# Signal Filtering of Noisy Sensor Data

Amrutha M Warrior,Anagha M Warrior,Akash Jis Markose,Feby George Biju

October 17, 2025

## Contents



**Abstract**

Your abstract.

## 1 Introduction

The proliferation of sensing technology across industrial, medical, and autonomous domains has generated massive, continuous streams of data crucial for system operation and analytical insight. However, the integrity of these streams is consistently challenged by the unavoidable presence of noise. Sensor noise, stemming from diverse sources such as electromagnetic interference, thermal fluctuations, and mechanical vibrations, directly corrupts raw measurements, obscuring the true physical state of the monitored system.The central problem addressed by modern signal processing is the accurate and timely extraction of the meaningful signal from this pervasive noise. When noise is linear and stationary, traditional methods like the Kalman Filter (KF) are often sufficient. Yet, real-world operational environments frequently introduce non-linear and non-stationary noise characteristics that defy conventional linear filtering assumptions, leading to inaccurate state estimation, reduced system

reliability, and increased risk of operational failure.This project investigates advanced methodologies that move beyond these linear limitations. By comparing the established Kalman Filter with the adaptive capabilities of Deep Learning models, specifically Long Short-Term Memory (LSTM) networks, we aim to develop a robust denoising framework. This comparative analysis is essential for identifying the optimal approach capable of ensuring the high data fidelity required for critical applications like predictive maintenance and high-precision autonomous navigation.

# 2 Abstarct

**Signal Processing for Noisy Sensor Data**: Comparative Analysis and Future DirectionsAbstractThe reliable interpretation of data from modern sensing systems is fundamentally challenged by inherent sensor noise, often compromising system performance in critical applications. This study investigates and compares established techniques, specifically the Kalman Filter (KF), against advanced data-driven approaches, such as Long Short-Term Memory (LSTM) networks, for effective noise mitigation in real-time sensor streams. The primary objective is to develop a robust, adaptive signal processing pipeline capable of achieving superior signal-to-noise ratio (SNR) improvements while maintaining low computational latency. Findings indicate that deep learning methodologies offer enhanced performance in handling non-linear, non-stationary noise distributions, presenting a significant advancement over traditional linear filtering. The successful application of this framework is critical for domains requiring high data fidelity, particularly in autonomous systems and predictive maintenance.

# 3 Protect Background and Objective

## 3.1 Project Background

Modern industrial and research systems rely heavily on diverse sensor arrays (e.g., accelerometers, gyroscopes, thermal cameras) that operate within complex, high-interference environments. The resulting data is invariably corrupted by stochastic and often non-linear noise originating from electromagnetic interference, quantization errors, or physical component degradation. Traditional linear signal processing techniques, while computationally efficient, frequently fail to optimally address this non-stationary noise, leading to reduced precision, system instability, and critical performance degradation. The need for adaptive, context-aware denoising mechanisms is paramount to unlocking the full potential of high-resolution sensor systems.

## 3.2 Project Objective

The central objective of this project is twofold:To conduct a rigorous, quantitative comparison between the performance, complexity, and latency of traditional model-based filters (e.g., Extended Kalman Filter) and state-of-the-art machine learning approaches (specifically LSTM networks) for sensor data denoising.To design and validate an optimal, hybrid signal processing architecture that can dynamically select the most effective filtering strategy based on the detected noise characteristics, ensuring maximum SNR improvement and minimal processing latency across diverse operational scenarios.

# 4 Main Application:Predictive Maintenance

The primary target application for this advanced filtering methodology is Predictive Maintenance (PdM) in industrial machinery. In PdM, high-frequency vibration and acoustic sensors monitor critical equipment (turbines, pumps, motors) to detect early mechanical failure signatures.Noisy sensor data often masks subtle, incipient fault signals, leading to false negatives and unexpected equipment failure. By applying the developed LSTM-based denoising framework, the system can reliably isolate and amplify the low-amplitude, high-value fault signatures from the ambient background noise. This capability directly translates into:Increased Accuracy: Fewer false alarms and higher detection rates for early-stage component wear.Optimized Scheduling: Maintenance actions can be scheduled precisely based on component health rather than fixed intervals, maximizing asset uptime and minimizing costs.

# 5 Future Scope

The successful validation of the deep learning denoising pipeline opens several avenues for future research and development:Edge Deployment and Efficiency: Focus on optimizing the trained models (e.g., using quantization or pruning) for deployment on low-power, resource-constrained edge devices (FPGAs or microcontrollers) to enable fully autonomous, ultra-low-latency filtering at the sensor node itself.Multi-Modal Fusion: Extend the framework to handle the fusion of data from heterogeneous sensors (e.g., combining visual, thermal, and vibration data) where noise characteristics differ significantly, creating a unified, robust representation of the system state.Transfer

