NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

Engineering Analysis and Design - ICICC10
PROJECT REPORT

OPTIMIZED MATHEMATICAL MODEL AND FORECASTING MODEL OF AN ELECTROMECHANICAL RESONATOR

Submitted To:
Dr Anuradha Tomar

Submitted By:
Kshitij Gupta - 2022UIC3556
Mayank Gautam - 2022UIC3543
Ekansh Kumar - 2022UIC3554

ABSTRACT

The escalating global concerns surrounding climate change and the depletion of fossil fuel reserves have intensified the exploration of sustainable energy alternatives. Among these alternatives, acoustic energy emerges as a promising yet underutilized resource with vast potential. This report delves into the intricate dynamics of harnessing acoustic energy through cutting-edge technologies such as piezoelectric transducers and acoustic panels. Despite its abundance in urban environments and industrial settings, tapping into acoustic energy faces formidable challenges. Existing solutions are hindered by inherent limitations, including constraints related to frequency dependence, operational adaptability, and scalability. Specifically, these solutions rely on materials with resonant frequencies that must closely match incoming sound waves for optimal energy conversion. This narrow frequency range restricts their versatility and efficiency, particularly in environments with diverse sound frequencies. Addressing these limitations is imperative to unlock the full potential of acoustic energy as a sustainable power source.

In response to these challenges, our methodology proposes a transformative approach centered around a sophisticated mathematical MATLAB model of an Electromechanical Helmholtz Resonator. This resonator represents a paradigm shift in acoustic energy harvesting technology by autonomously tuning its resonant frequency which is dependant on its geometry. This unique capability ensures that the resonator consistently operates at maximum efficiency, irrespective of changing ambient conditions. By dynamically adjusting its resonant frequency, the Electromechanical Helmholtz Resonator overcomes the limitations of traditional materials-based approaches, offering unparalleled adaptability and performance across a wide range of sound frequencies. Furthermore, our research endeavors encompass the development of a robust prediction model capable of accurately forecasting the power output generated by acoustic energy harvesting systems under fixed ambient conditions.

Our findings demonstrate a significant enhancement in energy harvesting efficiency with the implementation of our proposed methodology. We analyzed 100 minutes of audio files containing low-flying aircraft noises and 50 minutes of audio files containing noise recorded from a hospital corridor. We have compared the results of our tuned EMHR model with another model representing the existing solutions without tuning. The comparison reveals a notable increase of 'x' percent in power generation when utilizing our tuned model, underscoring the efficacy and practical applicability of our approach in real-world acoustic energy harvesting scenarios.

INTRODUCTION

NEED FOR SUSTAINABLE SOURCES OF ENERGY

The shift to sustainable energy is crucial for tackling 21st-century challenges like climate change and fossil fuel depletion. Sources like solar, wind, and hydro offer clean, unlimited power, cutting greenhouse gasses and preserving biodiversity. They create jobs, reduce reliance on imports, and stabilize energy costs, fostering economic resilience. Investment in renewables ensures energy security, promotes equity, and aligns with sustainable development goals. This transition isn't just an environmental necessity but a strategic opportunity, making the planet safer, healthier, and more prosperous for future generations.

ACOUSTIC ENERGY BEING AN UNTAPPED/NON-VIABLE SOURCE

Acoustic energy, abundant in our environment, remains an untapped renewable resource. Emerging technologies like piezoelectric transducers and acoustic panels enable the conversion of ambient sound into electricity. This innovation unlocks opportunities for sustainable energy harvesting from urban noise, industrial sites, and daily interactions. By capturing vibrations from traffic, machinery, and public spaces, acoustic energy complements traditional renewables, diversifying our energy mix. This approach addresses noise pollution while supporting global efforts to minimize reliance on fossil fuels and reduce environmental impacts.

LACK OF PRACTICAL VIABILITY OF EXISTING SOLUTIONS

The existing solution requires choosing a material whose resonant frequency is close to the frequency of the incoming sound waves. These materials will only generate maximum output of vibrations if the incoming sound waves are close to its natural frequency, therefore this model will only give efficient results for a specific range of frequencies and may not resonate optimally while capturing energy from a broader range of sound frequencies, which generally is the case for most places. Their inflexibility restricts versatility and operational flexibility, hindering their applicability in diverse environments and settings. Additionally, this will result in energy losses and reduced reliability over time. This fixed nature also poses challenges in scaling up the energy harvesting system for larger installations or broader applications. Conclusion is that the results of such a model produces very minimal output which makes it a non viable idea.

PROPOSED METHODOLOGY AND SOLUTION

In response to the above mentioned limitations of traditional solutions, our research proposes a novel approach centered on a mathematical MATLAB model of an Electromechanical Helmholtz Resonator. This resonator, equipped with the ability to autonomously adjust its resonant frequency by changing its geometrical characteristics, offers a promising avenue for maximizing energy conversion efficiency. Unlike traditional methods reliant on fixed resonant frequencies,

the adaptability of the Electromechanical Helmholtz Resonator ensures consistent performance across a range of ambient conditions. Additionally, our study encompasses the development of a predictive model tailored to forecast power generation within specific ambient conditions, enhancing our understanding of optimal energy harvesting strategies. Through the integration of these advanced methodologies, we aim to contribute to the advancement of acoustic energy harvesting technology and its broader adoption as a sustainable power source.

THEORETICAL BASICS

HELMHOLTZ RESONATOR

Helmholtz resonators are acoustic devices named after Hermann von Helmholtz, which are designed to efficiently absorb or generate sound waves at specific frequencies. They consist of a cavity (or chamber), a neck (or opening), and an inertial mass (the volume of air in the neck). When subjected to an external acoustic wave at the resonant frequency, Helmholtz resonators exhibit a significant increase in amplitude due to constructive interference.

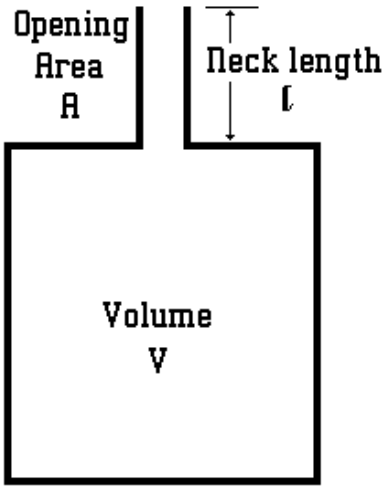


Fig 1 : Basic Schematic of a Helmholtz Resonator

Equation for Helmholtz Resonators:

The resonant frequency (f) of a Helmholtz resonator can be calculated using the Helmholtz resonance equation:

$$f = \frac{c}{2\pi}\sqrt{\frac{A}{Vl}} \qquad (1)$$

where :

- f is the resonant frequency of the device
- c is the speed of sound in air
- A is the area of the orifice or neck
- V is the volume of the cavity
- l is the effective length of the orifice

Working Principle:

When an acoustic wave with a frequency close to the resonant frequency of the Helmholtz resonator is incident upon the device, the pressure variation in the cavity causes the air inside to oscillate. At the resonant frequency, the inertial mass of air in the neck and the spring-like

compliance of the air in the cavity combine to produce maximum displacement at the mouth of the resonator. This amplifies the acoustic wave, resulting in resonance.

Assumption of No Damping:
In the absence of damping, Helmholtz resonators continue to resonate indefinitely at their natural frequency once excited. However, in practical applications, damping is present due to factors such as air viscosity and material losses, which gradually dissipate the energy of the oscillations, causing the resonance to decay over time.

Conclusion:
Helmholtz resonators are fundamental acoustic devices used in various applications, including noise reduction, sound amplification, and acoustic energy harvesting. Understanding their working principle and resonance equation is essential for designing and optimizing resonant systems for specific frequencies and applications.

PIEZOELECTRIC TRANSDUCER AND CONVERSION

Basics of Piezoelectricity:
Piezoelectricity is a property exhibited by certain materials, such as crystals, ceramics, and polymers, where they generate an electric charge in response to mechanical stress or deformation. Conversely, these materials also deform when subjected to an electric field. This phenomenon arises due to the asymmetry in the material's crystal structure, leading to the separation of positive and negative charges when the material is mechanically stressed.

Principle of Energy Conversion:
When a piezoelectric material is mechanically deformed, such as by bending or stretching, it generates an electric charge across its surface. Conversely, when an electric field is applied to the material, it undergoes mechanical deformation. This bidirectional conversion between mechanical and electrical energy forms the basis of piezoelectric transducers.

Basic Transducer Equations:

- Voltage Generated:
  The voltage (V) across the piezoelectric material is proportional to the applied mechanical stress (σ) or strain (ϵ):
  $V = g . \sigma$ (2)
  where:
  - V is the voltage generated
  - g is the piezoelectric voltage coefficient

Piezoelectric Energy Generation with Resonating Diaphragm:

When a piezoelectric material is attached to a resonating diaphragm, the mechanical vibrations of the diaphragm cause the piezoelectric material to deform. This deformation generates an electric charge across the piezoelectric material, converting the mechanical energy of the diaphragm's vibrations into electrical energy.

Applications:
Piezoelectric energy generation with resonating diaphragms finds applications in various fields, including:

- Energy harvesting from ambient vibrations for powering small electronic devices.
- Sensors for measuring vibrations, pressure, or acoustic waves.
- Actuators for generating mechanical vibrations in response to electrical signals.

Conclusion:
The integration of piezoelectric materials with resonating diaphragms enables the conversion of mechanical vibrations into electrical energy, exploiting the piezoelectric effect. Understanding the principles and equations governing piezoelectric transducers provides insight into their functionality and applications in energy harvesting, sensing, and actuation.

ELECTROMECHANICAL HELMHOLTZ RESONATOR

In an electromechanical Helmholtz resonator, a piezoelectric backplate is added to the resonator, creating an electromechanical coupling. This addition allows for the conversion between electrical and mechanical energy, making the resonator responsive to both acoustic and electrical signals.

