PPG HEART RATE DETECTION IN THE PRESENCE OF MOTION ARTIFACTS

A Thesis

by

SAI SRUJAN GUDIBANDI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirement for the degree of

MASTER OF SCIENCE

August 2016

Major Subject: Computer Engineering

ABSTRACT

Peripheral circulation can elicit a lot of relevant diagnostic information like heart rate and blood oxygenation level without the need of any invasive measurements. Photoplethysmographic(PPG) signals are obtained by such non-invasive measurements using pulse oximeters. PPG signals, although non-invasive, come with some inherent problems. In a non-hospital environment, like when using a wearable type of sensor, a measured PPG signal predominantly suffers from motion artifacts. Ambient light conditions, temperature, and respiratory artifacts are a few other noise sources that affect the PPG signals when trying to measure heart rates. Most motion artifacts lie in the same frequency range as that of the required noise free signal. So, simple filtering is unlikely to work. This work explores adaptive filtering techniques that are commonly used for noise removal. The current work also proposes to use a popular active noise cancellation technique combined with adaptive filtering and artificial neural networks to minimize the motion artifacts. Furthermore, the work proposes a wrapper algorithm that covers the deficiency of the other techniques. Finally, this work employs a smart peak identification technique to measure reliable heart rates from the MA reduced signals.

DEDICATION

To my mother, To my father, To my sister for the constant support and inspiration they provide me.

ACKNOWLEDGMENTS

I would like to sincerely thank Dr. Peng Li for being my advisor and guiding me through my thesis work. I would also like to thank Dr. Nancy Amato and Dr. Pierce Cantrell for accepting to be on my thesis committee.

I would like to sincerely acknowledge Dr. Gerard L. Coté, Director of the Center for Remote Health Technologies and Systems, for providing the de-identified PPG data. I would like to thank Dr. Trung Quoc Le for the help he offered with the data.

I would like to thank Mr. Honghuang Lin and Mr. Seungjai Ahn for helping me with the handling of the data.

I would like to thank Dr. Jiang Hu for considering to be on my thesis committee. I would like to thank Ms. Tammy Carda and Ms. Melissa Sheldon for all the help they offered me.

NOMENCLATURE

ANC	Active Noise Cancellation
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
FLANN	Functional Link Artificial Neural Network
FSLMS	Filtered s- Least Means Square
ICA	Independent Component Analysis
I_{inc}	Intensity of incident light
I_{matrix}	Identity matrix
IR	Infra Red
I_R	Intensity of reflected light
I_{trans}	Intensity of transmitted light
LED	Light Emitting Diode
LMS	Least Means Square
MA	Motion Artifact
MSE	Mean Square Error
PCA	Principal Component Analysis
PPG	PhotoPlethysmograph
RLS	Recursive Least Square
SpO_2	Arterial Oxygen Saturation
TAMU	Texas A&M University
WMA	Weighted Moving Average

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

LIST OF TABLES

1 INTRODUCTION TO PULSE OXIMETRY AND PPG

1.1 Pulse Oximetry

Oxygen is probably the most celebrated component of life. The oxygen content in blood is the most sought out measurement by a doctor to gauge a patient's health in some cases. Pulse oximetry literally translates to measuring percent oxygen saturation of hemoglobin using peripheral circulation. Not just primarily for a hospital environment, pulse oximetry is used in various other fields like monitoring the oxygen saturation during a mountain hike or an underwater scuba diving[1],[2]. The reason for the modern uprising of a demand for pulse oximetry is its ability to be accurate with non-invasive measurement techniques.

The percent saturation of oxygen in hemoglobin is determined by partial pressure of oxygen in hemoglobin. The partial pressure of oxygen indicates the health of transfer of oxygen to any cell tissue. Oxyhemoglobin and Deoxyhemoglobin have different light absorption properties. Pulse oximetry exploits this property to make keen observations. Beers-Lambert Law gives an equation that governs the intensity of transmitted light through a substance which is shown in the Eq. (1.1),

$$I_{trans} = I_{inc}e^{-DC\alpha} \tag{1.1}$$

where D is the distance traveled by light in the given substance (hemoglobin in our case), C is the concentration of the solution and α is called the extinction coefficient.

The Fig. 1.1 shows the extinction coefficients of oxy and deoxy hemoglobins. The

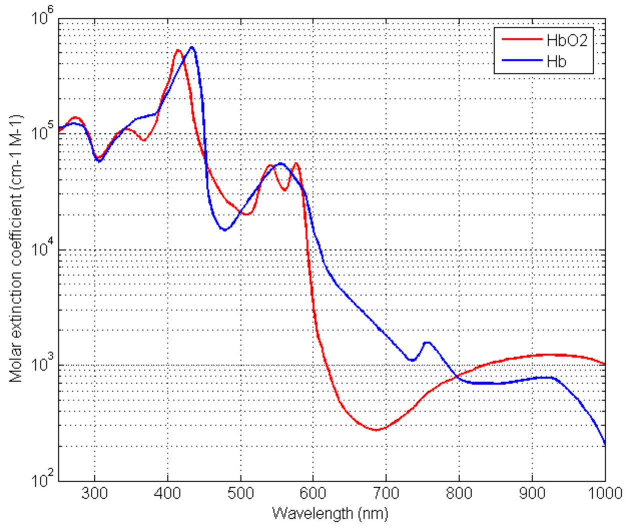

Figure 1.1: Extinction coefficients of hemoglobin and oxyhemoglobin[3]

two wavelengths Red, 660 nm and Infra Red, 940 nm yield the best results. The reason is that there is maximum separation between the two extinction coefficients at those wavelengths. So, if one were to use these wave lengths to transmit light through the pulsatile tissue, we would obtain a signal that is pulsatile and which tracks the heart beat. A signal obtained by using these wavelengths as incident light in a pulse oximeter is called a PhotoPlethysmograph (PPG) signal and appears like Fig. 1.2.

Furthermore, the Oxygen saturation can be calculated by the Eqs. (1.2) and (1.3). The terms AC and DC refer to Alternating and Direct components of a PPG signal which will be discussed in the next section. The Eq. (1.3) is an empirical formulation.

$$R = \frac{AC_{red}/DC_{red}}{AC_{infrared}/DC_{infrared}} \tag{1.2}$$

$$SpO_2 = 110 - 25R \tag{1.3}$$

1.2 Anatomy of a PPG Signal

As shown in Fig. 1.2, a PPG signal consists of a series of crests and troughs that closely follow the heart beat pattern. A slight dip can be observed after the peak which is called the “Dicrotic Notch”. Also, a PPG signal can be broken down into two components *viz.* AC and DC. The pulsatile signal can be regarded as the AC component and the drift/offset from x-axis (or base line) is the DC Component.

A typical PPG sensor or Pulse oximeter has a transmitter or a light source like an LED to shine the tissue. There will be a light detector which can detect light that was either bounced off of or transmitted through the tissue.

PPG signals are commonly obtained from clip type or ring type sensors. A clip type sensor usually clips on to the tip of a finger or the tip of an earlobe. The reason for selecting these tissues is that they have very high density of blood capillaries and the tissue at those locations will not present many involuntary movements. On the other hand, a ring type of sensors goes onto the finger as a ring.

There are also two types of PPG sensors *viz.* Transmission type and Reflectance type. In transmission type, the LED that incidents the light and the photo detector will be on the opposite sides of the sensor. In reflectance type of PPG sensors, they both will be on the same side.

The PPG recordings can be affected by respiratory artifacts, motion artifacts, ambient lighting and temperature conditions. Several of these noise sources can be countered with novel designs like limiting ambient light exposure by ring/band type of PPG sensors. The

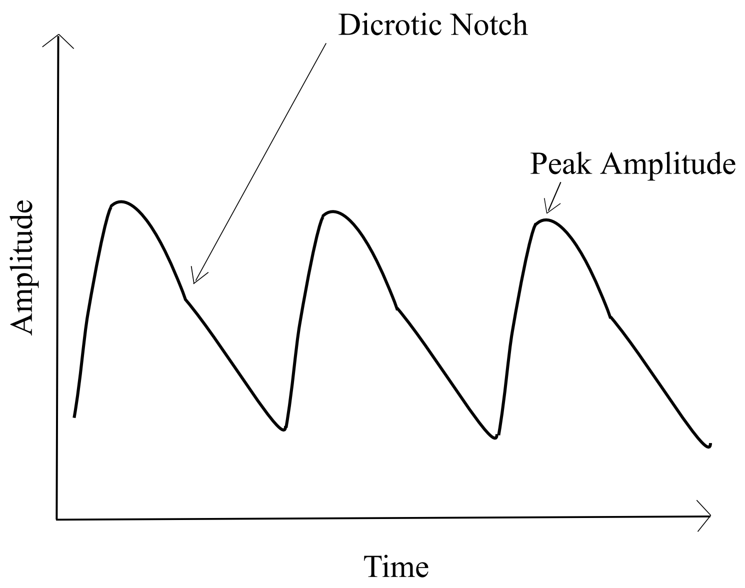

Figure 1.2: A typical resting PPG signal

most common type of noise source is the motion artifact in a non-hospital environment.

2 PROBLEM STATEMENT AND LITERATURE SURVEY

2.1 Problem Statement

Motion Artifacts are the most prominent noise sources in different types of wearable PPG sensors. A frequency domain analysis of PPG data riddled with motion artifacts reveals interesting facts. Most motion artifacts lie in the same frequency range as that of the required uncorrupted signal. This observation is bolstered by Fig. 4.1. When tried to separate noise from raw recordings using direct methods like FFT, we end up with a part of useful signal too. This fact makes it difficult to suppress the noise with simple FIR filters. Adaptive filters are highly useful in separating in-band noises from a signal by employing continuous reference signal comparison and adaptively correcting the signal.

So, the problem at hand can be expanded to be- cleaning the PPG recording to eliminate any motion artifacts using adaptive filtering and then establishing reliable heart rates using a suitable peak identification technique. To ensure that the adaptive filter performs as expected, the reference signal must be golden and be uncorrelated with the noise[4]. Unfortunately, it is difficult to obtain the golden reference for all types of environments. So, the next best option is to find a reliable noise reference signal that can be used to filter out the noise from the recorded signal.

The current work focuses on implementing adaptive filtering techniques like Least Means Squares (LMS) and Recursive Least Squares (RLS) methods to reduce the motion artifacts from the PPG recordings. A popular Active Noise Cancellation technique is proposed to be implemented for the current problem, the details of which are discussed in later sections.

2.2 Literature Survey

2.2.1 Adaptive Filtering

Wearable PPG sensors are the key to the future of remote health monitoring. Evidently, there is a lot of pertinent research to improve the quality and reliability of PPG sensors so that they can be used ubiquitously. The research may not primarily lie in the area of adaptive filtering. One of such techniques was proposed in [5] that explores a new way to acquire pulse oximetry. A logarithmic receiver is to be utilized that may result in removing any kind of probe-coupling artifacts. So, if motion artifact is to be removed deterministically, an additional light source along with the original one is used to generate a control signal to compute the difference.

[6] tried to express motion artifact in a mathematical expression. In doing so, they assumed that the motion artifact only influences venous blood volume and not arterial volume. With this assumption, they arrived at a conclusion that the motion artifact linearly combines with the required PPG signal to result in a raw recording and a simple LMS would prove to be successful. It is later discussed in this work that such an assumption can lead to a trouble in the presence of heavy motion artifacts. An example is recording with hand waving. Although the heart beat will not rise significantly, the motion artifact is so deep and intrusive, we observe that most of the error in heart rate calculation has been

contributed by hand wave activity.

Accelerometer data is used as a noise reference in [1]. The paper explored two types of schemes namely single axis and dual axes stress tests which were proposed in [5].On the other hand, the concept of generating synthetic noise using FFT and Inverse FFT, SVD and ICA was proposed in [4]. They used a variant of LMS called Adaptive Step Size LMS to extract clean PPG from corrupted ones. Kurtosis, a 4^{th} order moment, was used as a measure to select the appropriate synthetic noise.

As discussed earlier in the problem statement, removing in-band noise is inherently difficult. Widrow's Active Noise Cancellation[7] is an intelligent Active/Adaptive Noise Cancellation scheme that is capable of removing in-band noise. Widrow's ANC was utilized by [8],[9],[10] to reduce motion artifacts in PPG using accelerometer data. The authors of [8] also introduced Laguerre Models[11] instead of plain FIR taps/weights. They argued that using Laguerre models result in significantly less time delay that would have been introduced by a high order FIR filter.

Finally, the ANC technique proposed in [12] is one of the simplest yet most effective ANC techniques there are on the market. They developed a faster version of FSLMS in the same paper that makes it fast in terms of computational complexity. This technique is so pertinent to the current problem that developing hardware should not take much of the resources.

2.2.2 Heart Rate Calculations

Heart rates from PPG are calculated with the help of simultaneous ECG (EKG) in [13]. A completely independent and automatic way of calculating reliable heart rates from ECG and PPG waveforms is discussed in [14]. The technique is a combination of median filter and peak identification technique.

3 ADAPTIVE FILTERING ALGORITHMS

3.1 Least Means Square (LMS) Algorithm

LMS filter is the most commonly used classical adaptive filter[15]. A traditional LMS filter is shown in Fig. 3.1. *x(n)* is the input and if *w* represents the weight matrix that defines the adaptive filter shown with the transfer function $\hat{H}(z)$(The time domain impulse response representation is $\hat{h}(n)$). The algorithm tries to optimize the model with updating filter weights according to Eq. (3.1),

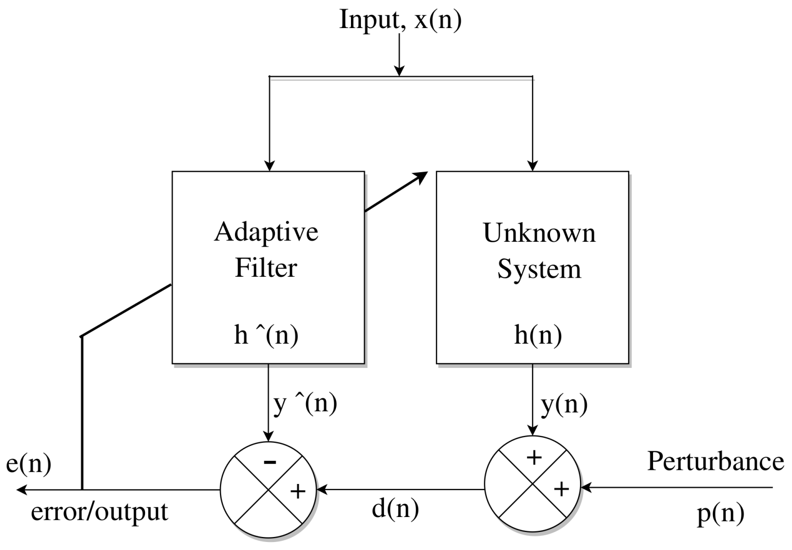

Figure 3.1: A typical LMS filter implementation

$$w(n+1) = w(n) - \mu \nabla \mathscr{C}(n) \tag{3.1}$$

where $\mathscr{C}$ is the Mean Square Error which is obtained by the mean of the square of the difference between expected signal $\hat{y}(n)$ and actual signal $d(n)$. ∇ is the gradient operator. μ is the learning rate which influences the rate at which the algorithm converges.

The negative sign implies that the weight update takes place in opposite direction to that of the MSE gradient slope. This can be understood with intuition. The MSE depends on the future weights by the following relations.

$$\hat{y}(n) = x(n) * \hat{h}(n) \tag{3.2}$$

$$\hat{y}(n) = w^T x \tag{3.3}$$

Thus MSE increases if the current MSE is positive and the future weight has increased from this time step. So, a negative sign checks this possibility. Finally, applying steepest descent method to the cost function described by Eq. (3.4),

$$\mathscr{C} = E[e(n)^2] \tag{3.4}$$

where the error e(n) is given by Eq. (3.5)

$$e(n) = d(n) - \hat{y}(n) = d(n) - w^T(n)x(n) \tag{3.5}$$

The gradient can be written as:

$$\nabla_w \mathscr{C} = \frac{\partial}{\partial w} E\left[e(n)^2\right] \tag{3.6}$$

$$\nabla_w \mathscr{C} = 2E\left[\frac{\partial\left(d(n) - w^T(n)x(n)\right)}{\partial w} e(n)\right] \tag{3.7}$$

$$\nabla_w \mathscr{C} = -2E[x(n)e(n)] \tag{3.8}$$

Finally, the weight update equation will be:

$$w(n+1) = w(n) + 2\mu E[x(n)e(n)] \tag{3.9}$$

3.2 Recursive Least Squares (RLS) Algorithm

Recursive Least Squares method is one of the oldest adaptive filtering techniques which was formulated by Gauss[16]. The notation remains the same as that of LMS and the same Fig. 3.1 can be used as a reference to understand the working of an RLS filter.

If *d(n)* is the actual signal, $\hat{d}(n)$ is the expected signal from the filter and *e(n)* is the error, we can write the cost function as:

$$\mathscr{C}(w_n) = \sum_{i=0}^{n} \lambda^{n-i} e^2(i) \tag{3.10}$$

where λ is called the forgetting factor and it penalizes older samples.

Similar to the LMS derivation, the gradient of the cost function is calculated according to Eq. (3.11),

$$\frac{\partial \mathscr{C}(w_n)}{\partial w_n(k)} = \sum_{i=0}^{n} 2\lambda^{n-i} e(i) \frac{\partial e(i)}{\partial w_n(k)} \tag{3.11}$$

where ‘k’ is the k^{th} coefficient of a p^{th}order filter. Replacing e(n) with the definition similar to Eq. (3.5) and equating the derivative to zero.

$$\sum_{l=0}^{p} w_n(l) \left[\sum_{i=0}^{n} \lambda^{n-i} x(i-l) x(i-k) \right] = \sum_{i=0}^{n} \lambda^{n-i} d(i) x(i-k) \tag{3.12}$$

The above equation can be re-branded as follows:

$$R_x(n) w_n = r_{dx}(n) \tag{3.13}$$

where $R_x(n)$ is the covariance matrix of the input *x(n)* and $r_{dx}(n)$ is the cross-covariance matrix of *d(n)* and *x(n)*. Finally, the optimal filter weights can be found out by Eq. (3.14).

$$w_n = R_x^{-1}(n) r_{dx}(n) \tag{3.14}$$

The name "Recursive" is justified from the fact that we try to use recursion to arrive at the final solution. After simple algebra, it can be shown that:

$$r_{dx}(n) = \lambda r_{dx}(n-1) + d(n)x(n) \tag{3.15}$$

Also

$$R_x(n) = \lambda R_x(n-1) + x(n)x^T(n) \tag{3.16}$$

The Eqs. (3.15) and (3.16) are clearly recursive relations. Although, we use iteration and not recursion when programming the RLS.

4 NOISE MODELS

Every adaptive filtering algorithm needs a good reference signal that has strong correlation with one or more components of the input. We can have a good clean reference signal or a good noise reference signal. The current work concentrates on using a noise reference. Two possible noise reference generation techniques are discussed. Although, the second method is technically not a generation technique, it can be considered a legitimate noise identification.

4.1 Synthetic Noise

4.1.1 Generation

Since the PPG recording has the motion artifacts, it is reasonable to assume that the signal contains the required noise reference signal. Several methods using FFT, ICA, PCA and SVD are discussed in [4]. A perfect noise model can be generated from an FFT method[4]. The following steps are involved in generating the synthetic noise:

1. Compute the FFT/DFT of the raw recording and compile the single sided spectrum from such computation.
2. Set the amplitudes of the components lying in the frequency range of 0.5 - 4 Hz to zero. This ensures that the required component of the recording i.e. the pure PPG signal is taken out of the recording.
3. Compute the inverse FFT of the resultant signal to get the synthetic noise.

