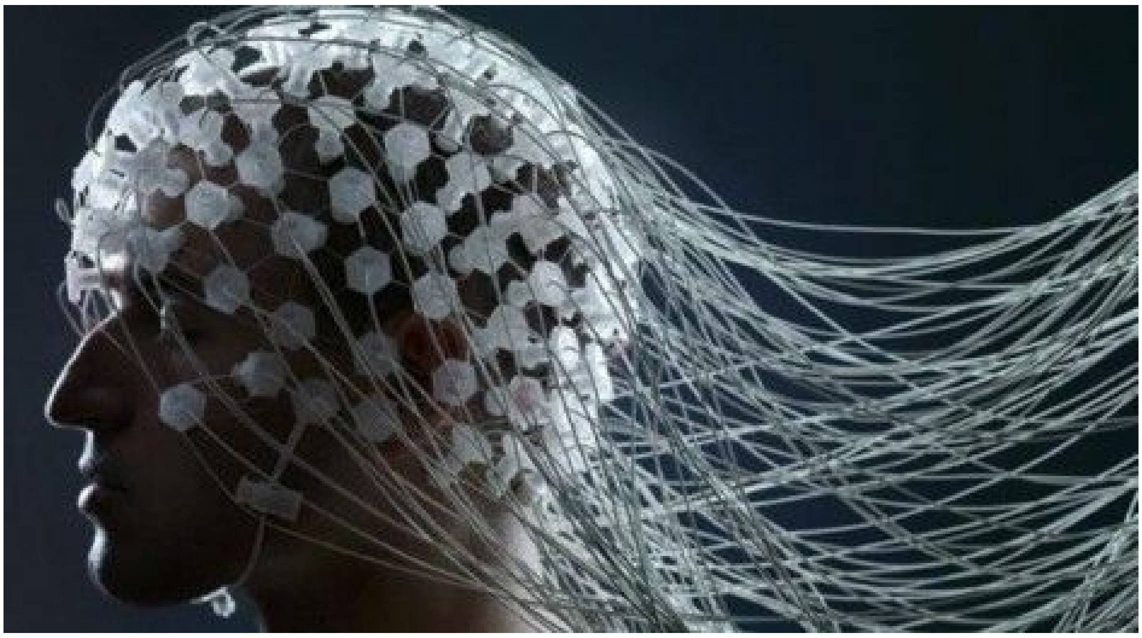
Removal Of Ocular Artifacts From Electro-encephalogram by Adaptive Filtering

ECE 516 PROJECT REPORT



Giri Balasubramanian
University of Illinois at Chicago
UIN: 654309242
Dec 15, 2017

Abstract

The electro-encephalogram (EEG) is useful for clinical diagnosis and in biomedical research. EEG signals, however, especially those recorded from frontal channels, often contain strong electro-oculogram (EOG) artifacts produced by eye movements. Existing regression-based methods for removing EOG artifacts require various procedures for pre-processing and calibration that are inconvenient and time consuming. The paper describes a method for removing ocular artifacts based on adaptive filtering. Results from experimental data demonstrate that the method is easy to implement and stable, converges fast and is suitable for on-line removal of EOG artifacts.

Contents

List of Figures

1 Introduction

EEG (Electro-encephalography) is a technique for measuring or recording electrical activity of the brain caused by firing of neurons in the brain. This measurement is done using electrodes as shown in fig:1

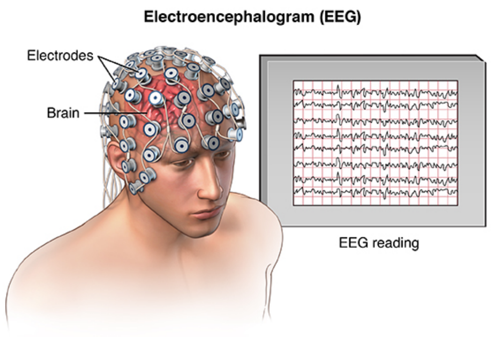


Figure 1: Electro-encephalography

Getting the right EEG is important cause it is useful for:-

1. Clinical diagnosis and treatment: ALS, Seizure, Neurofeedback Therapy.
2. Biomedical Research: Neural Encoding and more.

The eye forms an electric dipole, where the cornea is positive and the retina is negative. When the eye moves (saccade, blink or other movements), the electric field around the eye changes, producing an electrical signal (see fig:2) known as the electro-oculogram (EOG).

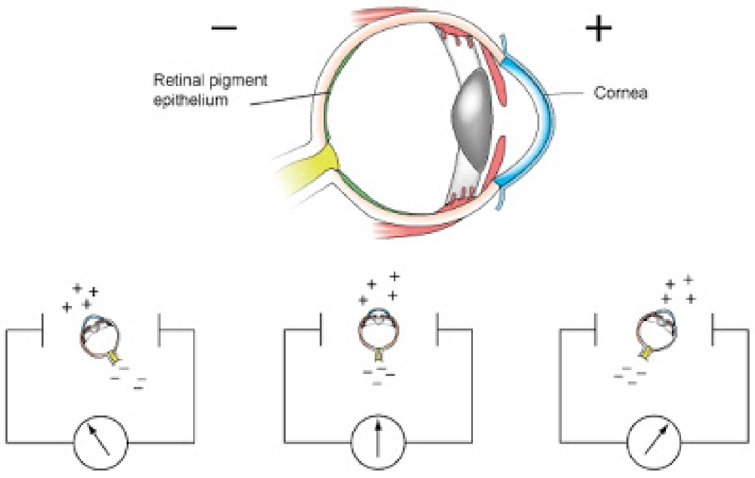


Figure 2: Ocular Artifact

As this signal propagates over the scalp, it appears in the recorded electro-encephalogram (EEG) as noise or artifacts that present serious problems in EEG interpretation and analysis.

Previous studies (CROFT and BARRY, 2000) have shown that there are at least two kinds of EOG artifact to be removed:-

1. Those produced by the vertical eye movement; called VEOG.

2. Those produced by the horizontal eye movement; called HEOG.

To correct or remove ocular artifacts from EEG, many regression-based techniques have been proposed, including Simple time-domain regression (VERLEGER et al., 1982; GRATTON et al., 1983), Multiple-leg time-domain regression (KENEMANS et al., 1991) and Regression in the frequency domain (WHITTON et al., 1978; WOESTENBURG et al., 1983). Independent Component Analysis(JUNG et al., 2000). In all these regression-based approaches, calibration trials are first conducted to determine the transfer coefficients between the EOG channels and each of the EEG channels (CROFT and BARRY, 2000). These coefficients are then used later in the 'correction phase' to estimate the EOG component in the EEG recording for removal by subtraction. These methods have the following characteristics:-

1. Off-line analysis and processing of data.

2. Data needs to be collected from a sufficiently large number of channels.

3. It works only when the correct noise components are identified.

When the applications require real-time removal of ocular artifacts, or when the calibration trials cannot be conducted owing to various constraints, the methods described above become unsuitable. An example for such application is - researchers are developing methods for accessing a pilot's functional state during flight as shown in fig:2, so that adaptive aid can be provided in the case of mental overload(WILSON, 2002; HANKINS and WILSON, 1998). As the pilot's activity is accompanied by a significant amount of eye movement, either voluntarily or involuntarily, EOG contamination is a serious problem in EEG-based analysis.

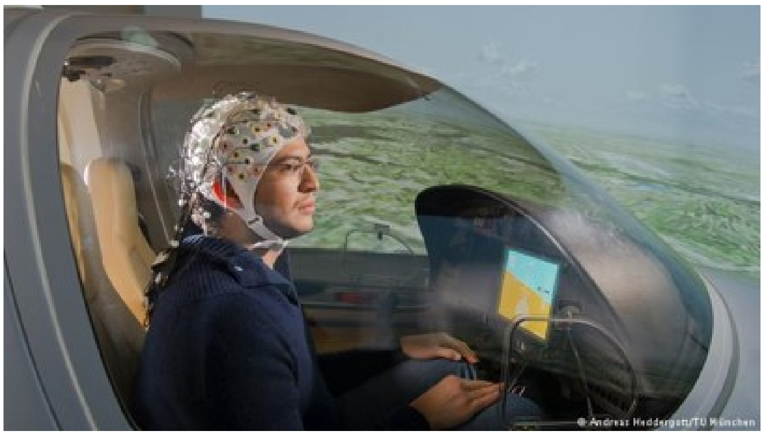


Figure 3: EEG based analysis to access pilot's functional state during flight

2 Motivation

The problem of EEG contamination cause of EOG artifacts was the motivation for my project. To resolve this, the algorithm presented in the paper - ”Removal of Ocular artifacts from electro-encephalogram by adaptive filtering” by P. He, G. Wilson and C. Russell was used. The paper describes a noise cancellation method based on adaptive filtering (WIDROW et al., 1975; HAYKIN, 1996) to remove ocular artifacts from EEG. This method was particularly suitable for my applications because it does not require calibration trials, and the EOG artifacts can be removed on-line.

