נושא:

Simulation for noise cancellation using LMS adaptive filter.

מנחים:
גולן עין צבי

פרופסור יוסף בן עזרא

מגישים:
רוסלן אוסמנוב - 327480026

טאל חיים - 312464639

תוכן עניינים

תקציר והסבר על המאמר

המאמר שלנו הוכיח את יישום אלגוריתם LMS על אות מורעש במסנן אדפטיבי, לצורך קבלת אות נקי ללא רעש של המנוע .

רקע:

רעשים (אותות בלתי רצויים) קיימים תמיד, בכל מקום ובכל מערכת.
ספציפית, כשמדברים על מערכות שידור גלי קול, ישנן 2 גישות להורדת רעשים.
האחת היא באופן פיזי מכאני חומרתי.
והשנייה באופן דיגיטלי תוכנתי מערכתי.
ל2 הגישות יתרונות וחסרונות אך לגישה הדיגיטלית יתרונות עדיפים בשל היעילות החסכון במשאבים והקומפקטיות שניתן להשיג וליישם אותה במערכות.
עם השנים הצליחו להגיע לרמות סינון מאוד גבוהות של רעש והמון פיתוחים נעשו בתחום.
אחת השיטות הדיגיטליות שפותחו לשם כך היא סינון אדפטיבי.

במאמר זה הוכיחו והראו איך אלגוריתם LMS , משפיע ומסנן אות מרעש לקבלת האות המקורי (עד ל8.9% שגיאה יחסית מהאות המקורי).
בתהליך המחקר לקחו 3 אותות (אות פגום עם רעש של מנוע, רעש של מנוע ואות המקורי להשוואה) עם קצב דגימה של 44100Hz .

הרעש מוכנס ככניסה למסנן הדיגיטלי y_2.

אות מורעש נכנס ככניסה y_1 שנכנס לסוכם עם מוצא המסנן דיגיטלי $y_{2_k^\wedge}$.

y_1 ו $y_{2_k^\wedge}$ עוברים פעולת הפרש והתוצאה e_k נכנסת כאינדיקציה חדשה לשינוי המשקלים של הפילטר.

פרמטר שינוי המסנן – גדול צעד (0<μ<1) נקבע מראש לפי ניסוי ותהייה על אופי התוצאות.
הפילטר מוגדר עם כמות טאפים (סדר מסנן, אצלינו 'M') קבועה מראש (גם כאן לפי ניסוי ותהייה).

אות היציאה הסופי של הLMS (הסיגנל המפולטר, אצלינו e_k) והאות המקורי (אצלינו 'signal') עוברים הפרש ביניהם והתוצאה של ההפרש נותנת אינדיקציה על יעילות הLMS בביטול הרעש בכך שמצפים לקבל הפרש אפסי ומבינים אם יש הבדל היכן ההבדל.
לאחר מכן מבצעים FFT לאות ללא הרעש ,לאות מורעש ולאות פגום כדי לזהות את תחום התדרים המבוטל.
[1]

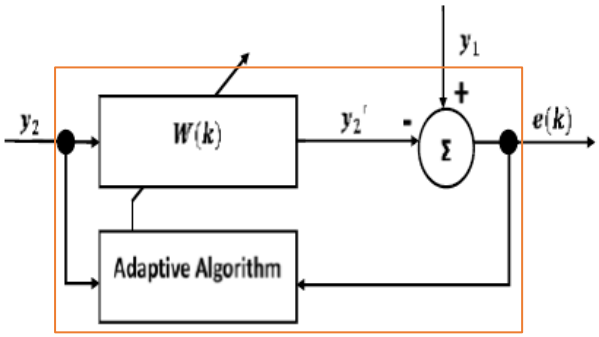
המסנן האדפטיבי:

- מסנן זה, יכול להתאים את המשקלים שלו בתגובה לשינויים באות הכניסה או לשינויים בסביבה שבה פועל המסנן.
- מטרתו היא להפחית רעש או הפרעות לא רצויות מאות, או לחלץ מידע שימושי מאות שנפגע על ידי רעש.
- מסננים אדפטיביים משתמשים באלגוריתם שמתאים את הפרמטרים שלו על סמך אות השגיאה בין הפלט הרצוי לפלט בפועל. האלגוריתם מעדכן את מקדמי המסנן בזמן אמת, כך שהמסנן יכול להסתגל לשינויים באות הכניסה.

מבנה: מסנן זה מורכב משני חלקים

- מסנן ספרתי עם מקדמים מתכוונים
- אלגוריתם מסתגל לצורך כיוונון המקדמים של מסנן

למסנן אדפטיבי יש שתי כניסות y_2 ו y_1, שני כניסות אלו נכנסות למסנן בו זמנית.
האות y_{1_k} הינו אות הרועש אשר מכיל את האות המבוקש s_k ואת הרעש n_k, ללא קורלציה ביניהם.
האות y_{2_k} מדידה של האות מורעש y_1 והוא נמצא ,בצורה כלשהיא ,בקורלציה עם n_k, כיוון שיש להם אותם מאפיינים סטטיסטיים (נמצאים באותה סביבה). y_{2_k} עובר דרך מסנן ספרתי W לצורך הפקת שיערוך $\hat{n_k} = y_{2_{\hat{k}}}$ של n_k. השיערוך של אות רצוי מתקבל בעזרת חיסור של המוצא המסנן הספרתי $\hat{n_k}$ מאות מורעש y_{1_k}, כלומר $\hat{s_k} = y_{1_k} - \hat{n_k} = s_k + n_k - \hat{n_k}$. נרצה ש $n_k - \hat{n_k}$ יהיה שווה לאפס ואז נקבל את האות נקי מהרעש.



שרטוט של מסנן אדפטיבי [1]

המטרה העיקרית בביטול רעש היא להפיק שערוך הטוב ביותר של אות הרצוי וזאת ניתן לקבל בעזרת כיוונון מקדמים של מסנן הספרתי ,דרך אלגוריתם מסתגל מתאים אשר מקטין את $\hat{s_k} = e_k$. אות המוצא e_k משרת שתי מטרות:

1. שערוך של אות הרצוי
2. משמש כאות שגיאה לצורך כיוונון המקדמים של מסנן הספרתי

סדר מסנן הספרתי (M)

אנו נרצה שיהיה לנו מסנן בעל סדר (אורך) נמוך ככל האפשר וזה לצורך לעבוד בזמן אמת, כמו כן אם השהייה במסנן תהיה ארוכה מידי והתכונות הסטטיסטיות של ערוץ כבר השתנו ,אזי לא תהיה משמעות למקדמי המסנן.

אלגוריתם LMS:

Least Mean Square – הינו פתרון יעיל ופשוט למסנן ווינר שבו המקדמים של המסנן הספרתי מתכוונים מדגימה לדגימה בצורה שבה השגיאה הריבועית הממוצעת עוברת תהליך של מינימיזציה, כלומר מנסים להקטין את פונקציה המטרה.

