1 Summary of our approach

We designed a filter based on the LMMSE theory. The data consists of noisy speech signal $x[n] = s[n] + v_1[n]$ where $v_1[n]$ is additive noise component added to the clean speech $s[n]$. In addition, we are given another noise signal $v_2[n]$ which is highly correlated with $v_1[n]$ (noise component in $x[n]$). Both signals are of length 74000 sampled at 32 kHz. Figure 1 shows the relation between all signal components and how we estimate the clean speech $\hat{s}[n]$ from $x[n]$.

The desired clean signal $s[n]$ can be obtained by $s[n] = x[n] - v_1[n]$. Since we do not know $v_1[n]$, we estimate $v_1[n]$ using $v_2[n]$ and the fact that the two noise components are highly correlated.

Let us denote $X_M[n] = [v_2[n], v_2[n-1], \ldots, v_2[n-M+1]]^T \in \mathbb{R}^{M\times 1}$ and $Y_L[n] = [v_1[n], v_1[n-1], \ldots, v_1[n-L+1]]^T \in \mathbb{R}^{L\times 1}$. The objective is to estimate $Y[n]$ as $\hat{Y}[n]$ to minimize the mean squared error $E(\|Y - \hat{Y}\|^2)$. From the LMMSE theory, we know that the LMMSE estimate of $Y[n]$ is -

$$\hat{Y}_o = Y_{\text{LMMSE}} = C_o^H X_M[n] = R_{YX} R_{XX}^{-1} X_M[n] \tag{1}$$

where $R_{XX} = E(X_M X_M^H)$ is $M \times M$ autocorrelation matrix and $R_{YX} = E(Y X_M^H) = R_{XY}^H$ is $L \times M$ cross correlation matrix. $C_o \in \mathbb{R}^{M\times L}$ is the optimum filter coefficient. For our purpose, we chose $L = 1$.

First we estimate the autocorrelation $r_{xx}[m]$ of given noisy audio signal $x[n]$, the autocorrelation $r_{v2}[m]$ of noise signal $v_2[n]$, and cross correlation sequence $r_{xv2}[m]$ between $x[n]$ and $v_2[n]$ each of length $2N-1$. Then, we compute correlation matrix R_{XX} and R_{YX} for filter order M -

$$R_{XX} = E(X_M[n] X_M[n]^H) = E\Big(\begin{bmatrix} v_2[n] \\ v_2[n-1] \\ \vdots \\ v_2[n-M+1] \end{bmatrix} \begin{bmatrix} v_2[n] & v_2[n-1] & \ldots & v_2[n-M+1] \end{bmatrix} \Big)_{M\times M} \tag{2}$$

$$\begin{aligned} R_{YX} &= E\Big(v_1[n] \begin{bmatrix} v_2[n] & v_2[n-1] & \ldots & v_2[n-M+1] \end{bmatrix} \Big)_{1\times M} \\ &= E\Big((x[n] - s[n]) \begin{bmatrix} v_2[n] & v_2[n-1] & \ldots & v_2[n-M+1] \end{bmatrix} \Big) \\ &= E\Big(x[n] \begin{bmatrix} v_2[n] & v_2[n-1] & \ldots & v_2[n-M+1] \end{bmatrix} \Big)_{1\times M} \end{aligned}$$

where we assume that the clean audio signal $s[n]$ is independent and uncorrelated with the noise $v_2[n]$. Then we compute the optimum filter coefficients C_o using equation 1. $C_o^{(M)} = [c_1, c_2, \ldots, c_M]^T$. Then, the estimated noise component $\hat{v_1}[n]$ is given by -

$$\hat{v_1}[n] = C_o^{(M)} X_M[n] = \begin{bmatrix} c_1 & c_2 & \ldots & c_M \end{bmatrix}_{1\times M} \begin{bmatrix} v_2[n] \\ v_2[n-1] \\ \vdots \\ v_2[n-M+1] \end{bmatrix}_{M\times 1}$$

We estimate the clean audio speech by subtracting the estimated noise $\hat{v_1}[n]$ from the original noisy signal $x[n]$

$$\hat{s}[n] = x[n] - v_1[n] \approx x[n] - \hat{v_1}[n] \tag{3}$$

Finally, we compute signal-to-noise ratio (SNR) in decibels to quantitatively evaluate the filter performance.

$$\text{SNR} = 10 \log_{10} \Big[\frac{\sum_{n=1}^{N} \hat{s}^2[n]}{\sum_{n=1}^{N} \hat{v_1}^2[n]} \Big] \tag{4}$$

where N is the signal and noise length.