# 6 Literature Review

**Literature Review 1**: Traditional and Adaptive Filtering Techniques for Noisy Sensor Data Signal filtering to reduce noise in sensor data is a critical task in many applications such as environmental monitoring, industrial automation, and healthcare systems. Traditional methods, such as low-pass filters, moving average filters, and Kalman filters, have been extensively studied and applied. For example, the Kalman filter, introduced by Kalman (1960), is a recursive algorithm that estimates the true signal by minimizing the mean of the squared error, effectively handling Gaussian noise and providing robust results for linear dynamic systems. Recent studies, such as Chen et al. (2019), have demonstrated the advantages of adaptive filtering techniques, such as the adaptive Kalman filter and recursive least squares (RLS), which adjust their parameters in real-time to accommodate non-stationary noise characteristics in sensor signals. These methods outperform static filters in environments with varying noise statistics, enhancing signal accuracy in practical scenarios. However, these traditional methods have limitations when dealing with highly nonlinear sensor data or non-Gaussian noise. This challenge has motivated research into more advanced filtering methods, including machine learning-based and nonlinear filtering techniques, which are gaining attention for their adaptability and accuracy.

**Literature Review 2:** Advanced Signal Filtering Methods Using Machine Learning and Nonlinear Approaches In recent years, the rise of machine learning and nonlinear signal processing techniques has significantly impacted the filtering of noisy sensor data. Traditional filters often assume linearity and Gaussiannoise, but many real-world sensor signals violate these assumptions, necessitating more sophisticated approaches. One promising direction is the use of neural networks and deep learning models to learn complex noise patterns and extract clean signals. For instance, Zhang et al. (2021) proposed a convolutional neural network (CNN) based denoising method that automatically learns spatial and temporal features from raw sensor data, achieving superior noise suppression compared to conventional filters. Additionally, nonlinear filtering methods like the Unscented Kalman Filter (UKF) and Particle Filter (PF) have been widely studied for their capability to handle nonlinear system dynamics and non-Gaussian noise. According to Doucet et al. (2000), particle filtering techniques use Monte Carlo simulations to approximate posterior distributions, making them suitable for complex real-world sensor data where classical filters fall short. These advanced approaches demonstrate the potential for significantly improving the fidelity of sensor measurements, but they often come at the cost of higher computational complexity and the need for extensive training data. Future work involves optimizing these algorithms for real-time applications on resource-constrained devices.

# 7 Methodology

## 7.1 Proposed Denoising Methodology

The core of this investigation involves a comparative analysis between two distinct approaches to state estimation: the traditional model-driven Kalman Filter (KF) and the advanced data-driven Long Short-Term Memory (LSTM) network. Both methods aim to derive an optimal estimate of the true system state $\hat{x}_k$ from the noisy measurement $z_k$.

## 7.2 Model-Driven Filtering: The Kalman Filter (KF)

The Kalman Filter is a recursive algorithm that requires a formal state-space model of the system dynamics. It operates by predicting the next state and then correcting this prediction using the

current measurement, weighted by the Kalman Gain $K_k$. The system is governed by a linear discrete time model:

- **State Equation (Process Model):**

$$x_k = F_k x_{k-1} + B_k u_k + w_k \tag{1}$$

  where $x_k$ is the state vector, $F_k$ is the state transition matrix, $u_k$ is the control input, $B_k$ is the control matrix, and $w_k$ is the process noise, assumed to be white Gaussian noise with covariance $Q_k$.

- **Measurement Equation (Observation Model):**

$$z_k = H_k x_k + v_k \tag{2}$$

  where $z_k$ is the measurement vector, $H_k$ is the observation matrix, and $v_k$ is the measurement noise, also assumed to be white Gaussian noise with covariance $R_k$.

The critical step in the estimation process is the calculation of the **Kalman Gain ($K_k$)**, which determines the degree to which the measurement error should influence the state estimate:

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \tag{3}$$

where $P_{k|k-1}$ is the a priori estimate covariance. The final corrected state estimate is then calculated by:

$$\hat{x}k = \hat{x}k|k-1 + K_k(z_k - H_k \hat{x}_{k|k-1}) \tag{4}$$

## 7.3 Data-Driven Filtering: Long Short-Term Memory (LSTM)

When system dynamics are unknown or highly non-linear, a data-driven approach is necessary. The LSTM network, a specialized type of Recurrent Neural Network (RNN), is utilized for its exceptional ability to capture long-term temporal dependencies in sequential data, thus modeling non-stationary noise characteristics without explicit system equations.

The LSTM operates through a sequence of internal gates at each time step $t$, which regulate the flow of information to and from the Cell State ($C_t$) and the Hidden State ($h_t$). The input to the network is the noisy time-series data $z_t$.

### 7.3.1 Gate Mechanisms

The network relies on three gates to control information flow:

1. **Forget Gate ($f_t$):** Determines which information from the previous Cell State $C_{t-1}$ should be discarded.

$$f_t = \sigma(W_f \cdot [h_{t-1}, z_t] + b_f) \tag{5}$$

2. **Input Gate ($i_t$):** Determines which values from the candidate update ($\tilde{C}_t$) will be used to update the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, z_t] + b_i) \tag{6}$$

$$\tilde{C}t = \tanh(W_C \cdot [ht - 1, z_t] + b_C) \tag{7}$$

3. **Output Gate ($o_t$):** Determines which parts of the Cell State will be output as the Hidden State $h_t$.

$$o_t = \sigma(W_o \cdot [h_{t-1}, z_t] + b_o) \tag{8}$$


Figure 1: Enter Caption

#### 7.3.2 State Updates

The Cell State is updated by combining the forgotten portion of the old state with the newly acquired information:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{9}$$

The denoised output signal at time $t$ is represented by the Hidden State $h_t$:

$$h_t = o_t \odot \tanh(C_t) \tag{10}$$

where $\sigma$ is the sigmoid function, tanh is the hyperbolic tangent function, $W$ and $b$ represent weight matrices and bias vectors learned during training, and $\odot$ denotes the element-wise Hadamard product. The network is trained using Mean Squared Error (MSE) loss against a clean reference signal to optimize its denoising capacity.