Note : Due to non feasibility of right equipment and unavailability of one reference signal, a noise canceler with only one reference inputs is implemented in my project.

3 Theory

Fig:4 shows the block diagram of the noise canceller used in the application.

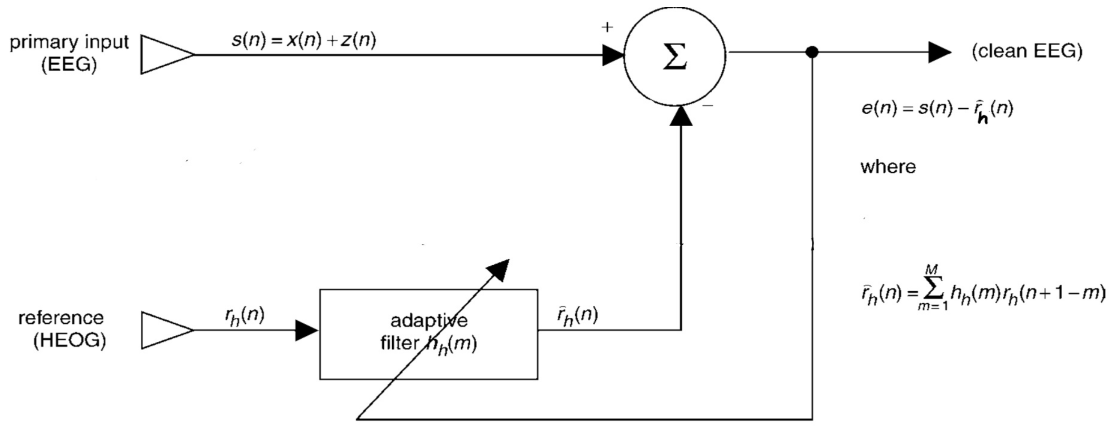


Figure 4: Block diagram of EOG noise canceller using adaptive filtering with two reference inputs

The primary input to the system is the EEG signal s(n), picked up by a particular electrode. This signal is modelled as a mixture of a true EEG x(n) and a noise component z(n) which represents the HEOG artifact in this application. $r_h(n)$ is the reference input HEOG. Signal $r_h(n)$ is correlated, in some unknown way with the noise component z(n) in the primary input. $h_h(m)$ represents a finite impulse response (FIR) adaptive filters of order M.

3.1 Mathematical Formulation

The output of the noise canceller is the error signal e(n), where

$$\begin{aligned} e(n) &= s(n) - \hat{r}_h(n) \\ &= x(n) + [z(n) - \hat{r}_h(n)] \end{aligned} \tag{1}$$

where

$$r_h(n) = \sum_{m=1}^{M} h_h(m) r_h(n+1-m) \tag{2}$$

Under the assumption that x is a zero-mean stationary random signal that is uncorrelated with z and r_h, the expected value of e^2 can be calculated as

$$\begin{aligned} \mathbb{E}[e^2] &= \mathbb{E}[(x + z - \hat{r}_h)^2] \\ &= \mathbb{E}[x^2] + \mathbb{E}[(z - \hat{r}_h)^2] \end{aligned} \tag{3}$$

The goal of the noise canceller is to produce an output signal e(n) that is as close to x(n) as possible, by adjusting the filter coefficient $h_h(m)$. Statistically, this requires a minimization of $\mathbb{E}[(z - \hat{r}_h)^2]$. As $E[x^2]$ is not affected by the adjustment of the filter coefficients, minimizing $\mathbb{E}[(z - \hat{r}_h)^2]$ is equivalent to minimizing $E[e^2]$.

3.2 Recursive Relationship

This adjustment of the filter coefficients is implemented using Recursive least-squares (RLS) algorithm, because it has the following characteristics:-

1. Does not require calibration trials.
2. Superior stability.
3. Fast convergence

With the interest of recursive relationship in mind, we reformulate the above equations in terms of samples available at a particular time t_n.

Assuming at time t_n, we have obtained the following samples: $s(i), r_h(i)$ and $e(i)$, for $i = 1, 2, ..., n$, we form the following target function e(n) to minimize:

$$\begin{aligned}\varepsilon(n) &= \sum_{i=M}^{n} \lambda^{n-i} e^2(i) \\ &= e^2(n) + \lambda e^2(n-1) + ... + \lambda^{n-M} e^2(M)\end{aligned} \tag{4}$$

where $0 < \lambda \leq 1$ is called the forgetting factor, and

$$e(i) = s(i) - \sum_{m=1}^{M} h_h(m) r_h(i+1-m) \tag{5}$$

In a recursive scenario, it would be impossible to get the entire statistic of the data. Hence we minimize $\varepsilon(n)$ instead of $\mathbb{E}[e^2]$, i.e. we simply use the sample mean to approximate the expected value. In addition, by introducing the forgetting factor λ, the algorithm can also be applied to a random process that is not strictly stationary.

The filter parameter $h_h(m)$, for $m = 1, 2, ..., M$, that minimizes e(n) can be obtained by solving the following equations (a total of M equations):

$$\begin{aligned}\frac{\partial \varepsilon(n)}{\partial h_h(m)} &= 2 \sum_{i=M}^{n} \lambda^{n-i} e(i) \frac{\partial e(i)}{\partial h_h(m)} \\ &= -2 \sum_{i=M}^{n} \lambda^{n-i} e(i) r_h(i+1-m) = 0\end{aligned} \tag{6}$$

for $m = 1, 2, ..., M$

After substituting for e(i), the above equation can be represented in the matrix form as

$$\underline{H} = [R(n)]^{-1} . \underline{P}(n) \tag{7}$$

where R(n) is $(M \text{x} M)$ square matrix, $\underline{\text{H}}$ and $\underline{\text{P}}$(n) are column vectors of dimension M and they are defined as follows -

$$R(n)(j,k) = \sum_{i=M}^{n} \lambda^{n-i} r_h(i+1-j) r_h(i+1-k) \tag{8}$$

$$\underline{P}(n)(j) = \sum_{i=M}^{n} \lambda^{n-i} s(i) r_h(i+1-j) \tag{9}$$

$$\underline{H} = [h_h(1) h_h(2) ... h_h(M)]^T \tag{10}$$

Calculating the filter coefficients by directly solving (7) involves matrix inversion, which is computationally expensive as its computation complexity for a matrix of order n is $n^{2.373}$. The computation load can be greatly reduced by using a recursive least-squares (RLS) algorithm that calculates the filter coefficients by implementing (7) recursively in n. From (8)-(10), we can show that

$$R(n) = \lambda R(n-1) + \underline{r}(n)\underline{r}(n)^T \tag{11}$$

$$\underline{P}(n) = \lambda \underline{P}(n-1) + s(n)\underline{r}(n) \tag{12}$$

where

$$\underline{r}(n) = [r_h(n)r_h(n-1)...r_h(n+1-M)]^T \tag{13}$$

Now using Matrix inversion Lemma on 11 we can obtain the following recursive relationship:

$$[R(n)]^{-1} = \lambda^{-1}[R(n-1)]^{-1} - \lambda^{-1}\underline{K}(n)\underline{r}(n)^T[R(n-1)]^{-1} \tag{14}$$

where

$$\underline{K}(n) = \frac{[R(n-1)]^{-1}\underline{r}(n)}{\lambda + \underline{r}(n)^T[R(n-1)]^{-1}\underline{r}(n)} \tag{15}$$

Finally, by substituting (14) and (12) into (7) and using (15), we obtain the following formula for updating the filter coefficients:

$$\underline{H}(n) = \underline{H}(n-1) + \underline{K}(n)e\left(\frac{n}{n-1}\right) \tag{16}$$

where $\underline{H}(n) = [R(n)]^{-1}.\underline{P}(n)$ are the filter coefficients at time n and $\underline{H}(n-1) = [R(n-1)]^{-1}.\underline{P}(n-1)$ are the filter coefficients at time (n-1); and

$$e\left(\frac{n}{n-1}\right) = s(n) - \underline{r}(n)^T\underline{H}(n-1) \tag{17}$$

3.3 RLS Algorithm

Based on the above formulations the RLS algorithm can be implemented in the following steps:

1. Set Initial values:

$$\begin{aligned} \underline{H}(n-1) &= 0 \\ [R(n-1)]^{-1} &= I/\sigma \end{aligned} \tag{18}$$

   where I is a $(M\text{x}M)$ identity matrix and $\sigma = 0.01$. Starting from $n = M$, for every set of new samples $s(n)$ and $r_h(n)$, perform the following steps:

2. Form $\underline{r}(n)$ based on (13);
   Calculate $\underline{K}(n)$ using (15);
   Calculate $e(n/n-1)$ using(17);
   Calculate $\underline{H}(n)$ using (16);
   Update $[R(n)]^{-1}$ using(14);
   Calculate

$$\begin{aligned} e(n) &= s(n) - \underline{r}(n)^T \underline{H}(n); \\ n &= n+1 \end{aligned} \tag{19}$$

   go back to (2).