1.1 Filter coefficients and SNR

1.1.1 M=4

$$C_o = \begin{bmatrix} 1.7462 & -1.2218 & -1.7792 & 1.6322 \end{bmatrix} \tag{5}$$

SNR computed using estimated audio $\hat{s}[n]$ and estimated noise $\hat{v_1}[n] = -1.5863$
Variance of estimated audio $\hat{s}[n] = 0.021782$

1.1.2 M=8

$$C_o = \begin{bmatrix} 2.8120 & -3.6999 & 1.0292 & -0.1468 & -1.1140 & 2.4156 & -0.2312 & -0.5946 \end{bmatrix} \tag{6}$$

SNR computed using estimated audio $\hat{s}[n]$ and estimated noise $\hat{v_1}[n] = -14.0813$
Variance of estimated audio $\hat{s}[n] = 0.0019975$

1.1.3 M=12

$$C_o = \begin{bmatrix} 2.8097 & -3.7000 & 1.0304 & -0.1462 & -1.2646 & 2.8599 & -0.6655 & -0.3174 & 0.1017 & -0.2659 \\ -0.3362 & 0.4734 \end{bmatrix} \tag{7}$$

SNR computed using estimated audio $\hat{s}[n]$ and estimated noise $\hat{v_1}[n] = -27.1649$
Variance of estimated audio $\hat{s}[n] = 0.00010183$

1.1.4 M=16

$$C_o = \begin{bmatrix} 2.8096 & -3.6995 & 1.0290 & -0.1446 & -1.2654 & 2.8602 & -0.6650 & -0.3184 & 0.1931 & -0.5017 \\ 0.0266 & 0.0922 & 0.1859 & 0.0006 & -0.0012 & 0.0008 \end{bmatrix} \tag{8}$$

SNR computed using estimated audio $\hat{s}[n]$ and estimated noise $\hat{v_1}[n] = -36.2599$
Variance of estimated audio $\hat{s}[n] = 1.2564 \times 10^{-5}$

1.1.5 M=20

$$C_o = \begin{bmatrix} 2.8096 & -3.6995 & 1.0290 & -0.1446 & -1.2654 & 2.8602 & -0.6650 & -0.3184 & 0.1931 & -0.5017 \\ 0.0266 & 0.0922 & 0.1859 & 0.0010 & -0.0023 & 0.0025 & -0.0018 & 0.0009 & 0.0001 & -0.0002 \end{bmatrix} \tag{9}$$

SNR computed using estimated audio $\hat{s}[n]$ and estimated noise $\hat{v_1}[n] = -36.2592$
Variance of estimated audio $\hat{s}[n] = 1.2566 \times 10^{-5}$

The SNR measures the strength of the signal amplitude to the strength of the noise amplitude. In our case, we can clearly see that our estimated audio signal quality improves as we can increase M from 4 to 20. But we do not see that in the SNR since the noise estimate remains almost similar in magnitude but the signal magnitude improves and noise components are removed from signal as M increases thus reducing its amplitude at several places. Hence the SNR decreases but the perceived sound quality improves. But on the other hand, the variance of the estimated audio signal decreases as M increases from 4 upto 20 showing that the variance in the audio data is decreasing which can also be observed in the estimated clean audio figures below.

1.2 Comment on the perceptual quality

Speech A: "Hey buddy, you wanna earn a couple of bucks"

For filter order M=4, speech A is not perceivable and we only hear noise. For M=8, we perceive (if listened carefully) that someone is speaking but it is mostly noise and we cannot make out what is being spoken. For M=12, we perceive both speech A and noise (approximately in equal proportions) and speech A is clearly audible and perceivable. For M=16 and 20, speech A is clearly audible with negligible background noise (if any). Based on these results for different filter orders, my guess is that the noise was generated using an AR model process with a model order $M \geq 9$ and ≤ 12.

