

REAL-TIME SINUSOIDAL VIBRATION CANCELLATION USING EMBEDDED
SYSTEMS

by

Hüseyin Deniz Gürhan

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Electrical and Electronics Engineering

Yeditepe University

2022

REAL-TIME SINUSOIDAL VIBRATION CANCELLATION USING EMBEDDED
SYSTEMS

APPROVED BY:

Prof. Dr. Cem Ünsalan

(Thesis Supervisor)

(Marmara University)

.....

Prof. Dr. Duygun Erol Barkana

(Yeditepe University)

.....

Prof. Dr. Serkan Topaloğlu

(Yeditepe University)

.....

Prof. Dr. Oğuzhan Urhan

(Kocaeli University)

.....

Assoc. Prof. Dr. Engin Maşazade

(Marmara University)

.....

DATE OF APPROVAL: / / 20.....

I hereby declare that this thesis is my own work and that all information in this thesis has been obtained and presented in accordance with academic rules and ethical conduct. I have fully cited and referenced all material and results as required by these rules and conduct, and this thesis study does not contain any plagiarism. If any material used in the thesis requires copyright, the necessary permissions have been obtained. No material from this thesis has been used for the award of another degree.

I accept all kinds of legal liability that may arise in case contrary to these situations.

Name, Last Name

Hüseyin Deniz Gürhan

Signature

.....

ACKNOWLEDGEMENTS

The work presented in this dissertation would not have been realized without the assistance of many individuals. I take this opportunity to express my sincere gratitude to everyone who helped me during this journey.

Foremost, I would like to thank my supervisor Prof. Dr. Cem Ünsalan for his continuous guidance, motivation, and feedback throughout the process. I am aware that I pushed the limits of his patience, but even then he did not give up on me. I am also grateful to Prof. Dr. Duygun Erol Barkana and Prof. Dr. Oğuzhan Urhan for their support and contributions.

I would like to thank Tekno Kaucuk Sanayii A.Ş., especially Asil Arif Aksekili and Hasan Çakır, who provided technical and theoretical support to carry out the mechanical experiments.

I would especially like to thank Hüseyin Büyükeşmeli, who has shared this journey with me. We have shared the same class, office, and house for the past fifteen years. He has been my biggest supporter in finishing this difficult path. I am also very grateful to Dilara Hisar for her love, support, and utmost patience.

Finally, I would like to thank my family for their endless love and support, which makes everything more beautiful.

ABSTRACT

REAL-TIME SINUSOIDAL VIBRATION CANCELLATION USING EMBEDDED SYSTEMS

Active vibration cancellation is used in various mechanical systems to prevent undesired effects of vibration such as affecting system performance, generating damage or waste energy, and causing discomfort to the user or the environment. One of these systems using active vibration cancellation is an active mount system for car engines. This system is used to cancel the vibration created by the car engine and prevent it from affecting the chassis of the car. There are many methods proposed in the literature for this particular system. They can be grouped as adaptive, classical, and robust control methods. Adaptive control methods provide better results under low uncertainties. But their performance drop under high uncertainties and they may even become unstable. On the other hand, classical control methods are more robust to uncertainties but have worse response times. Finally, robust control methods have the best stability under high uncertainties, but lose their overall performance to gain it. In this study, we propose a novel method to work with better performance under high uncertainties. The proposed method is based on Kay's sinusoidal parameter estimation algorithm, which can be used at high noise levels. However, it is improved for embedded implementation using different optimization steps. First, a crude frequency is found using a fast Fourier transform. Then, the exact frequency is estimated around this crude frequency using Kay's frequency estimator with a selected search algorithm. This step is further improved using a two-stage search approach, predefined look-up tables, and iterative calculations. Then for real-time implementation, partial input sampling and specific real-time phase delay calculations are performed. Frequency and phase/amplitude calculation parts are also separated for increasing the phase estimation accuracy. Finally, the proposed method is implemented on an Arm Cortex-M4 microcontroller. The loopback simulations, electrical experiments, and mechanical experiments are performed and the proposed method is compared with the state of the art in literature.

ÖZET

GÖMÜLÜ SİSTEMLER KULLANARAK GERÇEK ZAMANLI SİNÜZOİDAL TİTREŞİM ENGELLEME

Aktif titreşim sönümleme çeşitli mekanik sistemlerde, titreşimin sistem performansını etkileme, hasar veya gereksiz enerji üretme, kullanıcıya veya çevreye rahatsızlık verme gibi istenmeyen etkilerini önlemek için kullanılmaktadır. Aktif titreşim sönümleme kullanan bu sistemlerden biri de araba motorları için aktif takoz sistemleridir. Bu sistemler araba motorunun oluşturduğu titreşimi sönümlemek ve arabanın şasesini etkilemesini önlemek için kullanılır. Bu özel sistem için literatürde önerilen birçok yöntem vardır. Bunlar adaptif, klasik ve gürbüz kontrol yöntemleri olarak gruplandırılabilirler. Adaptif kontrol yöntemleri düşük seviye belirsizlikler altında daha iyi sonuçlar sağlar. Ancak performansları yüksek belirsizlikler altında düşer ve hatta kararsız hale gelebilirler. Öte yandan, klasik kontrol yöntemleri belirsizliklere karşı daha dayanıklıdır ancak daha kötü tepki sürelerine sahiptirler. Son olarak, gürbüz kontrol yöntemleri, yüksek belirsizlikler altında en iyi kararlılığa sahiptir, ancak bunu elde etmek için ortalama performanslarından kaybederler. Bu çalışmada, yüksek belirsizlikler altında daha iyi performansla çalışabilen yeni bir yöntem önermekteyiz. Önerilen yöntem, Kay'ın yüksek gürültü seviyelerinde kullanılabilen sinüzoidal parametre tahmin algoritmasına dayanmaktadır. Ancak, bu yöntem gömülü uygulamalarda kullanabilmek için farklı iyileştirme adımları kullanılarak geliştirilmiştir. İlk olarak, hızlı Fourier dönüşümü kullanılarak kabaca bir frekans değeri bulunur. Daha sonra, Kay'ın frekans kestiricisi bu kaba frekans etrafında seçilen bir arama algoritması ile kullanılır ve kesin frekans tahmin edilir. Bu adım, iki aşamalı bir arama yaklaşımı, önceden tanımlanmış arama tabloları ve yinelemeli hesaplamalar kullanılarak daha da geliştirilmiştir. Daha sonra gerçek zamanlı uygulama için kısmi giriş örnekleme ve özel gerçek zamanlı faz gecikmesi hesaplamaları uygulanır. Frekans ve faz/genlik hesaplama bölümleri de faz tahmin doğruluğunu artırmak için farklı hızlarda ayrı olarak hesaplanır. Son olarak, önerilen yöntem bir Arm Cortex-M4 mikrodenetleyici üzerinde çalıştırılmıştır. Geri döngü simülasyonları, elektriksel deneyler ve mekanik deneyler gerçekleştirilmiş ve önerilen yöntem literatürdeki yöntemler ile karşılaştırılmıştır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	iv
ABSTRACT.....	v
ÖZET.....	vi
TABLE OF CONTENTS.....	vii
LIST OF FIGURES	x
LIST OF TABLES.....	xiv
LIST OF SYMBOLS/ABBREVIATIONS.....	xvi
1. INTRODUCTION	1
1.1. PROBLEM STATEMENT.....	2
1.2. LITERATURE REVIEW.....	6
1.3. NOVELTY OF THE STUDY	9
1.4. COMPETITIVE METHODS IN LITERATURE.....	11
1.4.1. Phase-Locked Loop.....	12
1.4.2. Least Mean Squares Filter.....	14
1.4.3. Recursive Least Squares Filter.....	15
1.5. LAYOUT OF THE THESIS.....	17
2. THE PROPOSED ACTIVE VIBRATION CANCELLATION METHOD.....	18
2.1. KAY'S ALGORITHM FOR SINUSOIDAL PARAMETER ESTIMATION.....	18
2.2. THE IMPROVEMENTS IN KAY'S ALGORITHM.....	21
2.2.1. Applying Fast Fourier Transform.....	21
2.2.2. Using Search Algorithms.....	23
2.2.2.1. Bisection Method.....	23
2.2.2.2. Golden-Mean Search Method.....	25
2.2.2.3. Binary Search Method.....	26
2.2.2.4. Secant Method.....	26
2.2.2.5. Comparison of Search Methods.....	26
2.2.3. Predefined Look-up Tables and Iterative Calculations.....	29
3. EMBEDDED SYSTEMS USED IN IMPLEMENTATION.....	31
3.1. EMBEDDED HARDWARE PLATFORMS.....	31
3.2. EMBEDDED SOFTWARE PLATFORMS.....	34

3.2.1.	Development Environment	35
3.2.2.	Embedded Software Packages	35
3.2.3.	CMSIS Library	36
4.	EMBEDDED SYSTEM IMPLEMENTATION.....	37
4.1.	PERFORMANCE CRITERIA OF KAY'S ALGORITHM PARAMETERS	37
4.1.1.	Generating White Gaussian Noise.....	37
4.1.2.	Signal-To-Noise Ratio.....	37
4.1.3.	Frequency Step Size.....	39
4.1.4.	Number of Elements	40
4.2.	IMPROVING PERFORMANCE VIA THE STEPS OF THE PROPOSED METHOD	42
4.2.1.	Fast Fourier Transform.....	42
4.2.2.	Search Algorithm.....	43
4.2.3.	Iterative Calculations.....	44
4.3.	EMBEDDED IMPLEMENTATION IMPROVEMENTS.....	45
4.3.1.	Obtaining Input Samples.....	45
4.3.2.	Calculating Phase Delay	46
4.3.3.	Using Embedded Software Architecture.....	48
4.4.	EMBEDDED IMPLEMENTATION OF THE PROPOSED AND COMPETITIVE METHODS	50
4.4.1.	Proposed Method.....	50
4.4.2.	Least Mean Squares Filter.....	51
4.4.3.	Recursive Least Squares Filter.....	52
4.4.4.	Phase-Locked Loop.....	52
4.5.	PERFORMANCE OF HARDWARE PLATFORMS.....	54
5.	LOOPBACK SIMULATIONS FOR VERIFICATION.....	56
5.1.	SINGLE-FREQUENCY SIGNAL.....	57
5.1.1.	Proposed Method.....	57
5.1.2.	Comparison with Competitive Methods.....	60
5.2.	FREQUENCY SWEEP.....	62
5.2.1.	Proposed Method.....	63
5.2.2.	Comparison with Competitive Methods.....	66

5.3. MULTI-FREQUENCY SIGNAL	69
6. EXPERIMENTS ON ELECTRICAL SYSTEMS	70
6.1. ELECTRICAL SETUP	70
6.2. EXPERIMENTS	71
6.2.1. Single-Frequency Signal	71
6.2.1.1. Proposed Method	71
6.2.1.2. Comparison with Competitive Methods	73
6.2.2. Frequency Sweep	75
6.2.2.1. Proposed Method	75
6.2.2.2. Comparison with Competitive Methods	77
7. EXPERIMENTS ON MECHANICAL SYSTEMS	80
7.1. MECHANICAL SETUP	80
7.1.1. Actuators	81
7.1.1.1. Accelerometers	83
7.1.1.2. Vibration Generator	83
7.1.1.3. Motor Driver	85
7.2. SYSTEM IDENTIFICATION FOR VOICE COIL	86
7.3. EXPERIMENTS	88
7.3.1. Single-Frequency Signal	88
7.3.1.1. Proposed Method	89
7.3.1.2. Comparison with Competitive Methods	89
7.3.2. Frequency Sweep	92
7.3.2.1. Proposed Method	92
8. CONCLUSIONS	95
REFERENCES	98

LIST OF FIGURES

Figure 1.1.	Block diagram of the active vibration cancellation system of the car engine...	3
Figure 1.2.	3D model of (a) an active mount, and (b) its cross section image.....	4
Figure 1.3.	Sinusoidal vibration creation via ignition pulses of cylinders [1].....	4
Figure 1.4.	The sinusoidal vibration patterns of the car engine for (a) constant speed, (b) acceleration, (c) deceleration, and (d) gear change.....	5
Figure 1.5.	The instability of the adaptive control method for 30% uncertainties [2].....	12
Figure 1.6.	The block diagram of the PLL.....	13
Figure 1.7.	The PLL phase detector block.....	13
Figure 1.8.	The PLL loop filter block.....	14
Figure 1.9.	The block diagram of the LMS method.....	15
Figure 1.10.	The block diagram of the LMS predictor.....	16
Figure 2.1.	Example graph of MLE estimator used to find $\hat{f}_0$	21
Figure 2.2.	FFT results for different approaches.....	23
Figure 3.1.	STM32F429I-DISC1 board [3].....	33
Figure 3.2.	32F746GDISCOVERY board [4].....	33
Figure 3.3.	NUCLEO-H743ZI2 board [5].....	34
Figure 3.4.	STM32CubeIDE home screen.....	35
Figure 4.1.	Estimation MSE wrt SNR. (a) Frequency, (b) phase, (c) amplitude.....	39
Figure 4.2.	Estimation MSE wrt frequency step size. (a) Frequency, (b) phase, (c) amplitude.....	41

Figure 4.3. Estimation MSE wrt the number of elements. (a) Frequency, (b) phase, (c) amplitude.....	41
Figure 4.4. Obtaining input samples in a real-time system.....	46
Figure 4.5. 70 Hz. sinusoidal signal when the sampling frequency is one kHz.....	47
Figure 5.1. The block diagram of the loopback simulations.....	56
Figure 5.2. Simulation results for the proposed method. (a) Vibration, (b) noise, (c) cancellation, and (d) error signals.....	58
Figure 5.3. MSE of the noise removed error signal wrt SNR.....	58
Figure 5.4. FFT results for vibration and error signal.....	59
Figure 5.5. Vibration suppression level wrt SNR.....	59
Figure 5.6. The response time of the proposed method for (a) 10 dB, and (b) 30 dB SNR.....	60
Figure 5.7. MSE of the noise removed error signal wrt SNR for each method.....	61
Figure 5.8. Vibration suppression level wrt SNR for each method.....	61
Figure 5.9. The response time of (a) LMS, (b) RLS, and (c) PLL for 10 dB SNR.....	62
Figure 5.10. Simulation results for the proposed method. (a) Vibration, (b) noise, (c) cancellation, and (d) error signals.....	63
Figure 5.11. MSE of the noise removed error signal wrt SNR.....	64
Figure 5.12. The FFT results for vibration and error signals.....	65
Figure 5.13. The vibration suppression level wrt SNR.....	65
Figure 5.14. The response time of the proposed method for sweep cycle change.....	66
Figure 5.15. MSE of the noise removed error signal wrt SNR for each method.....	67
Figure 5.16. The vibration suppression level wrt SNR for each method.....	67

Figure 5.17. The response time of (a) LMS, (b) RLS, and (c) PLL for sweep cycle change at 10 dB SNR.....	68
Figure 6.1. The block diagram of the electrical experiment.....	70
Figure 6.2. The general electrical test setup.....	71
Figure 6.3. Obtained error, cancellation and vibration signals for the proposed method..	72
Figure 6.4. FFT results of the vibration and error signals for the proposed method.	72
Figure 6.5. The response time of the proposed method.....	73
Figure 6.6. Obtained error, cancellation and vibration signals for (a) LMS, (b) RLS, and (c) PLL methods.....	74
Figure 6.7. The response time of (a) LMS, (b) RLS, and (c) PLL methods for the electrical experiment.....	75
Figure 6.8. Obtained error, cancellation and vibration signals for the proposed method..	76
Figure 6.9. The FFT results of the vibration and error signals for the proposed method..	76
Figure 6.10. The response time of the proposed method.....	77
Figure 6.11. Obtained error, cancellation and vibration signals for (a) LMS, (b) RLS, and (c) PLL methods.....	78
Figure 6.12. The response time of (a) LMS, (b) RLS, and (c) PLL methods for the electrical experiment.....	79
Figure 7.1. The block diagram of the mechanical experiment.....	80
Figure 7.2. The general mechanical test setup.....	81
Figure 7.3. ERM [6], LRA [7], and piezo [8] actuators.....	82
Figure 7.4. The solenoid and voice coil actuators [9].....	82
Figure 7.5. The voice coil used for the mechanical experiment.....	83

Figure 7.6. The accelerometer used for the mechanical experiment.....	84
Figure 7.7. The shaker used for the mechanical experiment.....	84
Figure 7.8. The motor driver used for the mechanical experiment.....	85
Figure 7.9. The block diagram of the real-time implementation.....	86
Figure 7.10. The vibration signal obtained from the accelerometer connected to (a) the shaker, and (b) the voice coil.....	86
Figure 7.11. The frequency response of the obtained transfer function. (a) Magnitude, and (b) Phase.....	87
Figure 7.12. (a) The input voltage signal applied to the shaker, and (b) the voice coil vibration signal obtained from the accelerometer.....	87
Figure 7.13. The frequency response of the obtained transfer function. (a) Magnitude, and (b) Phase.....	88
Figure 7.14. Obtained error and vibration signals when the proposed method (a) is not activated, and (b) is activated.....	89
Figure 7.15. The FFT results of the vibration and error signals for the proposed method..	90
Figure 7.16. The response time of the proposed method.....	90
Figure 7.17. Obtained error and vibration signals when the PLL-based method (a) is not activated, and (b) is activated.....	91
Figure 7.18. The response time of the PLL-based method.....	92
Figure 7.19. Obtained error and vibration signals when the proposed method (a) is not activated, and (b) is activated.....	93
Figure 7.20. The FFT results of the vibration and error signals for the proposed method..	93
Figure 7.21. The response time of the proposed method.....	94

LIST OF TABLES

Table 1.1. The performance comparison of active vibration cancellation methods for active mounts.....	11
Table 1.2. The performance comparison of the proposed method and FxLMS method for high uncertainties.....	11
Table 3.1. Clock speed and memory space comparison of different Arm Cortex-M microcontrollers.....	32
Table 3.2. Current/power consumption and price comparison of different Arm Cortex-M microcontrollers.....	32
Table 4.1. The estimation results for varying SNR values.....	38
Table 4.2. The estimation results for varying frequency step size values.....	40
Table 4.3. The estimation results for varying number of elements values.....	42
Table 4.4. The estimation results for Kay's method and FFT-based method.....	43
Table 4.5. The estimation results for selected search algorithms.....	43
Table 4.6. The estimation results for improved binary search-based method.....	44
Table 4.7. The estimation results for the improved binary search-based method with iterative calculations.....	45
Table 4.8. Execution time comparison of selected hardware platforms.....	55
Table 4.9. The memory usage comparison of selected hardware platforms.....	55
Table 5.1. RAM and flash memory usage of the proposed method.....	60
Table 5.2. The memory usage comparison of vibration cancellation methods.....	62
Table 5.3. RAM and flash memory usage of the proposed method.....	66
Table 5.4. The response time comparison of vibration cancellation methods.....	68

Table 5.5.	The estimated frequencies when frequency separations are large.....	69
Table 5.6.	The estimated frequencies when frequency separations are small.	69
Table 6.1.	The vibration suppression comparison of the electrical experiment.	75
Table 6.2.	The vibration suppression comparison of the electrical experiment.	79
Table 7.1.	The vibration suppression comparison of the mechanical experiment for constant frequency.....	91



LIST OF SYMBOLS/ABBREVIATIONS

AVC	Active vibration cancellation
PID	Proportional-integral-derivative
FIR	Finite impulse response
IIR	Infinite impulse response
LMS	Least mean squares
RLS	Recursive least squares
DSP	Digital signal processor
MCS	Minimal controller synthesis
FxLMS	Filtered-x least mean square
IPI	Input point inertance
Er-MCSI	Error-based minimal control synthesis with integral action
LQR	Linear quadratic regulator
MRAC	Model reference adaptive control
MLE	Maximum likelihood estimation
FFT	Fast Fourier transform
PLL	Phase-locked loop
ANC	Active noise cancellation
VCO	Voltage controlled oscillator
NCO	Numerically controlled oscillator
WGN	White Gaussian noise
PDF	Probability density function
SNR	Signal-to-noise ratio
DFT	Discrete Fourier transform
FPU	Floating point unit
ADC	Analog to digital converter
DAC	Digital to analog converter
DMA	Direct memory access
USART	Universal synchronous/asynchronous receiver/transmitter
SPI	Serial peripheral interface

CAN	Controller area network
PWM	Pulse-width modulation
IDE	Integrated development environment
HAL	Hardware abstraction layer
CMSIS	Common microcontroller software interface standard
RTOS	Real-time operating system
MSE	Mean squared error
ERM	Eccentric rotating mass
LRA	Linear resonant actuator



1. INTRODUCTION

Vibration can be explained as the oscillating motion of a medium forced out of its state of equilibrium. Vibrations are present almost everywhere, as we can hear due to the vibration of the eardrums and see due to the vibration of light waves. However, vibrations are mostly undesirable, especially in mechanical and civil engineering. Because they can affect system performance or produce deformations, damage, and waste energy. They also cause discomfort to the user or the environment. But most importantly, when the vibration is equal to the natural frequency of the system, fatal failures or catastrophic destruction can occur. Hence, canceling or controlling these vibrations is a great necessity.

Vibration cancellation is a specific method for reducing the undesirable effects of vibration signals on a system. In general, special materials such as rubber or foam are used for vibration cancellation and this method is called passive vibration cancellation [10,11]. It is quite cheap and does not require an external energy source. But their performance is limited and complete vibration cancellation is not possible, especially for a wide range of vibration frequencies. With the rapid development of technology in recent years, additional electronic systems such as actuators, sensors, and controllers have been integrated into vibration cancellation systems. This approach is called active vibration cancellation (AVC). Vibration cancellation can be achieved efficiently using AVC at the expense of high complexity and high costs. AVC is used in daily life systems such as cars [12] and buildings [13], and more complex systems such as ultra-precision machines [14] and space shuttles [15].

In the literature, there are several methods for solving the active vibration cancellation problem for different systems. The first group of these methods is based on the feedback approach. Kang *et al.* [16] proposed an adaptive notch filter to cancel the resonant vibration of an actuator in a disk drive servo system. The proposed method quickly identifies the resonant frequency of the actuator and automatically adjusts the center frequency. Abdeljaber *et al.* [17] proposed a neural network-based controller to cancel the vibrations acting on the flexible cantilever plates. The proposed method forms a neuro controller along with a Kalman filter to adjust the optimal voltage applied on the piezoelectric patches. Hadi *et al.* [18] proposed an adaptive proportional–integral–derivative (PID) controller for vibration cancellation of the horizontal flexible plate structure. The parameters of the PID controller are tuned by an intelligent

particle swarm optimization algorithm. He *et al.* [19] proposed a novel recursive least squares (RLS) based adaptive filter for real-time active vibration cancellation. It uses Taylor series prediction to eliminate the delay-induced phase shift with higher accuracy. Zhang *et al.* [20] proposed an adaptive filtered-x LMS algorithm for active vibration cancellation of multistage gear transmission.

The second group is based on the feedforward control approach. Ma *et al.* [21] proposed an adaptive filtered-u least mean square (LMS) feedforward algorithm to control resonant oscillations of a piezoelectric clamped plate. The finite impulse response (FIR) filter has no poles in the transfer function, therefore the filtered-x LMS controller cannot be effectively tuned to cancel vibration when the reference signal is affected by the feedback control part. To solve this problem, a filtered-u LMS adaptive feed-forward controller with an infinite impulse response (IIR) filter structure is introduced. Alma *et al.* [22] proposed a model-based feedforward controller based on the H-infinity optimal control design. A controller order reduction technique is also proposed to reduce complexity without sacrificing performance. Chen *et al.* [23] proposed a robust adaptive feedforward controller to cancel the vibrations affecting the dual-stage actuator in the track-following mode in a hard disc drive with multiple actuators. A feedforward reference structure is used to make the adaptive controller robust to fluctuations in the system.

1.1. PROBLEM STATEMENT

The main purpose of this study is to cancel the vibration created by the car engine and prevent it from affecting the chassis of the car. This provides better comfort and extends the life of the engine. A general active vibration cancellation system of the car engine is given in Figure 1.1. Here, vibration created by the car engine is obtained by the controller with the help of an accelerometer. Then, a cancellation algorithm is applied to this signal, and a cancellation signal is created. Finally, this signal is applied to active mounts and canceled vibration is observed using the other accelerometer placed on the chassis.

In general, passive engine mounts are used to suppress vibration. But modern cars have higher speeds and more powerful engines, and need improved systems to cancel vibrations

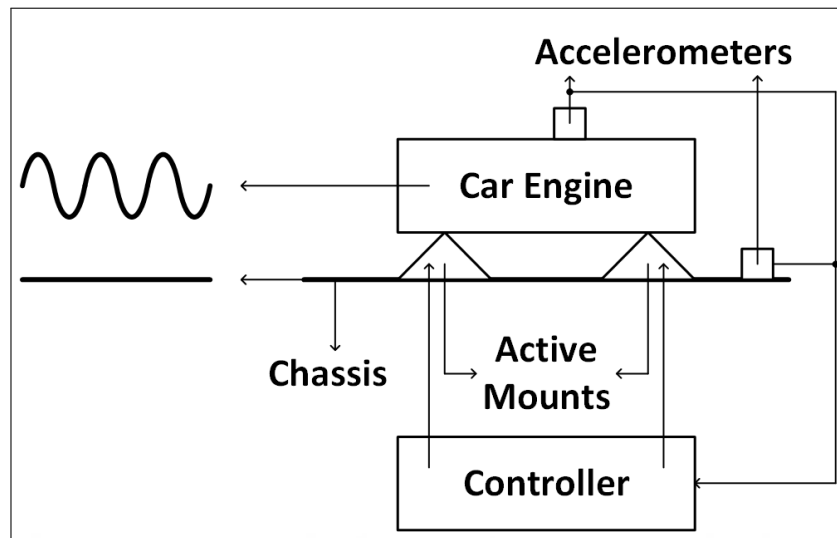


Figure 1.1. Block diagram of the active vibration cancellation system of the car engine.

that cannot be prevented with passive mounts. Hence, recent studies on active mounts are of great importance. Active mounts need an actuator to generate a cancellation signal. There are different types of actuators used in literature as solenoid [24], piezoelectric [25], and linear oscillatory [26] actuators. But voice coil actuators [27,28] are highly preferred to use in active mounts because they are special servo motors with linear control characteristics and high force output. A 3D model of an active mount with voice coil actuator and its cross section image is shown in Figure 1.2a and Figure 1.2b, respectively. Here, the voice coil placed at the bottom is used to change the pressure of the chamber formed by the upper and lower black rubber caps to cancel the engine vibration.

Vibrations generated by the engine at idle or from the road are typically at frequencies between 1 and 30 Hz with high amplitude. To control these vibrations, the engine mount must be rigid and strongly damped. However, the vibrations generated by the engine during operation are typically at frequencies between 30 and 200 Hz with decreasing amplitude. To control these vibrations, the engine mount must be flexible and lightly damped. So, the engine mount must satisfy these two essential but contradictory criteria [29]. Active mounts are used to fulfill these two criteria in the desired frequency range.

The frequency range we are interested in for active vibration cancellation is between 20 Hz and 120 Hz. Because the vibrations with frequencies below 20 Hz are canceled using passive vibration suppression. On the other hand, the vibrations with frequencies above 120

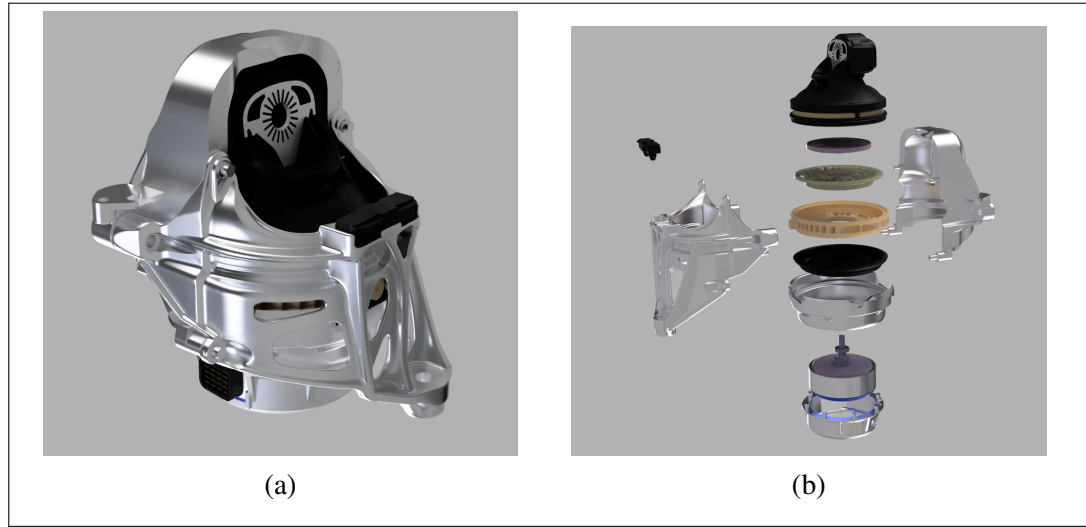


Figure 1.2. 3D model of (a) an active mount, and (b) its cross section image.

Hz are so small in amplitude, hence they are ignored. Moreover, most modern consumer cars today can drive up to a maximum of 7200 RPM. As shown in many studies [30,31,32], engine vibrations have a sinusoidal characteristic. Especially, Romling *et al.* [1] explains the relationship between the engine cylinder ignition sparks and the generated sinusoidal vibration signal in detail as shown in Figure 1.3.

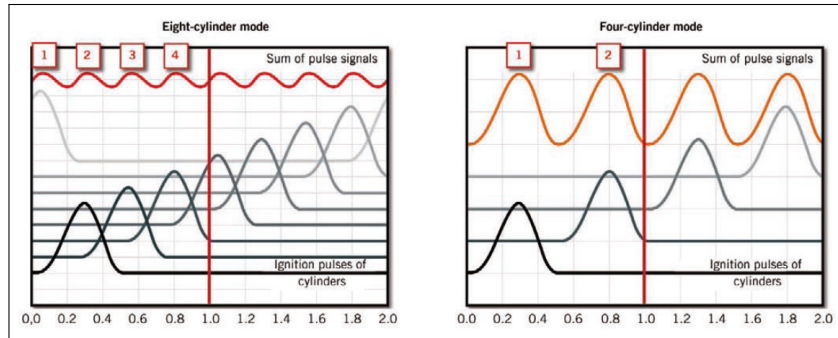


Figure 1.3. Sinusoidal vibration creation via ignition pulses of cylinders [1].

A car engine produces four types of sinusoidal vibration patterns, as shown in Figure 1.4. At constant speed, single frequency sinusoidal signal is produced as shown in Figure 1.4a. During acceleration and deceleration, frequency sweep signals are produced as shown in Figure 1.4b and Figure 1.4c. During a gear change, sudden frequency change is produced as shown in Figure 1.4d.

Actually, there are harmonic vibrations produced by the engine, as stated in [33,34,35].