4 Experiment

The paper on "Removal of Ocular Artifacts from Electro-encephalogram by adaptive filtering" by HE. et al, claims the following:-

1. This method is easy to implement and stable, converges fast and is suitable for on-line removal of EOG artifacts.
2. When the filtering algorithm is applied to an EEG recorded at a remote site, e.g. O_2, that contains very few EOG artifacts, the filters are basically shut down and the original EEG simply passes the system without visible changes.
3. The exact value of λ - the forgetting factor is not critical to the performance of the algorithm.
4. The performance of the adaptive filter is not sensitive to the choice of M - the filter order.
5. As the filter order increases, Mean square error (MSE) decreases.

In this project, I tried to test the above mentioned claims using the algorithm discussed earlier. Data published by Manousos Klados and P. Bamidis was used ($https://data.mendeley.com/datasets/wb6yvr725d/3$) for this experiment. 5601 EEG data samples were recorded from 3 sites F_z, C_z, O_2 positioned according to the international 10-20 Electrode System (JASPER, 1958) shown in fig:5. Although the algorithm for noise cancelling described above is suitable for real-time applications, in this experiment, all the data was recorded first and then applied the algorithm off-line. Real-time filtering was simulated by sequential feeding of the sample data to the program. The execution is done is MATLAB. The model parameters are set as $M = 3$ and $\lambda = 0.9999$, unless stated otherwise.

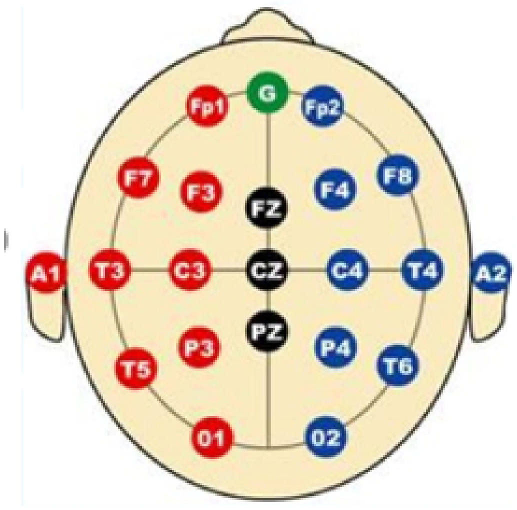


Figure 5: International 10-20 Electrode System

5 Result

The experiment was divided into 5 parts, one for each claim. Following are the results of each experiment.

5.1 Raw data

The data used for this experiment is shown in plot: 6.

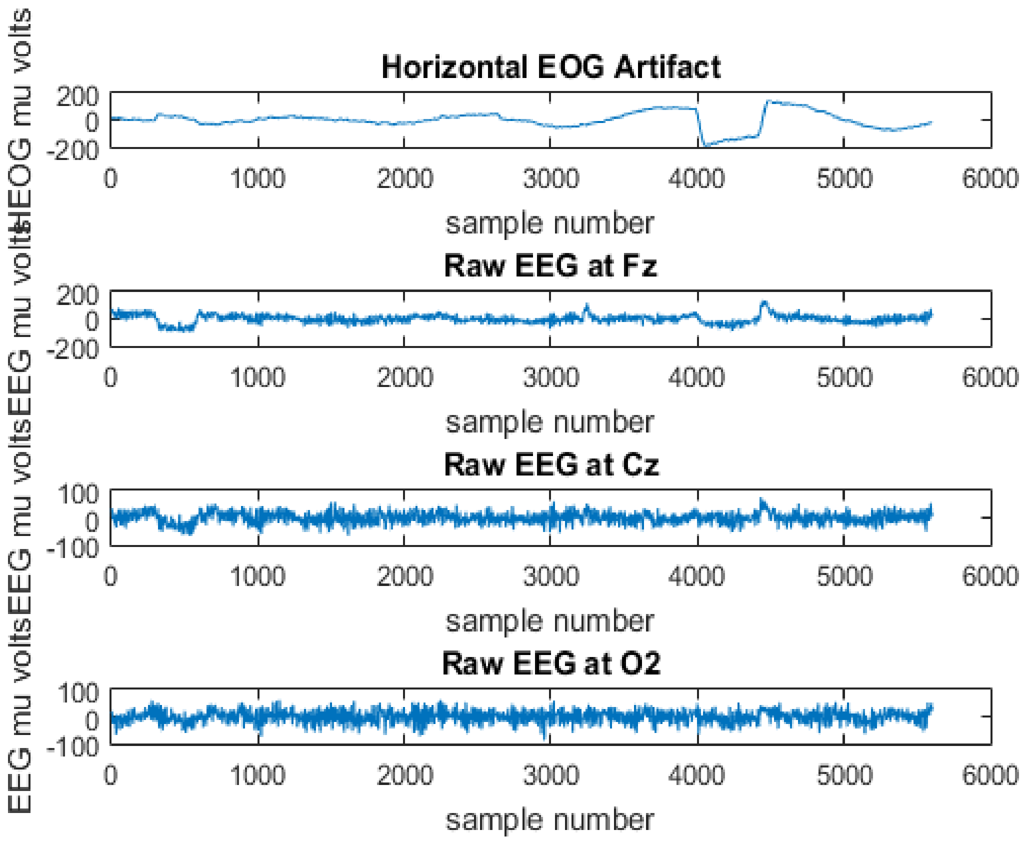


Figure 6: Plot for Raw EEG data and HEOG

5.2 Test for Claim 1

Here the claim - "This method is easy to implement and stable, converges fast and is suitable for on-line removal of EOG artifacts." was tested. The code for this is shown at the end, under subsection "B.Claim 1".

To test this claim, the adaptive filtering algorithm was applied to the EEG data from electrode at F_z position. The result for this is shown in plot: 7

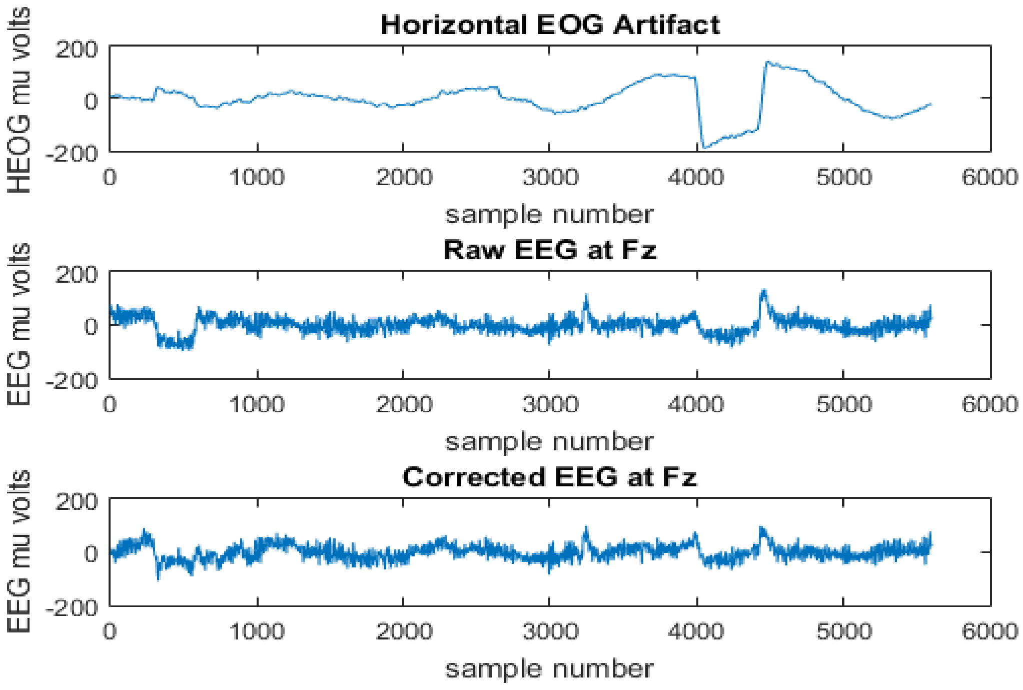


Figure 7: Adaptive Filtering Applied to EEG data from electrode at F_z position

As shown, the method is easy to implement and stable, converges fast and is suitable for online removal of EOG artifacts. Hence Claim 1 is verified.