2 Results

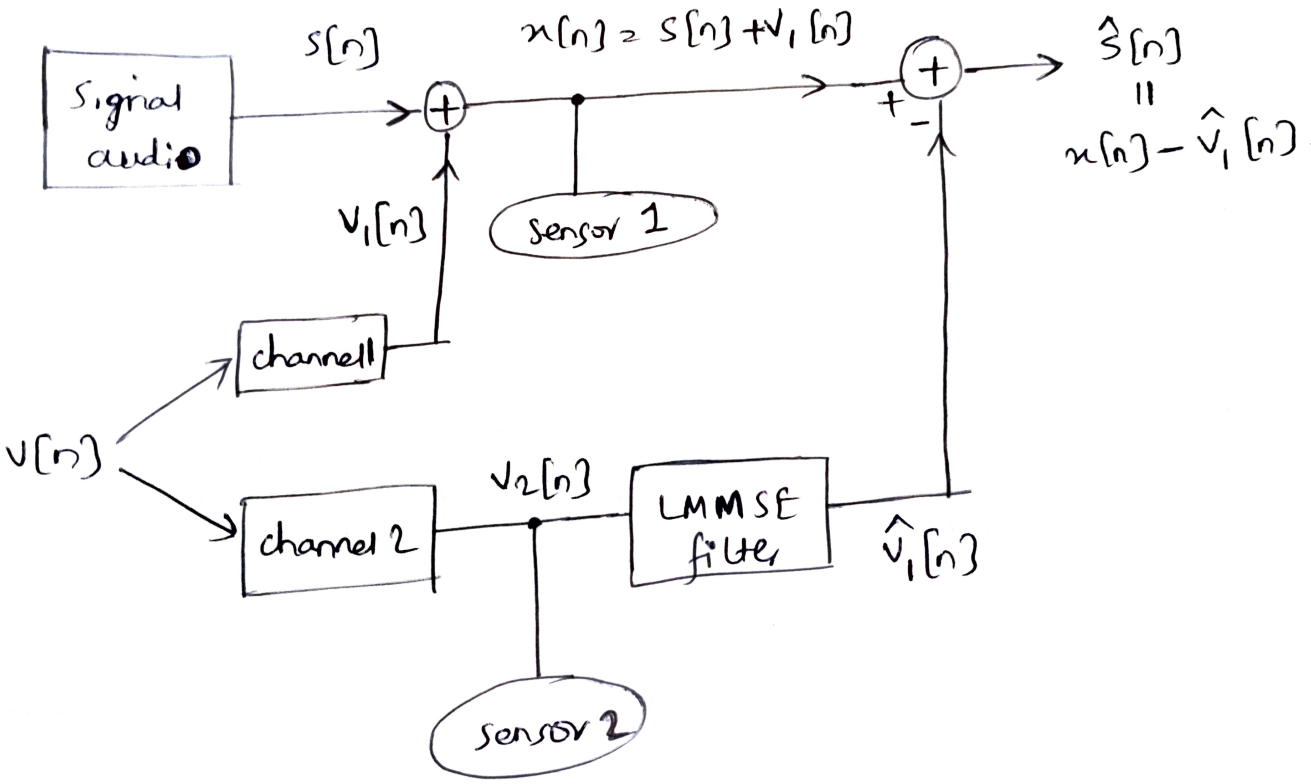


Figure 1: Flow diagram showing various signals available $x[n]$ and $v_1[n]$ and how we estimate clean audio $\hat{s}[n]$