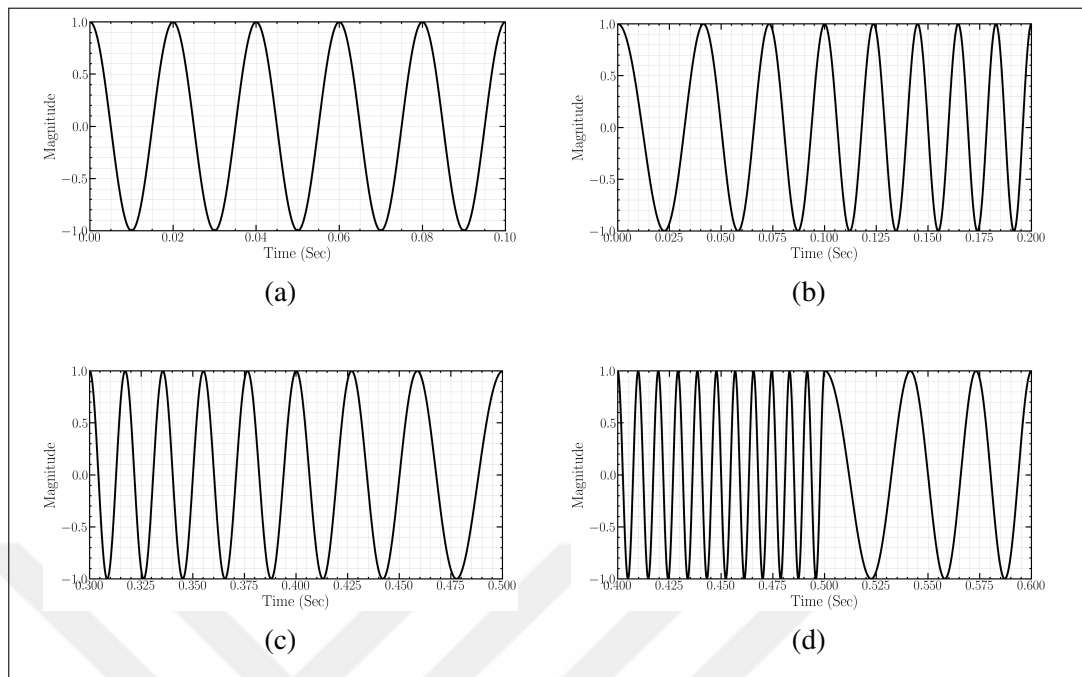


Figure 1.4. The sinusoidal vibration patterns of the car engine for (a) constant speed, (b) acceleration, (c) deceleration, and (d) gear change.

But harmonics by nature have a decreasing amplitude as the frequency increases. We also explain that the vibrations generated by the engine under operating conditions typically occur at frequencies between 30 and 200 Hz with decreasing amplitude. Hence, the harmonic vibrations are negligible and only the fundamental frequency is important for the active vibration cancellation system.

Simple microcontrollers are preferred to a PC or digital signal processor (DSP) as controllers in active vibration cancellation systems. They are simpler, cheaper, and have low power consumption. However, they have a lower processor speed than PC or DSP. They are also severely limited in terms of RAM and flash memory. Hence, the proposed method for active vibration cancellation should take these limitations into consideration.

Finally, the vibration data obtained from the accelerometer may be embedded in a high level of noise. This additional noise can be generated by accelerometers as a measurement error. The distance between the controller and the accelerometers also affects the measurement results. Road conditions can also generate additional noise for the system. Fakhari *et al.* [2] explained that these uncertainties can lead to instability of the designed controller. Hence, the proposed active vibration cancellation method should work even under excessive noise conditions.

Zhou *et al.* [36] also presented that the actual vehicle-vibration signal has disturbances and noise with an approximate Gaussian distribution and proved this through simulations. So the proposed method should also take this into account.

1.2. LITERATURE REVIEW

In this section, we review and list related work in the literature dealing with active vibration cancellation of car engine using active mounts in a broad spectrum. The first group of these methods is based on adaptive control approaches such as LMS filters and improved versions, RLS filters and improved versions, and Minimal Controller Synthesis (MCS) methods. Togashi *et al.* [37], Raoofy *et al.* [38], and Vahdati *et al.* [39] proposed basic filtered-x least mean square (FxLMS) based controllers to be used for vibration cancellation on active mount systems. FxLMS control algorithm has become the primary tool for reducing vibration or noise and it is the most common control strategy for controlling active engine mounts. The main design parameters of the FxLMS algorithm are the step size or convergence coefficient, the filter length, and the leakage factor. The convergence speed of the FxLMS algorithm is influenced by the step size. When the filter converges to a steady state, the step size has a slight influence on the performance. The overall stability of the system is affected by the leakage factor. Hausberg *et al.* [40,41] proposed the Newton FxLMS controller to achieve the same convergence rate at different engine speeds, while the FxLMS controller has different convergence rates at different engine speeds. Here, online adapted look-up tables are integrated as parameter maps. Experiments showed that sufficiently filled look-up tables can be obtained within a short driving time. Hence, no prior measurements are required to obtain suitable map data. In addition, an improvement in tracking behavior is observed with rapid changes in engine speed. Guo *et al.* [42] proposed an extended FxLMS controller to add incorporating body input point inertance (IPI) to the active control engine mount model when accelerometers are used in most practical applications. It is demonstrated through simulations that the errors in estimating the secondary path caused by neglecting the inertia of the body input points lead to a severe breakdown of control. It is validated that the conventional FxLMS algorithm no longer works once the phase of the acceleration frequency response exceeds 90 degrees. The simulation results also show that the acceleration on the

active engine mount has decreased by 80% for most of the time domain after applying the extended FxLMS controller. Bai *et al.* [43] proposed an adaptive generalized predictive controller based on a recursive least squares filter. Experiments indicated that the proposed method is able to reject the vibrations of not only fixed-speed rotors but also varying speed rotors in the presence of disturbances and noise of much smaller amplitude. Stoten *et al.* [44] proposed an error-based MCS controller using integral action. The test results showed that the proposed method performs similarly to the FxLMS algorithm in terms of the vibration cancellation level and convergence speed with lower processing overhead. But the magnitude of the integral gain sometimes increases with time in the presence of noise and disturbances and does not settle to a final value. This effect is called gain windup and in practice can cause the controller to become unstable. Hillis *et al.* [45] proposed a narrow band MCS controller to overcome the gain windup problem. Here, the frequency of the disturbance must be known or measured. When the frequency of the disturbance is known or measured, the simulations and vehicle tests show that the proposed method outperforms the broadband Error-based minimal control synthesis with integral action (Er-MCSI) algorithm in narrow-band applications and overcomes the gain windup problem in the Er-MCSI. As a result, the adaptive control approaches have the best performance in vibration cancellation and the best response time. But they are prone to disturbances and noise and may become unstable under these conditions.

The second group is based on classical control approaches such as PID and the linear quadratic regulator (LQR). Darsivan *et al.* [46] proposed a simple PID-based active vibration controller and performed Matlab simulations of the proposed method. The results indicate that a response time of 250 ms is achieved, but this is a slow response for the active vibration cancellation system for the car engine. Mahil *et al.* [47] proposed two different methods based on PID and LQR, and performed Matlab simulations of the proposed method. The results indicate that the response time is still problematic. However, when we compare these two methods, it is clear that the PID controller is much more effective than the LQR controller. But the PID controller is best suited when only a single process variable is to be controlled. The LQR controller is a better option than the PID controller when multiple outputs are to be considered. Guo *et al.* [48] proposed a PID controller based on particle swarm optimization for active control of the engine mount. Here, particle swarm optimization is used to obtain

more precise PID parameters than conventional methods. The simulation results show that the engine vibrations are canceled, although the response time is again problematic. As a result, classical control methods have longer response times than desired. But they do not need secondary path estimations as adaptive methods do. They are also more robust to noise and disturbances.

The third group is based on robust control approaches such as H_2 and H_∞ methods. Accurate models of the active engine mount system are difficult in model-based control, and controllers may not achieve the desired performance due to model uncertainties, disturbances, and noise. Robust controllers have robust performance in the presence of these perturbations. Olsson *et al.* [49] proposed a robust controller based on the H_2 method to suppress engine vibrations over a wide frequency range. Gain scheduling is used to deal with the nonlinearities of the system, such as nonlinear material properties of the engine mounts and large angular displacements of the engine. The results show that the proposed method handles system nonlinearities and engine vibrations well, but it does not work well at extremely high ramping speeds and nominal engine torque. Fakhari *et al.* [50] proposed a robust controller based on the H_∞ method and compared it with a robust controller based on the H_2 method. The simulations indicated that the proposed method provides better vibration cancellation performance, especially at low frequencies. For example, at 50 Hz, engine vibration decreased by 70% using the proposed method while only decreased by 35% with the H_2 method. The proposed method also requires a lower control effect and power to achieve the desired control performance. Then Fakhari *et al.* [2] improved the robust controller by implementing the model reference adaptive control (MRAC) and compared it with the classical H_∞ method and the classical MRAC control. The simulation results showed that the proposed method has better performance in canceling vibration after 40 Hz. Moreover, it requires less control effect and power than the H_∞ method to achieve the desired control performances. Finally, the proposed method shows robust performance in the presence of disturbances, while the MRAC method diverges and becomes unstable in the presence of uncertainties greater than 30%. Naseri *et al.* [51] proposed a robust controller based on the H_∞ method and a disturbance observer. The disturbance observer is designed to estimate unmeasured transmitted disturbances based on sensor measurements. These estimated disturbances are then used to minimize the disturbance effect. Simulation results have shown that the proposed method provides better vibration

cancellation performance than the classical H_∞ method and the robust MRAC method at high uncertainty levels and in return higher power consumption. As a result, robust control methods provide better performance in vibration cancellation performance than adaptive methods in noise, disturbances, and model uncertainties.

1.3. NOVELTY OF THE STUDY

As we mentioned in the previous section, the works in the literature dealing with active vibration cancellation of car engines using active mounts can be divided into three different groups: adaptive, classical and robust control based methods. In this study, we propose a novel method that, unlike other methods, uses a parameter estimation approach to solve the stated problem. In this approach, the frequency, phase and amplitude parameters of the vibration signal are estimated and the cancellation signal is generated according to these parameters.

The proposed method is based on Kay's classical maximum likelihood estimation (MLE) approach, which gives very good results in the presence of high noise. But the computational cost is too high to implement on a simple microcontroller. Li *et al.* [52] and Kulon *et al.* [53] further improved this approach by using fast Fourier transform (FFT) to reduce the execution time. They also used different windowing methods to reduce spectral leakage. We have also used this approach for the proposed method. However, we use an additional zero padding operation to increase the number of FFT bins and get a better FFT result, while using fewer input samples to further reduce the execution time.

Singh *et al.* [54] further improved this approach by using a search method based on the secant method after the FFT operation. We propose three different search methods: bisection, golden mean and binary search method. They are compared based on accuracy and execution time. The binary search method provides the best results among these methods.

Then we further improve this approach by using predefined look-up tables and iterative calculations. Here, predefined lookup tables are used during the search phase instead of calculating the frequency estimator at each step to reduce the execution time. Then, an additional two-stage search method is developed to further reduce the execution time

while reducing the memory requirement. Iterative calculations are also used to compute trigonometric elements of the H matrix to reduce calculation time. We will present the detailed results of these improvement steps in Section 4.2 using embedded implementation.

Finally, we use three different methods to improve the embedded implementation of the proposed method. First, the partial input sampling is used instead of obtaining all input samples to reduce the execution waiting time. Second, an effective phase delay calculation approach is used to add the running time of the phase calculation algorithm and specify the exact phase shift with respect to the number of elements. Finally, the frequency and phase/amplitude estimation parts are separated and different sampling rates are used for each part to increase the accuracy of the phase estimation. As a result, the proposed method has been improved to work on simple microcontrollers under high noise conditions. We will present the effects of these improvements on the embedded implementation in detail in Section 4.2.

Kay's original method of parameter estimation and these developmental steps are explained in detail in Chapter 2. As an additional note, we would like to point out that Kay's method was developed for estimating sinusoidal parameters under noise and was not used as a method for active vibration cancellation in mechanical active mount systems. In this study, we improved Kay's method to implement it on a simple microcontroller and used the developed method for vibration cancellation on a mechanical active mount system. The proposed method was not only tested with simulations but also on a real mechanical system whose parameters were determined using system identification.

The comparison of vibration suppression of the proposed method and selected active vibration cancellation methods are shown in Table 1.1. This table is obtained from the study of Ranglin [55]. Here, all methods are simulated for a modeled engine mount without disturbance and noise. The 30 Hz sinusoidal input corresponds to the idle speed and the 100 Hz sinusoidal input corresponds to the normal speed of the vehicle. It can be seen that the proposed method gives better results than other methods except for the FxLMS method.

But under disturbances and noise, the performance of adaptive methods decreases and they may even become unstable as stated by Fakhari [2]. This instability is shown in Figure 1.5 for 30% uncertainties as the sum of modeling errors, noise, and disturbances.

Table 1.1. The performance comparison of active vibration cancellation methods for active mounts.

Method	% vibration suppression at 30 Hz	% vibration suppression at 100 Hz
LMS	95.0	98.2
RMS	92.0	97.6
FxLMS	97.0	99.7
LQR	70.0	97.0
H_2	88.0	97.5
H_∞	90.0	99.0
PID	83.0	98.8
Proposed	95.8	99.1

The proposed method and FxLMS method are further compared in Table 1.2 for higher uncertainties. Computational complexity is also compared for both methods. Here, L is the FIR filter length, M is the FFT length, S is the search interval size and N is the number of input samples. It can be seen that the proposed method has higher computational complexity to achieve better results under high certainties.

Table 1.2. The performance comparison of the proposed method and FxLMS method for high uncertainties.

Method	30 Hz	100 Hz	Computational complexity
FxLMS	78.9	81.5	3L+1 Multiplication / 3L-2 Addition
Proposed	81.6	84.3	$(M/2 \cdot \log M) + ((2 \cdot \log S + 1) \cdot (8N + 6))$ Multiplication $(M \cdot \log M) + ((2 \cdot \log S + 1) \cdot (8N))$ Addition

As a result, the proposed method gives better results than adaptive methods under conditions of high noise and disturbances. We will demonstrate this with simulations and experiments in the following chapters.

1.4. COMPETITIVE METHODS IN LITERATURE

In the literature, there are various methods for active vibration cancellation as mentioned before. We choose the phase-locked loop (PLL), the LMS filter and the RLS filter to compare

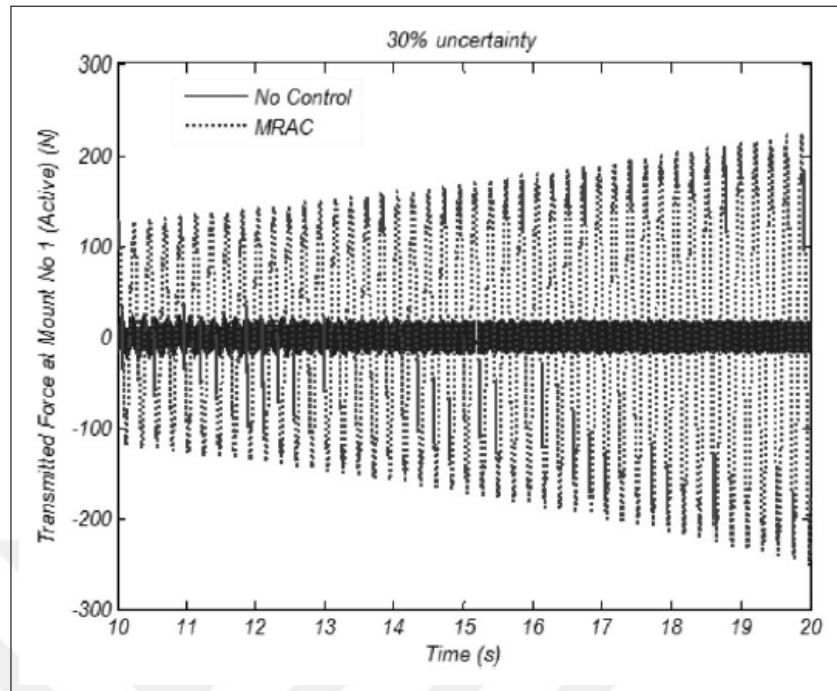


Figure 1.5. The instability of the adaptive control method for 30% uncertainties [2].

the proposed method with.

1.4.1. Phase-Locked Loop

PLL is used for parameter estimation of noisy sinusoidal signals [56,57]. It is also used for sinusoidal signal cancellation [58]. And finally, it is used for active noise cancellation (ANC) and active vibration cancellation systems [59,60].

A PLL is a closed-loop control system that generates an output signal depending on the frequency and phase of an input signal. The block diagram of a PLL system is given in Figure 1.6. A PLL circuit responds to both the frequency and phase of the input signal and automatically changes the frequency of a controlled oscillator until the output signal matches the reference in both frequency and phase [61].

The phase detector block is given in Figure 1.7. If the reference signal leads the generated signal, it gives a 1 to the output. If the generated signal leads the reference signal, it gives a -1 to the output. If they are synchronized, it gives a 0 to the output. These values are multiplied by the predefined gain K_p before the loop filter.

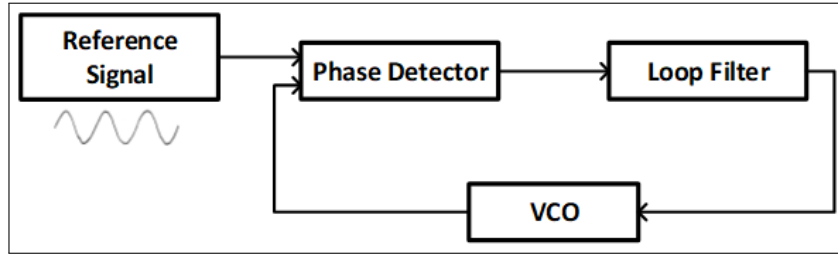


Figure 1.6. The block diagram of the PLL.

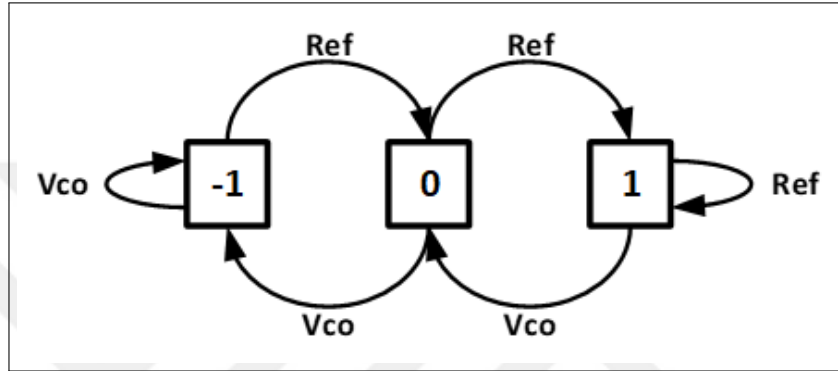


Figure 1.7. The PLL phase detector block.

The loop filter block is given in Figure 1.8. It can be seen that it is a simple 2nd order loop filter. This configuration is highly preferred because of its simplicity. This block filters the higher harmonics and noise at the output of the phase detector. The cutoff frequency of this filter decides the bandwidth and settling time of the system. The cutoff frequency can be lowered to reduce the noise, but this increases the settling time. So it must be chosen carefully.

The voltage controlled oscillator (VCO) block is actually an integrator. It can be implemented in code as a numerically controlled oscillator (NCO). It uses the equation given below to generate a cosine signal corresponding to the filter output V_{filter}

$$y[n] = \cos((2\pi n f_c / f_s) + (2\pi K_v VCO_{int})) \quad (1.1)$$

where $VCO_{int} = VCO_{int} + V_{filter} / f_s$.

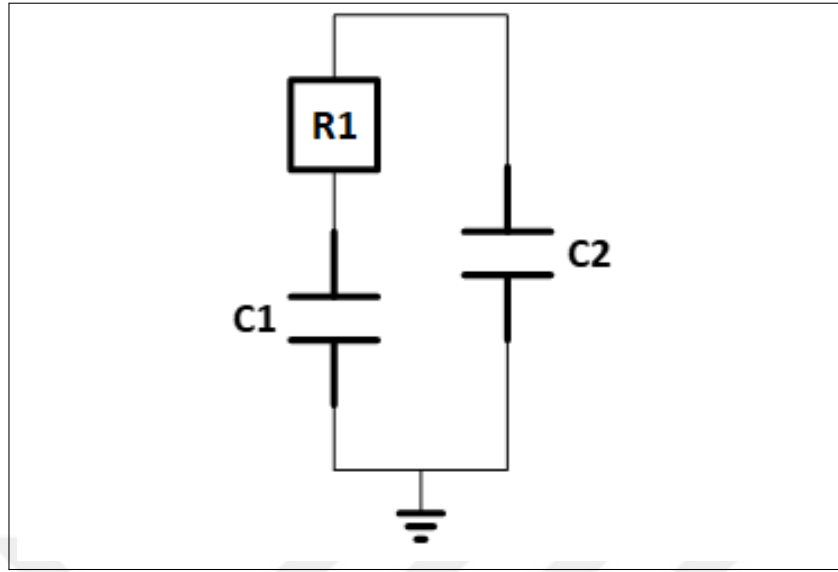


Figure 1.8. The PLL loop filter block.

1.4.2. Least Mean Squares Filter

Mainly because of their simplicity, LMS filters are highly preferred in AVC systems [62,63,64,65,66]. LMS filter is a special class of adaptive filters that update its coefficients depending on the error between the desired and the actual signal. The LMS method aims to approximate its cost function by the instantaneous value of the error square. This is a popular method for parameter adjustment in adaptive filtering. The reason for the popularity of this method is its simplicity. Therefore, it is easy to implement. The block diagram of the LMS method is given in Figure 1.9.

The LMS method can be explained as follows. First, the relationship between the input and output of an FIR filter can be described as

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] \quad (1.2)$$

The error signal is the difference between the desired output signal ($d[n]$) and the output signal and it is calculated as

$$e[n] = d[n] - y[n] \quad (1.3)$$

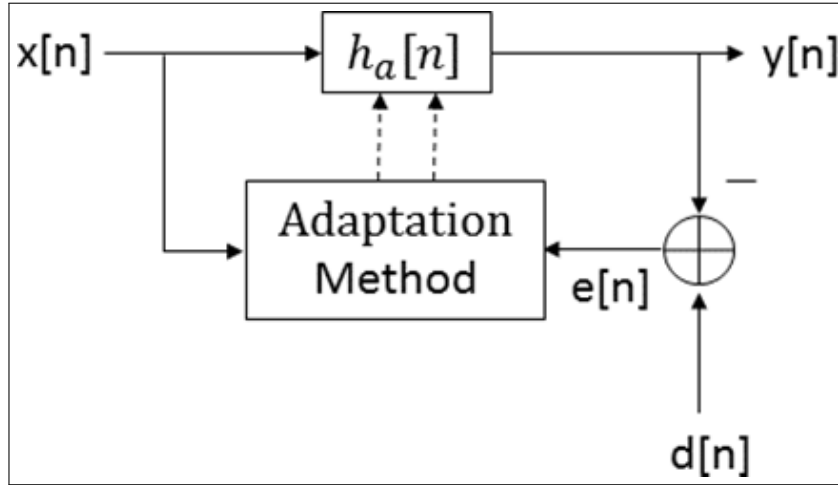


Figure 1.9. The block diagram of the LMS method.

As a result, the update rule for the filter coefficients ($h[n]$) of the LMS filter is given as

$$h^{new}[n] = h^{old}[n] + \mu e[n]x[n - k] \quad (1.4)$$

Here, a positive step size parameter, μ , is used to set the convergence rate. There is an inverse relationship between μ and the convergence time. But increasing μ can lead to an unstable system and it must be set between the ranges given below to ensure stability. Here, L is the number of coefficients of the LMS filter and P_x is the power of the input signal.

$$0 < \mu < \frac{2}{LP_x} \quad (1.5)$$

The LMS filter is used as a predictor for active vibration cancellation. Here, past values of the input signal are used to predict its future values. For the prediction process, a delayed version of the input signal is fed into the LMS system, while the original input signal is used as the desired signal. Finally, the adaptive filter can be used to predict the future values of the input signal after the parameter adjustment phase. The block diagram of the LMS filter when used as a predictor is shown in Figure 1.10.

1.4.3. Recursive Least Squares Filter

RLS filters are alternative adaptive filters used in AVC systems [43]. The RLS algorithm solves the least squares problem recursively. This results in computational savings and faster

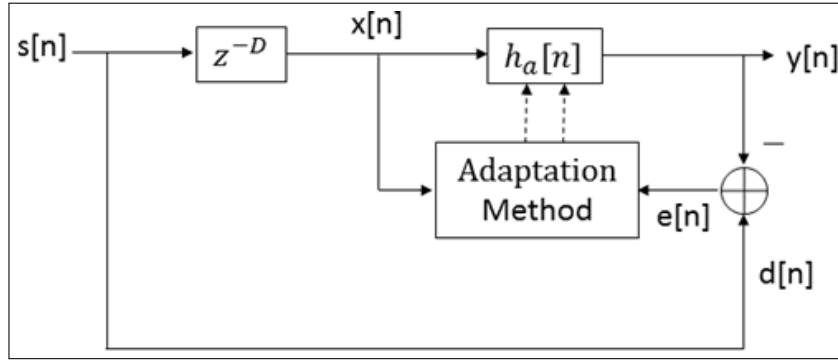


Figure 1.10. The block diagram of the LMS predictor.

convergence is also achieved.

The RLS method can be explained as follows. The relationship between the input and output of an RLS filter can be described as

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] \quad (1.6)$$

The error signal is the difference between the desired output signal ($d[n]$) and the output signal ($y[n]$) and it is calculated as

$$e[n] = d[n] - y[n] \quad (1.7)$$

Then filter gain vector (k) is calculated as below. Here, λ is the weighting factor and P is the inverse correlation matrix.

$$k = (P * x) / (\lambda + x^T * P * x) \quad (1.8)$$

Then the inverse correlation matrix, P , is updated as follows.

$$P = (P - k * x^T * P) / \lambda \quad (1.9)$$

Finally, the update rule for the filter coefficients ($h[n]$) of the RLS filter is given below.

$$h^{new}[n] = h^{old}[n] + k * e[n] \quad (1.10)$$

For the RLS algorithm, the initial values of the filter coefficients must be set to zero. The inverse correlation matrix must also be initialized as $P = \delta * I$. Here δ is chosen small for

systems with high noise and large for systems with low noise

1.5. LAYOUT OF THE THESIS

This work consists of eight chapters. The first chapter gives an overview of the study and the general concepts considered in it. It also contains a review of the literature. In the second chapter, the theoretical background of the proposed method is explained in detail. The third chapter presents the embedded hardware and software systems for the proposed method. The fourth chapter explains the details of the embedded implementation of the proposed method. The fifth chapter presents the loopback simulations of the proposed method and the competitive methods. They are also compared in terms of accuracy, suppression level, response time, and memory consumption. In the sixth chapter, electrical experiments are conducted with the proposed and competitive methods. In the seventh chapter, the mechanical setup used for the experiments is explained in detail. Then the results of the mechanical experiments are presented. Finally, we conclude our study in the eighth chapter.

2. THE PROPOSED ACTIVE VIBRATION CANCELLATION METHOD

Therefore, we will start by describing Kay's parameter estimation algorithm. Then, each development step of the proposed method will be explained. Therefore, we start with the description of Kay's parameter estimation algorithm. Then, the individual development steps of the proposed method are explained.

2.1. KAY'S ALGORITHM FOR SINUSOIDAL PARAMETER ESTIMATION

Kay's algorithm uses the MLE method for parameter estimation. In this algorithm, a sinusoidal signal under noise is described as

$$x[n] = A \cos(2\pi f_0 n + \phi) + \omega[n] \quad (2.1)$$

Here A , ϕ and f_0 stand for amplitude, phase and normalized frequency of the sinusoidal signal, respectively. $\omega[n]$ stands for white Gaussian noise (WGN) which affects the sinusoidal signal. According to the Nyquist sampling theorem, the sampling frequency must be selected at least twice the highest frequency component of the signal. Accordingly, the frequency range of a digital signal is described as $0 < f < 0.5f_s$ where f is the signal frequency and f_s is the sampling frequency. If this range is divided by f_s , the range of f_0 can be found as $0 < f_0 < 0.5$. The intervals for A and ϕ parameters are defined as $A > 0$ and $-\pi < \phi < \pi$, respectively.

To estimate the parameters A , ϕ and f_0 , the corresponding MLE estimators must first be derived. These estimators can be obtained using the probability density function (PDF), which is given as

$$p(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left[-\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} (x[n] - A \cos(2\pi f_0 n + \phi))^2\right] \quad (2.2)$$

where σ is the variance of the WGN signal and N is the number of signal samples. The MLE of the parameters A , ϕ and f_0 can be obtained by minimizing the exponential part of this

equation as

$$J(A, f_0, \phi) = \sum_{n=0}^{N-1} (x[n] - A \cos(2\pi f_0 n + \phi))^2 \quad (2.3)$$

To find MLE estimators for these parameters, the cosine is expanded using the trigonometric angle addition formula $\cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$. The resulting equation is

$$J(A, f_0, \phi) = \sum_{n=0}^{N-1} (x[n] - A \cos(\phi) \cos(2\pi f_0 n) + A \sin(\phi) \sin(2\pi f_0 n))^2 \quad (2.4)$$

Although J is a non-quadratic function for the parameters A and ϕ , it can be transformed into a quadratic function with $\alpha_1 = A \cos(\phi)$ and $\alpha_2 = -A \sin(\phi)$. The parameters A and ϕ can also be calculated using the inverse transformation of these equations. The resulting equations are

$$A = \sqrt{\alpha_1^2 + \alpha_2^2} \quad (2.5)$$

$$\phi = \arctan\left(\frac{-\alpha_2}{\alpha_1}\right) \quad (2.6)$$

Then Equation 2.4 can be rewritten using α_1 and α_2 parameters as

$$J'(\alpha_1, \alpha_2, f_0) = (\mathbf{x} - \mathbf{H}\boldsymbol{\alpha})^T (\mathbf{x} - \mathbf{H}\boldsymbol{\alpha}) \quad (2.7)$$

where $\boldsymbol{\alpha} = [\alpha_1 \ \alpha_2]^T$ and $\mathbf{H}$ matrix is defined as

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \\ \cos(2\pi f_0) & \sin(2\pi f_0) \\ \vdots & \vdots \\ \cos(2\pi f_0(N-1)) & \sin(2\pi f_0(N-1)) \end{bmatrix} \quad (2.8)$$

The MLE of $\boldsymbol{\alpha}$ can be found by minimizing Equation 2.7. Since this is a quadratic function of the elements of $\boldsymbol{\alpha}$, the global minimum point can be calculated by the taking derivative and equating it to 0. The result is described as

$$\hat{\boldsymbol{\alpha}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x} \quad (2.9)$$

Then Equation 2.7 can be rewritten using Equation 2.9 as

$$\begin{aligned}
 J'(\hat{\alpha}_1, \hat{\alpha}_2, f_0) &= (\mathbf{x} - \mathbf{H}\hat{\alpha})^T (\mathbf{x} - \mathbf{H}\hat{\alpha}) \\
 &= (\mathbf{x} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x})^T (\mathbf{x} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}) \\
 &= \mathbf{x}^T (\mathbf{I} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T) \mathbf{x}
 \end{aligned} \tag{2.10}$$

To find the estimated normalized frequency $\hat{f}_0$, Equation 2.10 must be minimized over f_0 or, equivalently, the second half of Equation 2.10 must be maximized over f_0 . So the MLE estimator for finding $\hat{f}_0$ can be found as

$$J'(\hat{f}_0) = \mathbf{x}^T (\mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T) \mathbf{x} \tag{2.11}$$

Here, the MLE estimator is calculated for different f_0 values and the f_0 value maximizes the estimator is accepted as the $\hat{f}_0$ of the input signal. After $\hat{f}_0$ is obtained, the parameters $\hat{\alpha}_1$ and $\hat{\alpha}_2$ are calculated using $\hat{f}_0$ in Equation 2.9. Finally, the estimated amplitude $\hat{A}$ and the estimated phase $\hat{\phi}$ can be calculated using Equation 2.5 and Equation 2.6. It can be seen that the estimation performance of $\hat{f}_0$ directly affects the estimation performance of the parameters $\hat{A}$ and $\hat{\phi}$. Therefore, the accuracy of frequency estimation is very important. It is difficult to achieve good phase and amplitude estimation with poor frequency estimation.

The PDFs of the estimated parameters are used to observe the performance analysis of the estimations. These PDFs for $\hat{f}_0$, $\hat{\phi}$, and $\hat{A}$ are given as [67].

$$N(\hat{f}_0, \frac{12}{(2\pi)^2 \eta N(N^2 - 1)}) \tag{2.12}$$

$$N(\hat{\phi}, \frac{2(2N - 1)}{\eta N(N + 1)}) \tag{2.13}$$

$$N(\hat{A}, \frac{2\sigma^2}{N}) \tag{2.14}$$

where η is the signal-to-noise ratio (SNR) and described as $\eta = \frac{A^2}{2\sigma^2}$. Examination of these PDFs shows that the parameters σ must be decreased or N must be increased to improve the performance of the estimations.