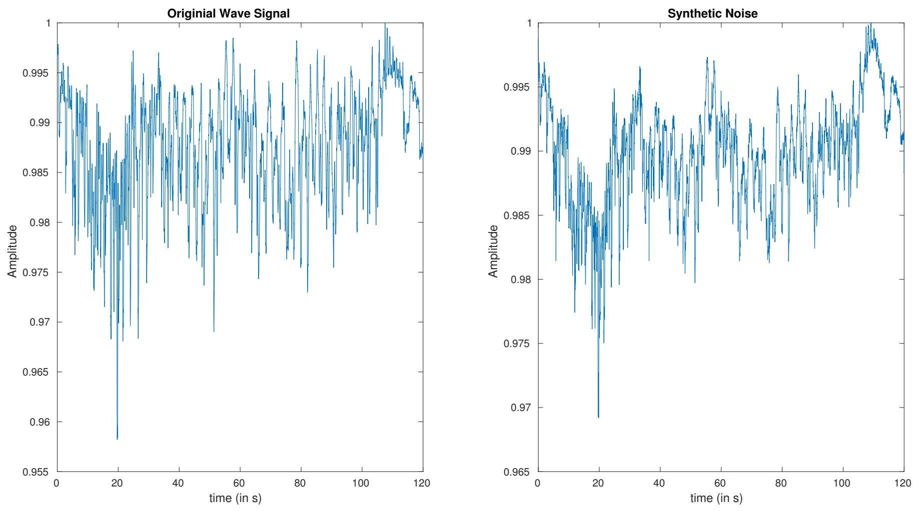

Figure 4.1: Original recording and generated synthetic noise

4.1.2 LMS/RLS Implementation with Synthetic Noise

There must be a way in which the synthetic noise that is generated is incorporated into the application of LMS/RLS algorithms on the recorded PPG. This is demonstrated in Fig. 4.2. The Noise Estimator block is similar to the adaptive filter block in Fig. 3.1 and is responsible for estimating the noise based on the "Noise Estimator" block that uses the PPG recording to extract synthetic noise. After several iterations, theoretically, we should be receiving the required clean signal as shown. This same block diagram can be used to understand the application of RLS with synthetic noise.

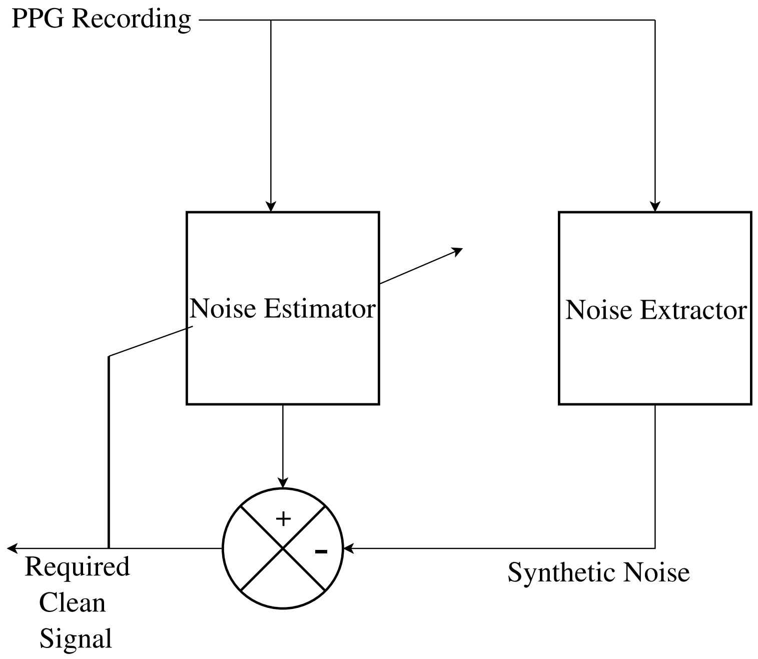

Figure 4.2: An LMS filter with synthetic noise as reference

4.2 Accelerometer Data as Noise

4.2.1 Theory

The aim of this work is to investigate and propose smart methods to filter out motion artifacts. We have seen till now how to create synthetic noise that can be used to clean out the MA interference. Naturally, there arises a question if there are any other alternate methods to obtain noise. Since accelerometers can track the motion of an object to which they are attached, they are naturally assumed as one of the best noise sources[17] for the current problem.

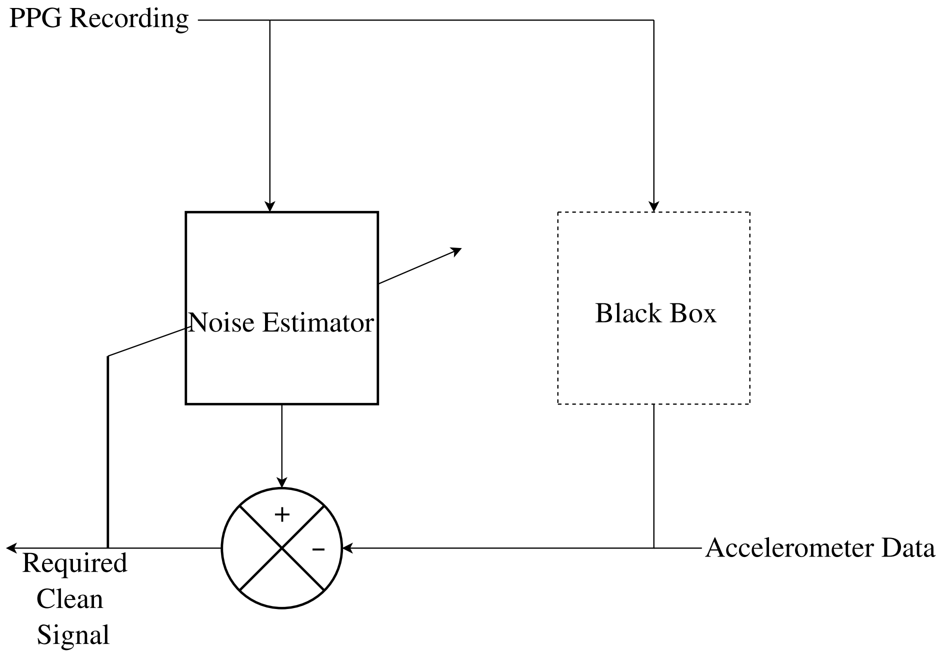

Figure 4.3: An LMS filter with accelerometer noise as reference

4.2.2 LMS/RLS Implementation with Accelerometer Noise

As in the earlier case, we can adopt this into implementation as shown in Fig. 4.3. To fit the LMS algorithm into this model, we assume that there is a black box that can convert the PPG recording into accelerometer data. Although, in reality, no such box exists as the recording and accelerometer signals are acquired independently. But, for the spirit of discussion, we assume that the accelerometer data, which is the noise, has been generated from the PPG recording. All of the notation is similar to that of a normal LMS implementation.

5 ACTIVE NOISE CANCELLATION

5.1 Theory

Noise pollution is very ubiquitous these days. Several noise cancellation techniques have been proposed in the past to address noise pollution. One of such techniques is Active Noise Cancellation (ANC) sometimes referred to as Active Noise Control or Adaptive Noise Cancellation.

In a typical ANC system as shown in Fig. 5.1, there is a path represented by parallel bars. There is a reference microphone that senses/represents the background noise. This microphone feeds a secondary speaker that generates the so-called anti noise to tackle the noise. This forms a static noise controller which will just change the anti noise whenever the noise source changes the pattern. If the noise source has a particular pattern, the system would not identify it on an intelligent level. If the system must work, it has to be adaptive to the changes in the environment and sensitive to changes in the noise source[18]. So, there is an error sensing microphone at the end of the path to use this signal to adaptively adjust the anti noise. Adaptive algorithms like LMS and RLS can be used to achieve such a purpose.

5.2 FLANN

The adaptive filters we have discussed until this point are linear in nature. Most of real world noises are non-linear. So, for better functionality, it is recommended to introduce a

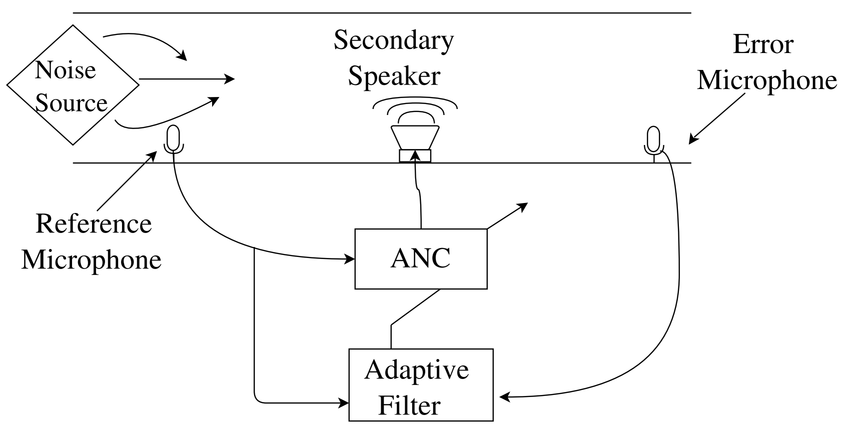

Figure 5.1: A typical active noise cancellation in sound domain

segment that interacts with the non-linear component of the noise. Using a FLANN based Neural Network for input expansion proves to be less computationally complex and a good alternative for non-linear prediction algorithms[12].

A Functional Link Artificial Neural Network (FLANN)[19, 20] is a single layered Artificial Neural Network. Instead of multiple layers, the input data is morphed into a set of linearly independent functions. This way, no new information is added but the dimensionality/representation of input data is enhanced. [20] and [21] report that this functional link simplifies the learning algorithms.

Let x be the input data with a size n, then each element x_i, where $1 \leq i \leq n$, is transformed into $f_j(x_i)$, where $1 \leq j \leq m$, using an appropriate function f. The most popular functions that are sought are power series expansion, trigonometric expansion, tensor product expansion. Legendre polynomials can also be used. The number m depends on the type of expansion we use. The following table demonstrates the way the three expansions work when an input of size 3 is to be used.

Table 5.1: Different functional expansions

Input	Order	Tensor	Power Series	Trigonometric
x_1, x_2, x_3	3	$x_1, x_2, x_3, x_1x_2, x_2x_3, x_1x_3, x_1x_2x_3$	$x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1^3, x_2^3, x_3^3$	$x_1, \sin(\pi x_1), \cos(\pi x_1), \sin(2\pi x_1), \cos(2\pi x_1), \sin(3\pi x_1), \cos(3\pi x_1), x_2, \sin(\pi x_2), \cos(\pi x_2), \sin(2\pi x_2), \cos(2\pi x_2), \sin(3\pi x_2), \cos(3\pi x_2), x_3, \sin(\pi x_3), \cos(\pi x_3), \sin(2\pi x_3), \cos(2\pi x_3), \sin(3\pi x_3), \cos(3\pi x_3)$

The column “order” in Table 5.1 doesn’t apply to Tensor type of expansion. For Tensor type of expansion, if the input size is *n*, we can determine *m* by Eq. (5.1).

$$m = \binom{n}{1} + \binom{n}{2} + ... + 1 \tag{5.1}$$

On the other hand, for trigonometric and power series expansions, if *P* is the order of expansion, then *m* is as given in Eq. (5.2).

$$m = n * (2P + 1) \tag{5.2}$$

A typical FLANN architecture can be seen in Fig. 5.2. This can be considered as a single output neuron type of implementation. If **w** represents the weight matrix, with the same size as that of transformed input **S**, we can write the matrix equation as follows:

$$Sw = y \tag{5.3}$$

$$w = S^{-1}y \tag{5.4}$$

The existence of the solution depends on the matrix **S** and it being non-singular. This would be possible only if all the columns of **S** are linearly independent. This is the reason we try to use functional expansion to make the input more linearly independent.

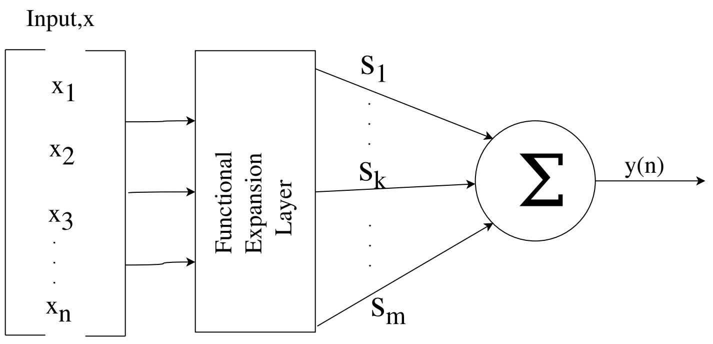

Figure 5.2: A FLANN architecture implementation

5.3 ANC with FLANN and Filtered-s LMS

Since we introduced a component that handles the nonlinearity, we can start discussing the adaptive filtering algorithm in conjunction with FLANN. The ANC setup with FLANN and the FSLMS is shown in Fig. 5.3. *B(z)* is the transfer function of path from reference microphone to the error microphone. *d(n)* represents the noise that is about to be canceled. *A(z)* is what is known as the transfer function of the secondary path. Below is the explanation of how FSLMS fits into the ANC algorithm.

The notations are: *x(n)* is the input noise, *d(n)* is the noise at the point where secondary speaker functions. $\hat{d}(n)$ is the expected anti noise that has been generated from the adaptive filter. *s* is the functionally expanded input matrix. *w* is the weight matrix ranging from 1 to *P* where P is the expansion order. *e(n)* is the error generated from *d(n)* and $\hat{d}(n)$.

As always “n” represents discrete time step. From FLANN, we have:

$$y(n) = \mathbf{w}^T(n)s(n) \tag{5.5}$$

$$e(n) = d(n) - \hat{d}(n) \tag{5.6}$$

$$\hat{d}(n) = a(n) * [\mathbf{w}^T(n)s(n)] \tag{5.7}$$

The boldface ***w*** indicates that it is a vector despite it being at a single time point. The weight update equation can be derived in a similar fashion as that of LMS.

$$\zeta = E[e^2(n)] \tag{5.8}$$

$$\Delta = \frac{\partial \zeta}{\partial \mathbf{w}} = 2e(n)\frac{\partial y(n)}{\partial \mathbf{w}} = 2e(n)s(n) * a(n) \tag{5.9}$$

According to the steepest descent update,

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{\mu}{2}\Delta \tag{5.10}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu e(n)[s(n) * a(n)] \tag{5.11}$$

5.4 ANC for the Current Problem

Even though both the problems at hand are different mathematically, the philosophy of the ANC can be applied to the current problem. *x(n)* will be the accelerometer signal. The signal *d(n)* should be considered the raw recording which is formed via convolution of accelerometer data with the primary path transfer function, *B(z)*. The convolution, *y(n)*a(n)*, is the expected noise that has been generated by the adaptive filter. The error signal from the adaptive filter will be our required signal.

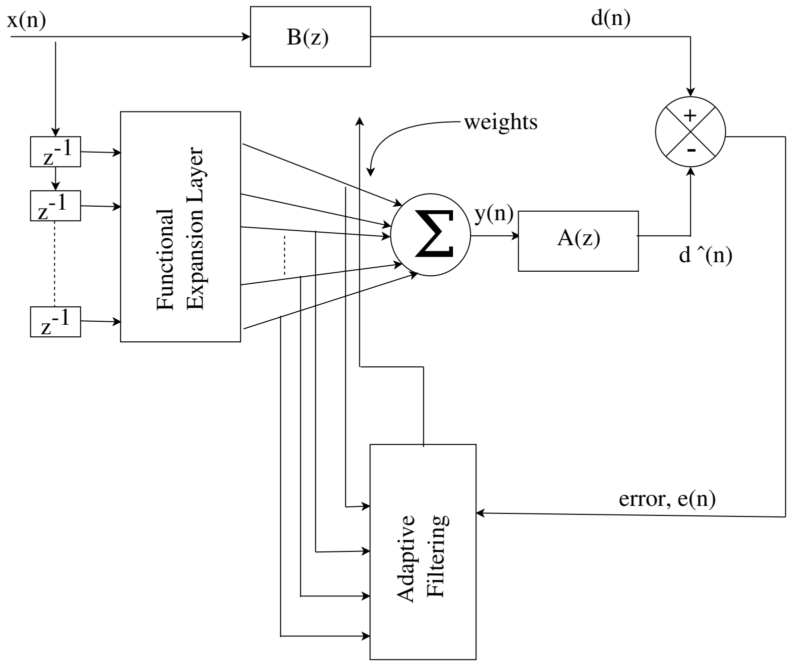

Figure 5.3: The complete ANC setup with FLANN

6 FINAL ALGORITHM AND ADAPIT

6.1 Final and Improved Algorithm

It can be observed from the results and from the discussion in Section 10.2 that the accelerometer noise model does not perform very well for situations where there is not much of motion. If we can detect such a situation, we can use the synthetic noise model to filter out our required signal instead of using accelerometer data. So, a new wrapper algorithm is proposed that will automatically detect if there is a heavy physical activity by measuring the variance of the accelerometer data. The maximum variance encountered until the current time is stored. If the normalized variance of the current activity is greater than 0.33 times the maximum variance until now, we can say that there is a moderate physical activity and then we use the ANC with FSLMS technique. On the other hand, if the normalized variance is not greater than 0.33 times the maximum variance until now, we end up using the synthetic noise model.

6.2 ADAPIT

After cleaning the signal of its motion artifacts, the next step is to automatically calculate the heart rates using an algorithm. Such an algorithm, ADAPIT, was proposed in [14]. A peak detection algorithm should do the trick of identifying heart rates. Each heart beat is marked by a peak in the PPG waveform. The algorithm that is to be deployed must be careful enough not to identify the secondary peak after the dicrotic notch as another heart beat. The following steps are applied to get automatic heart rates from the clean signal:

1. Apply a median filter of window length 550 ms to suppress the dicrotic notches.
2. Select an appropriate processing window size to calculate the heart beats by extrapolation. For example, a 5 second window extrapolates the heart rate in beats per minute by multiplying the number of peaks by 12.
3. Define a threshold T_1 equal to $2\sigma_1$, where σ_1 represents the standard deviation of the original waveform of selected window size.
4. Restrict the original waveform of selected window size to the range $[-T_1, T_1]$. Now, calculate the standard deviation of this waveform as σ_2. Set T_2 to be $3\sigma_2$ to be served as the second (a lower) threshold. We identify all the peaks that are greater than T_2 and mark them as the first estimate of reliable peaks that can determinate the heart rate.
5. To eliminate any existing amplified dicrotic notches, we define a threshold T_3 to be half of the median of all the identified peaks in the previous step and cut off all the peaks less than T_3.

After the above steps, we identify all the peaks which are clearly local maxima in their vicinities and calculate the moving heart rate per each second. A good size of a vicinity to find local maxima can be 0.25 seconds. We can safely assume that no two peaks can co-exist within a span of 0.25 seconds. The reasoning behind this comes from the assumption that the maximum heart rate cannot exceed 250 beats per minute.

7 TIME COMPLEXITY

To evaluate the time complexity of each algorithm that was implemented, we use Appendix A as reference to understand the number of computations involved. ***N*** is the length of the dataset and ***f*** is the filter order of the filter(s).

7.1 LMS with Synthetic Noise

The main computational part of the code is to compute the expectation vector $\hat{h}$ where ***f*** multiplications are done for ***N-1*** times. The asymptotic time complexity can be written as $O(Nf)$.The algorithm also involves calculating the Fourier and the Inverse Fourier Transforms of the entire data set. If an algorithm like FFT is used, the asymptotic time complexity can be written as $O(NlogN)$. So, the total asymptotic time complexity can be written as $O(Nf+NlogN)$.

7.2 RLS with Synthetic Noise

Similar to that of an LMS filter implementation, the asymptotic time complexity can be written as $O(Nf)$. There is also another computationally intensive part of the code where we calculate the covariance matrix from the input data which amounts to $O(N^2)$.

The algorithm also involves calculating the Fourier and the Inverse Fourier Transforms of the entire data set if a synthetic noise model is used. If an algorithm like FFT is used, the asymptotic time complexity can be written as $O(NlogN)$. So, the total asymptotic time complexity can be written as $O(Nf + NlogN + N^2)$. Unlike in the case of LMS, we can write the final expression for asymptotic time complexity as $O(N^2)$.