5.3 Test for Claim 2

Here the second claim - "When the filtering algorithm is applied to an EEG recorded at a remote site, e.g. O_2, that contains very few EOG artifacts, the filters are basically shut down and the original EEG simply passes the system without visible changes." was tested. The code for this is shown at the end, under subsection "B.Claim 2".

To test this claim, the adaptive filtering algorithm was applied to the EEG data from electrode at the remote sites C_z and O_2 position. The result for this is shown in plot: 8 and 9

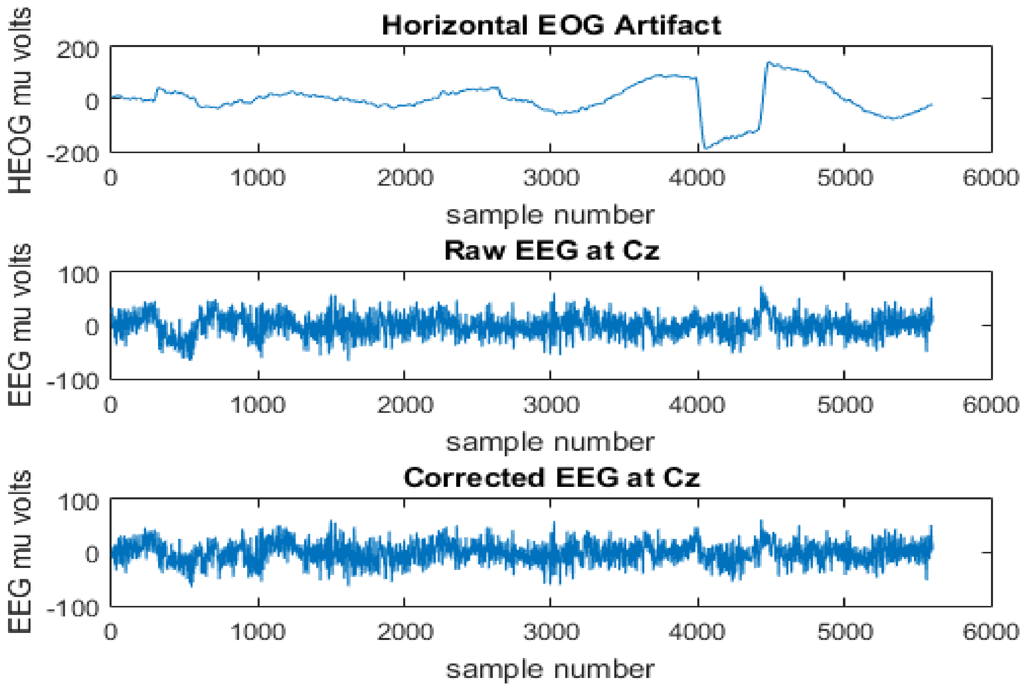


Figure 8: Adaptive Filtering Applied to EEG data from electrode at C_z position

Here, except the data at each position, the algorithm and the model parameters were kept the same.

Thus by comparing these results with the filtering result at F_z, it can verified that, when the filtering algorithm is applied to an EEG recorded at a remote site, e.g. O_2, that contains very few EOG artifacts, the filters are basically shut down and the original EEG simply passes the system without visible changes. Hence Claim 2 is verified.

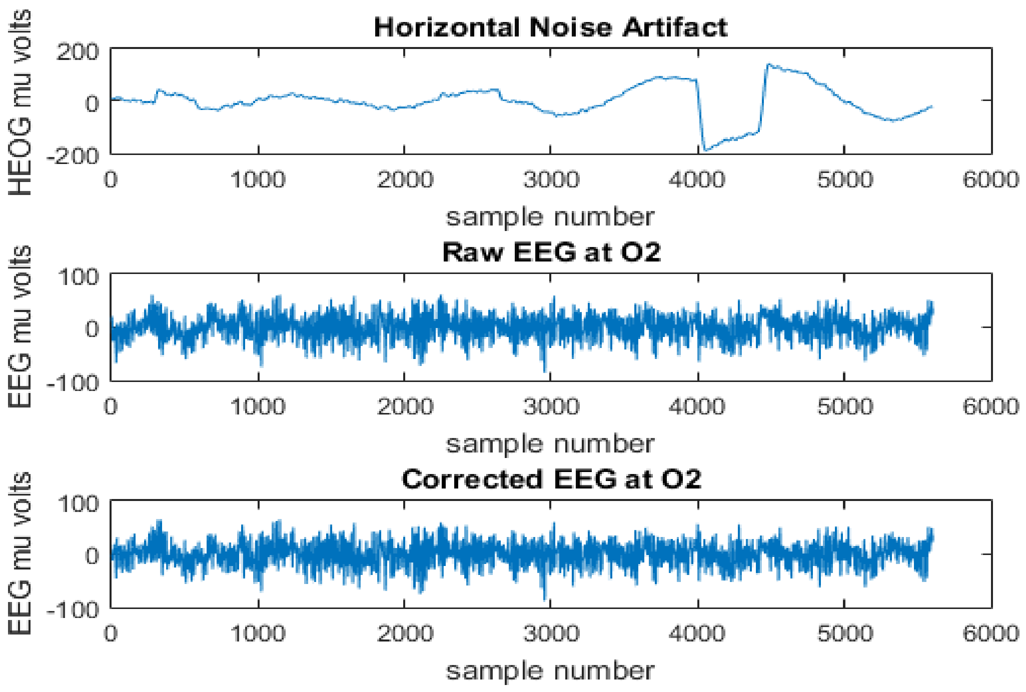


Figure 9: Adaptive Filtering Applied to EEG data from electrode at O_2 position

5.4 Test for Claim 3

Here the claim - ”The exact value of λ - the forgetting factor is not critical to the performance of the algorithm.” was tested. The code for this is shown at the end, under subsection ”B.Claim 3”.

To test this claim, the adaptive filtering algorithm was applied to the EEG data from electrode at F_z position with different values for λ. The different λ values are 0.9999, 0.995 and 1. The result for this is shown in plot: 10

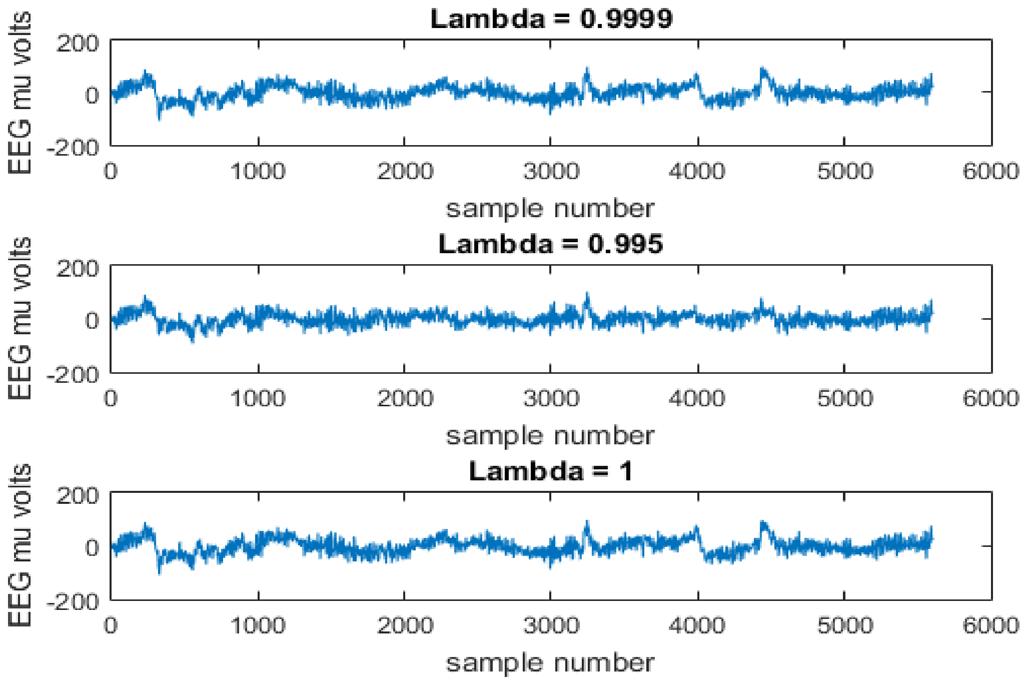


Figure 10: Adaptive Filtering Applied to EEG data with $\lambda = 0.9999$, 0.995 and 1

As shown, the exact value of λ - the forgetting factor is not critical to the performance of the algorithm. Hence Claim 3 is verified.