2.2. THE IMPROVEMENTS IN KAY'S ALGORITHM

An example graph of the MLE estimator used to determine $\hat{f}_0$ can be found in Figure 2.1. It can be seen that this estimator only gets high values around the estimated normalized frequency value. Therefore, the speed of the algorithm can be increased if it is concentrated around this point.

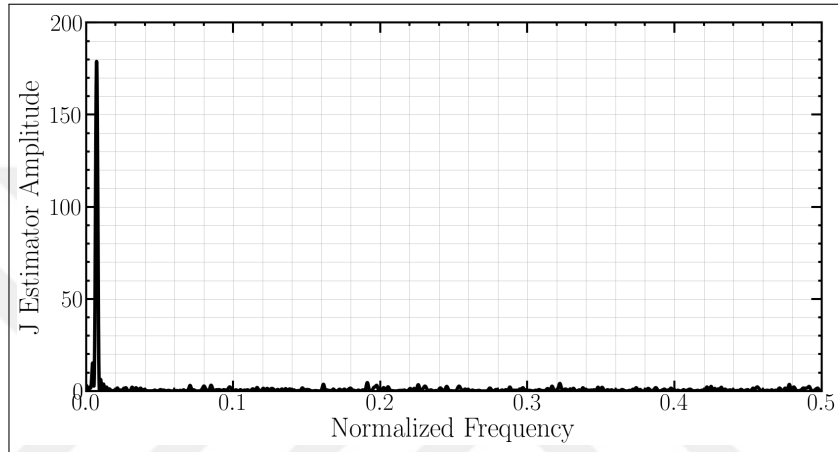


Figure 2.1. Example graph of MLE estimator used to find $\hat{f}_0$.

Also when the algorithm is analyzed in parts, it can be seen that more than 90% of the total calculation time is spent on computing the H arrays. As a result, speeding up this part will improve the overall time of the algorithm. Based on these observations, three methods are proposed to improve Kay's algorithm.

2.2.1. Applying Fast Fourier Transform

As we have already explained, the J estimator gets high values only around the estimated normalized frequency. So the estimation time can be drastically improved by finding a crude normalized frequency near this peak point and applying the J-estimator around this crude estimation point. The best method for this is to convert the input signal to the frequency domain by Fourier analysis using the discrete Fourier transform (DFT). However, the time complexity of DFT is $O(N^2)$, where N is the size of the input signal. Therefore, the fast Fourier transform (FFT) is preferred instead, whose time complexity is $N * \log(N)$.

In the proposed algorithm, the Fast Fourier Transform (FFT) is applied to the input signal to first obtain a crude frequency value. Then Kay's algorithm is applied around a certain interval. To reduce the calculation time, the number of elements of the input signal must be reduced. However, this reduces the number of FFT bins and a correct FFT result cannot be obtained. Zero padding is applied to the input signal to increase the number of FFT bins and obtain a better FFT result. As a side note, zero padding does not improve spectral frequency resolution. But it does allow you to get more accurate FFT amplitude estimates.

Another important point when applying the FFT to a signal is spectral leakage. Spectral leakage is the frequency components that are not related to the original signal. They can be created by non-periodic discrete time signals or additional noise components. To reduce the effect of spectral leakage on the accuracy of the FFT, windowing operation is applied to the input signal. There are various window functions such as the rectangular, Blackman-Harris, and Hanning windows. The rectangular window is used for low frequency dynamic range applications and the Blackman-Harris window is used for high frequency dynamic range applications. The Hanning window lies between these two extremes and is used for applications with medium frequency dynamic range. Therefore, it is selected for the proposed algorithm.

When applying the FFT, a Hanning window is first applied to the input signal to reduce spectral leakage. Then zero padding is applied to increase the number of FFT bins, and finally the FFT is applied to this signal. The obtained FFT outputs for these approaches when the signal frequency is selected as 72.6 Hz and the sampling frequency is selected as 1 kHz are shown in Figure 2.2. When the Hamming window and zero padding are not applied, the FFT result is 62.5 Hz. But when they are applied, the FFT result is 70.3125 Hz.

Li *et al.* [52] and Kulon *et al.* [53] suggested that the use of FFT reduces the execution time of the MLE-based parameter estimation method. They also used different windowing methods to reduce spectral leakage.

We further improved this approach by using an additional zero-padding operation to increase the number of FFT bins and to get a better FFT outcome while using lesser input samples to decrease execution time further. We further improved this approach by using an additional zero-padding operation to increase the number of FFT bins and obtain a better FFT result,

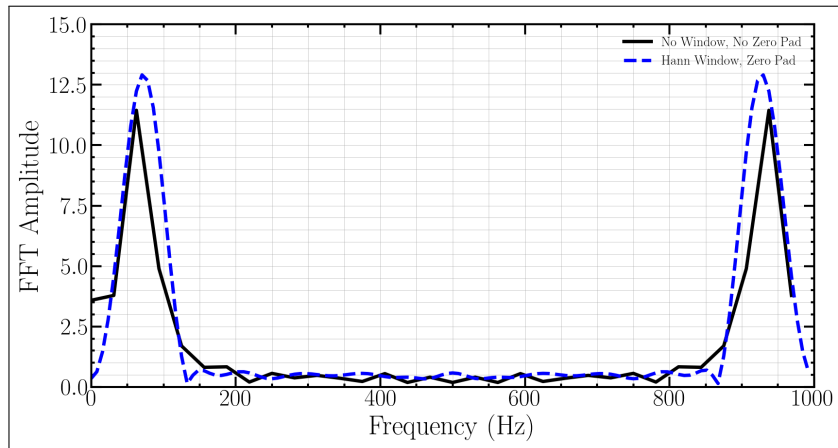


Figure 2.2. FFT results for different approaches.

while using fewer input samples to further reduce the execution time.

2.2.2. Using Search Algorithms

Search algorithms are used to find specific elements of a function or data set. It is predicted that search algorithms can be used for estimating normalized frequency values much faster in the proposed method. In the proposed algorithm, FFT is applied to an input signal for obtaining a crude frequency value at first. Normally, after this step, the Kay algorithm is applied at a certain interval. But instead of this approach, using search algorithms in this interval can give better results.

Singh *et al.* [54] proposed an MLE-based frequency estimation method based on the secant method. In addition, three different search algorithms are implemented: the bisection method, the golden mean search method and the binary search method. There are better search algorithms in the literature, but the main concern here is to speed up the estimation process and these algorithms are simpler than others. Moreover, it is predicted that there will be only one maximum point in the chosen interval and the selected algorithms are more than sufficient.

2.2.2.1. Bisection Method

The bisection method is generally used to find the roots of a given function. However, it can also be used to find a maximum point of a function. Suppose $f(x)$ is a function on a

closed, bounded interval $[a1, b1]$. The maximum point of this function can be found using the bisection method with the steps given in Algorithm 2.1.

Algorithm 2.1. Bisection method algorithm.

```

1:  $a1$  is the lower boundary
2:  $b1$  is the upper boundary
3:  $c$  is the middle point
4:  $x$  is the maximum point
5:  $\varepsilon$  is the selected tolerance
6:  $N$  is the iteration number
7:  $N\_max$  is the maximum iteration number
8: Compute  $c = (a1 + b1)/2$ 
9: while  $(b1 - a1 > \varepsilon)$  and  $(N < N\_max)$  and  $(f'(c) \neq 0)$  do
10:   Compute the derivatives of  $f(x)$  as  $f'(a1)$ ,  $f'(b1)$  and  $f'(c)$ 
11:   if  $f'(a1) * f'(c) < 0$  then
12:      $b1 \leftarrow c$ 
13:   end if
14:   if  $f'(b1) * f'(c) < 0$  then
15:      $a1 \leftarrow c$ 
16:   end if
17:   Compute  $c = (a1 + b1)/2$ 
18: end while
19: if  $f'(c) == 0$  then
20:    $x \leftarrow c$ 
21: end if
22: if  $b1 - a1 \leq \varepsilon$  then
23:   if  $|f(a1)| < |f(b1)|$  then
24:      $x \leftarrow b1$ 
25:   end if
26:   if  $|f(b1)| < |f(a1)|$  then
27:      $x \leftarrow a1$ 
28:   end if
29: end if
30: if  $N \geq N\_max$  then
31:    $x$  is incomputable in this interval
32: end if

```

The bisection method uses the derivative of a function to find the maximum point. The function has an ascending or descending trend if its derivative is nonzero at that point. On the other hand, it reaches its maximum when the derivative at that point is zero. The maximum point of a function can therefore be found by calculating the derivative at the lower, upper and middle points. If the derivative at the middle point is zero, the maximum point is found. If

the derivative at the middle point is non-zero, the sign of the derivative is compared with the results of the upper and lower point. Then the upper and lower limits are updated accordingly and the process is repeated from the beginning.

2.2.2.2. Golden-Mean Search Method

Suppose $f(x)$ is a function on a closed bounded interval $[a1, b1]$. The maximum point of this function can be derived using the golden-mean search method with steps given in the Algorithm 2.2.

Algorithm 2.2. Golden-mean search method algorithm.

```

1:  $a1$  is the lower boundary
2:  $b1$  is the upper boundary
3:  $a2$  is the new lower boundary
4:  $b2$  is the new upper boundary
5:  $r$  is the golden ratio
6:  $x$  is the maximum point
7:  $\varepsilon$  is the selected tolerance
8:  $N$  is the iteration number
9:  $N\_max$  is the maximum iteration number
10: Compute  $a2 = b1 - r(b1 - a1)$  and  $b2 = a1 + r(b1 - a1)$ 
11: while  $(b1 - a1 > \varepsilon)$  and  $(N < N\_max)$  and  $(f(a2) \neq f(b2))$  do
12:   if  $f(a2) < f(b2)$  then
13:      $a1 \leftarrow a2$ 
14:   end if
15:   if  $f(b2) < f(a2)$  then
16:      $b1 \leftarrow b2$ 
17:   end if
18:   Compute  $a2 = b1 - r(b1 - a1)$  and  $b2 = a1 + r(b1 - a1)$ 
19: end while
20: if  $b1 - a1 \leq \varepsilon$  then
21:   if  $|f(a1)| < |f(b1)|$  then
22:      $x \leftarrow b1$ 
23:   end if
24:   if  $|f(b1)| < |f(a1)|$  then
25:      $x \leftarrow a1$ 
26:   end if
27: end if
28: if  $(N \geq N\_max)$  or  $(f(a2) == f(b2))$  then
29:    $x$  is incomputable in this interval
30: end if

```

The golden-mean search method uses a golden ratio to recalculate the upper and lower

boundaries until the difference between them is less than a selected tolerance value. When this condition is met, the maximum point can be selected by comparing the function result at these two boundary points.

2.2.2.3. Binary Search Method

Binary search is an efficient algorithm for finding an element from a sorted list of elements. It involves repeatedly splitting the part of the list that might contain the item in half until you have narrowed down the possible items to one. However, this procedure can also be used to find the maximum point of a function if it has only one peak point.

Suppose $f(x)$ is a function on a closed bounded interval $[a_1, b_1]$. The maximum point of this function can be derived using the binary search method with steps given in Algorithm 2.3.

The binary search method calculates the function at the middle point c of the upper and lower boundaries. It also calculates the function at points $c - 1$ and $c + 1$. If the function at point c has a higher value, it means that the maximum point has been found. If the value of a function at points $c - 1$ or $c + 1$ is higher, the upper and lower limits are recalculated accordingly and the process is repeated from the beginning.

2.2.2.4. Secant Method

The secant method is generally used to find the roots of a given function. However, it can also be used to find the maximum point of a function over an interval for which the defined function has only one maximum point. Suppose $f(x)$ is a function on a closed, bounded interval $[a_1, b_1]$. The maximum point of this function can be found using the secant method with steps given in Algorithm 2.4.

2.2.2.5. Comparison of Search Methods

As a result, we have provided four different search methods: Bisection, golden mean, binary search, and the secant method proposed by Singh [54]. If we examine them in detail, we can see that the bisection method and the secant method contain derivation operations. But the bisection method also uses a basic midpoint approach during the search phase, so the secant method gives a better result. The golden mean search method, on the other hand, does not contain derivation operations, but two additional points and a certain ratio value

Algorithm 2.3. Binary search method algorithm.

```

1:  $a_1$  is the lower boundary
2:  $b_1$  is the upper boundary
3:  $c$  is the middle point
4:  $x$  is the maximum point
5:  $\varepsilon$  is the selected tolerance
6:  $N$  is the iteration number
7:  $N_{max}$  is the maximum iteration number
8: Compute  $c = (a_1 + b_1)/2$ 
9: while ( $b_1 - a_1 > \varepsilon$ ) and ( $N < N_{max}$ ) and ( $(f(c) < f(c-1))$  or  $(f(c) < f(c+1))$ ) do
10:   if  $f(c) < f(c+1)$  then
11:      $a_1 \leftarrow c + 1$ 
12:   end if
13:   if  $f(c) < f(c-1)$  then
14:      $b_1 \leftarrow c - 1$ 
15:   end if
16:   Compute  $c = (a_1 + b_1)/2$ 
17: end while
18: if ( $f(c) \geq f(c-1)$ ) and ( $f(c) \geq f(c+1)$ ) then
19:    $x \leftarrow c$ 
20: end if
21: if  $b_1 - a_1 \leq \varepsilon$  then
22:   if  $|f(a_1)| < |f(b_1)|$  then
23:      $x \leftarrow b_1$ 
24:   end if
25:   if  $|f(b_1)| < |f(a_1)|$  then
26:      $x \leftarrow a_1$ 
27:   end if
28: end if
29: if  $N \geq N_{max}$  then
30:    $x$  is incomputable in this interval
31: end if

```

Algorithm 2.4. Secant method algorithm.

```

1:  $a1$  is the lower boundary
2:  $b1$  is the upper boundary
3:  $c$  is the middle point
4:  $x$  is the maximum point
5:  $\varepsilon$  is selected the tolerance
6:  $N$  is the iteration number
7:  $N\_max$  is the maximum iteration number
8: Compute  $c = (a1 + b1)/2$ 
9: while  $(b1 - a1 > \varepsilon)$  and  $(N < N\_max)$  and  $(f'(c) \neq 0)$  do
10:   Compute the derivatives of  $f(x)$  as  $f'(a1)$ ,  $f'(b1)$  and  $f'(c)$ 
11:   if  $f'(a1) * f'(c) < 0$  then
12:      $b1 \leftarrow c$ 
13:   end if
14:   if  $f'(b1) * f'(c) < 0$  then
15:      $a1 \leftarrow c$ 
16:   end if
17:   Compute  $c = b1 - (f'(b1) * ((b1 - a1)/(f'(b1) - f'(a1))))$ 
18: end while
19: if  $f'(c) == 0$  then
20:    $x \leftarrow c$ 
21: end if
22: if  $b1 - a1 \leq \varepsilon$  then
23:   if  $|f(a1)| < |f(b1)|$  then
24:      $x \leftarrow b1$ 
25:   end if
26:   if  $|f(b1)| < |f(a1)|$  then
27:      $x \leftarrow a1$ 
28:   end if
29: end if
30: if  $N \geq N\_max$  then
31:    $x$  is incomputable in this interval
32: end if

```

are calculated for each iteration. This means an additional processing effort. Finally, the binary search method contains neither a derivation operation nor an additional point or ratio calculation. Therefore, it gives better results than the other methods. The comparison of the implementation of these methods can be found in section 4.2.2.

2.2.3. Predefined Look-up Tables and Iterative Calculations

The computation of H arrays is the most time-consuming part of the proposed method for active vibration cancellation. We propose two approaches to reduce this time as using predefined look-up tables and calculating some sine and cosine values with iterative calculations.

After finding the crude frequency, the desired search algorithm is applied and the H array is calculated for different frequency values in the selected interval. If the minimum frequency step is selected as x , the H array is calculated as $H(f + bx)$. Here f is the crude FFT frequency and b is the frequency point difference between crude FFT frequency and the point selected by the search algorithm. Thus, if all $H(bx)$ values are calculated and a look-up table is created based on them, the iterative H arrays can be easily determined using the trigonometric sum and difference formulas given below.

$$\cos(f \pm bx) = \cos(f) * \cos(bx) \mp \sin(f) * \sin(bx) \quad (2.15)$$

$$\sin(f \pm bx) = \sin(f) * \cos(bx) \pm \cos(f) * \sin(bx) \quad (2.16)$$

With this approach, the computation time is reduced, but the memory requirement increases greatly. To reduce this, a two-stage search method is used. In the first stage, the search algorithm is used to find the frequency with ten times the desired accuracy. For example, if the desired frequency accuracy is 0.1 Hz, the frequency is first estimated with an accuracy of one Hz. Then the search algorithm is applied again around this frequency value to estimate the frequency with the desired accuracy. This requires 80% less RAM space for look-up tables.

The second approach to reducing computation time is to use iterative calculations. It is already known that the H array contains iterative elements such as $\cos(x)$, $\cos(2x)$, $\cos(3x)$,

$\cos(4x)$ and $\sin(x)$, $\sin(2x)$, $\sin(3x)$, $\sin(4x)$, $\dots$ etc. Instead of iteratively calculating these values for cosine and sine, you can also derive them using the values for $\sin(x)$ and $\cos(x)$ by using the formulas given below [68].

$$\cos(nx) = \sum_{k=0}^n \binom{n}{k} \cos^k(x) \sin^{n-k}(x) \cos\left(\frac{1}{2}(n-k)\pi\right) \quad (2.17)$$

$$\sin(nx) = \sum_{k=0}^n \binom{n}{k} \cos^k(x) \sin^{n-k}(x) \sin\left(\frac{1}{2}(n-k)\pi\right) \quad (2.18)$$

But after a certain step, the iterative calculations take more time than the calculation of the original trigonometric functions. As a result of experiments, this value is found as the ninth step. That is, after calculating $\sin(9x)$ and $\cos(9x)$, this method is no longer suitable. But for an input signal with 32 elements, this approach reduces the calculation time of the H-matrix by 20%.

Consequently, we have further improved Kay's parameter estimation method than the method provided by Singh *et al.* [54]. Here, we found that using predefined lookup tables during the search phase reduces the execution time of the proposed method instead of calculating the frequency estimator at each step. Then, we additionally developed a two-step search method to further reduce the execution time while reducing the memory requirement. Finally, we used iterative calculations to compute the trigonometric elements of the H-matrix to reduce the computation time of the H-matrix by another 20%.

3. EMBEDDED SYSTEMS USED IN IMPLEMENTATION

In this study, the proposed active vibration cancellation method is executed on a microcontroller selected as an embedded hardware platform. Hence, the performance of the proposed method is restricted by the hardware limitations of the selected microcontroller. Thus, the selected microcontroller must have sufficient memory and clock frequency to meet the real-time requirements of the proposed method. In this chapter, the selected hardware and software platforms are explained in detail.

3.1. EMBEDDED HARDWARE PLATFORMS

The Arm Cortex-M series of microcontrollers dominate today's market, offering different solutions for different applications. For example, the Cortex-M0 and Cortex-M0+ based cores implement the ARMv6-M architecture. They have less memory and a lower clock speed, but are designed for low power consumption. Especially, the Cortex-M0+ based core is an optimized version of the Cortex-M0 based core and its pipeline stage is optimized for lower power consumption. The Cortex-M3 based core implements the ARMv7-M architecture and has higher performance than the Cortex-M0 based core. The Cortex-M4 based core implements the ARMv7E-M architecture. It is basically the Cortex-M3 based core with additional digital signal processing (DSP) instructions so that it offers higher performance, especially for DSP applications. Finally, the Cortex-M7 based core implements the ARMv7E-M architecture and has higher performance than the Cortex-M4 based core. The Cortex-M4 and Cortex-M7 based cores with additional floating point unit (FPU) are called Cortex-M4F and Cortex-M7F based cores and they are particularly suitable for applications with extensive floating point calculations. The clock speeds and memory spaces for various microcontrollers belonging to these cores are tabulated in Table 3.1. On the other hand, the current consumption, power consumption, and prices for the same microcontrollers are tabulated in Table 3.2.

The proposed active vibration cancellation method contains intensive floating point operations so Arm Cortex-M0, Cortex-M0+, and Cortex-M3 based microcontrollers are not preferred due to the given design constraints. On the other hand, Arm Cortex-M7F based microcontrollers

Table 3.1. Clock speed and memory space comparison of different Arm Cortex-M microcontrollers.

Microcontroller	Arm Core	Clock Speed (MHz)	Flash (kB)	RAM (kB)
STM32F030R8	Cortex-M0	48	64	8
STM32F207ZG	Cortex-M3	120	1024	128
STM32F429ZI	Cortex-M4	180	2048	256
STM32F746NG	Cortex-M7	216	1024	336
STM32H743ZI	Cortex-M7	480	2048	1056

Table 3.2. Current/power consumption and price comparison of different Arm Cortex-M microcontrollers.

Microcontroller	Arm Core	Cur. Con. (mA)	Pow. Con. (mW)	Price (\$)
STM32F030R8	Cortex-M0	8	26	2.94
STM32F207ZG	Cortex-M3	28	92	14.62
STM32F429ZI	Cortex-M4	44	145	16.35
STM32F746NG	Cortex-M7	178	587	20.63
STM32H743ZI	Cortex-M7	226	754	26.28

have higher performance but they are expensive and overkill for this application. Therefore, Arm Cortex-M4F based microcontrollers are selected in this study to perform the proposed method at a reasonable price and performance. The STM32F429I-DISC1 board given in Figure 3.1 is selected for this application.

The STM32F429I-DISC1 board contains an Arm Cortex-M4F based STM32F429ZI microcontroller. This microcontroller has a clock frequency of 180 MHz, 2 MB flash memory, 256 kB RAM, special low-power modes, three 12-bit analog to digital converter (ADC) modules, two 12-bit digital to analog converter (DAC) modules, general purpose direct memory access (DMA) module, and a variety of communication modules such as four universal synchronous/asynchronous receiver/transmitter (USART), six serial peripheral interface (SPI) and two controller area network (CAN) modules. It has 17 different timer modules to coordinate time-based operations and generate pulse-width modulation (PWM) signals. Finally, it has a true random number generator that is used to generate the WGN signal.

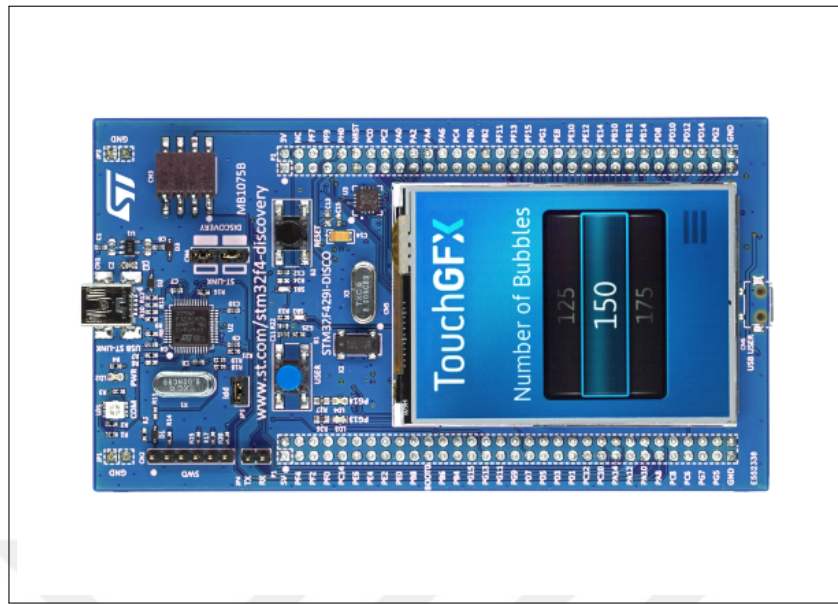


Figure 3.1. STM32F429I-DISC1 board [3].

We will also use two additional Arm Cortex-M7F based development boards for the performance comparison. The first is the 32F746GDISCOVERY board given in Figure 3.2.

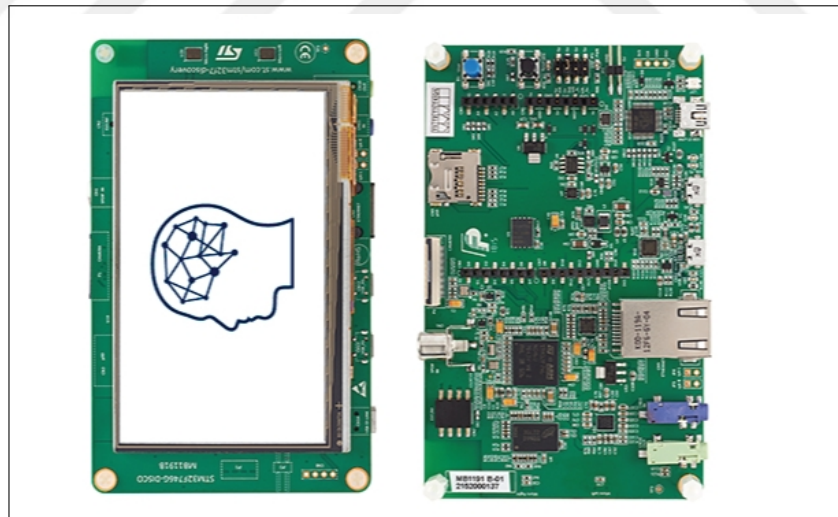


Figure 3.2. 32F746GDISCOVERY board [4].

The 32F746GDISCOVERY board contains an Arm Cortex-M7F based STM32F746NG microcontroller. This microcontroller has a clock frequency of 216 MHz clock, 1 MB flash memory, 336 kB RAM, special low-power modes, three 12-bit ADC modules, two 12-bit DAC modules, a DMA module, 18 timer modules, true random number generator module and a variety of communication modules such as four USART, six SPI and two CAN

modules.

We will use the Arm Cortex-M7F based NUCLEO-H743ZI2 development board given in Figure 3.3 for the performance comparison.

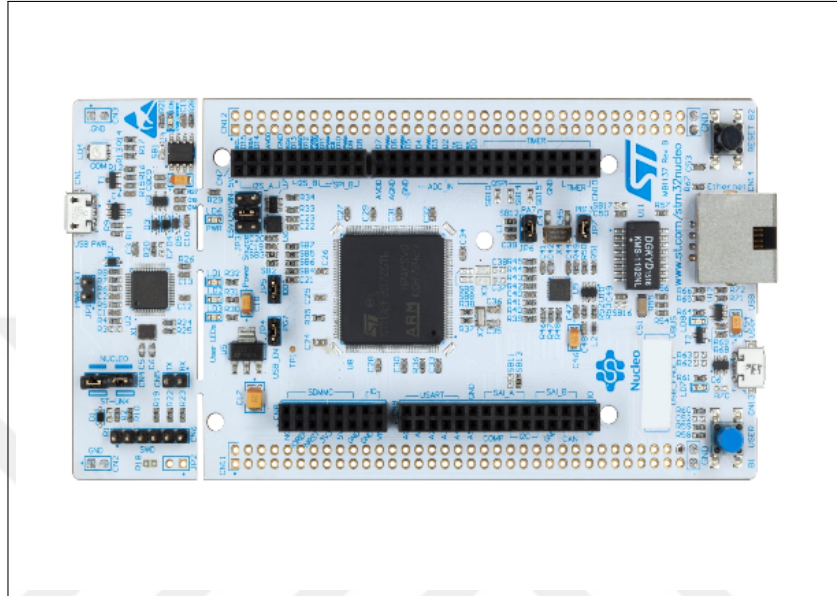


Figure 3.3. NUCLEO-H743ZI2 board [5].

The NUCLEO-H743ZI2 board contains an Arm Cortex-M7F based STM32F746NG microcontroller. This microcontroller has a clock frequency of 480 MHz, 2 MB flash memory, 1056 kB RAM, special low-power modes, three 16-bit ADC modules, two 12-bit DAC modules, DMA modules, 22 timer modules, true random number generator module and a variety of communication modules such as four USART, six SPI and two CAN modules.

3.2. EMBEDDED SOFTWARE PLATFORMS

To program selected microcontrollers, we need an integrated development environment (IDE) on a PC. In this section, we will introduce the selected development environment and the associated support files in detail.

3.2.1. Development Environment

There are several IDEs on the market to program microcontrollers using different programming languages. We will use the STM32CubeIDE platform shown in Figure 3.4 to program selected microcontrollers using C language.

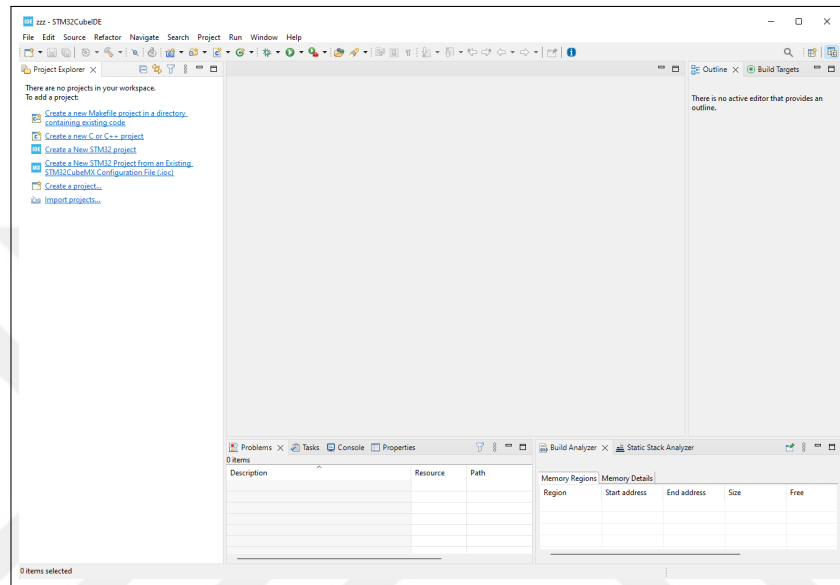


Figure 3.4. STM32CubeIDE home screen.

STMicroelectronics has developed the STM32CubeIDE specifically for use with their microcontrollers. It also has a special plugin called STM32CubeMX to visually modify and initialize hardware blocks. Finally, it can also be used with support packages and libraries, which are described in the following sections.

3.2.2. Embedded Software Packages

STMicroelectronics provides libraries for the hardware abstraction layer (HAL) to efficiently control the hardware peripherals of the STM32 microcontroller. These peripherals can be accessed, initialized, and operated using HAL libraries. HAL libraries are used in the proposed active vibration cancellation method to control peripherals such as UART, ADC, DMA, DAC, etc.

The programs created using HAL libraries are not specific to a single device and can be used

in other microcontrollers with minor modifications. Therefore, we will use HAL libraries in hardware control to use the same code in all selected microcontrollers.

3.2.3. CMSIS Library

FFT operations, matrix operations, and trigonometric calculations are extensively used in the proposed active vibration cancellation method. These operations are the most time-consuming parts of the algorithm and should be optimized. Arm provides a special, vendor-independent abstraction layer for this purpose, called the Common Microcontroller Software Interface Standard (CMSIS). CMSIS has a DSP library with optimized FFT, matrix, and trigonometric operations for floating point calculations.

Unlike the trigonometric functions defined for the C language, the trigonometric functions of CMSIS use predefined look-up tables to speed up the calculation process. When examined in detail, it can be seen that this speeds up to five times. The same is true for the FFT functions. CMSIS FFT functions are 2.5-15 times faster than their counterparts in the C language. Hence, the CMSIS DSP library is used for all mathematical operations of the proposed method.

4. EMBEDDED SYSTEM IMPLEMENTATION

When an algorithm is implemented on an embedded platform, it must be optimized for the selected hardware to meet real-time requirements. In this chapter, the optimization steps for the proposed active vibration cancellation method are described. First, the performance criteria of the parameters of the Kay algorithm are considered. Then, the improvements offered by the proposed algorithm are examined. After that, software optimization steps for the embedded implementation are explained in detail. Finally, details of the embedded implementation of the competitive methods are given.