מקדמי של המסנן הספרתי W_k מתכוננים בעזרת אלגוריתם בזמן אמת בצורה יעילה כאשר אין צורך לדעת את מטריצת אוטו קורלציה - R ואת וקטור קרוס קורלציה- P ואין צורך גם לחשב את מטריצה הפוכה R^{-1}

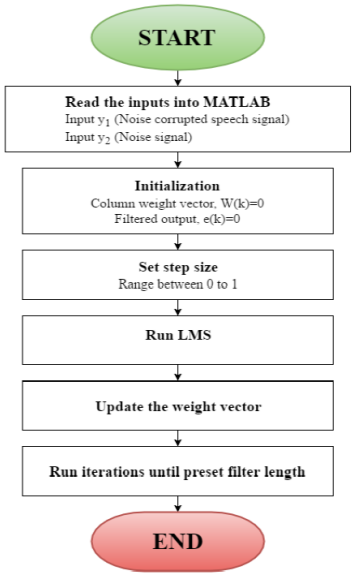
כמו במקרה של מסנן ווינר:

$$W = R^{-1}P \leftarrow \nabla = \frac{\partial J}{\partial W} = -2\mathrm{P} + 2\mathrm{RW} = 0$$

אלגוריתם לעדכון המשקלים מדגימה לדגימה נתון על ידי:

$$e_k = y_{1_k} - y_{2_k}\hat{} = y_{1_k} - w_k{}' y_{2_k}$$

$$w_{k+1} = w_k + 2\mu_k e_k y_{2_k}$$



אלגוריתם LMS בצורת סכמת בלוקים[1]

מימוש אלגוריתם LMS :

- שלב האתחול , $w_k = 0$, $e_k = 0$

יש לבצע את השלבים הבאים עד הקבלת המשקלים הרצויים:

- חישוב מוצע של מסנן $n_k\hat{} = \sum_{i=0}^{N-1} w_k(i) y_{2_{k-i}}$
- חישוב שגיאה $e_k = y_{1_k} - n_k\hat{}$
- עדכון משקלים $w_{k+1}(i) = w_k(i) + 2\mu_k e_k y_{2_{k-i}}$

- אנו שואפים להתכנס כמה שיותר מהר לנק' המינימום הגלובלי ולא לנק' מינימום מקומית
- אצלינו במאמר אין הכפלה ב 2

קצב שינוי המסנן (μ)

- קצב השינוי מאפשר לשלוט בקצב הלמידה המבוצעת למקדמי המסנן בכל איטרציה.
- ערך קטן של μ יגרום להתכנסות איטית אך גם מגדיל סיכוי להתכנס למינימום מקומי במקום מינימום גלובלי .
- ערך μ גדול יגרום להתכנסות מהירה והגדלת סיכוי חרגיה ממינימום הגלובלי.
- μ נע בין ערכים 0 עד 1.

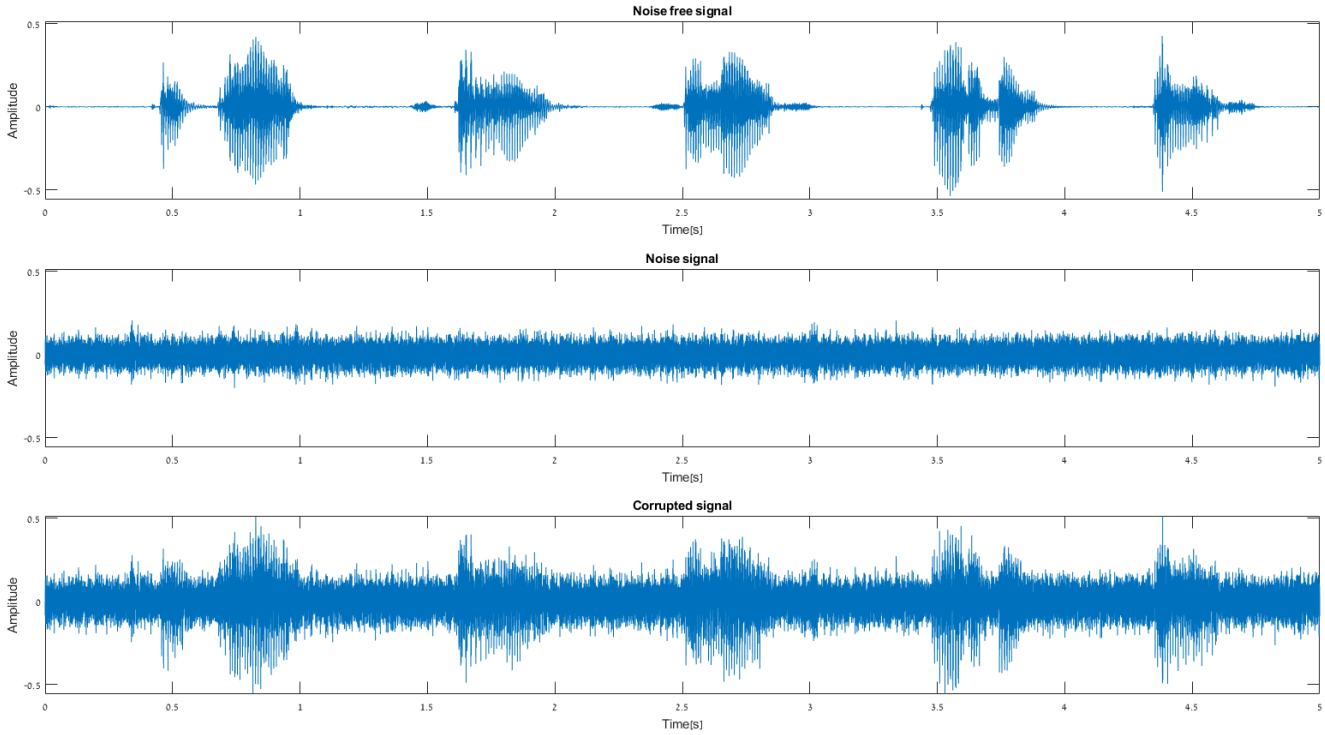
CODE AND PLOTS

מימשנו את המחקר מהמאמר בתור קוד עם קבצי של קול ורעש משלנו.
הקלטנו אות קול בתדר דגימה של 44100Hz במשך 5 שניות והקלטנו רעש של מנוע בתדר דגימה של 44100Hz במשך 18 שניות.