7.3 LMS with Accelerometer Noise

If an accelerometer data based noise model is used, the only difference from Section 7.1 calculations would be that there are no Fourier Transforms involved. The final time complexity in this case would be $O(Nf)$.

7.4 RLS with Accelerometer Noise

If an accelerometer data based noise model is used, the time complexity would still remain $O(N^2)$. The reason is that the disappearance of less dominant FFT calculation would not affect the worst case time complexity.

7.5 ANC with FSLMS and FLANN

The FLANN involves expanding N elements into $M = N(2P+1)$ elements where P is the expansion order. The time it takes to do that is $O(NP)$. Similar to what we did in LMS, the time complexity for filtering can be written as $O(MP)$ which is equal to $O(NP^2)$.

7.6 Final Algorithm

The time it takes to calculate the variance, which is one of the critical components of this algorithm, is $O(N)$.The other critical computations are already calculated in the above sections. For ANC with FSLMS and FLANN, the time complexity is $O(NP^2)$. For

FFT computation, the time complexity is $O(NlogN)$. So, the overall time complexity is $O(NP^2 + NlogN)$.

The Tables 7.1 to 7.4 show the relative times taken by the algorithmic implementations of all the aforementioned algorithms against the time taken by a 32 ordered LMS with accelerometer noise implementation on MATLAB®.

Table 7.1: Relative times taken for LMS with synthetic noise

Program	16 order	32 order
Time taken	1.5	1.6

Table 7.2: Relative times taken for RLS with synthetic noise

Program	16 order	32 order
Time taken	4	3.7

Table 7.3: Relative times taken for RLS with accelerometer noise

Program	16 order	32 order
Time taken	1.4	1.6

Table 7.4: Relative times taken

Program	ANC 32 order	Final Algorithm 32 order
Time taken	1.3	1.3

8 DATA SET DESCRIPTION

8.1 Main Data

The data [22] consists of PPG signals acquired from 8 channels of sensors. The collected data has been obtained from two different types of LEDs *viz.* 4 channels with Green LEDs and 4 channels with IR LEDs. The data has been obtained at a sampling frequency of 200 Hz. The data spanned for a total time of 10 minutes. The data included PPG recorded during the following activities- 2 minutes of sitting, 2 minutes of hand waving, 2 minutes of running and 2 minutes of sitting again. The reference heart rate was measured using a MIO sensor. The data set also has X, Y and Z axes accelerometer data.

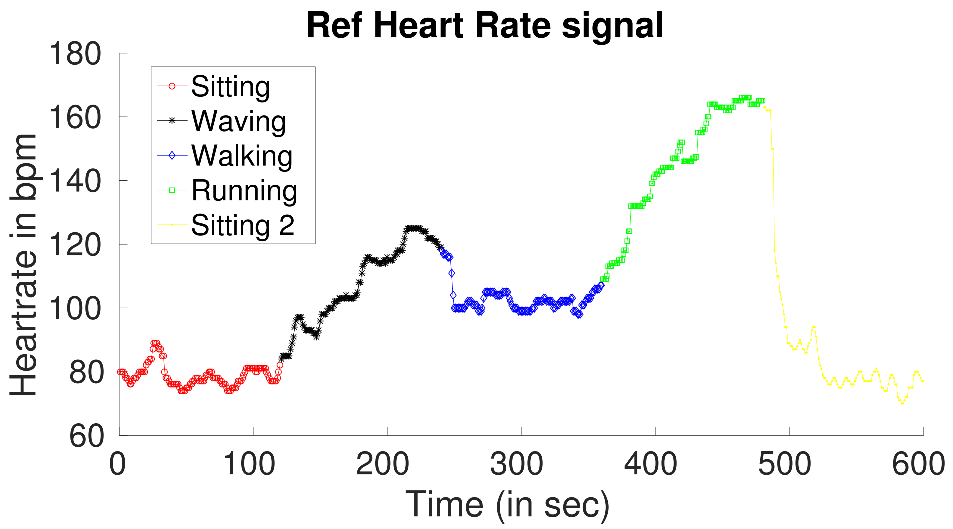

Figure 8.1: Reference heart rates measured by the MIO device

Below, Fig. 8.2 is a graph that shows the accelerometer data for different activities and of different axes. On the other hand, Fig. 8.3 shows normalized raw data for the four consecutive activities.

8.2 Auxiliary Data

To test the algorithm that it doesn't suffer from small data bias, an auxiliary data set described and used in [23] is used to test the final proposed wrapper algorithm. The data set is obtained from 8 subjects of which 7 are healthy and the remaining person suffers from a high blood pressure. Each subject does a specific set of actions that may involve forearm or hand type of movements or running. The PPG sensors use a green LED on a wrist mount type of a setup. The data is acquired at a sampling frequency of 125 Hz. The data set is also accompanied by 3 axes accelerometer data. A reference heart rate for an 8 second window with a 6 second overlapping sliding window has been provided. A sample data set is shown in Fig. 8.4.

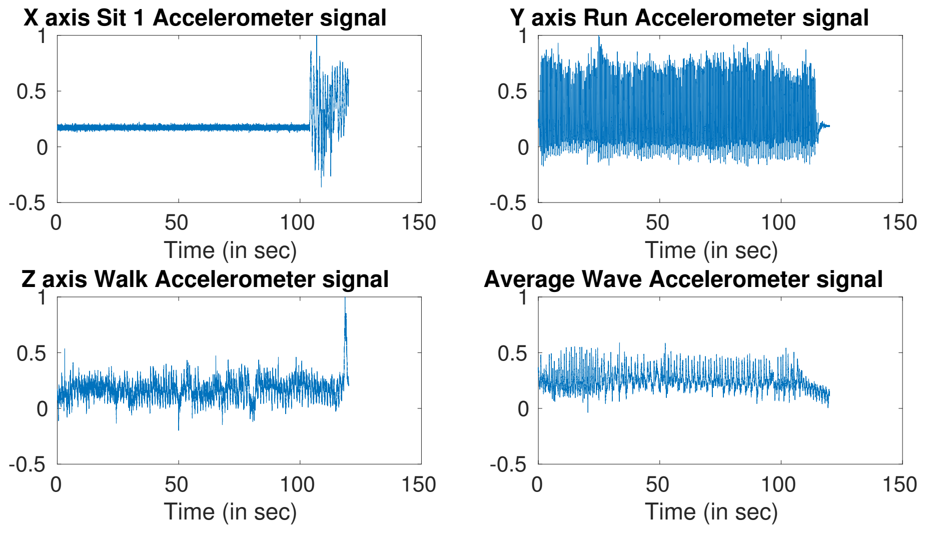

Figure 8.2: Different accelerometer signals

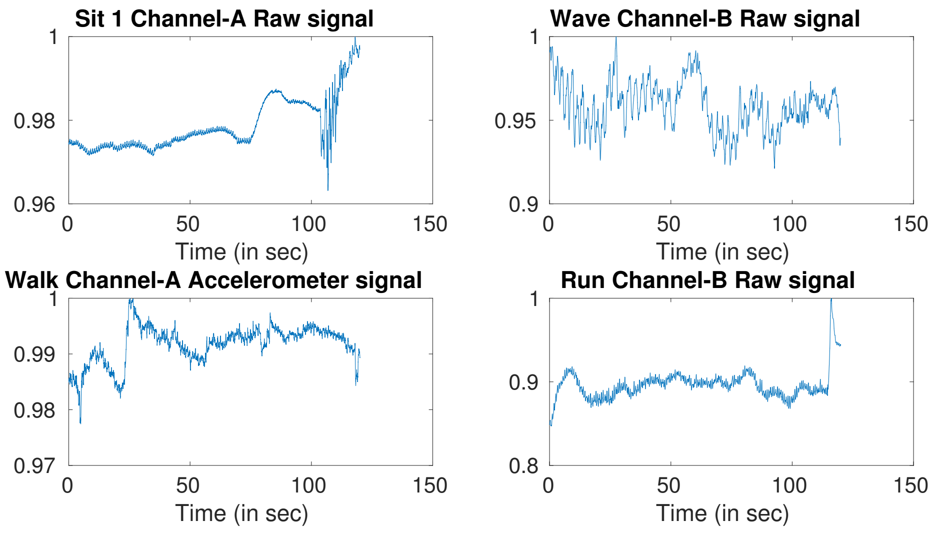

Figure 8.3: Raw signals

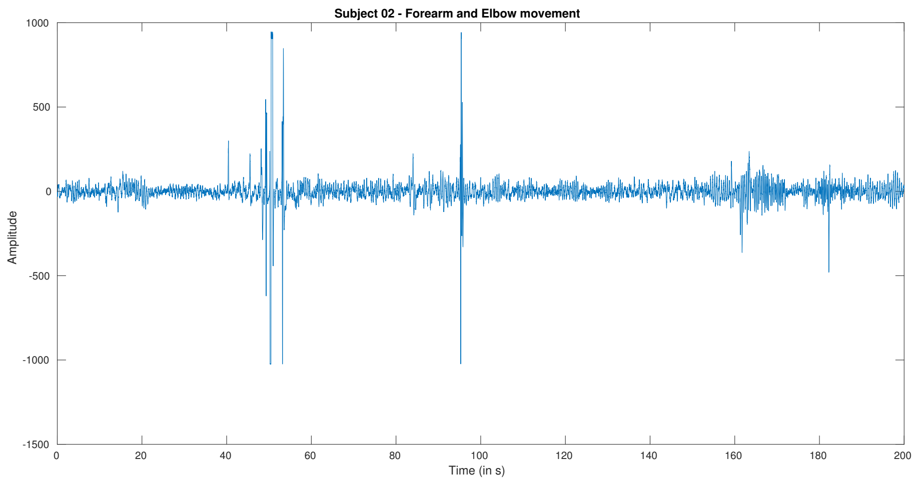

Figure 8.4: Sample PPG data recording of subject 2 from the auxiliary dataset

9 RESULTS

9.1 Synthetic Noise Model

9.1.1 MA Reduction Using LMS

A filter order of 16 is used for the LMS. The learning rate is settled at 0.01. All the results are passed through a band pass FIR filter with frequency range 0.04 Hz - 14 Hz. The following Figs. 9.1 to 9.5 represent processed signals on the left along with mean adjusted raw signals on the right using the generated synthetic noise as the golden reference for the LMS Algorithm.

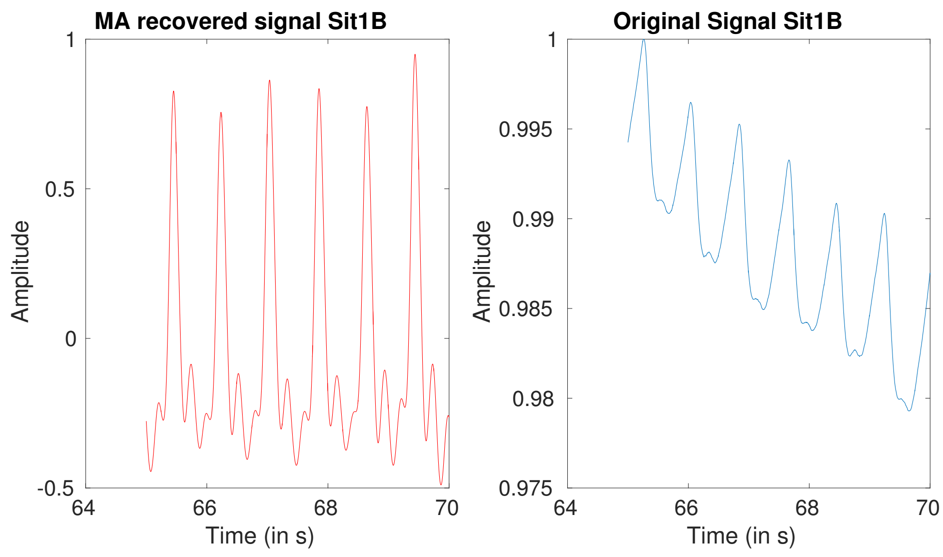

Figure 9.1: LMS output for "Sit" activity on channel-B

Furthermore, all the results are from signals acquired from channel B. Similar results can be expected for signals from channel A.

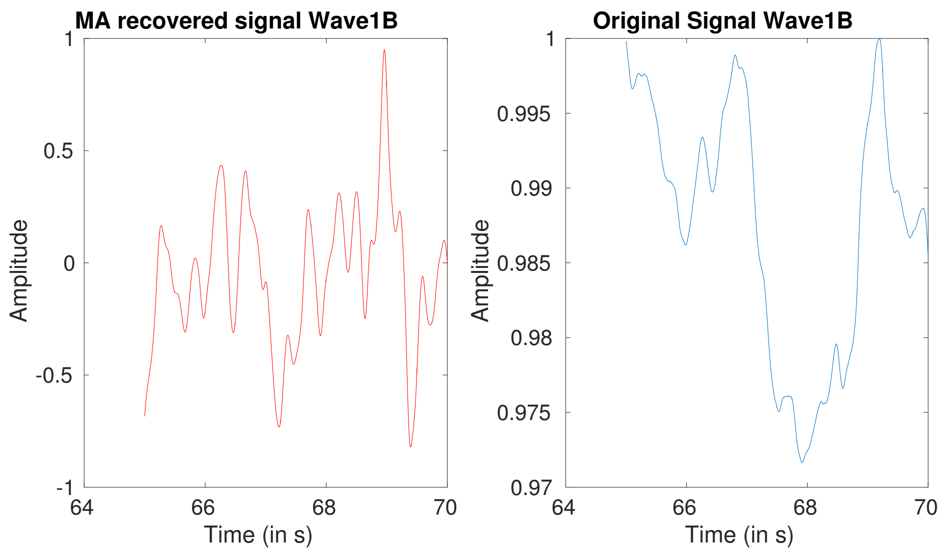

Figure 9.2: LMS output for “Wave” activity on channel-B

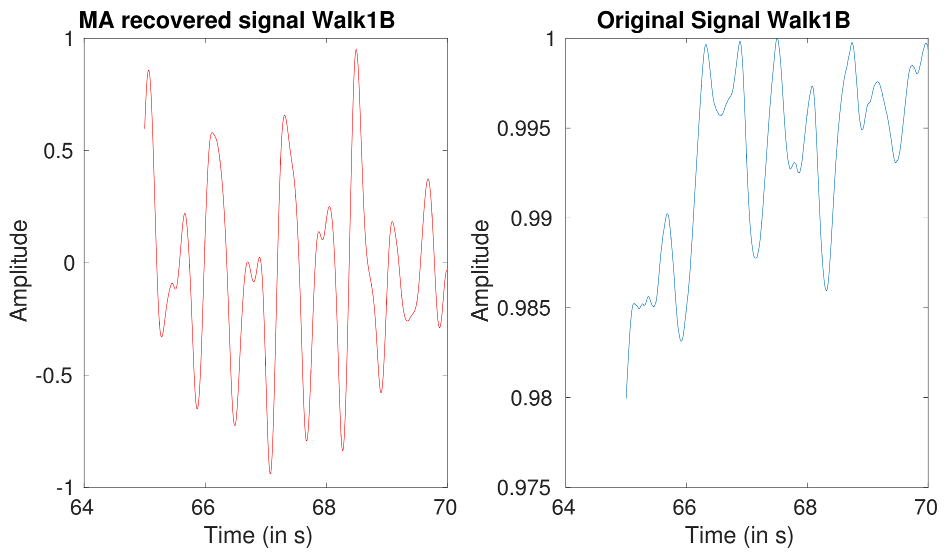

Figure 9.3: LMS output for “Walk” activity on channel-B

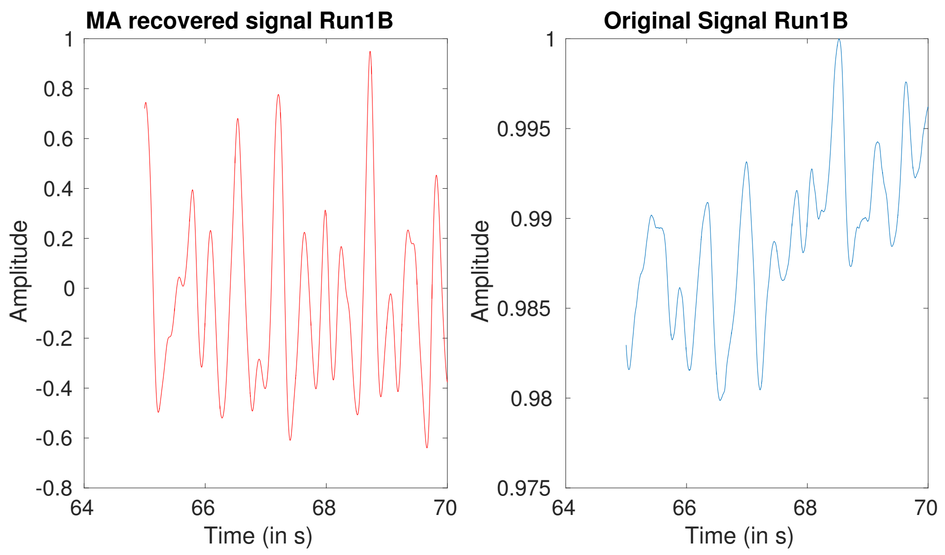

Figure 9.4: LMS output for “Run” activity on channel-B

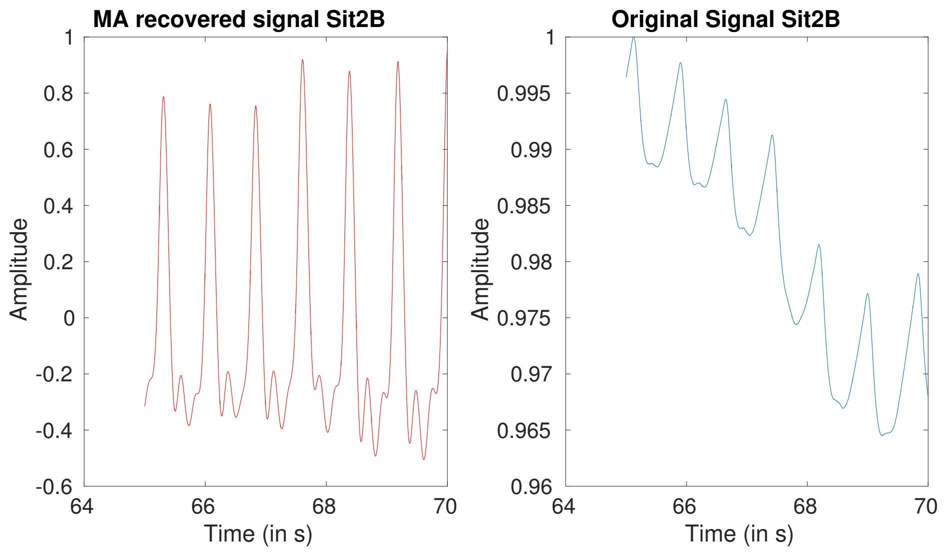

Figure 9.5: LMS output for “Sit 2” activity on channel-B

9.1.2 MA Reduction Using RLS

A filter order of 32 is used for RLS. The forgetting factor which has been translated into the starting covariance matrix is set at $10^5 * \mathrm{I}_{matrix}$. The following Figs. 9.6 to 9.10 represent processed signals along with raw signals using the generated synthetic noise as the golden reference for the RLS Algorithm. All the signals are mean adjusted.

Furthermore, all the results are from signals acquired from channel B. Similar results can be expected for signals from channel A.