4.1. PERFORMANCE CRITERIA OF KAY'S ALGORITHM PARAMETERS

In this section, we look in detail at the effects of the Kay algorithm's SNR, frequency step size, and number of elements parameters on parameter estimation time and accuracy. But before we explain them, we explain how to generate the WGN signal for offline experiments.

4.1.1. Generating White Gaussian Noise

WGN is a special noise signal that has a normal distribution in the time domain and a constant power spectral density in the frequency domain. STM32 microcontrollers have a specific RNG module to generate 32-bit random numbers. But these numbers have a uniform distribution instead of a normal distribution. Therefore, the WGN signal is obtained from the generated random numbers using the Box-Muller algorithm [69] given in Algorithm 4.1.

4.1.2. Signal-To-Noise Ratio

To observe the effect of SNR on the accuracy of parameter estimation, the SNR value is varied from -15 dB to 15 dB when the associated parameters are set as follows: A is 0.83, ϕ is 1.024 radians, f_s is 10 kHz, f is 72.6 Hz, the frequency step size is 0.1 Hz, and N is 1024. The average estimation results for 100 runs are given in Table 4.1. When total estimation times are compared, it can be seen that they have almost the same values, which means that

Algorithm 4.1. Box-Muller algorithm.

```

1: a and b are 32-bit uniformly distributed random numbers generated by the
   RNG module of the STM32 microcontroller
2: x and y are uniformly distributed random numbers
3: n1 and n2 are normally distributed random numbers
4: stddev is the standard deviation of the normal distribution
5: mean is the mean of normal distribution
6: while (r == 0.0) or (r >= 1.0) do
7:   Generate a and b
8:    $x \leftarrow 2.0 * a / 0xFFFFFFFF - 1.0$ 
9:    $y \leftarrow 2.0 * b / 0xFFFFFFFF - 1.0$ 
10:   $r \leftarrow x * x + y * y$ 
11: end while
12:  $n1 \leftarrow x * stddev * \sqrt{-2.0 * \log(r)/r} + mean$ 
13:  $n2 \leftarrow y * stddev * \sqrt{-2.0 * \log(r)/r} + mean$ 

```

changing the SNR value does not affect the total parameter estimation time.

Table 4.1. The estimation results for varying SNR values.

SNR (dB)	f (Hz)	A	P (rad)	Duration (sec)
-15	72.5	0.870	0.981	206.2
-10	72.5	0.836	0.986	206.2
-5	72.6	0.827	1.022	206.2
0	72.6	0.833	1.022	206.3
5	72.6	0.829	1.026	206.3
10	72.6	0.829	1.028	206.3
15	72.6	0.830	1.024	206.3

In these figures, the x axes show the SNR values in dB which is a logarithmic quantity. Therefore, the logarithmic representation is also used for the y-axes. It can be seen that the estimation accuracy for each parameter increases by approximately one decade when the SNR value changes by one decade. These results are coherent with Equation 2.12, Equation 2.13 and Equation 2.14. According to these equations, the estimated parameters $\hat{f}$, $\hat{\phi}$, and $\hat{A}$ are directly proportional to σ^2 and, if the amplitude is constant, to the SNR.

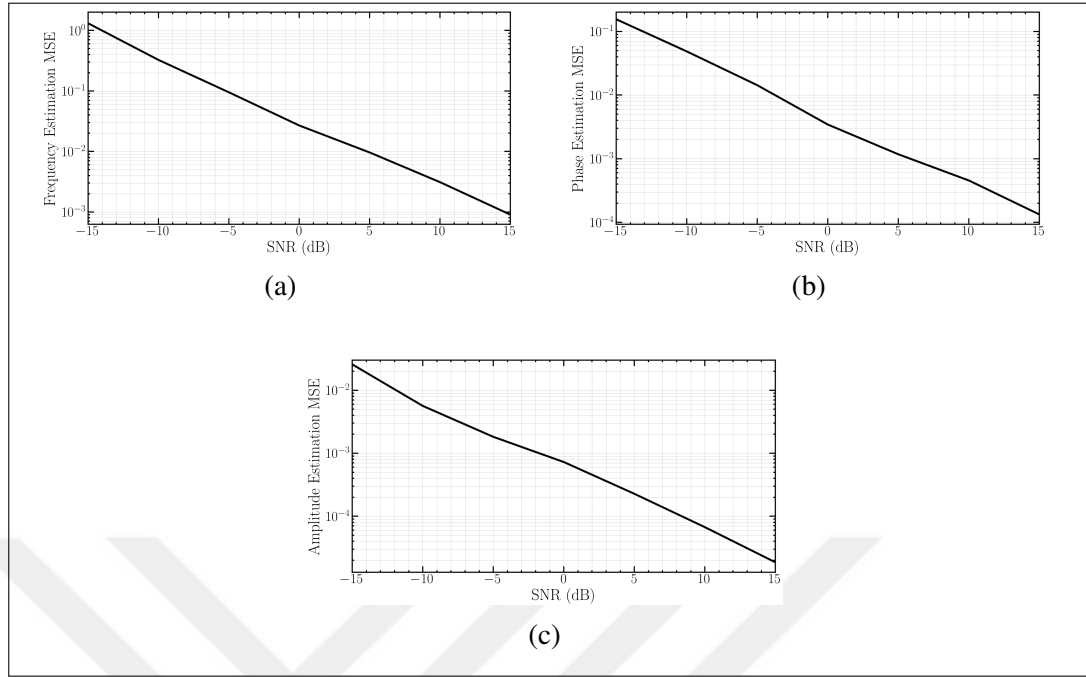


Figure 4.1. Estimation MSE wrt SNR. (a) Frequency, (b) phase, (c) amplitude.

4.1.3. Frequency Step Size

To observe the effect of frequency step size on the accuracy of parameter estimation, the value of frequency step size is varied from 1 to 0.01 when the associated parameters are set as follows: A is 0.83, ϕ is 1.024 radians, f_s is 10 kHz, f is 72.6 Hz, SNR is -10 dB, and the number of elements N is 1024. The average estimation results for 100 runs are given in Table 4.2. When total estimation times are compared, it can be seen that the frequency step size and estimation time are inversely proportional.

To better observe the effects of frequency step size on the accuracy of parameter estimation, the mean square error (MSE) is calculated for each estimate. The resulting graphs for frequency, phase, and amplitude estimation MSE are given in Figure 4.2a, Figure 4.2b, and Figure 4.2c respectively.

It can be seen that when the frequency step size is changed, estimation accuracy is not affected significantly according to MSE. But user cannot obtain 0.01 Hz frequency accuracy using one Hz frequency step size. So the user must decide the frequency step size according to the setup that will be used.

Table 4.2. The estimation results for varying frequency step size values.

Frequency step size (Hz)	f (Hz)	A	P (rad)	Duration (sec)
1	72.7	0.838	1.010	20.6
0.5	72.5	0.825	1.008	41.2
0.25	72.7	0.834	1.067	82.5
0.1	72.5	0.836	0.986	206.2
0.05	72.5	0.839	0.994	412.5
0.025	72.5	0.836	1.009	824.9
0.01	72.7	0.838	1.052	2062.1

It can be seen that the accuracy of the estimate does not change significantly when the frequency step size is changed. However, the user cannot achieve a frequency accuracy of 0.01 Hz with a frequency step size of one Hz. Therefore, the user must set the frequency step size according to the device being used.

4.1.4. Number of Elements

To observe the effect of the number of elements on the accuracy of parameter estimation, it is varied from 64 to 4096 when the associated parameters are set as follows: A is 0.83, ϕ is 1.024 radians, f_s is 10 kHz, f is 72.6 Hz, SNR is 0 dB, and frequency step size is 0.1 Hz. The average estimation results for 100 runs are given in Table 4.3. When total estimation times are compared, it can be seen that number of elements and the estimation time are inversely proportional.

To better observe the effects of the number of elements on the accuracy of the parameter estimate, the mean square error (MSE) is calculated for each estimate. The resulting graphs for frequency, phase, and amplitude estimation MSE are given in Figure 4.3a, Figure 4.3b, and Figure 4.3c respectively.

In these figures, the logarithmic representation is used for both the x-axis and the y-axis. It can be seen that the estimation accuracy for phase and amplitude increases by approximately one decade when the value of the number of elements changes by one decade. On the other

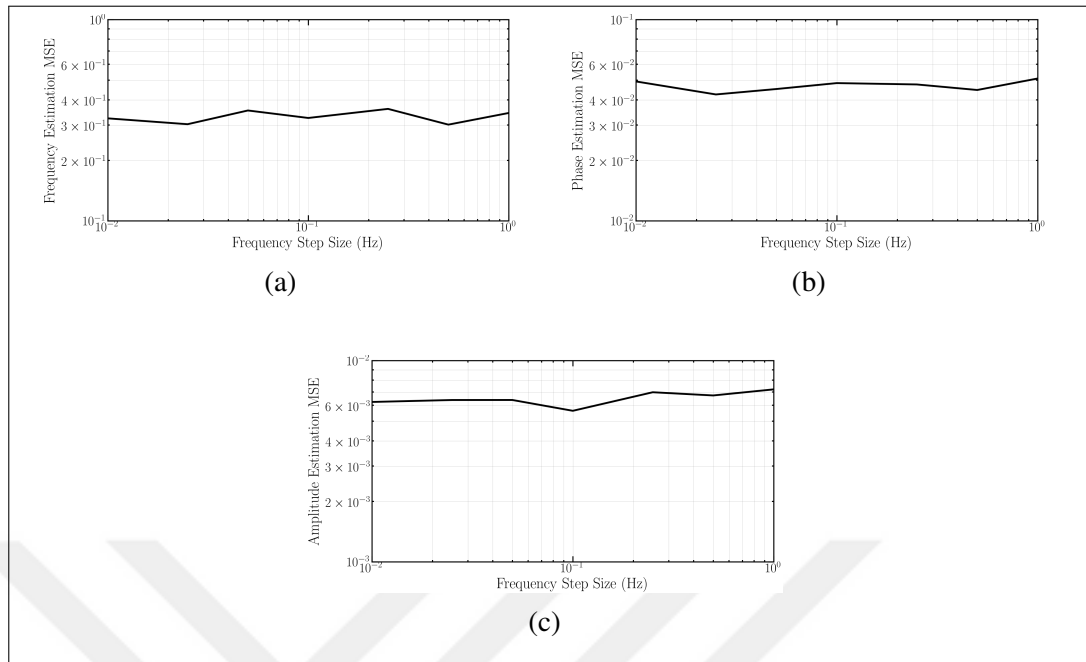


Figure 4.2. Estimation MSE wrt frequency step size. (a) Frequency, (b) phase, (c) amplitude.

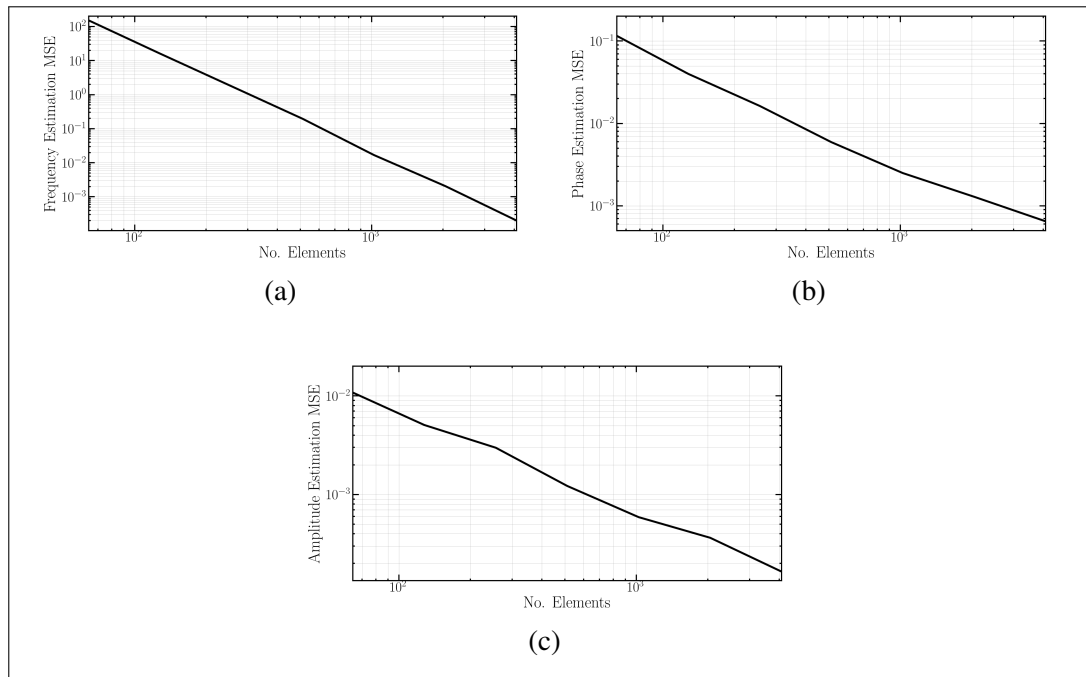


Figure 4.3. Estimation MSE wrt the number of elements. (a) Frequency, (b) phase, (c) amplitude.

Table 4.3. The estimation results for varying number of elements values.

No. Elements	f (Hz)	A	P (rad)	Duration (sec)
64	73.9	0.868	1.077	13.2
128	72.2	0.829	0.999	26.3
256	72.8	0.839	1.023	52.3
512	72.7	0.838	1.021	103.9
1024	72.6	0.828	1.006	206.9
2048	72.6	0.838	1.052	413.2
4096	72.6	0.834	1.023	828.4

hand, the estimation accuracy for frequency increases by about three decades. These results are consistent with Equation 2.12, Equation 2.13, and Equation 2.14. According to these equations, the estimated parameters $\hat{\phi}$, and $\hat{A}$ are inversely proportional to $1/N$, and $\hat{f}$ is inversely proportional to $1/N^3$.

4.2. IMPROVING PERFORMANCE VIA THE STEPS OF THE PROPOSED METHOD

This section considers in detail the impact of the improvements in the proposed method on the time and accuracy of parameter estimation.

4.2.1. Fast Fourier Transform

As we explained in Section 2.2.1, the first step of the improvements is to apply the FFT to the input signal to obtain a crude frequency value. To observe the effect of the FFT on the accuracy of the parameter estimation, both the Kay algorithm and the FFT-based approach are applied. Here, the corresponding parameters are set as follows: A is 0.83, ϕ is 1.024 radians, f_s is 1 kHz, f is 72.6 Hz, SNR is 5 dB, the number of elements is 32, and frequency step size is 0.1 Hz. The FFT is applied to 128 elements by applying zero padding for the remaining elements. The average estimation results for 1000 runs are given in Table 4.4.

Here the MSE value for each parameter is given to better understand the accuracy of the

Table 4.4. The estimation results for Kay's method and FFT-based method.

Method	f (Hz)	A	P (rad)	f MSE	A MSE	P MSE	Duration (ms)
Kay	72.3	0.836	0.998	3.50	6.73E-03	44.9E-03	699.3
FFT	72.8	0.831	1.056	3.43	5.45E-03	42.3E-03	28.2

parameter estimation. It can be seen that the accuracy of parameter estimation is almost identical in both approaches.

4.2.2. Search Algorithm

As we explained in Section 2.2.2, the second step of the improvements is to apply a search algorithm to estimate the signal frequency after the FFT. To observe the effect of search algorithms on the accuracy of parameter estimation, the same setup is applied separately to the bisection, golden mean, and binary search-based approaches and the secant method provided by Singh *et al.* [54]. Here the corresponding parameters are set the same as in Section 4.2.1. The average estimation results for 1000 runs are given in Table 4.5.

Table 4.5. The estimation results for selected search algorithms.

Method	f (Hz)	A	P (rad)	f MSE	A MSE	P MSE	Dur. (ms)
Bisection	72.4	0.840	1.011	3.53	6.30E-03	43.9E-03	2.47
Golden-Mean	72.7	0.843	1.025	3.54	5.96E-03	41.5E-03	1.82
Secant	72.4	0.836	1.009	3.58	5.87E-03	44.9E-03	1.68
Binary	72.5	0.838	1.023	3.41	6.09E-03	41.1E-03	1.55

When we analyze the MSE value for each parameter, it can be seen that the accuracy of parameter estimation is almost identical for all approaches. However, the binary search-based approach is faster than the proposed bisection and golden mean search methods. It can be seen that the secant method provided by Singh *et al.* [54] is faster than the bisection method, although it also involves derivation operations. This is because the bisection method uses a simple mean approach to find the maximum point. On the other hand, the binary search method is also faster than the secant method because of the additional derivative and division

operations. Hence, the binary search method is selected for the proposed method. It is found that the approach based on the binary search method is about 8% faster than the approach based on the secant method. Also, comparing the memory consumption shows that both methods use 1.67 kB RAM. But the secant method uses 10 kB of flash, while the binary method uses 7.5 kB of flash. So the binary method uses 25% less memory overall.

The method based on binary search can be improved one step further by applying a two-stage search. In the first stage, the binary search algorithm is applied to find the frequency with ten times the desired accuracy. For example, if the desired frequency accuracy is 0.1 Hz, the frequency is first estimated with an accuracy of one Hz. Then the binary search is applied again around this frequency value to estimate the frequency with the desired accuracy. The average estimation results for 1000 runs are given in Table 4.6.

Table 4.6. The estimation results for improved binary search-based method.

f (Hz)	A	P (rad)	f MSE	A MSE	P MSE	Dur. (ms)
72.5	0.839	1.021	3.63	6.63E-03	45.7E-03	1.31

It can be seen that the estimation time of the binary search-based method is improved by 15% with this approach. Consequently, this approach is about 22% faster than the secant-based approach.

4.2.3. Iterative Calculations

As we explained in Section 2.2.3, the last step of the improvements is to use predefined look-up tables and calculate some sine and cosine values with iterative calculations to reduce the computation time of the H-array. First, predefined look-up tables are applied to the system. Here the corresponding parameters are set in the same way as in Section 4.2.1. The average estimation results for 1000 runs are given in Table 4.7.

It can be seen that the estimation time of the binary search-based method is improved by 10% with this approach. Thus, this approach is about 30% faster than the secant-based approach. If we look at the memory usage, 21.8 kB RAM of memory is used for the look-up table approach. On the other hand, 16.8 kB RAM of memory is used for the normal binary search.

Table 4.7. The estimation results for the improved binary search-based method with iterative calculations.

f (Hz)	A	P (rad)	f MSE	A MSE	P MSE	Dur. (ms)
72.5	0.840	1.028	3.57	6.41E-03	44.3E-03	1.20

So, memory usage increases by 30% with this approach.

4.3. EMBEDDED IMPLEMENTATION IMPROVEMENTS

The embedded real-time implementation of an algorithm is a completely different process and has its own dynamics. This is because all the timing criteria of the system must be strictly adhered to in real time. In this section, we will explain three real-time implementation improvements for the proposed method.

4.3.1. Obtaining Input Samples

In real-time, it is not practical to obtain N samples of the input signal from ADC and then apply the proposed method to this data because the system must wait until all N samples are obtained. If the sampling frequency f_s is too low, it may take a very long time to obtain N samples. On the other hand, increasing f_s leads to an increase in the number of N samples. This leads to increased execution time and may be undesirable for the system.

So instead of getting N samples of the input signal, smaller n samples of the input signal are first obtained using DMA. Then the old input signal is shifted by n samples and the oldest n samples of the input data are shifted out. Finally, the new n samples are swapped with the old n samples of the input data, as shown in Figure 4.4.

After new n samples are obtained, the proposed method is applied to new data array with N samples. In this process, the total data acquisition time of DMA must be longer than the total execution time of the proposed method. Consequently, the algorithm is processed before new data arrives.

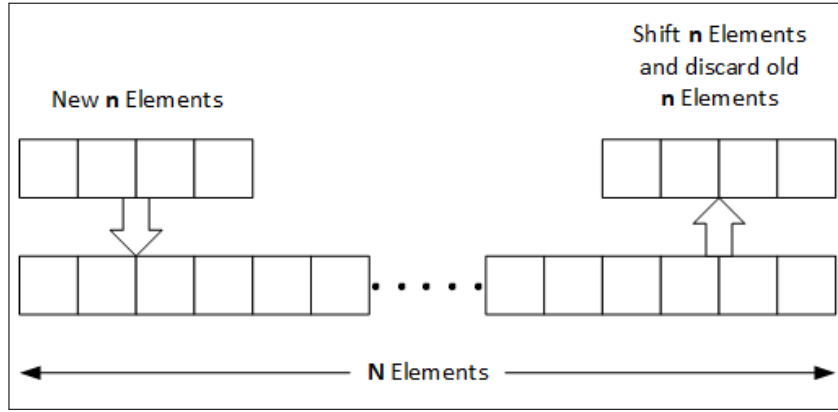


Figure 4.4. Obtaining input samples in a real-time system.

Therefore, we found that it would take a lot of time to get all the elements of the input array at once and apply the algorithm in the embedded real-time application. For example, if N is set to 32 and f_s is set to 1 kHz, the algorithm is applied every 32 ms, while the total execution time is about 7 ms. So the system has to wait another 25 ms to execute the algorithm again. Therefore, we have reduced the waiting time by fetching the input signal in smaller chunks using DMA. However, we have specified that the time to retrieve these elements should be longer than the runtime of the algorithm.

4.3.2. Calculating Phase Delay

In real time, the phase value estimated by the proposed method is not sufficient to generate a cancellation signal. This is because the phase of the input signal is constantly changing during the execution of the algorithm. In the real-time implementation, the phase delay n_d is calculated as the number of samples with $(t_p + t_c) * f_s$. Here t_p is the estimated phase value in seconds and t_c is the total execution time of the proposed method from the receipt of the input data to the generation of the cancellation signal sample.

But this approach produces an error if the input signal for the selected f_s is not periodic, as shown in Figure 4.5. Here f and f_s are selected as 70 Hz and one kHz respectively. The pseudo period can be calculated as $1000/70 = 14$. It can be seen that each pseudo period creates a phase error of 0.28 elements.

To prevent this, the frequency values in the bandwidth range of the input signal are divided

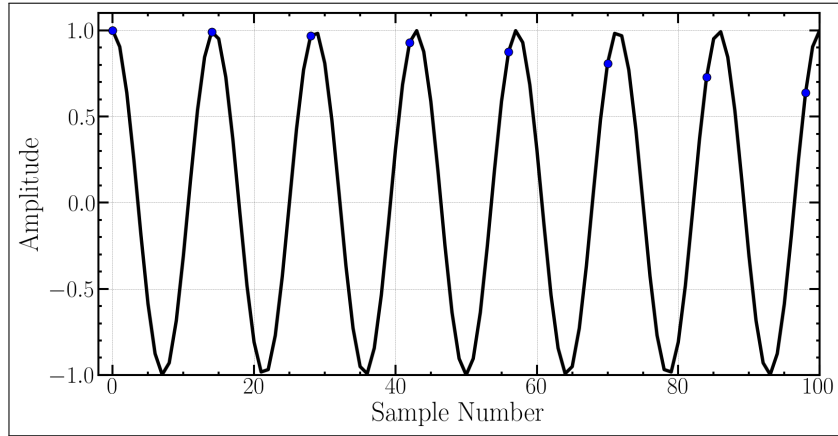


Figure 4.5. 70 Hz. sinusoidal signal when the sampling frequency is one kHz.

into four groups. The first group contains the frequencies that are periodic for the selected f_s . The second, third, and fourth groups contain frequencies that have a fundamental period of $f_s/10$, f_s and $10 * f_s$ respectively. For example, if f_s is selected as 10 kHz, frequency values such as 50, 62.5, 100, etc. are in the first group; 60, 70, 90, etc. are in the second group; 61, 62, 63, etc. are in the third group; and 61.1, 61.2, 61.3, etc. are in the fourth group.

For the last three groups, a pseudo-period is calculated as $(int)f_s/f$ and the phase error for one cycle as $f_s/f - (int)f_s/f$. Then a counter is continuously incremented between 0 and $f_s * 10$ to decide which cycle the signal is in. Then the phase error is calculated as $cycle_number * phase_error$ and added to n_d .

However, this approach leads to serious phase jumps if f_s is chosen very low to shorten the execution time of the algorithm, but the input signal is sampled under heavy noise. This is because the phase resolution is very low in such situations. Therefore, the parts of frequency estimation, phase & amplitude estimation and generation of the cancellation signal are divided into three parts: *Task1*, *Task2* and *Task3*.

Task1 performs the frequency estimation part of the proposed method. It has the lowest priority among these tasks. The sampling frequency for this part f_{sslow} is chosen lower than f_s to reduce the execution time of the algorithm. For example, in the mechanical implementation of the proposed method, f_{sslow} is chosen as 1 kHz and *Task1* is triggered with a DMA interrupt of the ADC1 module for every four samples. Then the frequency estimation is performed for 32 samples.

Task2 performs the amplitude and phase estimation parts of the proposed method. It has the middle priority among these tasks. The sampling frequency for this part f_{sfast} is chosen equal to f_s . For example, in the mechanical implementation of the proposed method, f_{sfast} is chosen as 10 kHz and *Task2* is triggered with a DMA interrupt of the ADC2 module for every 16 samples. Then the amplitude and phase estimation is performed for 128 samples. After this task is completed, a flag is set to calculate n_d in *Task3*.

Task3 performs the amplitude and phase estimation parts of the proposed method. It has the highest priority among these tasks. The sampling frequency for this part is f_s . For example, in the mechanical implementation of the proposed method, f_s is chosen as 10 kHz and *Task3* is triggered with a DMA interrupt of the ADC3 module for each sampling of two different channels. The phase delay calculation part of the proposed method is realized here when the flag is set. In addition, a cancellation signal is generated according to the estimated parameters and the DAC module is triggered accordingly.

Therefore, we found that the predicted phase value alone is not sufficient to generate the cancellation signal in the real-time embedded application. This is because the phase of the input signal is constantly changing during the execution of the algorithm. Therefore, using the estimated phase value alone can lead to catastrophic errors. Instead, we presented an effective method that adds the running time of the algorithm to the phase calculation and gives the resulting phase shift in terms of the number of elements. However, this approach caused serious phase jumps when a low sampling frequency was used to reduce the number of input samples and thus the execution time in the presence of high noise. Therefore, we increased the accuracy of the phase estimation by separating the frequency and phase/amplitude estimation parts and running them at different sampling rates.

4.3.3. Using Embedded Software Architecture

There are four simpler embedded software architectures in the literature for running our task-based system. These are simple round-robin, round-robin with interrupts, function queue scheduling, and real-time operating system (RTOS).

Simple round-robin is the simplest architecture among them because it has no interrupts. All

tasks are executed one after the other in a while loop. The maximum latency is the duration of the entire loop. But this architecture raises a problem when there is a task with a long execution time. This is because it prevents the execution of other tasks.

In round-robin with interrupts, tasks are divided into two groups: urgent tasks and non-urgent tasks. Urgent tasks are executed by interrupt service routines and have different priority levels. Non-urgent tasks have the same priority and are executed in a simple round-robin fashion. Data sharing is the most important issue in this architecture and must be carefully managed. The maximum latency for non-urgent tasks is the duration of the main loop plus the execution times for interrupts. However, urgent tasks are executed according to priorities as soon as they are triggered.

In function-queue scheduling, tasks are divided into two groups: urgent tasks and non-urgent tasks, as in the round-robin procedure with interrupts. Here, non-urgent tasks also have priorities and are added to the queue according to these priority levels. However, non-urgent tasks with low priority may never be executed and therefore need to be handled carefully.

In the RTOS, a scheduler takes care of scheduling all tasks according to their priorities. A task can be suspended by the RTOS to execute a task with higher priority. RTOS provides a stable system, independent of changes in the code. However, it is more complicated than other architectures and uses extra processor resources and memory.

As a result, we compared different software architectures for the embedded implementation of the proposed method. We have found that a simple round-robin procedure is not sufficient for our priority-based divided approach. We also found that function queue scheduling is not suitable as there is no non-urgent task that needs priority in the proposed method. The round-robin method with interrupts and RTOS is suitable for the application, but RTOS requires additional memory and processing overhead. If we compare the execution time for both architectures, it shows that the total execution time of the proposed method increases for RTOS but is negligible. On the other hand, the RTOS requires an additional 15 kB of flash memory (33 percent more) and 20 kB RAM (22 percent more) for the embedded implementation. Therefore, the round-robin method with interrupts is selected as the most efficient method for the embedded implementation.

4.4. EMBEDDED IMPLEMENTATION OF THE PROPOSED AND COMPETITIVE METHODS

In this section, the details of the embedded real-time implementation of the proposed active vibration cancellation method and the competitive methods are explained in detail.

4.4.1. Proposed Method

The final form of the proposed active vibration cancellation method is explained in this section. The outline of the algorithm is given in Algorithm 4.2.

Algorithm 4.2. Outline of the proposed method.

```

1: adc1_flag is the flag to control the frequency estimation process
2: x1 is the input array which will be used for the frequency estimation
3: Fhat is the estimated frequency value
4: Initialize peripheral modules
5: Create the Hanning window for FFT calculations
6: Create trigonometric look-up tables for H array calculations
7: Start related peripheral modules for periodic operations
8: while (1) do
9:   if adc1_flag == 1 then
10:    Apply Hanning window and zero padding to obtained x1 array
11:    Apply FFT to obtain crude frequency
12:    Apply binary search using look-up tables
13:    Estimate Fhat
14:    Reset adc1_flag
15:   end if
16: end while

```

Here, initialization procedures are carried out and the system is initiated by starting Timer, ADC and DMA modules. It can be seen that after the system is started, , the frequency estimation part of Task1 is performed in an infinite while loop. The process of frequency estimation is the most time-consuming part of the proposed algorithm and it does not make sense to perform this process within the interrupt function.

Task2, Task3, and the remaining processes of Task1 are controlled by interrupts in separate functions. As we explained in Section 4.3, the Task1 process is triggered by the DMA of the

ADC1 module at f_{sslow} rate. The pseudo code for the remaining Task1 operations executed in the corresponding interrupt function is given in Algorithm 4.3.

Algorithm 4.3. The process of Task1.

```

1: adc1_flag is the flag to control the frequency estimation process
2: x1 is the input array which will be used for the frequency estimation
3: for Every Trigger do
4:   Shift out the oldest  $n$  samples of x1 array
5:   Add new  $n$  samples obtained from the ADC1 module to x1 array
6:   Set adc1_flag to start the frequency estimation in the Main process
7: end for

```

As we explained in Section 4.3, the Task2 process is triggered by the DMA of the ADC2 module at f_{sfast} rate. The pseudo code for the Task2 operations executed in the corresponding interrupt function is given in Algorithm 4.4.

Algorithm 4.4. The process of Task2.

```

1: phase_flag is the flag to control the phase delay calculation process
2: x2 is the input array which will be used for the phase and amplitude estimation
3: for Every Trigger do
4:   Shift out the oldest  $m$  samples of x2 array
5:   Add new  $m$  samples obtained from the ADC2 to x2 array
6:   Estimate Phat and Ahat using the pre-estimated Fhat value and x2 array
7:   Set phase_flag to calculate the total phase delay in Task3
8: end for

```

As we explained in Section 4.3, the Task3 process is triggered by the DMA of the ADC3 module at f_s rate. The pseudo code for the Task3 operations performed in the associated interrupt function is given in Algorithm 4.5.

4.4.2. Least Mean Squares Filter

The outline of the LMS filter algorithm can be found in Algorithm 4.6. This process is triggered by DMA of the ADC3 module at f_s rate. A new sample of the desired signal is obtained and a delayed version of this signal is applied to the LMS filter. Then the coefficients

Algorithm 4.5. The process of Task3.

```

1: phase_flag is the flag to control the phase delay calculation process
2: data is the cancellation signal sample
3: DACdata is the DAC value to create desired data voltage at the output
4: error is the error signal
5: cnt is the counter to decide which cycle the input signal is in.
6: for Every Trigger do
7:   if phase_flag == 1 then
8:     Calculate total phase delay by using Phat and algorithm execution
       time, and by compensating non-periodic signal errors
9:     Reset the phase_flag
10:   end if
11:   Calculate the error using the samples obtained from two channels of
     the ADC3 module
12:   Create data using estimated parameters Fhat, Ahat and calculated
     phase delay.
13:   Calculate DACdata using data
14:   Apply DACdata to the system using the DAC module.
15:   Increase the cnt by 1 between 0 and  $f_s * 10$ 
16: end for

```

of the LMS filter are updated according to the error between the desired signal and the output of the LMS filter.