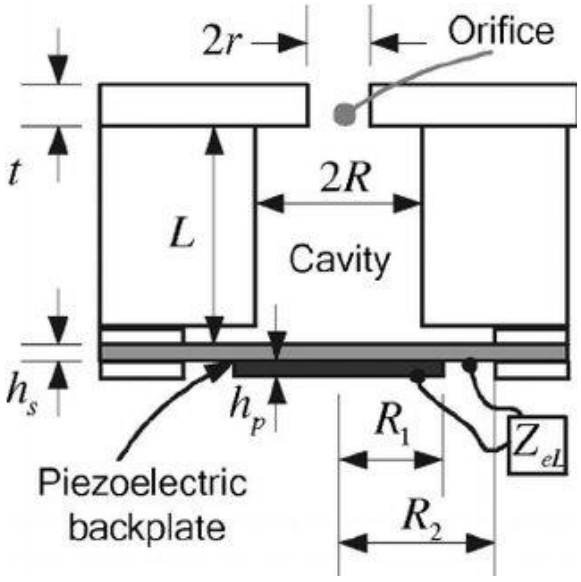


Fig 2 : Cross Section of Acoustic Electromechanical Helmholtz Resonator

Electromechanical Coupling:
The addition of a piezoelectric backplate introduces electromechanical coupling to the Helmholtz resonator system. When an acoustic wave interacts with the resonator, it causes the diaphragm to vibrate. These vibrations induce mechanical strain in the piezoelectric backplate, generating an electrical charge across its surface due to the piezoelectric effect. Conversely, an applied electrical signal across the piezoelectric material induces mechanical deformation, causing the diaphragm to vibrate.

Electrical Equivalent Circuit:
The electromechanical Helmholtz resonator can be represented by an electrical equivalent circuit, which includes components to represent the mechanical and electrical behavior of the system.

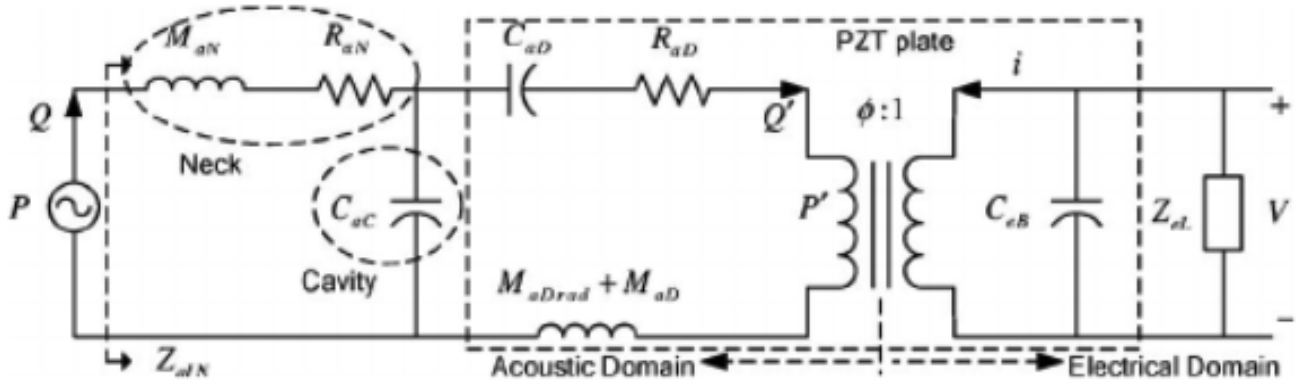


Fig 3 : Equivalent electroacoustic circuit representation of the EMHR

The electrical equivalent circuit typically consists of the following elements:

- Mechanical Compliance (C_m): Represents the compliance of the mechanical system, including the compliance of the air in the resonator chamber and the mechanical flexibility of the diaphragm.
- Mechanical Inertia (L_m): Represents the inertia of the mechanical system, including the mass of the air in the resonator chamber and the mass of the diaphragm.
- Electrical Capacitance (C_e): Represents the capacitance of the piezoelectric material, which converts mechanical strain into electrical charge and vice versa.
- Electrical Inductance (L_e): Represents the inductance associated with the electrical properties of the piezoelectric material and any external electrical components.

Resonant Frequency:
The resonant frequency of the electromechanical Helmholtz resonator is influenced by both the mechanical and electrical properties of the system. The resonant frequency can be determined by analyzing the impedance of the system, considering both mechanical and electrical components.

Amplitude Response:
Similar to the purely mechanical Helmholtz resonator, the electromechanical Helmholtz resonator will exhibit resonance behavior, with a peak in the amplitude response at the resonant frequency. However, in addition to acoustic excitation, the resonator can also respond to electrical signals, leading to complex amplitude responses depending on the input signal frequency and amplitude.

Applications:
Electromechanical Helmholtz resonators find applications in various fields, including:

- Acoustic transducers for converting electrical signals into acoustic waves and vice versa.
- Sensors for measuring acoustic pressure or vibration.
- Actuators for generating acoustic waves or mechanical vibrations in response to electrical signals.

Conclusion:
The addition of a piezoelectric backplate to a Helmholtz resonator creates an electromechanical system capable of converting between electrical and mechanical energy. This enhanced functionality enables a wide range of applications in sensing, actuation, and acoustic signal processing. The theory presented above provides a foundation for understanding the behavior and applications of electromechanical Helmholtz resonators.

MATHEMATICALLY MODELING SYSTEMS IN MATLAB

Mathematical models are representations of real-world systems using mathematical equations. These equations describe the relationships between various system components and variables, allowing us to predict and analyze the behavior of the system under different conditions. Mathematical models can be used to simulate physical, electrical, mechanical, biological, and many other types of systems.
MATLAB provides a powerful platform for simulating mathematical models of physical and electrical systems. Users can implement mathematical equations in MATLAB code and use built-in functions and toolboxes to simulate the system's behavior over time. MATLAB offers a variety of numerical solvers for ordinary and partial differential equations, optimization algorithms, and control design tools, making it suitable for simulating a wide range of systems.

PROPOSED SOLUTION

METHODOLOGY

Our proposed methodology introduces a novel approach to addressing the challenges in acoustic energy harvesting. Central to our methodology is the development of a sophisticated mathematical MATLAB model of an Electromechanical Helmholtz Resonator. This resonator signifies a significant departure from conventional methods, as it possesses the unique ability to autonomously tune its resonant frequency based on its geometric configuration. This self-tuning mechanism ensures that the resonator consistently operates at peak efficiency, regardless of fluctuations in ambient conditions.

Equation (1) in Theoretical Basics can be simplified to :

$$f = \frac{c}{2\pi}\sqrt{\frac{r_a^2}{r_v^2 l_v l_a}}$$

where :

- f = Resonant Frequency of the resonator
- c = Speed of light in air
- r_a = Radius of the area of neck/orifice
- r_v = Radius of the volume of cavity
- l_a = length of the neck/orifice
- l_v = length of the cavity

We implement a dynamic adjustment of the ratio $r_a : r_v$.

This adjustment is performed following the calculation of the required resonant frequency, which is set equal to the fundamental frequency of the input signal.

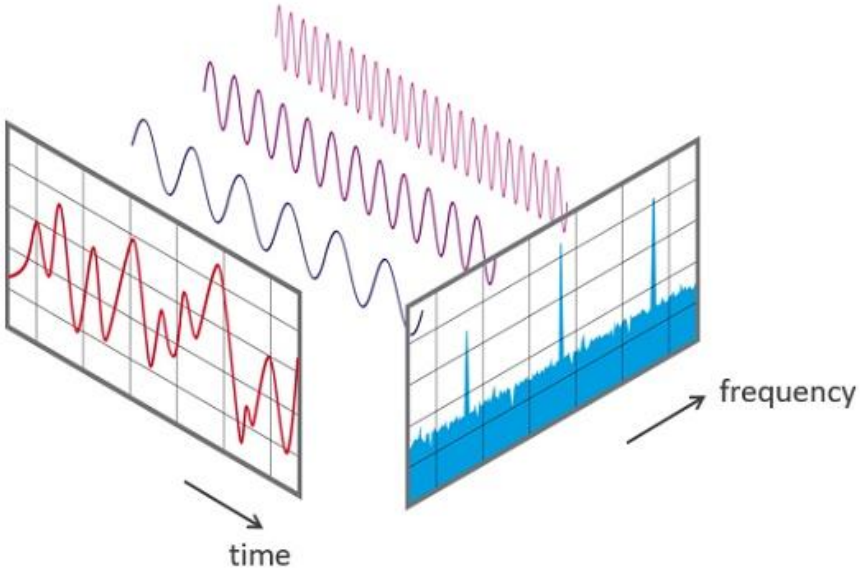


Fig 1: FFT to transfer from time domain to frequency domain

To determine the fundamental frequency accurately, we utilize a Fast Fourier Transform (FFT) on the time series data obtained from the input signal. This analysis allows us to identify the frequency component with the highest amplitude, representing the fundamental frequency of the sound wave. Upon obtaining the fundamental frequency, we adjust the resonator's geometry by modifying the ratio of the neck to cavity radius accordingly. Let the fundamental frequency f_{fund} of the input signal and f_{res} be the resonant frequency of the resonator.

