variation between fixed and tuned parameters, with average differences of 5.00 dB and 6.08 dB respectively. The RLS algorithm demonstrates the smallest average difference (2.30 dB), indicating its inherent robustness with fixed parameters across diverse noise environments. For ambient noise and speech scenarios, LMS and NLMS with fixed parameters fall substantially short of the RLS performance, highlighting the limitations of simpler algorithms with generalised parameters.

5.2.3 Performance on Public Heart Sound Dataset

Extending the analysis to the public heart sound dataset with diverse clinical noise types yields additional insights:

Table 5.6: Fixed vs. tuned parameters performance (Output SNR in dB) on public heart sound dataset.

Noise Type	Input SNR (dB)	LMS		NLMS		RLS	
		Fixed	Tuned	Fixed	Tuned	Fixed	Tuned
Artifacts	-3.87	25.50	29.26	25.49	27.66	32.12	31.44
Ambulance & Traffic	-4.64	16.69	19.75	16.65	18.31	18.29	30.33
Conversation	-0.03	36.95	37.86	36.96	37.85	36.03	44.21
Hospital Ambient	-0.26	20.67	29.44	23.44	30.46	32.50	30.27
Average		24.95	29.08	25.64	28.57	29.74	34.06

For the public dataset, the RLS algorithm with fixed parameters yields exceptional performance in artifacts, conversation, and hospital ambient noise scenarios, even outperforming tuned parameters in the hospital ambient case. The most challenging scenario for all algorithms with fixed parameters is ambulance and traffic noise, where only parameter tuning enables the RLS algorithm to exceed 30 dB SNR improvement. Unlike in the experimental recordings, the RLS algorithm shows similar average differences between fixed and tuned parameters (4.32 dB) as the LMS algorithm (4.13 dB), suggesting certain real-world clinical noises benefit more from customised parameter selection.

5.2.4 Combined Performance Analysis

To evaluate the overall effectiveness of the general adaptive filter parameters, Table 5.7 presents the aggregated performance metrics across both datasets:

Table 5.7: Consolidated performance analysis across both datasets (Output SNR in dB).

Dataset	LMS		NLMS		RLS	
	Fixed	**Tuned**	**Fixed**	**Tuned**	**Fixed**	**Tuned**
Experimental (Avg. SNR)	15.78	20.78	15.83	21.91	37.27	39.57
Public (Avg. SNR)	24.95	29.08	25.64	28.57	29.74	34.06
Overall Avg. SNR	20.37	24.93	20.74	25.24	33.51	36.82
% of Tuned Performance	81.7%		82.2%		91.0%	

The RLS algorithm with fixed parameters achieves 91.0% of the performance possible with optimal tuning, significantly outperforming LMS (81.7%) and NLMS (82.2%) in this regard. While LMS and NLMS with fixed parameters deliver acceptable average performance (20.37 dB and 20.74 dB respectively), they fail to meet the 20 dB SNR improvement for several individual noise scenarios. The RLS algorithm with fixed parameters consistently exceeds the target minimum performance across most scenarios, with only the ambulance and traffic noise in the public dataset presenting a significant challenge.

Based on these results, the RLS algorithm with filter length 8 and forgetting factor 0.9999 provides the most robust general-purpose configuration for heart sound denoising across diverse noise environments. This configuration offers an optimal balance between performance consistency and implementation simplicity, eliminating the need for scenario-specific parameter tuning while delivering near-optimal results in most cases.

5.3 FastICA for Heart Sound Data

This section evaluates the performance of the FastICA algorithm, employing deflation-based orthogonalisation to estimate clean heart sound signals from noisy recordings. We analyse both publicly available data and experimental recordings to demonstrate the algorithm's effectiveness.

5.3.1 Digitally created mixture of heart and artifact sound

This test case evaluates the system's ability to denoise a heartbeat signal contaminated with typical body movement artifacts, such as stethoscope handling noise and clothing friction. The required mixtures were digitally generated by combining the clean heartbeat recordings with noise signals at different mixing ratios, simulating the effect of having two

sensors placed at slightly different locations, each capturing a distinct linear mixture of the underlying source with no delay.

The spectrogram in figure 5.6 provides a time-frequency view of how effectively the FastICA algorithm removes noise from the signal.

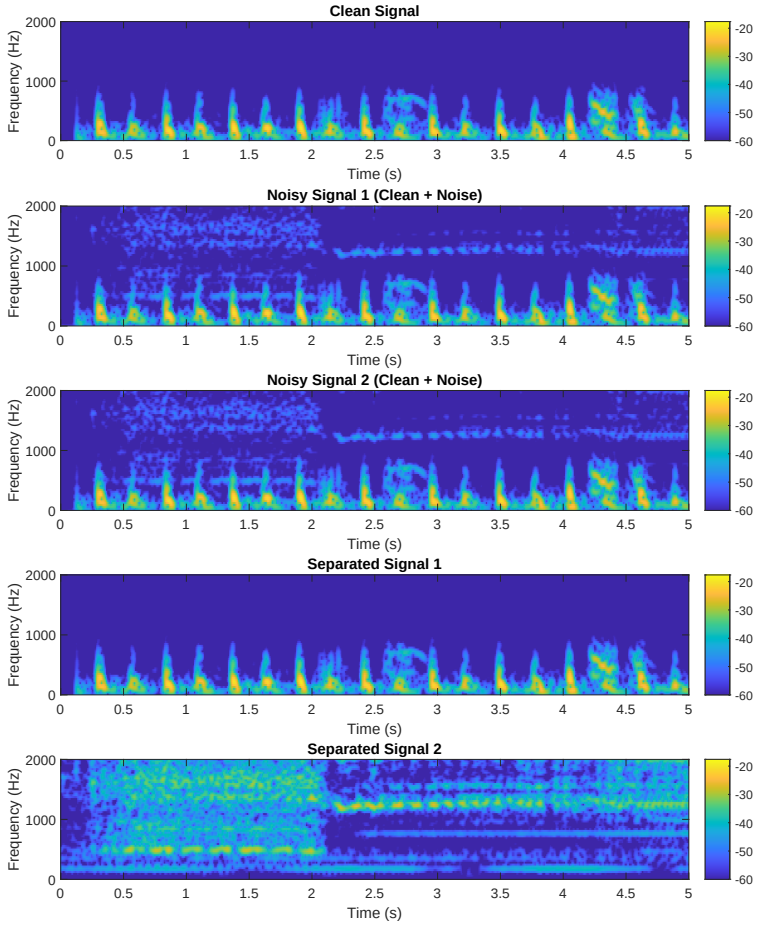


Figure 5.6: Mel Spectrograms showing FastICA estimation of heartbeat signal corrupted with artifacts.

The clean signal exhibits clear, periodic low-frequency components characteristic of heartbeats. Noisy signals display the heartbeats corrupted by artifacts. Separated Signal 1 closely matches the original clean heartbeat pattern, demonstrating successful noise removal. Separated Signal 2 isolates the artifacts. This visual representation confirms FastICA's ability to accurately estimate heartbeat sound from mixed recordings.

Table 5.8 summarises the SNR improvements achieved for each noise type across public datasets.

Table 5.8: Comprehensive SNR improvement summary for FastICA across different noise types.

Noise Type	Input SNR (dB)	Output SNR (dB)	SNR Improvement (dB)
Artifacts	-1.40	47.20	48.60
Ambulance & Traffic	-9.05	44.85	53.90
Conversation	-15.91	38.42	54.33
Hospital Ambient	-8.68	41.78	50.46
Average	-8.76	43.56	51.82

Detailed analyses for the ambulance & traffic, conversation, and hospital ambient noise are provided in Appendix A.4.

5.3.2 Simulating real world by added delay

Using the same digital mixture of heart and artifact sounds, a 10 ms delay is introduced between Noisy Signal 1 and Noisy Signal 2.

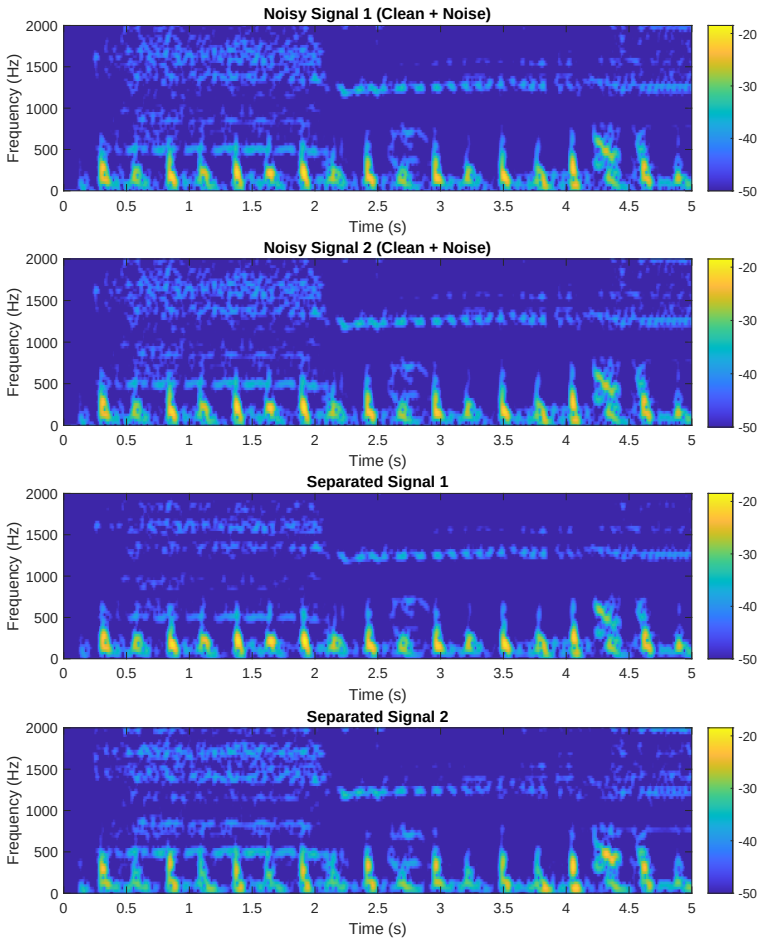


Figure 5.7: Mel Spectrograms showing FastICA estimation of heartbeat signal corrupted with artifacts with 10 ms delay between noisy 1 and 2

The spectrograms show that the fastICA fails in separating the components, because FastICA cannot track time-varying mixtures effectively.

5.3.3 Real world recording

FastICA was tested on real-world recordings in which speech interference corrupts the heartbeat signal. Two microphones were placed on the chest—one positioned close to the heart, and the other slightly farther away, capturing the heartbeat at a lower gain.

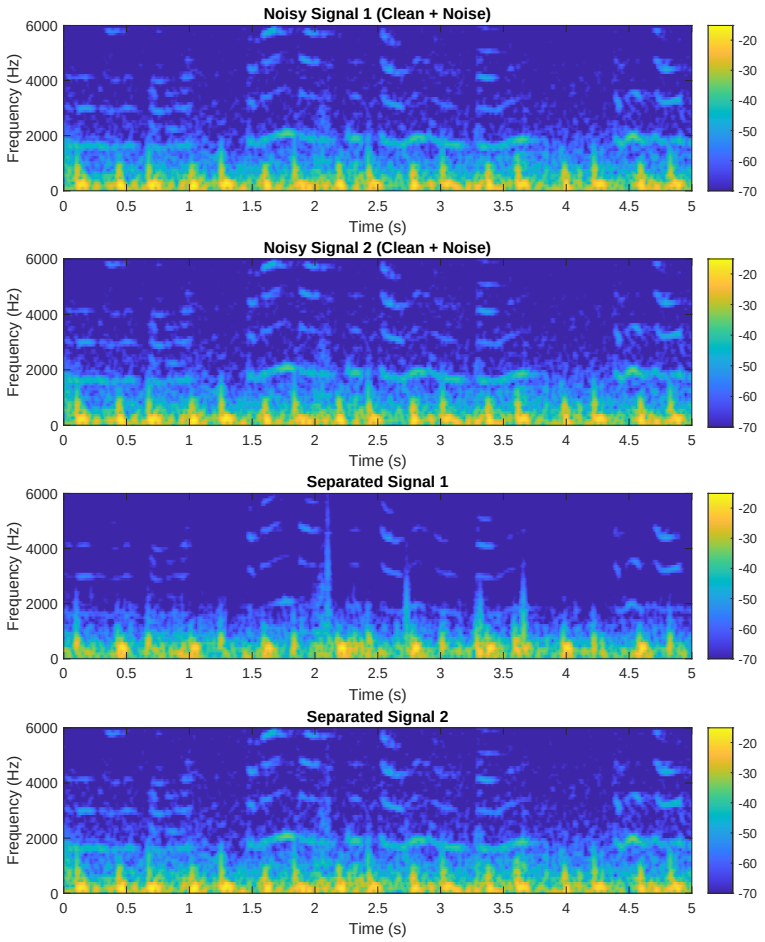


Figure 5.8: Mel Spectrograms showing FastICA estimation of heartbeat signal corrupted with speech from real world.

The spectrograms show that the fastICA fails in separating components. Compared to the digitally mixed case, highlighting the increased complexity of real-world audio separation. One reason it doesn't work is because two identical microphones capturing the same audio will still have a difference in real life as shown in figure 5.9.

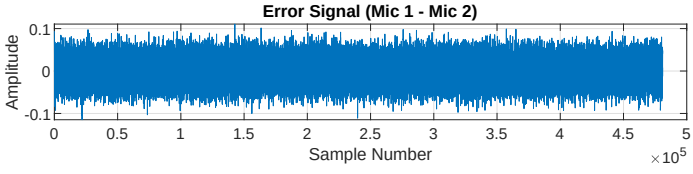


Figure 5.9: Waveform of mic 1 and mic 2 capturing ambient noise subtracted from each other still leaves us with noise.

Even though the error signal should be zero in theory as they are capturing the same ambient noise. In real-world settings two separate mics rarely capture perfectly synchronised audio due to slight hardware differences, clock drift, and unsynchronised recording starts, making precise alignment and separation not possible with FastICA.

5.4 Denoising Methods on Lung Sound Data

To assess the performance of different noise reduction techniques on lung sound data, recordings were obtained from publicly available databases, while noise samples were provided by industry partners at Ai Health Highway India Pvt Ltd [24, 5]. In this experiment, only hospital ambient noise is considered, including machinery beeps, footsteps, and background conversations. The required signals for FastICA were digitally generated by combining the clean heartbeat recordings with noise signals at different mixing ratios.

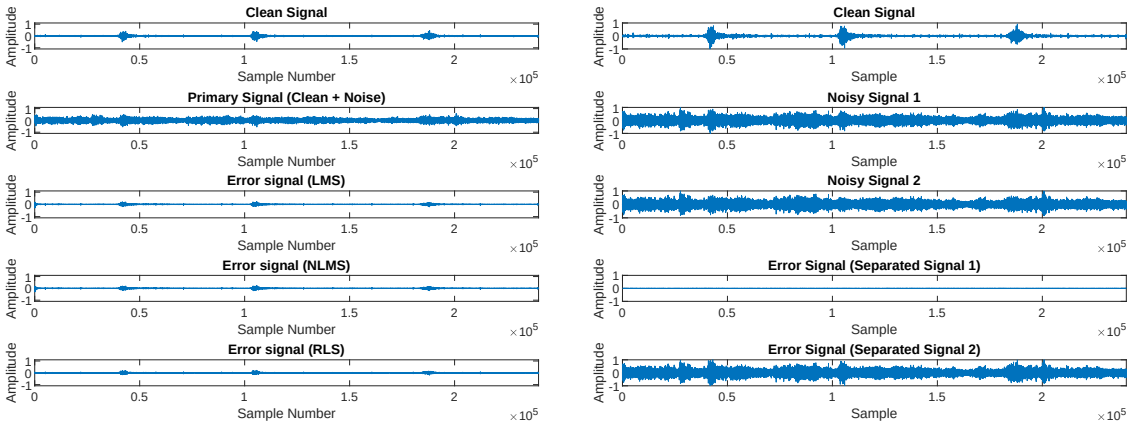
5.4.1 Hospital Ambient Noise

Both adaptive filtering and FastICA were applied to isolate lung sounds consisting of three consecutive inhalations corrupted by hospital ambient noise. For the adaptive filtering approach, parameters were optimised using grid search, as summarised in Table 5.9.

Table 5.9: Tuning parameters for adaptive filters — Lung Sounds.

Parameter	Value
Sampling rate	16 kHz
Filter length (LMS)	24
Filter length (NLMS)	24
Filter length (RLS)	32
Step size (LMS)	0.004
Step size (NLMS)	0.0015
Forgetting factor (RLS)	0.9998

Figure 5.10 presents a side-by-side comparison of the error signals produced by both methods.



(a) Adaptive filtering methods **(b)** FastICA method

Figure 5.10: Comparison of error signals — Lung sound corrupted by Hospital Ambient Noise.

In the adaptive filtering approach (left), all three filters (LMS, NLMS, and RLS) significantly reduce hospital ambient noise, with the RLS filter showing the cleanest error signal. For the FastICA approach (right), the algorithm successfully estimates the original source components from the mixed noisy inputs, with Separated Signal 1 representing the estimated lung component with near-zero error and Separated Signal 2 representing the estimated noise component.

Figure 5.11 illustrates how the output SNR varies with filter length for each adaptive filtering algorithm.

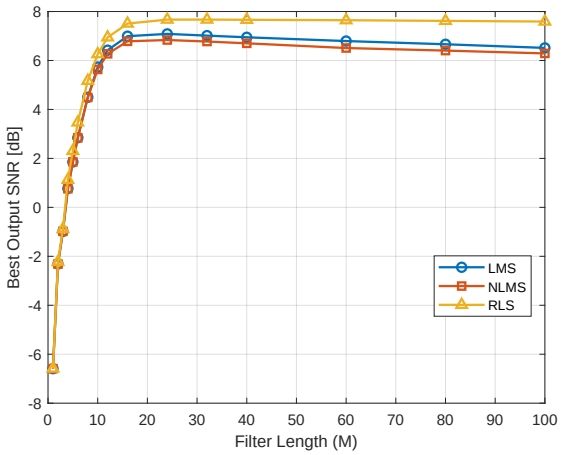
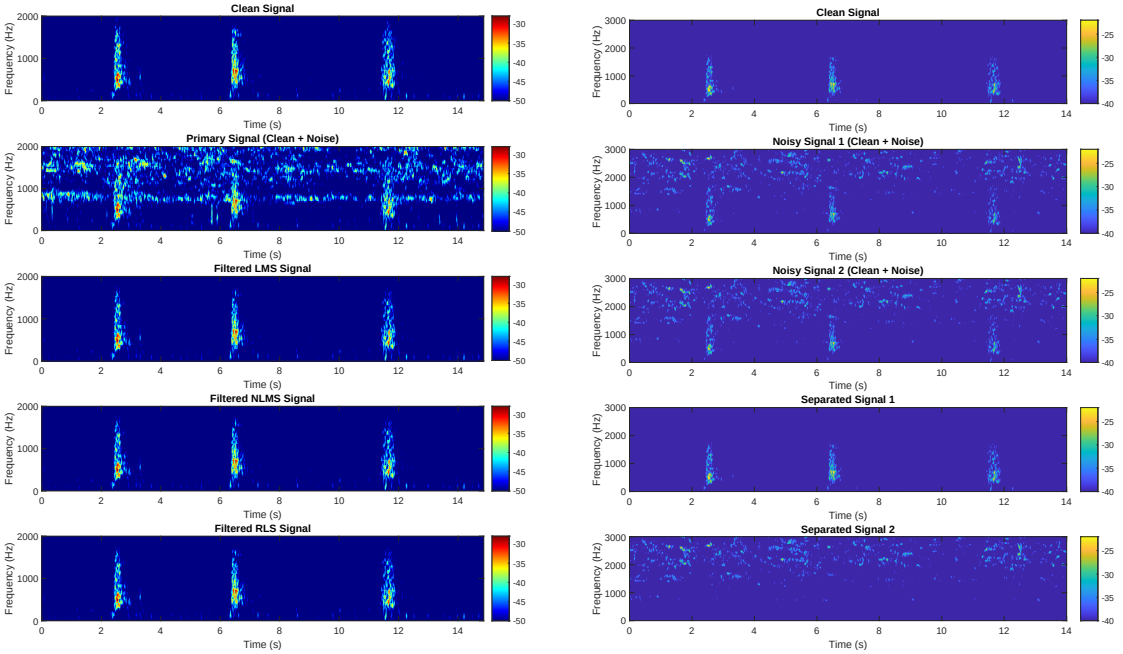


Figure 5.11: Output SNR vs Filter Length — Adaptive filtering of lung sounds with Hospital Ambient Noise.

The RLS filter consistently outperforms LMS and NLMS across all filter lengths, maintaining a 1-1.5 dB advantage throughout. All three filters show rapid SNR improvement from negative to positive values as filter length increases from 0 to 15, with RLS peaking at approximately 7.8 dB around filter lengths 30-40. LMS and NLMS perform almost identically, reaching maximum SNR values of about 6.5-7 dB at filter lengths 15-20.

Mel spectrograms provide time-frequency representations of both methods' ability to attenuate background noise while preserving lung sound characteristics.



(a) Adaptive filtering methods **(b)** FastICA method

Figure 5.12: Mel spectrograms of lung sounds before and after noise reduction using adaptive filtering and FastICA.

In both methods, the clean signal displays three distinct inhalations occurring at approximately 2.5, 6.5, and 11.5 seconds, with lung sound components concentrated primarily in the 400-1400 Hz frequency range. Both approaches successfully recover the original signal structure, significantly reducing noise across the spectrum. In the adaptive filtering approach, there are no obvious performance differences between the three algorithms based on these spectrograms. For the FastICA approach, Separated Signal 1 closely matches the original clean lung signal, while Separated Signal 2 effectively isolates

the hospital ambient noise.

5.4.2 Performance Comparison

Table 5.10 presents a comprehensive comparison of SNR improvements achieved by all methods.

Table 5.10: SNR improvement comparison — Lung sounds with hospital ambient noise.

Method	Initial SNR (dB)	Output SNR (dB)	SNR Improvement (dB)
		Adaptive Filtering	
No Filtering	-14.46		-
LMS	-14.46	7.09	21.55
NLMS	-14.46	6.84	21.30
RLS	-14.46	7.67	22.13
		FastICA	
No Filtering	-20.37		-
FastICA	-20.37	30.05	50.42

FastICA and adaptive filtering both effectively reduce hospital ambient noise in lung sound recordings. Among adaptive methods, RLS performs best with a 25.11 dB SNR improvement, followed by NLMS (24.84 dB) and LMS (23.17 dB). FastICA achieved an SNR improvement of 50.42 dB.

5.4.3 Fixed Hyperparameters

To evaluate the practicality of using a general adaptive filter configuration, fixed hyperparameter settings were tested on lung sound recordings contaminated with hospital ambient noise. The goal was to assess how well these fixed configurations perform compared to their individually tuned counterparts.

Table 5.11 presents a comparison of output SNR values (in dB) achieved by LMS, NLMS, and RLS algorithms under both fixed and tuned hyperparameter settings.

Table 5.11: Comparison of output SNR (in dB) for fixed vs. tuned hyperparameters under hospital ambient noise. The last row shows the percentage of tuned performance retained by the fixed configuration.

Noise Type	**LMS**		**NLMS**		**RLS**	
	Fixed	**Tuned**	**Fixed**	**Tuned**	**Fixed**	**Tuned**
Hospital ambient noise	3.96	7.09	4.46	6.84	5.12	7.67
% of Tuned Performance	55.85%		65.20%		66.73%	

The final row shows the performance of the fixed configuration as a percentage of the tuned performance, highlighting how close each method comes to its optimal configuration.

Discussion 6

This chapter critically evaluates the results presented in Chapter 5 against the requirements outlined in Chapter 3. The discussion examines the effectiveness of the implemented signal enhancement techniques, considers their limitations, and explores the theoretical and practical implications for clinical applications.

6.1 Evaluation of Results Against Requirements

Adaptive filtering with fixed parameters and FastICA both surpassed the **20 dB SNR improvement requirement**. RLS delivered the best results, averaging around 34 dB due to its fast convergence and tracking ability. LMS and NLMS showed moderate performance, averaging 20.37 dB and 20.74 dB respectively, but failed to meet the requirement under noise conditions like ambient noise and speech.