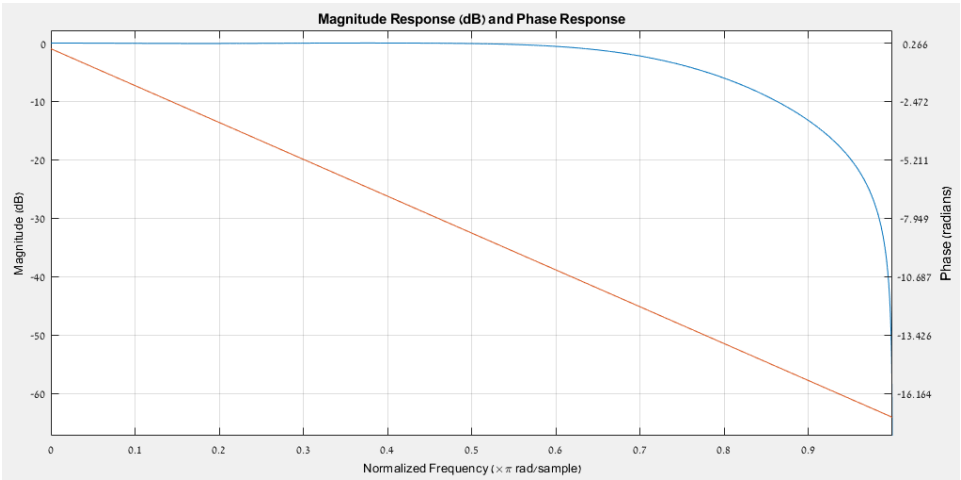
* הממצאים שנציג ככל הניראה יהיו שונים ממה שהוצג במאמר ,מעצם העובדה שהשתמשנו במיקרופון ומקור רעש אחר.
הקוד שלנו כולל :

- קליטת האותות והגדרת ערכים למודל.
- לקיחת חלק אקראי מתוך הרעש ומעבר דרך מסנן מעביר נמוכים וסכימה עם האות ללא הרעש לצורך יצירת האות הרצוי (y_1). בצורה זו הרעש שעובר דרך המסנן (y2) יהיה קורלטיבי בצורה כלשהיא לרעש המקורי לפני המסנן.
- העברה דרך אלגוריתם LMS שיצרנו.
- הצגת גרפים של האותות הרלוונטיים בזמן, ובתדר ושילוב (ספקטוגרמה).
- הצגת כמות אפשרויות שונות לסדרי מסנן וקצבי שינוי של מסנן לבחירת המפתח.
- הצגת גרף של שינוי משקולות כפונקציה של טאפים ואיטרציות לפי ערכי המסנן הספרתי שבחרנו.
- השוואה ואינדיקציה לכמות קרבה בין האות המסונן לבין האות המקורי ללא הרעש.

<u>נסביר תחילה על הגרפים שיוצאים לנו מהקוד ולאחר מכן נציג את הקוד.</u>

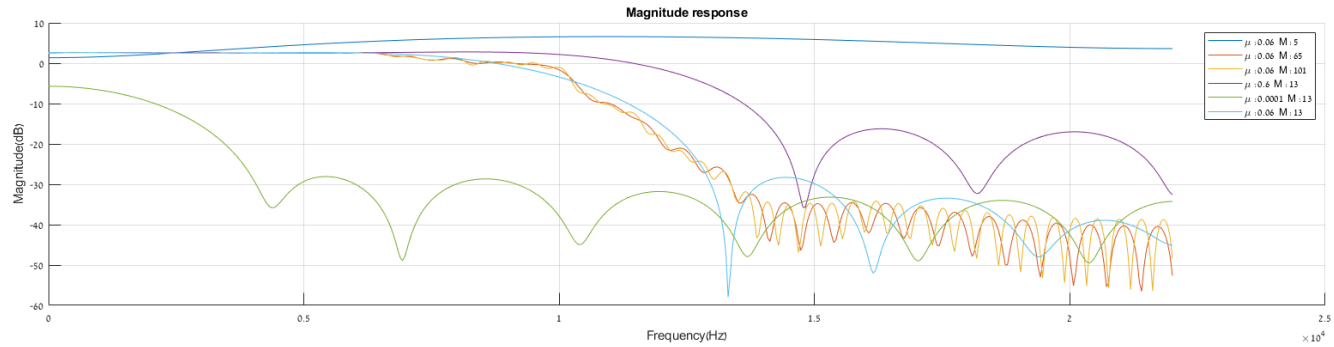


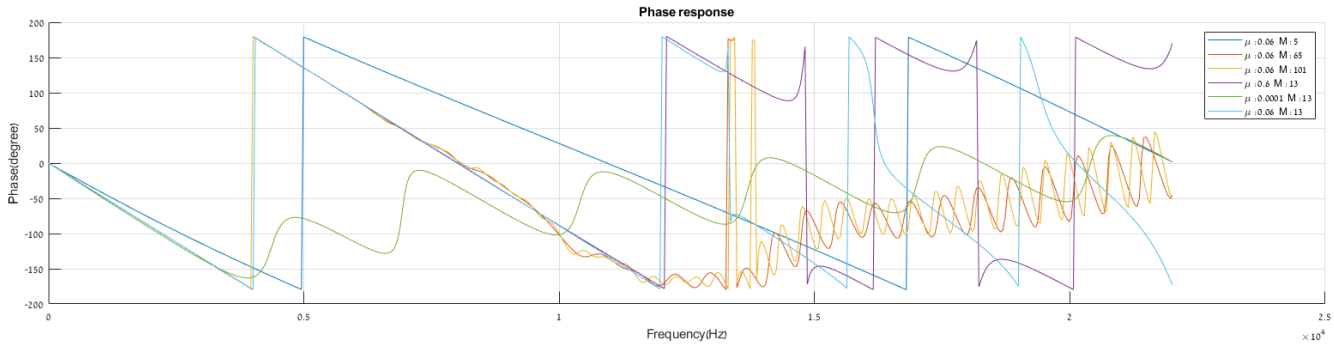
- הצגת גרפים במישור הזמן כאשר:
  גרף 1 – אות טהור ללא רעש.
  גרף 2 – אות רעש טהור.
  גרף 3 – אות חיבור בין אות מקורי ללא הרעש לבין הרעש שעבר דרך מסנן מעביר נמוכים לצורך יצירת השהייה ויצירת קורלציה כל שהיא בין הרעש טהור לרעש שיש באות פגום.