4.4.3. Recursive Least Squares Filter

The outline of the RLS filter algorithm can be found in the algorithm 4.7. This process is triggered by DMA of the ADC3 module at f_s rate. A new sample of the desired signal is obtained and a delayed version of this signal is applied to the RLS filter. Then the inverse correlation matrix, gain vector, and coefficients of the RLS filter are updated according to the error between the desired signal and the output of the RLS filter.

4.4.4. Phase-Locked Loop

The PLL algorithm is briefly described in Algorithm 4.8. This process is triggered by DMA of the ADC3 module at f_s rate. A new sample of the desired signal is obtained. The phase

Algorithm 4.6. The process of the LMS filter.

```

1: desired is input signal to be canceled
2: input is delayed version of desired signal
3: output is output of the LMS filter
4: coeffs is coefficients of the LMS filter
5: delay is the delay between desired and input signals by number of samples
6:  $\mu$  is the converge rate of the LMS filter
7: for Every Trigger do
8:   Obtain desired signal sample
9:   Update input as delayed version of the desired
10:  Compute output by filtering input with coefficients
11:   $error \leftarrow desired - output$ 
12:   $coefficients \leftarrow coefficients + \mu * error * input$ 
13: end for

```

Algorithm 4.7. The process of the RLS filter.

```

1: desired is the input signal to be canceled
2: input is the delayed version of desired signal
3: output is the output of the LMS filter
4: coeffs is the coefficients of the LMS filter
5: delay is the delay between desired and input signals by the number of samples
6:  $\lambda$  is the forgetting factor
7:  $P$  is the inverse correlation matrix
8:  $k$  is the gain vector
9: for Every Trigger do
10:  Obtain desired signal sample
11:  Update input as delayed version of the desired
12:  Compute output by filtering input with coefficients
13:   $error \leftarrow desired - output$ 
14:   $k \leftarrow (P * input) / (\lambda + input^T * P * input)$ 
15:   $P \leftarrow (P - k * input^T * P) * (1 / \lambda)$ 
16:   $coeffs \leftarrow coeffs + k * error$ 
17: end for

```

detector decides the leading position of the input signal and the input signal is amplified accordingly by the phase detector gain K_p . Then it is filtered through the loop filter to suppress higher harmonics and undesired noise. Finally, the value of the voltage-controlled oscillator is updated using this data and an output signal is generated.

Algorithm 4.8. The process of the PLL.

```

1: input is the input signal to be canceled
2: output is the output of the PLL block
3:  $vco_{int}$  is the integrator of the VCO
4:  $K_p$  is the phase detector gain
5:  $K_v$  is the VCO gain
6: state is the leading condition of the input signal
7: filt is the loop filter output
8: for Every Trigger do
9:   Obtain input signal sample
10:  Update state according to phase detector
11:  Calculate phase detector output as  $K_p * state$ 
12:  Filter this signal using the loop filter and obtain filt
13:   $vco_{int} \leftarrow vco_{int} + filt/f_s$ 
14:   $output \leftarrow \cos((2 * PI * f/f_s) + (2 * PI * K_v * vco_{int}))$ 
15: end for

```

4.5. PERFORMANCE OF HARDWARE PLATFORMS

In this section, we will compare selected STM32F429I-DISC1, 32F746GDISCOVERY and NUCLEO-H743ZI2 boards. We implement the proposed method on each hardware platform and compare the results in terms of speed and memory consumption.

For all implementations, f_{slow} is selected as 4 kHz, f_{fast} is selected as 10 kHz and f_s is selected as 10 kHz. *Task1* is triggered with a DMA interrupt of the ADC1 module for every 32 samples. Then the frequency estimation is performed for 128 samples. *Task2* is triggered with a DMA interrupt of the ADC2 module every 32 samples. Then the amplitude and phase estimation is performed for 256 samples. *Task3* is triggered with a DMA interrupt of the ADC3 module for each sample from two different channels.

First, we compare the execution time of Task1, Task2, and Task3 for each hardware platform separately. The results can be found in Table 4.8. It can be seen that the NUCLEO-H743ZI2

is the fastest among them. This is an expected result as it has a much higher clock speed than the others. It is particularly effective for running Task1, which contains intensive FFT operations, matrix operations, and trigonometric calculations.

Table 4.8. Execution time comparison of selected hardware platforms.

Platform	Task1 (ms)	Task2 (ms)	Task3 (ms)
STM32F429I-DISC1	6.86	0.126	0.0249
32F746GDISCOVERY	5.55	0.118	0.0241
NUCLEO-H743ZI2	2.42	0.105	0.0213

Next, we compare the total RAM and flash usage of the proposed algorithm for each hardware platform separately. The results can be found in Table 4.9. It can be seen that the RAM usage for the proposed algorithm is the same for each hardware platform as they are run under the same parametric conditions. However, the flash usage is different for each hardware platform because the implementations of the HAL and the CMSIS library are different.

Table 4.9. The memory usage comparison of selected hardware platforms.

Platform	RAM (kB)	RAM (%)	Flash (kB)	Flash (%)
STM32F429I-DISC1	94.25	49.09	45.47	2.22
32F746GDISCOVERY	94.4	38.02	66.8	6.52
NUCLEO-H743ZI2	94.5	18.46	72.6	3.54

So we see that using the 32F746GDISCOVERY board does not give us many advantages over the STM32F429I-DISC1 board. On the other hand, if the NUCLEO-H743ZI2 board is chosen for the implementation, the performance can be increased considerably. We can increase the number of samples for the estimation parts and increase the accuracy without exceeding the current execution time of the algorithm. However, for the selected active vibration cancellation system, the accuracy achieved with the STM32F429I-DISC1 board is sufficient. So using the NUCLEO-H743ZI2 is unnecessary and overpriced for the designed system.

5. LOOPBACK SIMULATIONS FOR VERIFICATION

Loopback is a special technique that generates and relays electronic signals within the microcontroller to test desired peripherals. For example, USART peripherals can be tested in loopback mode by connecting transmit and receive pins and comparing the transmitted and received data in real-time.

In this section, we perform loopback simulations for the proposed active vibration cancellation method in real-time as shown in Figure 5.1. Here, all operations are performed on the microcontroller. The vibration signal received from the engine accelerometer is generated in the microcontroller in real-time. Then the cancellation algorithm is performed on this signal. Finally, the cancellation signal is generated but not applied to active mounts. Instead, the vibration and cancellation signals are subtracted and the error signal obtained from the chassis accelerometer is also generated within the microcontroller.

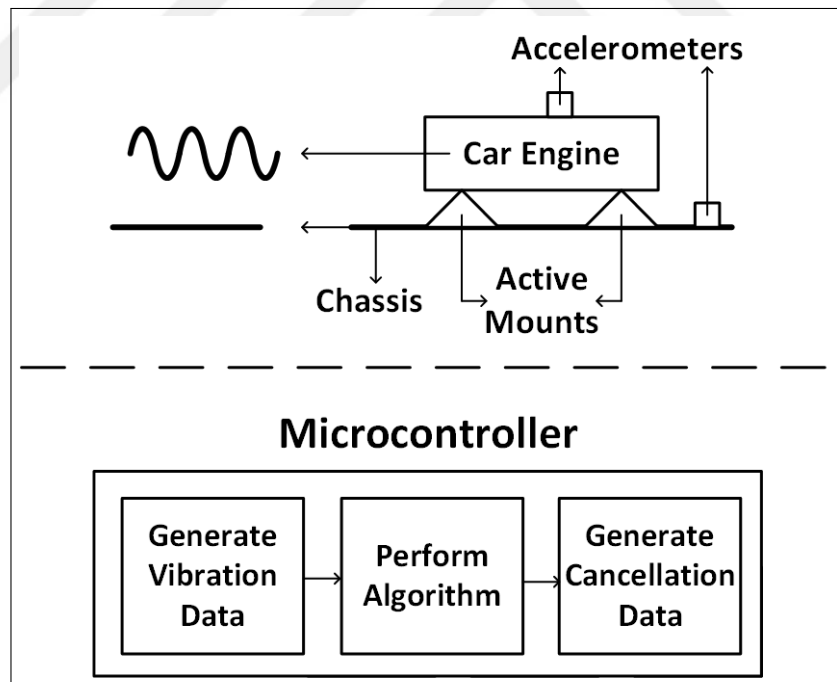


Figure 5.1. The block diagram of the loopback simulations.

We will also perform loopback simulations for competitive methods in real-time to compare the performance of the algorithms for different aspects.

5.1. SINGLE-FREQUENCY SIGNAL

In this section, the proposed and competitive active vibration cancellation algorithms are simulated for constant frequency sinusoidal signals. They are then compared in terms of accuracy, vibration suppression, speed, and memory consumption.

5.1.1. Proposed Method

Here, a sinusoidal signal with constant frequency is applied to the proposed algorithm. Signal parameters are selected as f is 50 Hz, f_s is 10 kHz and A is 1.0. A WGN signal with 10 dB SNR shown in Figure 5.2b is also added to the input signal. The resulting vibration signal is given in Figure 5.2a.

For the real-time simulation, *Task1* is triggered with a frequency of 4 kHz and 32 samples of the vibration signal are created each time. Then the frequency estimation part is realized for 128 samples. Task *Task2* is triggered with a frequency of 10 kHz and 32 samples of the vibration signal are created each time. Then the amplitude and phase estimation part is realized for 256 samples. *Task3* is triggered at a rate of 10 kHz and one sample of the vibration signal is generated each time. Then the cancellation and error signals are determined.

When the proposed algorithm is simulated under the given conditions, the resulting cancellation signal and the error signal cleared of noise are shown in Figure 5.2c and Figure 5.2d, respectively.

Here, the error signal freed from noise is used to observe the accuracy of the proposed algorithm. To understand this better, the MSE of this signal is calculated over a long period of time. The MSE of the noise removed error signal for SNR values between 0 and 30 dB is shown in Figure 5.3.

In this figure, the x-axis shows the SNR values in dB, which is a logarithmic quantity. Therefore, the logarithmic representation is also used for the y-axis. It can be seen that the accuracy of the estimate decreases linearly with decreasing SNR value.

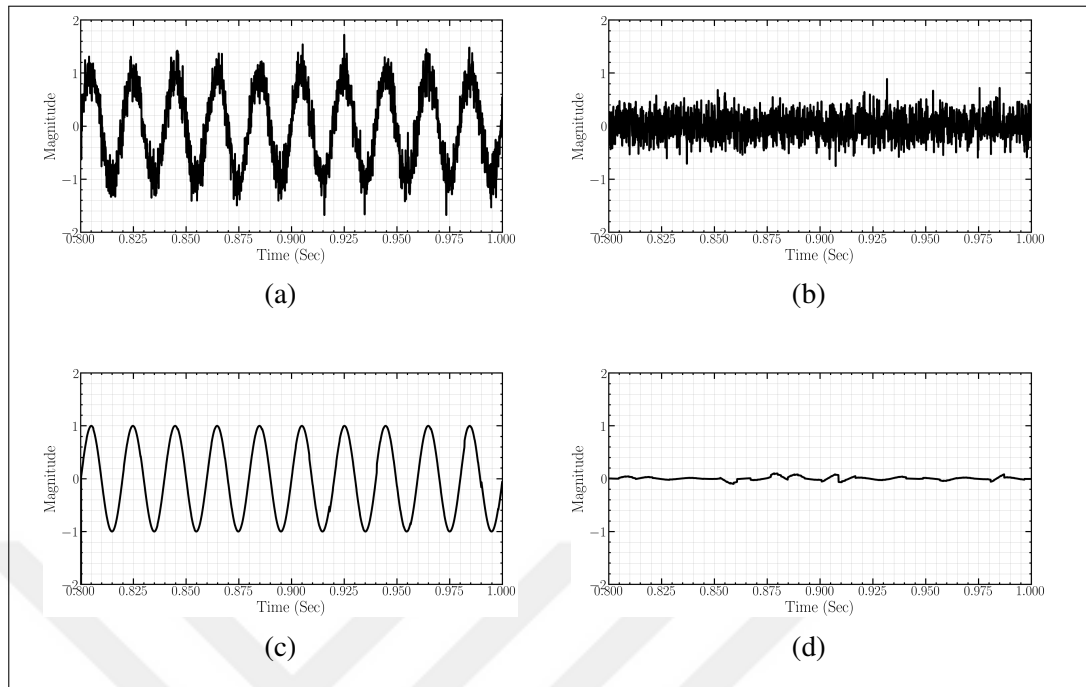


Figure 5.2. Simulation results for the proposed method. (a) Vibration, (b) noise, (c) cancellation, and (d) error signals.

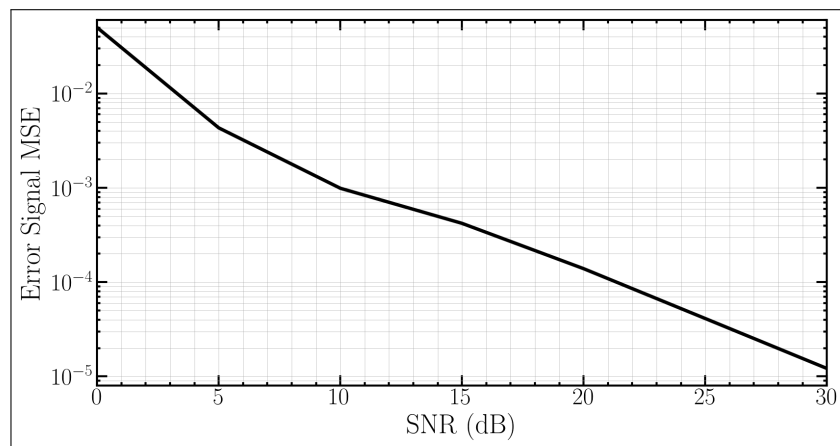


Figure 5.3. MSE of the noise removed error signal wrt SNR.

We need to examine the vibration and error signals in the frequency domain to measure the performance of the proposed method. The FFT results of these signals are shown in Figure 5.4. Here, the vibration suppression level is calculated as 42.2 dB at 50 Hz.

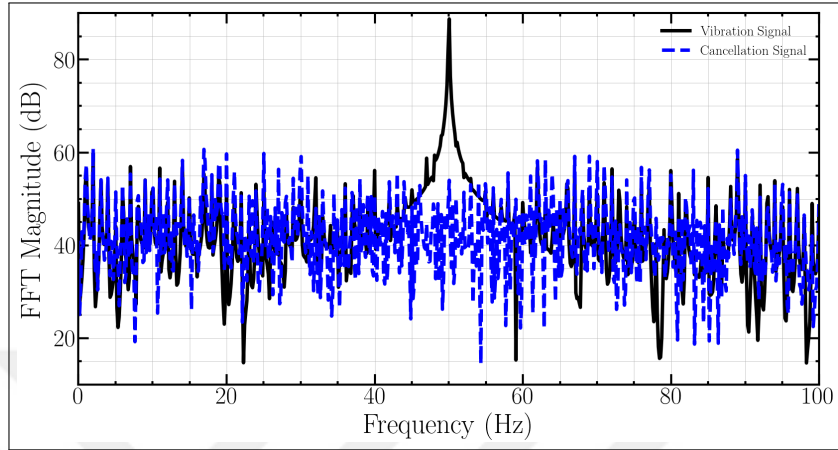


Figure 5.4. FFT results for vibration and error signal.

To better understand the performance of the proposed method, the vibration suppression level is calculated for SNR values between 0 and 30 dB. It can be seen that the vibration suppression level increases logarithmically with increasing SNR.

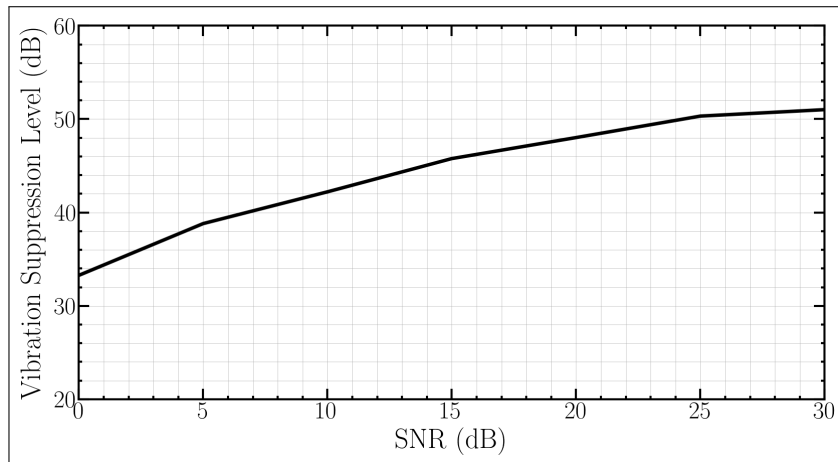


Figure 5.5. Vibration suppression level wrt SNR.

We observe the first 50 ms. of the error signal as in Figure 5.6a to obtain the response time of the proposed algorithm. It can be seen that it is measured as 19.5 ms. The same system is also simulated for an SNR value of 30 dB to observe the effect of SNR on the response time. The result is given in Figure 5.6b. It can be seen that the response time of the proposed algorithm is not affected by the SNR.

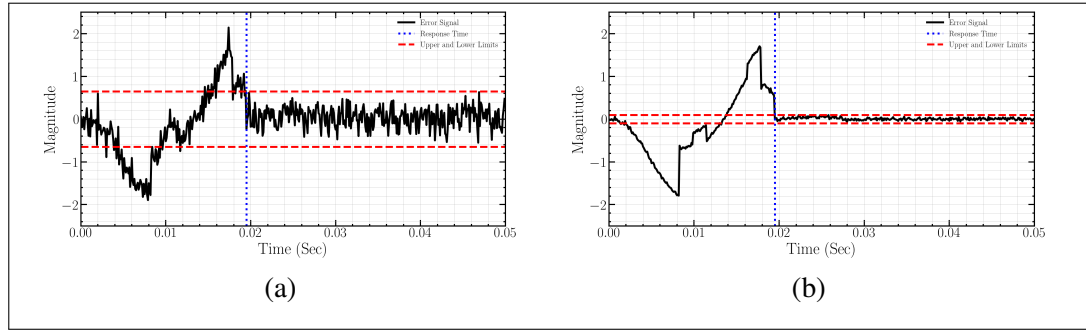


Figure 5.6. The response time of the proposed method for (a) 10 dB, and (b) 30 dB SNR.

Finally, the RAM and flash memory usage of the proposed method are given in Table 5.3. 49.1 percent of the RAM and 2.2% of the flash memory are used by the proposed method.

Table 5.1. RAM and flash memory usage of the proposed method.

Used RAM (kB)	Used Flash (kB)
94.25	45.45

5.1.2. Comparison with Competitive Methods

In this section, we compare the proposed method with competitive methods in terms of accuracy, suppression level, response time, and memory consumption. Each method is simulated with the same vibration signal with parameters f 50 Hz, f_s 10 kHz and A 1.0. Besides response time and memory consumption, other performance parameters are observed for different SNR values.

To compare the accuracy of the methods, the MSE of the noise-removed error signal is calculated for SNR values between 0 and 30 dB for each method. The results are given in Figure 5.7. In this figure, the x-axis shows the SNR values in dB, which is a logarithmic quantity. Therefore, the logarithmic representation is also used for the y-axis. As can be seen in this figure, the accuracy of the proposed method is the best among competitive methods. On the other hand, PLL has the worst accuracy among them.

Next, we will compare the vibration suppression level among the suggested methods as given in Figure 5.16. As can be seen in this figure, the vibration suppression level of the proposed

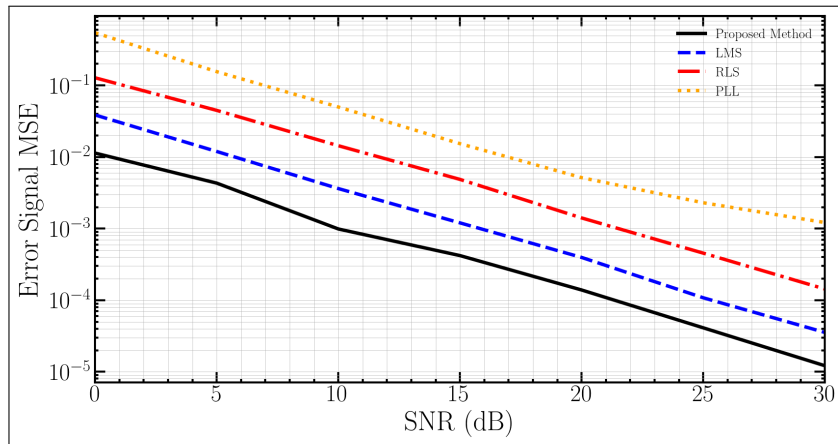


Figure 5.7. MSE of the noise removed error signal wrt SNR for each method.

method is higher than RLS when the SNR is below 16.5 dB and higher than LMS when the SNR is below 10 dB. On the other hand PLL has the worst vibration suppression level among them.

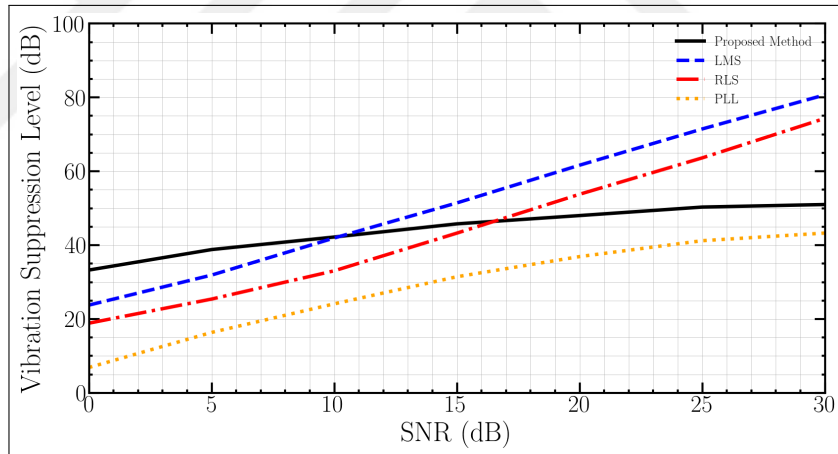


Figure 5.8. Vibration suppression level wrt SNR for each method.

We will use Figure 5.9a, Figure 5.9b, and Figure 5.9c to observe the response time of LMS, RLS, and PLL methods respectively. When we analyze them together with Figure 5.6a, it can be seen that the fastest methods are LMS with 12.9 ms. and RLS with 13.2 ms. The proposed method is slightly slower than these methods with a response time of 19.5 ms. On the other hand, PLL has the worst response time among them with 67.4 ms.

Finally, the RAM and flash memory consumption of the proposed method and competitive methods are given in Table 5.4. It can be seen that the proposed method uses the most RAM and flash memory as it uses various two dimensional matrix calculations and predefined

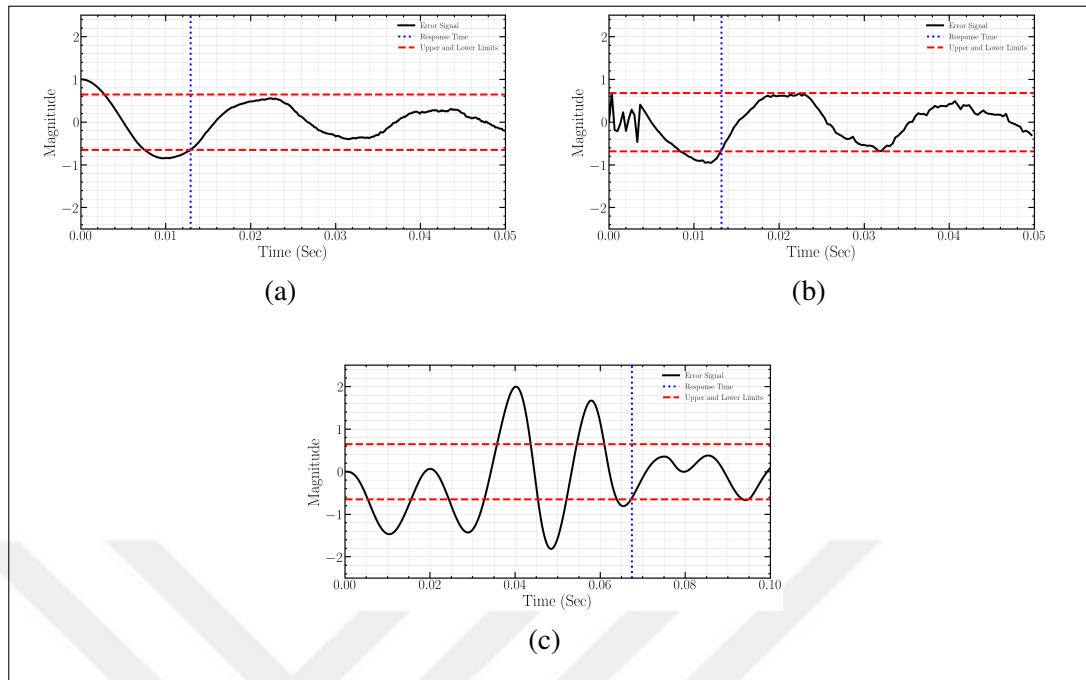


Figure 5.9. The response time of (a) LMS, (b) RLS, and (c) PLL for 10 dB SNR.

look-up tables.

Table 5.2. The memory usage comparison of vibration cancellation methods.

Method	RAM (kB)	RAM (%)	Flash (kB)	Flash (%)
Proposed Method	94.25	49.09	45.47	2.22
LMS	42.23	22.00	29.00	1.42
RLS	16.26	8.47	29.87	1.46
PLL	41.63	21.68	28.65	1.40

5.2. FREQUENCY SWEEP

In this section, the proposed and competitive active vibration cancellation algorithms are simulated for frequency sweep signals. They are then compared in terms of accuracy, vibration suppression, speed, and memory consumption.

5.2.1. Proposed Method

Here, a frequency sweep signal is applied to the proposed algorithm. The signal parameters are selected as f_{start} is 20 Hz, f_{stop} is 120 Hz, the duration of the sweep cycle is 2 seconds, f_s is 10 kHz and A is 1.0. A WGN signal with 10 dB SNR shown in Figure 5.10b is also added to the input signal. The resulting vibration signal is given in Figure 5.10a.

For the real-time simulation, *Task1* is triggered with a frequency of 4 kHz and 32 samples of the vibration signal are created each time. Then the frequency estimation part is realized for 128 samples. Task *Task2* is triggered with a frequency of 10 kHz and 32 samples of the vibration signal are created each time. Then the amplitude and phase estimation part is realized for 256 samples. *Task3* is triggered at a rate of 10 kHz and one sample of the vibration signal is generated each time. Then the cancellation and error signals are determined.

When the proposed algorithm is simulated under the given conditions, the resulting cancellation signal and the error signal cleared of noise are shown in Figure 5.10c and Figure 5.10d respectively.

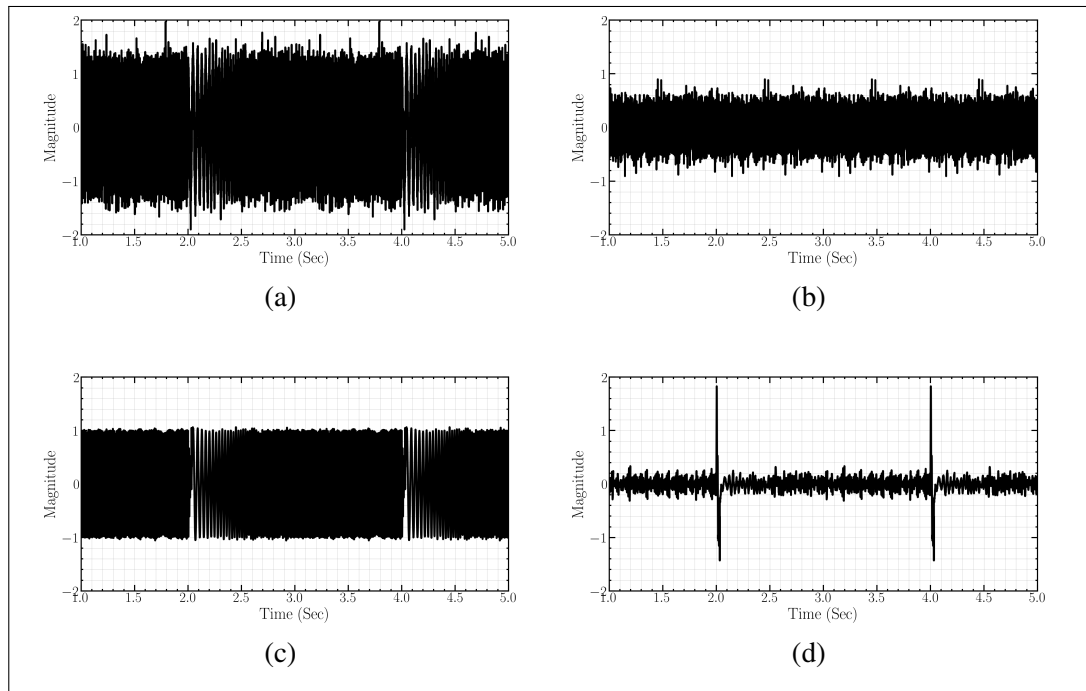


Figure 5.10. Simulation results for the proposed method. (a) Vibration, (b) noise, (c) cancellation, and (d) error signals.

Here, the error signal freed from noise is used to observe the accuracy of the proposed algorithm. To understand this better, the MSE of this signal is calculated over a long period of time. The MSE of the noise removed error signal for SNR values between 0 and 30 dB is given in Figure 5.11.

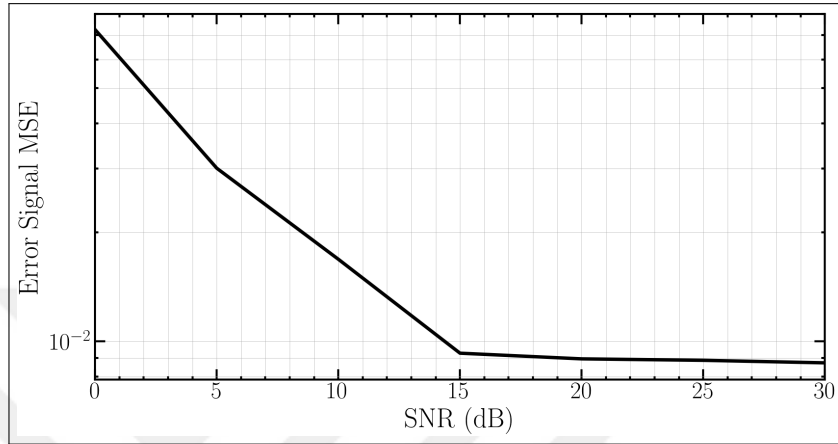


Figure 5.11. MSE of the noise removed error signal wrt SNR.

In this figure, the x-axis shows the SNR values in dB, which is a logarithmic quantity. Therefore, the logarithmic representation is also used for the y-axis. It can be seen that the accuracy of the estimation decreases almost linearly with decreasing SNR value up to 15 dB. After that, the estimation accuracy remains almost unchanged, as the error peaks shown in Figure 5.10d repeat with each change of the sweep cycle and produce consistently large MSE values at low noise.

We need to examine the vibration and error signals in the frequency domain to measure the performance of the proposed method. The FFT results of these signals are shown in Figure 5.12. The average vibration suppression level is calculated as 18.04 dB between the sweep frequency range.

To better understand the performance of the proposed method, the vibration suppression level is calculated for SNR values between 0 and 30 dB. The result is shown in Figure 5.13. It can be seen that the vibration suppression level increases logarithmically with increasing SNR.