While tuned parameters achieved optimal performance for each specific noise type through grid search optimisation, this approach is impractical for real-world deployment as it requires prior knowledge of the noise characteristics and computational resources for parameter optimisation. The fixed parameter analysis revealed that RLS maintained 91.0% of its optimal tuned performance, significantly outperforming LMS (81.7%) and NLMS (82.2%) in robustness. This finding is crucial for practical deployment, as it eliminates the need for scenario-specific parameter tuning and makes the system usable without expert knowledge.

FastICA achieved an average SNR improvement of 51.82 dB across the four tested noise types when applied to digitally mixed signals. This highlights FastICA's theoretical strength in exploiting statistical independence for blind source separation. However, FastICA's practical applicability was limited when tested on real recordings where it failed to produce meaningful results. Potential reasons for this discrepancy are discussed in Section 6.3.

Although the primary focus was heart sound enhancement, early tests on lung sounds showed promising results. Adaptive filtering and FastICA improved lung sound quality corrupted by hospital ambient noise. The RLS algorithm achieved a 22.13 dB SNR improvement, while LMS and NLMS achieved 21.55 dB and 21.30 dB, respectively. Fixed-parameter adaptive filters still provided moderate noise reduction, retaining 55.85%, 65.20%, and 66.73% of the tuned performance for LMS, NLMS, and RLS. However, these fixed parameters were originally tuned for heart sounds, not lung sounds. Ideally, lung sound enhancement should be guided by dedicated tuning experiments because of their different characteristics. Due to time constraints, this was not done here, so future work should optimize parameters specifically for lung sounds.

FastICA achieved an SNR improvement of 50.42 dB in the digitally mixed lung sound scenario, though as with the heart sounds this result was not replicated in real-world recordings.

6.2 Adaptive Filtering: Theory vs. Practice

Among the adaptive filtering algorithms tested, RLS consistently demonstrated superior performance but at the cost of higher computational complexity. This trade-off is important to consider in practical applications, especially for portable or battery-powered devices. The LMS algorithm, while simpler and less computationally intensive, provided reasonable performance in many scenarios and might be preferable in resource-constrained implementations.

The NLMS algorithm, which normalises the step size by the power of the input signal, did not consistently outperform standard LMS despite its theoretical advantages in handling signals with varying power levels.

A likely mistake occurred in the preprocessing steps, such as normalising the input signals before filtering, because this removed the natural power differences between signals that the NLMS filter relies on to adjust properly. Therefore, normalisation should have been reserved only for the plotting stage to make the data visually comparable without affecting the filter's operation.

6.3 FastICA: Theory vs. Practice

In theory FastICA works well in separating heart/lung signals from noise. However, FastICA does have some limitations that are important to keep in mind. One issue is that it can't figure out the exact scale or order of the separated components on its own. This means some extra post-processing is needed to figure out which component actually corresponds to the heart/lung sound.

Another assumption FastICA makes is that the mixing process, how sources combine in the microphones, is linear and stationary meaning it doesn't change over time [26, 27]. Testing revealed that even a 10 ms mismatch between input signals was enough to cause the algorithm to break down, proving its sensitivity to time alignment.

This is not an issue when combining the signals digitally but in practice two identical mics rarely capture perfectly synchronised audio due to slight hardware differences, clock drift, and unsynchronised recording starts.

On top of that, real environments are always changing. People move around, background noise changes, and rooms affect how sound travels. Even if nothing is moving, sound still hits each mic differently because of things like walls, echoes, and where the mics are placed. This makes the mixing non-stationary, which breaks one of the main things FastICA relies on [26, 27].

To sum up, while FastICA has strong theoretical foundations, it lacks robustness in real-world applications. This isn't too surprising, since the field of source separation has come a long way since FastICA was introduced back in the late '90s [15, 28]. These days, deep learning methods have taken the lead. They are better at handling the messy, unpredictable nature of real-life sound environments and generally perform more reliably than older approaches like ICA [29].

6.4 Comparison of Methods

Both adaptive filtering and FastICA can enhance heart and lung sounds, but adaptive filters are currently the better choice for practical use. RLS in particular offers strong performance with known trade-offs in computational cost. LMS and NLMS are simpler but still effective in many cases.

FastICA, while powerful in theory and working well on digitally mixed signals, did not perform well with real recordings due to challenges like signal misalignment and changing acoustic conditions. Because of this, adaptive filtering remains the preferred method until FastICA or other ICA variants can be adapted for real-life scenarios.

In summary, adaptive filtering is currently more reliable and easier to implement, making it the practical option for heart and lung sound enhancement in clinical settings.

6.5 Limitations and Future Work

Even though the results were encouraging, there are a few important limitations that should be pointed out. First, most of the evaluation was based on SNR values. While SNR is a helpful way to measure signal quality from a technical point of view, it doesn't necessarily reflect how the signals sound to a human listener or whether they're still useful for medical diagnosis. It would be a good idea for future work to include some kind of feedback involving experts in the medical field to check if the important features of the heart or lung sounds are still there after processing.

Another limitation is that this project focused only on normal heart sounds. This choice made sense given the available data and the goal of avoiding false negatives in screening, but it also means we don't yet know how well the methods would work on abnormal heart sounds. Early tests with normal lung sounds showed potential, so it's definitely worth exploring further.

The current setup doesn't support real-time processing, which would be really important for clinical or mobile applications. Going forward, it would make sense to work on optimising the code for real-time use and trying it out on portable devices, like the ones that might be used in clinics or in the field.

When talking about real time processing looking into the computational complexity is another important factor, especially when comparing filtering algorithms, as it affects how suitable each algorithm is for real-time or embedded applications.

Furthermore, while SNR served as the main performance requirement in this project, more requirements could have been added to get a fuller understanding of how the system performs. For instance, the MSE provides a measure of how close the filtered signal is to the desired one, and the convergence rate shows how quickly the algorithm adapts to

changes in the environment both of which are valuable in real-time or rapidly changing conditions. Because of limited time, these additional requirements were not included in the evaluation. However, incorporating them in future work would help make the system more reliable, especially for real-world use in medical settings.

Conclusion 7

Adaptive filtering has been investigated for its ability to enhance heart and lung sound recordings in both clinical and non-clinical noisy environments. The motivation stemmed from the need to support heart sound screening in resource-constrained settings, such as rural India, where environmental and equipment-related noise often compromises recording quality.

The problem statement to address was as following: “How can adaptive noise cancellation techniques improve heart and lung sound recordings in clinical and non-clinical environments?” Results showed that among the LMS, NLMS, and RLS adaptive filtering methods, all were effective in reducing noise and improving signal clarity, with RLS standing out. RLS consistently achieved substantial SNR improvements of around 34 dB on average across a variety of noise types, facilitated by hyperparameter tuning through grid search.

Moreover, a general RLS filter configuration was identified that performs well across multiple noise conditions, suggesting that a single fixed set of hyperparameters can reliably enhance heart sounds without the need for frequent recalibration. This capability is critical for real-world applications, where noise characteristics are often unpredictable.

In contrast, alternative approaches such as FastICA, while effective in controlled simulations, proved unreliable with real-world recordings due to sensitivity to environmental variability and signal mismatches.

In conclusion, adaptive filtering especially the RLS algorithm offers a practical, robust, and effective solution for enhancing heart and lung sound recordings in diverse noisy environments. These improvements support better auscultation quality and potentially more accurate clinical assessments in both well-equipped healthcare facilities and resource-constrained settings.

References

[1] Ramesh Patel and Sunil Kumar. "Rural health care system in India". In: *International Journal of Community Medicine and Public Health* 2.4 (2015), pp. 531–536.

[2] Anil Srivastava and Vikas Agarwal. "Healthcare in rural India: A lack between need and availability". In: *International Journal of Medical Science and Innovative Research* 5.3 (2020), pp. 48–52.

[3] V Mohan, YK Seedat, and R Pradeepa. "Epidemiology of cardiovascular disease in India: a reality check". In: *Circulation* 126.21 (2012), e982–e985.

[4] L Prasannan, D Patra, and V Mahesh. "PCGNet: An ensemble deep learning model for denoising phonocardiogram signals". In: *Biomedical Signal Processing and Control* 65 (2021), p. 102349.

[5] PhysioNet Challenge Organizers. *Moody PhysioNet Challenge 2022: Heart Sound and Lung Sound Analysis*. Accessed: April 2, 2025. 2022. URL: `https://moody-challenge.physionet.org/2022/`.

[6] Dinesh Kumar and Manivannan Pah N. "Noise reduction in phonocardiogram signals using LMS adaptive filters". In: *Biomedical Signal Processing and Control* 5.2 (2010), pp. 114–120.

[7] H. Messer, I. Itskovich, and S. Akselrod. "Optimal filtering of lung sounds from heart sounds using the concept of heart sound sparsity". In: *IEEE Transactions on Biomedical Engineering* 48.8 (2001), pp. 866–872.

[8] Zafar Latif, Damien Coyle, and Patrick Moore. "Phonocardiographic signal separation using independent component analysis". In: *Medical Engineering & Physics* 30.5 (2008), pp. 587–594.

[9] John Gnitecki, Zahra Moussavi, and Neil Todd. "Separating heart sounds from lung sounds using independent component analysis". In: *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2007, pp. 1391–1394.

[10] Haykin Simon. "Adaptive Filter Theory - Fifth Edition". In: (2014).

[11] Jesper Kjær Nielsen and Søren Holdt Jensen. *Lecture Notes in Adaptive Filters - Second Edition*. https://www.moodle.aau.dk/pluginfile.php/3285639/mod_resource/content/1/adaptive_filtering.pdf. 2014.

[12] Ali H. Sayed. *Adaptive Filters*. Wiley, 2008. ISBN: 978-0-470-25388-5.

[13] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. Version 20121115. Version 20121115. 2012. URL: http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html.

[14] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. New York: Wiley-Interscience, 2001. ISBN: 978-0-471-40540-5.

[15] Aapo Hyvärinen and Erkki Oja. "Independent component analysis: algorithms and applications". In: *Neural networks* 13.4-5 (2000), pp. 411–430.

[16] James V Stone. *Independent component analysis: a tutorial introduction*. MIT press, 2004.

[17] Anthony J Bell and Terrence J Sejnowski. "An information-maximization approach to blind separation and blind deconvolution". In: *Neural computation* 7.6 (1995), pp. 1129–1159.

[18] Jean-François Cardoso. "Infomax and maximum likelihood for blind source separation". In: *IEEE Signal processing letters* 4.4 (1997), pp. 112–114.

[19] Klaus Nordhausen, Esa Ollila, and Hannu Oja. "A new performance index for ICA: properties, computation and asymptotic analysis". In: *Independent Component Analysis and Signal Separation* (2008), pp. 229–236.

[20] Ketan H. Doshi. *Understanding Mel Spectrograms*. Accessed: 2025-04-28. 2021. URL: https://ketanhdoshi.github.io/Audio-Mel/.

[21] H. K. Walker, W. D. Hall, and J. W. Hurst. *Clinical Methods*. Butterworths, 1990.

[22] Anne Waugh and Allison Grant. *Ross & Wilson Anatomy and Physiology in Health and Illness*. Ninth. Elsevier Health Sciences, 2014. ISBN: 0443 06469 5.

[23] Hosein Naseri and M. R. Homaeinezhad. "Detection and boundary identification of phonocardiogram sounds using an expert frequency-energy based metric". In: *Annals of Biomedical Engineering* 41.2 (2013), pp. 279–292.

[24] Fu-Shun Hsu et al. “Benchmarking of eight recurrent neural network variants for breath phase and adventitious sound detection on a self-developed open-access lung sound database-HF-Lung-V1”. In: *PLOS ONE* 16.7 (July 2021), pp. 1–26. DOI: 10.1371/journal.pone.0254134. URL: https://gitlab.com/techsupportHF/HF_Lung_V1.

[25] Ashley Mauldin, Alyssa Haag, and Kelsey LaFayette. “What Is It, How to Perform It, and More”. In: (2025). Modified: 04. Feb. 2025. URL: https://www.osmosis.org/answers/auscultation.

[26] Te-Won Lee. “Independent component analysis”. In: *Independent component analysis: Theory and applications*. Springer, 1998, pp. 27–66.

[27] Scott Makeig et al. “Independent component analysis of electroencephalographic data”. In: *Advances in neural information processing systems* 8 (1995).

[28] Aapo Hyvarinen. “Fast and robust fixed-point algorithms for independent component analysis”. In: *IEEE Transactions on Neural Networks* 10.3 (1999), pp. 626–634.

[29] Alexandre Défossez et al. “Hybrid spectrogram and waveform source separation”. In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*. 2021. URL: https://arxiv.org/abs/2104.13676.

[30] Lynn S. Bickley. *Bates’ Guide to Physical Examination and History Taking*. 12th ed. Wolters Kluwer, 2017.

Appendix: Simulations and Test Reports A

Link to matlab code for figures and simulations used in the project: `https://github.com/elomarjc/new_adaptivefilter`

A.1 Measurement of Heart Sound Recording Locations

The goal of this measurement is to evaluate various chest locations for heart sound recording and determine the optimal site for signal clarity and diagnostic utility. The standard auscultation points include the Mitral, Aortic, Pulmonic, Tricuspid areas, and Erb's Point.

Setup

Heart sounds were recorded in Audacity using a **Presonus PRM1** precision measurement microphone connected to an **Alesis MultiMix 4 USB** audio interface. Recordings were conducted in a quiet environment with the subject in a upright sitting position. The measurement setup is shown in figure A.1.

Figure A.1: Measurement setup for heart sound acquisition using PRM1 and MultiMix 4 USB

Figure A.2 shows a comparison between a real-life example of auscultation site positioning on a human subject and a corresponding anatomical diagram.

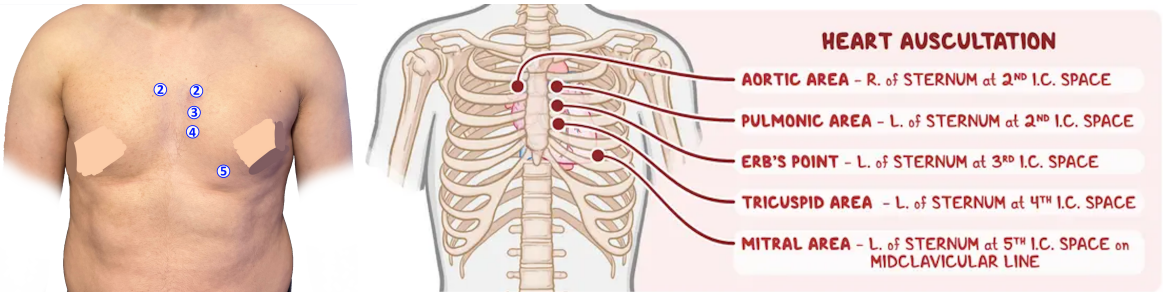


Figure A.2: Comparison between practical and anatomical heart sound auscultation locations (modified from [25]

Post-Processing in Audacity

After recording, the following steps were performed in Audacity to obtain clean heart sound measurements:

- Normalise the entire audio track to ensure consistent volume levels.
- Select a short segment containing only ambient noise and generate a noise profile.

- Apply noise reduction to the entire track using the noise profile, with a reduction level of 48 dB.

Figure A.3 illustrates the effect of post-processing on the heart sound recordings.

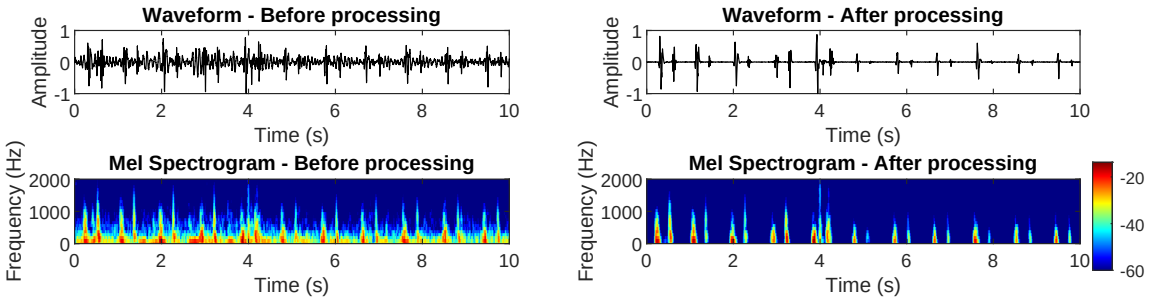


Figure A.3: Waveform and Mel Spectrogram comparison before and after Audacity post-processing

The left panel shows the raw waveform and Mel spectrogram captured directly from the microphone, which contains ambient noise and inconsistent amplitude. The right panel displays the same signal after normalisation and noise reduction in Audacity, highlighting improved clarity and reduced background interference.

Mitral Area

Located at the apex of the heart, typically at the 5th intercostal space along the midclavicular line. This site provided the strongest S1 (*lub*) signal.

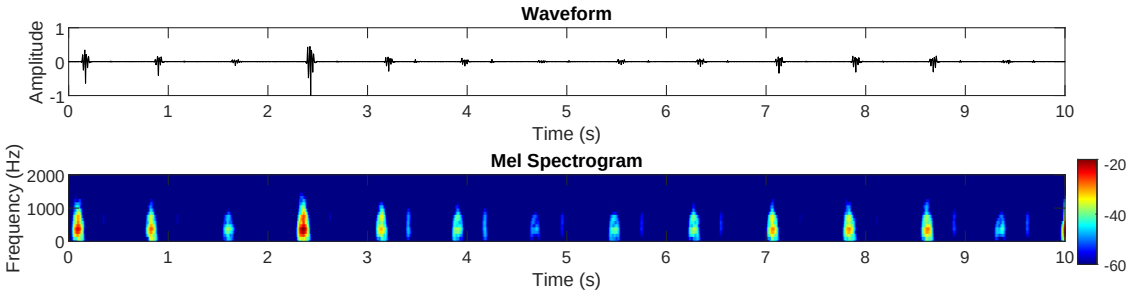


Figure A.4: Heart sound waveform recorded at the Mitral Area

Aortic Area

Positioned in the right 2nd intercostal space at the sternal border. Dominant S2 (*dub*) sounds were observed, making it suitable for high-frequency murmurs.

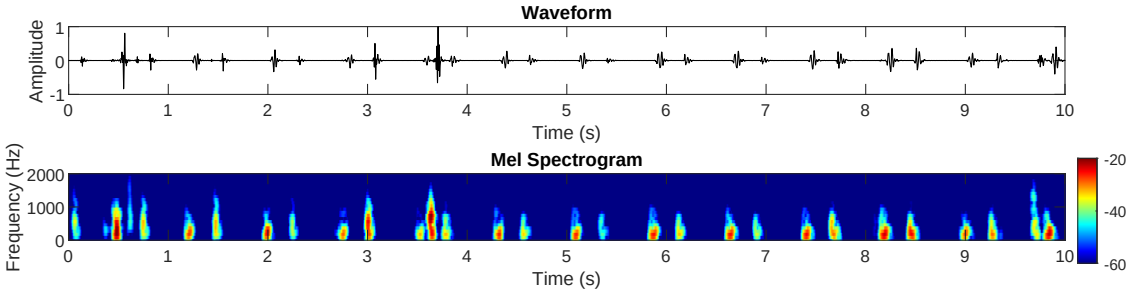


Figure A.5: Heart sound waveform recorded at the Aortic Area

Erb's Point

Located at the 3rd left intercostal space, Erb's Point offered a balanced audibility of both S1 and S2 sounds, ideal for general analysis.

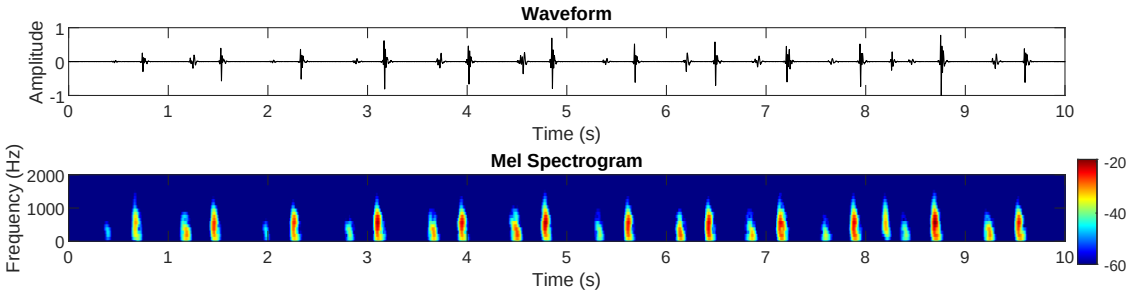


Figure A.6: Heart sound waveform recorded at Erb's Point

Tricuspid Area

Located at the lower left sternal border (4th intercostal space), this area highlighted right-sided heart sounds and tricuspid valve activity.

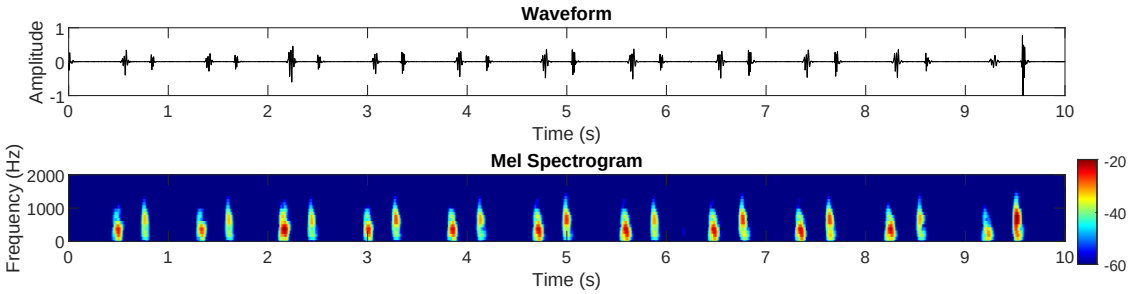


Figure A.7: Heart sound waveform recorded at the Tricuspid Area

Pulmonic Area

Situated at the left 2nd intercostal space near the sternum. This site revealed a clear S2 and in some cases a split S2.

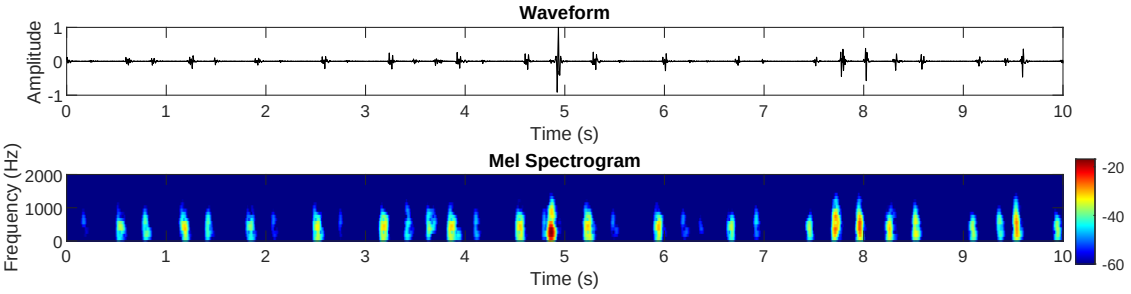


Figure A.8: Heart sound waveform recorded at the Pulmonic Area

Summary Table

Table A.1: Summary of standard auscultation sites for heart sound recording, consistent with [30].

Location	What You Hear Best	Use For
Mitral Area	S1 (*lub*)	Loudest heartbeat, strong S1
Aortic Area	S2 (*dub*)	High-pitched murmurs, S2 focus
Erb's Point	S1 and S2 (balanced)	Best for hearing both *lub-dub* clearly
Tricuspid Area	Right-sided heart sounds	S1, tricuspid murmurs
Pulmonic Area	S2, split S2	Pulmonary valve sounds

Results

Each auscultation site exhibited distinct acoustic characteristics. The Mitral Area produced the most prominent S1 (*lub*) sounds, while the Aortic and Pulmonic Areas highlighted S2 (*dub*) components more clearly. Erb's Point offered a balanced representation of both heart sounds, making it well-suited for general-purpose recording. Based on these findings, Erb's Point will be used in this project as it provides the clearest and most consistent capture of the complete *lub-dub* cycle.