5.5 Test for Claim 4

Here the claim - ”The performance of the adaptive filter is not sensitive to the choice of M - the filter order.” was tested. The code for this is shown at the end, under subsection ”B.Claim 4”.

To test this claim, the adaptive filtering algorithm was applied to the EEG data from electrode at F_z position with different values for M. The different M values are 3, 1 and 12. The result for this is shown in plot: 11

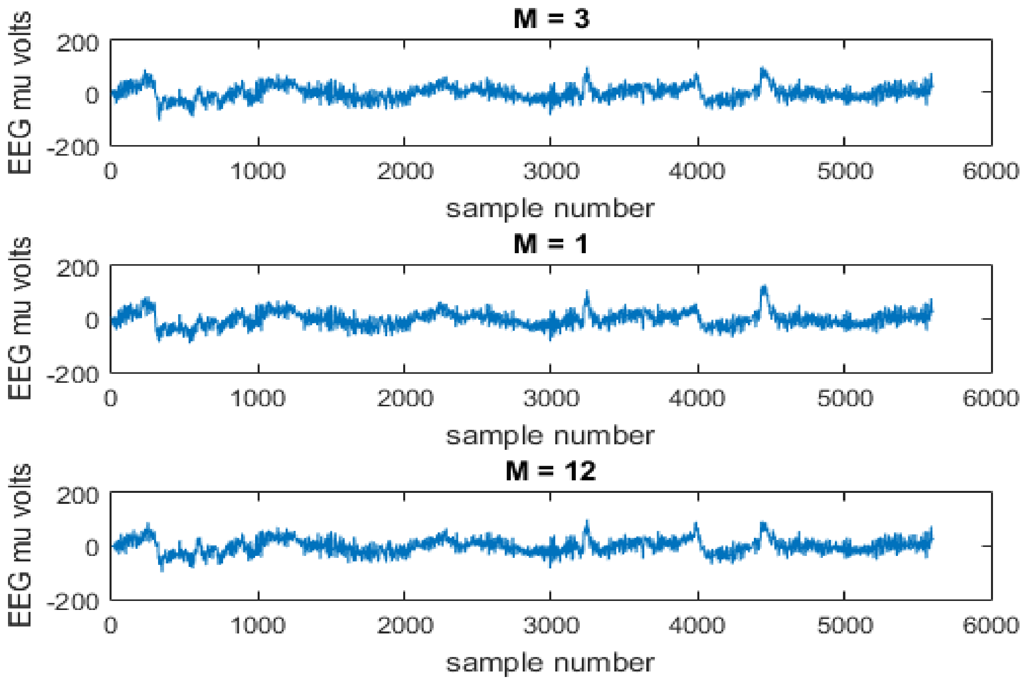


Figure 11: Adaptive Filtering Applied to EEG data with M = 3, 1 and 12

As shown, the performance of the adaptive filter is not sensitive to the choice of M - the filter order. Hence Claim 4 is verified.

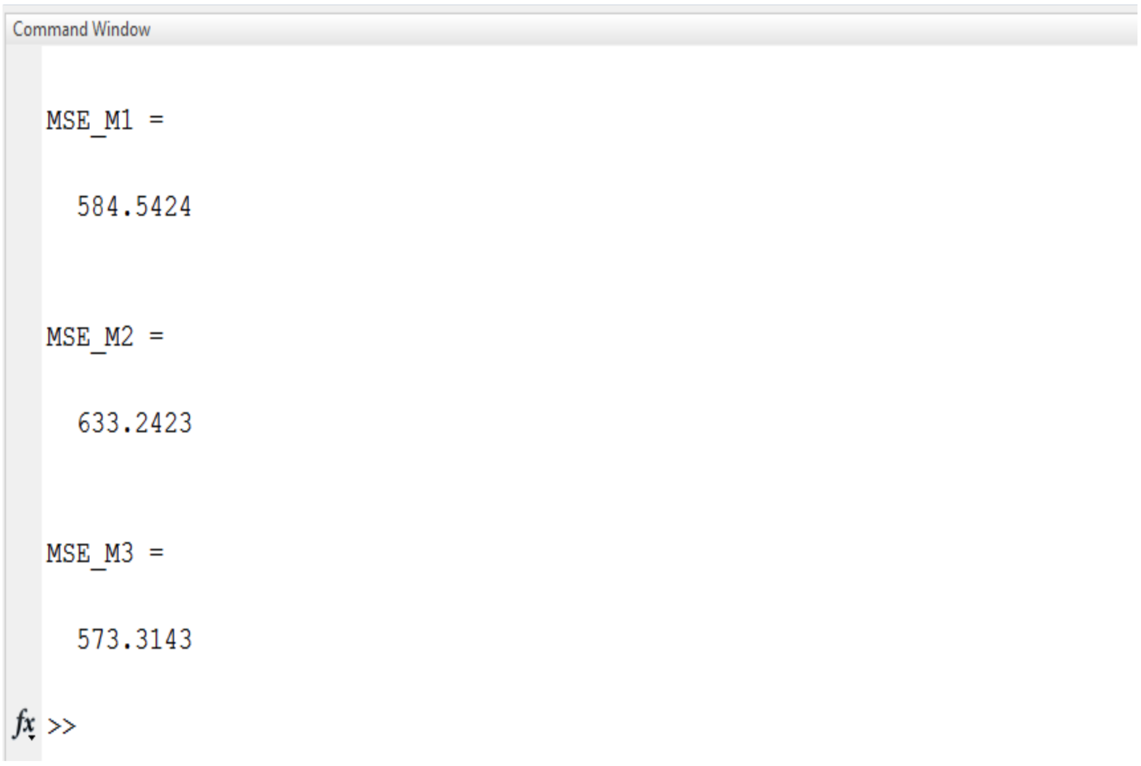
5.6 Test for Claim 5

Here the claim - ”As the filter order increases, Mean square error (MSE) decreases.” was tested. The code for this is shown at the end, under subsection ”B.Claim 5”.

To test this claim, the adaptive filtering algorithm was applied to the EEG data from electrode at F_z position with different values for M. The different M values are 3, 1 and 12. The corresponding MSE was calculated using the formula -

$$MSE = \sum_{i=M}^{5601} \frac{\lambda^{5601-i} e^2(i)}{2000 - M + 1} \tag{20}$$

The resulting MSE is shown in fig: 12



```
MSE_M1 =

  584.5424


MSE_M2 =

  633.2423


MSE_M3 =

  573.3143

fx >>
```

Figure 12: MSE when Adaptive Filtering Applied to EEG data with M = 3, 1 and 12 respectively

As shown, Mean square error (MSE) decreases as filter order increases. Hence Claim 5 is verified.

6 Conclusion

A noise canceller based on Adaptive Filtering was implemented to remove EOG artifacts from corrupted EEG data.

Along with this several properties (mentioned as claims) of the Adaptive filter were verified. These properties are -

1. This method is easy to implement and stable, converges fast and is suitable for on-line removal of EOG artifacts.
2. When the filtering algorithm is applied to an EEG recorded at a remote site, e.g. O_2, that contains very few EOG artifacts, the filters are basically shut down and the original EEG simply passes the system without visible changes.
3. The exact value of λ - the forgetting factor is not critical to the performance of the algorithm.
4. The performance of the adaptive filter is not sensitive to the choice of M - the filter order.
5. As the filter order increases, Mean square error (MSE) decreases.