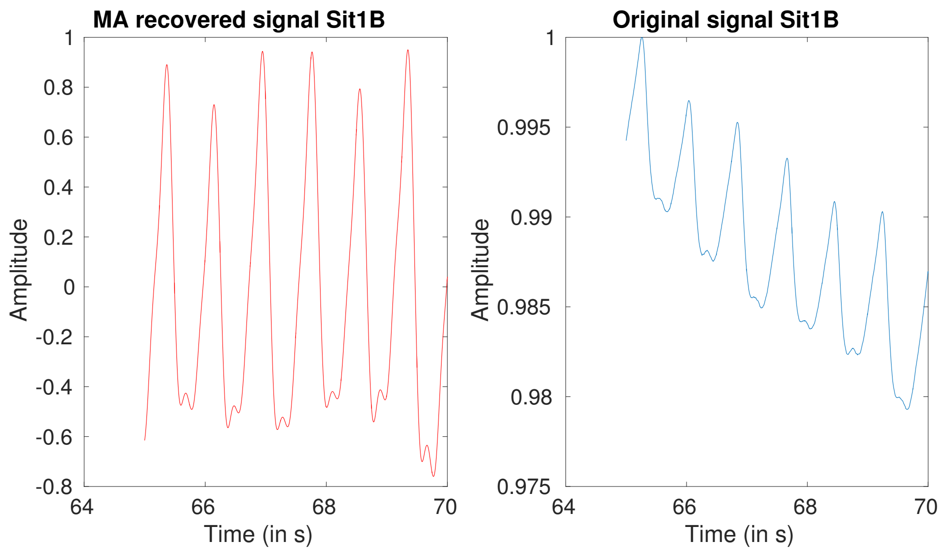

Figure 9.6: RLS output for “Sit” activity on channel-B

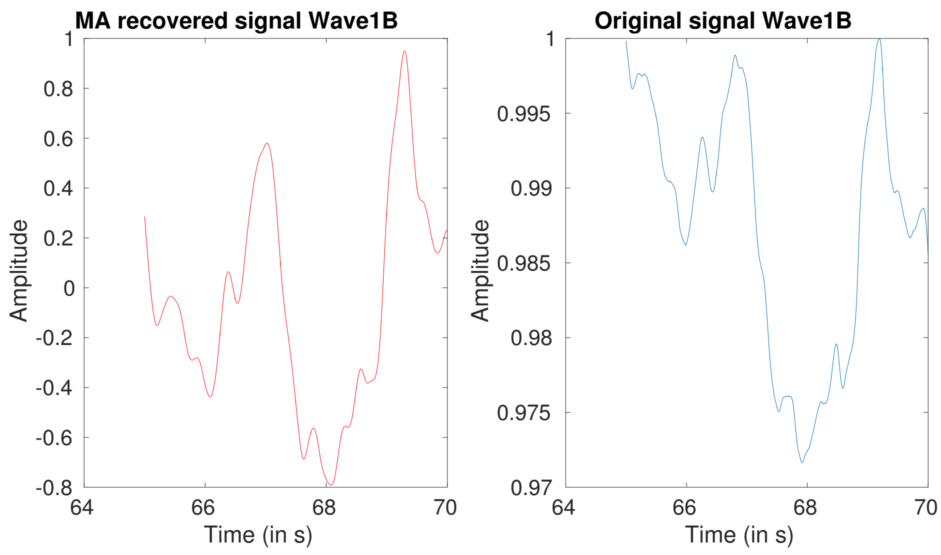

Figure 9.7: RLS output for “Wave” activity on channel-B

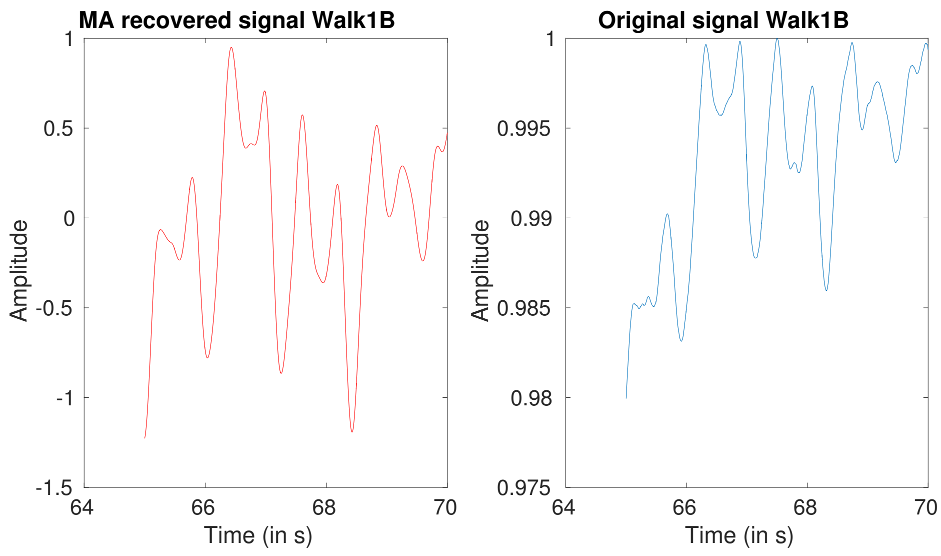

Figure 9.8: RLS output for “Walk” activity on channel-B

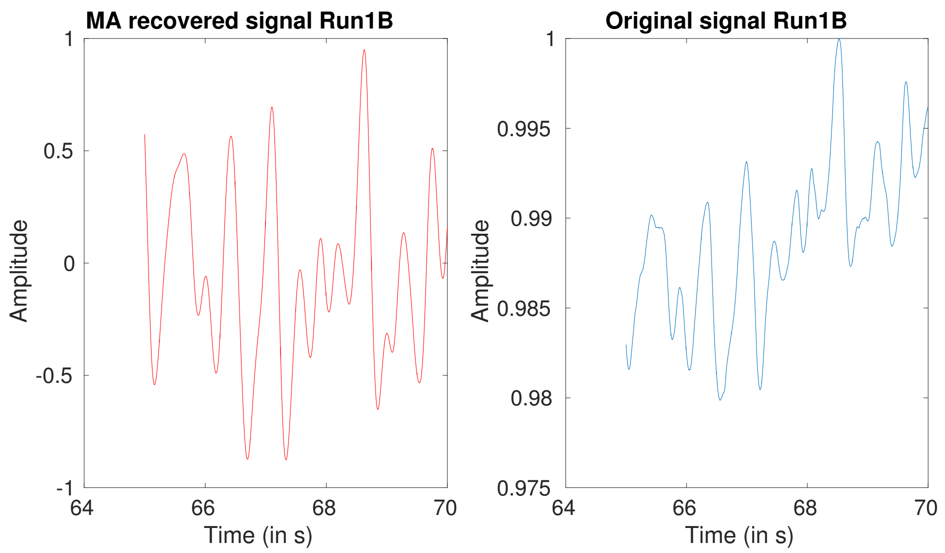

Figure 9.9: RLS output for “Run” activity on channel-B

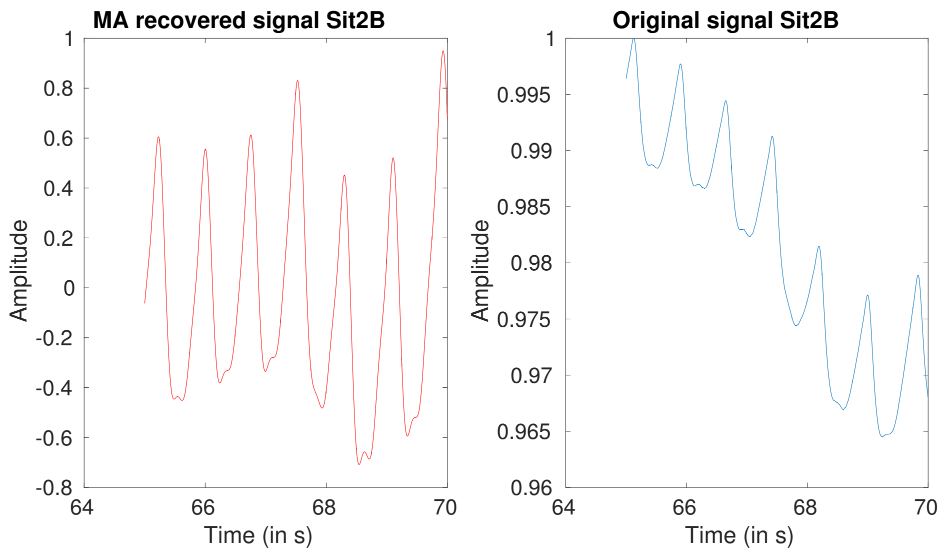

Figure 9.10: RLS output for “Sit 2” activity on channel-B

9.2 Accelerometer Data Based Noise Model

9.2.1 MA Reduction Using LMS

A filter order of 16 is used for the LMS. The learning rate is settled at 0.001. All the results are passed through a band pass FIR filter with frequency range 0.04 Hz - 14 Hz. The following Figs. 9.11 to 9.18 represent processed signals on the left along with mean adjusted raw signals on the right using the accelerometer data as the golden reference for the LMS Algorithm. LMS is run for all the accelerometer data from X,Y and Z axes as noise sources. Another signal is obtained by averaging the processed signals from all the axes.

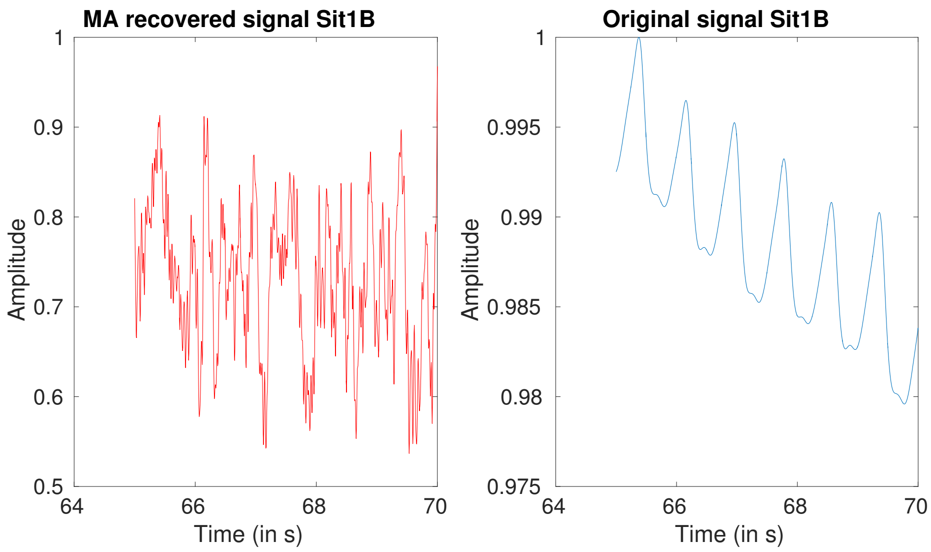

Figure 9.11: Averaged LMS output for "Sit" on channel-B

Furthermore, all the results are from signals acquired from channel B. Similar results can be expected for signals from channel A.