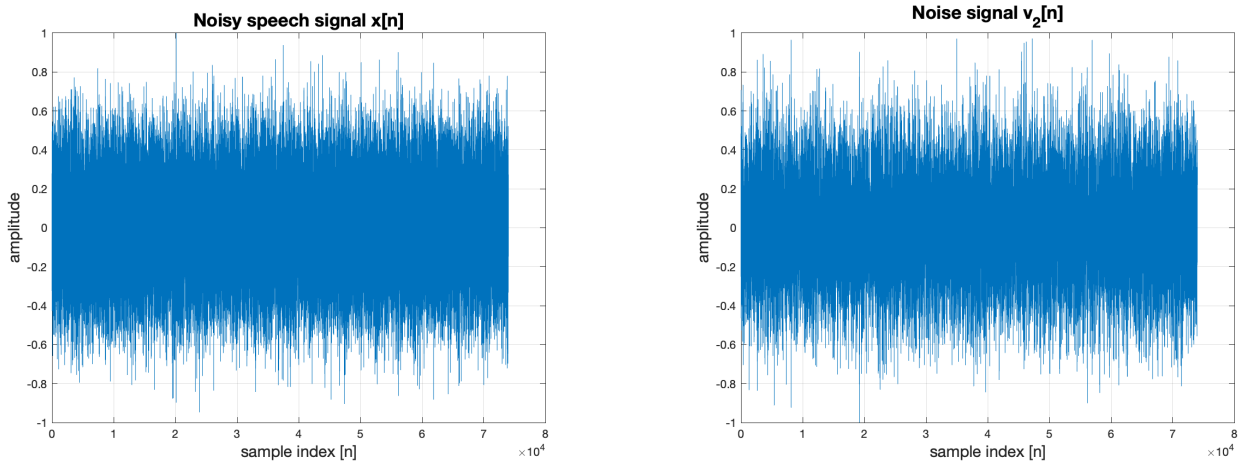


Figure 2: (Left) Original noisy audio signal $x[n]$. (Right) Noise signal $v_2[n]$ highly correlated with noise signal $v_1[n]$ present in $x[n]$

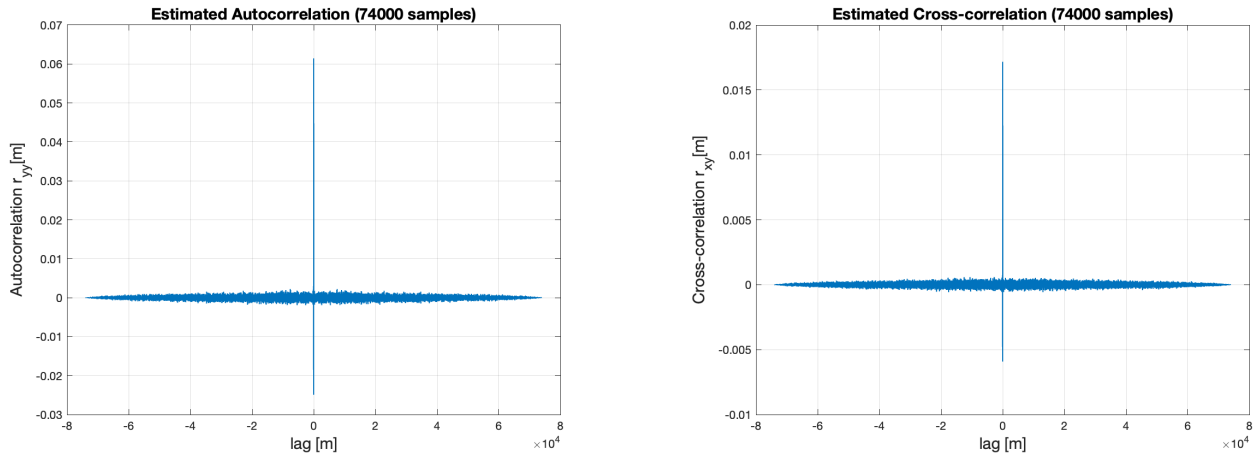


Figure 3: (Left) Autocorrelation sequence $r_{v2}[m]$ of the noise signal $v_2[n]$. (Right) Cross-correlation sequence $r_{xv2}[m]$ between the noisy audio signal $x[n]$ and noise signal $v_2[n]$