7 MATLAB Source Code

7.1 Raw Data

```
clc
clear
close all
%% Data Acquisition

%Import Data
data = xlsread('Data');    %Fz|Cz|O2|HEOG

%Data Extraction
eog_HEOG = -data(:,4);      %HEOG artifact
eeg_Fz = data(:,1);         %Raw at Fz
eeg_Cz = data(:,2);         %Raw at Cz
eeg_O2 = data(:,3);         %Raw at O2

%Initial data plots
figure(1)

subplot(4,1,1)                %HEOG plot
plot(eog_HEOG)
title('Horizontal EOG Artifact')
xlabel('sample number')
ylabel('HEOG mu volts')

subplot(4,1,2)                %Raw at Fz plot
plot(eeg_Fz)
title('Raw EEG at Fz')
xlabel('sample number')
ylabel('EEG mu volts')

subplot(4,1,3)                %Raw at Cz plot
plot(eeg_Cz)
title('Raw EEG at Cz')
xlabel('sample number')
ylabel('EEG mu volts')

subplot(4,1,4)                %Raw at O2 plot
plot(eeg_O2)
title('Raw EEG at O2')
xlabel('sample number')
ylabel('EEG mu volts')
```

7.2 Claim 1

```
clc
clear
close all
%% Data Acquisition

%Import Data
data = xlsread('Data');    %Fz|Cz|O2|HEOG

%Data Extraction
eog_HEOG = -data(:,4);      %HEOG artifact
eeg_Fz = data(:,1);         %Raw at Fz

%% RLS ALGORITHM EXECUTION FOR Denoising at Fz is a easy, stable and
   fast convergence method suitable for online removal of EOG artifacts

% Execution for Fz electrode
% Variable Initialization
sample_no = size(eeg_Fz);     % No of samples/time points
order = 3;                    % Order of the Adaptive Filter (User
   Tunable)
sigma = 0.01;                 % Initializing variable
lambda = 0.9999;              % Forgetting Factor for RLS Algorithm (
   User Tunable)
H = zeros(order,1);           % Initial filter Coefficients
R = sigma*eye(order,order);   % Initial value for Reference combination


% RLS Algorithm for Adaptive Denoising
for n = 1:sample_no           % Loop to simulate reality situation

    s = eeg_Fz(n,1);          % eeg @ Fz at that time point
    if n>=order

        j = 1;                % calculation of last "order" reference
            vector
        for i = n:-1:(n+1-order)

            r(j,1) = eog_HEOG(i,1);
            j = j+1;
        end

        % Calculation of Filter Coefficients
        K = ((inv(R))*r)*(inv(lambda+r'*(inv(R))*r));
        e = s - (r'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r)')*(inv(R
            ))));
        % Calculation Of RLS algorithm estimated signal
        correct_Fz(n,1) = s - (r'*H);

    end
end

% Fz Denoising plot
```

```
figure(2)

subplot(3,1,1)               %HEOG plot
plot(eog_HEOG)
title('Horizontal EOG Artifact')
xlabel('sample number')
ylabel('HEOG mu volts')

subplot(3,1,2)               %Raw at Fz plot
plot(eeg_Fz)
title('Raw EEG at Fz')
xlabel('sample number')
ylabel('EEG mu volts')

subplot(3,1,3)               %denoised eeg at Fz plot
plot(correct_Fz)
title('Corrected EEG at Fz')
xlabel('sample number')
ylabel('EEG mu volts')
```

7.3 Claim 2

```
clc
clear
close all
%% Data Acquisition

%Import Data
data = xlsread('Data');    %Fz|Cz|Oz|HEOG

%Data Extraction
eog_HEOG = -data(:,4);      %HEOG artifact
eeg_Fz = data(:,1);        %Raw at Fz
eeg_Cz = data(:,2);        %Raw at Cz
eeg_O2 = data(:,3);        %Raw at O2

%% RLS ALGORITHM EXECUTION FOR Denoising at Fz

% Execution for Fz electrode
% Variable Initialization
sample_no = size(eeg_Fz);     % No of samples/time points
order = 3;                    % Order of the Adaptive Filter (User
   Tunable)
sigma = 0.01;                 % Initializing variable
lambda = 0.9999;              % Forgetting Factor for RLS Algorithm (
   User Tunable)
H = zeros(order,1);           % Initial filter Coefficients
R = sigma*eye(order,order);   % Initial value for Reference combination


% RLS Algorithm for Adaptive Denoising
for n = 1:sample_no           % Loop to simulate reality situation

    s = eeg_Fz(n,1);          % eeg @ Fz at that time point
    if n>=order

        j = 1;                % calculation of last "order" reference
           vector
        for i = n:-1:(n+1-order)

            r(j,1) = eog_HEOG(i,1);
            j = j+1;
        end

        % Calculation of Filter Coefficients
        K = ((inv(R))*r)*(inv(lambda+r'*(inv(R))*r));
        e = s - (r'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r)')*(inv(R
           ))));
        % Calculation Of RLS algorithm estimated signal
        correct_Fz(n,1) = s - (r'*H);

    end
end
```

```
% Fz Denoising plot
figure(2)

subplot(3,1,1)              %HEOG plot
plot(eog_HEOG)
title('Horizontal EOG Artifact')
xlabel('sample number')
ylabel('HEOG mu volts')

subplot(3,1,2)              %Raw at Fz plot
plot(eeg_Fz)
title('Raw EEG at Fz')
xlabel('sample number')
ylabel('EEG mu volts')

subplot(3,1,3)              %denoised eeg at Fz plot
plot(correct_Fz)
title('Corrected EEG at Fz')
xlabel('sample number')
ylabel('EEG mu volts')

%% RLS ALGORITHM EXECUTION FOR Denoising at Cz


% Execution for Cz electrode
% Variable Initialization
sample_no = size(eeg_Cz);    % No of samples/time points
order = 3;                   % Order of the Adaptive Filter (User
   Tunable)
sigma = 0.01;                % Initializing variable
lambda = 0.9999;             % Forgetting Factor for RLS Algorithm (
   User Tunable)
H = zeros(order,1);          % Initial filter Coefficients
R = sigma*eye(order,order);  % Initial value for Reference combination


% RLS Algorithm for Adaptive Denoising
for n = 1:sample_no          % Loop to simulate reality situation

    s = eeg_Cz(n,1);         % eeg @ Cz at that time point
    if n>=order

        j = 1;               % calculation of last "order" reference
           vector
        for i = n:-1:(n+1-order)

            r(j,1) = eog_HEOG(i,1);
            j = j+1;
        end

        % Calculation of Filter Coefficients
        K = ((inv(R))*r)*(inv(lambda+r'*(inv(R))*r));
        e = s - (r'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r)')*(inv(R
           ))));
        % Calculation Of RLS algorithm estimated signal
        correct_Cz(n,1) = s - (r'*H);

    end
end

% Cz Denoising plot
figure(3)

subplot(3,1,1)              %HEOG plot
plot(eog_HEOG)
title('Horizontal EOG Artifact')
xlabel('sample number')
ylabel('HEOG mu volts')

subplot(3,1,2)              %Raw at Cz plot
plot(eeg_Cz)
title('Raw EEG at Cz')
xlabel('sample number')
ylabel('EEG mu volts')

subplot(3,1,3)              %denoised eeg at Cz plot
plot(correct_Cz)
title('Corrected EEG at Cz')
xlabel('sample number')
ylabel('EEG mu volts')


%% RLS ALGORITHM EXECUTION FOR Denoising at O2

% Execution for O2 electrode
% Variable Initialization
sample_no = size(eeg_O2);    % No of samples/time points
order = 3;                   % Order of the Adaptive Filter (User
   Tunable)
sigma = 0.01;                % Initializing variable
lambda = 0.9999;             % Forgetting Factor for RLS Algorithm (
   User Tunable)
H = zeros(order,1);          % Initial filter Coefficients
R = sigma*eye(order,order);  % Initial value for Reference combination


% RLS Algorithm for Adaptive Denoising
for n = 1:sample_no          % Loop to simulate reality situation

    s = eeg_O2(n,1);         % eeg @ O2 at that time point
    if n>=order

        j = 1;               % calculation of last "order" reference
           vector
        for i = n:-1:(n+1-order)

            r(j,1) = eog_HEOG(i,1);
            j = j+1;
        end
        % Calculation of Filter Coefficients
        K = ((inv(R))*r)*(inv(lambda+r'*(inv(R))*r));
        e = s - (r'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r)')*(inv(R
           ))));
        % Calculation Of RLS algorithm estimated signal
        correct_O2(n,1) = s - (r'*H);

    end
end

% Oz Denoising plot
figure(4)

subplot(3,1,1)              %HEOG plot
plot(eog_HEOG)
title('Horizontal Noise Artifact')
xlabel('sample number')
ylabel('HEOG mu volts')

subplot(3,1,2)              %Raw at O2 plot
plot(eeg_O2)
title('Raw EEG at O2')
xlabel('sample number')
ylabel('EEG mu volts')

subplot(3,1,3)              %denoised eeg at O2 plot
plot(correct_O2)
title('Corrected EEG at O2')
xlabel('sample number')
ylabel('EEG mu volts')
```