Then : $\frac{r_a}{r_v} = \frac{f_{fund}}{K}\sqrt{l_v l_a}$ where $K = \frac{c}{2\pi}$

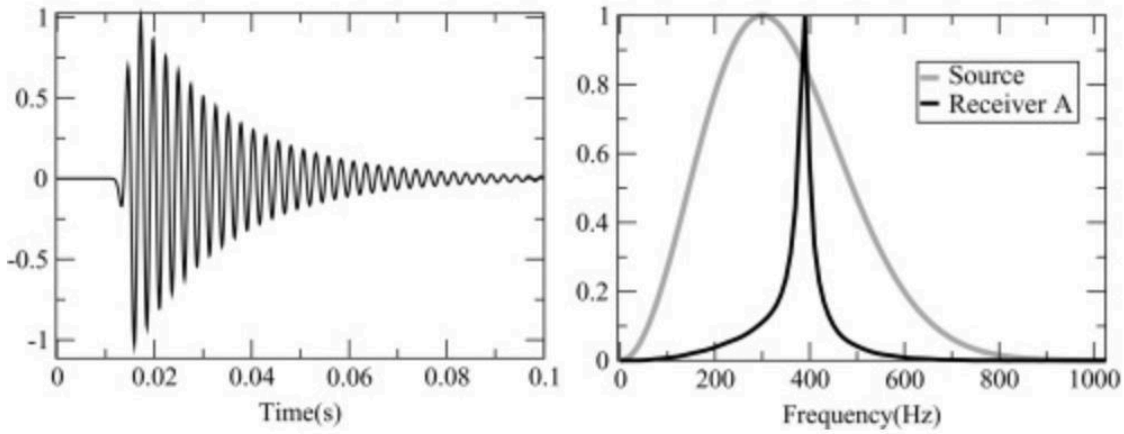


Fig 2 : Resonance Performance Characteristics of an oscillator

This dynamic adaptation ensures that the resonator's resonant frequency aligns precisely with the incoming sound wave, maximizing energy harvesting efficiency. By continuously monitoring and updating the resonator's configuration based on real-time input signal characteristics, our methodology overcomes the limitations of fixed resonant materials and offers unparalleled adaptability and performance across a wide range of sound frequencies.

The backplate of the EMHR is a circular disc made of a piezoceramic material and it is connected to the diaphragm of the EMHR. The vibrations from the diaphragm, caused due to resonance in the cavity of the EMHR, cause strain in the piezoelectric backplate which then produces an output voltage. The magnitude of the output voltage depends on the strain and the material parameters. The piezoceramic material chosen in this process is PZT-5, which has the following values of the listed material properties:

- $g = -1.1025 * 10^{13}\ C/N\ (coulombs/newton)$
- $d = 0.31 * 10^{-6}\ m$

where:

- g is the piezoelectric strain constant (also known as the piezoelectric charge constant or piezoelectric coefficient), which is a material property
- d is the thickness of the piezoelectric material (in meters)

The output voltage generated by a piezoelectric material is given by the equation:

$$V = g * d * P$$

where:

- V is the output voltage
- g is the piezoelectric voltage constant (also known as the piezoelectric charge constant or piezoelectric coefficient), which is a material property
- d is the thickness of the piezoelectric material (in meters)
- P is the applied pressure (in Pascals)

The piezoelectric voltage constant g is related to the material properties as follows:

$$g = d_{31} * Y$$

where:

- d_{31} is the piezoelectric strain coefficient
- Y is the Young's Modulus of the material

Substituting the above listed parameters in the output voltage formula, we get:

$$V = -3.41825 * 10^{-3} * P$$

Average power is calculated from the voltage generated after the piezoelectric conversion.
A standard value(200Ω) of load is assumed for the calculation of power. Instantaneous Power is calculated using the formula:

$$P = \frac{V^2}{R}$$

at every time interval corresponding to each data point in the voltage array generated after the piezoelectric conversion and then those values are added in order to get the Total Average Power over the complete duration.

$$P_{avg} = \frac{V_1^2+V_2^2+V_3^2+\ldots+V_n^2}{Z}$$

where:

- P_{avg} is the total average power generated over the complete duration
- $V_1, V_2, V_3, \ldots, V_n$ are the instantaneous voltages
- Z is the standard load

IMPLEMENTATION

The implementation of the process includes various steps listed below:

Sound Input:

- Capture or import the sound signal into MATLAB.
- Define parameters such as sampling frequency, duration, and characteristics of the sound signal (e.g., frequency spectrum, amplitude).

Conversion to Pressure Levels:

- Convert the input sound signal from the time domain to the frequency domain using FFT.
- Apply appropriate calibration factors to convert the FFT magnitude spectrum to sound pressure levels (SPL) in decibels (dB).
- SPL is typically calculated using the formula: $SPL(dB) = 20 * log_{10}(\frac{p_{ref}}{p})$
  where:
  - p_{ref} is the reference sound pressure level (usually 20 μPa)
  - p is the sound pressure level of the input signal

- MATLAB provides functions like $audioread()$ for importing audio files and $fft()$ for FFT computation.

FFT Calculation for Fundamental Frequency:

- Apply Fast Fourier Transform (FFT) to the sampled noise signal to convert it from the time domain to the frequency domain.
- Find the dominant frequency component, which represents the fundamental frequency of the noise signal.
- MATLAB provides the $fft()$ function to perform FFT.

Tuning EMHR's Resonant Frequency:

- Define the EMHR's resonant frequency and characteristics.
- Use mathematical models or experimental data to establish the relationship between the radius of the orifice and the resonant frequency.
- Implement algorithms to adjust the radius of the orifice based on the calculated fundamental frequency of the input noise signal.
- This step requires understanding the physics of the EMHR system and its resonance properties.

Output Voltage Calculation:

- Simulate the EMHR system's response to the tuned resonant frequency.
- Use circuit modeling techniques or system equations to calculate the output voltage.
- Consider factors such as impedance matching, damping effects, and noise filtering.
- MATLAB's Simulink can be used for simulating the EMHR system response.

Average Power Calculation:

- Integrate the squared output voltage over time to calculate instantaneous power.
- Average the instantaneous power over the entire duration of the signal to obtain the average power.
- MATLAB's numerical integration functions ($trapz()$, $quad()$, etc.) can be used for this purpose.

Visualization and Analysis:

- Visualize the input noise signal, its FFT spectrum, EMHR response, and output voltage.
- Analyze the calculated average power and compare it with theoretical predictions or experimental measurements.
- Use MATLAB's plotting functions (plot, spectrogram, etc.) for visualization and analysis.

Optimization and Performance Evaluation:

- Optimize the code for efficiency and accuracy.

- Evaluate the performance of the implemented system under different noise conditions and tuning strategies.
- Consider parameter sensitivity analysis and robustness testing.

By following these steps and implementing them in MATLAB, we can create a comprehensive simulation framework for sampling input noise, tuning EMHR's resonant frequency, and calculating output voltage and average power.

POWER WITH TUNING

Sound pressure levels

```
source = dsp.AudioFileReader('/Users/rxnkshitij748/Documents/MATLAB/PROJECT/DRILLAUDIO.
fs = source.SampleRate;

player = audioDeviceWriter('SampleRate',fs);

SPL = splMeter( ...
    'Bandwidth','1/3 octave', ...
    'SampleRate',fs);
centerFrequencies = getCenterFrequencies(SPL);
LeqPrevious = zeros(size(centerFrequencies));
while ~isDone(source)
    x = source();
    player(x);
    [~,Leq] = SPL(x);

end

dbMax = max(max(max(Leq, [], 2)));

release(source)
release(player)
release(SPL)
```

Fig 1: MATLAB Code for calculating Sound Pressure Level

The code begins by initialising an **AudioFileReader** object to read audio data from an MP3 file and extract the sample rate of the audio. It then creates an **audioDeviceWriter** object for real-time audio output and a **splMeter** object for sound pressure level measurement with a specified bandwidth.

The code enters a loop, where it continuously reads audio frames from the file, plays them in real-time, and calculates the equivalent continuous sound level (Leq) across different octave bands using the SPL meter object. The maximum Leq value across all bands is determined, and the loop continues until all audio data is processed.

Finally, the resources associated with the audio input, output, and SPL meter objects are released. This code provides a comprehensive framework for real-time audio processing and analysis, enabling tasks such as sound level monitoring and analysis in MATLAB.

Time domain to frequency domain

```
[audioIn,fs] = audioread("DRILLAUDIO.mp3");
% audioIn(:,2) = [];%incase 2 signals
L_Signal = length(audioIn);
t = (0:L_Signal-1)/fs;
Y_fft = fft(audioIn);
subplot(2,1,1);
plot(fs/L_Signal*(0:L_Signal-1), abs(Y_fft),"LineWidth", 1.5);
title("Frequencyplot");
% Y_2fft = fftshift(Y_fft);
% plot(fs/L_Signal*(-L_Signal/2:L_Signal/2-1), abs(Y_2fft), "LineWidth",1.5);
% title("ShiftedFFt");
P2 = abs(Y_fft);%Double sided spectrum
amp_Spectrum = P2(1:L_Signal/2+1);
amp_Spectrum(2:end-1) = 2*(amp_Spectrum(2:end-1));
f_spectrum = fs/L_Signal*(0:(L_Signal/2));%half freq spectrum
subplot(2,1,2);
plot(f_spectrum,amp_Spectrum,"LineWidth",1.5);
title("Frequency Amplitude Plot");
[max, maxIndex] = max(amp_Spectrum);
FundamentalFrequency = f_spectrum(maxIndex);
```

Fig 2: MATLAB Code for computing frequency spectrum from time spectrum(FFT)

This code conducts a Fast Fourier Transform (FFT) analysis on an MP3 audio file to pinpoint its fundamental frequency. It starts by loading the audio file "DRILLAUDIO.mp3" using audioread, extracting the audio signal and its sampling rate. The script prepares for a mono signal processing by optionally removing a second channel, a step included but commented out for versatility. It calculates the signal's length to construct a time vector, essential for time-domain analysis, though the vector's direct use in plotting is not demonstrated in this snippet.

The core operation is the FFT, transforming the time-domain audio signal into the frequency domain, revealing the spectrum of frequencies present. The code meticulously processes the FFT output, focusing on the first half due to the symmetric nature of the FFT for real signals. It adjusts the amplitude spectrum to reflect actual signal power and generates a corresponding frequency vector.

A pivotal moment in the analysis is identifying the peak in the amplitude spectrum, which signifies the audio signal's fundamental frequency. By locating the maximum amplitude and its index, the script calculates the fundamental frequency, crucial for understanding the audio signal's primary harmonic content. This MATLAB routine elegantly blends audio signal processing techniques to extract a key characteristic—the fundamental frequency—demonstrating a practical application of FFT in audio analysis within a concise and informative framework.