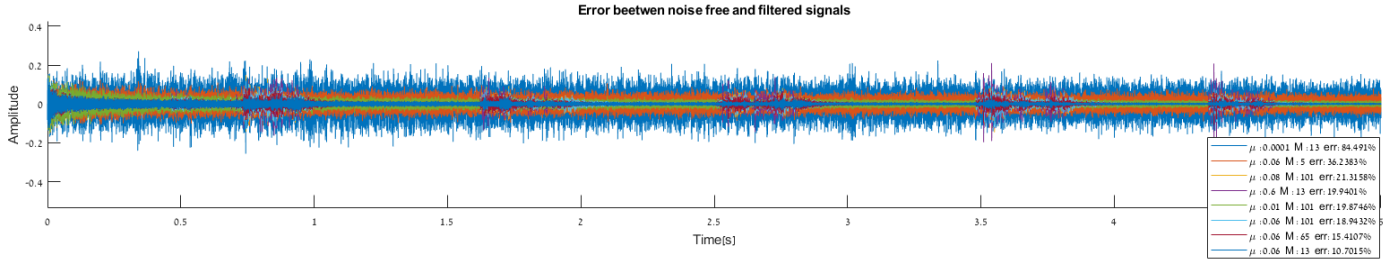


- תגובת תדר של מסנן FIR (LPF $\theta_c = 0.8\frac{\pi \cdot rad}{sec}$)