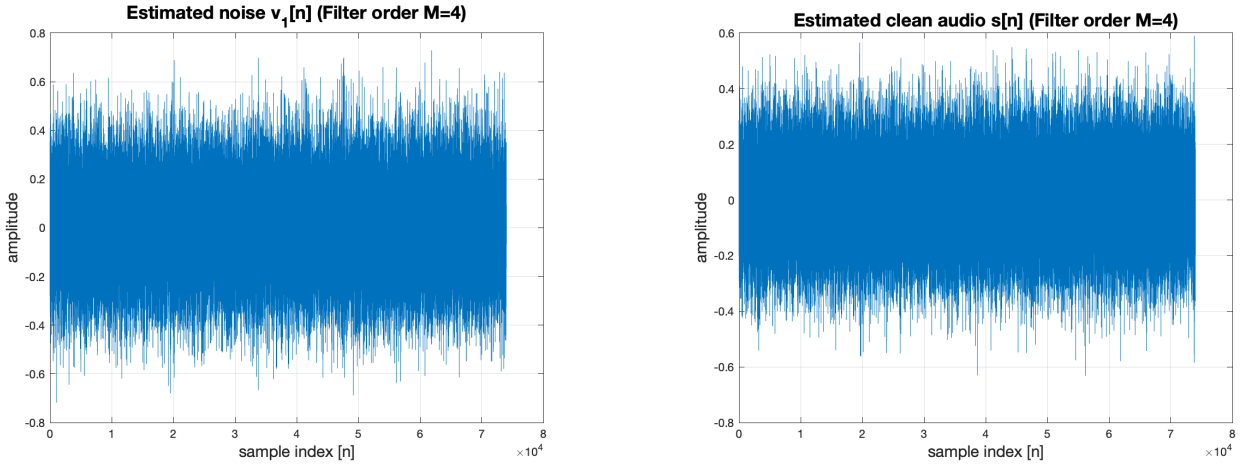


Figure 4: (Left) Estimated noise signal component $v_1[n]$ and (Right) estimated clean audio signal $s[n]$ for filter order M = 4

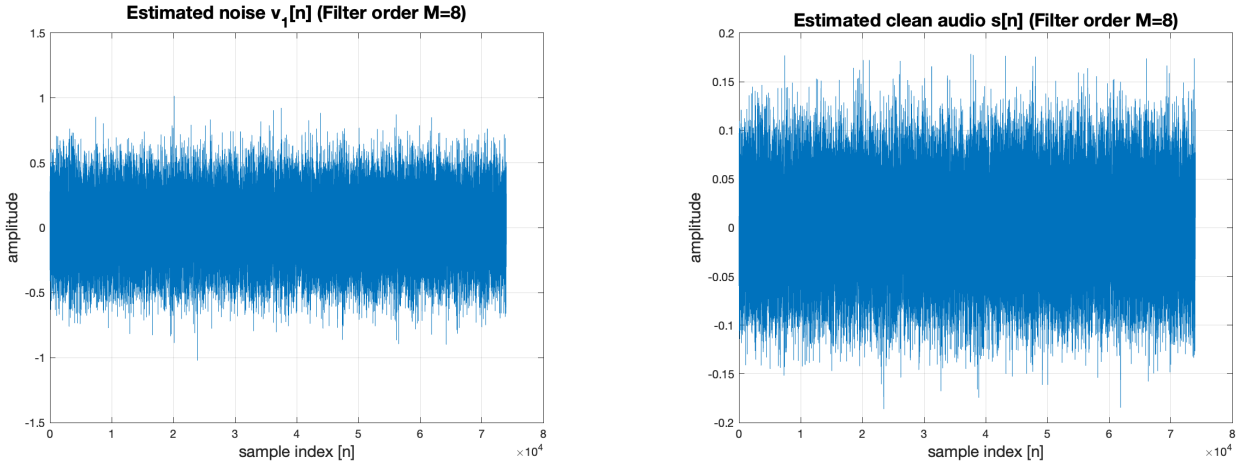
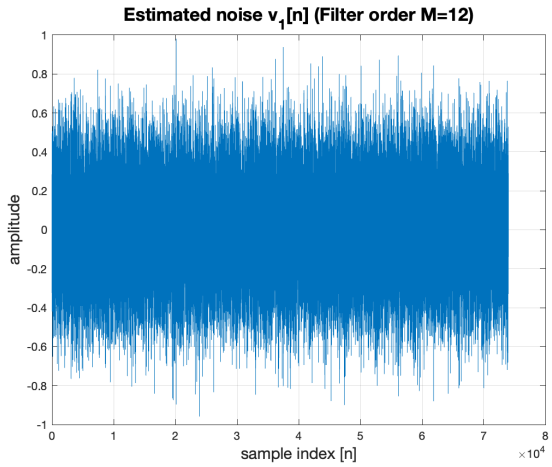


Figure 5: (Left) Estimated noise signal component $v_1[n]$ and (Right) estimated clean audio signal $s[n]$ for filter order M = 8