Mathematical Formula

```
A_standard = 18.398; r_a = 2.42; la = 3.16;
V_standard = 2073.49; r_v = 6.34; lv = 16.42;
K = 54590.145; % K = c/2pi (mm/sec)
%GeometricConstant = (FundamentalFrequency/K) * sqrt(l);
resonant_freq = K * sqrt(A_standard/(V_standard*la));
ini_gConstant = r_a/r_v;
new_gConstant = (FundamentalFrequency/K) * sqrt(lv*la);
changingFactor = new_gConstant/ini_gConstant; %mechanical energy change constant
```

Fig 3: MATLAB Code for calculating geometrical parameters of the resonator

In this code, the resonant frequency of the resonator considering the standard values of dimensions is calculated using the formula as explained in the methodology. The initial and new geometric constants($\frac{r_a}{r_v}$) are calculated by using resonant frequency and fundamental frequency respectively. Both the constants are then compared and the change factor is stored for analysis.

Piezo Electric conversion

```
%(ASSUME DAMPING FACTOR 1 : CRITICAL DAMPING
%Properties : fres = 18khz : D = 12mm : Y = 6.3 * 10^10 : density = 7700 kg/m^3
%: T = 0.31 um : d31, piezoelectric strain constant = -175
D = 0.012; T = 0.31 * 10^-6 ; d31 = -175 ; Y = 6.3 * 10^10;
pascalMax = 10^(dbMax/20);
maxParray = repmat(pascalMax, length(t), 1); %since operating at resonating freq and critical damping
VoltageGen = (-3.418 * 10^-3).*maxParray;
Z_load= 1000 ; %resistance in ohms
Total_Time= t(end)-t(1); tdel = 2.2676e-05;
powerValues = (VoltageGen.^2) / Z_load;
totalEnergy = sum(powerValues * tdel); % Total energy in Joules
totalTime = length(VoltageGen) * tdel; % Total time
averagePowerOutput = totalEnergy / totalTime; % Average power in Watts
averagePowerOutput
```

Fig 4: MATLAB Code for calculating the Output Voltage and Average Power

This code computes the average power output of an electrical system designed to harness acoustic energy efficiently. It initialises parameters such as the damping factor, material thickness, piezoelectric strain constant, and Young's modulus. Using the maximum sound pressure level previously obtained, it calculates the maximum pressure in pascals. Subsequently, it generates an array with these maximum pressure values, reflecting operation at the resonating frequency and critical damping. The voltage generated by the piezoelectric material is then computed based on this pressure array and a predetermined coefficient. The load resistance is set to 1000 ohms, and the total time duration of voltage generation, along with the time step, is calculated. Power values generated at each time step are computed considering the generated voltage and load resistance. The total energy produced in joules is obtained by summing these power values. Similarly, the total time duration is calculated based on the length of the voltage array and the time step. Finally, the average power output is derived by dividing the total energy

by the total time duration, providing a critical assessment of the system's efficiency in converting acoustic energy into electrical power.

POWER WITHOUT TUNING

Sound Pressure levels

```
source = dsp.AudioFileReader("DRILLAUDIO.mp3");
fs = source.SampleRate;

player = audioDeviceWriter(SampleRate=fs);
SPL = splMeter(TimeWeighting="Fast", ...
    FrequencyWeighting="A-weighting", ...
    SampleRate=fs, ...
    TimeInterval=2);

while ~isDone(source)
    x = source();
    player(x);
    [Lt,Leq,Lpeak,Lmax] = SPL(x);
end

release(source)
release(player)
release(SPL)

dbMax = max(max(Leq, [], 2));
```

Fig 5: MATLAB Code for calculating Sound Pressure Level

This MATLAB code performs audio analysis by reading, playing, and measuring the sound pressure level (SPL) of "DRILLAUDIO.mp3". It uses an AudioFileReader to process the file and an audioDeviceWriter for playback, both set to the audio's original sample rate. The splMeter is configured with A-weighting and fast time weighting, mimicking human ear response and rapid sound level changes, respectively, and measures SPL every 2 seconds.
Within a loop, each audio segment is analysed for SPL while being played back. The splMeter calculates the equivalent continuous sound level (Leq), among other metrics, providing insights into the audio's loudness over time. After processing, resources are released, and the maximum Leq value is extracted, indicating the highest average loudness level within any 2-second period of the audio file. This approach is valuable for audio quality assessment, environmental noise analysis, and other applications requiring detailed sound level information. The code showcases

efficient audio processing and SPL measurement in MATLAB, leveraging digital signal processing tools for comprehensive audio analysis within a concise and practical framework.

Time domain to frequency domain

```
29      [audioIn,fs] = audioread("DRILLAUDIO.mp3");
30      % audioIn(:,2) = [];%incase 2 signals
31      Lt(:,2) = [];
32      L_Signal = length(audioIn);
33      t = (0:L_Signal-1)/fs;
34      Y_fft = fft(audioIn);
35      subplot(2,1,1);
36      plot(fs/L_Signal*(0:L_Signal-1), abs(Y_fft),"LineWidth", 1.5);
37      title("Frequencyplot");
38      % Y_2fft = fftshift(Y_fft);
39      % plot(fs/L_Signal*(-L_Signal/2:L_Signal/2-1), abs(Y_2fft), "LineWidth",1.5);
40      % title("ShiftedFFt");
41      P2 = abs(Y_fft);%Double sided spectrum
42      amp_Spectrum = P2(1:L_Signal/2+1);
43      amp_Spectrum(2:end-1) = 2*(amp_Spectrum(2:end-1));
44      f_spectrum = fs/L_Signal*(0:(L_Signal/2));%half freq spectrum
45      subplot(2,1,2);
46      plot(f_spectrum,amp_Spectrum,"LineWidth",1.5);
47      title("Frequency Amplitude Plot");
48      max, maxIndex] = max(amp_Spectrum);
49      FundamentalFrequency = f_spectrum(maxIndex);
```

Fig 6: MATLAB Code for computing frequency spectrum from time spectrum(FFT)

The MATLAB script begins by reading an audio file named "DRILLAUDIO.mp3" using the **audioread** function, which extracts the audio data into the variable **audioIn** and its sample rate into **fs**. The script calculates the length of the audio signal and generates a time vector **t** representing the time indices of each sample. Utilising the **fft** function, it computes the Fast Fourier Transform of the audio signal, transforming it from the time domain to the frequency domain, resulting in **Y_fft**. Subsequently, the script plots the frequency spectrum of the audio signal using **plot**, with frequency on the x-axis and the magnitude of **Y_fft** on the y-axis. It then computes the absolute values of **Y_fft** and stores it in **P2**. The next step involves extracting the positive half of the spectrum, represented by amp_Spectrum, and doubling its amplitude values, considering the symmetry of the FFT output. The script calculates the corresponding frequency values for each FFT bin and assigns them to **f_spectrum**. Another subplot is set up, where the amplitude spectrum is plotted against the frequency values. The maximum amplitude and its corresponding index in **amp_Spectrum** are determined using the max function, with the index representing the fundamental frequency's location. Finally, the fundamental frequency is extracted from **f_spectrum** using the index and stored in FundamentalFrequency. This comprehensive script provides insights into the frequency content of the audio signal, facilitating further analysis and interpretation

Mathematical Formula

```
A_standard = 18.398; r_a = 2.42; l_a = 3.16;
V_standard = 2073.49; r_v = 6.34; %l = 16.42;
K = 54590.145; % K = c/2pi (mm/sec)
resonantFrequency = K * sqrt(A_standard/(V_standard * l_a));
%GeometricConstant = (FundamentalFrequency/K) * sqrt(l);
% ini_gConstant = r_a/r_v;
% new_gConstant = (FundamentalFrequency/K) * sqrt(l);
% changingFactor = new_gConstant/ini_gConstant; %mechanical energy change constant

target = 2892;    %target value
closest = interp1(f_spectrum,f_spectrum,target,'nearest');
freqIndex = find(f_spectrum == closest);
ampAtResonant = amp_Spectrum(freqIndex);
volt_Factor = ampAtResonant/max;
```

Fig 7: MATLAB Code for calculating geometrical parameters of the resonator and volt factor

In this code, the resonant frequency of the resonator considering the standard values of dimensions is calculated using the formula as explained in the methodology. The initial and new geometric constants($\frac{r_a}{r_v}$) are calculated by using resonant frequency and fundamental frequency respectively. Both the constants are then compared and the change factor is stored for analysis. The volt factor is calculated which is the ratio of the amplitude of signal at resonant frequency and the maximum value of the amplitude spectrum.