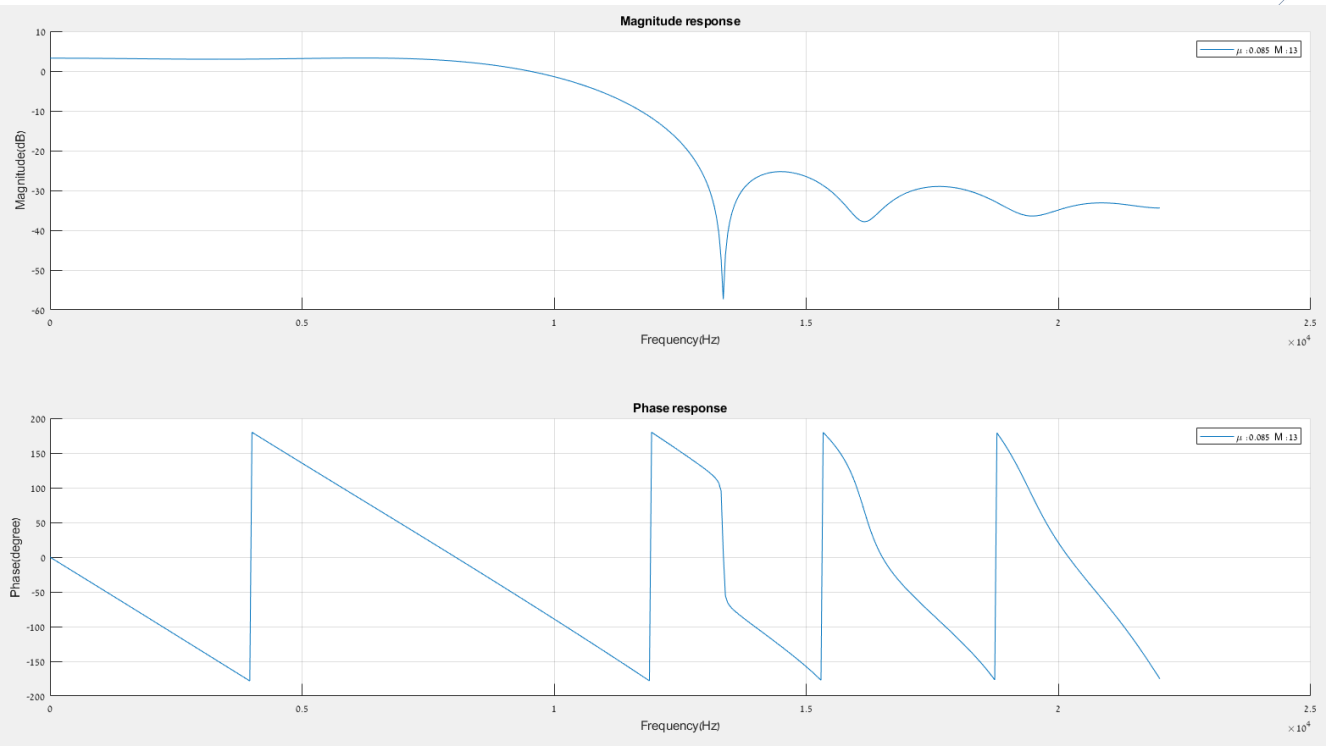
עבודת גמר – נושאים נבחרים בעיבוד אותות



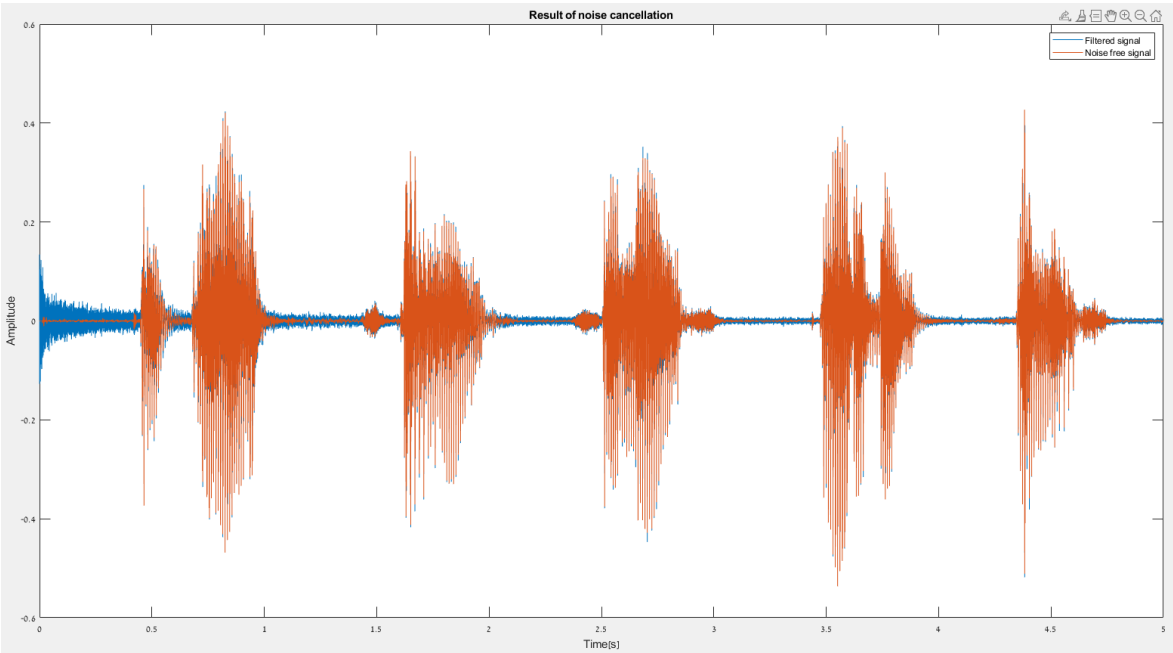




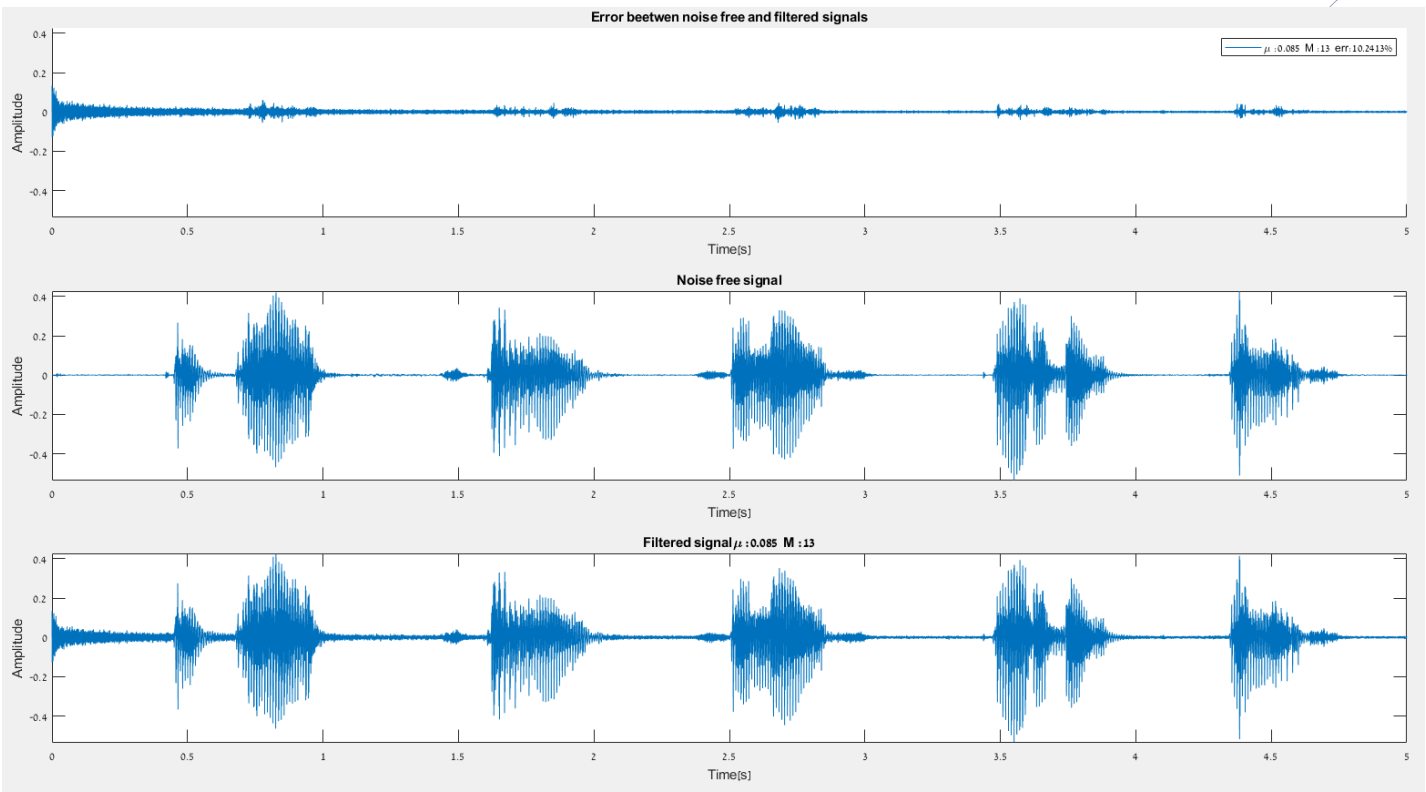
- גרפים במישור התדר של תגובת הפילטר דיגיטלי בסיום איטרציות של אלגוריתם, אמפליטודה והפאזה של המסנן האדפטיבי בסדרים שונים וגודל צעד שונים.
- גרף במישור הזמן שמראה את שגיאה בין אות המקורי לאות לאחר הפילטר למμ וM שונה.
- ניתן לראות תגובה שונה של כל מסנן האדפטיבי למμ וM שונה.
- ככל ש M גדול יותר יש השהייה גדולה יותר וזה גורר לשגיאה כי אות נהיה מוזז בזמן
- כאשר M קטן מדי מסנן אדפטיבי לא מצליח לשערך את הרעש נכון וכתוצאה השגיאה הינה גדולה
- ניתן לבחור גודל צעד וסדר המסנן לפי הצורך של המשתמש לאפליקציה ספציפית.



- תגובת התדר של מסנן אדפטיבי באיטרציה אחרונה (LPF)
- עבור $M = 13$ ו $\mu = 0.085$



- גרפים השוואה בין אות מקורי לאות מסונן במישור הזמן.
- ניתן לראות שהמסנן האדפטיבי מצליח לשחזר את אופי האות עם הזמן (איטרציות).
- אות מקורי ומסונן ביחד – כתום מקורי, כחול מסונן