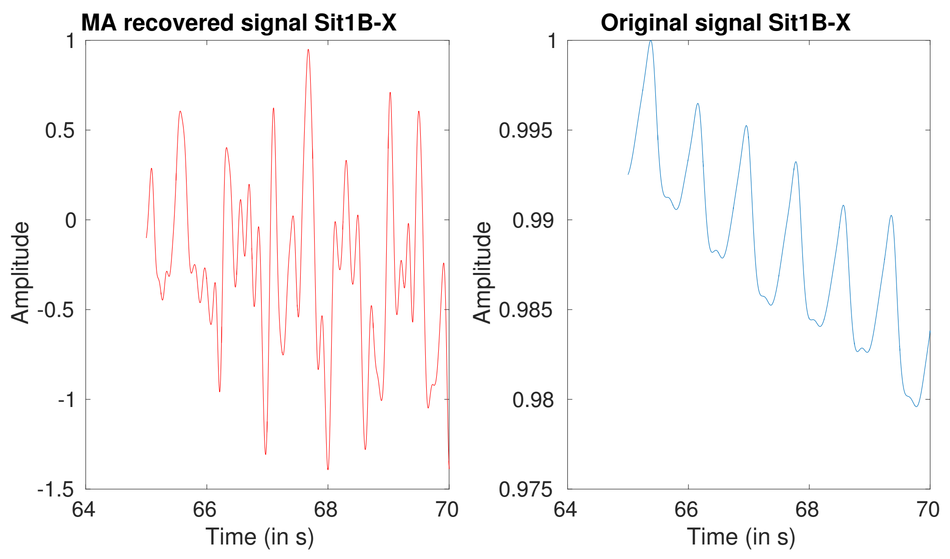

Figure 9.12: LMS output for “Sit” on channel-B with X-axis accelerometer noise

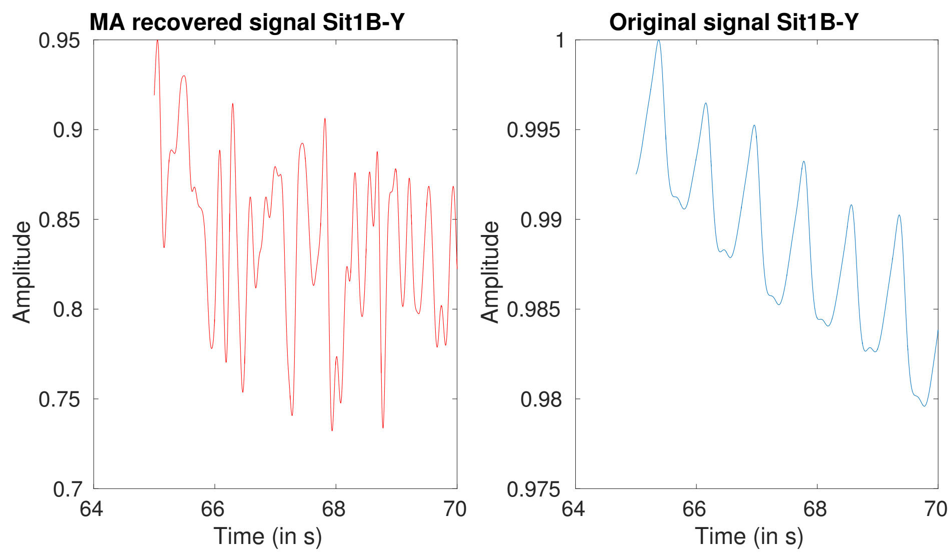

Figure 9.13: LMS output for “Sit” on channel-B with Y-axis accelerometer noise

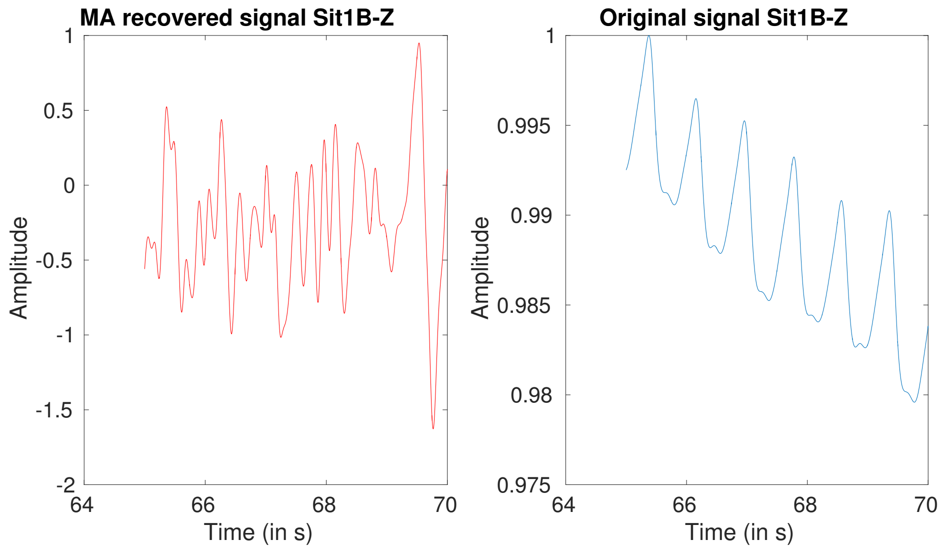

Figure 9.14: LMS output for “Sit” on channel-B with Z-axis accelerometer noise

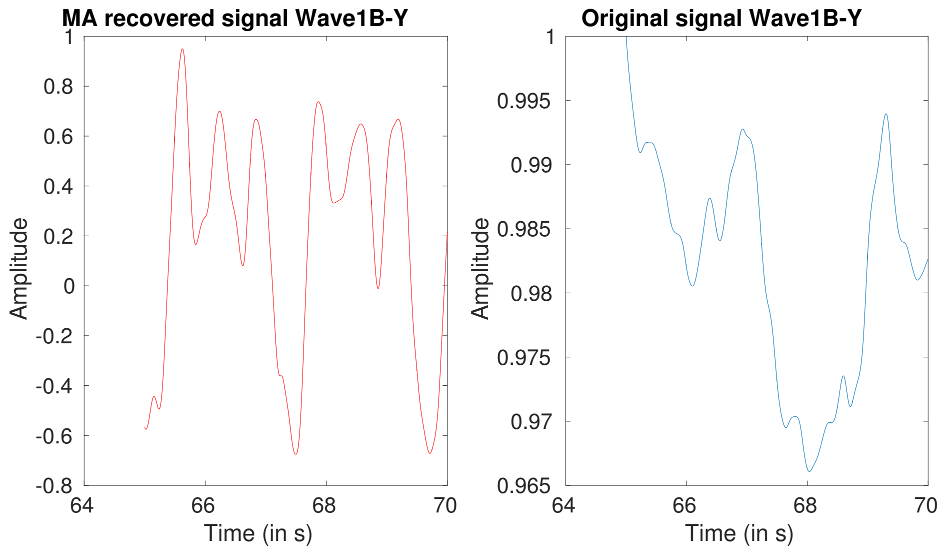

Figure 9.15: LMS output for “Wave” on channel-B with Y-axis accelerometer noise

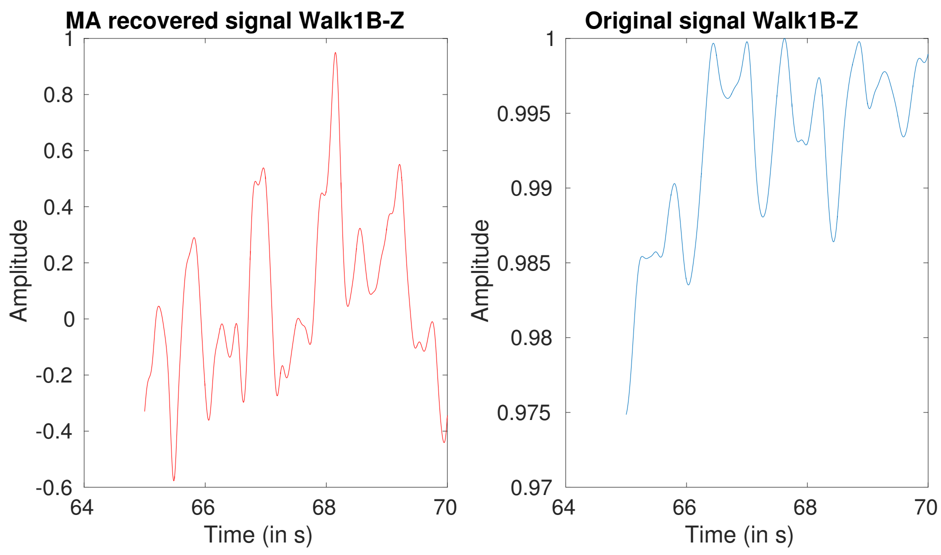

Figure 9.16: LMS output for “Walk” on channel-B with Z-axis accelerometer noise

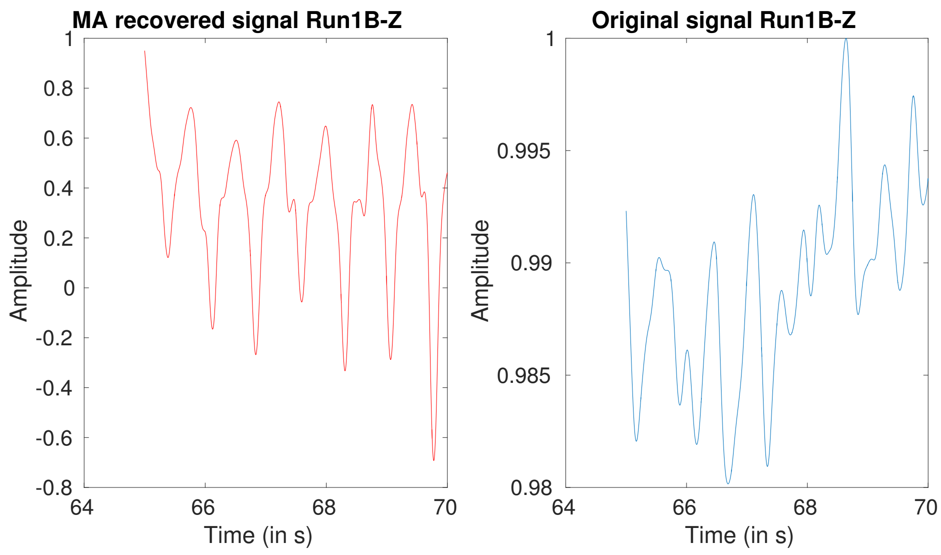

Figure 9.17: LMS output for “Run” on channel-B with Z-axis accelerometer noise

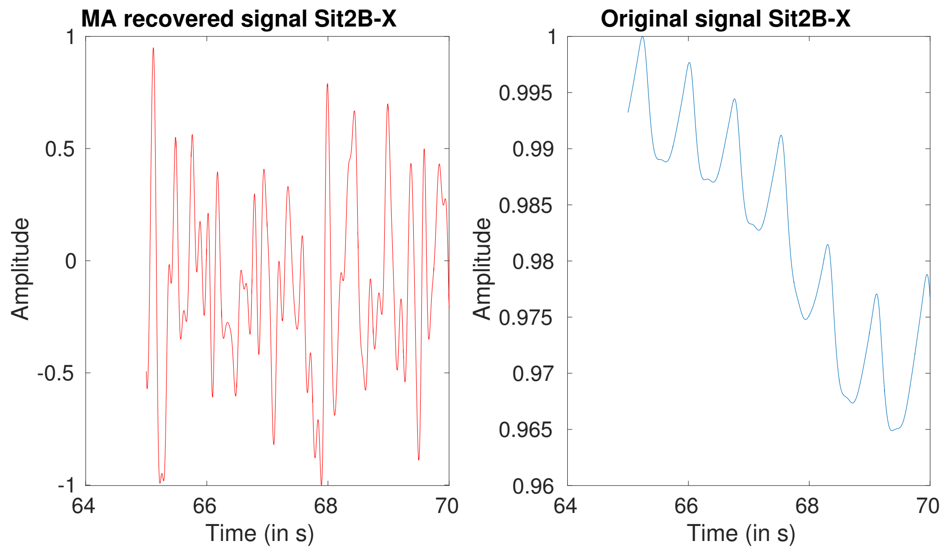

Figure 9.18: LMS output for “Sit 2” on channel-B with X-axis accelerometer noise

9.2.2 MA Reduction Using RLS

A filter order of 32 is used for the RLS. The forgetting factor which has been translated into the starting covariance matrix is set at $10^5 * \mathrm{I}_{matrix}$. All the results are passed through a band pass FIR filter with frequency range 0.04 Hz - 14 Hz. The following Figs. 9.19 to 9.26 represent processed signals on the left along with mean adjusted raw signals on the right using the accelerometer data as the golden reference for the RLS Algorithm. RLS is run for all the accelerometer data from X,Y and Z axes as noise sources. Another signal is obtained by averaging the processed signals from all the axes.

Furthermore, all the results are from signals acquired from channel B. Similar results can be expected for signals from channel A.

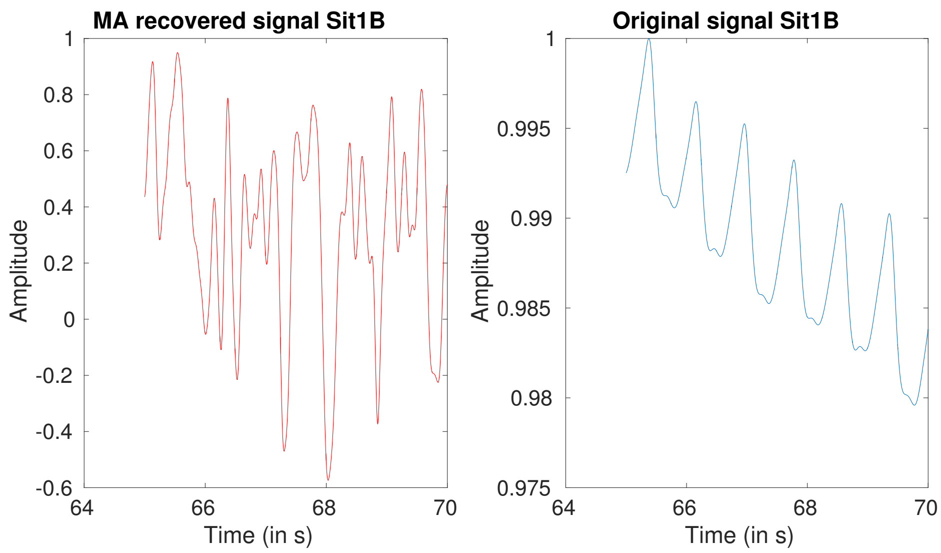

Figure 9.19: Averaged RLS output for “Sit” on channel-B

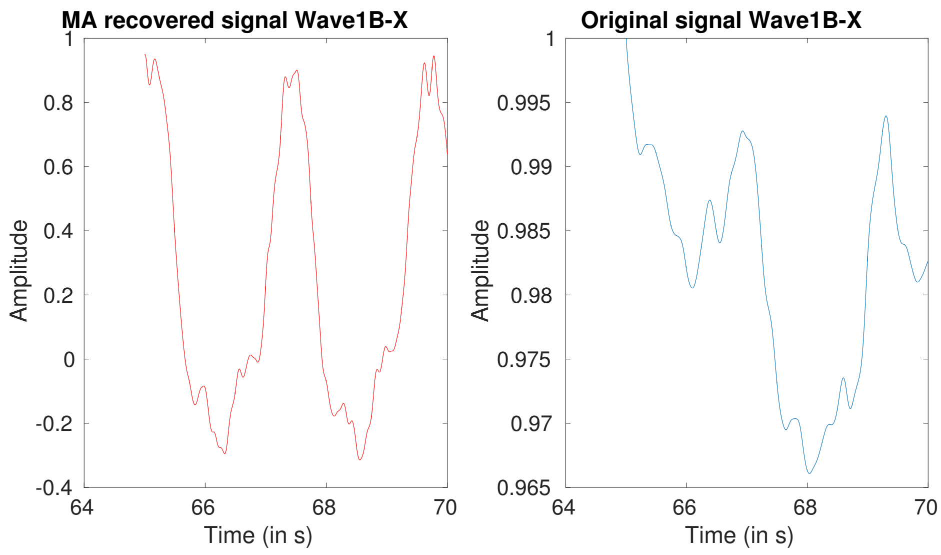

Figure 9.20: RLS output for “Wave” on channel-B with X-axis accelerometer noise

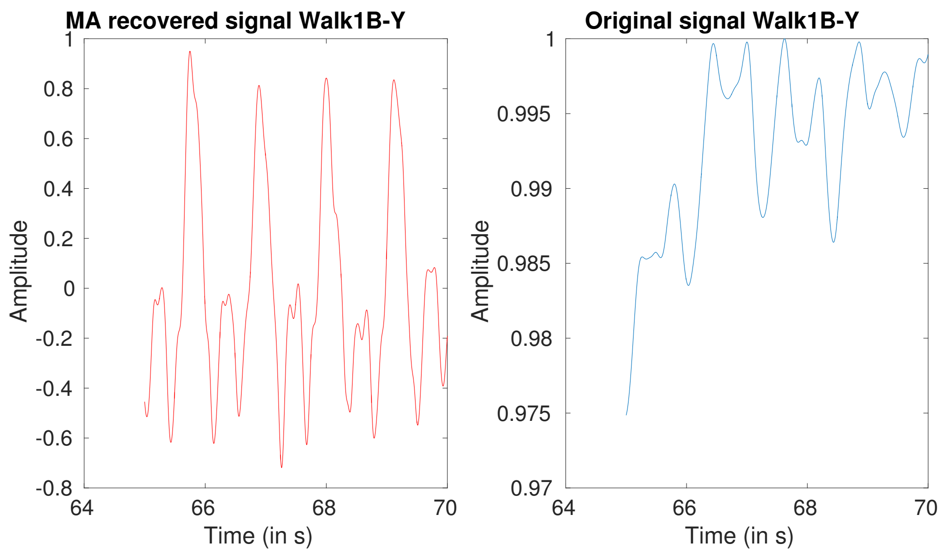

Figure 9.21: RLS output for “Walk” on channel-B with Y-axis accelerometer noise

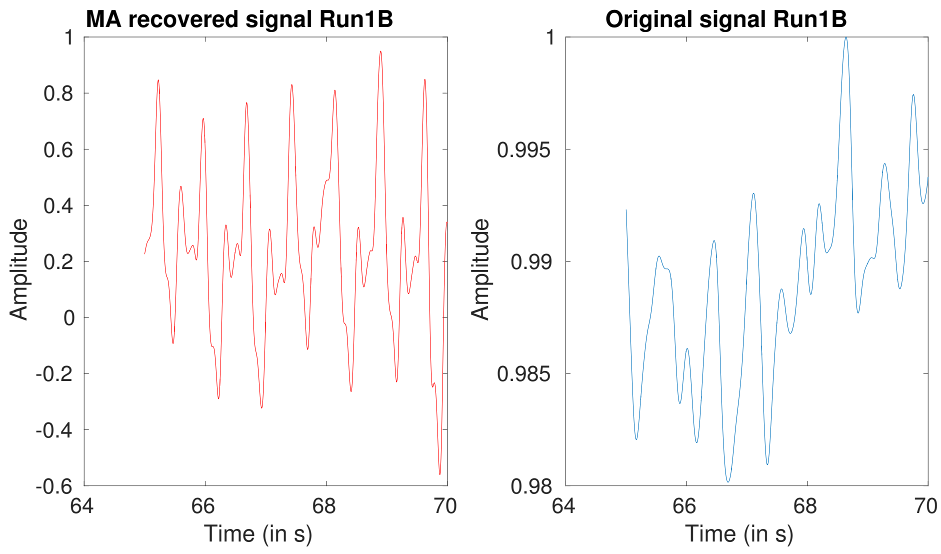

Figure 9.22: Averaged RLS output for “Run” on channel-B

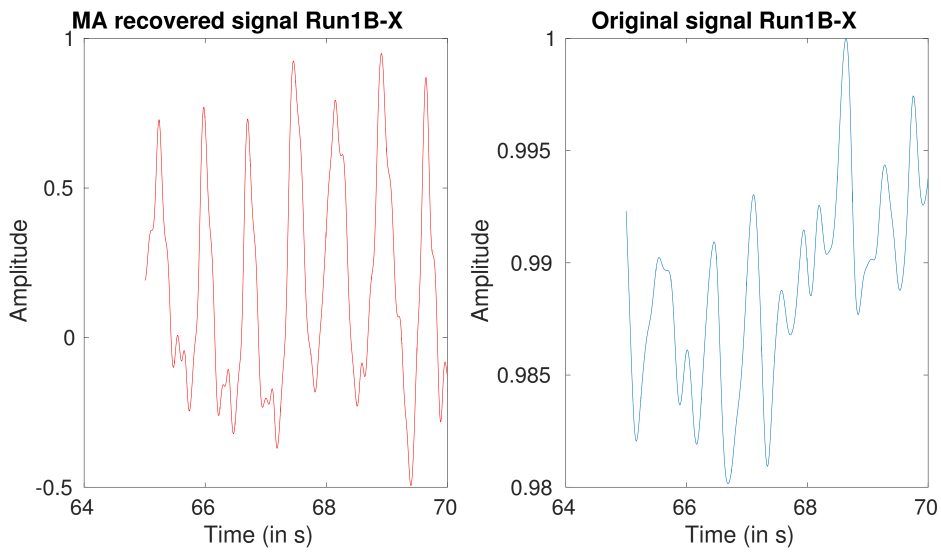

Figure 9.23: RLS output for “Run” on channel-B with X-axis accelerometer noise

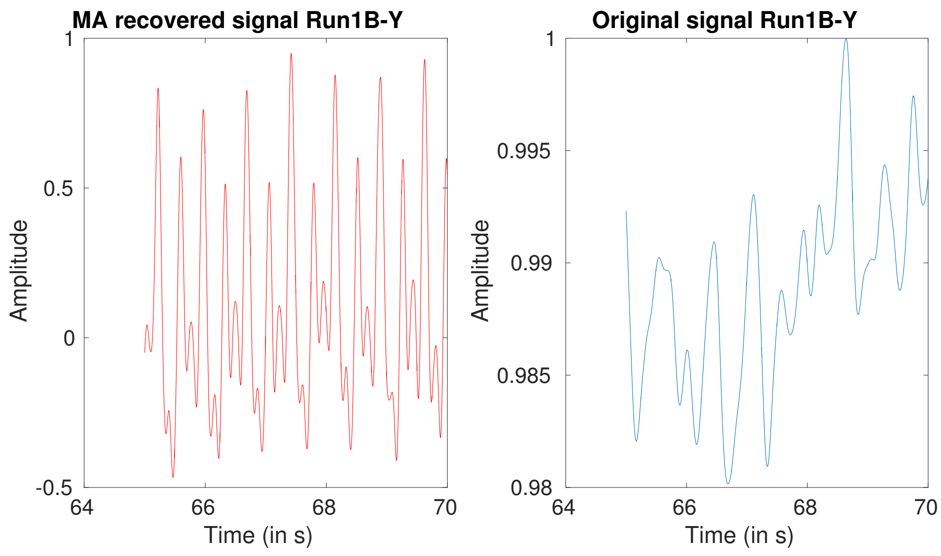

Figure 9.24: RLS output for “Run” on channel-B with Y-axis accelerometer noise

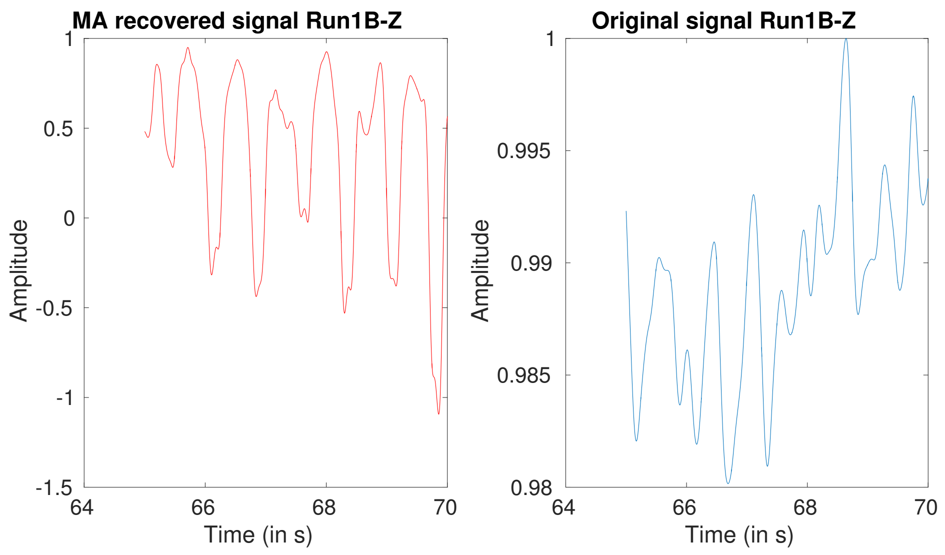

Figure 9.25: RLS output for “Run” on channel-B with Z-axis accelerometer noise

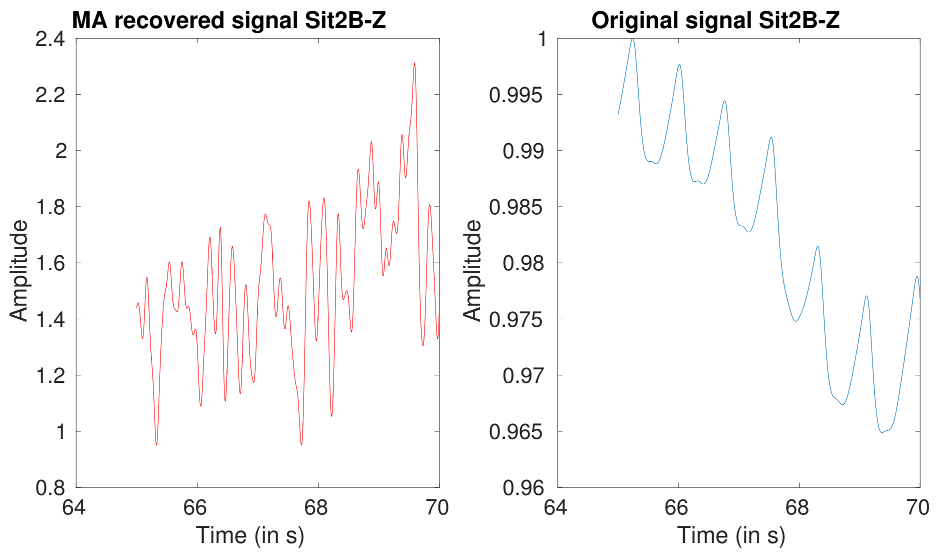

Figure 9.26: RLS output for “Sit 2” on channel-B with Z-axis accelerometer noise

9.2.3 MA Reduction Using ANC with FLANN and FSLMS

A trigonometric type expansion is used in the FLANN. An expansion order of 32 is used. A filter order of 32 is used for the FSLMS. The learning rate for FSLMS is settled at 0.1. All the results are passed through a band pass FIR filter with frequency range 0.04 Hz - 14 Hz. The following Figs. 9.27 to 9.31 represent processed signals on the left along with mean adjusted raw signals on the right using the accelerometer data as the input for the Active Noise Cancellation program implemented with FLANN and FSLMS. The algorithm is run for all the accelerometer data from X,Y and Z axes as noise sources. Another signal is obtained by averaging the processed signals from all the axes.

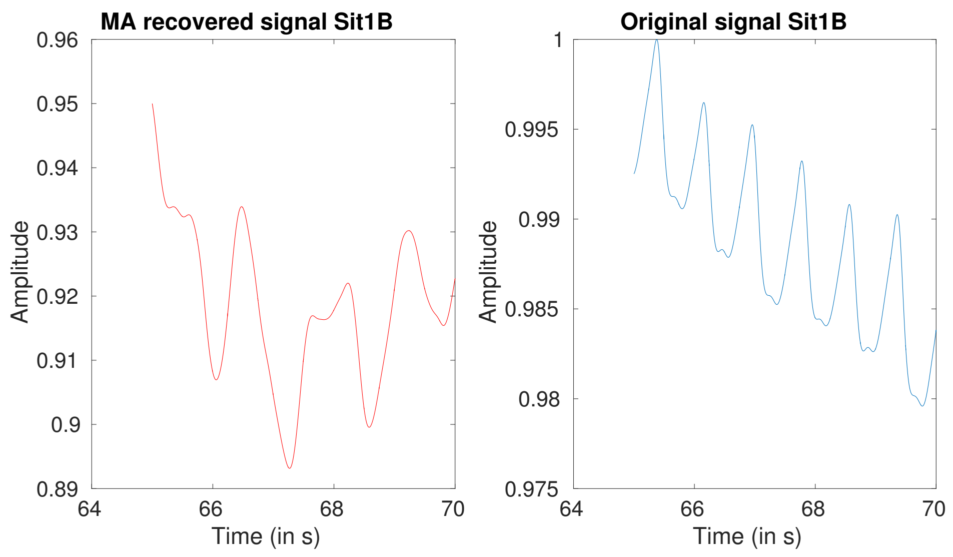

Figure 9.27: ANC output for “Sit” on channel-B

Furthermore, all the results are from signals acquired from channel B. Similar results can be expected for signals from channel A.