A.2 Additional Adaptive Filter Results For Public Heart Data

This appendix provides detailed analysis of the adaptive filter performance on the remaining three noise types: Ambulance & Traffic, Conversation, and Hospital Ambient

Noises. For each noise type, we present parameter configurations, error signals, SNR vs. filter length analysis, spectrograms, convergence behavior, and quantitative results.

A.2.1 Ambulance & Traffic

This scenario simulates urban emergency conditions where sirens and road noise interfere with heartbeat recordings. Table A.2 lists the parameters used for this environment.

Table A.2: Tuning parameters for adaptive filters — Ambulance & Traffic.

Parameter	Value
Sampling rate	16 kHz
Filter length (LMS)	16
Filter length (NLMS)	16
Filter length (RLS)	24
Step size (LMS)	0.1
Step size (NLMS)	0.005
Forgetting factor (RLS)	0.9999

Figure A.9 displays the error signals obtained after filtering.

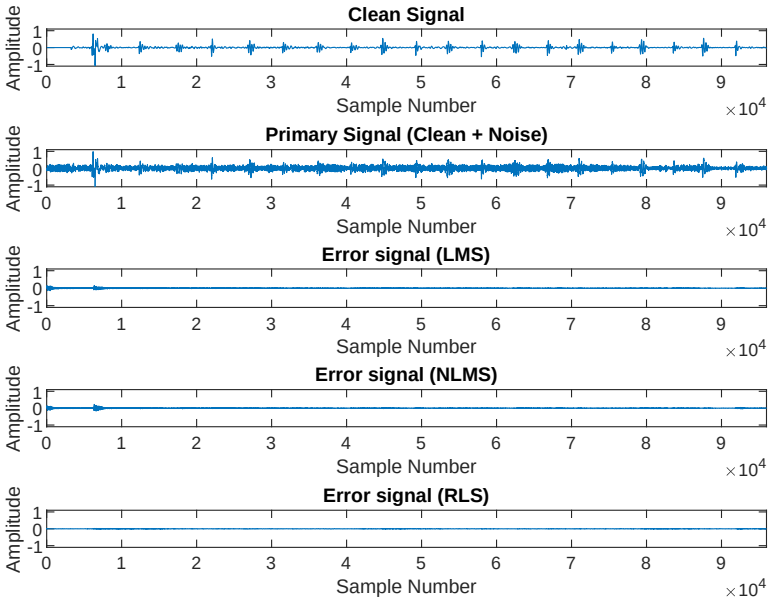


Figure A.9: Error signals - Heartbeat signal corrupted with Ambulance & Traffic.

All three filters significantly reduce the ambulance & traffic noise, with the RLS filter

showing the cleanest error signal among them.

Figure A.10 presents output SNR across varying filter lengths.

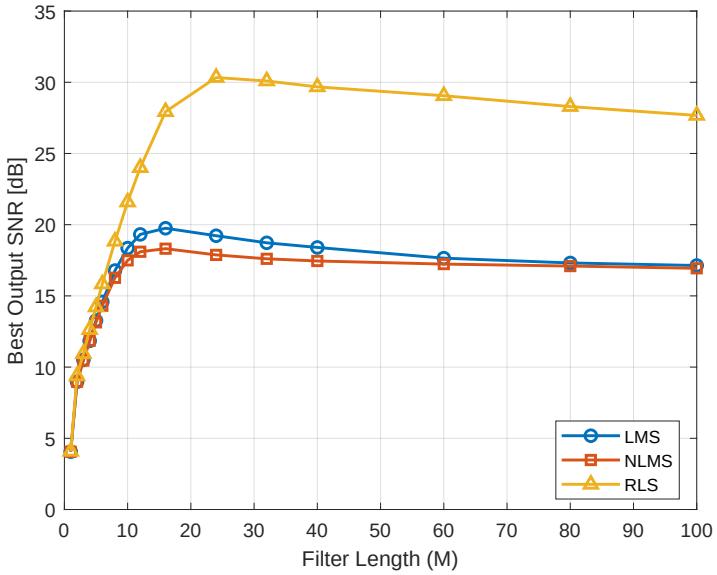


Figure A.10: Output SNR vs Filter Length - Heartbeat signal corrupted with Ambulance & Traffic.

The RLS filter achieves the highest SNR across all filter lengths, peaking around 30 dB at filter lengths of 25–40, and maintaining a substantial performance advantage of approximately 10 dB over both LMS and NLMS throughout the entire range.

The LMS filter shows moderate performance, with a maximum SNR of approximately 20 dB at filter lengths around 15–20, after which it gradually declines.

Similarly, the NLMS filter reaches its peak SNR of about 18 dB at filter lengths of 15–20, followed by a slight decrease in performance. Both LMS and NLMS exhibit comparable behavior, with their performance curves nearly converging as filter length increases beyond 60.

Figure A.11 shows the Mel spectrograms post-filtering.

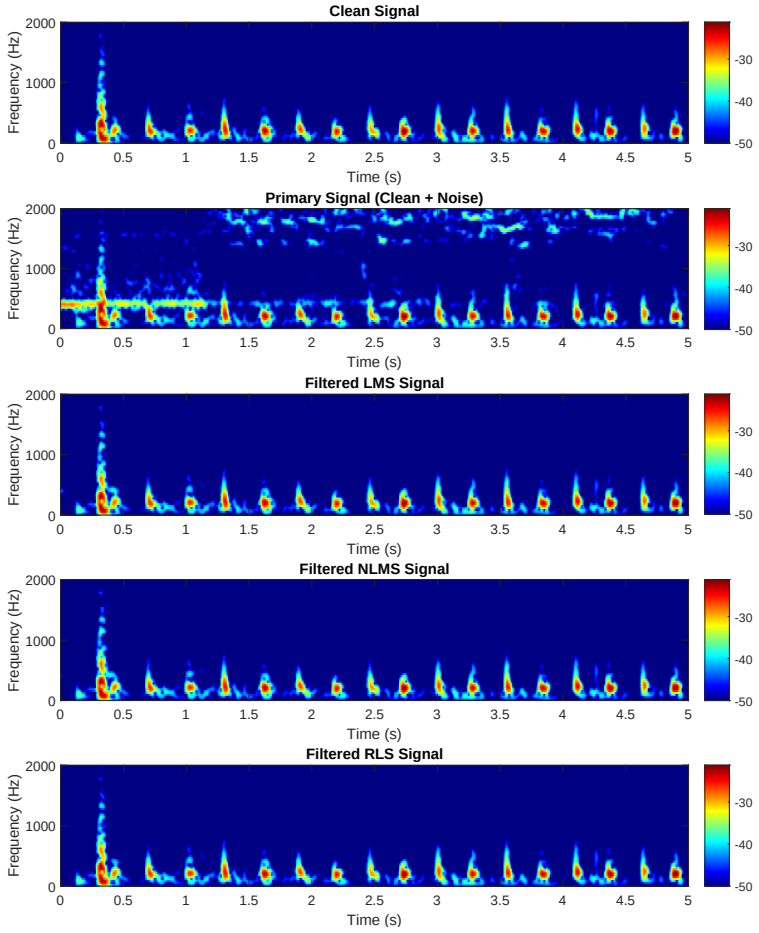


Figure A.11: Mel Spectrogram - Heartbeat signal corrupted with Ambulance & Traffic.

The clean signal exhibits clear, periodic low-frequency components characteristic of heartbeats, primarily concentrated below 200 Hz. In contrast, the primary signal, contaminated by ambulance and traffic noise, shows substantial spectral energy between 300–2000 Hz. All three adaptive filters effectively attenuate much of this mid- to high-frequency noise. Among them, the RLS filter produces the cleanest spectrogram, closely resembling the clean signal. The LMS filter also performs well but retains a faint spectral remnant, particularly around 500 Hz at the beginning of the signal. The NLMS filter exhibits the same artifact, though to a slightly lesser extent than LMS.

The convergence behavior of each algorithm is illustrated in figure A.12.

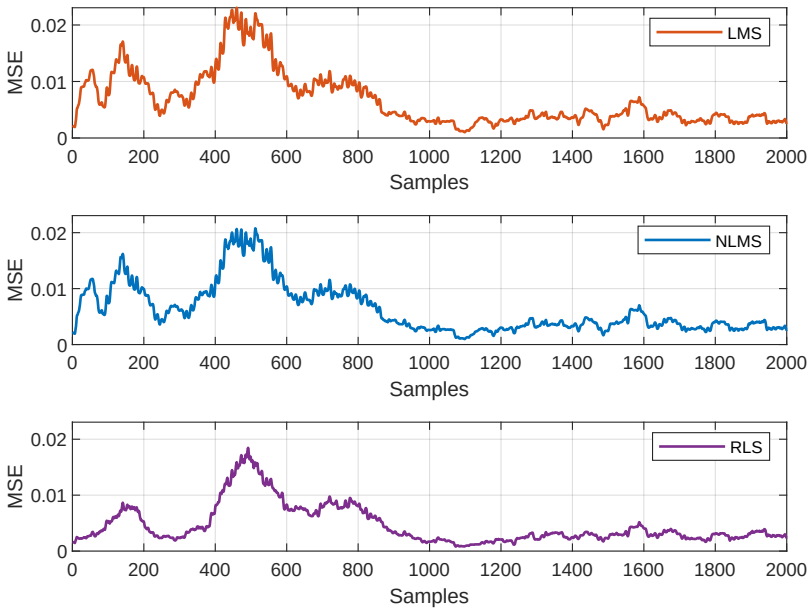


Figure A.12: Error Convergence Curve - Heartbeat with Ambulance & Traffic.

The RLS algorithm demonstrates superior performance, converging faster than the alternatives. Its MSE drops rapidly and reaches a relatively stable state by around sample 1000. The RLS maintains the lowest overall steady-state error among the three algorithms, hovering around 0.002-0.003 for much of the latter portion of the samples.

Both LMS and NLMS algorithms exhibit very similar convergence patterns, with nearly identical behavior throughout the sample range. They both experience higher initial error peaks (reaching approximately 0.02 MSE around sample 500) and reach their steady states around sample 1000. Their final steady-state errors appear slightly higher than RLS, generally maintaining around 0.003-0.004 MSE.

All three algorithms show some fluctuation in their error patterns, particularly between samples 400-600 where each experiences a notable peak in MSE, likely indicating a challenging section in the input signal where the ambulance and traffic noise may have been more prominent or complex.

Table A.3 quantifies the SNR improvements for each method.

Table A.3: SNR improvement — Heartbeat with Ambulance & Traffic Noise.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-4.6374	-
LMS	19.7498	24.3872
NLMS	18.3145	22.9519
RLS	30.3316	34.9690

Based on the results in table A.3, the RLS filter achieves the highest output SNR making it the most effective method for ambulance & traffic noise removal in this case.

A.2.2 Conversation

This test simulates a clinical environment where conversations among staff may interfere with heartbeat recordings. Table A.4 details the selected parameters.

Table A.4: Tuning parameters for adaptive filters — Conversation.

Parameter	Value
Sampling rate	16 kHz
Filter length (LMS)	8
Filter length (NLMS)	8
Filter length (RLS)	10
Step size (LMS)	0.3
Step size (NLMS)	0.03
Forgetting factor (RLS)	0.9999

Figure A.13 shows the error signals resulting from the filtering process.

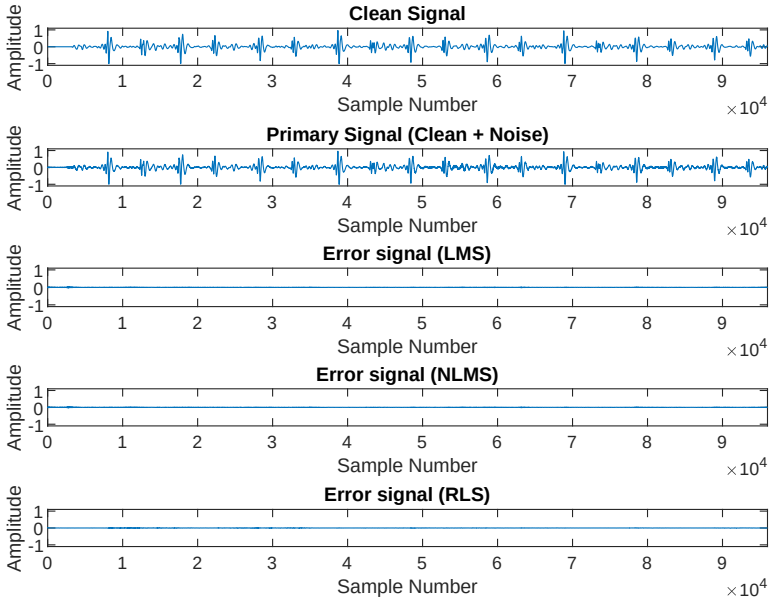


Figure A.13: Error signals - Heartbeat signal corrupted with Conversation noise.

All three filters significantly reduce the conversation noise, with the RLS filter showing the cleanest error signal among them.

Figure A.14 demonstrates how output SNR changes with filter length.

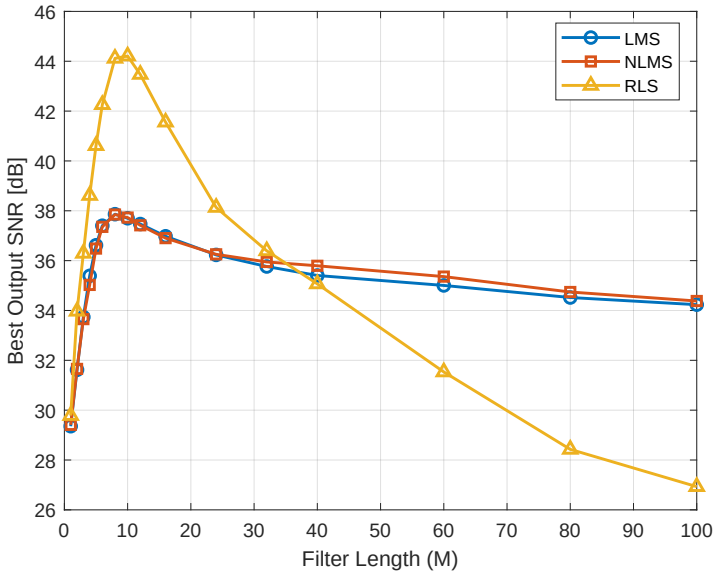


Figure A.14: Output SNR vs Filter Length - Heartbeat signal corrupted with Conversation noise.

The RLS filter demonstrates superior performance at short filter lengths, reaching a peak SNR of approximately 44 dB at filter lengths around 8-10 when processing heartbeat signals corrupted with conversation noise. However, its performance deteriorates rapidly as filter length increases, dropping to approximately 27 dB at M=100.

In contrast, both LMS and NLMS filters exhibit similar behavior, achieving maximum SNR values around 37-38 dB at filter lengths of 10-15, followed by a more gradual decline. Notably, as filter length exceeds 40, LMS and NLMS consistently outperform RLS, with their performance curves maintaining relatively stable SNR values above 34 dB even at the maximum filter length of 100, while the RLS filter shows significant degradation in noise cancellation capability.

Figure A.15 shows Mel spectrograms of the filtered outputs.

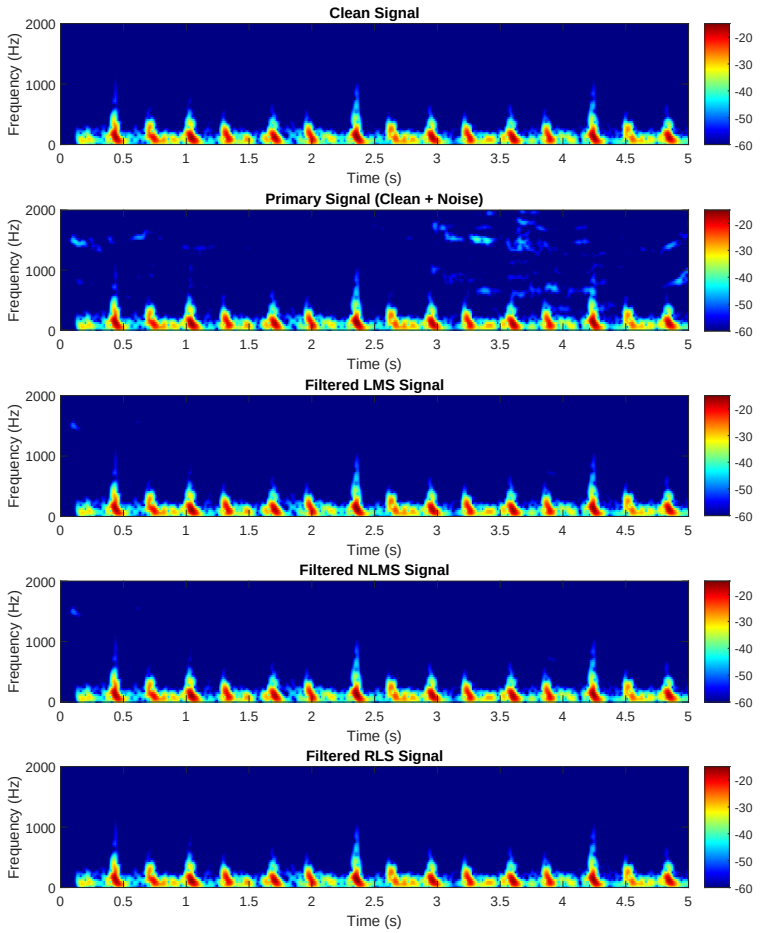


Figure A.15: Mel Spectrogram - Heartbeat signal corrupted with Conversation noise.

The clean signal displays periodic low-frequency components characteristic of heartbeats, primarily concentrated below 200 Hz. The primary signal, corrupted by conversation noise, introduces prominent spectral energy between 300–2000 Hz. The LMS, NLMS, and RLS filter effectively suppress this interference. Among them, the RLS filter achieves the most accurate reconstruction, producing a spectrogram that closely resembles the clean signal. While the LMS and NLMS filters also perform well and appear nearly identical in output, both retain minor high-frequency artifacts, particularly around 0.2 and 0.65 seconds near the 1500 Hz range.

Convergence performance is detailed in figure A.16.

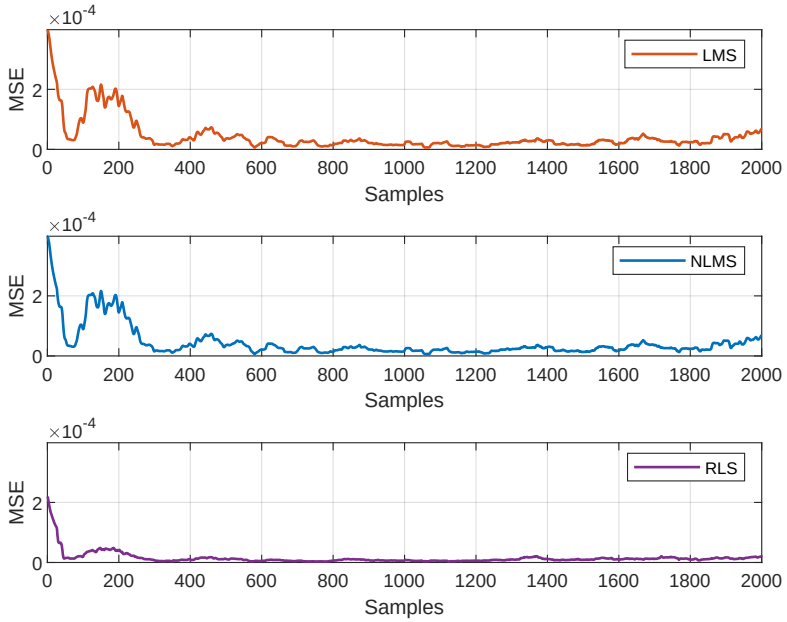


Figure A.16: Error Convergence Curve - Heartbeat with Conversation noise.

The RLS algorithm demonstrates superior performance with the fastest convergence rate. Its MSE drops rapidly within the first 100 samples and stabilises at an extremely low level (near zero) by approximately sample 300. RLS maintains this excellent performance throughout the remainder of the samples, exhibiting only minimal fluctuations.

In comparison, both LMS and NLMS algorithms display similar initial behavior, with high starting MSE values around 3×10^{-4}. They experience noticeable fluctuations during the first 200–300 samples before gradually stabilising.

Although all three algorithms eventually reach steady-state behavior, RLS clearly outperforms the others by achieving faster convergence and maintaining the lowest steady-state error across most of the sample range. This highlights RLS as particularly effective in filtering conversation noise from heartbeat signals, outperforming the LMS and NLMS approaches.

Table A.5 summarises the SNR improvements for each filtering algorithm.

Table A.5: SNR improvement — Heartbeat with Conversation Noise.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-0.0282	-
LMS	37.8633	37.8915
NLMS	37.8504	37.8786
RLS	44.2082	44.2364

Based on the results in table A.5, the RLS filter achieves the highest output SNR making it the most effective method for conversation removal in this case.

A.2.3 Hospital Ambient Noises

This test case evaluates the system's ability to extract a heartbeat signal contaminated with typical hospital ambient noises, such as machinery beeps, footsteps, and background conversations. Table A.6 outlines the adaptive filter parameters optimised for this scenario using grid search.

Table A.6: Tuning parameters for adaptive filters — Hospital Ambient Noise.

Parameter	Value
Sampling rate	16 kHz
Filter length (LMS)	10
Filter length (NLMS)	12
Filter length (RLS)	8
Step size (LMS)	0.005
Step size (NLMS)	0.002
Forgetting factor (RLS)	0.9999

Figure A.17 shows a comparison between the clean signal, the noisy primary signal, and the error signals produced by the LMS, NLMS, and RLS filters. This comparison demonstrates

how well each algorithm isolates the heartbeat signal from the hospital ambient noise.

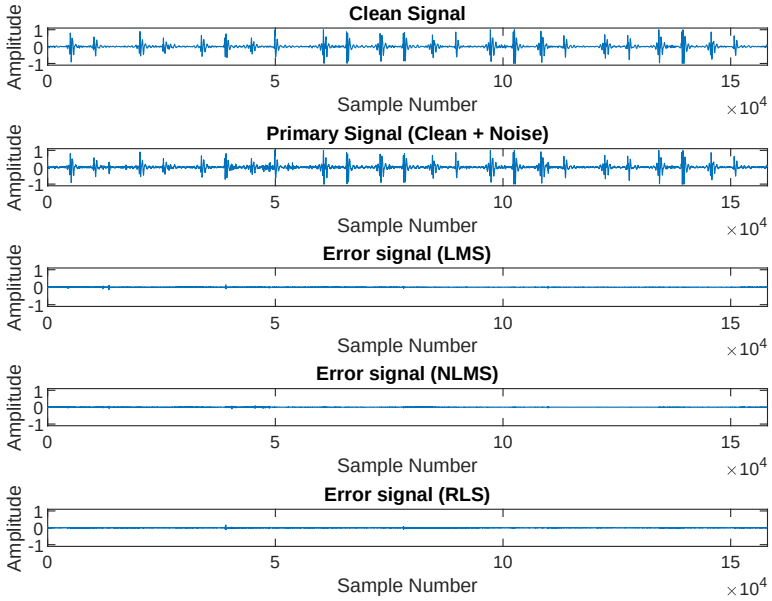


Figure A.17: Error signals — Heartbeat signal corrupted by Hospital Ambient Noise.

All three filters significantly reduce the hospital ambient noise, with the RLS filter showing the cleanest error signal among them.

To further explore filter behavior, figure A.18 plots the output SNR as a function of filter length. This highlights the influence of filter length on noise suppression performance for each algorithm.