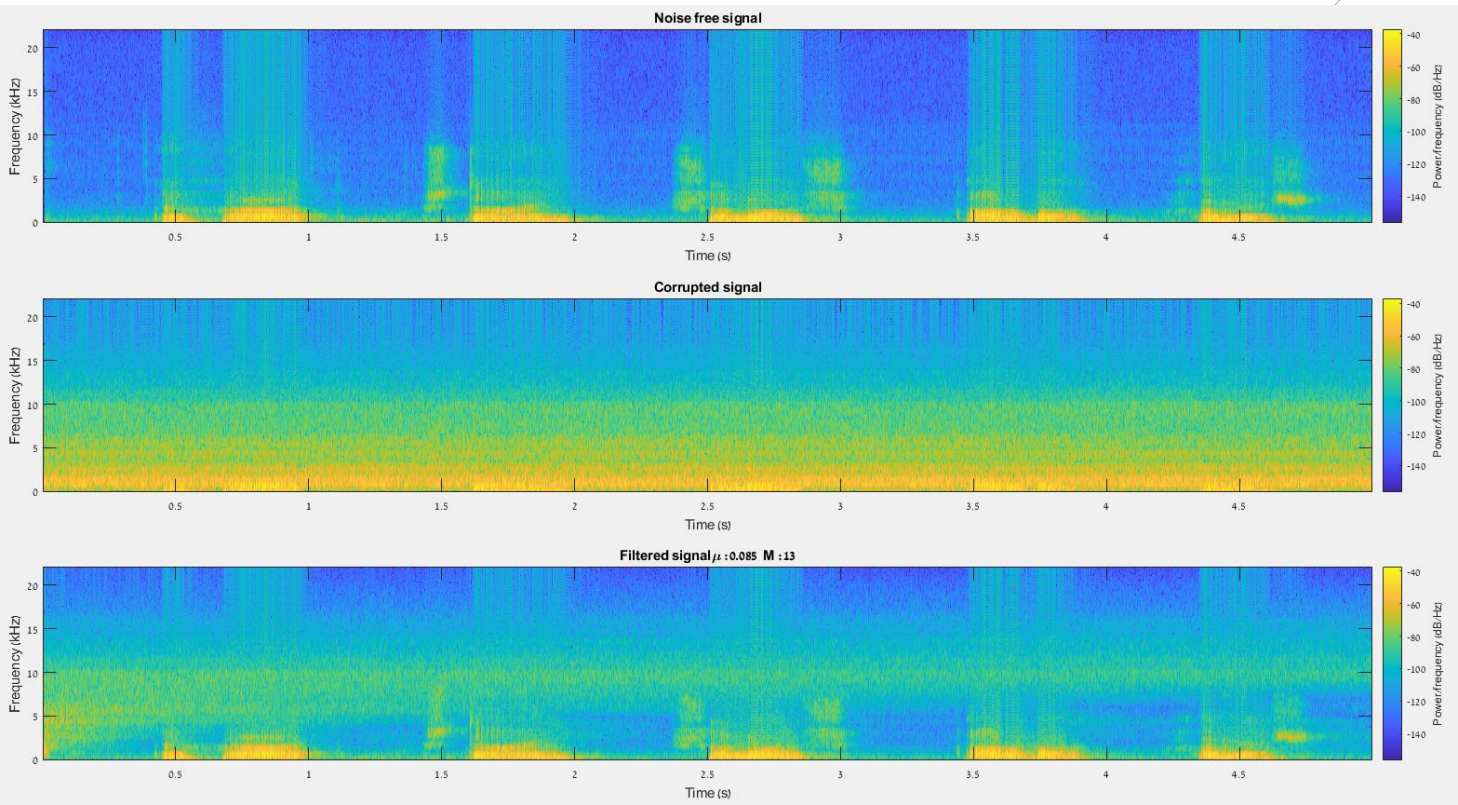


- גרף ראשון – שגיאה בין אות מקורי לאות מסונן
- גרף שני ושלישי– אות מקורי ומסונן.
- אחוז השגיאה היחסית חושב בקוד ונתן תוצאה של 10.24% לפי הנוסחה :