7.4 Claim 3

```
clc
clear
close all
%% Data Acquisition

%Import Data
data = xlsread('Data');   %Fz|Cz|Oz|HEOG

%Data Extraction
eog_HEOG = -data(:,4);     %HEOG artifact
eeg_Fz = data(:,1);        %Raw at Fz

%% Claim 3: Exact value of lambda is not critical to the performance of
    the algorithm

% Execution for Fz electrode with Lambda = 0.9999
% Variable Initialization
sample_no = size(eeg_Fz);     % No of samples/time points
order = 3;                    % Order of the Adaptive Filter (User
   Tunable)
sigma = 0.01;                 % Initializing variable
lambda = 0.9999;              % Forgetting Factor for RLS Algorithm (
   User Tunable)
H = zeros(order,1);           % Initial filter Coefficients
R = sigma*eye(order,order);   % Initial value for Reference combination


% RLS Algorithm for Adaptive Denoising
for n = 1:sample_no           % Loop to simulate reality situation

    s = eeg_Fz(n,1);          % eeg @ Fz at that time point
    if n>=order

        j = 1;                % calculation of last "order" reference
            vector
        for i = n:-1:(n+1-order)

            r(j,1) = eog_HEOG(i,1);
            j = j+1;
        end

        % Calculation of Filter Coefficients
        K = ((inv(R))*r)*(inv(lambda+r'*(inv(R))*r));
        e = s - (r'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r)')*(inv(R
            ))));
        % Calculation Of RLS algorithm estimated signal
        correct_Fz_lambda1(n,1) = s - (r'*H);

    end
end
```

```
% Execution for Fz electrode with Lambda = 0.995
% Variable Initialization
sample_no = size(eeg_Fz);     % No of samples/time points
order = 3;                    % Order of the Adaptive Filter (User
   Tunable)
sigma = 0.01;                 % Initializing variable
lambda = 0.995;              % Forgetting Factor for RLS Algorithm (User
    Tunable)
H = zeros(order,1);           % Initial filter Coefficients
R = sigma*eye(order,order);   % Initial value for Reference combination


% RLS Algorithm for Adaptive Denoising
for n = 1:sample_no           % Loop to simulate reality situation

    s = eeg_Fz(n,1);          % eeg @ Fz at that time point
    if n>=order

        j = 1;                % calculation of last "order" reference
           vector
        for i = n:-1:(n+1-order)

            r(j,1) = eog_HEOG(i,1);
            j = j+1;
        end

        % Calculation of Filter Coefficients
        K = ((inv(R))*r)*(inv(lambda+r'*(inv(R))*r));
        e = s - (r'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r)')*(inv(R
           ))));
        % Calculation Of RLS algorithm estimated signal
        correct_Fz_lambda2(n,1) = s - (r'*H);

    end
end


% Execution for Fz electrode with Lambda = 1
% Variable Initialization
sample_no = size(eeg_Fz);     % No of samples/time points
order = 3;                    % Order of the Adaptive Filter (User
   Tunable)
sigma = 0.01;                 % Initializing variable
lambda = 1;               % Forgetting Factor for RLS Algorithm (User
   Tunable)
H = zeros(order,1);           % Initial filter Coefficients
R = sigma*eye(order,order);   % Initial value for Reference combination


% RLS Algorithm for Adaptive Denoising
for n = 1:sample_no           % Loop to simulate reality situation

    s = eeg_Fz(n,1);          % eeg @ Fz at that time point
    if n>=order

        j = 1;              % calculation of last "order" reference
           vector
        for i = n:-1:(n+1-order)

            r(j,1) = eog_HEOG(i,1);
            j = j+1;
        end

        % Calculation of Filter Coefficients
        K = ((inv(R))*r)*(inv(lambda+r'*(inv(R))*r));
        e = s - (r'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r)')*(inv(R
           ))));
        % Calculation Of RLS algorithm estimated signal
        correct_Fz_lambda3(n,1) = s - (r'*H);

    end
end

% Fz Denoising plot with Different Lambda values
figure(5)

subplot(3,1,1)              %denoised eeg at Fz plot with Lambda =
   0.9999
plot(correct_Fz_lambda1)
title('Lambda = 0.9999')
xlabel('sample number')
ylabel('EEG mu volts')

subplot(3,1,2)              %denoised eeg at Fz plot with Lambda =
   0.995
plot(correct_Fz_lambda2)
title('Lambda = 0.995')
xlabel('sample number')
ylabel('EEG mu volts')

subplot(3,1,3)              %denoised eeg at Fz plot with Lambda = 1
plot(correct_Fz_lambda3)
title('Lambda = 1')
xlabel('sample number')
ylabel('EEG mu volts')
```

7.5 Claim 4

```
clc
clear
close all
%% Data Acquisition

%Import Data
data = xlsread('Data');    %Fz|Cz|Oz|HEOG

%Data Extraction
eog_HEOG = -data(:,4);      %HEOG artifact
eeg_Fz = data(:,1);        %Raw at Fz
eeg_Cz = data(:,2);        %Raw at Cz
eeg_Oz = data(:,3);        %Raw at Oz

%% Claim 4: The performance of the adaptive filter is not very
   sensitive to choice of M

% Execution for Fz electrode with M = 3
% Variable Initialization
sample_no = size(eeg_Fz);     % No of samples/time points
order = 3;                    % Order of the Adaptive Filter (User
   Tunable)
sigma = 0.01;                 % Initializing variable
lambda = 0.9999;              % Forgetting Factor for RLS Algorithm (
   User Tunable)
H = zeros(order,1);           % Initial filter Coefficients
R = sigma*eye(order,order);   % Initial value for Reference combination


% RLS Algorithm for Adaptive Denoising
for n = 1:sample_no           % Loop to simulate reality situation

    s = eeg_Fz(n,1);          % eeg @ Fz at that time point
    if n>=order

        j = 1;                % calculation of last "order" reference
           vector
        for i = n:-1:(n+1-order)

            r(j,1) = eog_HEOG(i,1);
            j = j+1;
        end

        % Calculation of Filter Coefficients
        K = ((inv(R))*r)*(inv(lambda+r'*(inv(R))*r));
        e = s - (r'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r)')*(inv(R
           ))));
        % Calculation Of RLS algorithm estimated signal
        correct_Fz_M1(n,1) = s - (r'*H);

    end
end
```

```
% Execution for Fz electrode with M = 1
% Variable Initialization
sample_no = size(eeg_Fz);     % No of samples/time points
order = 1;                    % Order of the Adaptive Filter (User
   Tunable)
sigma = 0.01;                 % Initializing variable
lambda = 0.9999;              % Forgetting Factor for RLS Algorithm (
   User Tunable)
H = zeros(order,1);           % Initial filter Coefficients
R = sigma*eye(order,order);   % Initial value for Reference combination


% RLS Algorithm for Adaptive Denoising
for n = 1:sample_no           % Loop to simulate reality situation

    s = eeg_Fz(n,1);          % eeg @ Fz at that time point
    if n>=order

        j = 1;                % calculation of last "order" reference
           vector
        for i = n:-1:(n+1-order)

            r1(j,1) = eog_HEOG(i,1);
            j = j+1;
        end

        % Calculation of Filter Coefficients
        K = ((inv(R))*r1)*(inv(lambda+r1'*(inv(R))*r1));
        e = s - (r1'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r1)')*(inv(
           R))));
        % Calculation Of RLS algorithm estimated signal
        correct_Fz_M2(n,1) = s - (r1'*H);

    end
end



% Execution for Fz electrode with M = 12
% Variable Initialization
sample_no = size(eeg_Fz);     % No of samples/time points
order = 12;                    % Order of the Adaptive Filter (User
   Tunable)
sigma = 0.01;                 % Initializing variable
lambda = 0.9999;              % Forgetting Factor for RLS Algorithm (
   User Tunable)
H = zeros(order,1);           % Initial filter Coefficients
R = sigma*eye(order,order);   % Initial value for Reference combination


% RLS Algorithm for Adaptive Denoising
```

```
for n = 1:sample_no           % Loop to simulate reality situation

    s = eeg_Fz(n,1);          % eeg @ Fz at that time point
    if n>=order

        j = 1;                % calculation of last "order" reference
            vector
        for i = n:-1:(n+1-order)

            r(j,1) = eog_HEOG(i,1);
            j = j+1;
        end

        % Calculation of Filter Coefficients
        K = ((inv(R))*r)*(inv(lambda+r'*(inv(R))*r));
        e = s - (r'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r)')*(inv(R
            ))));
        % Calculation Of RLS algorithm estimated signal
        correct_Fz_M3(n,1) = s - (r'*H);

    end
end

% Fz Denoising plot with Different M values
figure(6)

subplot(3,1,1)               %denoised eeg at Fz plot with M = 3
plot(correct_Fz_M1)
title('M = 3')
xlabel('sample number')
ylabel('EEG mu volts')

subplot(3,1,2)               %denoised eeg at Fz plot with M = 1
plot(correct_Fz_M2)
title('M = 1')
xlabel('sample number')
ylabel('EEG mu volts')

subplot(3,1,3)               %denoised eeg at Fz plot with M = 12
plot(correct_Fz_M3)
title('M = 12')
xlabel('sample number')
ylabel('EEG mu volts')
```