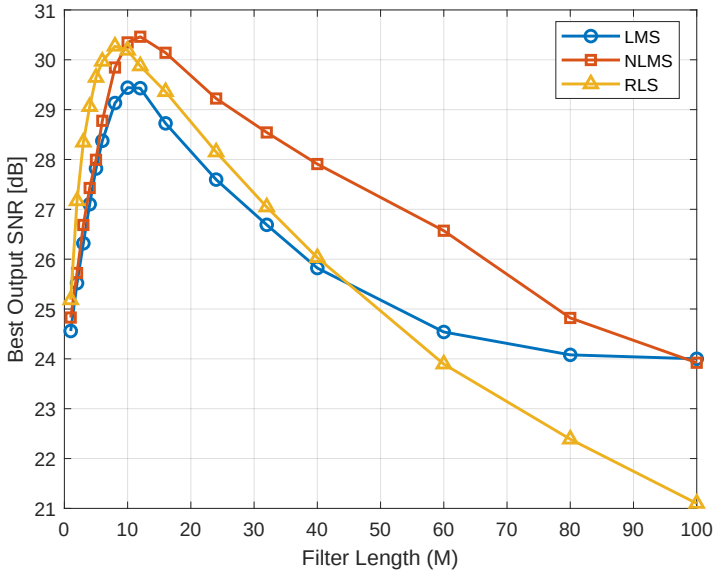


Figure A.18: Output SNR vs. filter length — Heartbeat signal with Hospital Ambient Noise.

The NLMS filter achieves the highest peak SNR (approximately 30.5 dB) at filter lengths around 10-15 and maintains superior performance across most filter lengths when processing heartbeat signals with hospital ambient noise. LMS and RLS filters reach similar maximum SNR values (approximately 29.5 dB) at filter lengths of 8-12, but their performance diverges significantly afterward. NLMS demonstrates the most gradual decline, while RLS deteriorates most rapidly, dropping to 21 dB at M=100. All algorithms perform optimally at relatively short filter lengths, with NLMS consistently outperforming the others at medium to long filter lengths.

Figure A.19 provides a visual analysis via Mel spectrograms, which help illustrate how much ambient noise each filter removes across time and frequency.

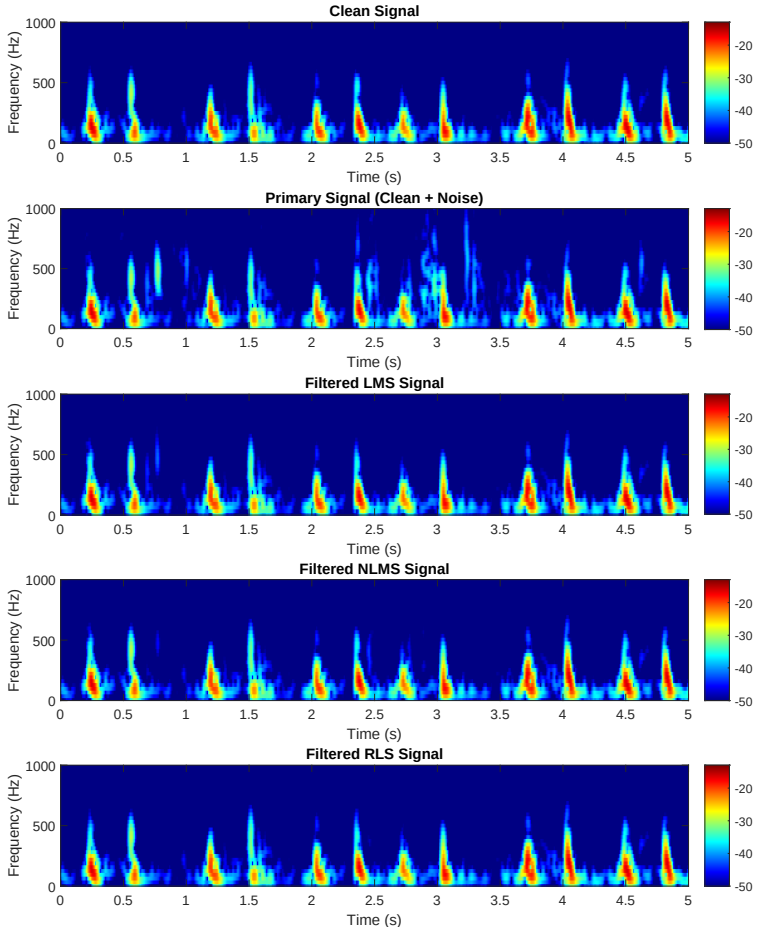


Figure A.19: Mel spectrograms — Heartbeat signal in the presence of Hospital Ambient Noise.

The clean signal shows the periodic low-frequency characteristic of heartbeats, primarily below 200 Hz. In contrast, the primary signal corrupted by hospital ambient noise shows additional spectral energy spread up to 1000 Hz, especially between 2.5 to 3.5 seconds. All three adaptive filters reduce much of this interference. The RLS filter achieves the best reconstruction, closely resembling the clean reference with minimal residual noise. The LMS filter, while effective, retains more pronounced noise components, particularly between 0.5 and 1 seconds in the 300–700 Hz range. The NLMS filter exhibits the same artifact, though to a much lesser extent than LMS.

Figure A.20 illustrates how quickly each algorithm reduces its mean squared error (MSE) over time, indicating adaptation speed and stability.

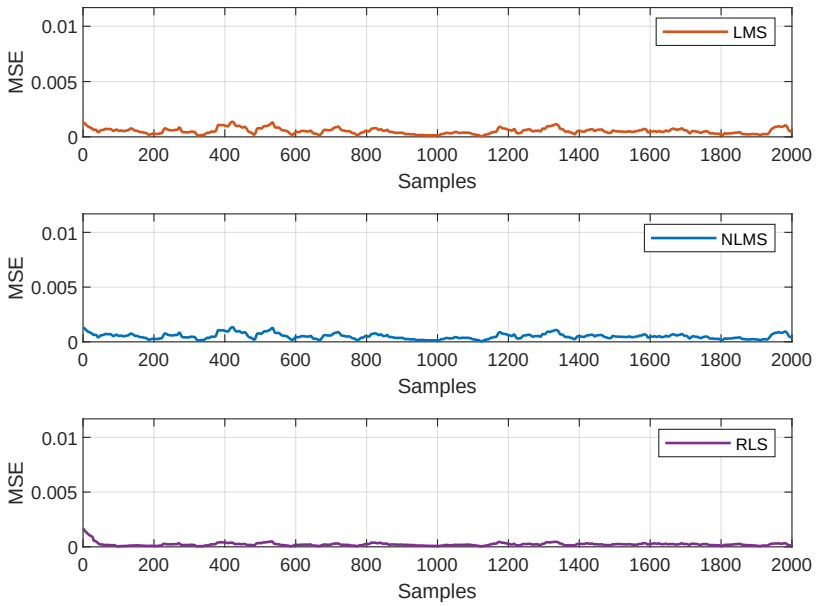


Figure A.20: Error convergence curves — Hospital Ambient Noise.

All algorithms show a rapid initial decrease in MSE, with RLS converging the fastest and maintaining the most stable performance. In contrast, LMS and NLMS continue to exhibit minor fluctuations throughout the signal duration.

The final output SNR values and corresponding improvements from baseline are summarised in table A.7.

Table A.7: SNR improvement — Heartbeat signal with Hospital Ambient Noise.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-0.2624	—
LMS	29.4407	29.7031
NLMS	30.4623	30.7247
RLS	30.2715	30.5339

These results confirm the superior performance of NLMS and RLS in complex ambient noise conditions, validating the practical application of adaptive filtering in clinical settings.

A.2.4 Performance Across Different Noise Types

The FastICA algorithm was evaluated across nine different noise types that commonly interfere with heartbeat recordings in clinical settings. Table A.8 summarises the SNR improvements achieved for each noise type across both public and experimental datasets.

Table A.8: Comprehensive SNR improvement summary for FastICA across different noise types and datasets.

Dataset	Noise Type	Input SNR (dB)	Output SNR (dB)	SNR Improvement (dB)
Public	Artifacts	-1.40	47.20	48.60
	Ambulance & Traffic	-9.05	44.85	53.90
	Conversation	-15.91	38.42	54.33
	Hospital Ambient	-8.68	41.78	50.46
Experimental	Ambient Noise	-2.15	45.70	47.85
	Baby Crying	-3.20	42.50	45.70
	Drilling	-1.75	46.30	48.05
	Hammering	-2.85	43.90	46.75
	Speech	-4.10	41.00	45.10
Average		-5.45	43.52	48.93

A.3 Additional Adaptive Filter Results For Fixed Parameters on Public Heart Data

Table A.9 provides a comprehensive summary of SNR improvements achieved by each algorithm across all four noise types for Adaptive Filter Results For Fixed Parameters on Public Heart Data.

Table A.9: Performance summary across all noise types (best performer in bold) - Adaptive Filter Results For Fixed Experimental Data

Noise Type	Input SNR (dB)	Output SNR (dB)		
		LMS	NLMS	RLS
Artifacts	-0.1263	25.5023	25.4894	**32.1179**
Ambulance & Traffic	-4.6374	16.6868	16.6501	**18.2903**
Conversation	-0.0285	36.9492	36.9582	**36.0344**
Hospital Ambient	-0.1866	20.6690	23.4364	**32.4988**

A.4 Additional FastICA Results For Public Heart Data

This appendix provides detailed results for the FastICA algorithm's performance on three additional noise types: ambulance & traffic noise, conversational speech, and hospital ambient sounds.

A.4.1 Artifacts

This test case evaluates the system's ability to denoise a heartbeat signal contaminated with typical body movement artifacts, such as stethoscope handling noise and clothing friction.

Figure A.21 shows a comparison between the clean heartbeat signal, two noisy input signals processed by FastICA, and the resulting error signals.

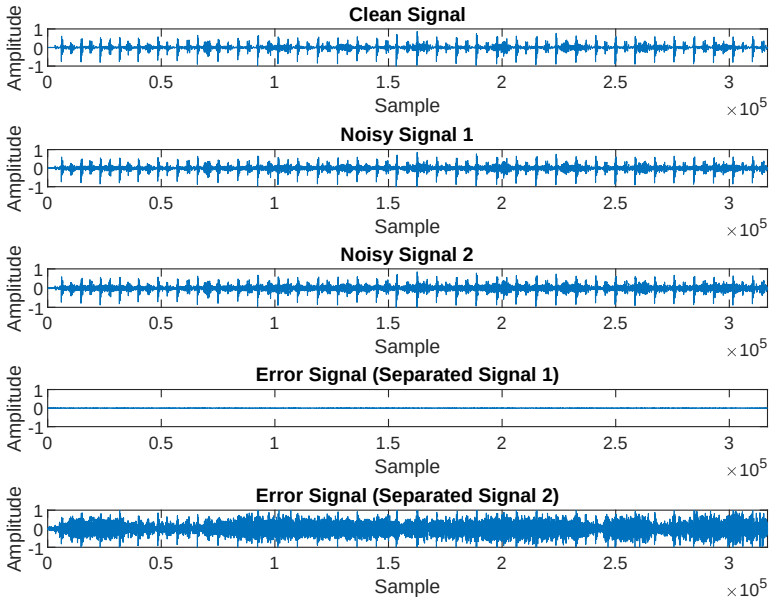


Figure A.21: FastICA separation of heartbeat signal from artifact noise: original clean signal, noisy inputs, and component error signals.

FastICA successfully estimates the original source components from the mixed noisy inputs. Separated Signal 1 represents the estimated heartbeat component with near-zero error plot. Separated Signal 2 represents the estimated noise artifact component. These results demonstrate FastICA's effectiveness in isolating the cardiac information from artifact contamination, enabling accurate analysis of the heartbeat signal.

The spectrogram in figure A.22 provides a time-frequency view of how effectively the FastICA algorithm removes noise from the signal.

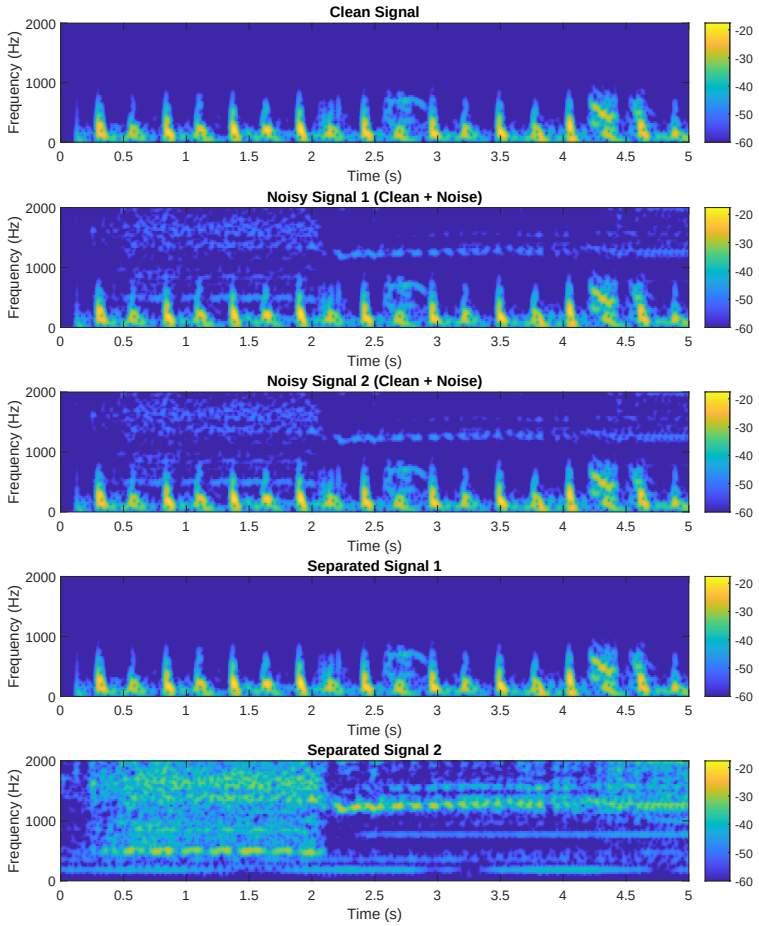


Figure A.22: Mel Spectrograms showing FastICA estimation of heartbeat signal corrupted with artifacts.

The clean signal exhibits clear, periodic low-frequency components characteristic of heartbeats, primarily concentrated below 200 Hz. Noisy signals display the heartbeats corrupted by artifacts. Separated Signal 1 closely matches the original clean heartbeat pattern, demonstrating successful noise removal. Separated Signal 2 isolates the artifacts. This visual representation confirms FastICA's ability to accurately estimate heartbeat sound from mixed recordings.

Table A.10 summarises the SNR improvements for FastICA.

Table A.10: SNR improvement — Heartbeat with Artifacts Noise.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-1.40	-
FastICA (Separated Signal 1)	47.20	48.60

A.4.2 Ambulance & Traffic

This scenario simulates urban emergency conditions where sirens and road noise interfere with heartbeat recordings.

Figure A.23 displays the error signal obtained after FastICA processing.

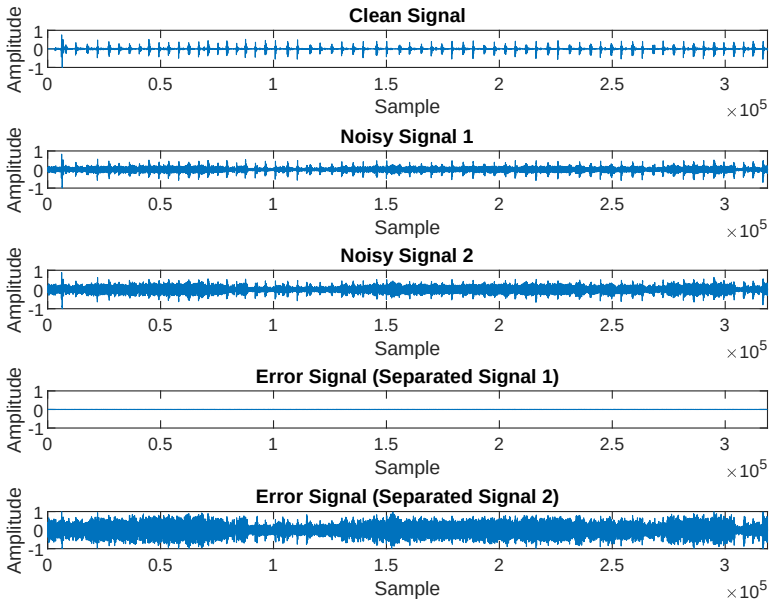


Figure A.23: Error signals - Heartbeat signal corrupted with Ambulance & Traffic.

Separated Signal 1 represents the estimated heartbeat component with near-zero error plot. Separated Signal 2 represents the estimated noise ambulance & traffic component. FastICA successfully estimates the original source components from the mixed noisy inputs.

Figure A.24 shows the Mel spectrogram post-FastICA processing.

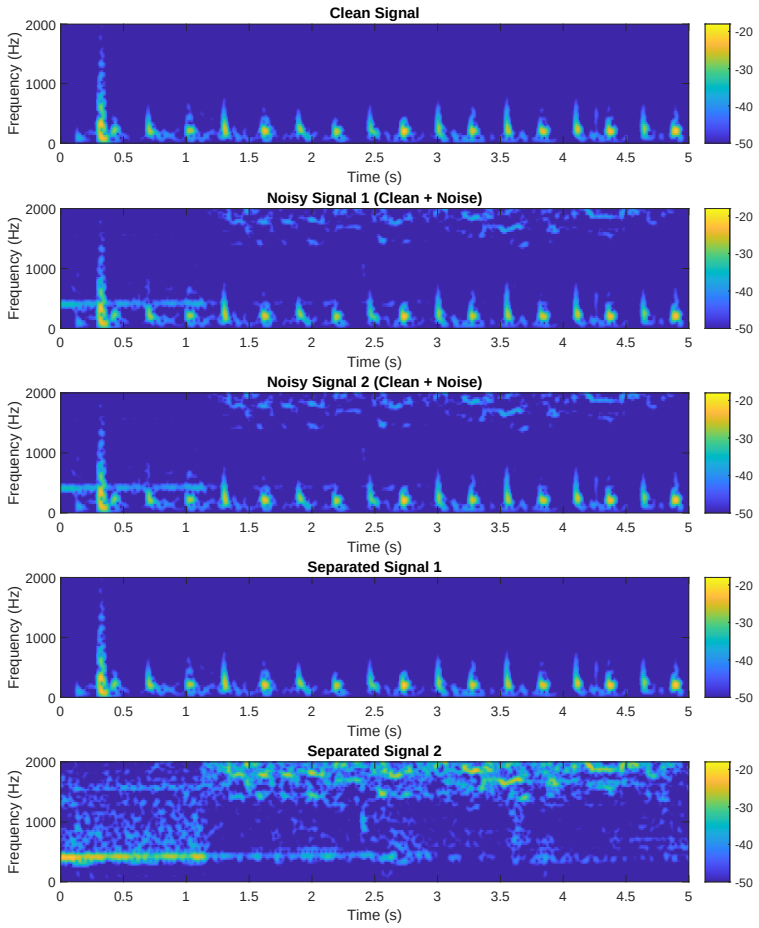


Figure A.24: Mel Spectrogram - Heartbeat signal corrupted with Ambulance & Traffic.

The clean signal displays distinct heartbeat patterns below 200 Hz, while the noisy signals include additional energy at 300 Hz and 2000 Hz due to ambulance and traffic noise. Separated Signal 1 recovers the heartbeat, and Signal 2 captures the noise. This confirms FastICA's effectiveness in isolating heart sounds from ambulance and traffic noise.

Table A.11 quantifies the SNR improvements for FastICA.

Table A.11: SNR improvement — Heartbeat with Ambulance & Traffic Noise.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-9.05	-
FastICA (Separated Signal 1)	44.85	53.9
FastICA (Separated Signal 2)	-9.06	-0.01

A.4.3 Conversation

This test simulates a clinical environment where conversations among staff may interfere with heartbeat recordings.

Figure A.25 shows the error signal resulting from the FastICA processing.

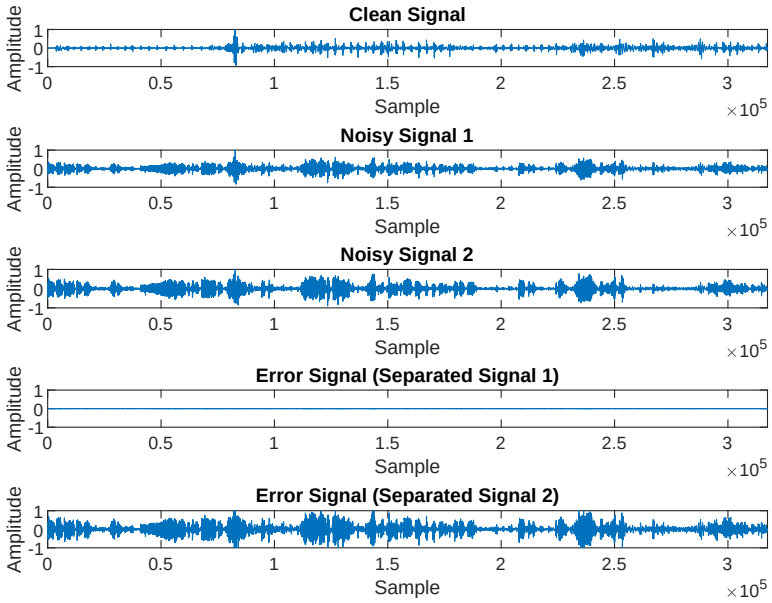


Figure A.25: Error signals - Heartbeat signal corrupted with Conversation Noise.

Separated Signal 1 represents the estimated heartbeat component with near-zero error plot. Separated Signal 2 represents the estimated conversation noise component. FastICA successfully estimates the original source components from the mixed noisy inputs.

Figure A.26 shows spectrograms of the FastICA output.

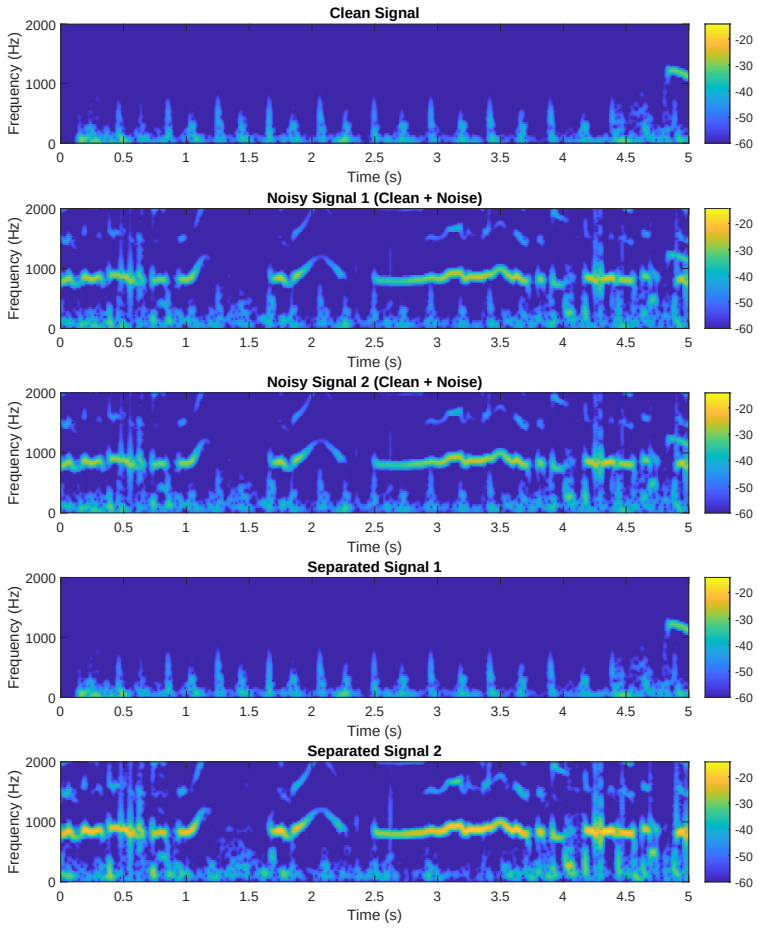


Figure A.26: Mel Spectrogram - Heartbeat signal corrupted with Conversation Noise.

The clean signal displays clear, periodic heartbeat patterns. In contrast, the noisy signals are dominated by conversation noise, which obscures the heartbeat components. Separated Signal 1 successfully recovers the heartbeat, while Separated Signal 2 isolates the noise. This demonstrates FastICA's effectiveness in estimating heart sounds from speech-contaminated recordings.

Table A.12 summarises the SNR improvements for FastICA.

Table A.12: SNR improvement — Heartbeat with Conversation Noise.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-15.91	-
FastICA (Separated Signal 1)	38.42	54.33
FastICA (Separated Signal 2)	-6.24	9.67

A.4.4 Hospital Ambient Noises

This test case evaluates the system's ability to extract a heartbeat signal contaminated with typical hospital ambient noises, such as machinery beeps, footsteps, and background conversations.