Piezo Electric conversion

```
% (ASSUME DAMPING FACTOR 1 : CRITICAL DAMPING
%Properties : fres = 18khz : D = 12mm : Y = 6.3 * 10^10 : density = 7700 kg/m^3
%: T = 0.31 um : d31, piezoelectric strain constant = -175
D = 0.012; T = 0.31 * 10^-6 ; d31 = -175 ; Y = 6.3 * 10^10;
pascalMax = 10.^(dbMax/20);
factor = 0.707^((FundamentalFrequency - resonantFrequency)/126.3);
maxParray = repmat(Lt, length(Lt)/1024, 1); %since operating at resonating freq and critical damping
VoltageGen = (-3.418 * 10^-3 * volt_Factor).*maxParray ;
Z_load= 200 ; %resistance in ohms
Total_Time= t(end)-t(1);
Power= (1/Total_Time)*sum((VoltageGen.^2))/Z_load;
powerValues = (VoltageGen.^2) / Z_load;
tdel = 2.2676e-05;
totalEnergy = sum(powerValues * tdel); % Total energy in Joules
totalTime = length(VoltageGen) * tdel; % Total time
averagePowerOutput = totalEnergy / totalTime; % Average power in Watts
averagePowerOutput
```

Fig 8: MATLAB Code for calculating geometrical parameters of the resonator

RUN PROGRAM

```
% Specify the folder where the files live.
myFolder = '/Users/rxnkshitij748/Documents/MATLAB/PROJECT/Hospital noise original/test3files';
% Check to make sure that folder actually exists.  Warn user if it doesn't.
if ~isfolder(myFolder)
    errorMessage = sprintf('Error: The following folder does not exist:\n%s\nPlease specify a new folder.', myFolder);
    uiwait(warndlg(errorMessage));
    myFolder = uigetdir(); % Ask for a new one.
    if myFolder == 0
         % User clicked Cancel
         return;
    end
end
% Get a list of all files in the folder with the desired file name pattern.
filePattern = fullfile(myFolder, '*.wav'); % Change to whatever pattern you need.
theFiles = dir(filePattern);
numberfiles = length(theFiles);
powerArraywithTune = [1,numberfiles];
powerArraywithoutTune = [1,numberfiles];
for n = 1 : numberfiles
    filepath = string(theFiles(n).folder) + "/" + string(theFiles(n).name);
    powerGenTune = PowerwithTuning(filepath);
    powerGenwithoutTune = PowerwithoutTuning(filepath);
    powerArraywithTune(n) = powerGenTune;
    powerArraywithoutTune(n) = powerGenwithoutTune;
end

total_powerwithTune = sum(powerArraywithTune);
total_powerwithoutTune = nansum(powerArraywithoutTune);
```

Fig 9: MATLAB Code for using multiple audio files for power calculation

In this code, multiple audio files, from a folder, are sampled and their details are stored in an array. The array is then used for performing voltage and power calculations, with and without tuned resonator conditions, for every audio file and the result is stored in another array. Then the total average power is calculated by adding the instantaneous powers of each time interval. In case of without tune, the power comes out to be NaN, so to rectify this issue “nansum” is used instead of “sum”.

RESULTS AND INFERENCE

DATASETS UTILIZED

Our analysis integrated several datasets to assess the performance of the acoustic energy harvester. The datasets utilized were sourced from Kaggle, an online platform for datasets, and various other online repositories. They include:

1. 100 minutes of audio files featuring low-flying aircraft noises: These recordings were sourced from Kaggle and provided a comprehensive dataset capturing ambient noise generated by low-flying aircraft, facilitating an understanding of the harvester's response to such environmental conditions.
2. 50 minutes of audio files recorded from a hospital corridor: Similarly sourced from Kaggle, these recordings provided insights into ambient noise prevalent in hospital environments, enabling the evaluation of the harvester's performance under conditions typical of healthcare facilities.
3. Single-tone audio files spanning frequencies from 100 Hz to 10 kHz: These files, obtained from various online repositories, covered a wide frequency range, allowing for comprehensive analysis of the harvester's response across different frequency bands.
4. Audio signal of a drill: This dataset, sourced from miscellaneous online sources, simulated localized noise sources akin to machinery or industrial equipment, facilitating the evaluation of the harvester's efficacy in attenuating specific noise sources encountered in industrial settings.

CALCULATION OF TUNABLE CONSTANT AND FUNDAMENTAL FREQUENCY

To determine the fundamental frequency accurately, we utilize a Fast Fourier Transform (FFT) on the time series data obtained from the input signal. This analysis allows us to identify the frequency component with the highest amplitude, representing the fundamental frequency of the sound wave. We ran our model on :
Dataset Utilized : Drill Signal
Size : 291KB - 7 Seconds Industrial Drill Audio

1. Calculation of Changing Factor to tune the resonant frequency

CONSTANT	INITIAL VALUE	FINAL VALUE	FACTOR
$r_a : r_v$	0.3817	0.1341	0.3514

Table : Constants and Changing Factor Evaluated

2. Calculation of Fundamental Frequency of Input Signal

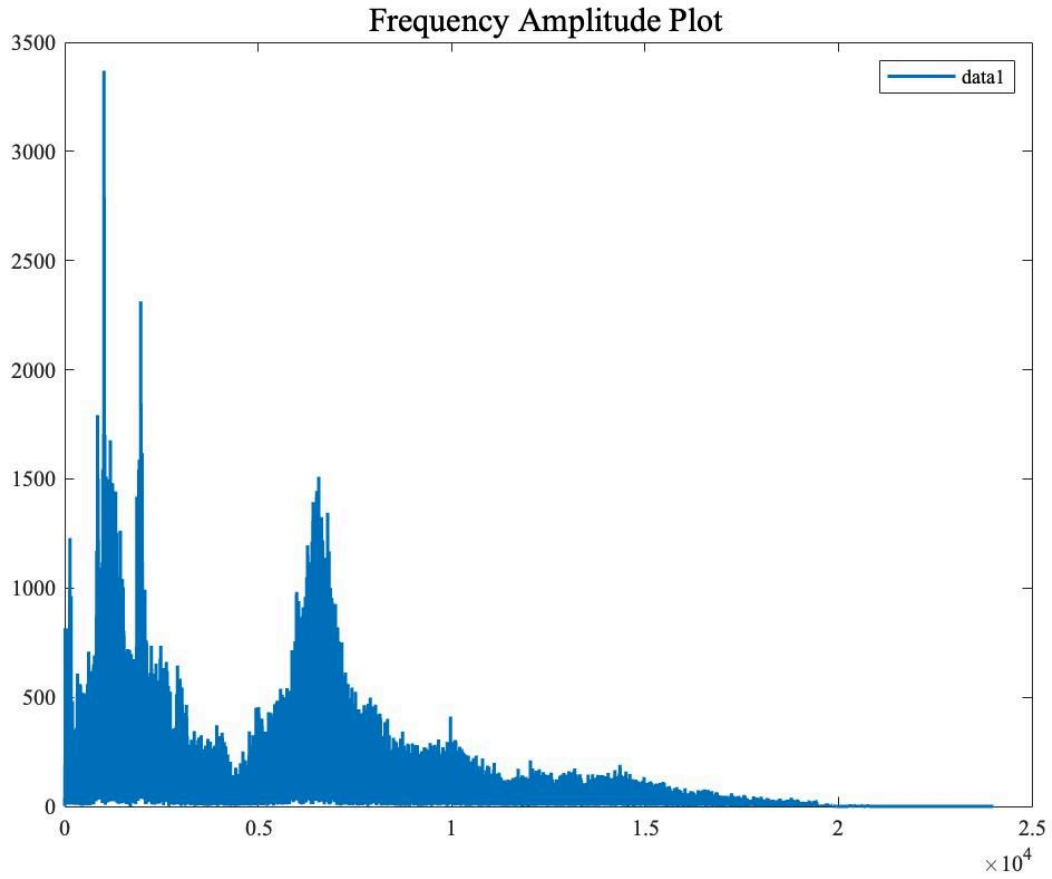


Fig : Frequency vs Amplitude Plot Of Drill Signal

Fundamental Frequency Calculated : 1016.694 Hz

COMPARISON BETWEEN OUR TUNABLE MODEL AND NON TUNABLE MODEL

Through rigorous MATLAB simulations, we compared the performance of our tunable model against that of a conventional fixed model. Our analysis revealed a significant improvement in the average power output of our tunable model across all datasets. On average, our tunable approach demonstrated an increase by a factor of 2.933×10^6 in power generation efficiency compared to the fixed model, highlighting the effectiveness of our innovative and novel mathematical model.

DETAILED ANALYSIS OF OUR SIMULATION :

Dataset Utilized : Hospital Ambient Noise
Source : Kaggle : https://www.kaggle.com/datasets/nafin59/hospital-ambient-noise
Size : 248.9 MB - 562 '.wav' files of 5 seconds each - 47 min

Total Power With tuning (in Watts)	Total Power Without Tuning (in Watts)	Average Increase In Power (P1/P2)
16.0232914775031	2.0913e-05	2.93303e+06

Table : Power Generation Comparison Of the 2 Models With Resistive 1k Load

FINAL INFERENCE AND LIMITATIONS OF OUR MODEL:

Our study showcases the integration of tunability into the Helmholtz resonator geometry for enhanced acoustic energy harvesting. While our results demonstrate promising advancements in power generation efficiency, it's essential to acknowledge the inherent limitations of our model. These include assumptions of idealized conditions and the need for further validation through experimental testing to account for real-world factors such as manufacturing tolerances and environmental variability.

In conclusion, our findings underscore the potential of tunable Helmholtz resonators in optimizing acoustic energy harvesting across diverse noise environments. Future research endeavors should focus on addressing practical challenges and refining the design for widespread implementation in renewable energy applications.

PREDICTION MODEL

DATASETS USED

Our analysis integrated several datasets to assess the performance of the acoustic energy harvester. The datasets utilized were sourced from Kaggle, an online platform for datasets, and various other online repositories. They include:

- 100 minutes of audio files featuring low-flying aircraft noises: These recordings were sourced from Kaggle and provided a comprehensive dataset capturing ambient noise generated by low-flying aircraft, facilitating an understanding of the harvester's response to such environmental conditions.
- 50 minutes of audio files recorded from a hospital corridor: Similarly sourced from Kaggle, these recordings provided insights into ambient noise prevalent in hospital environments, enabling the evaluation of the harvester's performance under conditions typical of healthcare facilities.

NEURAL NET TIME SERIES APP

The Neural Network Time Series app in MATLAB is a graphical user interface (GUI) tool that facilitates the creation, training, and validation of neural network models specifically designed for time series forecasting and prediction tasks. Time series data refers to sequential observations collected over time, such as stock prices, weather data, or sensor readings.
The key functionalities of the Neural Network Time Series app are:
Data Import and Preprocessing:

- The app allows users to import time series data from various sources, including files (e.g., CSV, Excel) or MATLAB workspace variables.
- Users can preprocess the data, such as normalizing or standardizing it to ensure consistency and improve model performance.