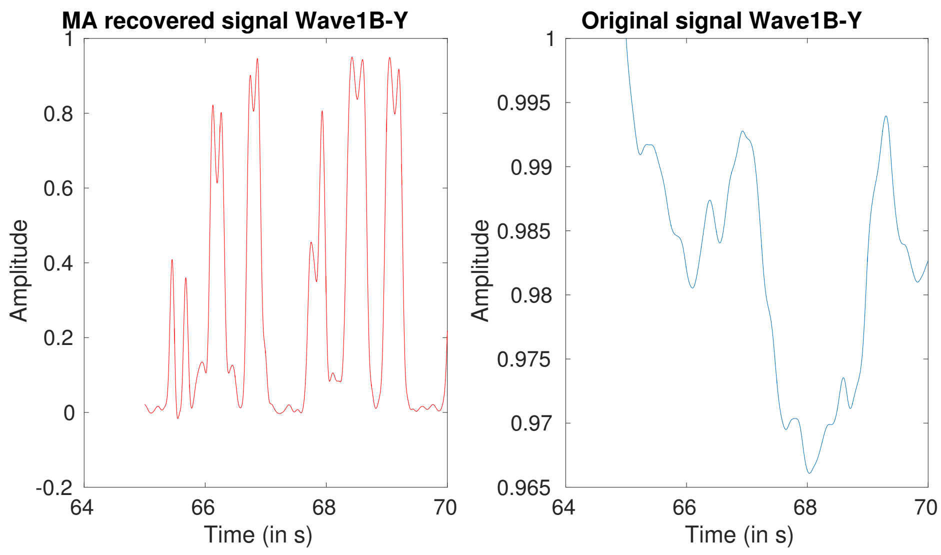

Figure 9.28: ANC output for “Wave” on channel-B with Y-axis accelerometer noise

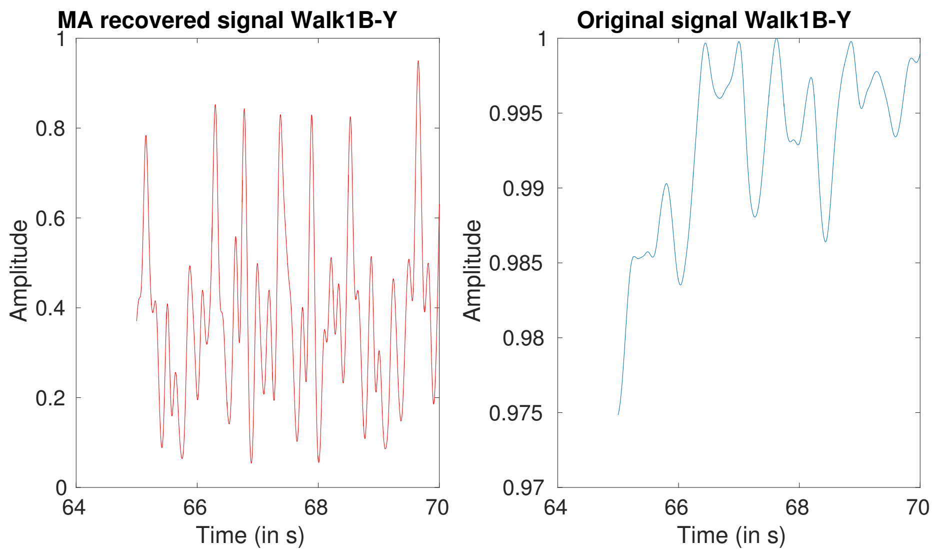

Figure 9.29: ANC output for “Walk” on channel-B with Y-axis accelerometer noise

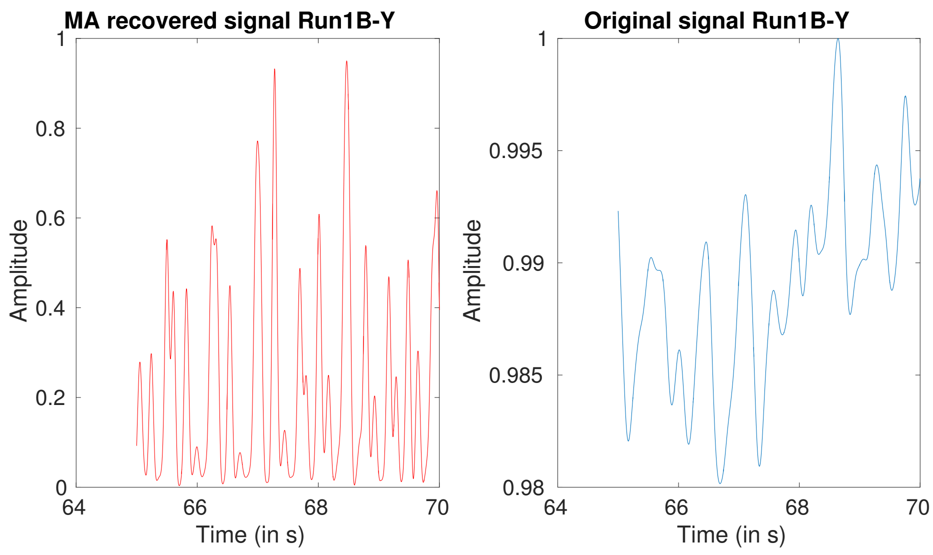

Figure 9.30: ANC output for “Run” on channel-B

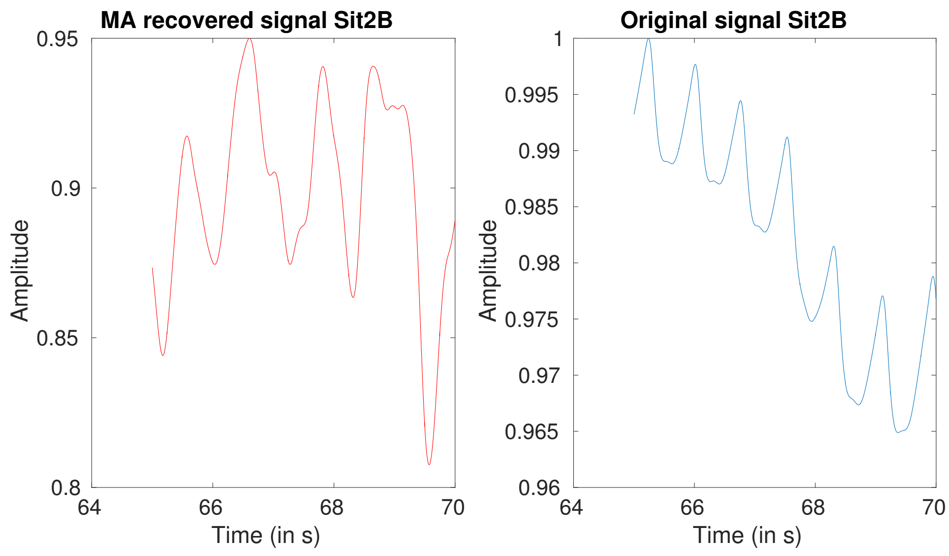

Figure 9.31: ANC output for “Sit 2” on channel-B

10 DISCUSSION AND INTERPRETATION OF THE RESULTS

10.1 Synthetic Noise Results

The synthetic noise model captures the noise information from the raw signal. Even though there are implementation differences, the results for LMS and RLS are observed to be similar. So, for the following section, the inferences hold for both LMS and RLS algorithms. As briefly explained in the previous sections, most motion artifacts like finger movements, involuntary muscle jitters may lie in the most observable frequency region (>0.25 Hz and <5 Hz). This can be intuitively understood by flicking your finger as fast as you can. 5 Hz is when you can flick your finger 5 times in a second. When working with wearables, one often has to consider the extreme cases of motion artifact impediments that can affect our acquired signal. So, following this train of argument, the synthetic noise that we generated will only be useful if the motion artifacts don't coincide with useful signal.

We can thus observe the results when LMS and RLS schemes are employed in this kind of an environment from Figs. 9.1 to 9.10. The results in Fig. 9.1 and Fig. 9.5 for LMS and Fig. 9.6 and Fig. 9.10 for RLS are quite evident and lucid. As shown in the Fig. 10.1, one can observe the dicrotic notch that is circled. Also, the low frequency drift that is present in the raw signal has been eliminated by the adaptive filters without the need of a high pass filter. The results are crisp owing to the fact that the activity that corresponds to the figures discussed is sitting (at the start and at the end of the experiment), which may not possess many motion artifacts inherently.

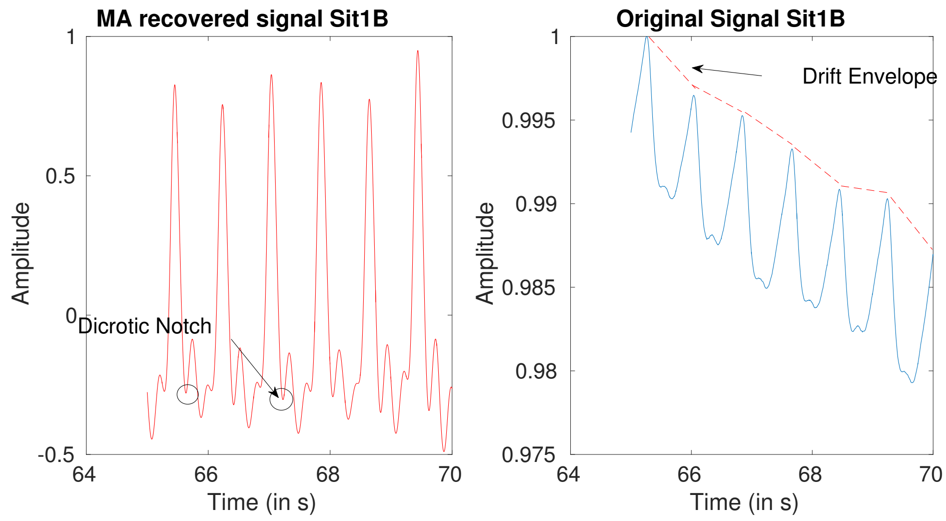

Figure 10.1: Few features of LMS/RLS filtered “Sit” activity PPG signal

At this juncture, if one can find the noise by suppressing the required band of frequencies in the FFT domain, it begs the question of whether we can directly use the band we suppressed and claim that as the required signal. To understand the subtlety of this particular logic, we need to direct our attention to the Fig. 10.2. The figure shows both the signals obtained from LMS output(error in our case) and subtracting the synthetic noise from original signal. Some features like observable dicrotic notches are missing from signals obtained from plain subtraction. Also, in some activities like Walk and Wave, LMS output recovered some peaks while the subtracted signal seriously missed all peaks which can be potentially linked to heart rate measurements.

On careful thought, one can infer that LMS being an iterative algorithm, modifies the entire signal more than once in the span of multiple iterations. In doing so, in each iteration the signal gets modified with a particular learning rate which tries to emulate the noise reference signal. So, more noise features leave the error signal every iteration.

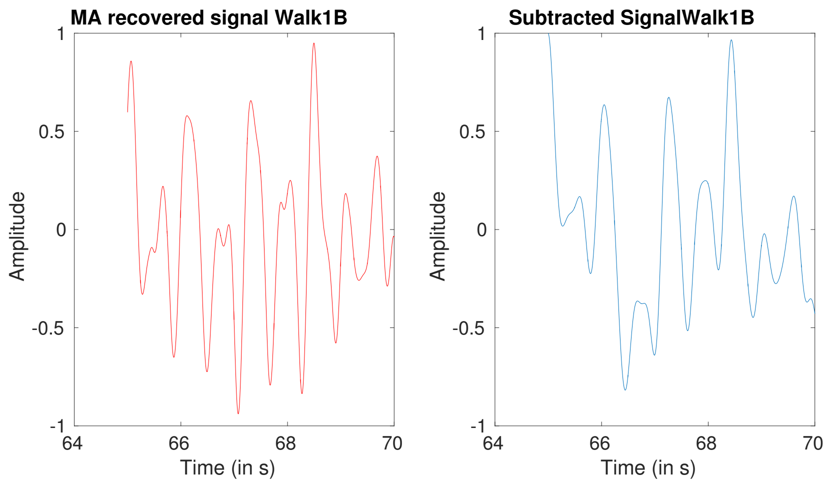

Figure 10.2: Comparing filtered and subtracted signals for “Walk” activity

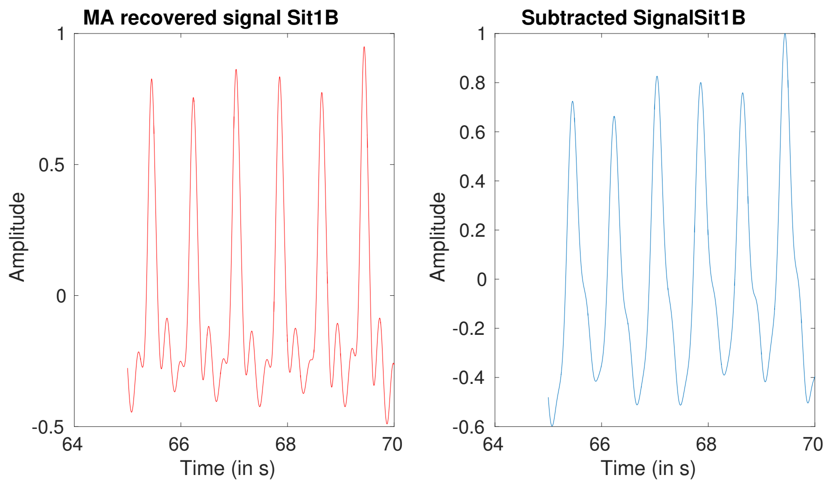

Figure 10.3: Comparing filtered and subtracted signals for “Sit” activity

10.2 Accelerometer Noise Results

This section assumes that accelerometer signals approximate a pure noise source. The results are rather interesting in more than one aspect. A look at Figs. 9.11 to 9.26 unravels many conclusions on the type of motion that dominates the artifact domain for a particular activity and also the reliability of such accelerometer noise models.

Let us consider an example and dissect the results to get a clear understanding of the performance of this accelerometer based model. The Fig. 10.4 is the final result when all the LMS results of X,Y and Z axes of Running activity are averaged. Ideally, a simple hand count of the predominant peaks must give us an estimate of heart rate.

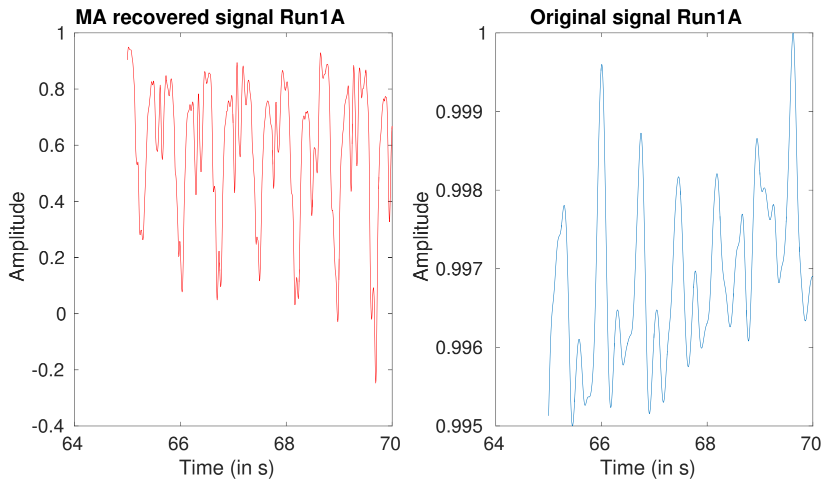

Figure 10.4: Averaged LMS filtered signal

Unfortunately, the averaged signal gives a count that is not accurate. So, when looked at the other results *viz.* Figs. 10.5 to 10.7 we can make an interesting observation. The results when Y-axis accelerometer is taken as a noise source seem favorable. A hand count will yield 13 peaks from 65th second to 70th second which leads to an approximate heart rate of 156 beats per minute. The actual measured heart rate according to the MIO device is 148 beats per minute(147.4 rounded to the next integer). Thus, we can infer that the predominant motion is in Y-axis. The results from X and Z axes aren't that impressive for this activity. Thus the predominant axis of motion also determines the noise activity. Similar results can be seen in motion intensive activities like Walking and Waving.

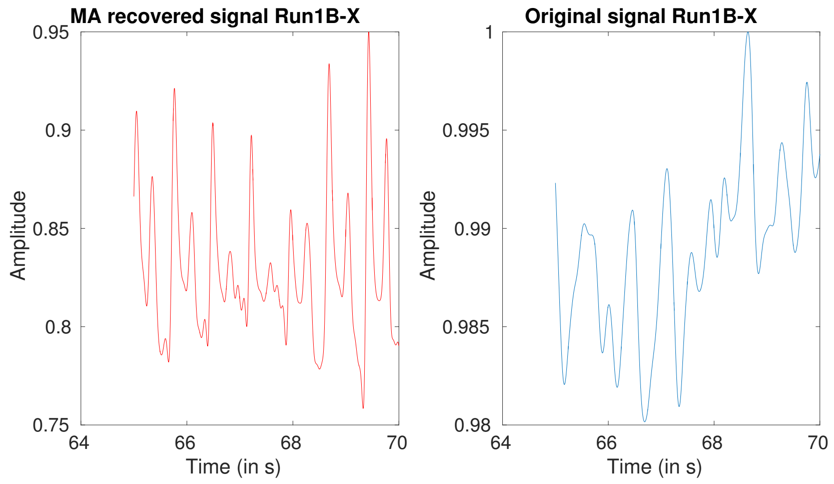

Figure 10.5: LMS filtered signal with X-axis accelerometer reference

There is a catch to this model. This model seems to extract accurate information when there is a significant motion artifact interference. When surveyed, the results when motion is not significant points out the drawback of this model. The Figs. 9.11 to 9.14, 9.18, 9.19 and 9.26 show the performance of the algorithm during Sitting activity, both before and after starting the experiment. One can notice severe irregularities and lack of clear peaks in all the results. As there is no significant motion in any of these results, the

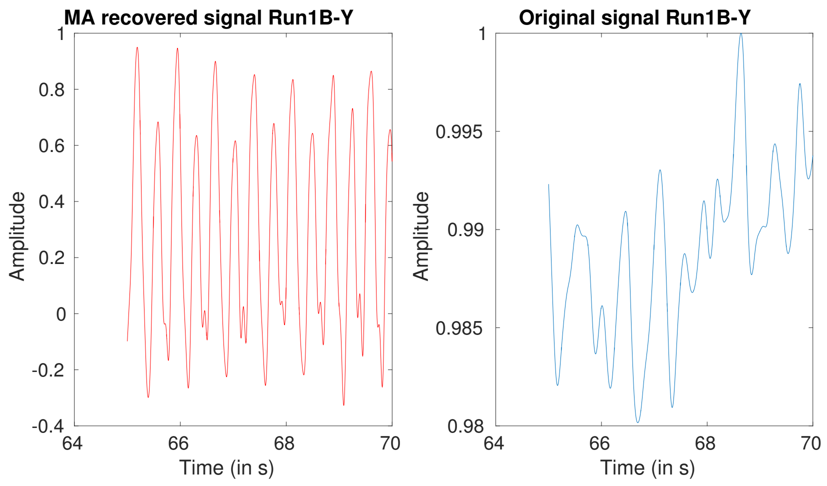

Figure 10.6: LMS filtered signal with Y-axis accelerometer reference

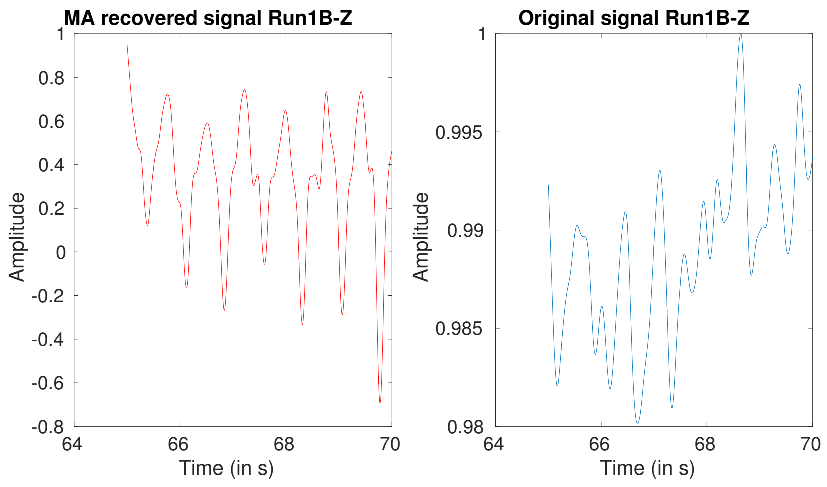

Figure 10.7: LMS filtered signal with Z-axis accelerometer reference

accelerometer signal changes affected the filter despite any huge changes on a grand scale. The Fig. 10.8 shows the variation of accelerometer data even in the sitting activity. The relative changes of the signal when compared with other activities is very less. But these changes dictate the noise model.