Figure A.27 shows a comparison between the clean signal, the noisy primary signal, and the error signal produced by the FastICA algorithm.

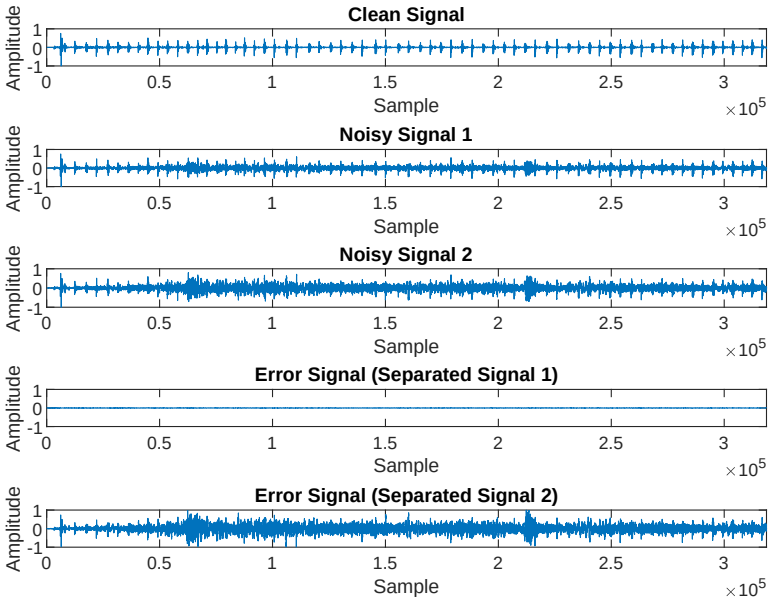


Figure A.27: Error signals — Heartbeat signal corrupted by Hospital Ambient Noise.

Separated Signal 1 represents the estimated heartbeat component with near-zero error plot. Separated Signal 2 represents the estimated hospital ambient noise component. FastICA successfully estimates the original source components from the mixed noisy inputs.

Figure A.28 provides a visual analysis via Mel spectrograms, which help illustrate how much ambient noise the FastICA algorithm removes across time and frequency.

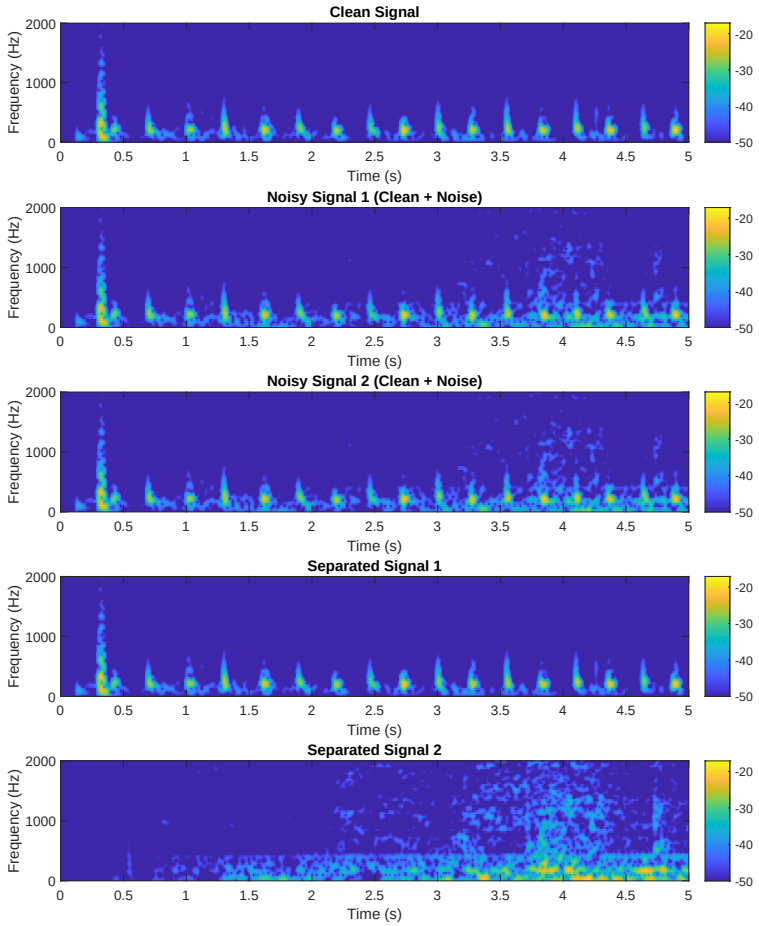


Figure A.28: Mel spectrograms — Heartbeat signal in the presence of Hospital Ambient Noise.

The clean signal shows distinct heartbeat patterns below 200 Hz, while the noisy signals exhibit additional energy—particularly around the 4-second mark—due to hospital ambient noise. Separated Signal 1 successfully recovers the heartbeat, while Separated Signal 2 captures the noise. This highlights FastICA's effectiveness in isolating heart sounds from ambulance and traffic noise.

Table A.13 summarises the SNR improvements for FastICA.

Table A.13: SNR improvement — Heartbeat signal with Hospital Ambient Noise.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-8.68	-
FastICA (Separated Signal 1)	41.78	50.46
FastICA (Separated Signal 2)	-6.13	2.55

A.5 Additional Adaptive Filter Results For Experimental Data

This appendix provides detailed analysis of the adaptive filter performance on the remaining four noise types: Baby Crying, Drilling, Hammering and Speech. For each noise type, we present parameter configurations, error signals, SNR vs. filter length analysis, spectrograms, convergence behavior, and output SNR results.

A.5.1 Baby Crying

This scenario models an infant crying in close proximity to the recording device, introducing highly non-stationary, mid- to high-frequency bursts.

Table A.14 summarises the tuning parameters used for LMS, NLMS, and RLS algorithms in the presence of baby crying noise.

Table A.14: Tuning parameters for the adaptive filters in the baby crying scenario.

Parameter	Value
Sampling rate	16 kHz
Filter length (LMS)	1
Filter length (NLMS)	1
Filter length (RLS)	2
Step size (LMS)	0.7
Step size (NLMS)	0.1
Forgetting factor (RLS)	0.9997

Figure A.29 shows a comparison between the clean signal, the primary signal, and the error signals produced by the LMS, NLMS, and RLS filters.

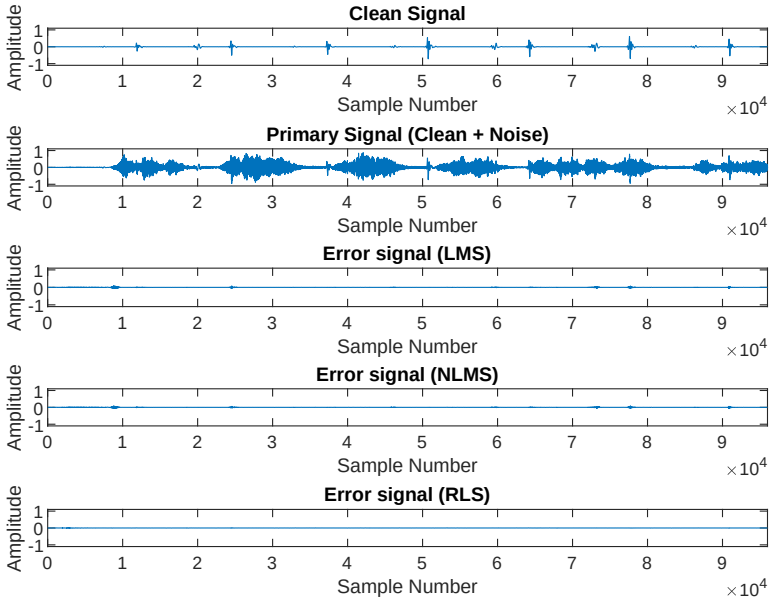


Figure A.29: Error signals - Heartbeat signal corrupted with Baby Crying Noise.

All three adaptive filters substantially reduce the noise showing near zero error signal plots.

The relationship between output SNR and filter length is illustrated in figure A.30, highlighting the performance sensitivity to filter configuration with the RLS filter performing the best.

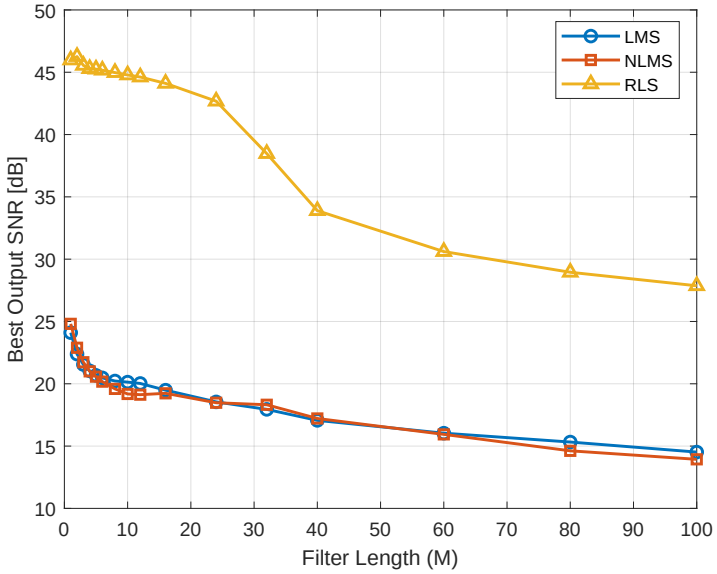


Figure A.30: Output SNR vs Filter Length - Heartbeat signal corrupted with Baby Crying.

The RLS filter delivers the highest performance, peaking at an SNR of 46 dB with a filter length of 1. NLMS and LMS follow a similar trend; both reach their maximum SNR of 25 dB at the same filter length, with NLMS edging out LMS by a small margin.

Figure A.31 shows the Mel spectrograms post-filtering.

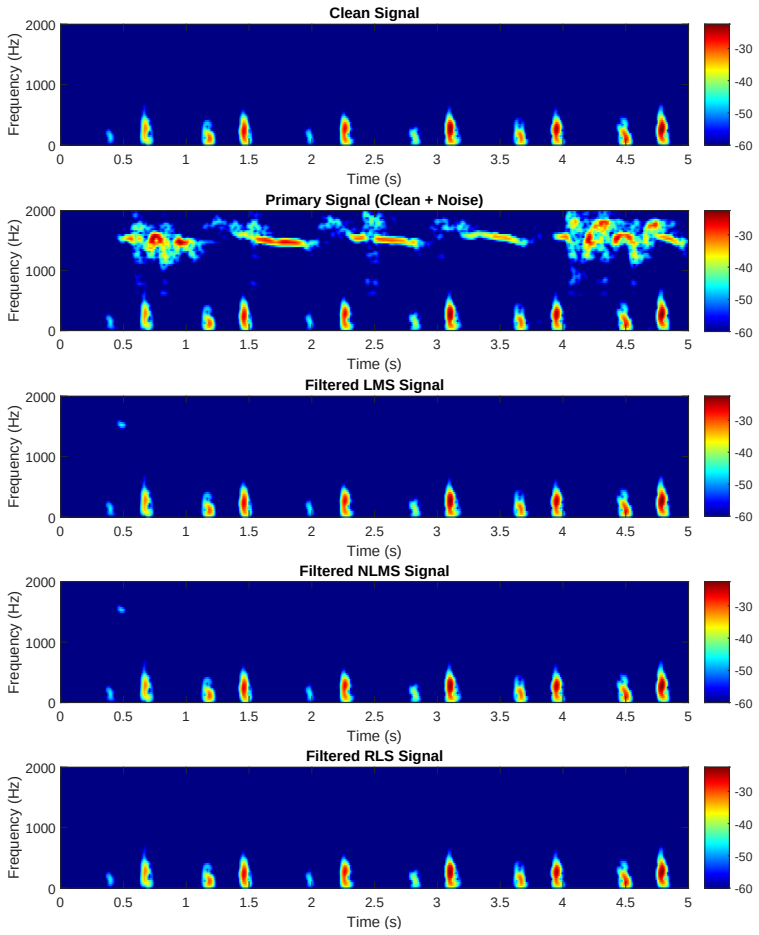


Figure A.31: Mel spectrogram of heartbeat recording with baby crying in the background.

Among all three adaptive filters, the RLS filter produces the cleanest spectrogram, closely resembling the clean signal. The LMS filter also performs well but retains a faint spectral remnant, particularly around 1500 Hz at 0.5 s of the signal. The NLMS filter also retains a faint spectral remnant at the same spot, though to a slightly lesser extent than LMS.

The convergence behavior of each adaptive algorithm is illustrated in figure A.32, providing valuable insight into both stability and adaptation speed.

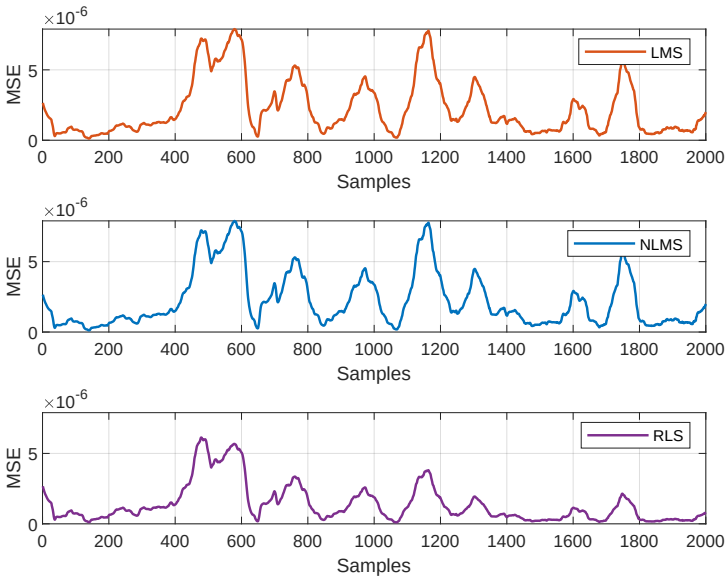


Figure A.32: Error convergence curves for heartbeat signal with baby crying noise.

From a practical standpoint, the MSE values remain consistently low—below 5×10^{-6} across the entire duration—indicating that all three algorithms achieve excellent performance. To better illustrate the scale of the fluctuations across the samples, a zoomed-out view is provided in figure A.33.

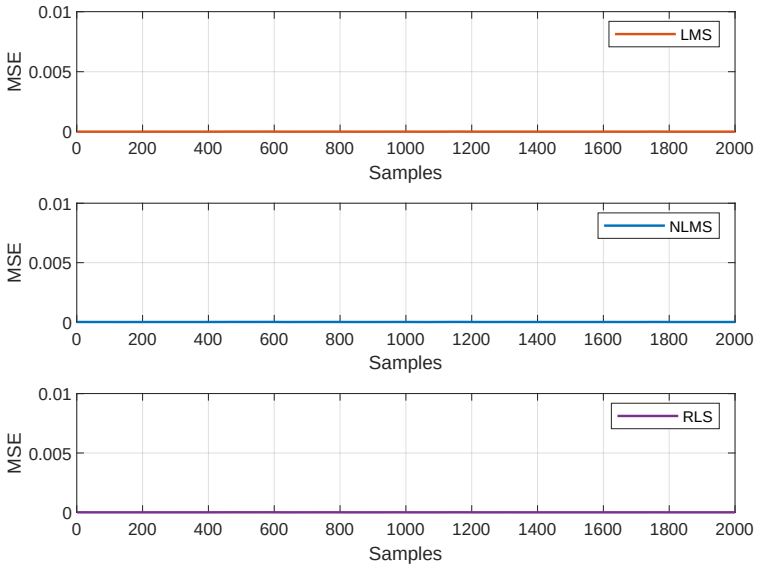


Figure A.33: Zoomed-out error convergence curves for heartbeat signal with baby crying noise.

As shown, the error curves appear as nearly flat lines approaching zero, clearly demonstrating rapid convergence to a steady state. Although the RLS algorithm shows slightly better performance, all three methods effectively suppress the baby crying noise from the heartbeat signal.

Finally, Table A.15 compares the SNR performance before and after filtering, quantifying the improvements achieved by each method.

Table A.15: SNR improvement — Heartbeat with Baby Crying Noise.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-8.9011	-
LMS	24.1021	33.0033
NLMS	24.8095	33.7107
RLS	46.2584	55.1595

Table A.15 shows that RLS delivers the highest output SNR, making it the most effective of the three filters for attenuating the baby-cry noise.

A.5.2 Drilling

This test involves mechanical noise from an electric drill being used periodically, representing a high-intensity, narrow-band interference source. It evaluates the system's robustness to intervals of high-energy noise.

Table A.16 details the parameters used for the adaptive filters in the drilling noise scenario.

Table A.16: Tuning parameters for the adaptive filters in the drilling scenario.

Parameter	Value
Sampling rate	16 kHz
Filter length (LMS)	1
Filter length (NLMS)	1
Filter length (RLS)	1
Step size (LMS)	0.9
Step size (NLMS)	0.9
Forgetting factor (RLS)	0.9999

Figure A.34 displays the error signals, showing how effectively each algorithm suppresses

the drilling interference.

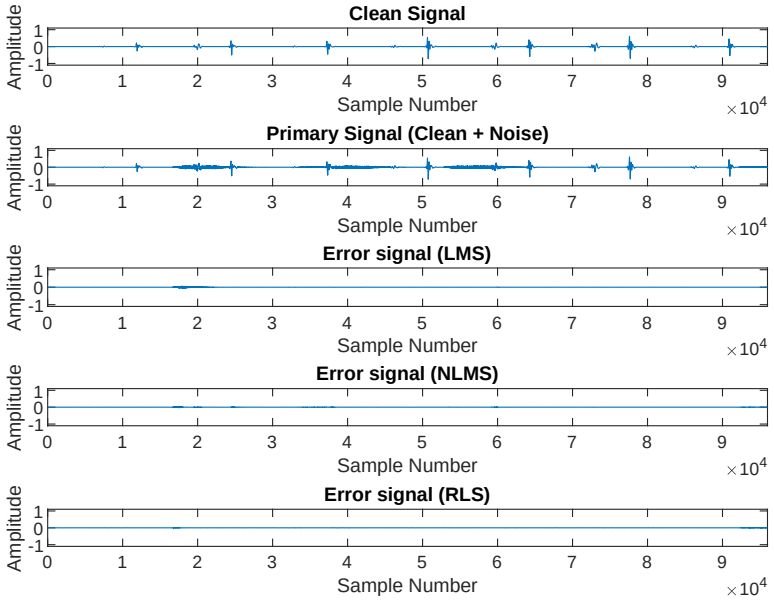


Figure A.34: Error signals - Heartbeat signal corrupted with Drilling Noise.

All three filters significantly reduce the drilling noise, with the RLS filter showing the cleanest error signal among them.

Figure A.35 illustrates how SNR varies with different filter lengths, helping identify optimal configurations.

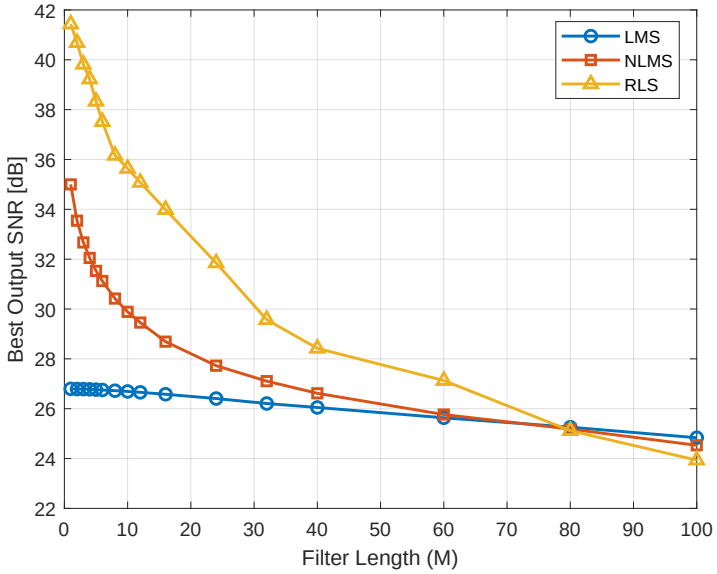


Figure A.35: Output SNR vs Filter Length - Heartbeat signal corrupted with Drilling.

RLS peaks at 41 dB, ahead of NLMS at 35 dB and LMS at 27 dB. As filter length increases, all three methods converge to about 24–25 dB at filter length 100, showing diminishing returns beyond that point. RLS remains the best performer overall, though its SNR falls more steeply than the others as the filter length grows.

Figure A.36 shows the Mel spectrograms post-filtering.

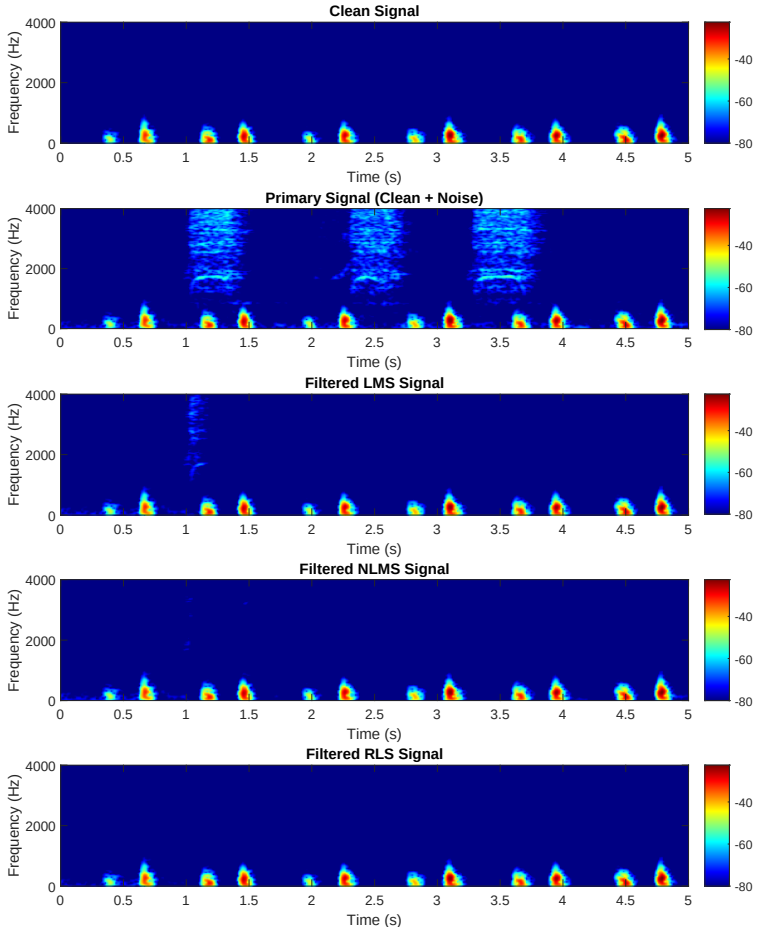


Figure A.36: Mel spectrogram of heartbeat recording with drilling noise.

The primary signal shows the heartbeat corrupted with drilling noise, visible as diffuse energy bands across 1000-4000 Hz. The RLS filter achieves superior noise reduction, closely resembling the clean signal, while LMS and NLMS leaves some residual noise at around the 1 s mark. All filters effectively preserve the heartbeat's fundamental frequency components while suppressing the drilling interference.

To visualise algorithm stability and speed, figure A.37 shows the convergence of error signals during adaptation.