Model Configuration:

- Users can configure various aspects of the neural network model, including architecture (e.g., number of layers, number of neurons per layer), activation functions, and training parameters (e.g., learning rate, optimization algorithm).
- The app supports a variety of neural network architectures, including feedforward networks, recurrent neural networks (RNNs), and Long Short-Term Memory (LSTM) networks, which are well-suited for time series data modeling.

Training and Validation:

- Users can train the neural network model using the provided time series data.

- The app offers options for splitting the data into training, validation, and test sets to assess the model's performance and prevent overfitting.
- Training progress and performance metrics, such as loss functions and validation errors, are displayed in real-time, allowing users to monitor and evaluate the model's convergence and generalization ability.

Forecasting and Prediction:

- Once trained, the neural network model can be used to make predictions on unseen or future time series data.
- Users can visualize the model's predictions alongside the actual time series data to assess its accuracy and reliability.
- The app provides tools for evaluating prediction errors and assessing the model's performance metrics, such as mean squared error (MSE) or mean absolute error (MAE).

Deployment and Integration:

- After satisfactory training and validation, users can export the trained neural network model for deployment in MATLAB applications or as standalone executable files.
- The app provides options for generating MATLAB code to reproduce the entire modeling process, enabling users to customize and integrate the neural network model into their own scripts or applications.

Overall, the Neural Network Time Series app in MATLAB offers a user-friendly interface for building, training, and deploying neural network models for time series forecasting tasks, making it accessible to users with varying levels of expertise in neural networks and time series analysis.

ALGORITHM USED

The Levenberg-Marquardt algorithm, implemented in MATLAB's Neural Network Time Series app, is a robust optimization method used to train neural network models for time series analysis. It combines the advantages of gradient descent and Gauss-Newton methods, adjusting the learning rate dynamically to accelerate convergence. By iteratively updating the network weights, it minimizes the mean squared error between predicted and actual time series data, leading to accurate forecasting and prediction results. This algorithm is particularly effective for training complex neural network architectures, such as recurrent or LSTM networks, in time series applications

AIRCRAFT NOISES

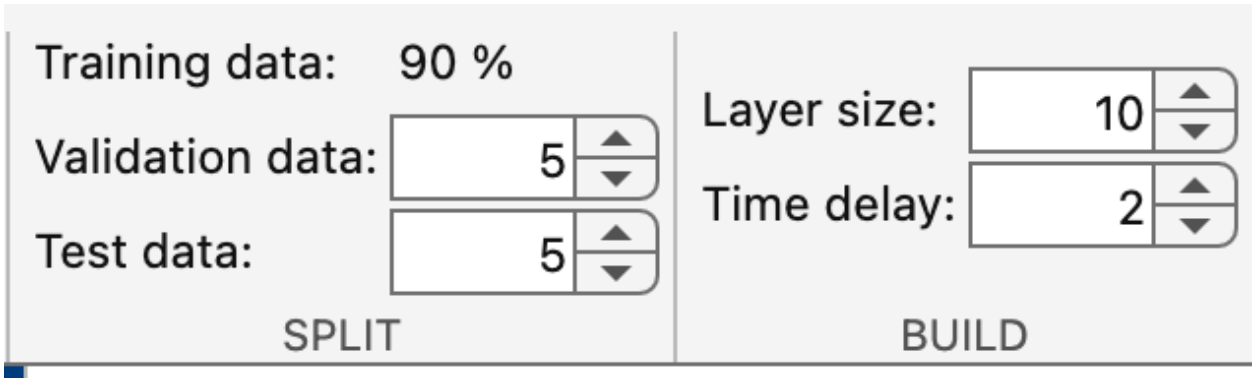


Fig : Training, Validation and Test Data Neurons Split

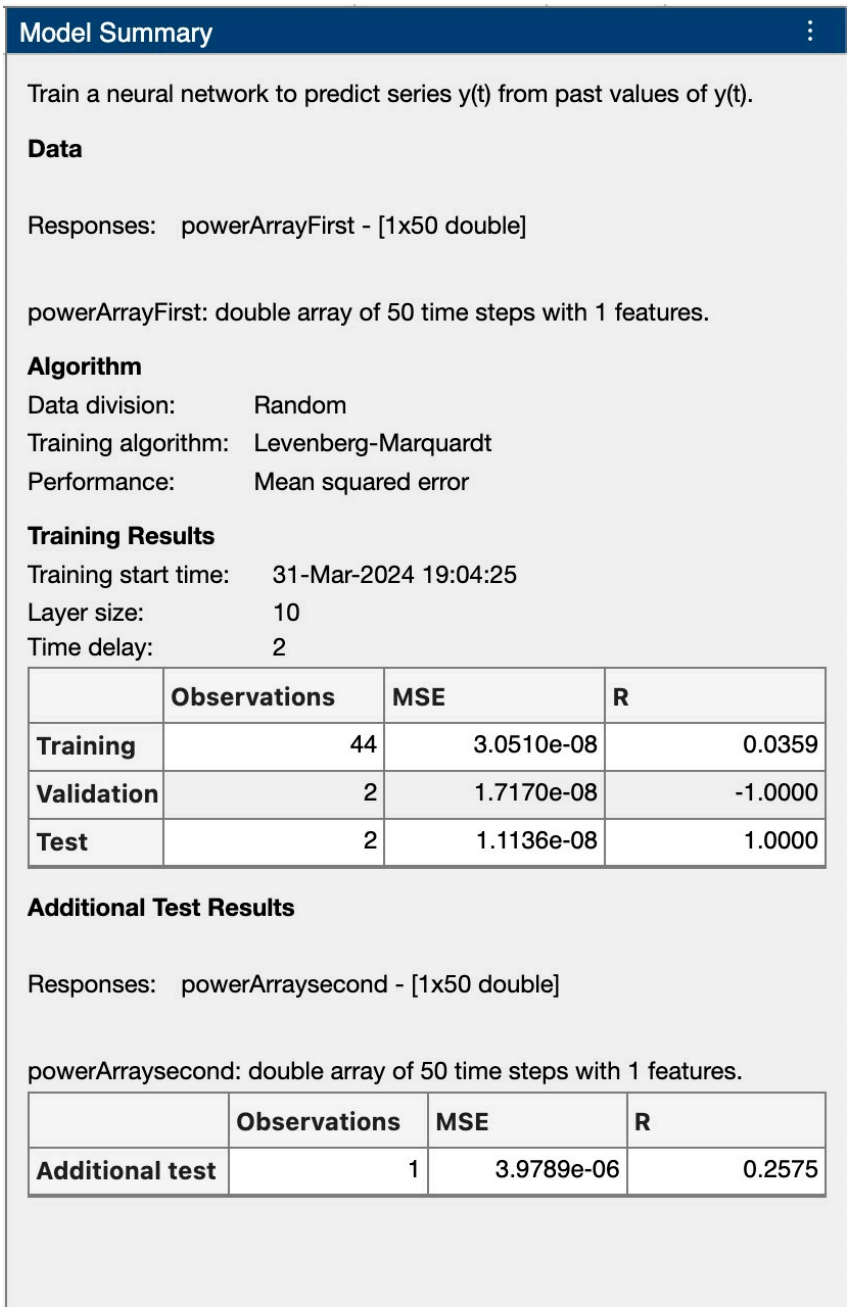


Fig : Summary Of Model

Training Results

Training finished: Reached minimum gradient

Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	5	1000
Elapsed Time	-	00:00:00	-
Performance	1.03e-07	3.05e-08	0
Gradient	4.76e-07	4.7e-08	1e-07
Mu	0.001	1e-08	1e+10
Validation Checks	0	0	6