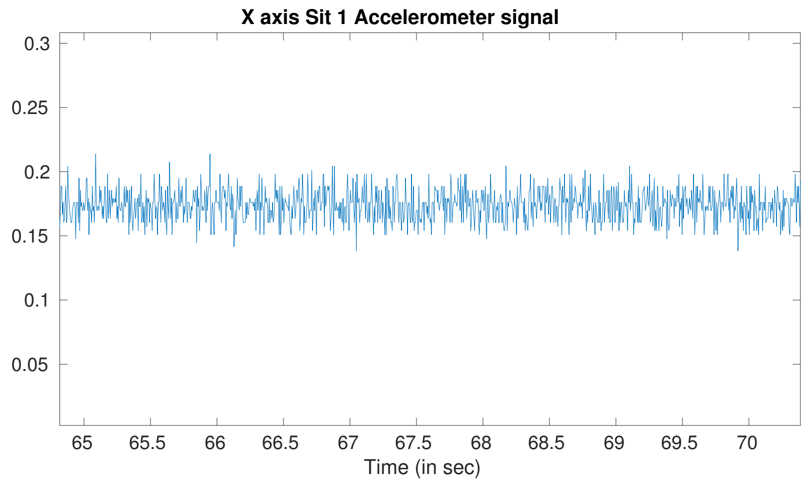

Figure 10.8: X-axis “Sit” accelerometer signal from 65 to 70 seconds

10.3 Heart Rate Calculations

The algorithm described in Chapter 6 is used on the final algorithm which too was described in the very same chapter. A sliding window size ‘s’ is selected and is used to extrapolate the heart rate in beats per minute. To identify the peaks after applying the T_3 threshold, we use the classical peak searching algorithm given by the following boolean function

$$\begin{cases} 1 & x(i-1) < x(i) > x(i+1) \\ 0 & Otherwise \end{cases}$$

where *x(i)* is the current signal value which will be compared against both *x(i-1)*, the previous and *x(i+1)*, the next value to see if its a local maxima. If we find the local maxima, we will note it as a peak and skip 0.25 seconds worth waveform in the hope that we will not find a peak.

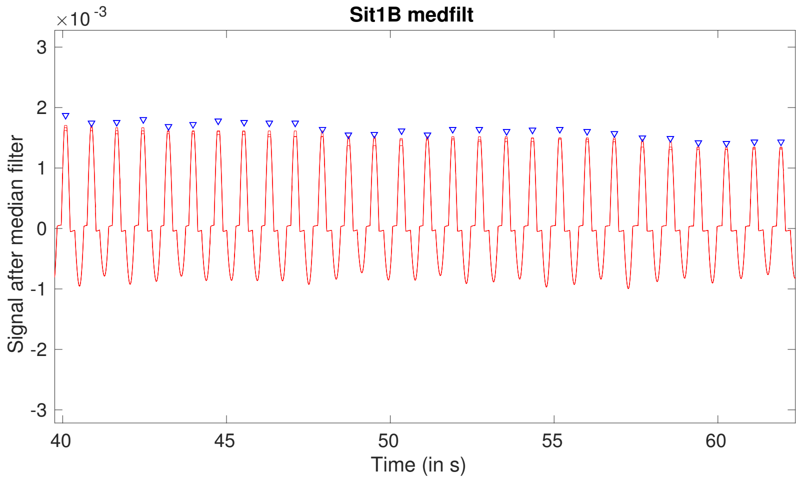

Figure 10.9: Processed “Sit” signal after median filtering showing peaks

With such a scheme as discussed above, there is a possibility that the peak gets stuck at a local maxima as a result of dicrotic notch. So, another small validation code is added to ensure that the peak we discovered is in fact a local maxima in the vicinity of 0.125 seconds. With this short improvement, the peaks in Fig. 10.9 and Fig. 10.10 are clearly categorized correct.

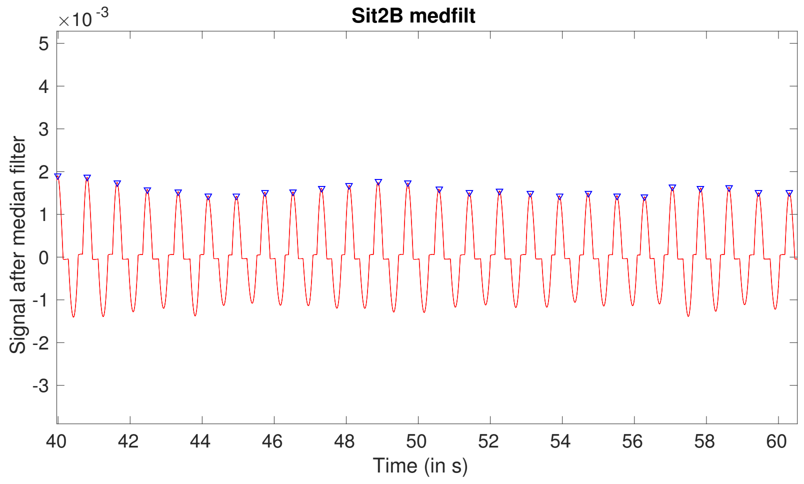

Figure 10.10: Processed "Sit 2" signal after median filtering showing peaks

Most of the peaks in Figs. 10.11 to 10.13 are correctly identified. Although a few of the peaks are missed and a few false positives made it through. The following tables depict the calculated and measured heart rates for a particular second per activity. Two processing window sizes are taken *viz.* 4 and 5 and the results are averaged together to get the figures given in Table 10.1. Choosing one window size results in a quantized set of heart rate values and so two window sizes are considered.

Table 10.1: Heart rates for final algorithm at the 54^{th} second

Activity	**Reference Heart Rate**	**Measured Heart Rate**	**Error %**
Sitting 1	76	76	0
Waving	116	102	-12
Walking	102	108	5.8
Running	162	157	-3
Sitting 2	75	71	-5.3

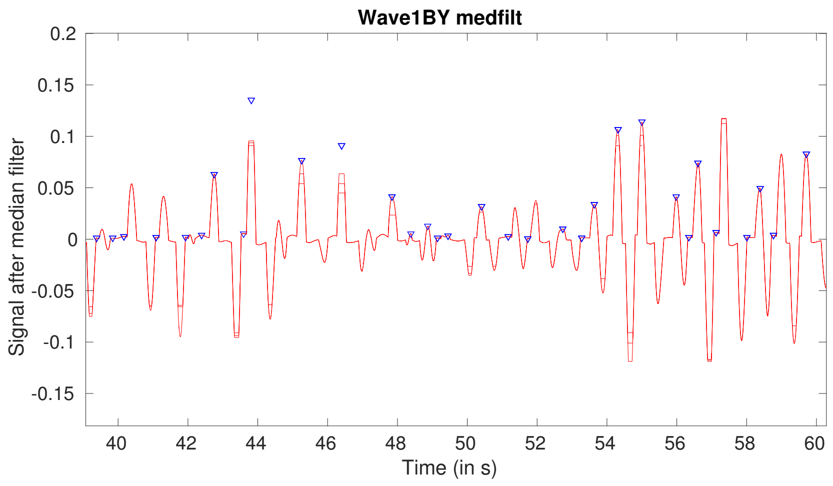

Figure 10.11: Processed “Wave” signal after median filtering showing peaks

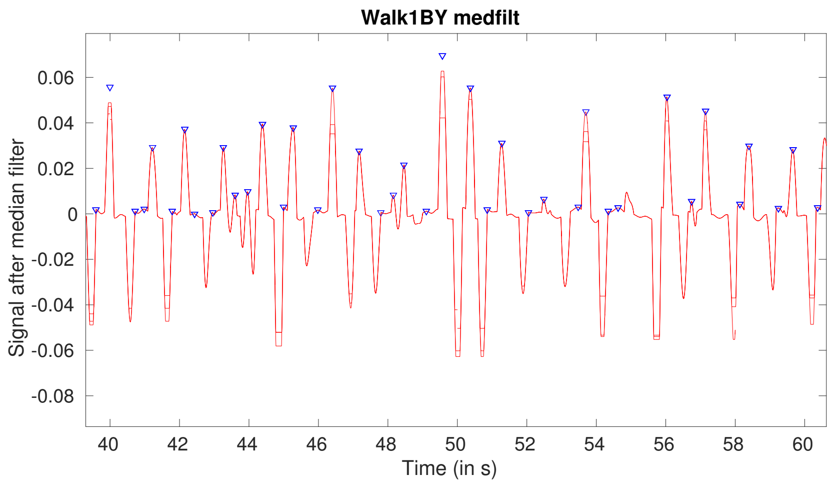

Figure 10.12: Processed “Walk” signal after median filtering showing peaks

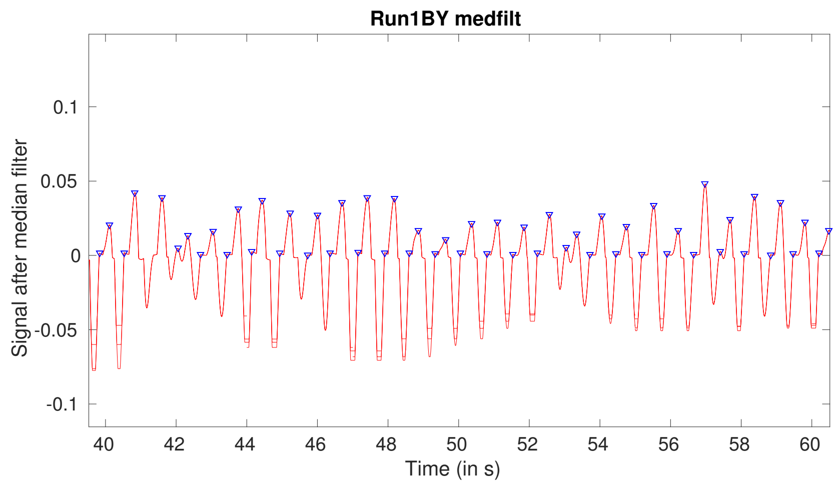

Figure 10.13: Processed "Run" signal after median filtering showing peaks

So, when all the heart rates for each second are calculated, the data is presented in Fig. 10.14. The first 15 seconds of the data from each activity is dropped owing to the small window length and insignificance of heart rates generated at the start.

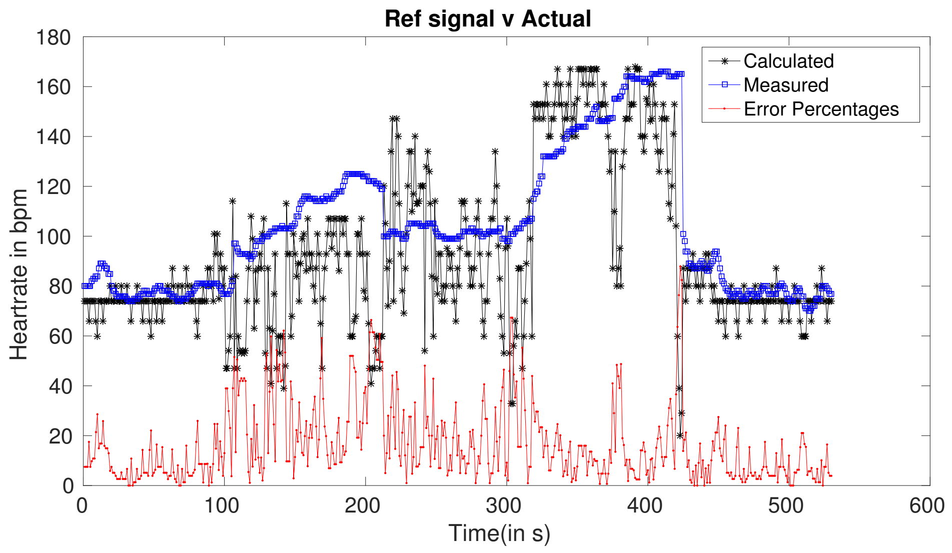

Figure 10.14: Calculated vs measured heart rates

The bar graph in the Fig. 10.15 shows the relative trend on the error percentages of heart rate calculation. From the numbers, we have:

1. 23 % of the measurements have an error less than or equal to 5 %.
2. 72 % of the measurements have an error less than or equal to 20 %.
3. 86.6 % of the measurements have an error less than or equal to 35 %.

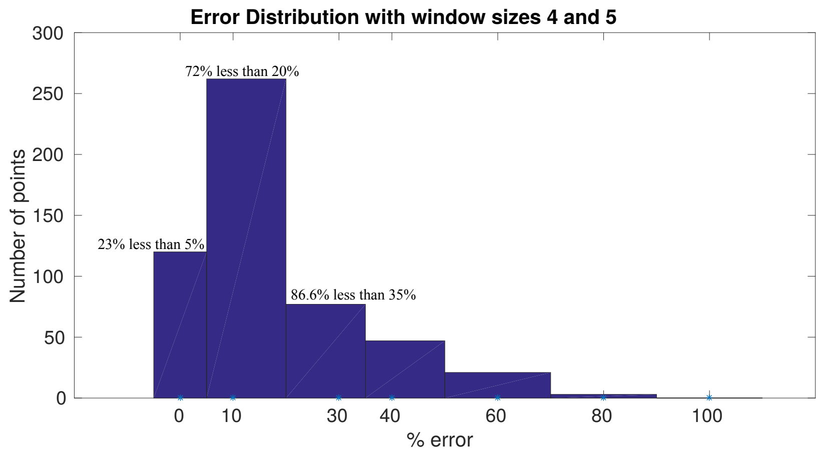

Figure 10.15: Error distribution bar graph when processing window sizes are 4 and 5

10.4 Comparing Different Algorithms

The Tables 10.2 to 10.6 show the comparison of different algorithms' performance at a particular second into a specific activity. To indicate the performance of all the algorithms overall for all activities, the Figs. 10.16 to 10.20 are shown.

Comparing these results with that of the final algorithm clearly show a pattern. The results for physically dormant activities like sitting are accurate with algorithms taking synthetic noise while algorithms taking accelerometers as noise perform good for physically intense activities. The final algorithm tries to cover the deficiencies of both the approaches.

Table 10.2: Heart rates for "LMS with synthetic noise" at the 54th second

Activity	**Reference Heart Rate**	**Measured Heart Rate**	**Error %**
Sitting 1	76	76	0
Waving	116	66	-43
Walking	102	60	-41
Running	162	80	-51
Sitting 2	75	71	-5.3

Table 10.3: Heart rates for "RLS with synthetic noise" at the 54th second

Activity	**Reference Heart Rate**	**Measured Heart Rate**	**Error %**
Sitting 1	76	76	0
Waving	116	74	-36
Walking	102	92	-10
Running	162	101	-38
Sitting 2	75	71	-5.3

Table 10.4: Heart rates for "LMS with accelerometer noise" at the 54th second

Activity	**Reference Heart Rate**	**Measured Heart Rate**	**Error %**
Sitting 1	76	171	125
Waving	116	87	-24
Walking	102	143	40
Running	162	164	1.3
Sitting 2	75	145	93.3

Table 10.5: Heart rates for “RLS with accelerometer noise” at the 54th second

Activity	**Reference Heart Rate**	**Measured Heart Rate**	**Error %**
Sitting 1	76	110	45
Waving	116	78	-34
Walking	102	104	2.1
Running	162	164	1.3
Sitting 2	75	124	64.6

Table 10.6: Heart rates for “ANC with FSLMS” at the 54th second

Activity	**Reference Heart Rate**	**Measured Heart Rate**	**Error %**
Sitting 1	76	55	-27
Waving	116	102	-12
Walking	102	108	5.8
Running	162	157	-3
Sitting 2	75	51	-31.4

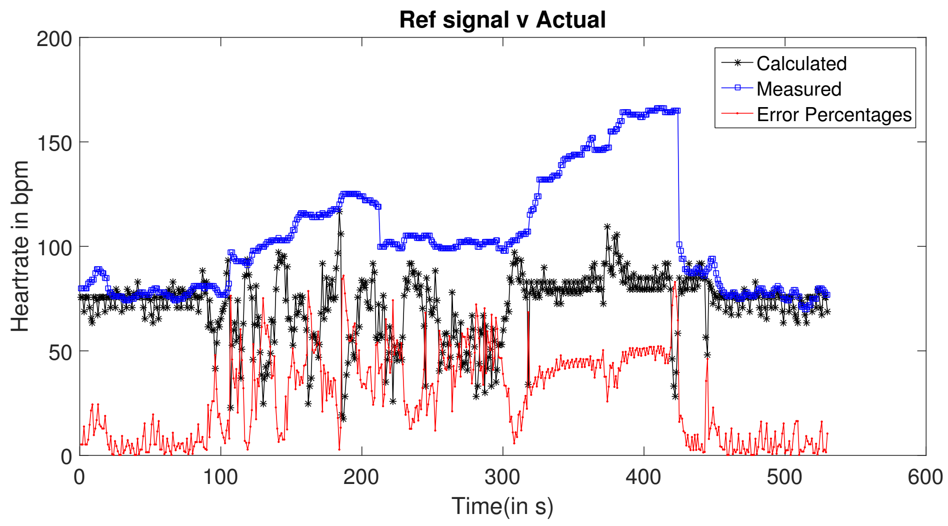

Figure 10.16: Calculated vs measured heart rates for “LMS with synthetic noise”

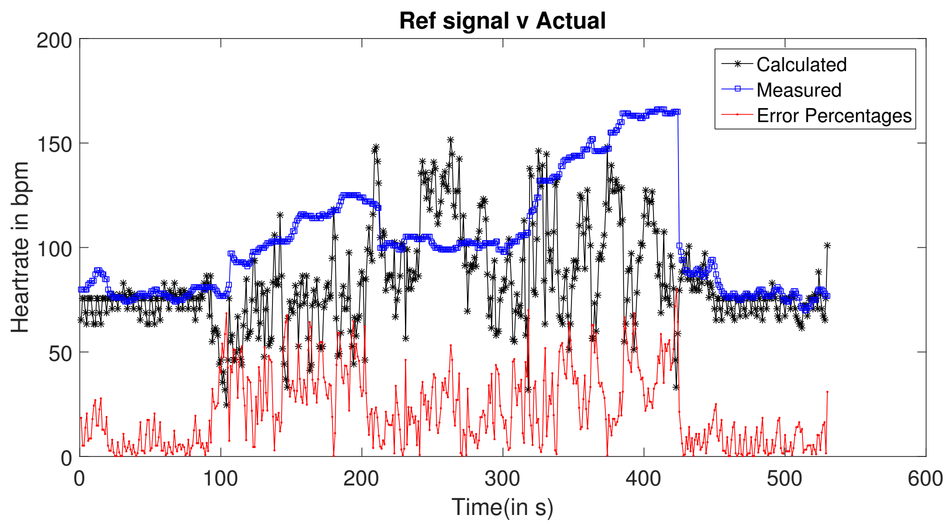

Figure 10.17: Calculated vs measured heart rates for “RLS with synthetic noise”