We observe the first 100 ms. of the error signal as in Figure 5.14 to obtain the response time of the proposed algorithm. It can be seen that it is measured as 32.5 ms. Since the frequency is immediately changed from 120 Hz to 20 Hz, this is the worst response time of the proposed

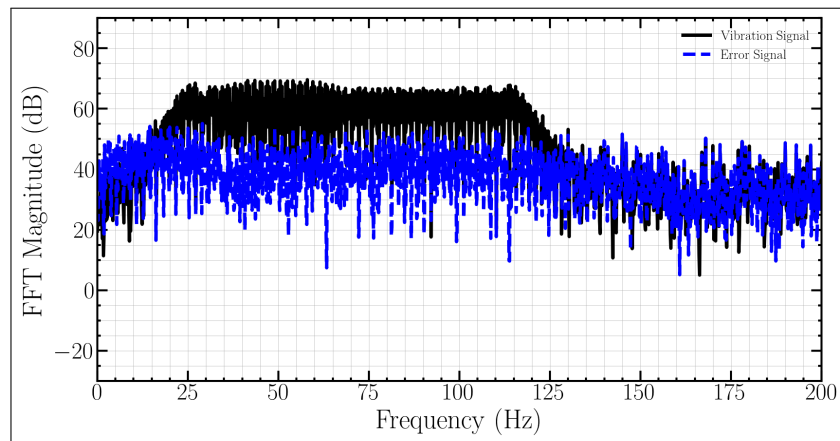


Figure 5.12. The FFT results for vibration and error signals.

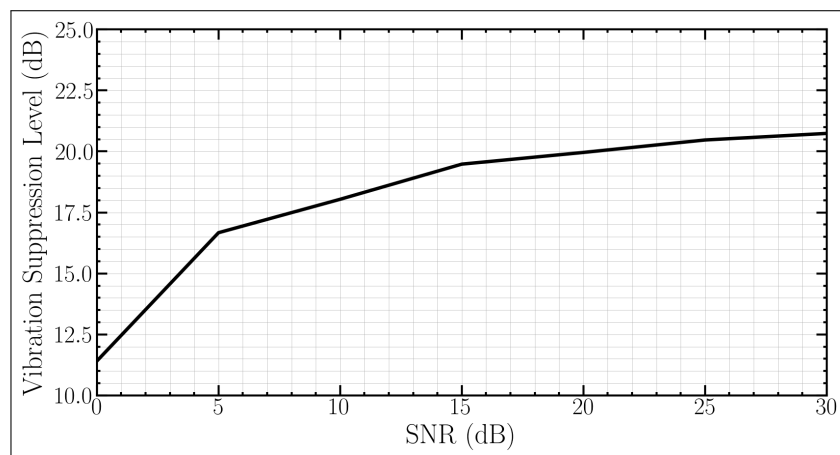


Figure 5.13. The vibration suppression level wrt SNR.

method.

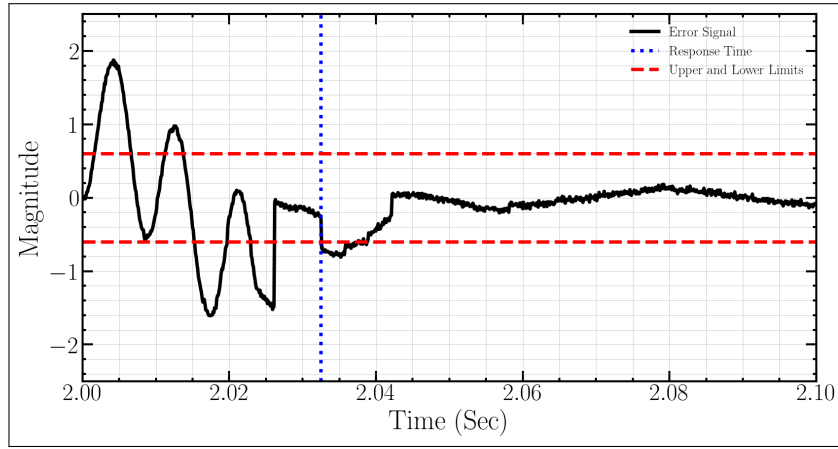


Figure 5.14. The response time of the proposed method for sweep cycle change.

The RAM and flash memory consumption of the proposed method are given in Table 5.3. It can be seen that these are the same memory values as for constant frequency. This is because the proposed method is not changed for constant frequency and frequency sweep.

Table 5.3. RAM and flash memory usage of the proposed method.

Used RAM (kB)	Used Flash (kB)
94.25	45.45

5.2.2. Comparison with Competitive Methods

In this section, we compare the proposed method with competitive methods in terms of accuracy, suppression level, response time, and memory consumption. Each method is simulated with the same vibration signal as a frequency sweep with parameters f_{start} is 20 Hz, f_{stop} is 120 Hz, sweep cycle duration is 2 seconds, f_s is 10 kHz and A is 1.0. Besides the response time and memory consumption, other performance parameters are also observed for different SNR values.

To compare the accuracy of the methods, the noise-removed error signal MSE is calculated for SNR values between 0 and 30 dB for each method. The results are given in Figure 5.15. In this figure, the x-axis shows the SNR values in dB, which is a logarithmic quantity. Therefore,

the logarithmic representation is also used for the y-axis. As can be seen in this figure, the accuracy of the proposed method is the best among competitive methods. On the other, hand PLL has the worst accuracy among them.

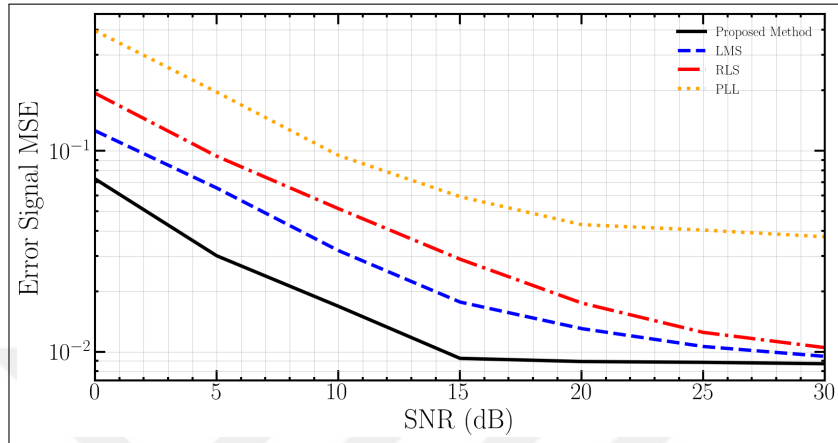


Figure 5.15. MSE of the noise removed error signal wrt SNR for each method.

Next, we will compare the vibration suppression level among the suggested methods as given in Figure 5.16. As can be seen in this figure, the vibration suppression level of the proposed method is higher than RLS when the SNR is below 16.5 dB and higher than LMS when the SNR is below 10 dB. On the other hand, PLL has the worst vibration rejection among them.

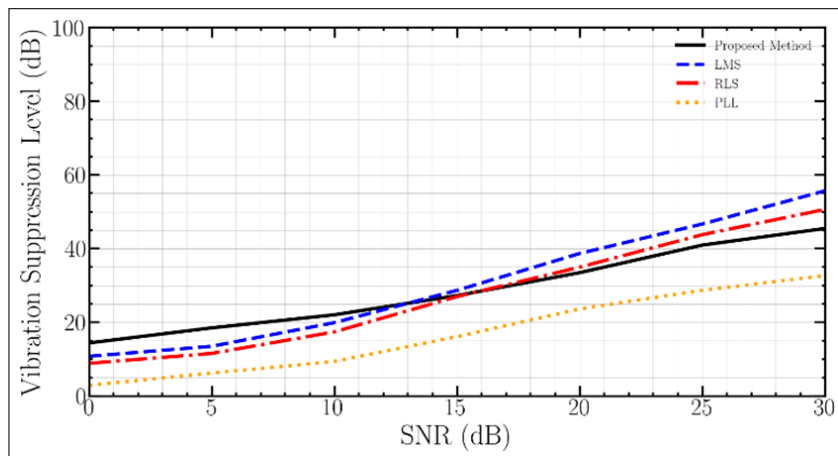


Figure 5.16. The vibration suppression level wrt SNR for each method.

We will use Figure 5.17a, Figure 5.17b, and Figure 5.17c to observe the response time of LMS, RLS, and PLL methods respectively. When we analyze them together with Figure 5.14, it can be seen that the fastest methods are LMS with 14.9 ms. and RLS with 15.7 ms. The proposed method is slower than these methods with a response time of 30.5 ms. On the other

hand, PLL has the worst response time among them with 152 ms.

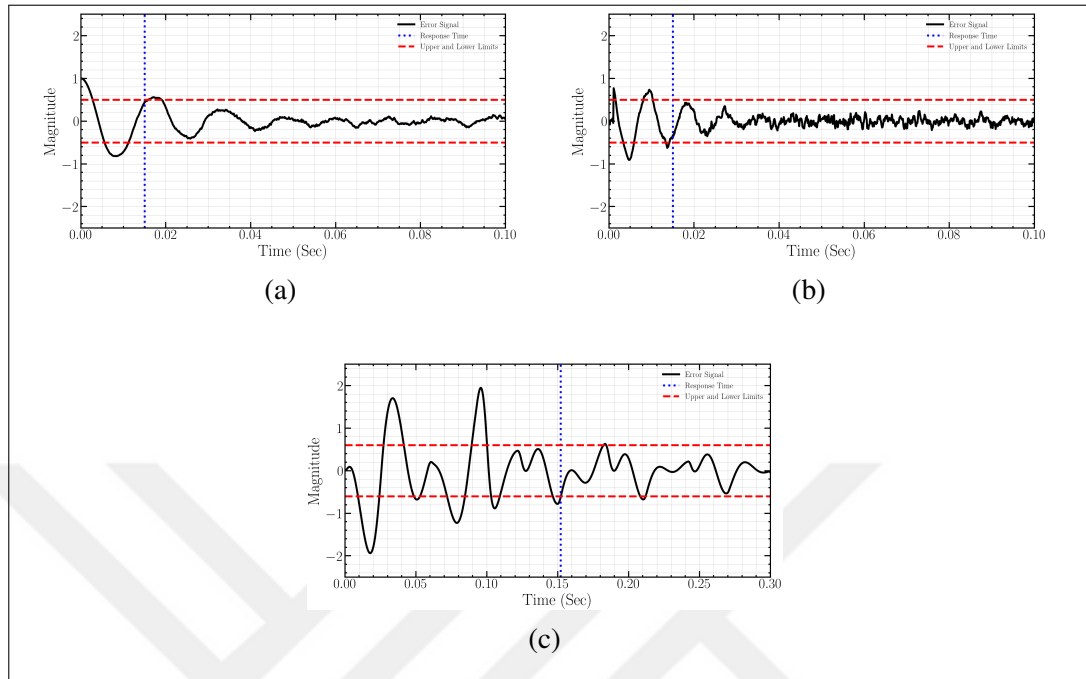


Figure 5.17. The response time of (a) LMS, (b) RLS, and (c) PLL for sweep cycle change at 10 dB SNR.

Finally, the RAM and flash memory usage of the proposed method and competitive methods are given in Table 5.4. It can be seen that these are the same memory values as for constant frequency. This is because the vibration cancellation methods use the same parameters for constant frequency and frequency sweep. Again, the proposed method uses the most RAM and flash memory as it uses various two-dimensional matrix calculations and predefined look-up tables.

Table 5.4. The response time comparison of vibration cancellation methods.

Method	RAM (kB)	RAM (%)	Flash (kB)	Flash (%)
Proposed Method	94.25	49.09	45.47	2.22
LMS	42.23	22.00	29.00	1.42
RLS	16.26	8.47	29.87	1.46
PLL	41.63	21.68	28.65	1.40

5.3. MULTI-FREQUENCY SIGNAL

In this section, we use the summation of multiple sinusoidal signals instead of a single sinusoidal signal as input to observe how the proposed method responds to this approach. The obtained results are given in Table 5.5 and Table 5.6. It can be seen that the proposed method works quite well for large frequency differences between the signals even at higher noise levels. This approach actually represents the signal harmonics. But when the frequency differences between the signals are small, the proposed method works poorly even when there is no noise in the system. The reason is that the proposed method is based on FFT and if the frequency differences between the signals are too small, they influence each other too much.

Table 5.5. The estimated frequencies when frequency separations are large.

Signals	w/o Noise	30 dB SNR	20 dB SNR	10 dB SNR
First Signal (62.6 Hz)	62.6 Hz	62.7 Hz	62.9 Hz	62.1 Hz
Second Signal (92.4 Hz)	92.5 Hz	92.3 Hz	92.1 Hz	92.8 Hz
Third Signal (120.2 Hz)	120 Hz	120 Hz	120.5 Hz	120.6 Hz

Table 5.6. The estimated frequencies when frequency separations are small.

Signals	w/o Noise	30 dB SNR
First Signal (62.6 Hz)	63.6 Hz	64.5 Hz
Second Signal (66.4 Hz)	67.9 Hz	68.8 Hz
Third Signal (70.2 Hz)	68.2 Hz	67.1 Hz

6. EXPERIMENTS ON ELECTRICAL SYSTEMS

In this section, we will perform electrical experiments for the proposed active vibration cancellation method in real-time as shown in Figure 6.1. Here, the vibration signal of the engine, which normally comes from the accelerometer, is generated by a signal generator and evaluated in real-time using the ADC module of the microcontroller. Then the cancellation algorithm is performed on this signal. Here, the vibration signal generated by the signal generator and the cancellation signal generated by the microcontroller is observed together with the oscilloscope.

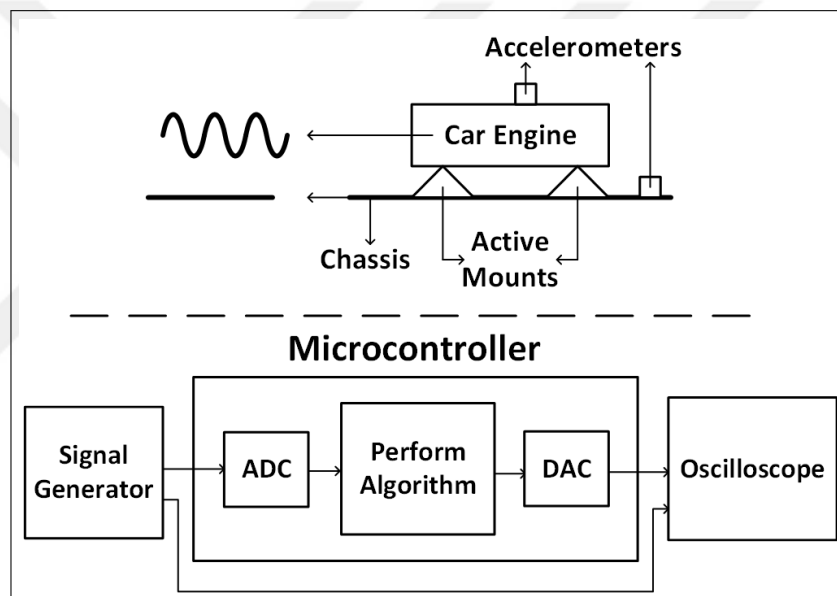


Figure 6.1. The block diagram of the electrical experiment.

6.1. ELECTRICAL SETUP

The general setup for the electrical experiments is shown in Figure 6.2. Here, a signal generator is used to produce a vibration signal, an STM32F429I-DISC1 board is used to apply the proposed method and produce a cancellation signal, and an oscilloscope is used to observe the results.

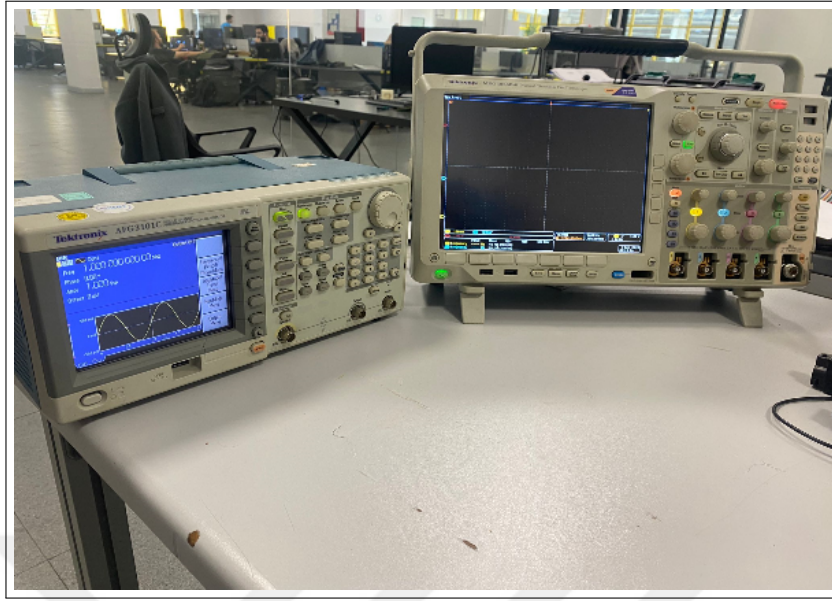


Figure 6.2. The general electrical test setup.

6.2. EXPERIMENTS

6.2.1. Single-Frequency Signal

In this section, the proposed and competitive active vibration cancellation methods are performed on an electrical setup for a sinusoidal signal with constant frequency. They are then compared in terms of vibration suppression and speed.

6.2.1.1. Proposed Method

Here, a sinusoidal signal with constant frequency is applied to the proposed algorithm. The signal parameters are selected as f is 100 Hz and A is 1.0 V. around 1.5 V. offset. This signal is sampled by the ADC module with a sampling frequency of 10 kHz and the cancellation signal is generated by the proposed method. The results can be observed on the oscilloscope as shown in Figure 6.3. The yellow signal at the bottom is the vibration signal, the green signal in the middle is the cancellation signal and the red signal at the top is the error signal.

We need to examine the vibration and error signals in the frequency domain to measure the performance of the proposed method. The FFT results of these signals are shown in Figure 6.4. Here, the vibration suppression level is calculated as 34.98 dB at 100 Hz.

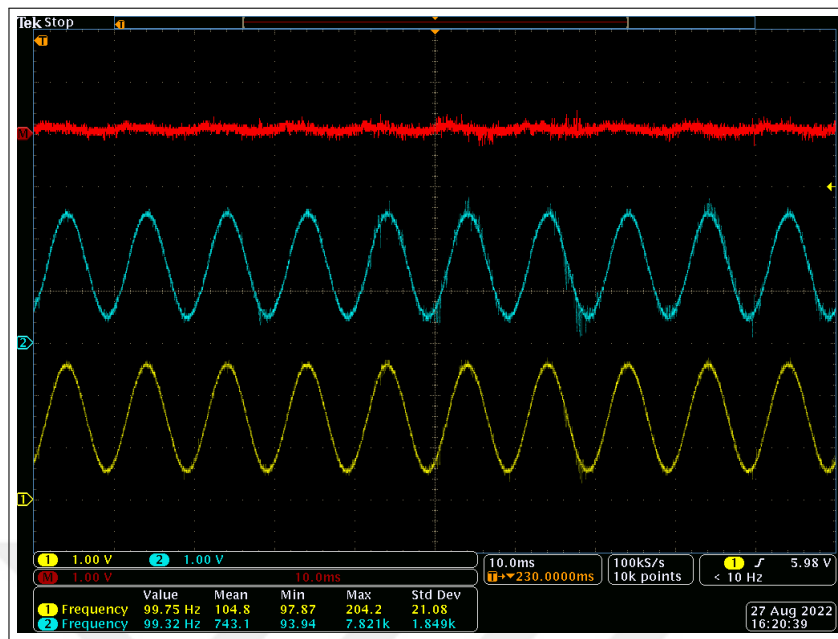


Figure 6.3. Obtained error, cancellation and vibration signals for the proposed method.

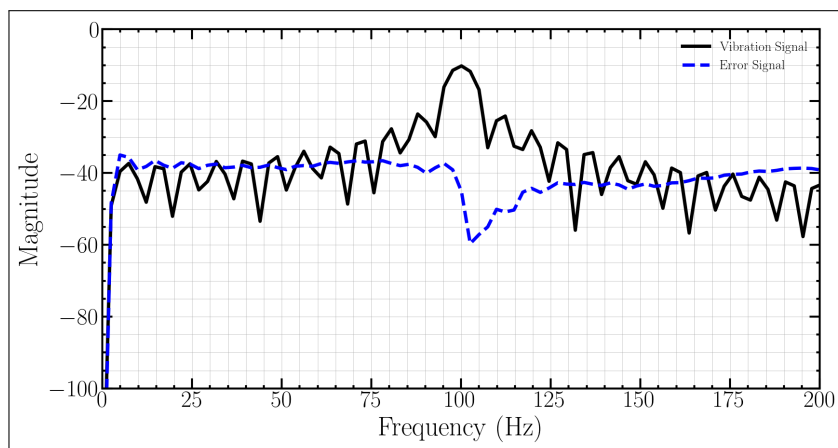


Figure 6.4. FFT results of the vibration and error signals for the proposed method.

To calculate the reaction time of the proposed method, we use the error signal given in Figure 6.5. It can be seen that it is measured as 13.5 ms.

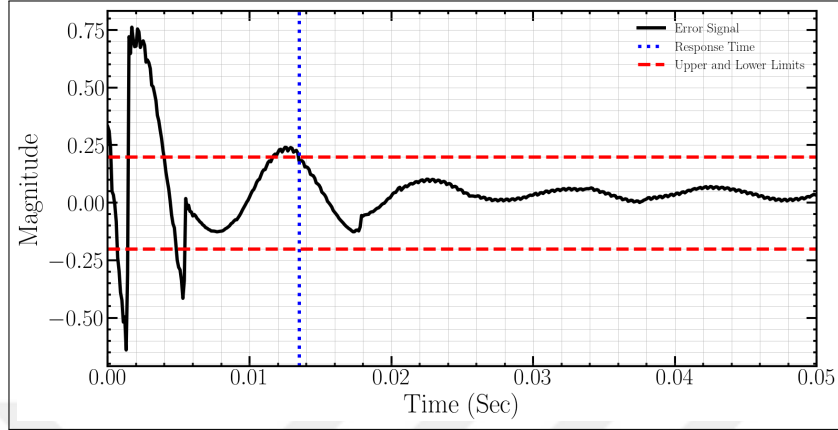


Figure 6.5. The response time of the proposed method.

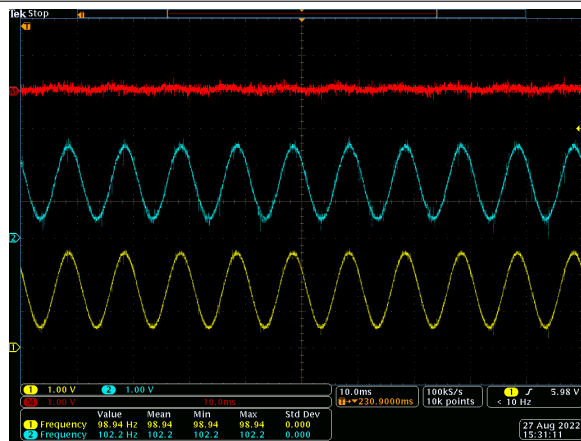
6.2.1.2. Comparison with Competitive Methods

In this section, we compare the proposed method with competitive methods in terms of vibration suppression level and response time. Each method is simulated with the same vibration signal with parameters f is 100 Hz and A is 1.0 V. around 1.5 V. offset. This signal is sampled by the ADC module with a sampling frequency of 10 kHz and the cancellation signal is generated with the selected method.

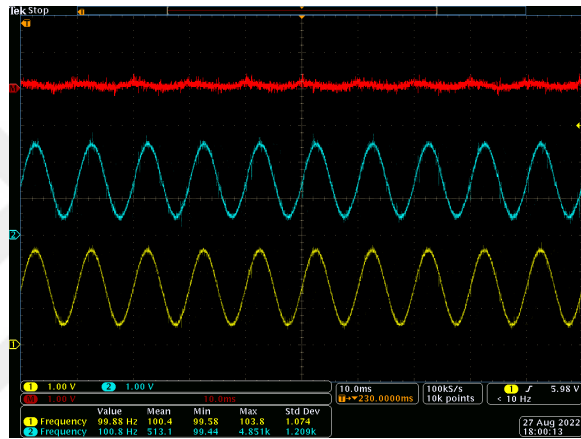
The results for the LMS, RLS, and PLL methods can be observed on the oscilloscope as shown in Figure 6.6a, Figure 6.6b, and Figure 6.6c respectively. In each figure, the yellow signal at the bottom is the vibration signal, the green signal in the middle is the cancellation signal and the red signal at the top is the error signal.

When we examine the vibration and error signals of each method in the frequency domain, we obtain the vibration suppression results given in Table 6.1. It can be seen that the proposed method has the best vibration suppression among the other methods, while the PLL is the worst, as expected.

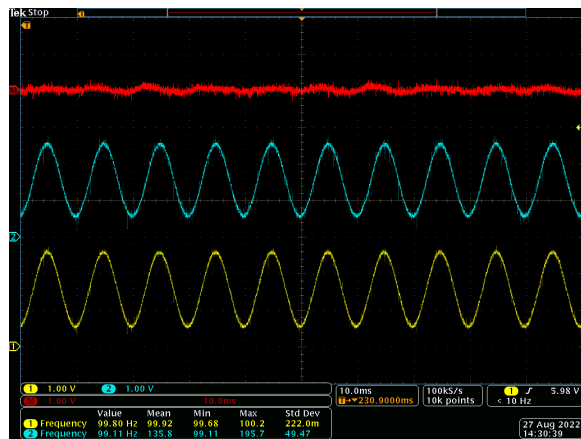
We will use Figure 6.7a, Figure 6.7b, and Figure 6.7c to observe the response time of LMS, RLS, and PLL methods respectively. When we analyze them together with Figure 6.5, it can be seen that the fastest methods are LMS with 9.4 ms. and RLS with 10.1 ms. The proposed method is slightly slower than these methods with a response time of 13.5 ms. On the other



(a)



(b)



(c)

Figure 6.6. Obtained error, cancellation and vibration signals for (a) LMS, (b) RLS, and (c) PLL methods.

Table 6.1. The vibration suppression comparison of the electrical experiment.

Method	Proposed	LMS	RLS	PLL
Suppression Level (dB)	34.98	27.05	23.86	14.62

hand, PLL has the worst response time among them with 82.1 ms.

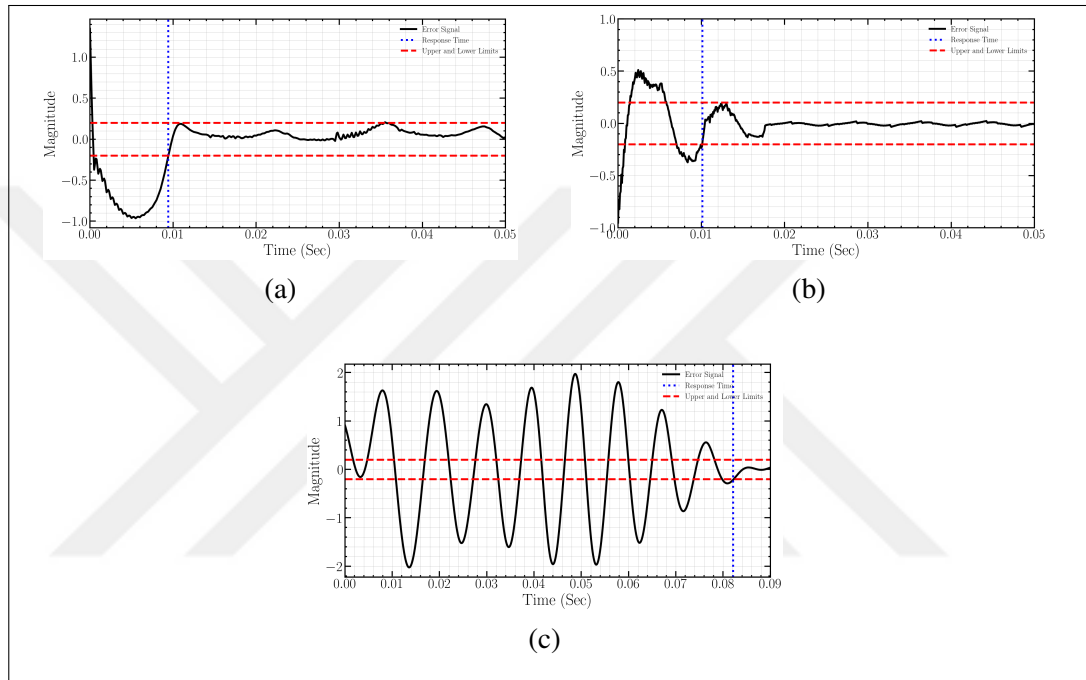


Figure 6.7. The response time of (a) LMS, (b) RLS, and (c) PLL methods for the electrical experiment.

6.2.2. Frequency Sweep

In this section, the proposed and competitive active vibration cancellation methods are performed on an electrical setup for a frequency sweep signal. Then they are compared in terms of vibration suppression and speed.

6.2.2.1. Proposed Method

Here, a frequency sweep signal with parameters f_{start} is 20 Hz, f_{stop} is 120 Hz, the duration of the sweep cycle is 6 seconds and A is 1.0 V. around 1.5 V. offset. is applied to the proposed algorithm. This signal is sampled by the ADC module with a sampling frequency of 10 kHz

and the cancellation signal is generated by the proposed method. The results can be observed on the oscilloscope as shown in Figure 6.8. The yellow signal at the bottom is the vibration signal, the green signal in the middle is the cancellation signal and the red signal at the top is the error signal.

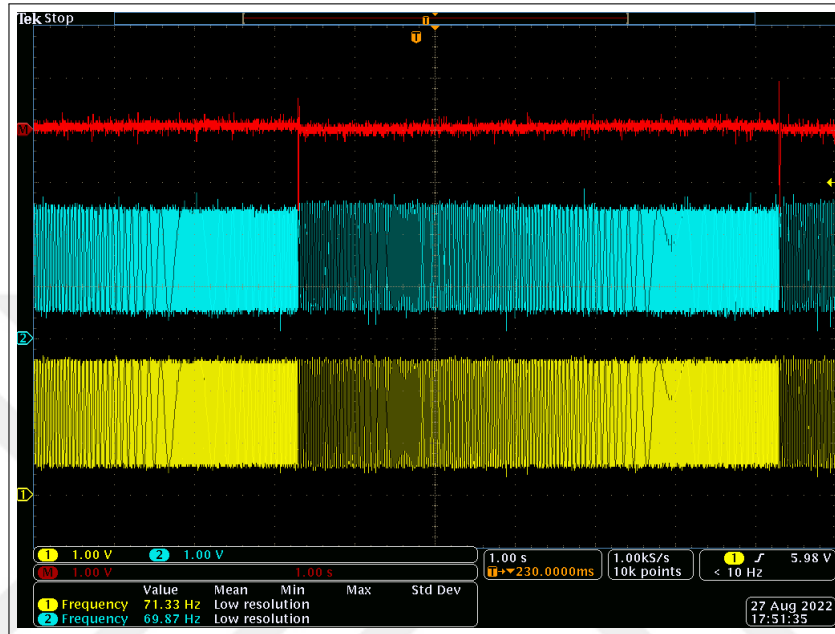


Figure 6.8. Obtained error, cancellation and vibration signals for the proposed method.

The FFT results of the vibration and error signals given in Figure 6.9 is used to measure the performance of the proposed method. Here, the average vibration suppression level is calculated as 26.26 dB between the sweep frequency range.