Fig : Training Results

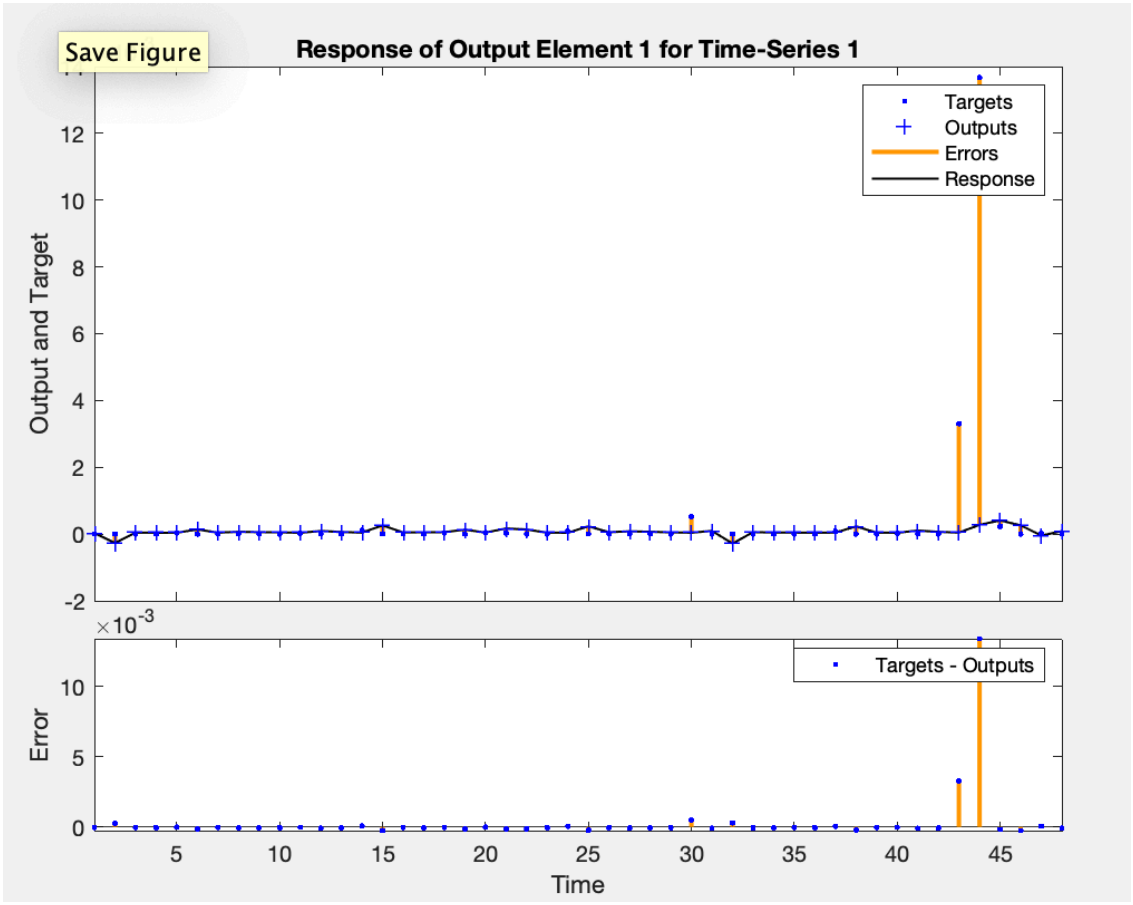


Fig: Response Plot

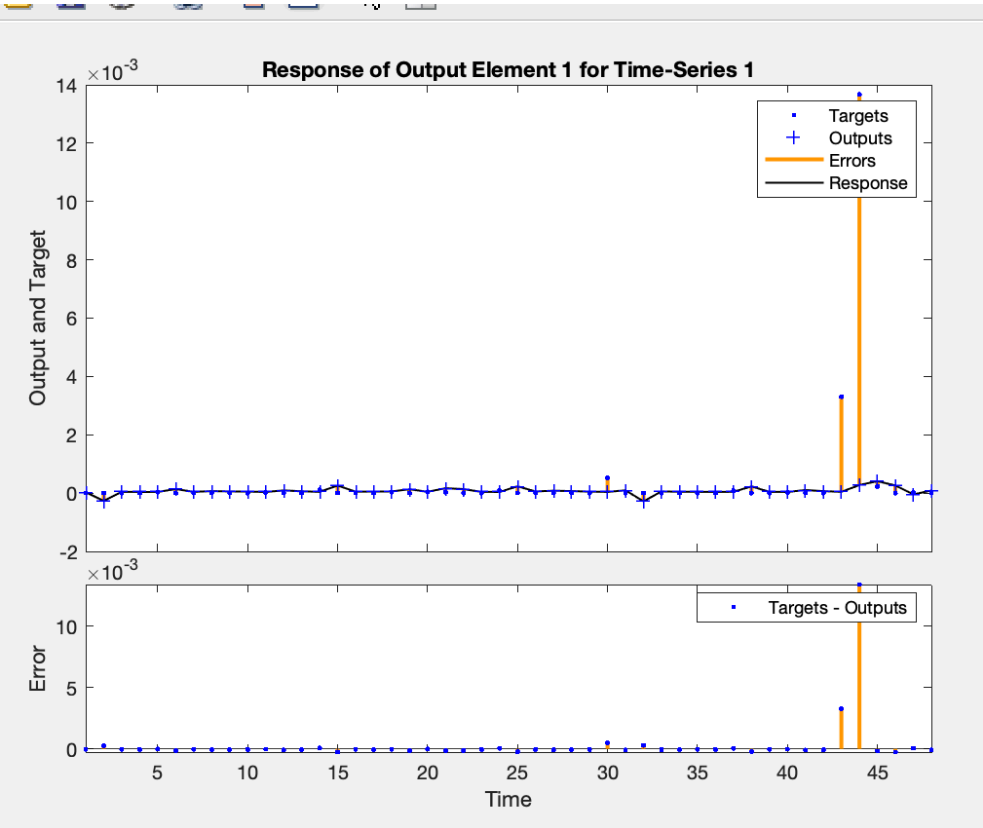


Fig: Test Response Plot

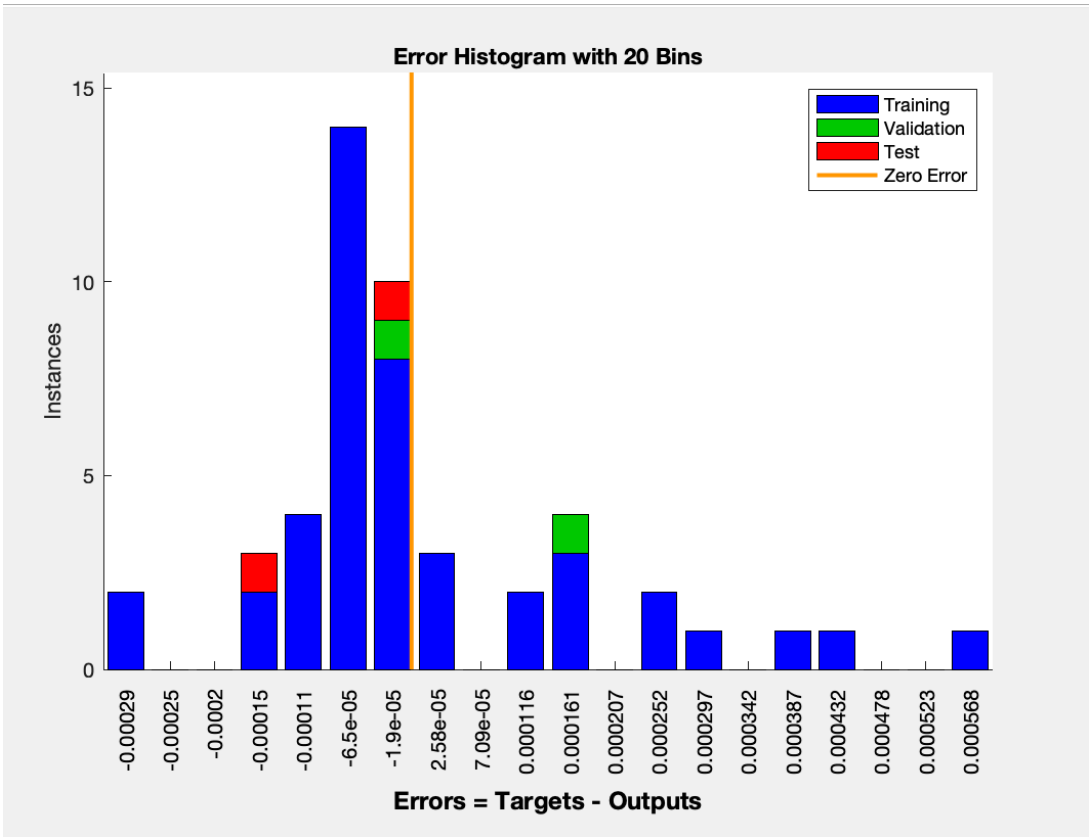


Fig: Error Histogram Plot

HOSPITAL CORRIDOR

In this test, we used the first 228 audio files from the Hospital Ambient Noise dataset to train the model and then the next 333 audio files were used to test the model.

Dataset Utilized : Hospital Ambient Noise
Source : Kaggle : https://www.kaggle.com/datasets/nafin59/hospital-ambient-noise
Size : 248.9 MB - 562 '.wav' files of 5 seconds each - 47 min

The validation split is as follows:

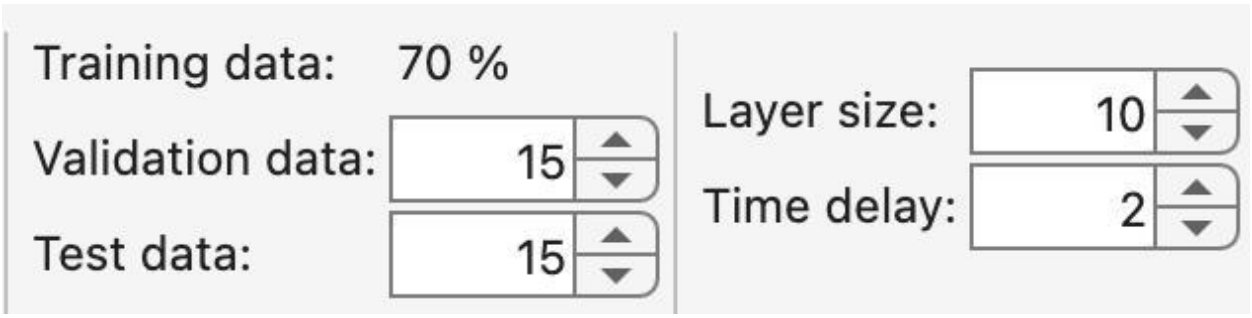
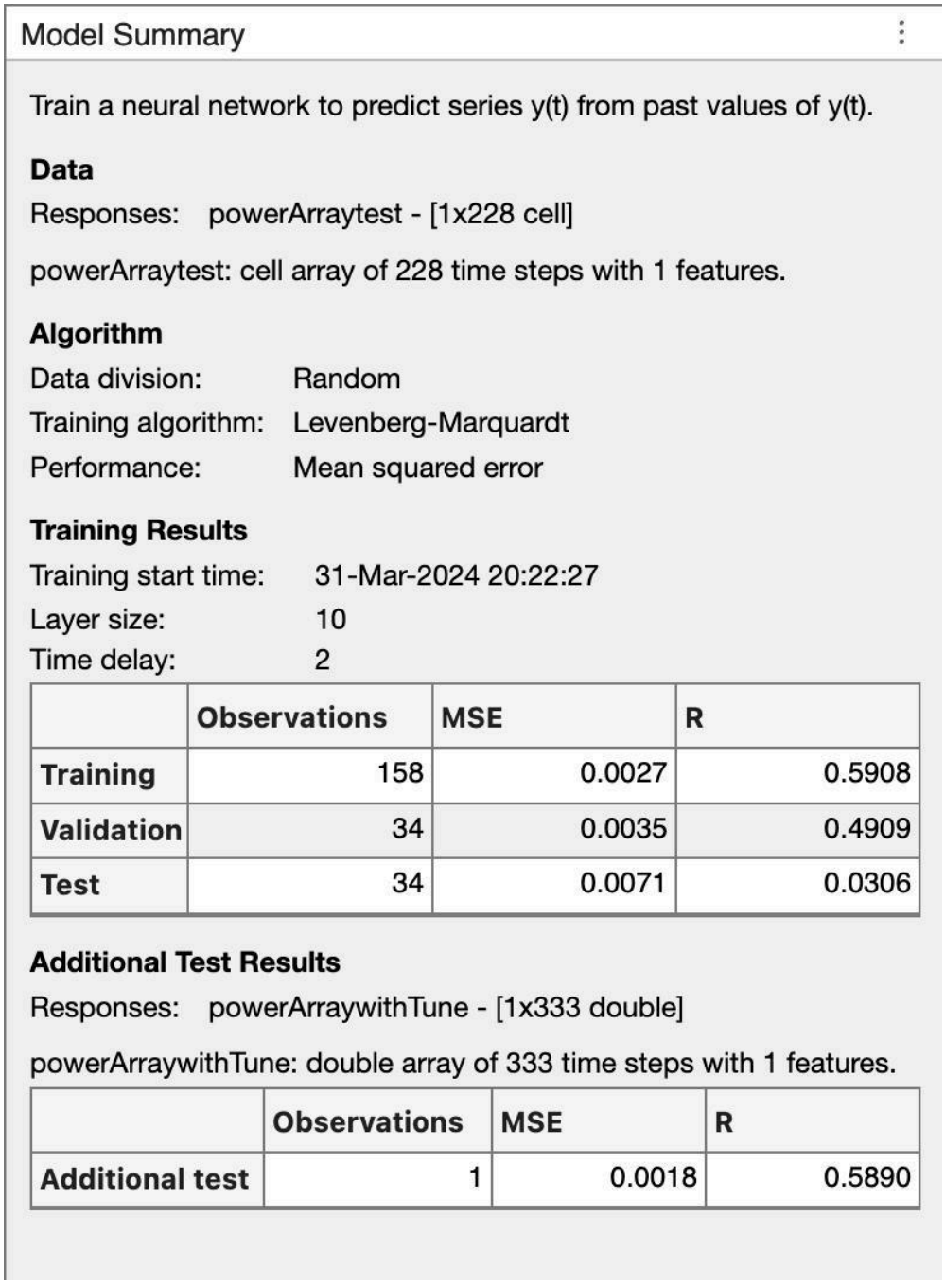