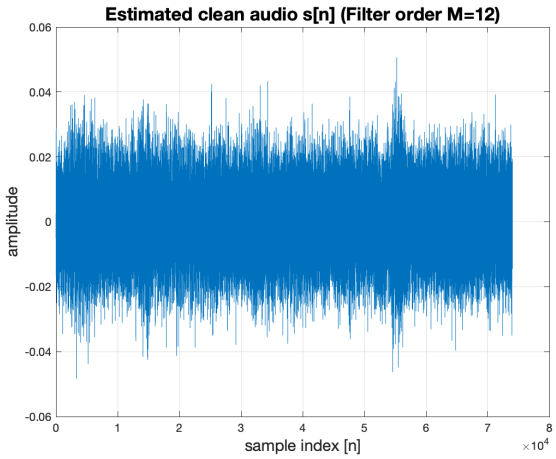
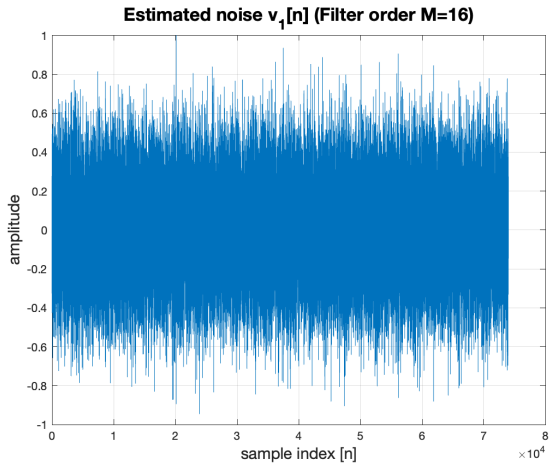


Figure 6: (Left) Estimated noise signal component $v_1[n]$ and (Right) estimated clean audio signal $s[n]$ for filter order M = 12



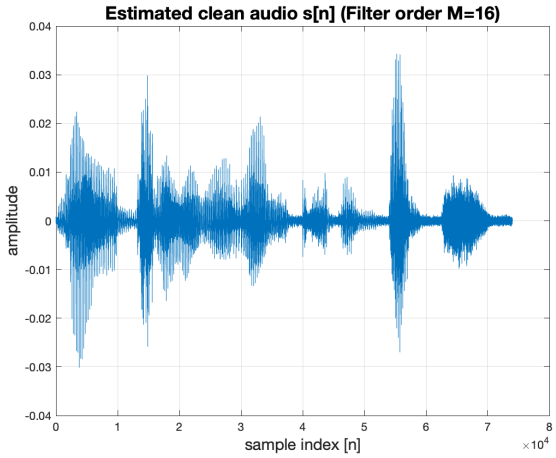
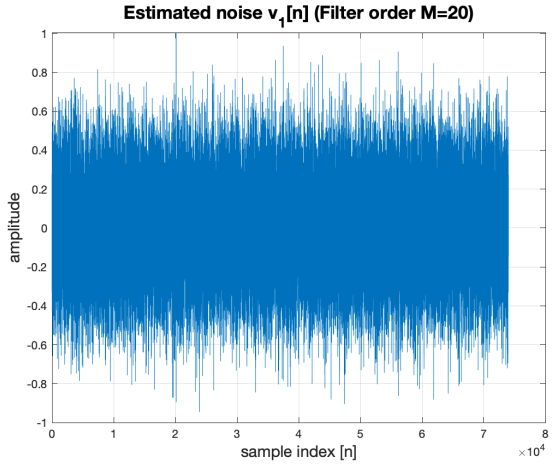


Figure 7: (Left) Estimated noise signal component $v_1[n]$ and (Right) estimated clean audio signal $s[n]$ for filter order M = 16