$$\boxed{\frac{\|e_{(k)} - s_k\|}{\|s_k\|} \cdot 100\%}$$

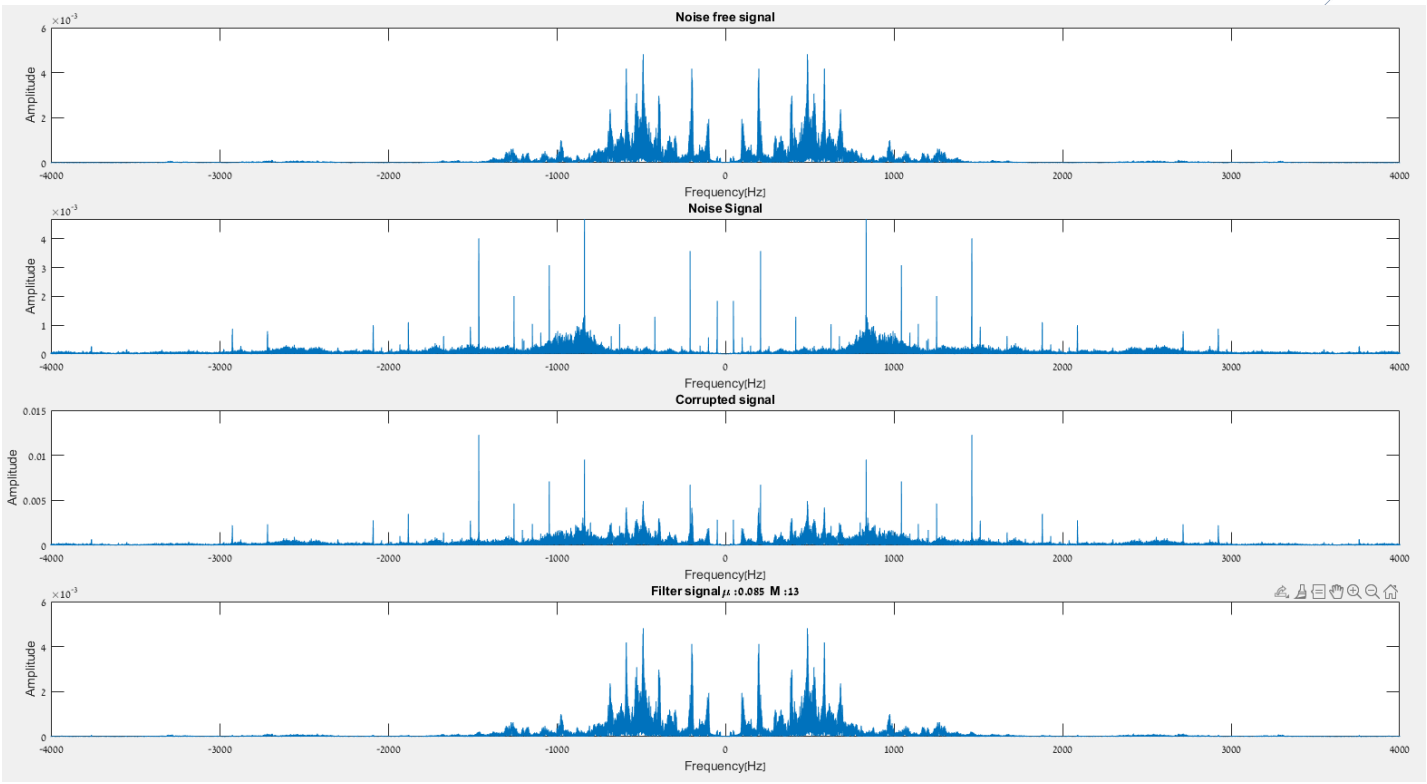
s_k - אות מקורי ללא הרעש

$e_{(k)}$ - אות לאחר הסינון בעזרת מסנן אדפטיבי

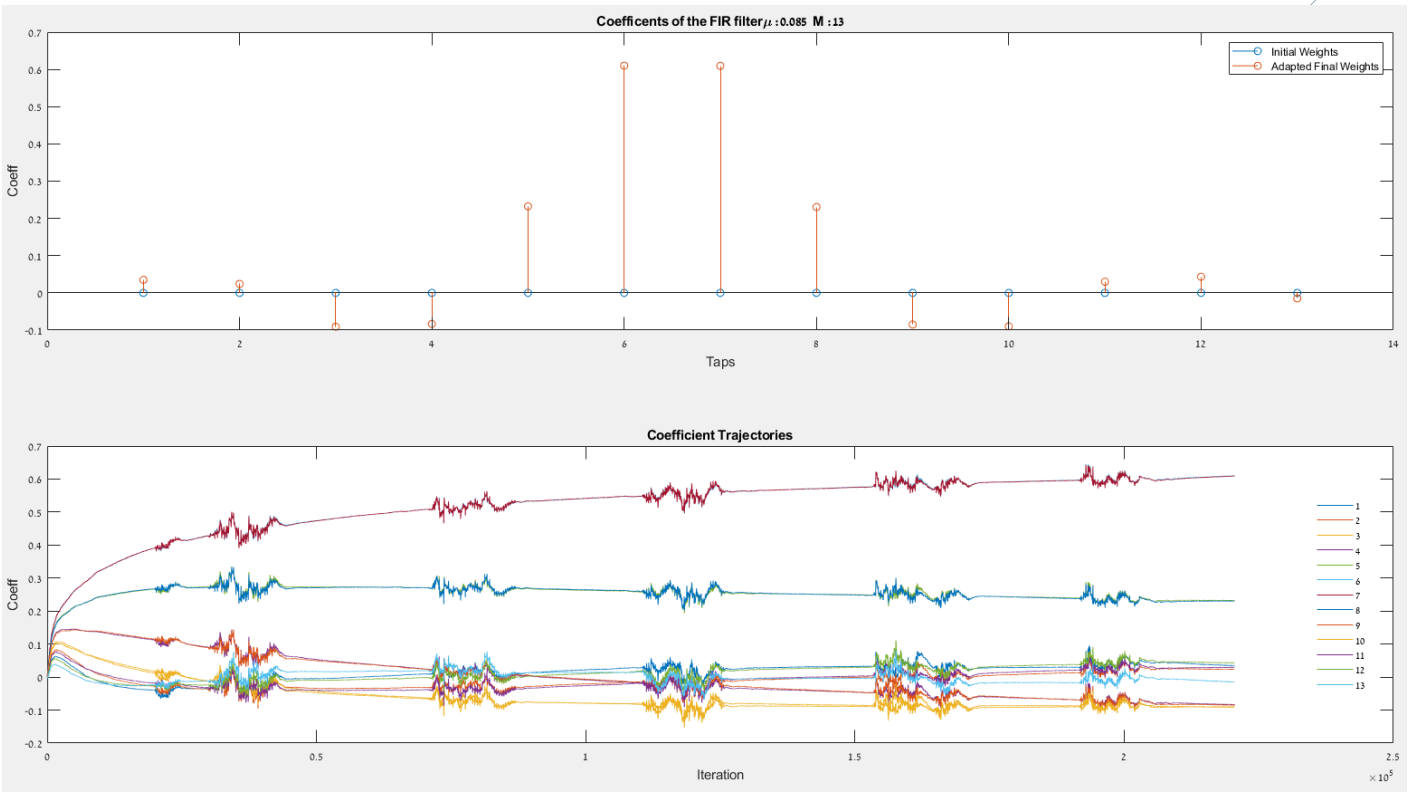


הערה – באופן כללי לא משתמשים בהתמרות FFT לאותות לא סטציונריים (התמרת FFT לא מסוגלת להבחין ולתת אינדיקציה מספיק טובה באותות אלו כי אין שום אינדיקציה לזמן), לכן עשינו הצגת ספקטוגרמה (STFT) של האותות.

- ניתן לראות את PSD בכל פרק זמן ותדר בספקטוגרמה לכל גרף, כאשר :
  גרף 1 – ספקטוגרמה לאות המקורי ללא הרעש
  גרף 2 – ספקטוגרמה לאות רועש
  גרף 3 – ספקטוגרמה לאות המסונן
- ע״פ הממצאים, ניראה כי ישנה התאמה בעוצמות התדרים של האות המקורי לאות המסונן, לאחר כל איטרציה מקבלים התאמה טובה יותר (סינון רעשים טוב יותר).



- הצגת גרפים במישור התדר (FFT) כאשר:
  גרף 1 – אות טהור ללא רעש.
  גרף 2 – אות רעש טהור.
  גרף 3 – אות רועש.
  גרף 4 – אות מסונן בעזרת מסנן אדפטיבי.
- ניתן לראות כי התדרים הגבוהים (שם נמצאים תדרי הרעש) מסוננים.



- גרפים להמחשת שינוי המשקולות של המסנן הספרתי עם $\mu = 0.085$ $M = 13$.
- גרף עליון מראה על שינוי המשקולות בין התחלה צבע כחול לסוף צבע כתום.
- גרף תחתון מראה על שינוי המשקולות לפי כל האיטרציות באלגוריתם האדפטיבי לעדכון משקולות של המסנן.

סיכום

אנו הצגנו מודל לסינון רעשים באמצעות אלגוריתם מסנן אדפטיבי של LMS.
הגרפים והחישוב לשגיאה יחסית מורים על כך שהאלגוריתם מצליח לעדכן את וקטור המשקלות של המסנן הספרתי עם הזמן כך שהשגיאה תצא מזערית ובזאת להגיע לסינון רעש כמעט אופטימלי.
לאחר חישוב אחוז ההפרש היחסי בין האות המקורי לאות התוצאה מהמודל ניראה כי התוצאה היא 10.24%.
בגלל שיש השפעה של עוצמות של סיגנלים על קצב ההתכנסות יש צורך בעדכון של גודל הצעד μ , לפתרון של בעיה זו ניתן להשתמש באלגוריתמים משוכללים יותר כמו RLS, NLMS.

ביבליוגרפיה

[1] J.-H. Lee, L.-E. Ooi, Y.-H. Ko, and C.-Y. Teoh, “Simulation for noise cancellation using LMS Adaptive Filter,” *IOP Conference Series: Materials Science and Engineering*, vol. 211, p. 012003, 2017.

MATLAB CODE

```
%In this model we will show the process of canceling noise from corrupted
%signal using an adaptive filter applied by LMS algorithem as an
%implementation to an article called:
%'Simulation for noise cancellation using LMS adaptive filter.'
```

The process included:

```
% 1. Import and export noise free and noise signal as a wav file.
% 2. Implement noise components on the signal file randomly and create correlated noise
% 3. Plot corrupted ,noise free and noise signal
% 4. Run LMS algorithm according to:
%      Diffirent Length of FIR filter (M)
%      Diffirent Step Size (mu)
% 5. Plot frequency responce of the last itteration coefficents
% 6. Plot Spectogram of filtered ,noise free and corrupted signals
% 7. Plots:
%      Absolute error beetwen noise free and filtered signal
%      Noise free signal
%      Filtered signal
% 8. Relative error beetwen original and filtered signal
% 9. Plots of signal and filter result combined
%10. Plots of the coeffs and weights:
%     Initial Weights,Adapted Final Weights
%     Coefficient Trajectories
%11. FFT:
%     Noise free signal
%     Filtered signal
%     Corrupted signal
%     Noise signal

close all
clear all
```

Load noise free and noise signals

```
[signal,Fs] = audioread('Signal.wav');
[noise,~] = audioread('Noise1.wav');

% Set the noise as a random configuration
index = randi(numel(noise) - numel(signal) + 1,1,1);
noiseSegment = noise(index:index + numel(signal) - 1);
% To ensure that the noise is correlated to desired , pass the noise through
% a lowpass FIR filter and then add the filtered noise to the signal.
filt = dsp.FIRFilter;
filt.Numerator = fir1(11,0.8);
fnoise = filt(noiseSegment);
freqz(filt)
%fnoise = circshift(noiseSegment,11); % shift noise for correlated noise
% Calculate the power components of the siganls
```

```
speechPower = sum(signal.^2);
noisePower = sum(fnoise.^2);
noise_factor =sqrt(speechPower/noisePower); %snr, to ensure that noise and signal have same power

% Define corrupted signal with noise factor
d = signal + noise_factor*fnoise;
corrcoef(fnoise,noiseSegment)
```