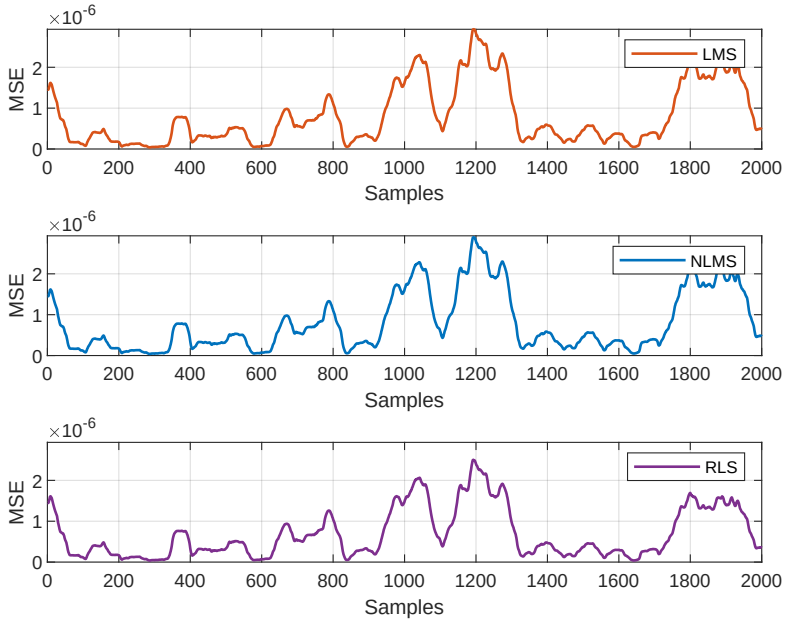


Figure A.37: Error Convergence Curves - Heartbeat with Drilling Noise.

From a practical standpoint, the MSE values remain consistently low—below 5×10^{-6} across the entire duration—indicating that all three algorithms achieve excellent performance.

Table A.17 summarises the resulting output SNR and corresponding improvements from each adaptive filter.

Table A.17: SNR improvement — Heartbeat with Drilling Noise.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-0.2283	-
LMS	26.7973	27.0255
NLMS	35.0002	35.2285
RLS	41.4302	41.6585

Table A.17 indicates that RLS yields the highest output SNR, making it the most effective filter for suppressing the drilling noise.

A.5.3 Hammering

This test evaluates the filtering system's ability to manage hammering, which features sharp and short components that can easily mask important signal details.

Table A.18 lists the tuning parameters for each adaptive algorithm under hammering noise conditions.

Table A.18: Tuning parameters for the adaptive filters in the hammering scenario.

Parameter	Value
Sampling rate	16 kHz
Filter length (LMS)	1
Filter length (NLMS)	1
Filter length (RLS)	1
Step size (LMS)	0.9
Step size (NLMS)	0.4
Forgetting factor (RLS)	0.9999

Figure A.38 illustrates the error signals after adaptive filtering, reflecting how well the methods suppress impulsive hammer strikes.

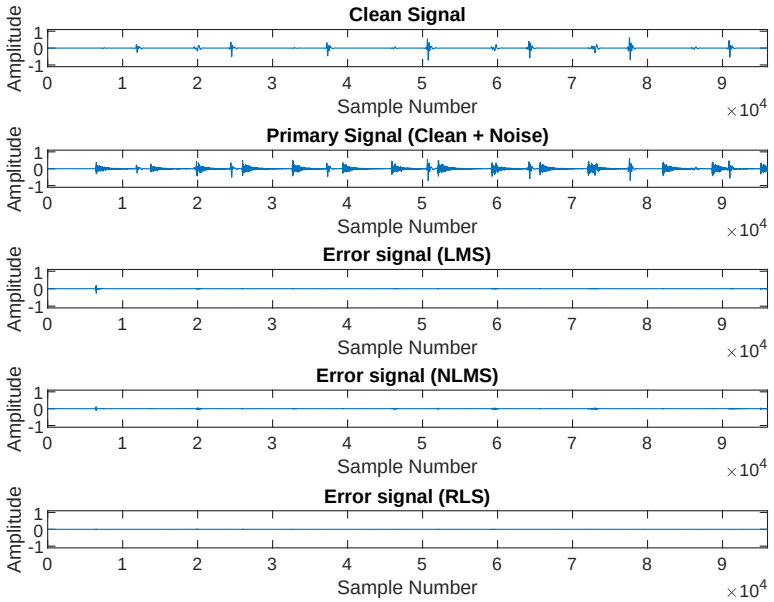


Figure A.38: Error signals - Heartbeat signal corrupted with Hammering Noise.

All three filters significantly reduce the drilling noise, with the RLS filter showing the cleanest error signal among them.

SNR variation with respect to filter length is plotted in figure A.39, providing insights into filter size optimisation.

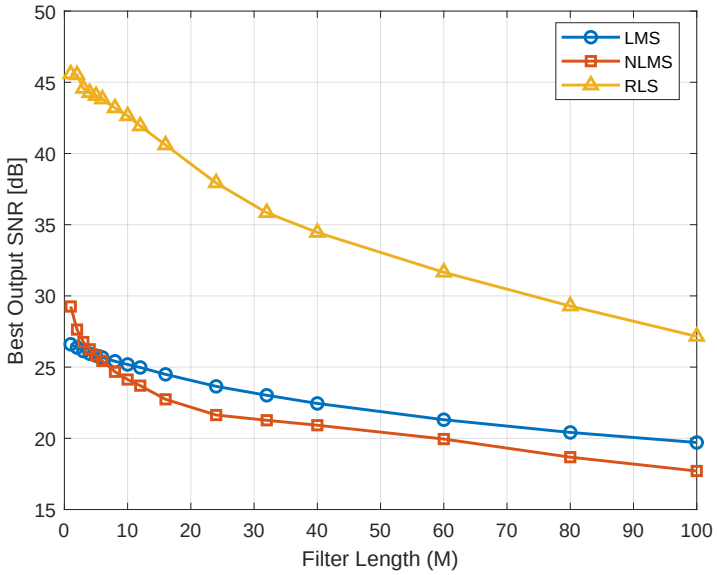


Figure A.39: Output SNR vs Filter Length - Heartbeat signal corrupted with Hammering.

RLS demonstrates significantly superior performance across all filter lengths, starting at 46 dB and declining to 27 dB at filter length 100. LMS and NLMS show similar but much lower performance 26-29 dB initially, with NLMS performing slightly worse than LMS as filter length increases. Unlike previous results with drilling noise, the algorithms maintain distinct performance differences even at maximum filter length, with RLS consistently outperforming the others by approximately 8-10 dB.

Figure A.40 displays the Mel spectrogram of the noisy recording, revealing how hammer impacts distort the signal's frequency domain.

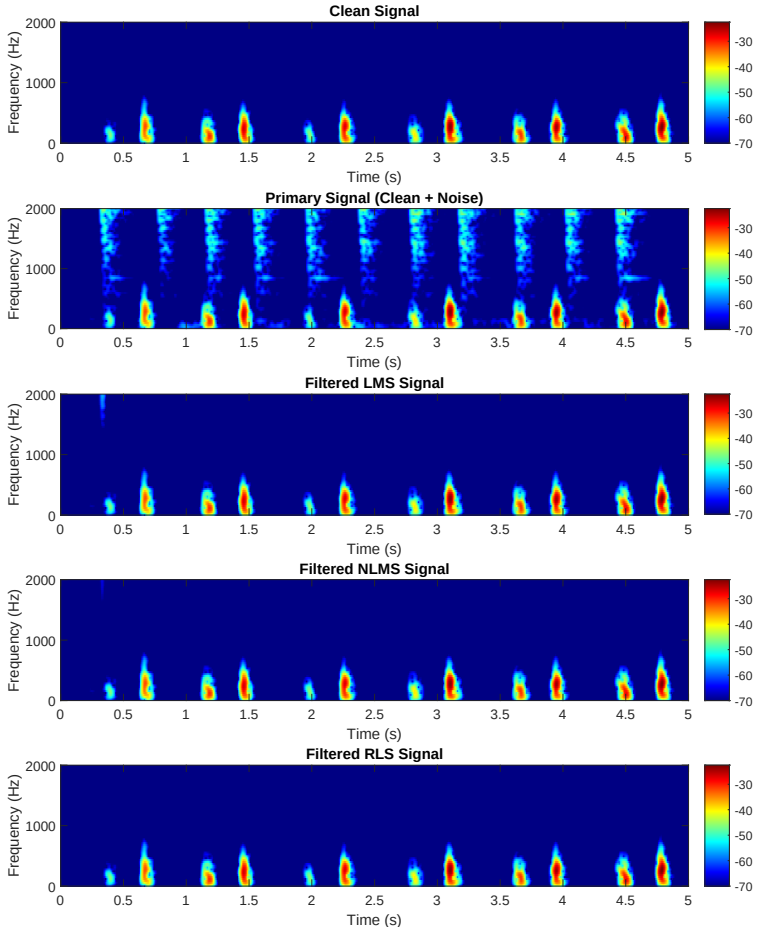


Figure A.40: Mel spectrogram of heartbeat recording with hammering in the background.

The primary signal combines the heartbeat with hammering noise, seen as vertical bands extending from 0-2000 Hz. RLS removes most of this interference, yielding a spectrogram that closely matches the clean reference, whereas LMS and NLMS leave residual noise near 0.4 s. Despite these differences, all three filters retain the heartbeat's fundamental low-frequency components while reducing the hammering noise.

To evaluate convergence behavior, figure A.41 shows the learning curves for each algorithm.

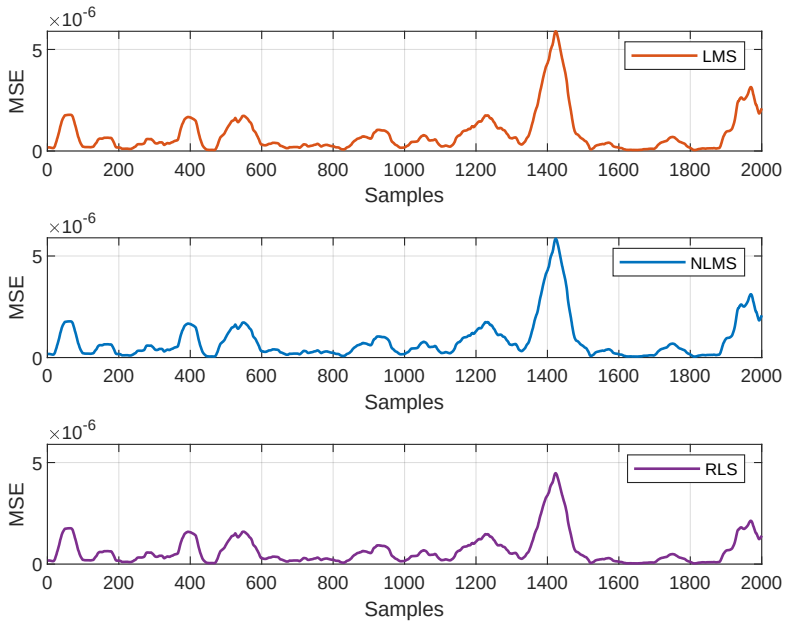


Figure A.41: Error Convergence Curves - Heartbeat with Hammering Noise.

From a practical standpoint, the MSE values remain consistently low—below 5×10^{-6} across the entire duration—indicating that all three algorithms achieve excellent performance.

Table A.19 reports SNR values before and after filtering, comparing the effectiveness of LMS, NLMS, and RLS.

Table A.19: SNR improvement — Heartbeat with Hammering Noise.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-1.4565	-
LMS	26.6031	28.0597
NLMS	29.2561	30.7127
RLS	45.5780	47.0346

Table A.19 shows that RLS gives the highest output SNR, making it the most effective filter for reducing the hammering noise.

A.5.4 Speech

In this test case, background speech interferes with heartbeat recordings. This introduces challenges due to the wideband, non-stationary nature of speech that may overlap spectrally with the target signal.

Table A.20 provides the algorithm parameters tailored for this speech interference scenario.

Table A.20: Tuning parameters for the adaptive filters in the speech scenario.

Parameter	Value
Sampling rate	16 kHz
Filter length (LMS)	1
Filter length (NLMS)	1
Filter length (RLS)	1
Step size (LMS)	0.2
Step size (NLMS)	0.025
Forgetting factor (RLS)	0.9999

The filtered error signals are plotted in figure A.42, indicating how well each method reduces spoken interference.

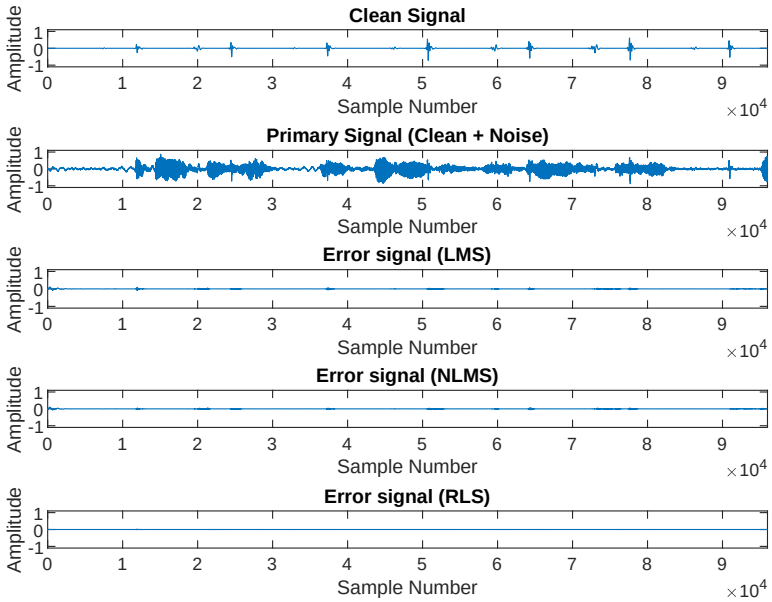


Figure A.42: Error signals - Heartbeat signal corrupted with Speech Noise.

All three filters significantly reduce the speech noise, with the RLS filter showing the cleanest error signal among them.

Figure A.43 shows the effect of filter length on SNR for each method in the presence of background speech.

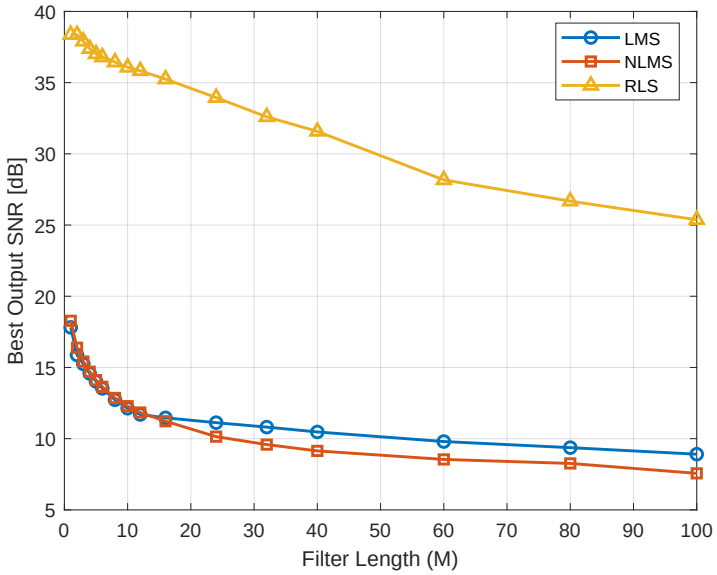


Figure A.43: Output SNR vs Filter Length - Heartbeat signal corrupted with Speech.

RLS demonstrates significantly superior performance across all filter lengths, starting at 38 dB and declining to 25 dB at filter length 100. LMS and NLMS show similar but much lower performance 17-18 dB initially, with NLMS performing slightly worse than LMS as filter length increases. RLS consistently outperformes the others by approximately 20 dB.

The Mel spectrogram in figure A.44 shows how background speech masks the heartbeat in both time and frequency domains.

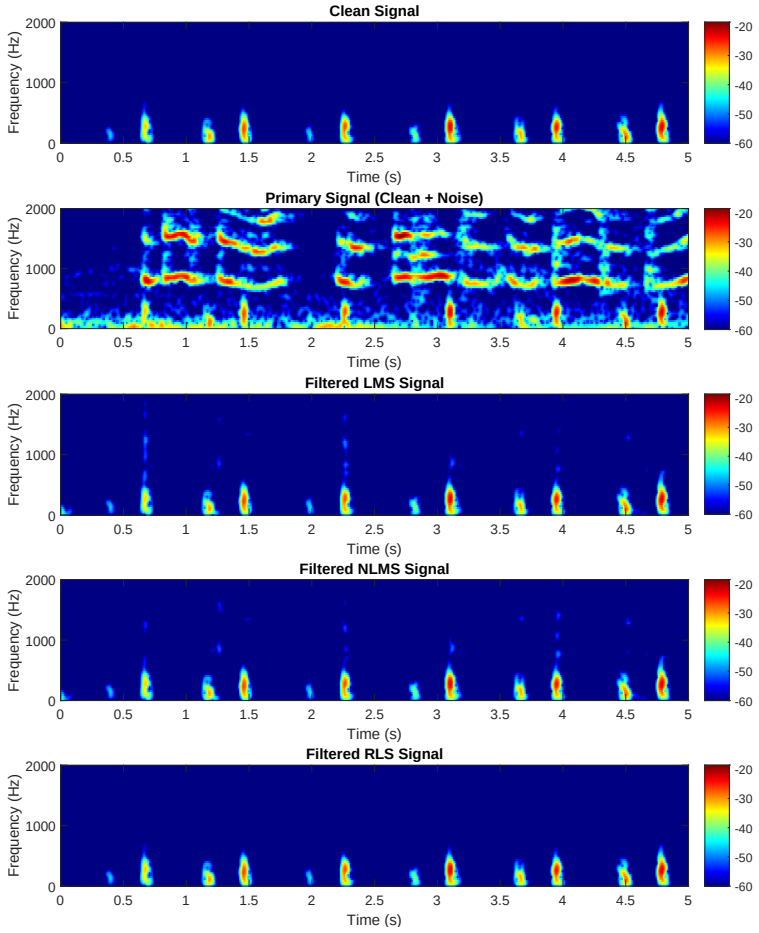


Figure A.44: Mel spectrogram of heartbeat recording with background speech.

The primary signal consists of the heartbeat mixed with speech noise, as evident in the spectrogram. Among the three algorithms, RLS performs the best, effectively removing most of the interference and producing a spectrogram that closely resembles the clean reference. In contrast, LMS and NLMS leave behind noticeable remnants of the speech noise around each heartbeat. Nevertheless, all three filters successfully preserve the heartbeat's fundamental low-frequency components while significantly attenuating the unwanted speech interference.

The convergence curves for the three adaptive methods in this speech scenario are shown in figure A.45.

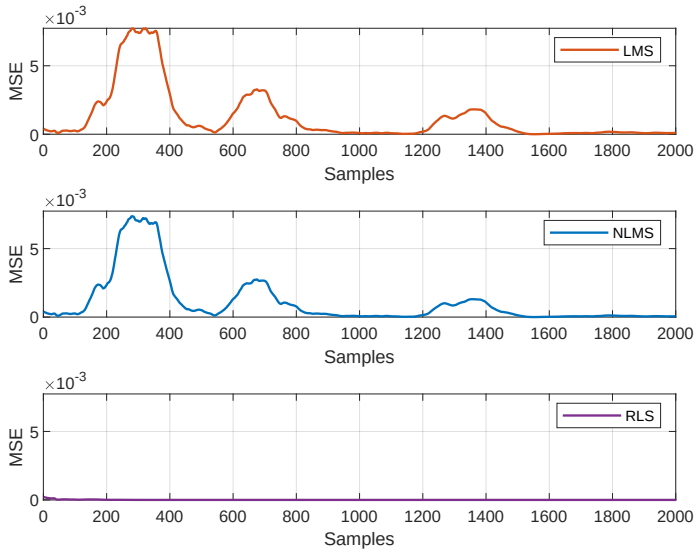


Figure A.45: Error Convergence Curves - Heartbeat with Speech Noise.

From a practical standpoint, the MSE values remain consistently low—below 5×10^{-3} across the entire duration—indicating that all three algorithms achieve excellent performance. Although the RLS algorithm shows better performance.

Table A.21 details the SNR improvements obtained using each method.

Table A.21: SNR improvement — Heartbeat with Speech Noise.

Filtering Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-9.2980	-
LMS	17.8150	27.1130
NLMS	18.2781	27.5762
RLS	38.3709	47.6689

Table A.21 shows that RLS gives the highest output SNR, making it the most effective filter for reducing the speech.

A.5.5 Summary Table

Table A.22 provides a comprehensive summary of SNR improvements achieved by each algorithm across all four noise types.

Table A.22: Performance summary across all noise types (best performer in bold).

Noise Type	Input SNR (dB)	Output SNR (dB)		
		LMS	NLMS	RLS
Ambient Noise	-8.8182	13.9755	14.0150	**26.2265**
Baby Crying	-8.9011	24.1021	24.8095	**46.2584**
Drilling	-0.2283	26.7973	35.0002	**41.4302**
Hammering	-1.4565	26.6031	29.2561	**45.5780**
Speech	-9.2980	17.8150	18.2781	**38.3709**

As evident from table A.22, the adaptive filtering system effectively enhances heartbeat signals across all noise environments tested. The RLS algorithm consistently outperforms both LMS and NLMS in all five noise scenarios, with particular strength in handling baby crying noise where it achieves an exceptional 46 dB output SNR. The NLMS filter slightly outperforms the LMS through all tests.

These results confirm that adaptive filtering provides an effective solution for heart sound denoising across diverse noisy environments.

A.6 Additional Adaptive Filter Results For Fixed Parameters on Experimental Data

This appendix provides detailed analysis of the adaptive filter performance on the remaining four noise types: Baby Crying, Drilling, Hammering and Speech using the fixed parameters from A.23.

Table A.23: Filter parameters based on clinical testing

Parameter	Value
Sampling rate	16 kHz
Filter length (LMS)	8
Filter length (NLMS)	8
Filter length (RLS)	8
Step size (LMS)	0.3
Step size (NLMS)	0.03
Forgetting factor (RLS)	0.9999

A.6.1 Baby Crying

This scenario models an infant crying in close proximity to the recording device, introducing highly non-stationary, mid- to high-frequency bursts.

Figure A.46 shows a comparison between the clean signal, the primary signal, and the error signals produced by the LMS, NLMS, and RLS filters.

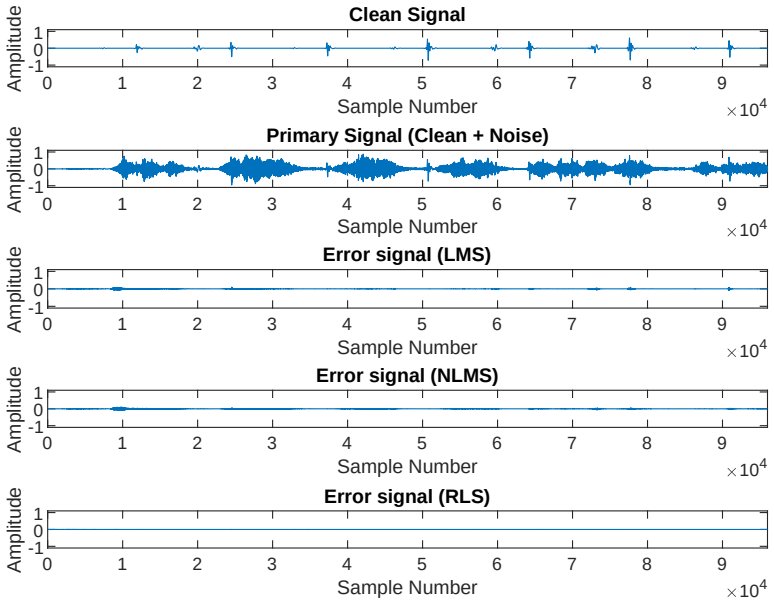


Figure A.46: Error signals - Heartbeat signal corrupted with Baby Crying Noise.

All three adaptive filters substantially reduce the noise showing near zero error signal plots.

Figure A.47 shows the Mel spectrograms post-filtering.