7.6 Claim 5

```
clc
clear
close all
%% Data Acquisition

%Import Data
data = xlsread('Data');   %Fz|Cz|Oz|HEOG

%Data Extraction
eog_HEOG = -data(:,4);      %HEOG artifact
eeg_Fz = data(:,1);        %Raw at Fz
eeg_Cz = data(:,2);        %Raw at Cz
eeg_Oz = data(:,3);        %Raw at Oz

%% Claim 5: As order increases the MSE monotonically decreases

% Execution for Fz electrode with M = 3
% Variable Initialization
sample_no = size(eeg_Fz);    % No of samples/time points
order = 3;                   % Order of the Adaptive Filter (User
    Tunable)
sigma = 0.01;                % Initializing variable
lambda = 0.9999;             % Forgetting Factor for RLS Algorithm (
    User Tunable)
H = zeros(order,1);          % Initial filter Coefficients
R = sigma*eye(order,order);  % Initial value for Reference combination

MSE_M1 = 0;
MSE_M2 = 0;
MSE_M3 = 0;


% RLS Algorithm for Adaptive Denoising
for n = 1:sample_no          % Loop to simulate reality situation

    s = eeg_Fz(n,1);         % eeg @ Fz at that time point
    if n>=order

        j = 1;               % calculation of last "order" reference
            vector
        for i = n:-1:(n+1-order)

            r(j,1) = eog_HEOG(i,1);
            j = j+1;
        end

        % Calculation of Filter Coefficients
        K = ((inv(R))*r)*(inv(lambda+r'*(inv(R))*r));
        e = s - (r'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r)')*(inv(R
            ))));
        % Calculation Of RLS algorithm estimated signal
        correct_Fz_M1(n,1) = s - (r'*H);
        MSE_M1 = ((lambda^(sample_no(1,1)-n))*((correct_Fz_M1(n,1))^2))
            + MSE_M1;
    end
end

MSE_M1 = MSE_M1/(sample_no(1,1) - order + 1)

% Execution for Fz electrode with M = 1
% Variable Initialization
sample_no = size(eeg_Fz);     % No of samples/time points
order = 1;                    % Order of the Adaptive Filter (User
    Tunable)
sigma = 0.01;                 % Initializing variable
lambda = 0.9999;              % Forgetting Factor for RLS Algorithm (
    User Tunable)
H = zeros(order,1);           % Initial filter Coefficients
R = sigma*eye(order,order);   % Initial value for Reference combination


% RLS Algorithm for Adaptive Denoising
for n = 1:sample_no           % Loop to simulate reality situation

    s = eeg_Fz(n,1);          % eeg @ Fz at that time point
    if n>=order

        j = 1;                % calculation of last "order" reference
            vector
        for i = n:-1:(n+1-order)

            r1(j,1) = eog_HEOG(i,1);
            j = j+1;
        end

        % Calculation of Filter Coefficients
        K = ((inv(R))*r1)*(inv(lambda+r1'*(inv(R))*r1));
        e = s - (r1'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r1)')*(inv(
            R))));
        % Calculation Of RLS algorithm estimated signal
        correct_Fz_M2(n,1) = s - (r1'*H);

        MSE_M2 = ((lambda^(sample_no(1,1)-n))*((correct_Fz_M2(n,1))^2))
            + MSE_M2;

    end
end

MSE_M2 = MSE_M2/(sample_no(1,1) - order + 1)

% Execution for Fz electrode with M = 12
% Variable Initialization
sample_no = size(eeg_Fz);     % No of samples/time points
order = 12;                    % Order of the Adaptive Filter (User
    Tunable)
```

```
sigma = 0.01;                  % Initializing variable
lambda = 0.9999;               % Forgetting Factor for RLS Algorithm (
   User Tunable)
H = zeros(order,1);            % Initial filter Coefficients
R = sigma*eye(order,order);   % Initial value for Reference combination


% RLS Algorithm for Adaptive Denoising
for n = 1:sample_no            % Loop to simulate reality situation

    s = eeg_Fz(n,1);           % eeg @ Fz at that time point
    if n>=order

        j = 1;                 % calculation of last "order" reference
           vector
        for i = n:-1:(n+1-order)

            r(j,1) = eog_HEOG(i,1);
            j = j+1;
        end

        % Calculation of Filter Coefficients
        K = ((inv(R))*r)*(inv(lambda+r'*(inv(R))*r));
        e = s - (r'*H);
        H = H + (K*e);
        R = inv((inv(lambda)*inv(R)) - ((inv(lambda))*(K)*((r)')*(inv(R
           ))));
        % Calculation Of RLS algorithm estimated signal
        correct_Fz_M3(n,1) = s - (r'*H);

        MSE_M3 = ((lambda^(sample_no(1,1)-n))*((correct_Fz_M3(n,1))^2))
           + MSE_M3;

    end
end

MSE_M3= MSE_M3/(sample_no(1,1) - order + 1)
```

References

[1] HE,P., WILSON,G.,RUSSELL,C. (2004): *Removal of Ocular Artifacts from Electro-encephalogram by adaptive filtering* Medical and Biological Engineering and Computing,Vol.42

[2] COMSTOCK, J. R., and ARNEGARD, R. J. (1992): *The multi-attribute task battery for human operator and strategic behavior research* NASA Technical Memorandum 104174

[3] CROFT, R. J., and BARRY, R. J. (2000): *Removal of oculax artifact from the EEG: a review* Neurophysiologie Clinique, 30, pp. 5-19

[4] GRATTON, G., COLES, M. G. H., and DONCHIN, E. (1983): *A new method for off-line removal of ocular artifacts* Electroenceph.Clin. Neurophysiol., 55, pp. 468-484.

[5] HANKINS, T. C., and WILSON, G. E (1998): *A comparison of heart rate, eye activity, EEG and subjective measures of pilot mental workload during flight* Aviat. Space Environ. Med., 69, pp. 360-367

[6] HAYKIN, S. (1996): *Adaptive filter theory, 3rd edn* (Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1996)

[7] JASPER, H. H. (1958): *The ten-twenty electrode system of the International Federation* Electroenceph. Clin. Neurophysiol., 10,pp. 371-375

[8] JUNG, T. R, MAKEIG, S., WESTERFIELD, M., TOWNSEND, J., COURCHESNE, E., and SEJNOWSrd, T. J. (2000): *Removal of eye activity artifacts from visual event-related potential in normal and clinical subjects* Clin. Neurophysiol., 111, pp. 1745-1758

[9] KENEMANS, J. L., MOLENAAR, E C. M., VERBATEN, M. N., and SLANGEN, J. L. (1991): *Removal of the ocular artifact from the EEG: A comparison of time and frequency domain methods with simulated and real data* Psychophysiology, 28, pp. 115-121

[10] VASEGHI, S. V (1996): *Advanced signal processing and digital noise reduction* John Wiley and Sons and B. G. Teubner, New York, USA, 1996, 172-177

[11] VERLEGER, R., GASSER, T., and MOCKS, J. (1982): *Correction of EOG artifacts in event-related potentials of the EEG: Aspects of reliability and validity* Psychophysiology, 19, pp. 472-480

[12] WHITTON, J. L., Lug, E, and MOLDOFSKY, H. (1978): *A spectral method for removing eye movement artifacts from the EEG* Electroenceph. Clin. Neurophysiol., 44, pp. 735-741

[13] WIDROW, B., GLOVER, J. R., McCOOL, J. M., KAUNITZ, J., WILLIAMS, C. S., HEARN, R. H., ZEIDLER, J. R., DONG, E., and GOODLIN, R. C. (1975): *Adaptive noise canceling: Principles and applications* Proc. IEEE, 63, pp. 1692-1716

[14] WILSON, G. E, and RUSSELL, C. (1999): *Operator functional state classification using neural networks with combined physiological and performance features* Proc. Human Factors" and Ergonomics Society, 43rd, Annual Meeting. pp. 1099-1101

[15] WILSON, G. E (2002): *An analysis of mental workload in pilots during flight using multiple psychophysiological measures* Int. J. Aviat. Psychol., 12, pp. 3-18

[16] WILSON, G. F., and RUSSELL, C. (2003): *Real-time assessment of mental workload using psychophysiological measures* Human Factors', 45, pp. 635-643

[17] WOESTENBURG, J. C., VERBATEN, M. N., and SLANGER, J. L. (1983): *The removal of the eye-movement artifact from the EEG by regression analysis in the frequency domain* Biological Psychology, 16, pp. 127-147