Fig : Validation Split for Hospital Ambient Noise Dataset

Result of Training

The model training summary is as follows:



Model Summary

Train a neural network to predict series y(t) from past values of y(t).

Data

Responses: powerArraytest - [1x228 cell]

powerArraytest: cell array of 228 time steps with 1 features.

Algorithm

Data division: Random

Training algorithm: Levenberg-Marquardt

Performance: Mean squared error

Training Results

Training start time: 31-Mar-2024 20:22:27

Layer size: 10

Time delay: 2

	Observations	MSE	R
Training	158	0.0027	0.5908
Validation	34	0.0035	0.4909
Test	34	0.0071	0.0306

Additional Test Results

Responses: powerArraywithTune - [1x333 double]

powerArraywithTune: double array of 333 time steps with 1 features.

	Observations	MSE	R
Additional test	1	0.0018	0.5890

Fig : Training Summary for Hospital Ambient Noise Dataset

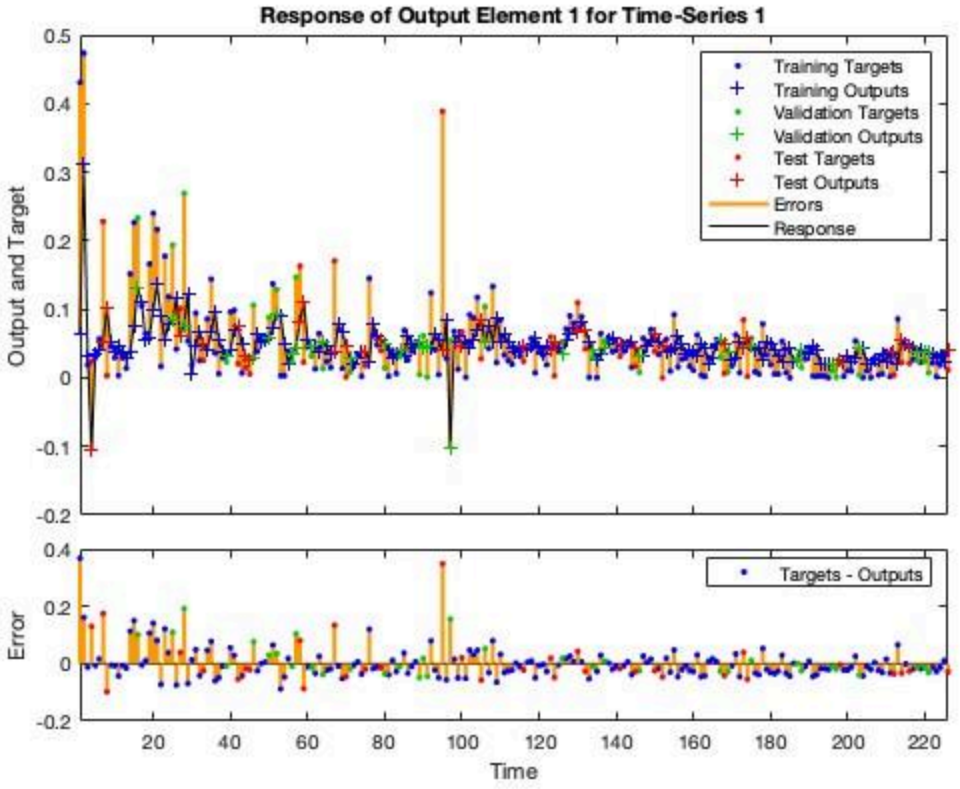


Fig : Training Response of data from Hospital Noise Dataset

Result of Testing

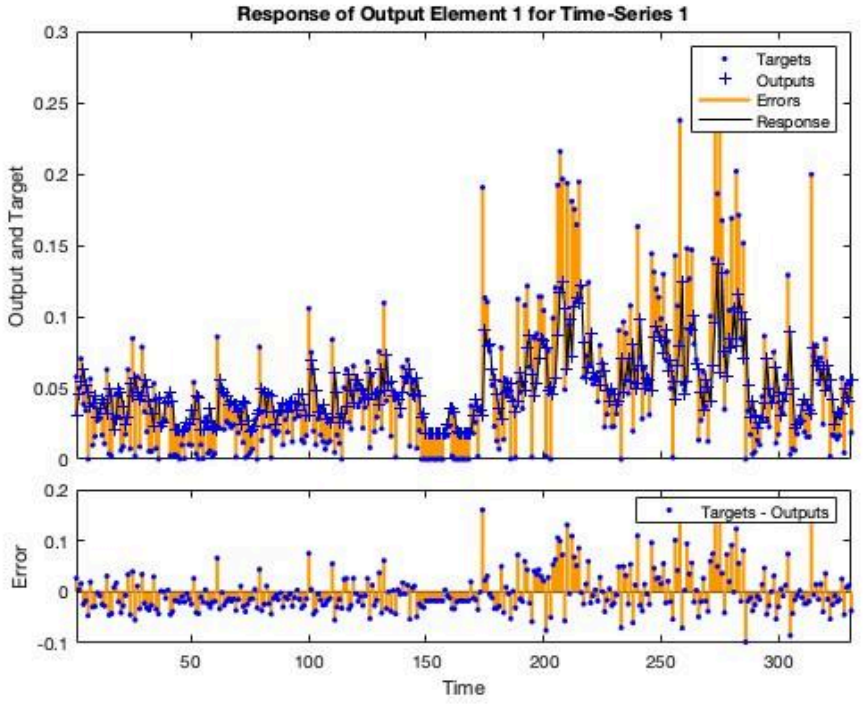


Fig : Training Response of model trained from data from Hospital Noise Dataset

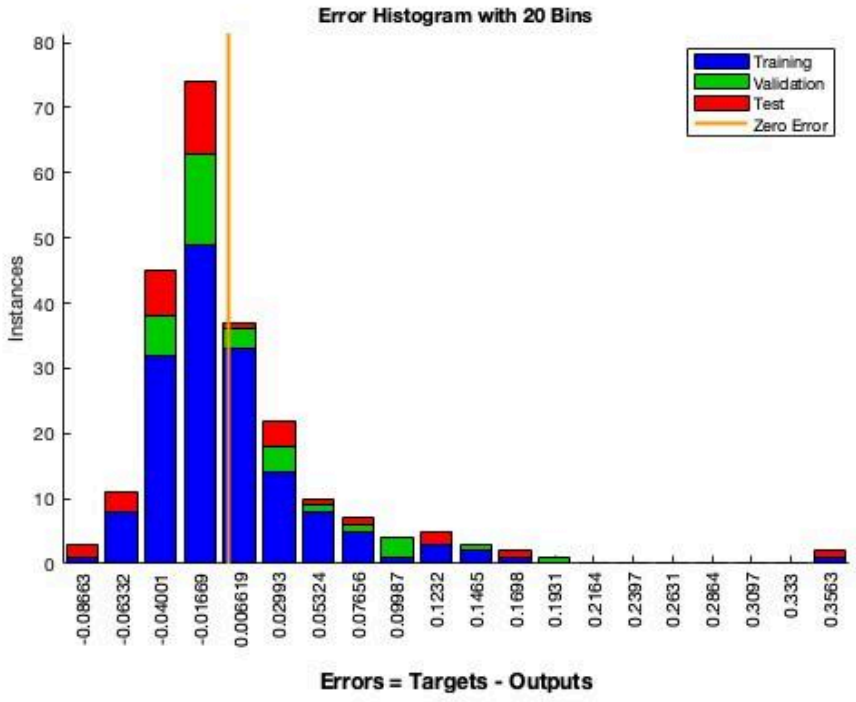


Fig : Error Histogram of model trained from Hospital Noise Dataset