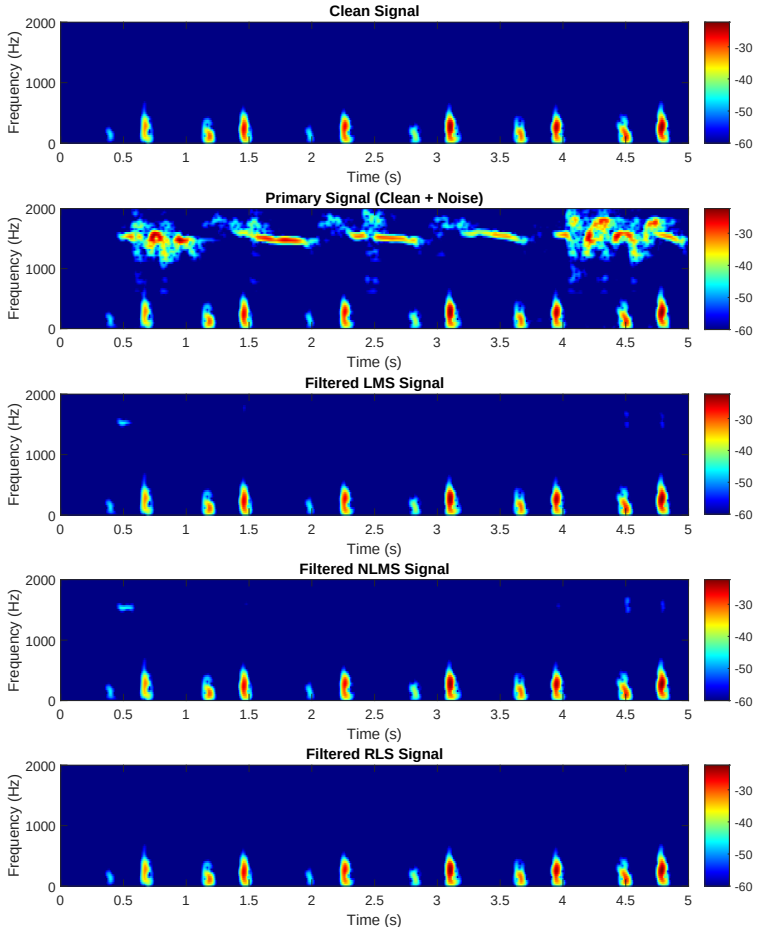


Figure A.47: Mel spectrogram of heartbeat recording with baby crying in the background.

Among all three adaptive filters, the RLS filter produces the cleanest spectrogram, closely resembling the clean signal. The LMS filter also performs well but retains a faint spectral remnant, particularly around 1500 Hz at 0.5 s of the signal. The NLMS filter also retains a faint spectral remnant at the same spot, though to a slightly lesser extent than LMS.

The convergence behavior of each adaptive algorithm is illustrated in figure A.48, providing valuable insight into both stability and adaptation speed.

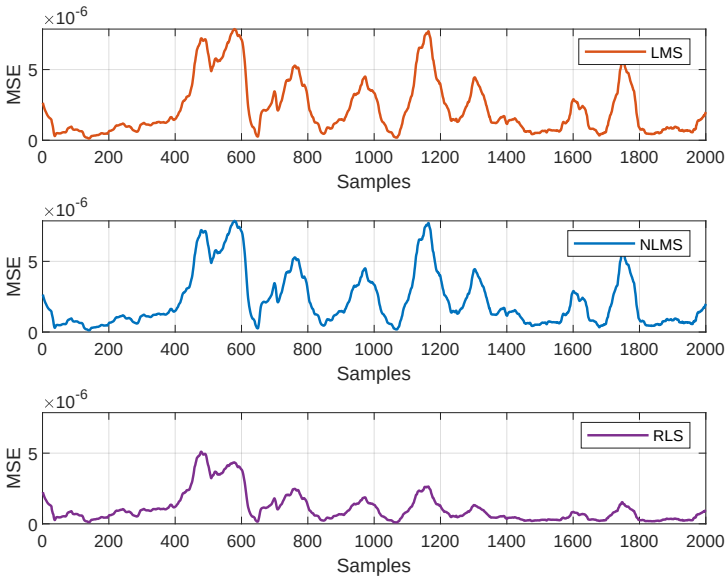


Figure A.48: Error convergence curves for heartbeat signal with baby crying noise.

From a practical standpoint, the MSE values remain consistently low—below 5×10^{-6} across the entire duration—indicating that all three algorithms achieve excellent performance.

A.6.2 Drilling

This test involves mechanical noise from an electric drill being used periodically, representing a high-intensity, narrow-band interference source. It evaluates the system's robustness to intervals of high-energy noise.

Figure A.49 displays the error signals, showing how effectively each algorithm suppresses the drilling interference.

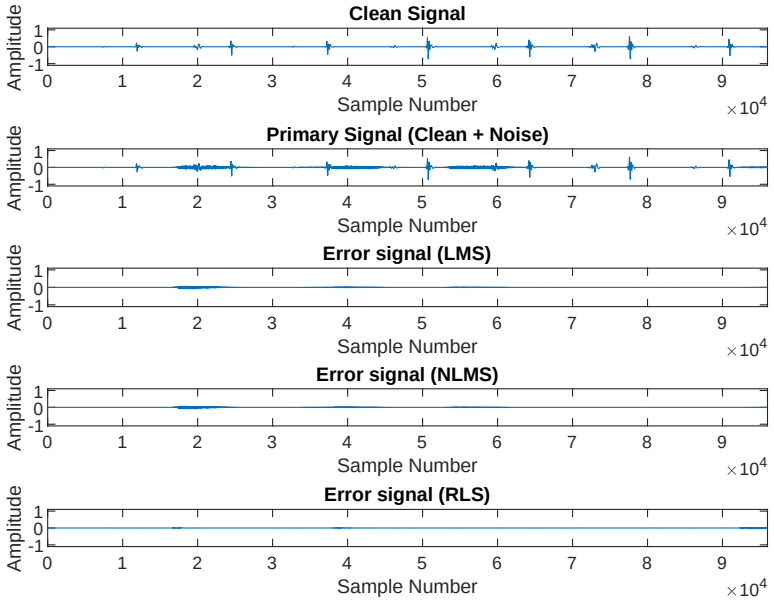


Figure A.49: Error signals - Heartbeat signal corrupted with Drilling Noise.

All three filters significantly reduce the drilling noise, with the RLS filter showing the cleanest error signal among them.

Figure A.50 shows the Mel spectrograms post-filtering.

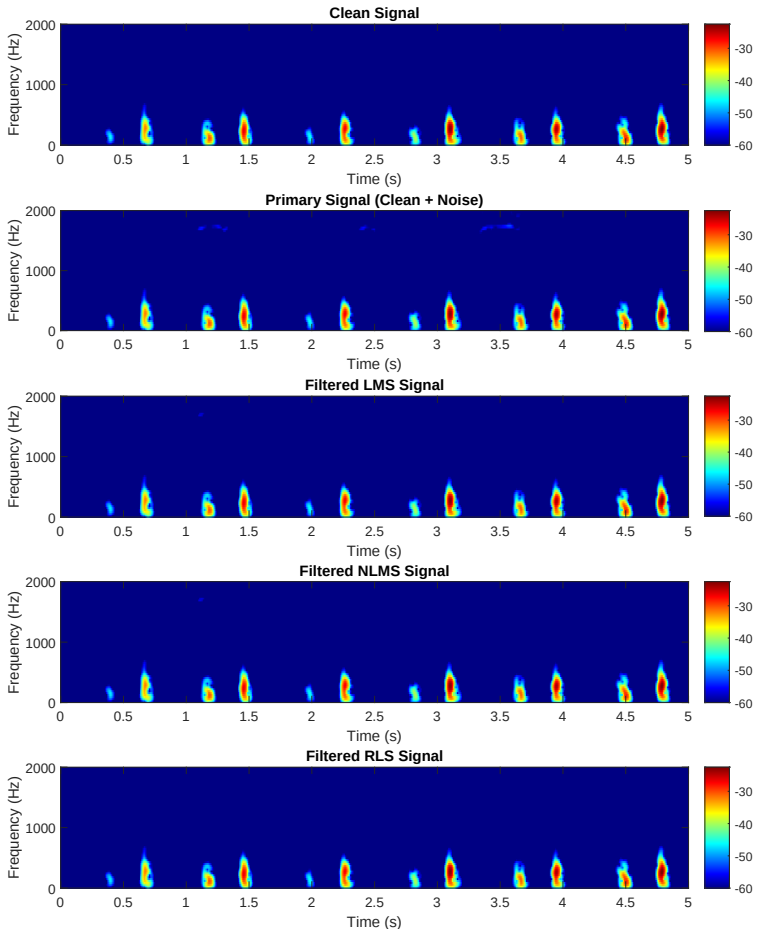


Figure A.50: Mel spectrogram of heartbeat recording with drilling noise.

The primary signal shows the heartbeat corrupted with drilling noise, visible as diffuse energy bands across 1000-4000 Hz. The RLS filter achieves superior noise reduction, closely resembling the clean signal, while LMS and NLMS leaves some residual noise at around the 1 s mark. All filters effectively preserve the heartbeat's fundamental frequency components while suppressing the drilling interference.

To visualise algorithm stability and speed, figure A.51 shows the convergence of error signals during adaptation.

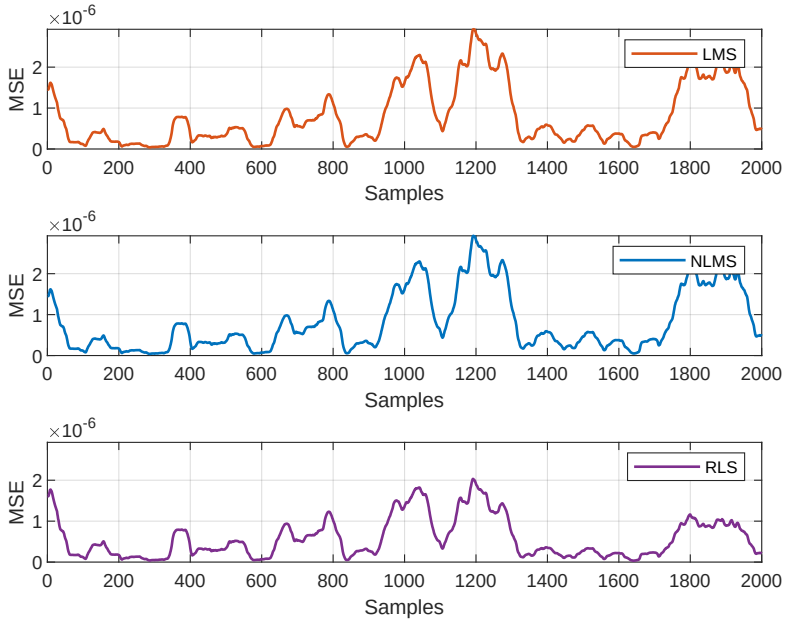


Figure A.51: Error Convergence Curves - Heartbeat with Drilling Noise.

From a practical standpoint, the MSE values remain consistently low—below 5×10^{-6} across the entire duration—indicating that all three algorithms achieve excellent performance.

A.6.3 Hammering

This test evaluates the filtering system's ability to manage hammering, which features sharp and short components that can easily mask important signal details.

Figure A.52 illustrates the error signals after adaptive filtering, reflecting how well the methods suppress impulsive hammer strikes.

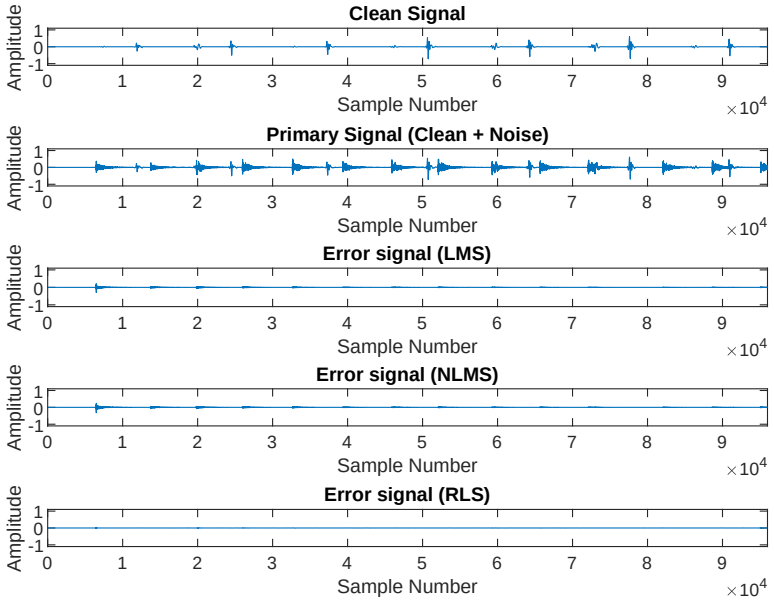


Figure A.52: Error signals - Heartbeat signal corrupted with Hammering Noise.

All three filters significantly reduce the drilling noise, with the RLS filter showing the cleanest error signal among them.

Figure A.53 displays the Mel spectrogram of the noisy recording, revealing how hammer impacts distort the signal's frequency domain.

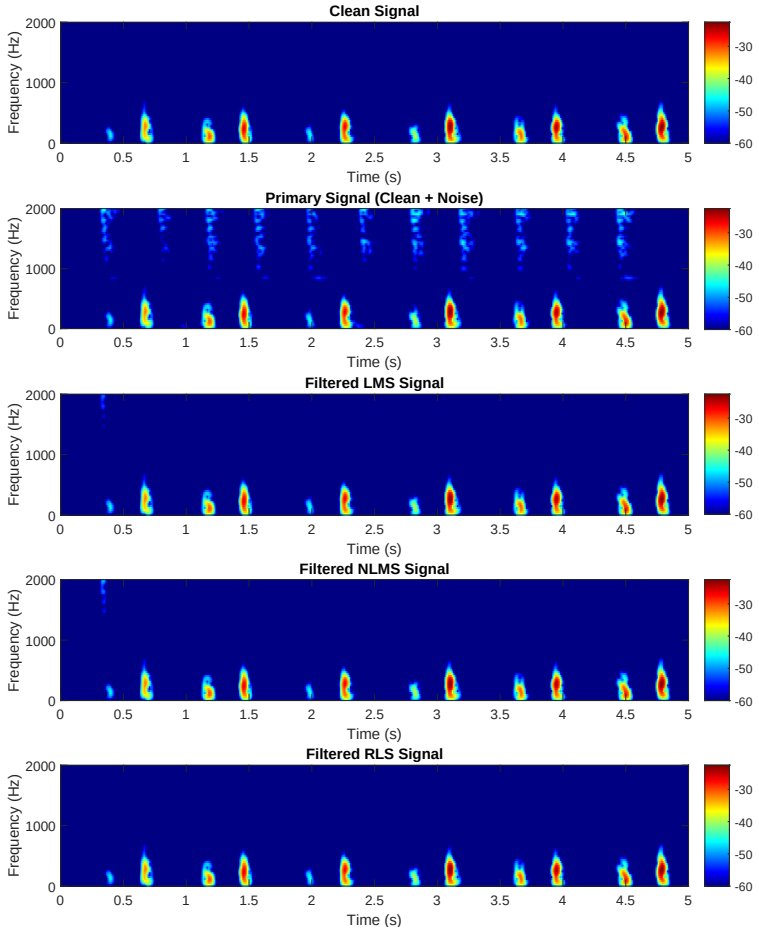


Figure A.53: Mel spectrogram of heartbeat recording with hammering in the background.

The primary signal combines the heartbeat with hammering noise, seen as vertical bands extending from 0-2000 Hz. RLS removes most of this interference, yielding a spectrogram that closely matches the clean reference, whereas LMS and NLMS leave residual noise near 0.4 s. Despite these differences, all three filters retain the heartbeat's fundamental low-frequency components while reducing the hammering noise.

To evaluate convergence behavior, figure A.54 shows the learning curves for each algorithm.

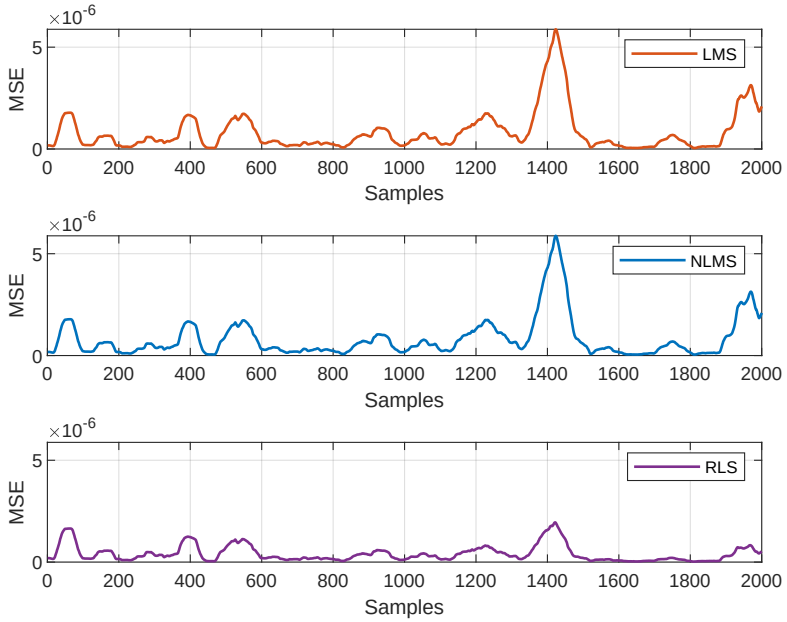


Figure A.54: Error Convergence Curves - Heartbeat with Hammering Noise.

From a practical standpoint, the MSE values remain consistently low—below 5×10^{-6} across the entire duration—indicating that all three algorithms achieve excellent performance.

A.6.4 Speech

In this test case, background speech interferes with heartbeat recordings. This introduces challenges due to the wideband, non-stationary nature of speech that may overlap spectrally with the target signal.

The filtered error signals are plotted in figure A.55, indicating how well each method reduces spoken interference.

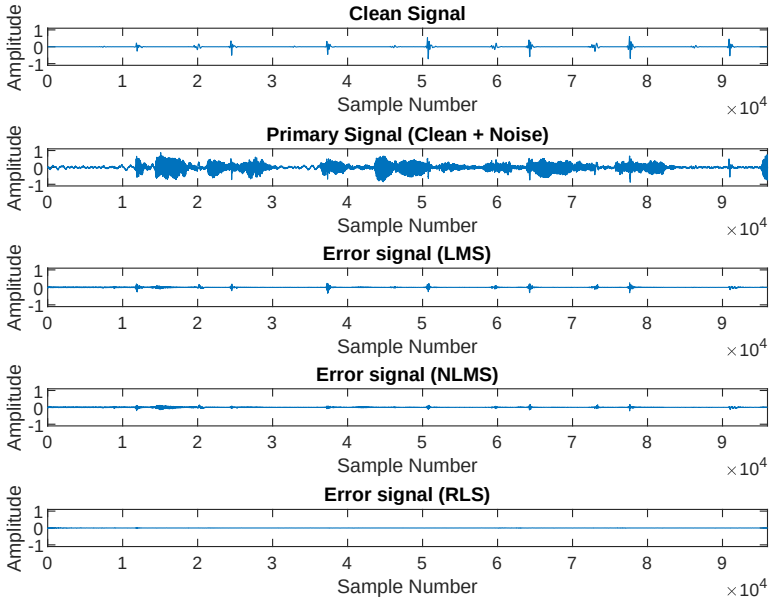


Figure A.55: Error signals - Heartbeat signal corrupted with Speech Noise.

All three filters significantly reduce the speech noise, with the RLS filter showing the cleanest error signal among them.

The Mel spectrogram in figure A.56 shows how background speech masks the heartbeat in both time and frequency domains.

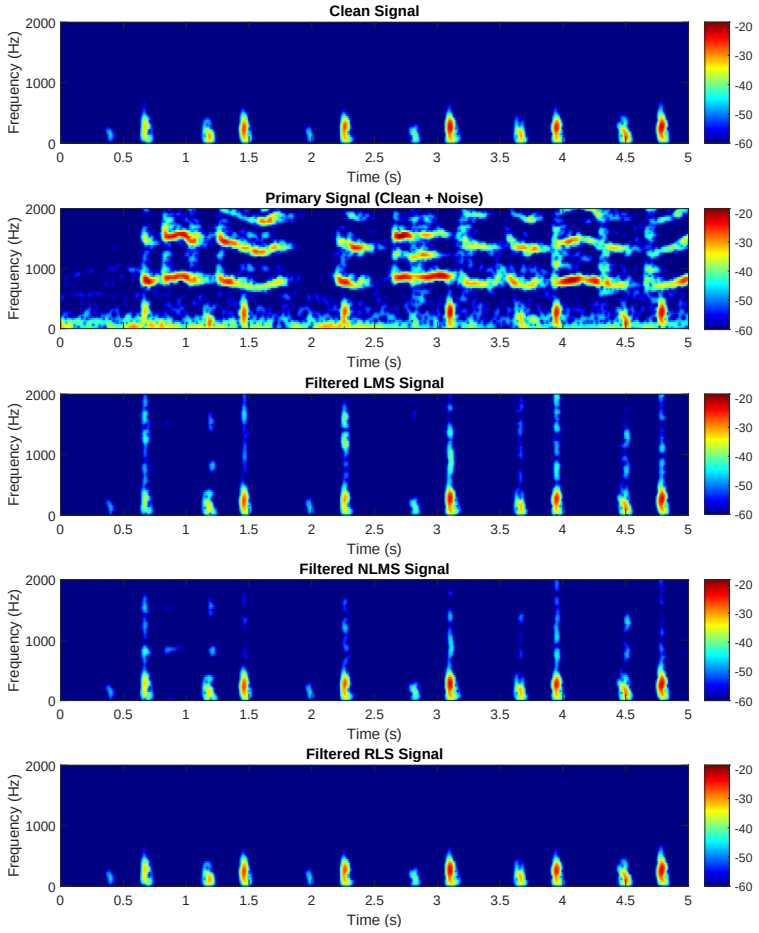


Figure A.56: Mel spectrogram of heartbeat recording with background speech.

The primary signal consists of the heartbeat mixed with speech noise, as evident in the spectrogram. Among the three algorithms, RLS performs the best, effectively removing most of the interference and producing a spectrogram that closely resembles the clean reference. In contrast, LMS and NLMS leave behind noticeable remnants of the speech noise around each heartbeat. Nevertheless, all three filters successfully preserve the heartbeat's fundamental low-frequency components while significantly attenuating the unwanted speech interference.

The convergence curves for the three adaptive methods in this speech scenario are shown in figure A.57.

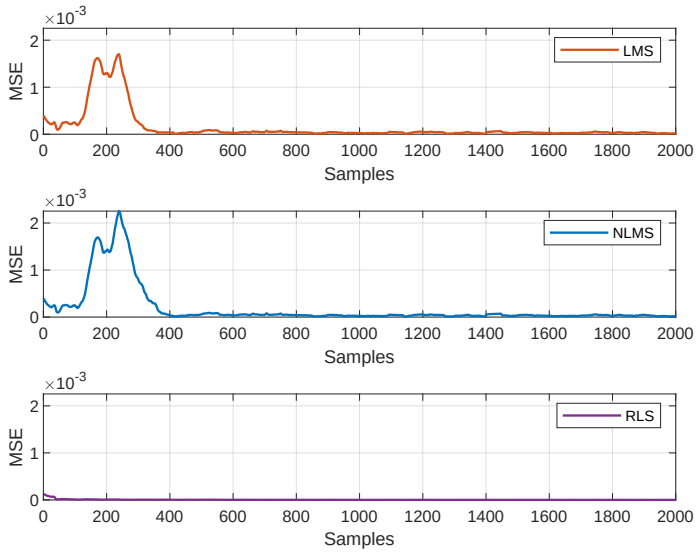


Figure A.57: Error Convergence Curves - Heartbeat with Speech Noise.

From a practical standpoint, the MSE values remain consistently low—below 5×10^{-3} across the entire duration—indicating that all three algorithms achieve excellent performance. Although the RLS algorithm shows better performance.

A.6.5 Summary Table

Table A.24 provides a comprehensive summary of SNR improvements achieved by each algorithm across all four noise types for Adaptive Filter Results For Fixed Parameters on Experimental Data.

Table A.24: Performance summary across all noise types (best performer in bold) - Adaptive Filter Results For Fixed Experimental Data

Noise Type	**Input SNR (dB)**	**Output SNR (dB)**		
		LMS	**NLMS**	**RLS**
Ambient Noise	-8.8182	5.0992	6.9643	**25.5552**
Baby Crying	-8.9011	20.2224	18.1210	**44.9098**
Drilling	-0.2283	22.0850	21.7362	**36.1780**
Hammering	-1.4565	21.9232	19.7391	**43.2578**
Speech	-9.2980	9.5512	12.5831	**36.4451**

A.7 Additional FastICA Results For On Experimental Data

This appendix presents the detailed FastICA separation results for noise types not fully discussed in the main text. For each noise type, we show the error signals, Mel spectrograms, and SNR improvements.