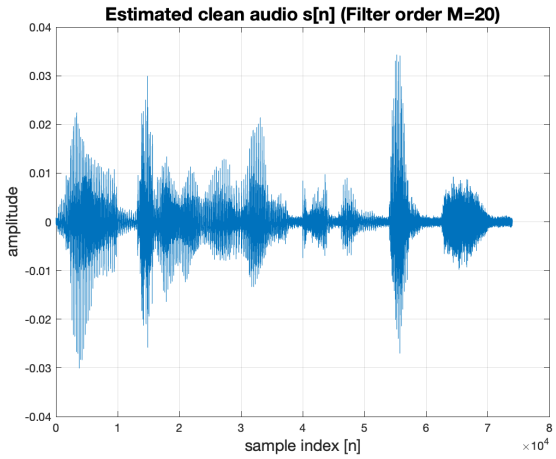


Figure 8: (Left) Estimated noise signal component $v_1[n]$ and (Right) estimated clean audio signal $s[n]$ for filter order M = 20

3 MATLAB Code

3.1 speech_denoising.m

```
clc; clear;

[xn, Fs] = audioread("../data/x.wav");
[v2n, ~] = audioread("../data/v2.wav");  % highly correlated with v1[n]; no speech
Nx = size(xn,1);
Nv2 = size(v2n,1);

% sound(xn,Fs);
% sound(v2n,Fs);

fig = figure;
plot(0:Nx-1, xn);
xlabel("sample index [n]", FontSize=14);
ylabel("amplitude", FontSize=14);
title("Noisy speech signal x[n]", FontSize=16);
grid on;
saveas(fig, "../plots/noisyAudio.png");
close;

fig = figure;
plot(0:Nv2-1, v2n);
xlabel("sample index [n]", FontSize=14);
ylabel("amplitude", FontSize=14);
title("Noise signal v_2[n]", FontSize=16);
grid on;
saveas(fig, "../plots/v2Noise.png");
close;

% rxnxn = estimate_autocorr(xn,true,"../plots/noisySpeechAutocorr.png");
rv2v2 = estimate_autocorr(v2n,true,"../plots/v2NoiseAutocorr.png");
rxnv2 = estimate_crosscorr(xn,v2n,true,"../plots/noisySpeechv2NoiseCrossCorr.png");
% rv2xn = estimate_crosscorr(v2n,xn,true,"../plots/noisySpeechv2NoiseCrossCorr.png");

% rv2v2 = xcorr(v2n,v2n);
% rxnv2 = xcorr(xn,v2n);

% rxnv2_cpy = zeros(2*Nx-1,1);
% rxnv2_cpy(1:Nx-1) = flip(rv2xn(Nx+1:2*Nx-1));
% rxnv2_cpy(Nx,1) = rv2xn(Nx,1);
% rxnv2_cpy(Nx+1:2*Nx-1) = flip(rv2xn(1:Nx-1));
% assert( sum(abs(rxnv2 - conj(rxnv2_cpy)), 'all') == 0 );

filtOrd = [4,8,12,16,20];
```

```
for M = filtOrd

    RXX = compute_corrmat(rv2v2, M);   % (M,M)
    % RYX = compute_corrmat(rxnv2, M);
    RYX = transpose(rxnv2(Nx:Nx+M-1));  % (L,M)

    Co = (RYX * inv(RXX))';         % (M,1)
    disp("Filter coefficients M="+M);
    disp(Co');
    v1nhat = zeros(Nx,1);
    for i=1:Nx
        if i < M
            X = transpose([flip(v2n(1:i))', zeros(1,M-i)]);
        else
            X = flip(v2n(i-M+1:i));
        end
        v1nhat(i) = Co' * X;
    end

    v1nhat2 = filter(Co,1,v2n);
    assert(sum(abs(v1nhat - v1nhat2), 'all') < 1e-10);
    %     v1nhat = rescale(v1nhat, -1, 1);

    fig = figure;
    plot(0:Nx-1, v1nhat);
    xlabel("sample index [n]", FontSize=14);
    ylabel("amplitude", FontSize=14);
    title("Estimated noise v_1[n] (Filter order M="+M+")", FontSize=16);
    grid on;
    saveas(fig, "../plots/estv1n_M"+M+".png");
    close;

    snhat = (xn - v1nhat);
%     snhat = rescale(snhat, -1, 1);
%     sound(snhat,Fs);
    audiowrite("../results/cleanAudio_M"+M+".wav", snhat, Fs);

    fig = figure;
    plot(0:Nx-1, snhat);
    xlabel("sample index [n]", FontSize=14);
    ylabel("amplitude", FontSize=14);
    title("Estimated clean audio s[n] (Filter order M="+M+")", FontSize=16);
    grid on;
    saveas(fig, "../plots/cleanAudio_M"+M+".png");
    close;

    % snhat2 = smoothdata(xn, "gaussian", 500);
```

```
    % sound(snhat2,Fs);
    % fig = figure;
    % plot(0:Nx-1, snhat2);
    % xlabel("samples [n]");
    % ylabel("amplitude");
    % grid on;
    % saveas(fig, "../plots/cleanSpeech2.png");
    % close;

    % compute SNR
    snrVal = snr(snhat, v1nhat);
    disp("SNR based on estimated audio and estimated noise for M"+M+":"+snrVal);
    snrVal2 = 10 * log10(sum(snhat.^2) / sum(v1nhat.^2));
    disp("SNR based on estimated audio and estimated noise for M"+M+":"+snrVal2);
end
```