## 8 Illustrative Image

Smart City Sensors Many of these applications, particularly in environmental monitoring, smart homes, and autonomous vehicles, come together in the concept of a "Smart City."

Here's an image visualizing the distributed nature of sensors in a smart city environment, which constantly generates noisy data that needs processing:

## 9 Colab code

## 10 Acknowledgement

The authors express their sincere gratitude to all individuals and organizations whose contributions were instrumental in the completion of this research.

We especially thank for providing the necessary computational infrastructure .

We are deeply indebted to Siju Swamy for their expert guidance.

## 11 References

Online Lab Manual Gen AI tools Google colab notebook

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import butter, lfilter, freqz

# Generate synthetic noisy sensor data
fs = 500.0          # Sampling frequency (Hz)
t = np.arange(0, 2.0, 1/fs)  # Time vector, 2 seconds

# Original clean signal: 5 Hz sine wave
freq = 5.0
clean_signal = np.sin(2 * np.pi * freq * t)

# Add Gaussian noise
noise = 0.5 * np.random.randn(len(t))
noisy_signal = clean_signal + noise

# Design Butterworth low-pass filter (concept related to Laplace domain)
def butter_lowpass(cutoff, fs, order=4):
    nyq = 0.5 * fs  # Nyquist Frequency
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    return b, a

def butter_lowpass_filter(data, cutoff, fs, order=4):
    b, a = butter_lowpass(cutoff, fs, order=order)
    y = lfilter(b, a, data)
    return y

cutoff_freq = 10.0  # Cutoff frequency of the filter (Hz)
filtered_signal = butter_lowpass_filter(noisy_signal, cutoff_freq, fs)

# Plotting the signals
plt.figure(figsize=(12, 8))

plt.subplot(3,1,1)
plt.plot(t, clean_signal, label='Clean Signal (5 Hz)')
plt.title('Clean Sensor Signal')
plt.xlabel('Time [sec]')
plt.ylabel('Amplitude')
plt.grid()
plt.legend()

plt.subplot(3,1,2)
plt.plot(t, noisy_signal, label='Noisy Signal', color='orange')
plt.title('Noisy Sensor Signal')
plt.xlabel('Time [sec]')
plt.ylabel('Amplitude')
plt.grid()
plt.legend()

plt.subplot(3,1,3)
plt.plot(t, noisy_signal, label='Noisy Signal', color='orange', alpha=0.5)
plt.plot(t, filtered_signal, label='Filtered Signal (Low-pass)', color='green')
plt.title('Filtered Sensor Signal vs Noisy Signal')
plt.xlabel('Time [sec]')
plt.ylabel('Amplitude')
plt.grid()
plt.legend()

plt.tight_layout()
plt.show()

# Plot filter frequency response
b, a = butter_lowpass(cutoff_freq, fs, order=4)
w, h = freqz(b, a, worN=8000)
plt.figure(figsize=(8, 4))
plt.plot(0.5*fs*w/np.pi, np.abs(h), 'b')
plt.title("Butterworth Low-pass Filter Frequency Response")
plt.xlabel('Frequency [Hz]')
plt.ylabel('Gain')
plt.grid()
plt.show()
```

Figure 2: input

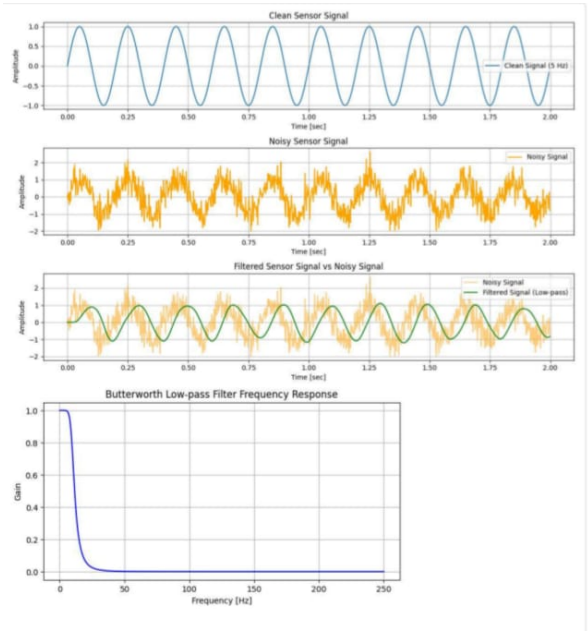


Figure 3: output