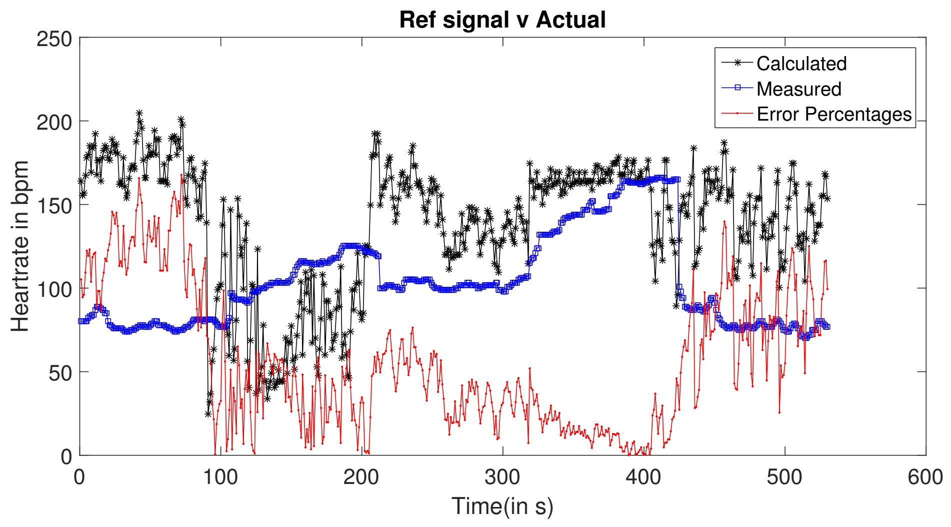

Figure 10.18: Calculated vs measured heart rates for “LMS with accelerometer noise”

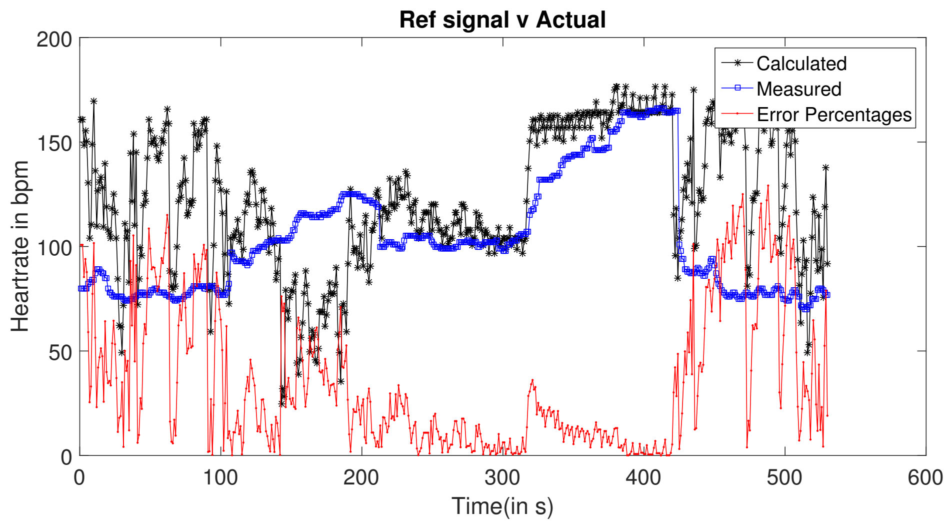

Figure 10.19: Calculated vs measured heart rates for “RLS with accelerometer noise”

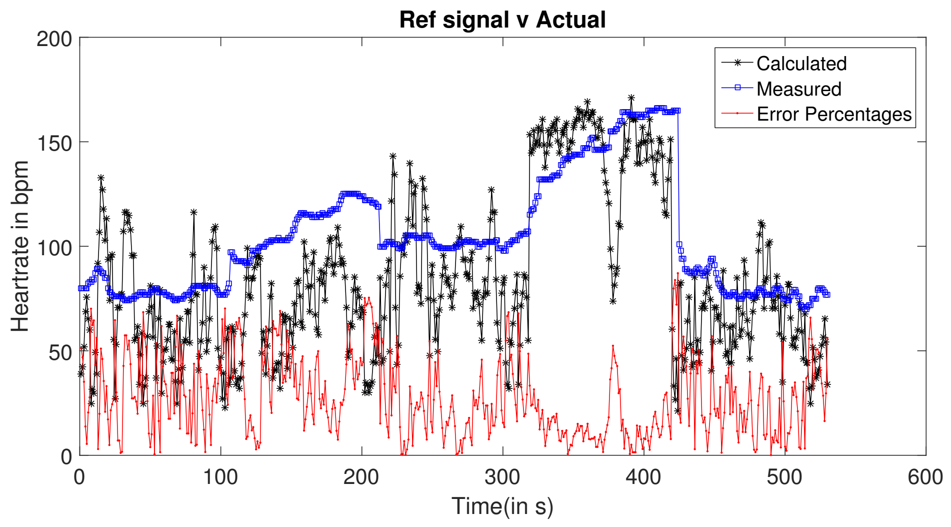

Figure 10.20: Calculated vs measured heart rates for “ANC with FSLMS”

10.5 Results from Auxiliary Data

The final algorithm is applied to the auxiliary data described in [23]. The description of the data set has been given in Chapter 8. The results are shown in Fig. 10.21 and Fig. 10.22. The reference heart rates are calculated by the data provider using a window of size 8 seconds. The subsequence points are calculated using a 6 second overlapping window. That is, if the first heart rate corresponds to the window 1 through 8 seconds, the second point corresponds to the window 3 through 10.

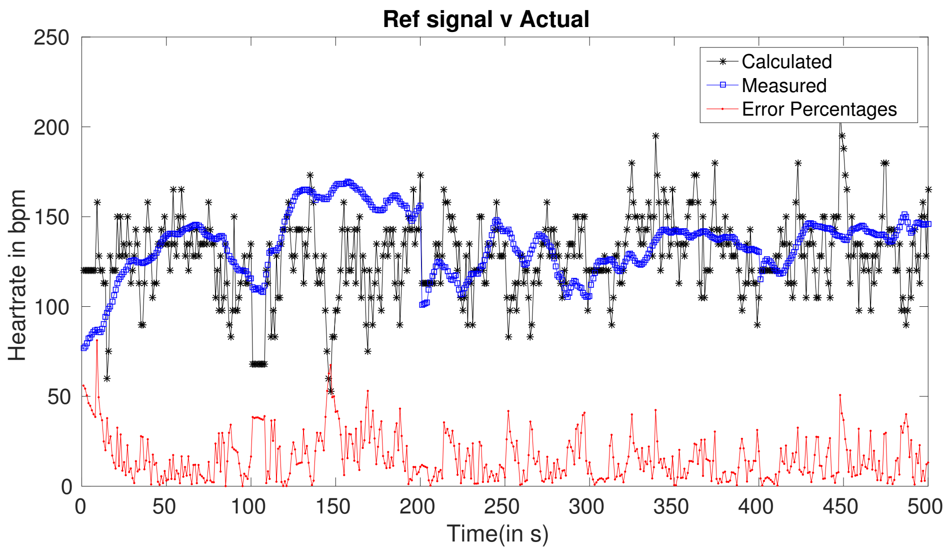

Figure 10.21: Calculated vs measured heart rates for auxiliary data set

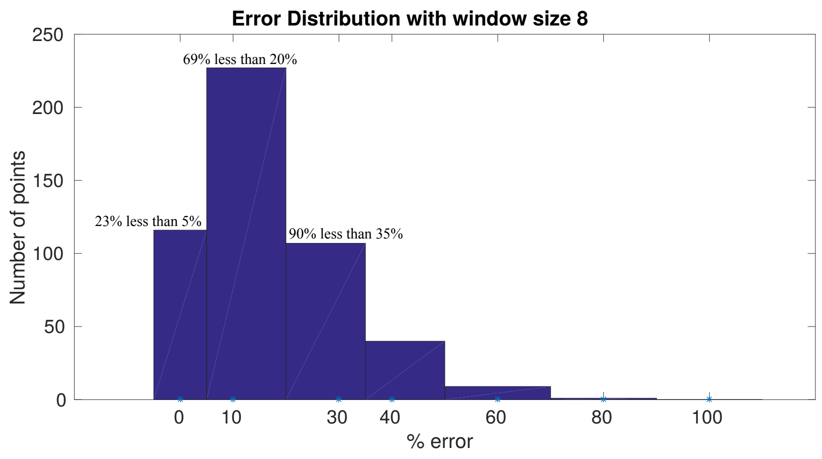

Figure 10.22: Error distribution bar graph with window size 8 for auxiliary data set

11 CONCLUSION

Adaptive filtering algorithms like LMS and RLS are applied to the pertinent problem and the results are carefully analyzed. Different types of noise models are discussed and implemented. The effect of a type of noise model and the ability to reduce motion artifacts is analyzed. A popular ANC technique is adopted to fit the problem of motion artifact reduction.

Although accelerometer based noise model did not prove good for situations where motion artifacts are not dominant, the problems associated with implementing such technique standalone are identified and discussed. A new wrapper algorithm is proposed that overcomes the drawbacks of a single noise model. Finally, automatic heart rate detection is implemented to measure the relative success of the wrapper algorithm. The effects of various window sizes on the calculation of heart rate and the effects of filter orders are also considered and analyzed. The error rates might seem extravagant; But the accuracy comes with a lot of clutter and PPG based sensors offer high portability and ease of use when compared to ECG sensors.

REFERENCES

[1] T. L. Rusch, R. Sankar, and J. E. Scharf, “Signal processing methods for pulse oximetry,” *Computers in Biology and Medicine*, vol. 26, no. 2, pp. 143–159, 1996.

[2] J. W. Severinghaus and J. F. Kelleher, “Recent developments in pulse oximetry,” *Anesthesiology*, vol. 76, no. 6, pp. 1018–1038, 1992.

[3] “The molar extinction coefficient of hbo2 and hb(https://commons.wikimedia.org/wiki/file:fig_1_-_the_molar_extinction_coefficient_of_hbo2_and_hb.png) by zhun310(https://commons.wikimedia.org/wiki/user:zhun310) is licensed under CC BY-SA 3.0 US(https://creativecommons.org/licenses/by-sa/3.0/us/).”

[4] M. R. Ram, K. V. Madhav, K. N. Reddy, E. H. Krishna, and K. A. Reddy, “A novel approach for motion artifact reduction in ppg signals based on AS-LMS adaptive filter,” *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 5, pp. 1445–1457, 2012.

[5] M. J. Hayes and P. R. Smith, “A new method for pulse oximetry possessing inherent insensitivity to artifact,” *IEEE Transactions on Biomedical Engineering*, vol. 48, no. 4, pp. 452–461, 2001.

[6] Z. Da, W. Haitao, and W. Yuqi, “A method of pre-processing photoplethysmographic signal based on adaptive filter for pulse oximeter,” *International Conference on Intelligent Computation Technology and Automation*, vol. 1, pp. 168–170, 2010.

[7] B. Widrow, R. C. Goodlin, J. R. Glover, J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, and J. E. Dong, "Adaptive noise canceling:principles and applications," *Proceedings of IEEE*, vol. 63, no. 12, pp. 1692–1716, 1975.

[8] L. B. Wood and H. H. Asada, "Low variance adaptive filter for cancelling motion artifact in wearable photoplethysmogram sensor signals," *29th Annual International Conference of the IEEE EMBS*, pp. 652–655, 2007.

[9] H.-H. Jiang, H. H. Asada, and P. T. Gibbs, "Active noise cancellation using MEMS accelerometers for motion tolerant wearable biosensors," *26th Annual International Conference of the IEEE EMBS*, vol. 1, pp. 2157–2160, 2004.

[10] P. T. Gibbs, L. B. Wood, and H. H. Asada, "Active motion artifact cancellation for wearable health monitoring sensors using MEMS accelerometers," *Proceedings of SPIE*, vol. 5765, 2005.

[11] B. Wahlberg, "System identification using laguerre models," *IEEE Transactions on Automatic Control*, vol. 36, no. 5, pp. 551–562, 1991.

[12] D. P. Das and G. Panda, "Active mitigation of nonlinear noise processes using a novel filtered-s LMS algorithm," *IEEE Transactions on Speech and Audio Processing*, vol. 12, no. 3, pp. 313–322, 2004.

[13] K. Nakajima, T. Tamura, and H. Miike, "Monitoring of heart and respiratory rates by digital filtering technique," *Medical Engineering and Physics*, vol. 18, no. 5, pp. 365–372, 1996.

[14] C. Yu, Z. Liu, M. Thomas, T. R. Andrew, and R. Jaques, “A method for automatic identification of reliable heart rates calculated from ECG and PPG waveforms,” *American Medical Informatics Association*, vol. 13, no. 3, pp. 309–320, 2006.

[15] B. Widrow, “Adaptive filters,” in *Aspects of Network and System Theory* (R. E. Kalman and N. DeClaris, eds.), pp. 563–586, New York: Holt, Rinehart, Winston, Inc, 1971.

[16] C. F. Gauss, *Theoria combinationis observationum erroribus minimis obnoxiae*, vol. 2. Göttingen: Apud Henricum Dieterich, 1825.

[17] A. R. Relente and L. G. Sison, “Characterization and adaptive filtering of motion artifacts in pulse oximtery using accelerometers,” *Proceedings of the Second Joint EMBS/BMES Conference*, vol. 2, pp. 1769–1770, 2002.

[18] P. Strauch and B. Mulgrew, “Active control of non-linear noise processes in a linear duct,” *IEEE Transactions on Signal Processing*, vol. 46, no. 9, pp. 2404–2412, 1998.

[19] J. C. Patra and R. N. Pal, “A functional link artificial neural network for adaptive channel equalization,” *Elsevier Signal Processing*, vol. 43, no. 2, pp. 181–195, 1995.

[20] J. C. Patra, R. N. Pal, B. N. Chatterji, and G. Panda, “Identification of nonlinear dynamic system using functional link artificial neural network,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 29, no. 2, pp. 254–262, 1999.

[21] N. Sadegh, “A perception network for functional identification and control of nonlinear sytems,” *IEEE Transactions on Neural Networks*, vol. 4, no. 6, pp. 982–988, 1993.

[22] “The de-identified PPG data for this thesis has been provided by the Optical Sensing Lab within the Department of Biomedical Engineering at Texas A&M University.”

[23] Z. Zhang, Z. Pi, and B. Liu, “TROIKA: A general framework for heart rate monitoring using wrist-type photoplethysmographic signals during intensive physical exercise,” *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 2, pp. 522–531, 2015.

APPENDIX A

PSEUDO CODE

LMS Algorithm with Synthetic Noise

If ‘Data’ is the PPG recording, ‘filter’ is the array of weights and ‘filterorder’ is the order of the LMS filter, then we can write the algorithm for LMS as follows

```
 1: procedure LMS_Synth(Data)
 2:     $Freq\_data \leftarrow FFT(Data)$
 3:     $Freq\_data(0.2 \leq frequency \leq 4) \leftarrow 0$
 4:     $Noise \leftarrow IFFT(Freq\_data)$
 5:     for iterations do
 6:         for i = 1 to length(Noise) do
 7:             $\hat{h} \leftarrow filter * Noise(i\ to\ i+filterorder)$
 8:             $error(i) \leftarrow Data(i) - \hat{h}$
 9:             $filter \leftarrow filter + learningrate * error(i) * Noise(i\ to\ i+filterorder)$
10:     $MAfree \leftarrow BandPass(error)$
```

RLS Algorithm with Synthetic Noise

If ‘Data’ is the PPG recording, ‘filter’ is the array of weights and ‘filterorder’ is the order of the RLS filter and I_{matrix} is the Identity Matrix, then we can write the algorithm for RLS as follows

```
1: procedure RLS_Synth(Data)
2:     $Freq\_data \leftarrow FFT(Data)$
3:     $Freq\_data(0.2 \leq frequency \leq 4) \leftarrow 0$
4:     $Noise \leftarrow IFFT(Freq\_data)$
5:     $R \leftarrow 10^5 * I_{matrix}$
6:     for iterations do
7:         for i = 1 to length(Noise) do
8:             $\hat{h} \leftarrow filter*Noise(i\ to\ i+filterorder)$
9:             $error(i) \leftarrow Data(i)-\hat{h}$
10:            $Z \leftarrow R * Noise(i\ to\ i+filterorder)$
11:            $q \leftarrow Noise(i\ to\ i+filterorder)^T *Z$
12:            $v \leftarrow \frac{1}{1+q}$
13:            $Z_t \leftarrow v * Z$
14:            $filter \leftarrow filter + error(i)*Z_t$
15:            $R \leftarrow R - Z_t^T *Z$
16:     $MAfree \leftarrow BandPass(error)$
```

LMS Algorithm with Accelerometer Noise

If Accel is the Accelerometer data, then we can write the algorithm for LMS as follows

```
1: procedure LMS_Accel(Data,Accel)
```

```
2:      for iterations do
3:          for i = 1 to length(Accel) do
4:              $\hat{h} \leftarrow filter*Accel(i\ to\ i{+}filterorder)$
5:              $error(i) \leftarrow Data(i)\text{-}\hat{h}$
6:              $filter \leftarrow filter + learningrate*error(i)*Accel(i\ to\ i{+}filterorder)$
7:      $MAfree \leftarrow BandPass(error)$
```

RLS Algorithm with Accelerometer Noise

If Accel is the Accelerometer data, then we can write the algorithm for RLS as follows

```
1:  procedure RLS_Accel(Data,Accel)
2:      $R \leftarrow 10^5 * I_{matrix}$
3:      for iterations do
4:          for i = 1 to length(Accel) do
5:              $\hat{h} \leftarrow filter*Accel(i\ to\ i{+}filterorder)$
6:              $error(i) \leftarrow Data(i)\text{-}\hat{h}$
7:              $Z \leftarrow R * Accel(i\ to\ i{+}filterorder)$
8:              $q \leftarrow Accel(i\ to\ i{+}filterorder)^T *Z$
9:              $v \leftarrow \frac{1}{1+q}$
10:             $Z_t \leftarrow v * Z$
11:             $filter \leftarrow filter + error(i)*Z_t$
```

```
12:             $R \leftarrow R - Z_t^T * Z$
13:         $MAfree \leftarrow BandPass(error)$
```

ANC with FSLMS

If ‘Accel’ is the Accelerometer data and ‘P’ is the expansion order of the FLANN and all other notations remain the same as above, we can write the algorithm as follows

```
1:  procedure ANC(Data,Accel)
2:      for i = 1 to P do
3:          $Exp\_noise \leftarrow [Exp\_noise sin(i * Accel) cos(i * Accel)]$
4:      for iterations do
5:          for i = 1 to length(Noise) do
6:              $\hat{h} \leftarrow filter*Exp\_noise(i\ to\ i+P)$
7:              $error(i) \leftarrow Data(i)-\hat{h}$
8:              $filter \leftarrow filter + learningrate*error(i)*Noise(i\ to\ i+P)$
9:      $MAfree \leftarrow BandPass(error)$
```

Final Algorithm

If Proc_size is the processing size of each activity, we can write the algorithm as follows

```
1:  procedure FIN(Data,Accel)
2:      for i = 1 to length(Data)-Proc_size do
```

```
3:    if i==1 then
4:        maximum_var=variance(Accel(1 to Proc_size+1))
5:    if variance(Accel(i to i+Proc_Size))<0.33*maximum_var then
6:        LMS_Synth(Data(i to i+Proc_Size))
7:    else
8:        if (variance(Accel(i to i+Proc_Size))>maximum_var then
9:            maximum_var ←variance(Accel(i to i+Proc_Size))
10:       ANC(Data(i to i+Proc_Size),Accel(i to i+Proc_Size))
```

IFFT represents Inverse Fast Fourier Transform and $\{.\}^T$ represents the transpose of a matrix.