3.2 estimate_autocorr.m

```
function [rxx1] = estimate_autocorr(xn, tosave, savepath)
    % estimate autocorrelation sequence using N samples
    N = size(xn, 1);
    rxx1 = zeros(2 * N - 1, 1);    % 2(N-1)+1

    for m = 0:N-1
        rxx1(N+m, 1) = sum(xn(1+m:N, 1) .* conj(xn(1:N-m, 1)));
    end
    rxx1(1:N-1, 1) = flip(rxx1(N+1:2*N-1, 1));
    rxx1 = rxx1 / N;

%     rxx2 = zeros(2 * N - 1, 1);    % 2(N-1)+1
%     for m = 0:N-1
%         rxx2(N+m,1) = sum(xn(1+m:N,1) .* conj(xn(1:N-m,1)));
%     end
%     for m = -(N-1):-1
%         rxx2(N+m,1) = sum(xn(1:N-abs(m), 1) .* conj(xn(abs(m)+1:N, 1)));
%     end
%     rxx2 = rxx2 / N;
%     disp(sum(abs(rxx1 - rxx2)));

    if tosave == true
        fig = figure;
        plot(-(N-1):(N-1), rxx1, LineWidth=1);
    %     plot(-(N-1):(N-1), rxx2, LineWidth=2); hold on;
        xlabel("lag [m]", FontSize=14);
        ylabel("Autocorrelation r_{yy}[m]", FontSize=14);
        title("Estimated Autocorrelation ("+N+" samples)", FontSize=14);
```

```
        grid on;
        saveas(fig, savepath);
        close;
    end
end
```

3.3 estimate_crosscorr.m

```
function [rxy] = estimate_crosscorr(xn, yn, tosave, savepath)
    % estimate autocorrelation sequence using N samples
    Nx = size(xn,1);
    Ny = size(yn,1);
    assert(Nx == Ny);
    N = Nx;

    rxy = zeros(2*N-1,1);    % 2(N-1)+1
    for m = 0:N-1
        rxy(N+m,1) = sum(xn(1+m:N,1) .* conj(yn(1:N-m,1)));
    end
    for m = -(N-1):-1
        rxy(N+m,1) = sum(xn(1:N-abs(m), 1) .* conj(yn(abs(m)+1:N, 1)));
    end
    rxy = rxy / N;

    if tosave == true
        fig = figure;
        plot(-(N-1):(N-1), rxy, LineWidth=1);
    %     plot(-(N-1):(N-1), rxx2, LineWidth=2); hold on;
        xlabel("lag [m]", FontSize=14);
        ylabel("Cross-correlation r_{xy}[m]", FontSize=14);
        title("Estimated Cross-correlation ("+N+" samples)", FontSize=14);
        grid on;
        saveas(fig, savepath);
        close;
    end
end
```

3.4 compute_corrmat.m

```
function [corr_mat] = compute_corrmat(ryy, P)
    N = (size(ryy, 1) + 1) / 2;

    corr_mat = zeros(P, P);
    for i = 1:P
        corr_mat(i, :) = ryy(N - i + 1 : N - i + P);
    end
```

```
    first_row = ryy(N:N+P-1);
    corr_mat2 = toeplitz(first_row);

    assert(size(corr_mat,1) == size(corr_mat2,1));
    assert(size(corr_mat,2) == size(corr_mat2,2));
    assert( sum(abs(corr_mat2 - corr_mat), 'all') < 1e-10 );
end
```