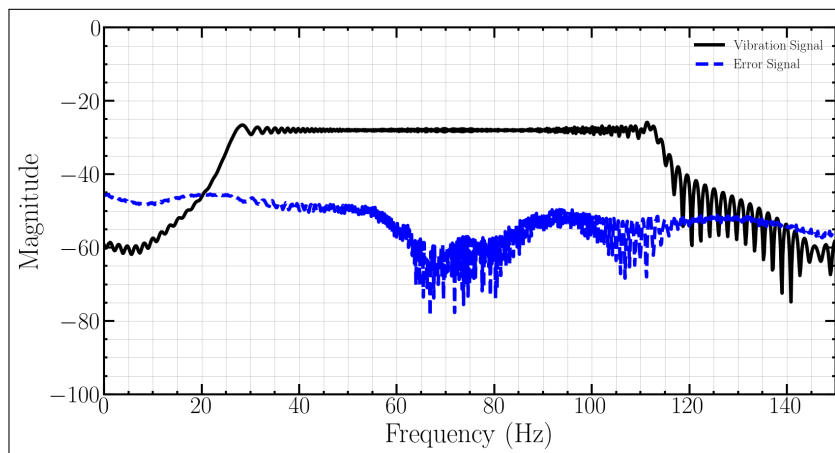


Figure 6.9. The FFT results of the vibration and error signals for the proposed method.

To calculate the reaction time of the proposed method, we use the error signal given in

Figure 6.10. It can be seen that it is measured as 24.8 ms.

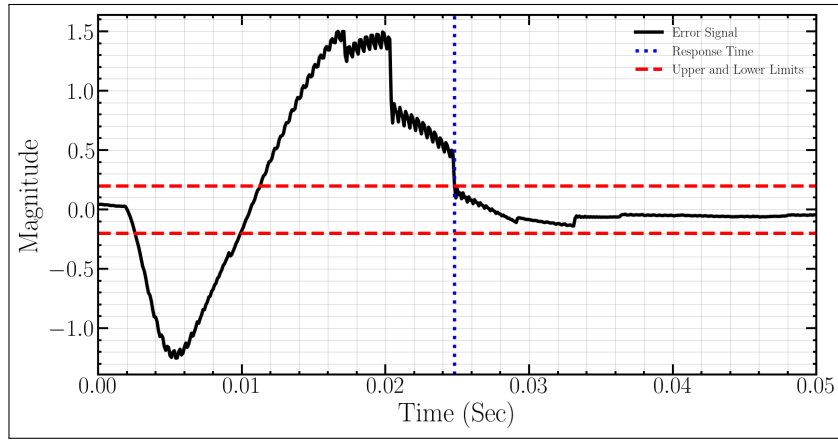


Figure 6.10. The response time of the proposed method.

6.2.2.2. Comparison with Competitive Methods

In this section, we compare the proposed method with competitive methods in terms of vibration suppression level and response time. Each method is simulated using the same vibration signal with parameters f_{start} is 20 Hz, f_{stop} is 120 Hz, the duration of the sweep cycle is 6 seconds, and A is 1.0 V. around 1.5 V. offset. This signal is sampled by the ADC module with a sampling frequency of 10 kHz and the cancellation signal is generated with the selected method.

The results for the LMS, RLS, and PLL methods can be observed on the oscilloscope as shown in Figure 6.11a, Figure 6.11b, and Figure 6.11c respectively. In each figure, the yellow signal at the bottom is the vibration signal, the green signal in the middle is the cancellation signal and the red signal at the top is the error signal.

When we examine the vibration and error signals of each method in the frequency domain, we obtain the vibration suppression results given in Table 6.2. The noise component of the vibration signal generated by the signal generator is very low. Therefore, the vibration suppression level of the proposed method, LMS, and RLS is almost the same. On the other hand, PLL has the worst vibration suppression level among them as expected.

We will use Figure 6.12a, Figure 6.12b, and Figure 6.12c to observe the response time of LMS, RLS, and PLL methods respectively. When we analyze them together with Figure 6.10, it can be seen that the fastest methods are LMS with 10.6 ms. and RLS with 10.7 ms. The

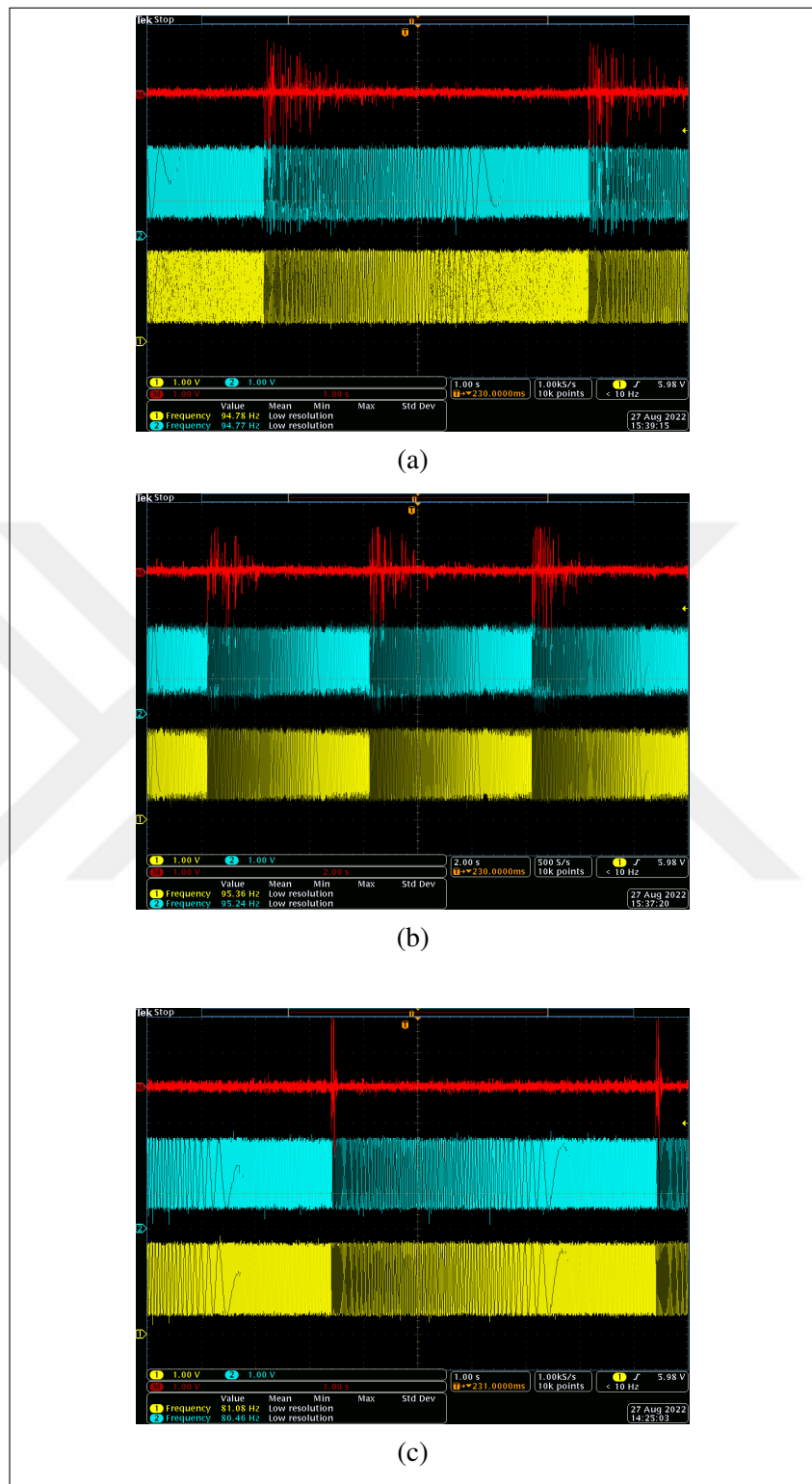


Figure 6.11. Obtained error, cancellation and vibration signals for (a) LMS, (b) RLS, and (c) PLL methods.

Table 6.2. The vibration suppression comparison of the electrical experiment.

Method	Proposed	LMS	RLS	PLL
Suppression Level (dB)	26.26	28.68	26.97	15.12

proposed method is slower than these methods with a response time of 24.8 ms. On the other hand, PLL has the worst response time among them with 117.3 ms.

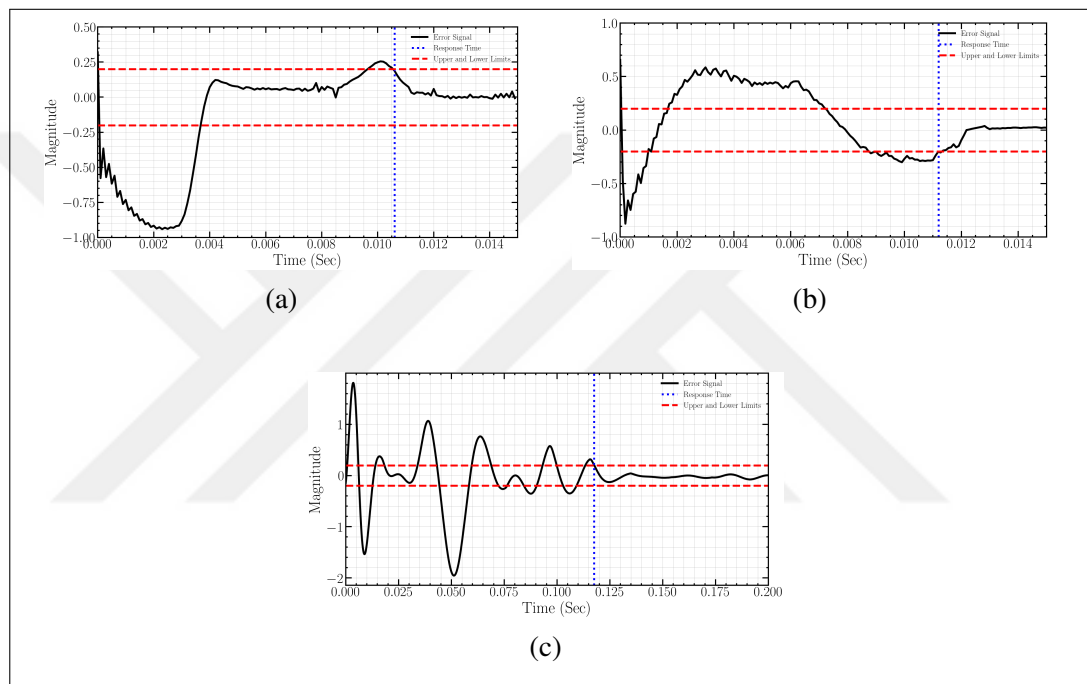


Figure 6.12. The response time of (a) LMS, (b) RLS, and (c) PLL methods for the electrical experiment.

7. EXPERIMENTS ON MECHANICAL SYSTEMS

In this section, we perform mechanical experiments for the proposed real-time active vibration cancellation method as shown in Figure 7.1. Here, the vibration signal of the engine, which normally comes from an accelerometer, is generated by a shaker and obtained from the corresponding accelerometer in real time using the ADC module of the microcontroller. Then the cancellation algorithm is applied to this signal. Finally, the cancellation signal is generated in the microcontroller and the voice coil is controlled according to this signal using the DAC module. Then voice coil vibration is obtained using ADC from the related accelerometer and eliminated vibration is observed.

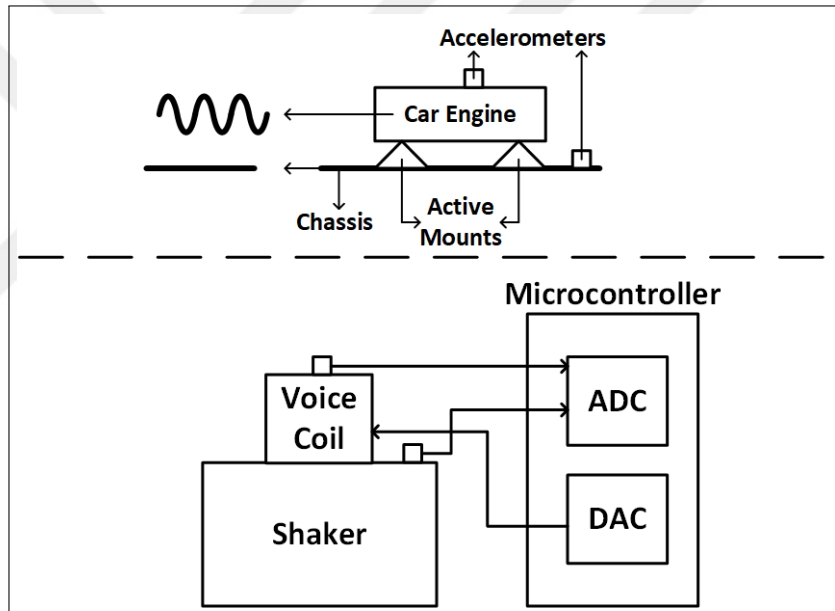


Figure 7.1. The block diagram of the mechanical experiment.

7.1. MECHANICAL SETUP

The general setup used for mechanical experiments is shown in Figure 7.2. Here, a voice coil is used as an actuator, two analog accelerometers are used to measure the vibration signal and observe the error signal, a shaker is used to generate vibration, a motor driver is used to drive the voice coil and an STM32F429 board is used to apply the proposed method to the active vibration cancellation system.

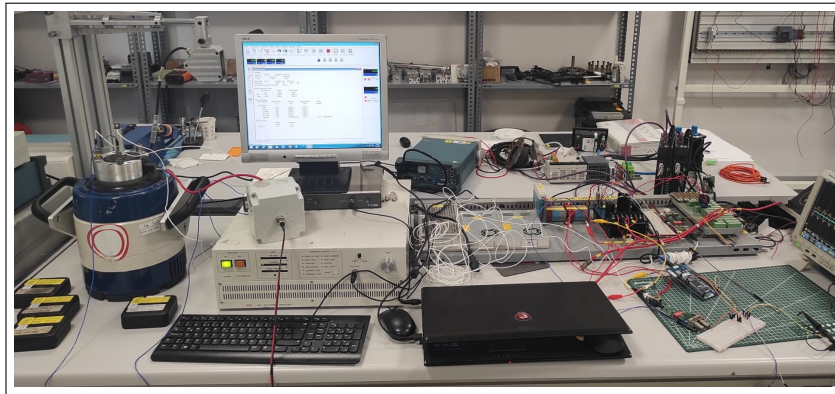


Figure 7.2. The general mechanical test setup.

7.1.1. Actuators

There are 5 different types of actuator for generating sinusoidal vibration and damping signals. These are an eccentric rotating mass (ERM), a linear resonant actuator (LRA), a piezo actuator, a solenoid coil, and a voice coil.

The first three actuators shown in figure 7.3 are used for our basic mechanical construction. An ERM is a DC motor with an eccentric mass that rotates to produce vibrations. The ERM vibrates due to rotational forces, so there is acceleration in two axes. They are cheaper than other actuators and easy to control because they are actually DC motors. So the vibration of the ERM can be controlled with a pulse width modulated (PWM) signal. But an ERM must move an eccentric mass using the current supplied to it, so the frequency and amplitude of the vibration cannot be changed independently. Instead, the frequency at which the eccentric mass rotates correlates linearly with the linear acceleration (force) of the eccentric mass. In addition, the ERM has a longer response time (more than 30 ms).

An LRA is a spring-mass system that oscillates in a linear motion. Inside the LRA, there is a coil suspended by springs and when a voltage is applied, the coil generates a magnetic field. The coil interacts with the magnet and the mass. As the magnetic field varies with the applied control signal, the magnet and mass move up and down, creating a force. Noticeable vibrations only occur at the resonant frequency due to the spring constant. The typical response time for an LRA is about 10-20 ms.

The voice coil inside the LRA is driven by an AC current, and the frequency and amplitude

can be changed independently, unlike the ERM. Although the frequency can be changed, the LRA is usually operated in a narrow frequency range to optimize power consumption at ± 2 Hz, which is around the resonance frequency.

Piezoelectric materials are a type of material that deforms when a voltage is applied, allowing them to produce vibrations in an axis using a variable voltage. Piezo actuators have a faster response time (less than 5 ms), a wide frequency bandwidth, and stronger vibrations compared to ERMs and LRAs. But they are more expensive compared to LRAs and ERMs and require higher voltages to drive.

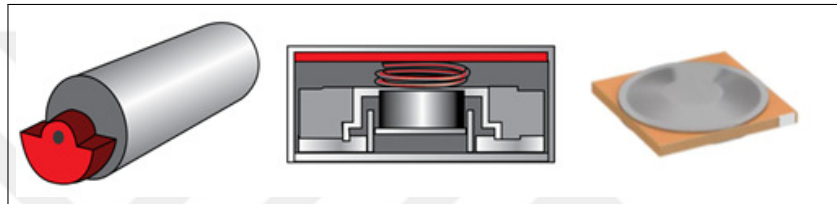


Figure 7.3. ERM [6], LRA [7], and piezo [8] actuators.

The other two actuators shown in Figure 7.4 are used for advanced mechanical setup. Here the solenoid is a cheaper actuator. However, the reverse motion in the solenoid is provided by a spring so it may not produce a suitable sinusoidal motion. A voice coil is much more expensive than a solenoid. But the reverse motion can be produced by changing the direction of the current, so it can easily produce an oscillating movement with the desired force.

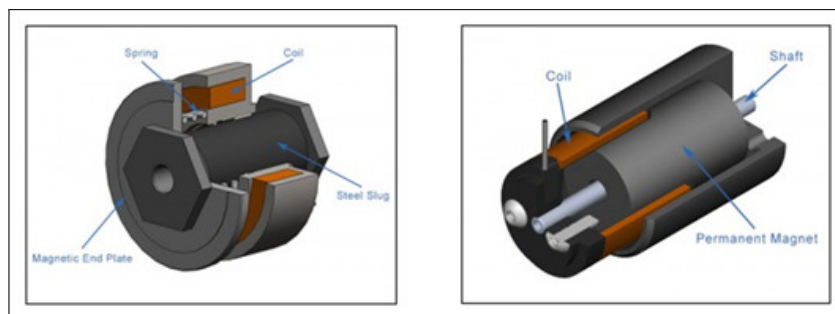


Figure 7.4. The solenoid and voice coil actuators [9].

The actuator used for the mechanical experiments is shown in Figure 7.5. It is a special voice coil that is part of an active hydra mount used for active vibration cancellation in cars.

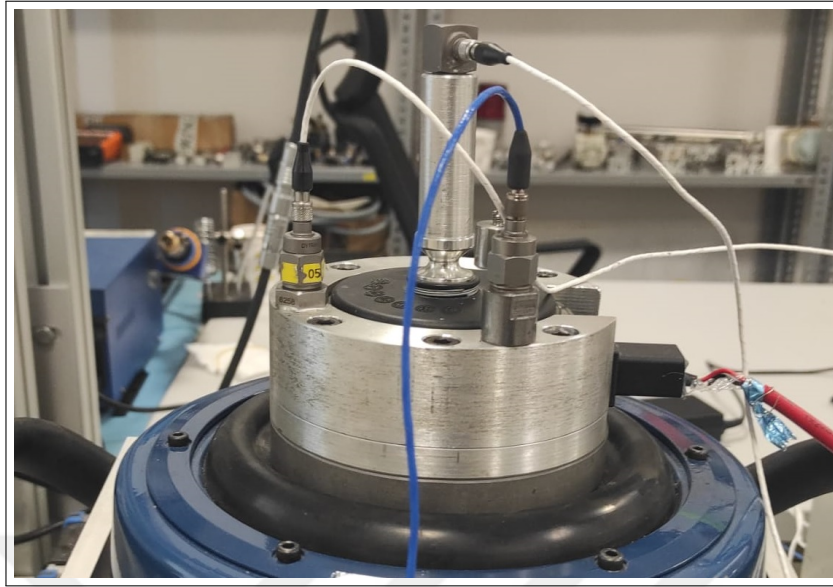


Figure 7.5. The voice coil used for the mechanical experiment.

7.1.1.1. Accelerometers

An accelerometer is a device that measures proper acceleration and has many uses in industry and science. In active vibration cancellation, they are used to separately detect the engine vibration and the damping signal. There are two different types of accelerometers, which are divided into digital and analog accelerometers. Digital accelerometers are usually MEMS-based accelerometers that provide a digital output signal corresponding to the acceleration. Their noise immunity is high, but their sensitivity is lower than analog accelerometers. Analog accelerometers, on the other hand, offer higher sensitivity and are preferred for vibration suppression applications in vehicles.

The accelerometer used for the mechanical experiments is shown in Figure 7.6. It is an analog accelerometer made by Dytran Instruments. There are different types of accelerometers with sensitivity ranges between 1 mV/G and 500 mV/G. In the mechanical setup, a 100 mV/G accelerometer is used to measure the vibration signal and a 500 mV/G accelerometer is used to observe the cancellation signal.

7.1.1.2. Vibration Generator

In the mechanical setup, a vibration generator is needed to simulate the vibrations of a car engine. A shaker shown in Figure 7.7 is used for this purpose.



Figure 7.6. The accelerometer used for the mechanical experiment.

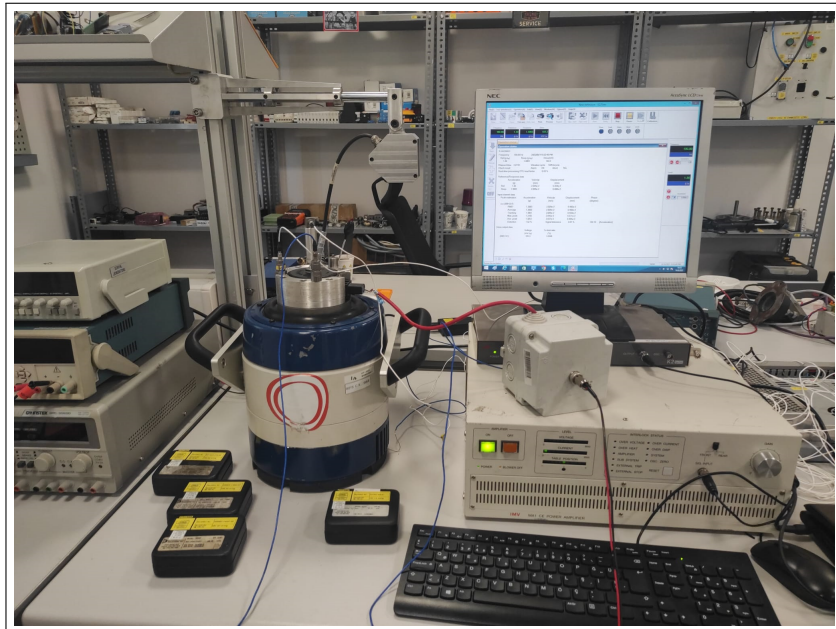


Figure 7.7. The shaker used for the mechanical experiment.

It is a compact shaker system manufactured by IMV Corporation with the model designation m060/MA1-CE. This compact system contains a main shaker, a power amplifier, and a vibration controller. The IMV compact shaker (m-series) uses a permanent magnet for the magnetic circuit and can generate vibration frequencies up to 3 kHz. To increase the flexibility of the system expansion, a DC cooling fan is built into the shaker.

7.1.1.3. Motor Driver

The cancellation signal generated by the DAC module of the STM32F429 board is not powerful enough to drive the voice coil. Therefore, a motor driver is needed to amplify this signal and feed it to the voice coil. In this mechanical setup, the motor driver shown in Figure 7.8 is used for this purpose.

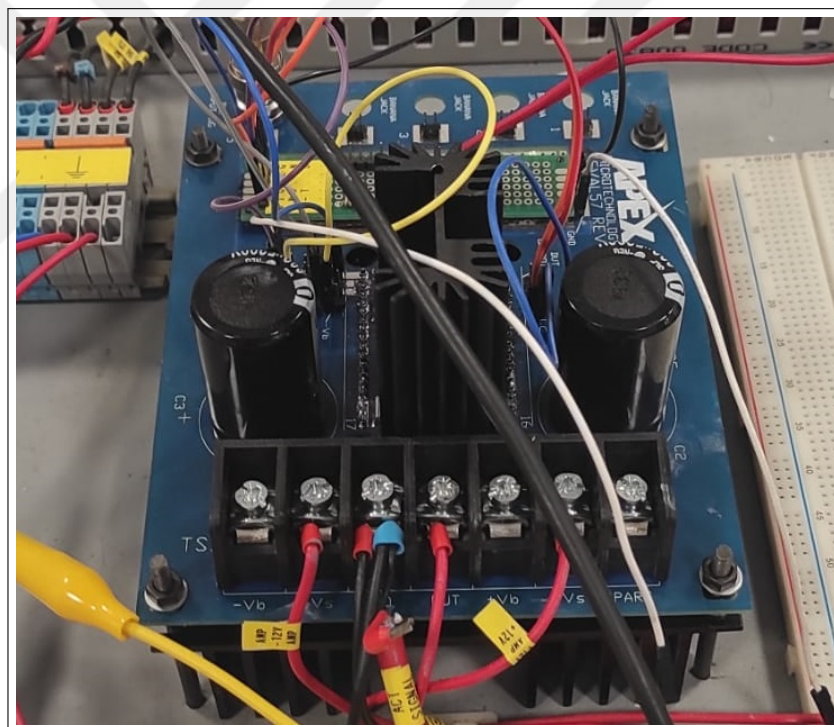


Figure 7.8. The motor driver used for the mechanical experiment.

It is the EK57 Evaluation kit from Apex Microtechnology. It contains the MP111 power operational amplifier, which can produce an output signal of up to 100 V voltage and 15 A current. This evaluation kit has special connectors to easily connect the necessary signal cables and it also has a built-in heat sink for high current values.

7.2. SYSTEM IDENTIFICATION FOR VOICE COIL

The proposed mechanical system has two additional delay factors for real-time implementation. The block diagram of the mechanical setup with these transfer functions is given in Figure 7.9.

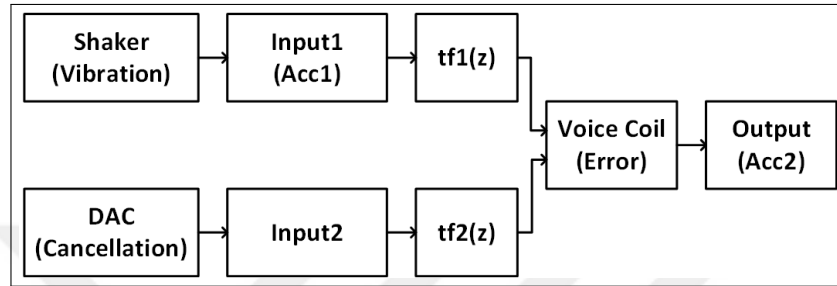


Figure 7.9. The block diagram of the real-time implementation.

The first transfer function is obtained from the relationship between the vibration generated by the shaker and its projection on the voice coil. In the proposed method, we get the vibrations from the accelerometer on the shaker to cancel the vibrations on the voice coil. So we need to establish a relationship between these two signals. To do this, we apply a sweep signal of 10 Hz to 150 Hz to the shaker for one minute. The vibration signals measured by the accelerometers connected to the shaker and the voice coil are given in Figure 7.10a and Figure 7.10b respectively.

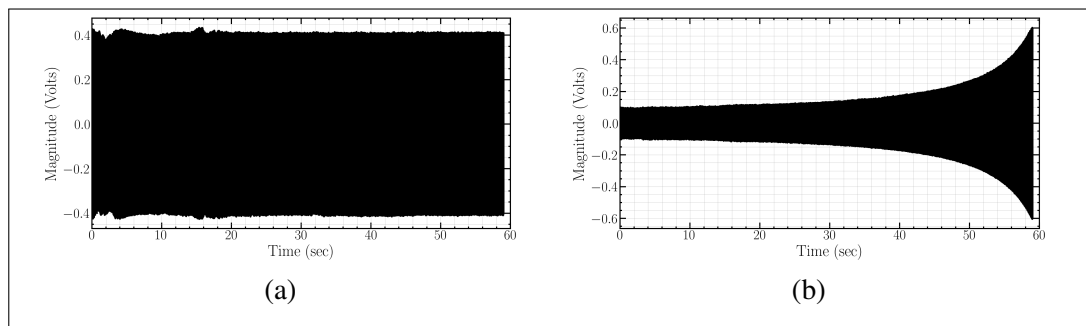


Figure 7.10. The vibration signal obtained from the accelerometer connected to (a) the shaker, and (b) the voice coil.

We can obtain the transfer function between these two signals using MATLAB's system

identification toolbox with 96.76% fit as follows.

$$tf1[z] = \frac{0.0014}{z^2 - 1.9885z + 0.995} \quad (7.1)$$

The magnitude and phase responses of the obtained transfer function are given in Figure 7.11a and Figure 7.11b respectively. It can be seen that the magnitude response peaks at about 130 Hz and the phase drops off rapidly. This point is the resonant frequency of the shaker. However, the frequency range we are interested in is between 20 and 120 Hz. Therefore, it has no influence on the implementation of the system.

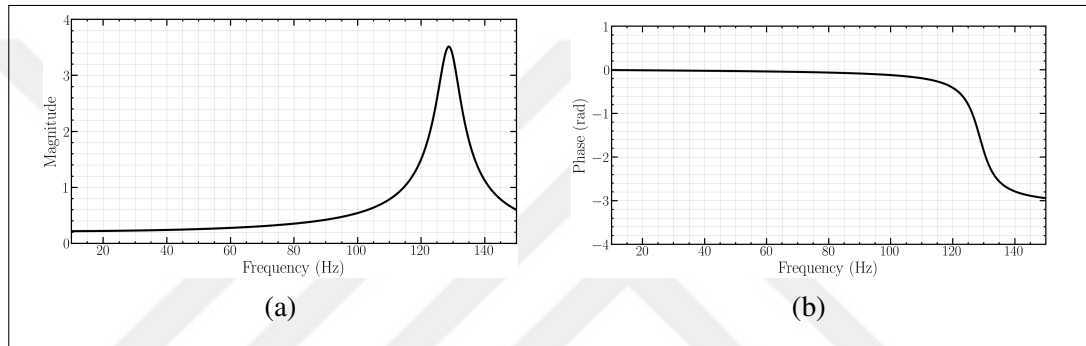


Figure 7.11. The frequency response of the obtained transfer function. (a) Magnitude, and (b) Phase.

The second transfer function is obtained using the input-output relationship of the voice coil itself. When we apply a voltage to the voice coil, a cancellation signal is generated based on the transfer function of this relationship. Therefore, we apply a sweep signal of 10 Hz to 150 Hz to the voice coil for 30 seconds. This input signal and the vibration signal of the voice coil measured by the accelerometer are given in Figure 7.12a and Figure 7.12b respectively.

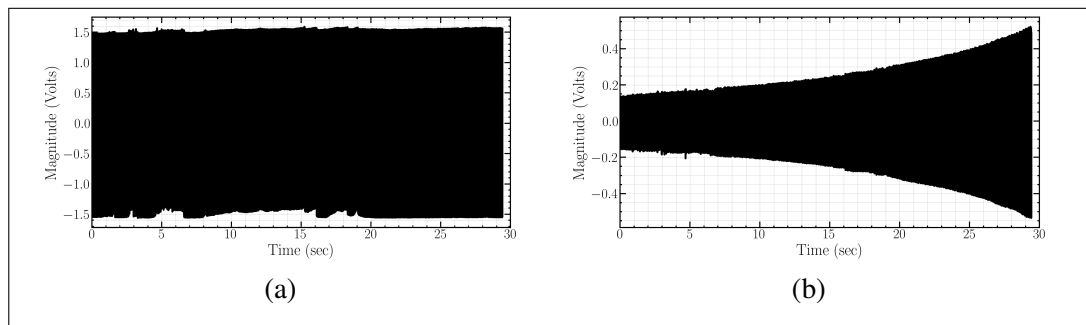


Figure 7.12. (a) The input voltage signal applied to the shaker, and (b) the voice coil vibration signal obtained from the accelerometer.

We again use the MATLAB system identification toolbox to find the transfer function between these two signals. The obtained transfer function with 94.33% fit as follows.

$$tf2[z] = \frac{0.0219z^2 - 0.0409z + 0.0189}{z^4 - 1.8403z^3 - 0.0504z^2 + 1.6418z - 0.7484} \quad (7.2)$$

The magnitude and phase responses of the obtained transfer function are given in Figure 7.13a and Figure 7.13b respectively.