A.7.1 Ambient Noise

This case investigates FastICA's ability to separate heartbeat sounds contaminated with typical ambient hospital noise.

Figure A.58 compares the original clean heartbeat signal, the noisy input mixtures, and the resulting error signals after FastICA.

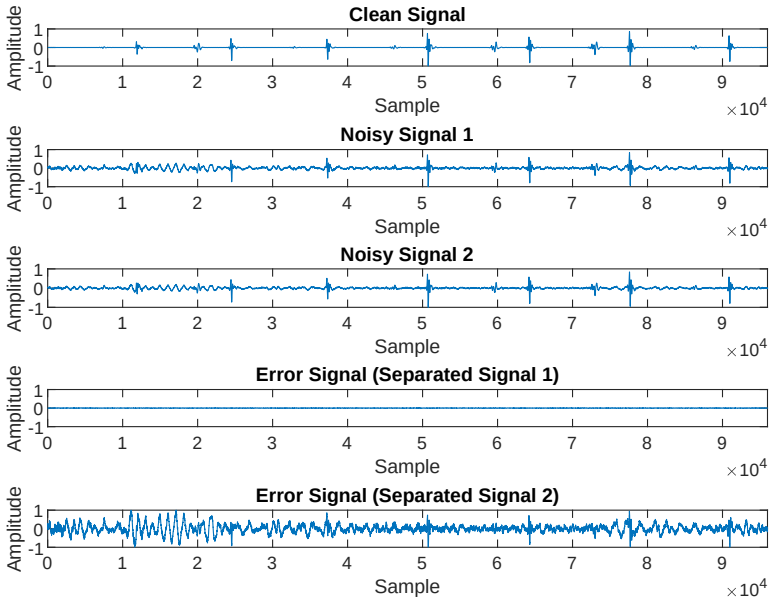


Figure A.58: FastICA separation of heartbeat signal from ambient noise: original clean signal, noisy inputs, and error signals.

FastICA effectively isolates the heartbeat component, as seen by the low error in Separated Signal 1, while Separated Signal 2 captures the ambient noise.

The Mel spectrogram in figure A.59 highlights the frequency content and confirms the removal of noise.

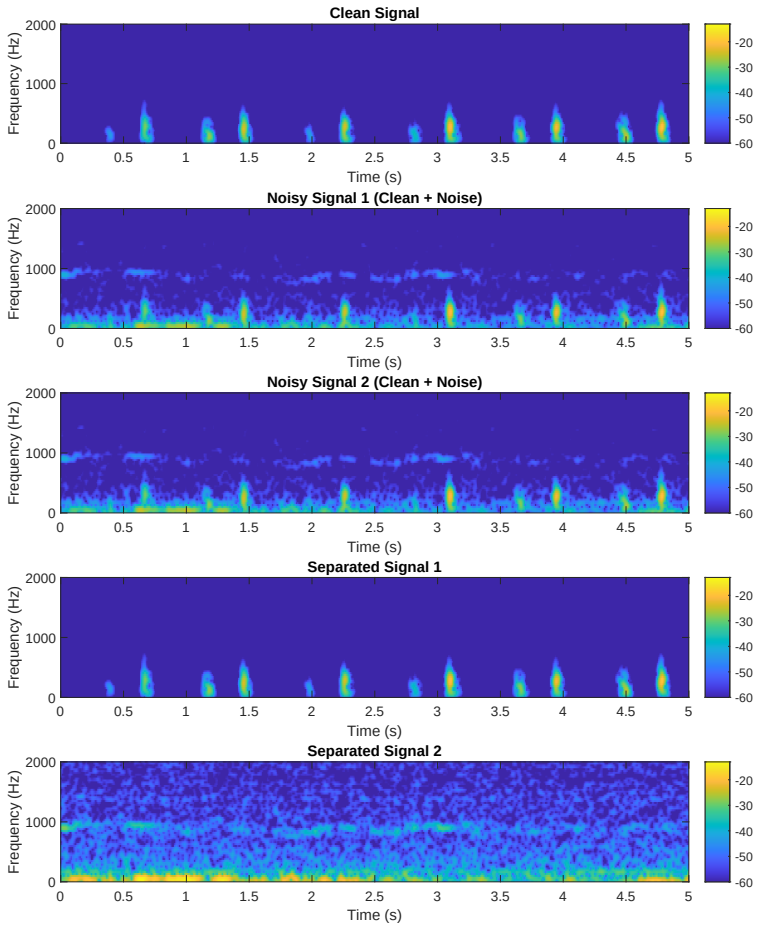


Figure A.59: Mel Spectrograms showing FastICA separation for heartbeat corrupted by ambient noise.

FastICA effectively separates the underlying sources from the observed noisy mixture. Separated Signal 1 reveals a clear reconstruction of the heartbeat, whereas Separated Signal 2 captures primarily the noise.

A.7.2 Baby Crying

This subsection analyses separation performance when heartbeat recordings are contaminated by baby crying noise, a highly non-stationary source.

Figure A.60 shows the FastICA error signals alongside input and clean signals.

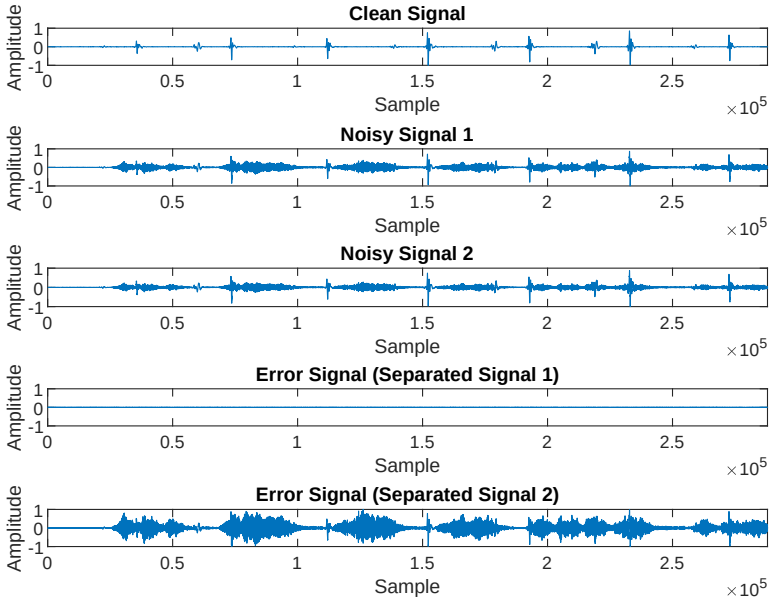


Figure A.60: FastICA separation of heartbeat from baby crying noise: error signals comparison.

FastICA effectively isolates the heartbeat component, as seen by the low error in Separated Signal 1, while Separated Signal 2 captures the baby crying.

The Mel spectrogram, figure A.61, shows how FastICA successfully recovers heartbeat components while isolating the baby crying noise.

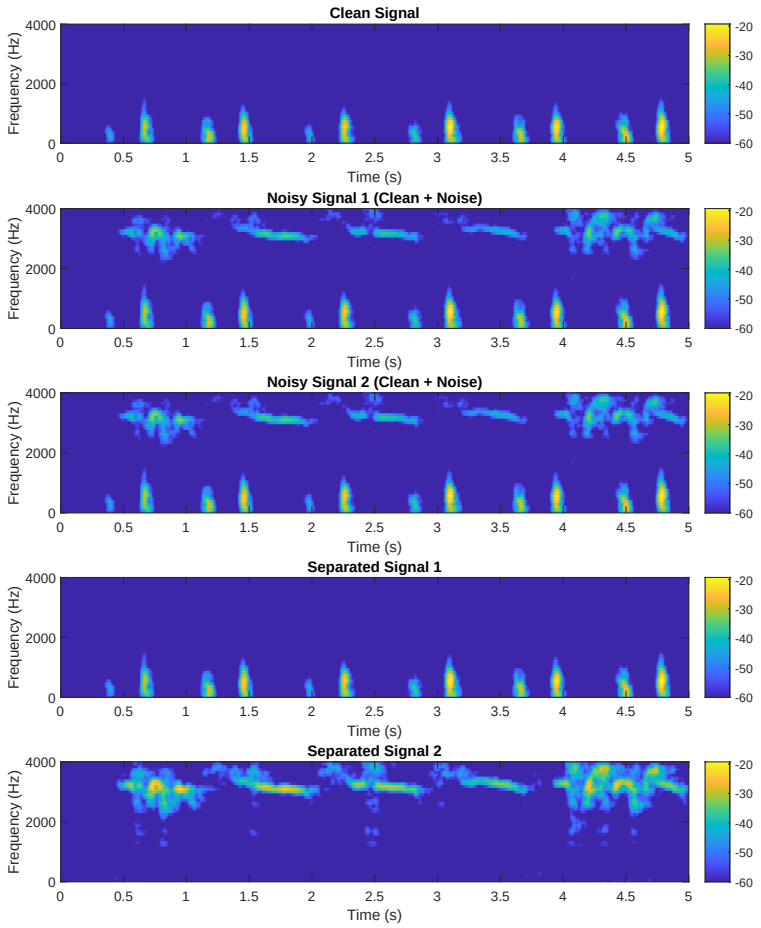


Figure A.61: Mel Spectrograms for FastICA separation of heartbeat corrupted by baby crying noise.

FastICA effectively separates heartbeat signal from the noise leaving us with Separated Signal 1 which is a clear reconstruction of the heartbeat with no noise, whereas Separated Signal 2 captures primarily the noise.

Table A.25 presents the corresponding SNR improvements.

Table A.25: SNR improvement — Heartbeat with Baby Crying Noise.

Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-3.20	-
FastICA (Separated Signal 1)	42.50	45.7
FastICA (Separated Signal 2)	-2.10	1.10

A.7.3 Drilling

Here, the effect of construction noise from drilling on heartbeat signal separation is evaluated.

Figure A.62 illustrates the error signals after FastICA processing.

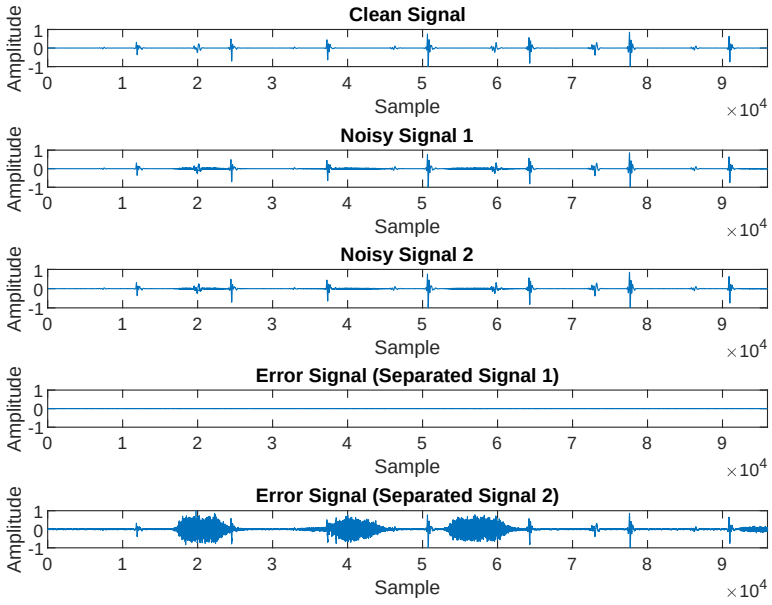


Figure A.62: FastICA error signals for heartbeat corrupted by drilling noise.

FastICA effectively isolates the heartbeat component, as seen by the low error in Separated Signal 1, while Separated Signal 2 captures the drilling noise.

The Mel spectrogram (figure A.63) confirms successful heartbeat extraction.

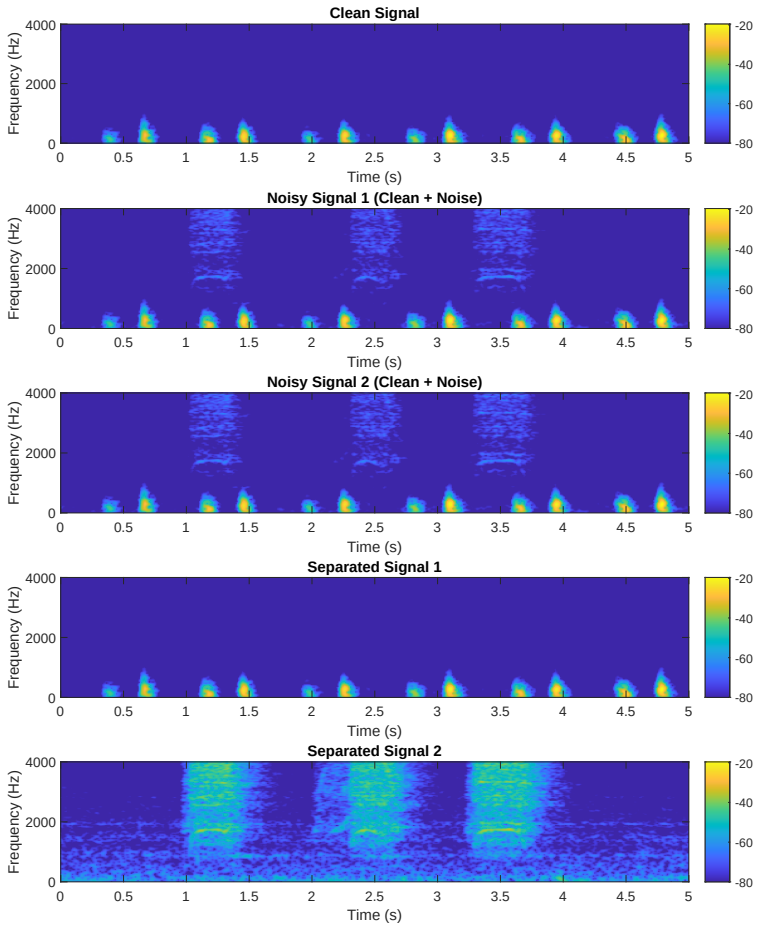


Figure A.63: Mel Spectrograms after FastICA separation for heartbeat with drilling noise.

FastICA effectively separates the heartbeat from drilling noise, producing Separated Signal 1 as a clear, noise-free heartbeat and Separated Signal 2 as primarily the isolated drilling noise.

SNR improvements are summarised in table A.26.

Table A.26: SNR improvement — Heartbeat with Drilling Noise.

Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-1.75	-
FastICA (Separated Signal 1)	46.30	48.05
FastICA (Separated Signal 2)	-4.20	-2.45

A.7.4 Hammering

This subsection examines the FastICA performance when heartbeat signals are corrupted by hammering noise.

Figure A.64 presents the error signal comparison.

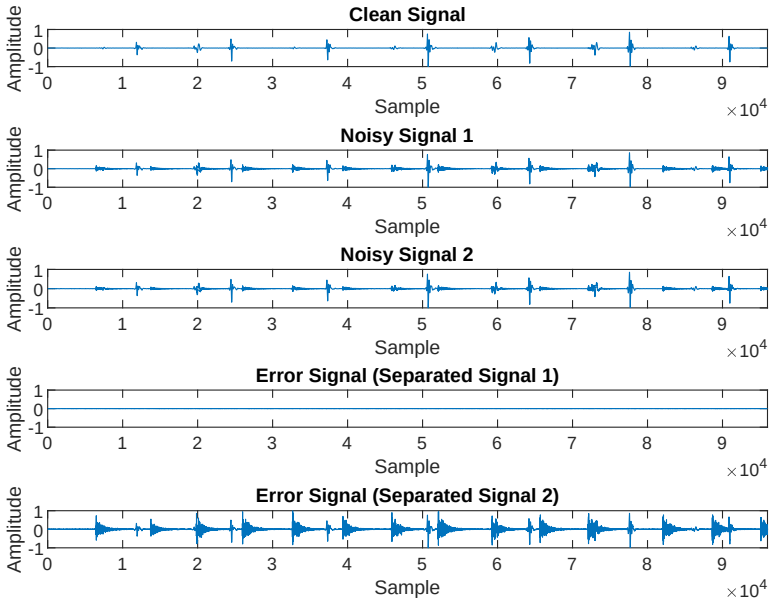


Figure A.64: Error signals for heartbeat separation from hammering noise via FastICA.

FastICA effectively isolates the heartbeat, evident from the low error in Separated Signal 1, while Separated Signal 2 contains the hammering noise.

The Mel spectrogram shown in figure A.65 demonstrates the clear separation of heartbeat and hammering noise components.

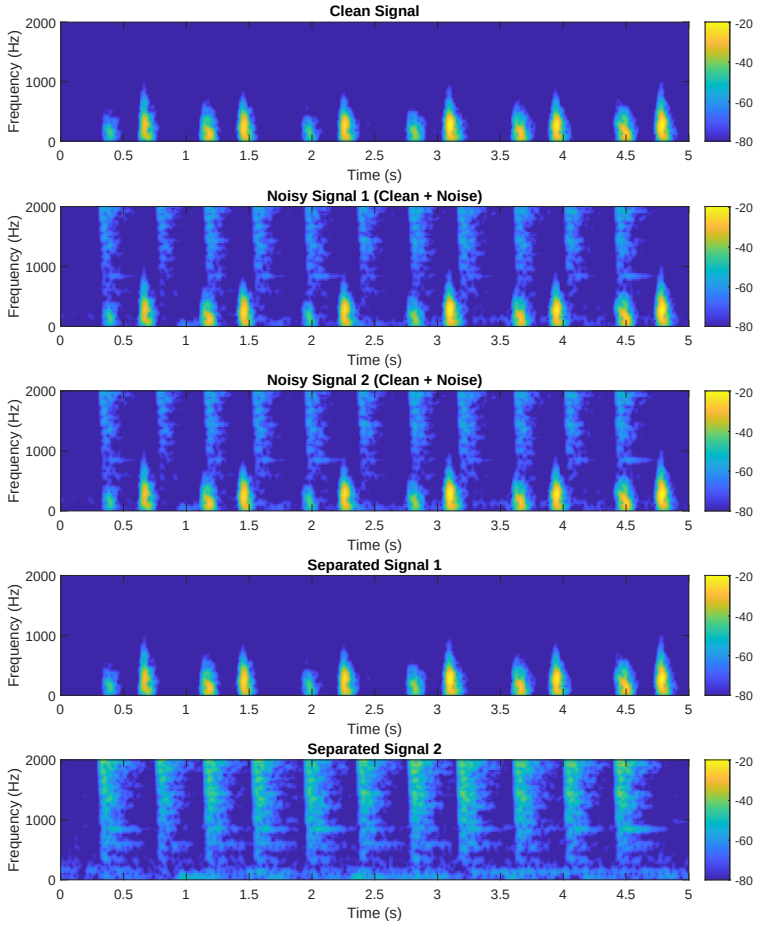


Figure A.65: Mel Spectrograms for FastICA separated signals from hammering noise corrupted heartbeat.

FastICA effectively separates the heartbeat from noise, producing Separated Signal 1 as a clear, noise-free heartbeat and Separated Signal 2 as primarily the isolated noise.

Table A.27 quantifies SNR improvements.

Table A.27: SNR improvement — Heartbeat with Hammering Noise.

Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-2.85	-
FastICA (Separated Signal 1)	43.90	46.75
FastICA (Separated Signal 2)	-3.95	-1.10

A.7.5 Speech

The final experimental noise case involves separation of heartbeat signals corrupted by conversational speech.

Figure A.66 shows error signals obtained after FastICA separation.

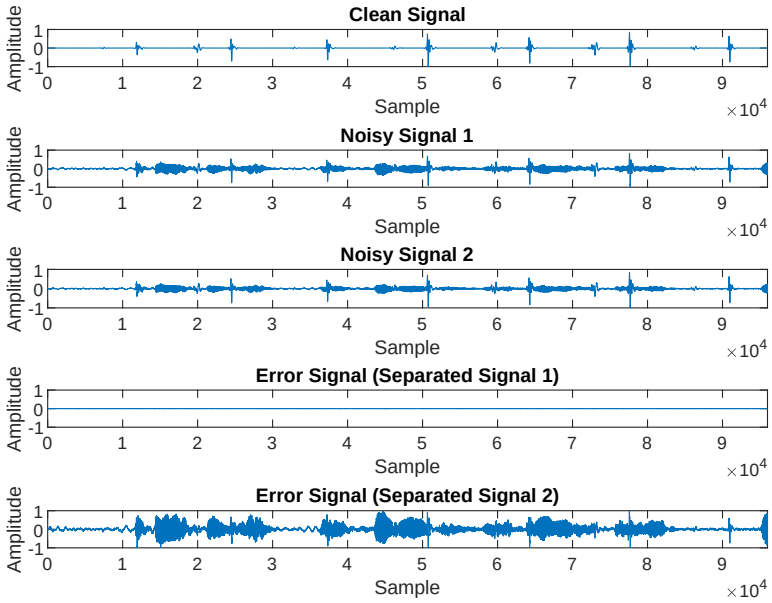


Figure A.66: Error signals after FastICA separation of heartbeat signal contaminated by speech.

FastICA effectively isolates the heartbeat, as indicated by the low error in Separated Signal 1, while Separated Signal 2 predominantly contains the speech.

The Mel spectrogram in figure A.67 illustrates the separation of heartbeat and speech components.

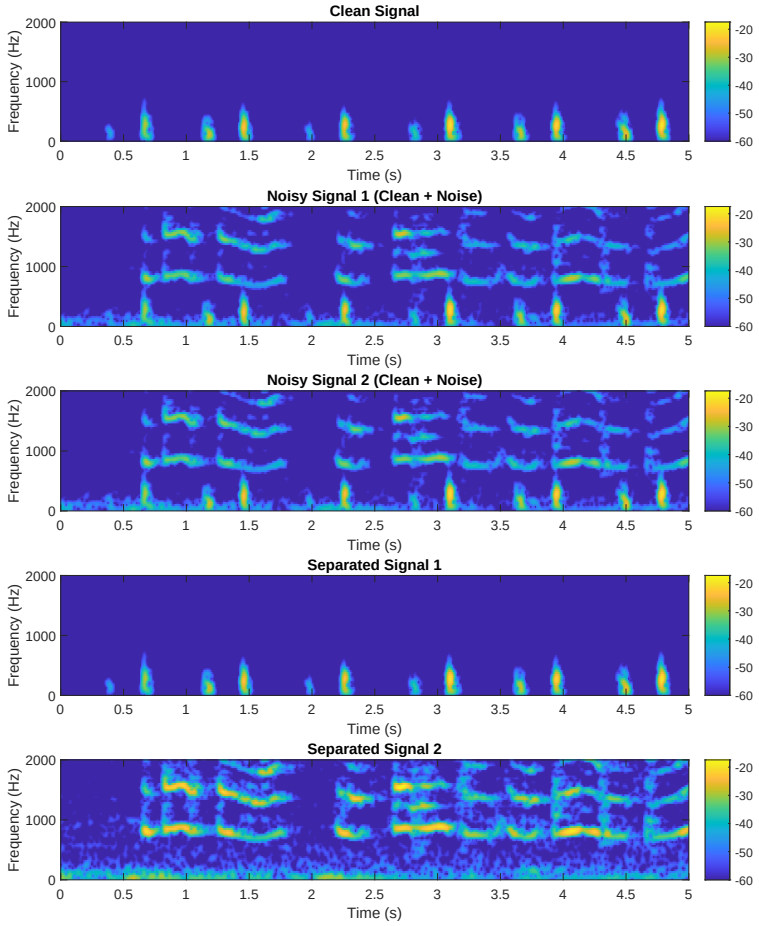


Figure A.67: Mel Spectrograms for heartbeat separated from speech by FastICA.

FastICA successfully extracts the heartbeat from speech interference, with Separated Signal 1 containing a clean heartbeat and Separated Signal 2 capturing the speech noise.

Table A.28 summarises the SNR improvements.

Table A.28: SNR improvement — Heartbeat corrupted with Speech.

Method	Output SNR (dB)	SNR Improvement (dB)
No Filtering	-4.10	-
FastICA (Separated Signal 1)	41.00	45.10
FastICA (Separated Signal 2)	-2.50	1.60