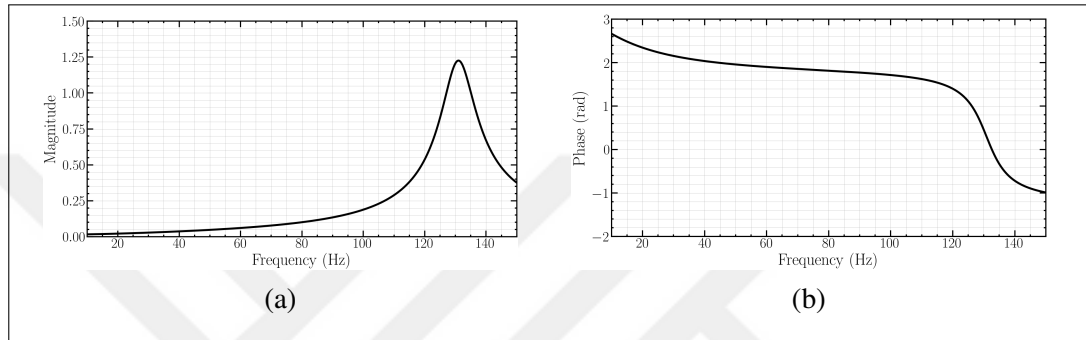


Figure 7.13. The frequency response of the obtained transfer function. (a) Magnitude, and (b) Phase.

Using these two transfer functions, we can implement an additional delay and amplitude gain changes. For the estimated frequency, the additional delay time resulting from these two transfer functions will be added to the estimated phase value. We will also calculate the amplitude gain for the estimated frequency and multiply the estimated amplitude value by this factor.

7.3. EXPERIMENTS

7.3.1. Single-Frequency Signal

In this section, the proposed and competitive active vibration cancellation methods are performed a mechanical setup for a sinusoidal vibration signal with constant frequency. They are then compared in terms of vibration suppression and speed. Here, a sinusoidal vibration signal with a constant frequency of 100 Hz and an amplitude of 0.75 g is generated by the shaker. The generated vibration signal is measured by an accelerometer connected to the base of the shaker. This signal is sampled by the ADC module with a sampling frequency of

10 kHz.

7.3.1.1. Proposed Method

Here, the proposed method is used the vibration signal obtained from the accelerometer to create the cancellation signal. The results can be observed on the oscilloscope when the proposed active vibration cancellation method is turned off and turned on as in Figure 7.14a and Figure 7.14b, respectively. The green signals at the top are the error signals obtained from the accelerometer connected to the voice coil and the yellow signals at the bottom are the vibration signals obtained from the accelerometer connected to the shaker base.



Figure 7.14. Obtained error and vibration signals when the proposed method (a) is not activated, and (b) is activated.

We examine the obtained vibration and error signals in the frequency domain to measure the performance of the proposed method on mechanical systems. The FFT results of these signals are shown in Figure 7.15. Here, the vibration suppression level is calculated as 41.3 dB at 100 Hz.

We use the initial response of the error signal given in Figure 7.16 to calculate the response time of the proposed method on the mechanical setup. It can be seen that it is measured as 16.1 ms.

7.3.1.2. Comparison with Competitive Methods

In this section, we compare the proposed method with PLL in terms of vibration suppression level and response time. The results when the PLL-based active vibration cancellation method is turned off and turned on are given in Figure 7.17a and Figure 7.17b, respectively.

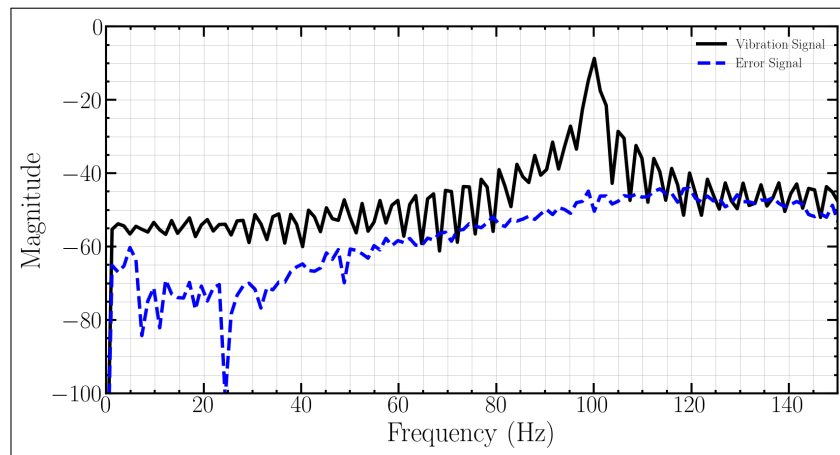


Figure 7.15. The FFT results of the vibration and error signals for the proposed method.

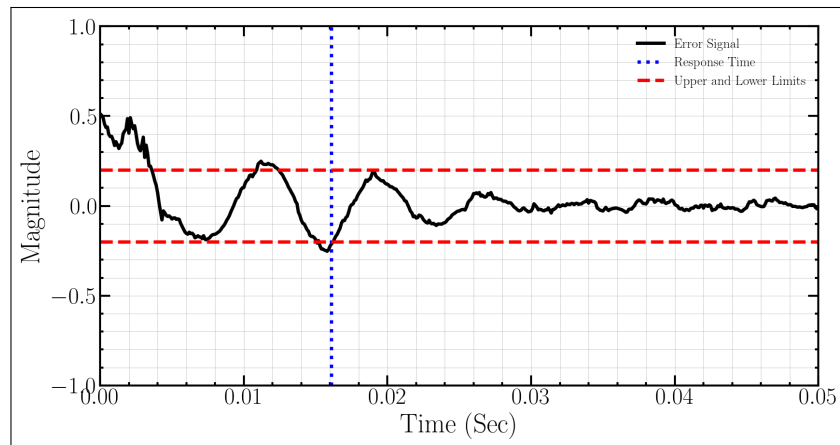


Figure 7.16. The response time of the proposed method.

The green signals at the top are the error signals obtained from the accelerometer connected to the voice coil and the yellow signals at the bottom are the vibration signals obtained from the accelerometer connected to the shaker base.



Figure 7.17. Obtained error and vibration signals when the PLL-based method (a) is not activated, and (b) is activated.

When we examine the vibration and error signals of the PLL-based method in the frequency domain, we obtain the vibration suppression results given in Table 7.1. It can be seen that the proposed method gives better results than the PLL-based method.

Table 7.1. The vibration suppression comparison of the mechanical experiment for constant frequency.

Method	Proposed	PLL
Suppression Level (dB)	41.3	17.7

We will use Figure 7.18 to observe the response time of the PLL-based active vibration cancellation method. When we analyze it together with Figure 7.16, it can be seen that the proposed method has a better response time than the PLL-based method, which has a response time of 91.0 ms.

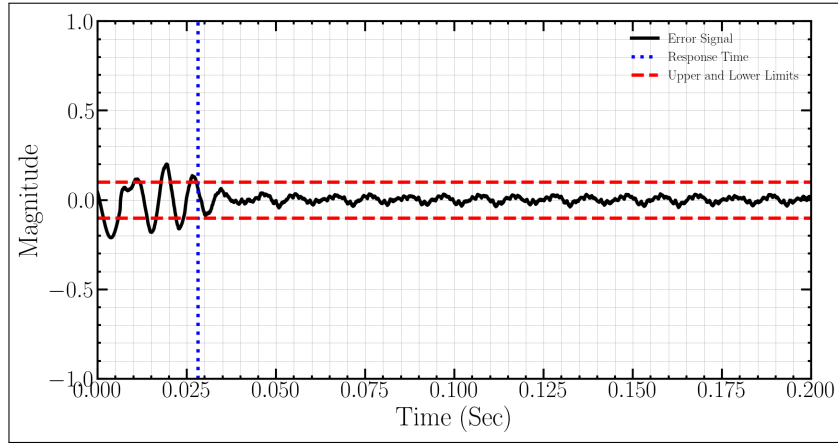


Figure 7.18. The response time of the PLL-based method.

7.3.2. Frequency Sweep

7.3.2.1. Proposed Method

In this section, the proposed method is performed on mechanical setup for frequency sweep vibration signal. Here, a frequency sweep signal with f_{start} is 40 Hz, f_{stop} is 120 Hz, a sweep cycle duration of 6 seconds and 0.75 g amplitude is generated by the shaker. The generated vibration signal is obtained from an accelerometer connected to the base of the shaker. This signal is sampled by the ADC module with a sampling frequency of 10 kHz and the proposed method is applied to this signal to generate a cancellation signal. The results can be observed on the oscilloscope when the proposed active vibration cancellation method is turned off and turned on as in Figure 7.19a and Figure 7.19b, respectively. The green signals at the top are the error signals obtained from the accelerometer connected to the voice coil, and the yellow signals at the bottom are the vibration signals obtained from the accelerometer connected to the shaker base.

We examine the obtained vibration and error signals in the frequency domain to measure the performance of the proposed method on the mechanical system when a frequency sweep signal is applied. The FFT results of these signals are shown in Figure 7.20. Here, the average vibration suppression level is calculated as 27.6 dB.

We use the initial response of the error signal given in Figure 7.21 to calculate the response time of the proposed method on the mechanical setup. It can be seen that it is measured as

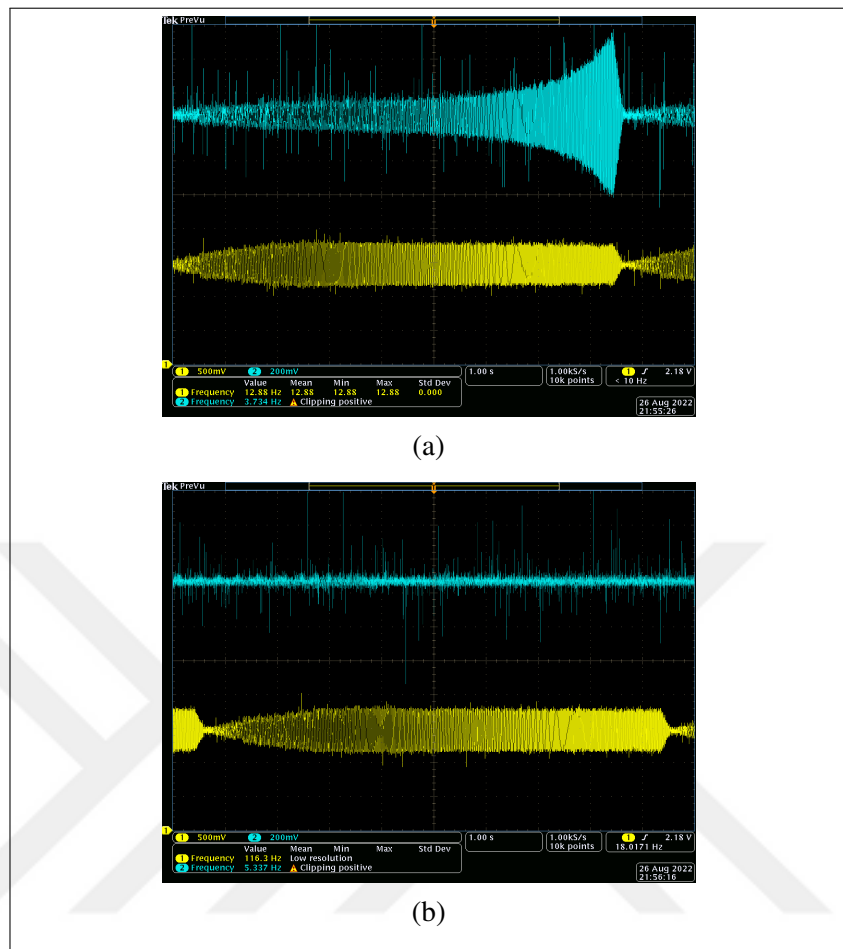


Figure 7.19. Obtained error and vibration signals when the proposed method (a) is not activated, and (b) is activated.

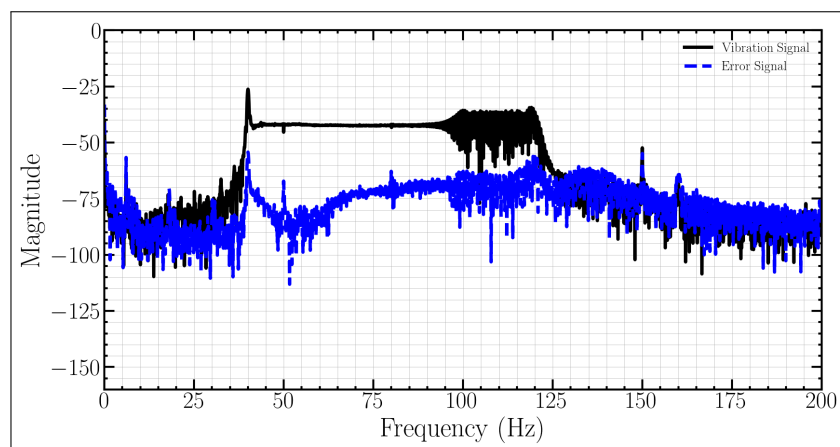


Figure 7.20. The FFT results of the vibration and error signals for the proposed method.

26.6 ms.

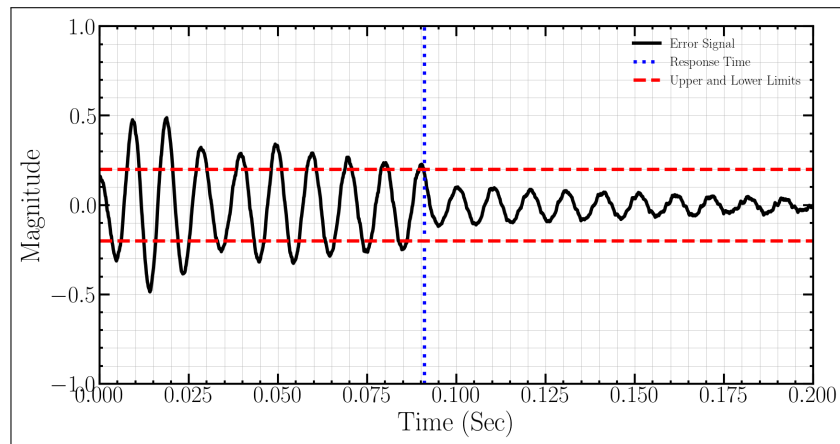


Figure 7.21. The response time of the proposed method.

8. CONCLUSIONS

The primary objective of the dissertation was to develop and analyze an active vibration cancellation method for an active engine mount system that can be implemented on a simple microcontroller. When we examined the system in detail, we stated that the vibrations created by the vehicle engine have sinusoidal characteristics with negligible harmonics between the 20 Hz to 120 Hz frequency range. We also stated that the system may be affected by various uncertainties such as noises caused by measurement errors or the distance between the controller and accelerometers. The condition of the roads may also produce additional disturbances to the system. Finally, we specified that these uncertainties have a probability density function equal to approximate Gaussian distribution.

Then we reviewed the related work in literature dealing with active vibration cancellation for an active engine mount system in a broad spectrum. We divided them into 3 groups adaptive, classical and robust control methods. When we examined them in detail, we stated that adaptive control methods have the best performance under low uncertainties. But their performance drop under high uncertainties and they may even become unstable. On the other hand, classical control methods are more robust to uncertainties but they have slow response times. Finally, robust control methods have the best stability under high uncertainties, but they have an average performance for overall conditions. In this study, we proposed a novel method to work under high uncertainties with better performance. The proposed method was based on Kay's sinusoidal parameter estimation algorithm which can be used under high noise levels. But it contains heavy matrix and trigonometric calculations. So the speed and memory of a microcontroller are not suitable for Kay's original parameter estimation algorithm.

In Chapter 2, we improved Kay's method by applying different improvement steps. First, we used FFT to find a crude frequency value. In this step, the Hanning window is applied to the input signal first to decrease spectral leakage. Then, zero padding is applied to increase the number of FFT bins and obtain better FFT results while using lesser input samples to decrease execution time. Second, we used search algorithms to find exact frequency estimation around this crude frequency. Here, we proposed four simple search methods as bisection, golden-mean, binary, and secant. When we examined them in detail, it can

be seen that bisection and secant methods contain derivative operations. But the bisection method also uses a basic midpoint approach during the search phase so the secant method gives a better result. On the other hand, the golden-mean search method does not contain any derivative operation but two additional points and a specific ratio value are calculated for each iteration. Hence, these bring additional processing load. Finally, the binary search method does not contain any derivative operation, additional point, or ratio calculation. Hence it gives better results than other methods. Then, we improved the search stage further by using predefined look-up tables instead of calculating the frequency estimator at each step to decrease execution time. Then we additionally developed a two-stage search method to further decrease the execution time while reducing memory usage. Finally, we used iterative calculations for computing trigonometric elements of the H matrix to reduce the calculation time.

In Chapter 3 and Chapter 4, we explained the real-time embedded implementation details of the proposed method. First, we compared different Arm-based microcontrollers and selected Arm Cortex-M4 based STM32F429ZI microcontroller for implementation. Then, we compared each improvement step of the proposed method for implementation in terms of execution time and accuracy. As a result, we stated that the proposed method is 580 times faster than Kay's Algorithm with negligible accuracy differences and can be implemented on the microcontroller. But for real-time implementation, we further improved the proposed method. First, we used partial input sampling to decrease execution waiting time. Then, we presented an effective phase delay calculation method that adds the algorithm running time to the phase calculation and gives the resulting phase shift in terms of the number of elements. Finally, we increased the phase estimation accuracy by separating the frequency and phase/amplitude estimation parts and running them with different sampling rates.

Then we performed loopback simulations, electrical experiments, and mechanical experiments on the proposed method and the state of the art in literature. Each experiment was performed using single frequency and frequency sweep signals separately to observe different operating conditions of the engine. The results demonstrated that the proposed method gives similar results to adaptive methods under low uncertainties and provides better performance under moderate to high uncertainties with appropriate response times. We also compared our result with the study of Ranglin [55]. It has been seen that the proposed method

provides better performance than adaptive control methods such as LMS and RLS, classical control methods such as PID and LQR, and robust control methods such as H_2 and H_∞ when there are no uncertainties affecting the system. Only the FxLMS method gives better results than the proposed method. But as stated by Fakhari [2], the performance of adaptive methods drops and they may even become unstable under high uncertainties. We also stated that the proposed method provides better results under moderate to high uncertainties with the experiments.

The following recommendations are made as a result of this research. First, the basic multiple-frequency experiments proposed in Section 5.3 can be further improved by using more powerful microcontrollers. Here, further harmonic numbers can be estimated in real-time embedded implementation using the proposed method. Also, more complex signals can be broken down into sinusoidal components and the proposed method can be applied to them using FPGA. Finally, mechanical experiment comparison can be performed for more diverse methods in the literature.

REFERENCES

1. Romling S, Vollmann S, Kolkhorst T. Active Engine Mount System In The New Audi S8. *MTZ worldwide*. 2013;74(1):34—38.
2. Fakhari V, Ohadi A, Talebi HA. A robust adaptive control scheme for an active mount using a dynamic engine model. *Journal of Vibration and Control*. 2015;21(11):2223—2245.
3. Discovery Kit with STM32F429ZI MCU [cited 2016 20 May]. Available from: <https://www.st.com/en/evaluation-tools/32f429idiscovery.html>.
4. Discovery kit with STM32F746NG MCU [cited 2016 17 May]. Available from: <https://www.st.com/en/evaluation-tools/32f746gdiscovery.html>.
5. STM32 Nucleo-144 development board with STM32H743ZI MCU [cited 2017 11 March]. Available from: <https://www.st.com/en/evaluation-tools/nucleo-h743zi.html>.
6. Eccentric Rotating Mass [cited 2015 21 July]. Available from: <https://engineering.stackexchange.com/questions/3644/what-is-this-small-rotating-part-from-a-cell-phone>.
7. Haptics Technology: Picking Up Good Vibrations: Page 3 of 6 [cited 2011 25 July]. Available from: <https://engineering.stackexchange.com/questions/3644/what-is-this-small-rotating-part-from-a-cell-phone>.
8. New Piezoelectric Actuators Bring Enhanced Haptic Feedback to Portable Touchscreen Designs [cited 2020 13 February]. Available from: <https://www.digikey.com/en/articles/new-piezoelectric-actuators-bring-enhanced-haptic-feedback>.
9. Voice Coil Actuators vs. Solenoids: What is the difference [cited 2015 28 October]. Available from: <https://www.aerospacemanufacturinganddesign.com/article/coil-actuators-vs-solenoids-manufacturing-aerospace-automotive-102815/>.
10. Bigdeli Y, Kim D. Damping effects of the passive control devices on structural vibration control: TMD, TLC and TLCD for varying total masses. *Journal of Civil Engineering*.

2016;20(1):301-8.

11. Alekseev P, Gordeev B, Kiryukhin A, Kostarev S, Sinev A, Chistyakov A. Reduction of vibration from subway trains in the environment by a passive track vibration isolation system. *Journal of Machinery Manufacture and Reliability*. 2008;37(4):387-90.
12. Sun W, Gao H, Yao B. Adaptive Robust Vibration Control of Full-Car Active Suspensions With Electrohydraulic Actuators. *Transactions on Control Systems Technology*. 2013;21(6):2417-22.
13. Hongsheng H, Juan W, Suxiang Q, Linfang Q. Active Vibration Control by Piezoelectric Self-sensing Actuator for Beam under a Moving Mass. *8th International Conference on Electronic Measurement and Instruments*. IEEE. 2007:600-5.
14. Gai Y, Huiying L, Guo Q, Dong S. Study on active vibration isolation system for the bed of ultra-precision turning machine. *International Technology and Innovation Conference (ITIC 2006)*. IEEE. 2006:960-4.
15. Ting S, Dongfang Z, Liang H, Chunyan Z, Fei H. The distributed active vibration suppression program for great flexible spacecraft based on aptitude material. *34th Chinese Control Conference (CCC)*. IEEE. 2015:126-30.
16. Kang CI, Kim CH. An adaptive notch filter for suppressing mechanical resonance in high track density disk drives. *Microsystem Technologies*. 2005;11(1):638-52.
17. Abdeljaber O, Avci O, Inman DJ. Active vibration control of flexible cantilever plates using piezoelectric materials and artificial neural networks. *Journal of Sound and Vibration*. 2016;363(1):33-53.
18. Hadi MS, Darus IZ, Ab Talib MH, Yatim HM, Tokhi MO. Vibration suppression of the horizontal flexible plate using proportional–integral–derivative controller tuned by particle swarm optimization. *Journal of Low Frequency Noise, Vibration and Active Control*. 2021;40(3):1540-57.
19. He Y, Fan Y, Tan UX. Real-time Active Vibration Compensation: A Novel Scheme with Adaptive Filter and Forecasting. *International Conference on Manipulation, Automation*

- and Robotics at Small Scales (MARSS)*. IEEE. 2022:126-30.
20. Zhang D, Guan C. Active Vibration Control of Robot Gear System Based on Adaptive Control Algorithm. *Journal of Sensors*. 2022;2022(1):1-8.
 21. Ma B, Qiu Zc, Zhang Xm, Han Jd. Experiments on resonant vibration suppression of a piezoelectric flexible clamped–clamped plate using filtered-U least mean square algorithm. *Journal of Intelligent Material Systems and Structures*. 2016;27(2):166-94.
 22. Alma M, Martinez JJ, Landau ID, Buche G. Design and Tuning of Reduced Order H-Infinity Feedforward Compensators for Active Vibration Control. *Transactions on Control Systems Technology*. 2012;20(2):554-61.
 23. Ting Z, Prakash NP, Horowitz R. Adaptive Feedforward Reference Design for Active Vibration Rejection in Multi-Actuator Hard Disk Drives. *American Control Conference (ACC)*. IEEE. 2022:3783-8.
 24. Mansour H, Arzanpour S, Golnaraghi F. Design of a solenoid valve based active engine mount. *Journal of Vibration and Control*. 2011;18(8):1221-32.
 25. Kraus R, Herold S, Millitzer J. Development of Active Engine Mounts Based on Piezo Actuators. *ATZ worldwide*. 2014;116(1):46—51.
 26. Choi I, Hirata K, Niguchi N. Design and analysis of a linear oscillatory actuator for active control engine mounts. *11th International Symposium on Linear Drives for Industry Applications (LDIA)*. IEEE. 2017:3783-8.
 27. Fursdon P, Harrison A, Stoten D. The design and development of a self-tuning active engine mount. *European Conference on Vehicle Noise and Vibration*. IMechE. 2000:21-32.
 28. Hillis A, Harrison A, Stoten D. A comparison of two adaptive algorithms for the control of active engine mounts. *Journal of Sound and Vibration*. 2013;286(1):37-54.
 29. Mansour H. Design and Development of Active and Semi-Active Engine Mounts. Master Thesis, Simon Fraser University, 2010.
 30. Landau ID, Airimitoiaie TB, Castellanos-Silva A, Constantinescu A. *Adaptive and Robust*

Active Vibration Control Methodology and Tests. Switzerland: Springer International Publishing; 2017.

31. Turnip A, Hong KS, Park S. Modeling of a hydraulic engine mount for active pneumatic engine vibration control using the extended Kalman filter. *Journal of Mechanical Science and Technology*. 2009;23(1):232-9.
32. Tikani R, Vahdatib N, Ziaei-Rada S. Two-mode operation engine mount design for automotive applications. *Journal of Shock and Vibration*. 2012;19(1):1267-80.
33. Camara LG. Computer Vision Tracking Techniques Applied to Vibration Analysis. Master Thesis, King Juan Carlos University, 2016.
34. Qiu Y, Xu W, Li Z. Performance evaluation of a novel intelligent distributed mounting system for marine mechanical equipment. *International Journal of Advanced Robotic Systems*. 2019;1(1):1—12.
35. Ooi LE, Ripin ZM. Optimization of an engine mounting system with consideration of frequencydependent stiffness and loss factor. *Journal of Vibration and Control*. 2016;22(10):2406–2419.
36. Zhou H, Wang ZW. A New Approach for Road-Vehicle Vibration Simulation. *Packaging Technology and Science*. 2018;31(1):246—260.
37. Togashi C, Nakano M, Nagai M. A Study on Active Hydraulic Engine Mount to Reduce Interior Car Noise and Vibration over Wide Frequency Band. *SAE 2011 Noise and Vibration Conference and Exhibition*. SAE International. 2011:1-9.
38. Raoofy A, Fakhari V, Hamedani O. Vibration control of an automotive engine using active mounts. *The Journal of Engine Research*. 2013;30(1):3-14.
39. Vahdati N, Heidari S. DEVELOPMENT OF AN ELECTROMAGNETIC ACTIVE ENGINE MOUNT. *2015 International Mechanical Engineering Congress and Exposition*. ASME. 2015:1-6.
40. Hausberg F, Hecker S, Rupp M. Incorporation of Adaptive Grid-Based Look-Up Tables in Adaptive Feedforward Algorithms for Active Engine Mounts. *12th International*

Symposium on Advanced Vehicle Control. SAE International. 2014:535-40.

41. Hausberg F, Hecker S, Rupp M. Combination of map-based and adaptive feedforward control algorithms for active engine mounts. *Journal of Vibration and Control*. 2017;23(19):3092—3107.
42. Guo R, Wei X, Gao J. Extended filtered-x-least-mean-squares algorithm for an active control engine mount based on acceleration error signal. *Advances in Mechanical Engineering*. 2017;9(9):1—11.
43. Bai MR, Ou KY. Experimental Evaluation of Adaptive Predictive Control for Rotor Vibration Suppression. *Transactions on Control Systems Technology*. 2002;10(6):895-901.
44. Stoten D, Neild S. The error-based minimal control synthesis algorithm with integral action. *Systems and Control Engineering*. 2002;217(1):187—201.
45. Hillis A, Neild S, Stoten D, Harrison J. A minimal controller synthesis algorithm for narrow-band applications. *Systems and Control Engineering*. 2005;219(1):591—607.
46. Darsivan FJ, Faris WF, Martono W. Active engine mounting controller using extended minimal resource allocating networks. *Int J Vehicle Noise and Vibration*. 2008;4(2):150-68.
47. Mahil AM, Shah MA, Faris WF. Active engine mounting controller design using linear quadratic regulator and proportional integral derivative. *Int J Automation and Control*. 2011;5(3):284-97.
48. Guo R, Wei X, Gao J. An Improved PID Controller Based on Particle Swarm Optimization for Active Control Engine Mount. *SAE World Congress Experience*. SAE International. 2017:1-8.
49. Olsson C. Active automotive engine vibration isolation using feedback control. *Journal of Sound and Vibration*. 2006;294(1):162—176.
50. Fakhari V, Ohadi A. Robust control of automotive engine using active engine mount. *Journal of Vibration and Control*. 2012;19(7):1024—1050.

51. Naseri R, Talebi HA, Ohadi A, Fakhari V. A robust active control scheme for automotive engine vibration based on disturbance observer. *ISA Transactions*. 2020;100(1):13—27.
52. Li S, Wan H, Huang Y, Huo W. An Improved Frequency Estimator for Single Sinusoid Signal. *International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE. 2007:1216-9.
53. Kulon J, Zhang L, Yan Y. Maximum Likelihood Estimator, FFT Estimator and Cramer-Rao Bounds of the Laser Doppler Frequency. *International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*. IEEE. 2015:1-6.
54. Singh P, Singhal A, Joshi SD. An efficient ML frequency estimation of a sinusoid using the Secant method. *International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE. 2017:1-5.
55. Ranglin F, Weicun Z, Dongshuai F. Comparative Study of Active Engine Mount Control Methods. *9th International Conference on Modelling, Identification and Control*. IEEE. 2017:600-5.
56. Wu B, Bodson M. A Magnitude/Phase-Locked Loop Approach to Parameter Estimation of Periodic Signals. *Transactions on Automatic Control*. 2003;48(4):612-8.
57. Pin G, Karimi-Ghartemani M, Chen B, Parisini T. Sinusoidal Signal Estimation from a Noisy-Biased Measurement by an Enhanced PLL with Generalized Error Filtering. *53rd IEEE Conference on Decision and Control*. IEEE. 2014:4071-6.
58. Nascimento PS, de Souza HE, Neves FA, Limongi LR. FPGA Implementation of the Generalized Delayed Signal Cancelation—Phase Locked Loop Method for Detecting Harmonic Sequence Components in Three-Phase Signals. *Transactions on Industrial Electronics*. 2013;60(2):645-58.
59. Kubo M, Taniguchi J, Kato M. Howling reduction by analog phase-locked loop and active noise control circuits. *Applied Acoustics*. 2015;87(1):174-82.
60. Bodson M, Jensen J, Douglas S. Active noise control for periodic disturbances. *Transactions on Control Systems Technology*. 2001;9(1):200-5.

61. Daniels B. Analysis and Design of High Order Digital Phase Locked Loops. Ph.D. Thesis, The National University of Ireland, 2008.
62. Nakaji Y, Satoh T, Kimura T, Hamabe Y, Akatsu Y. Development of an active control engine mount system. *International Journal of Vehicle Mechanics and Mobility*. 1999;32(2-3):185-98.
63. Riley B, Bodie M, Hoying J, Majeed K. Active Vibration and Noise Cancellation Control of Four Cylinder Engines—An Application. *Society of Automotive Engineers*. IEEE. 1995:539-48.
64. Leva A, Piroddi L. FPGA-based implementation of high-speed active noise and vibration controllers. *Control Engineering Practice*. 2001;19(8):798-808.
65. Lee Y, Lee C. Dynamic analysis and control of an active engine mount system. *Control Engineering Practice*. 2002;216(1):921-31.
66. Guo1 Y, Lee C. Extended filtered-x-least-mean-squares algorithm for an active control engine mount based on acceleration error signal. *Advances in Mechanical Engineering*. 2017;9(9):1-11.
67. Kay S. *Fundamentals of Statistical Signal Processing: Estimation Theory*. New Jersey: Prentice Hall; 1993.
68. Weisstein, Eric W. "Multiple-Angle Formulas." From MathWorld—A Wolfram Web Resource [cited 2000 28 February]. Available from: <https://mathworld.wolfram.com/Multiple-AngleFormulas.html>.
69. Box G, Muller ME. A Note on the Generation of Random Normal Deviates. *The Annals of Mathematical Statistics*. 1958;29(2):610-1.