Plot corrupted, noise free and noise signal

```
figure(1)
dt = 1/Fs;
t = 0:dt:(length(signal)-1)*dt; % create time vector

subplot(3,1,1)
plot(t,signal);
title('Noise free signal');
xlabel('Time[s]');
ylabel('Amplitude');

subplot(3,1,2)
plot(t,noiseSegment);
title('Noise signal');
xlabel('Time[s]');
ylabel('Amplitude');

subplot(3,1,3)
plot(t,d);
title('Corrupted signal');
xlabel('Time[s]');
ylabel('Amplitude');
linkaxes([subplot(3,1,1) subplot(3,1,2) subplot(3,1,3)], 'xy');
```

LMS Adapt Filter

```
mu = input('Step size mu = '); % Set the step size
M = input('Length of sequence M = ');  % Filter length (num of taps)
model_info = strcat('\mu : ',string(mu) ,'  M : ',string(M));

% Initialization of weights
coeffs = zeros(M,1); % column vector of init weights
S.coeffs = coeffs; % insert weights to struct
S.step = mu; % insert step size to the struct

% Perform LMS-algo
[~,e,S] = LMSadapt(noiseSegment,d,S);
w = S.coeffs;
```

```
%    Trying to use LMS filter from build in in package function (We get tha same rusult)
```

```
% lms_nonnormalized = dsp.LMSFilter(M,'StepSize',mu,...
```

```
%        'Method','LMS','InitialConditions',coeffs);
% [~,e,w] = lms_nonnormalized(noiseSegment,d);
```

Frequency Response of Adaptive filter

```
figure(2)
[h,f] = freqz(w,1,[],Fs);
subplot(2,1,1);
hold on
plot(f,20*log10(abs(h)),'DisplayName',model_info); % we will use 20log10() for ploting the mag.
response in dB
title('Magnitude response')
grid on % turning the grid on
xlabel('Frequency(Hz)')
ylabel('Magnitude(dB)')
legend
subplot(2,1,2);
hold on
legend
plot(f,rad2deg(angle(h)),'DisplayName',model_info);
title('Phase response')
grid on
xlabel('Frequency(Hz)')
ylabel('Phase(degree)')
hold off
```

Spectogram

```
figure(3)
subplot(3,1,1)
spectrogram(signal,128,120,[],Fs,'yaxis' );
title('Noise free signal');
subplot(3,1,2)
spectrogram(d,128,120,[],Fs,'yaxis' );
title('Corrupted signal');
subplot(3,1,3)
spectrogram(e,128,120,[],Fs,'yaxis' );
title(strcat('Filtered signal ',model_info));
linkaxes([subplot(3,1,1) subplot(3,1,2) subplot(3,1,3)], 'xy');
%view(0,0)
```

Time Domain plots

```
figure(4)
subplot(3,1,1)
hold on
plot(t,e-signal,'DisplayName',model_info);% Filt.effectiveness
title('Error beetwen noise free and filtered signals');
xlabel('Time[s]');
ylabel('Amplitude');
legend
```

```
disp(['Relative error beetwen noise free and filtered signal :',num2str(norm(e-
signal)/norm(signal)*100) ,' %'])
hold off

subplot(3,1,2)
plot(t,signal);
title('Noise free signal');
xlabel('Time[s]');
ylabel('Amplitude');

subplot(3,1,3)
plot(t,e);
title(strcat('Filtered signal  ',model_info));
xlabel('Time[s]');
ylabel('Amplitude');
linkaxes([subplot(3,1,1) subplot(3,1,2) subplot(3,1,3)], 'xy');
% Combined plot of noise free and filter signal
figure(5)
plot(t,e,t,signal);
legend('Filtered signal','Noise free signal');
title('Result of noise cancellation');
xlabel('Time[s]');
ylabel('Amplitude');
```

Plot of the coeffs and weights

```
figure(6)
subplot(2,1,1)
stem(coeffs)
hold on
stem(w)
legend('Initial Weights','Adapted Final Weights');
title(strcat('Coefficents of the FIR filter ',model_info));
xlabel('Taps');
ylabel('Coeff');
hold off

subplot(2,1,2)
nn = length(e);
plot(1:nn,S.W(:,1:nn))
title('Coefficient Trajectories');
xlabel('Iteration');
ylabel('Coeff');
legend(string(1:M),'Location','best')
legend('boxoff')
```

Frequency domain plots

```
figure(7)
limit = [-4e3,4e3];% Relevant spectrum of regular speech frequency
subplot(4,1,1)
[FFT_amp,FFT_freq] = FFT(Fs,signal,0);
```

```
plot(FFT_freq,FFT_amp)
xlim(limit)
title('Noise free signal');
xlabel('Frequency[Hz]');
ylabel('Amplitude');

subplot(4,1,2)
[FFT_amp_n,FFT_freq] = FFT(Fs,noise,0);
plot(FFT_freq,FFT_amp_n)
xlim(limit)
title('Noise Signal');
xlabel('Frequency[Hz]');
ylabel('Amplitude');

subplot(4,1,3)
[FFT_amp_d,FFT_freq] = FFT(Fs,d,0);
plot(FFT_freq,FFT_amp_d)
xlim(limit)
title('Corrupted signal');
xlabel('Frequency[Hz]');
ylabel('Amplitude');

subplot(4,1,4)
[FFT_amp_filt,FFT_freq] = FFT(Fs,e,0);
plot(FFT_freq,FFT_amp_filt)
xlim(limit)
title(strcat('Filter signal  ',model_info));
xlabel('Frequency[Hz]');
ylabel('Amplitude');

linkaxes([subplot(4,1,1) subplot(4,1,2) subplot(4,1,3) subplot(4,1,4)], 'x');

sound(e,44100)

sound(d,44100)

sound(signal,44100)

sound(noise,44100)
```

Functions

```
function [yn,en,S] = LMSadapt(un,dn,S)
    mu = S.step;
    N = length(un); % number of samples un = dn
    yn = zeros(N,1); % initialize filter output vector (estimation y2')
    w = S.coeffs; % initialize filter coefficient vector
    en = zeros(N,1); % initialize error vector
    M = length(S.coeffs);
    S.W = zeros(M,N); % filter coefficient matrix for coeff. history
    for i = 1:N
      if i <= M % assume zero-samples for delayed data that isn't available
          k = i:-1:1;
          u = [un(k); zeros(M-numel(k),1)];
      else
          u = un(i:-1:i-M+1); % M samples of x in reverse order
      end
      yn(i) = w'*u; % filter output
      en(i) = dn(i) - yn(i); % error
      w = w + mu*en(i)'*u; % update filter coefficients
      S.W(:,i) = w; % store current filter coefficients in matrix
    end
    S.coeffs  = w ;
end
```

```
function [FFT_amp,FFT_freq] = FFT(Fs,signal,display_plot)
    N = 2^nextpow2(10*length(signal));
    yf_singal = abs(fftshift(fft(signal,N)));
    FFT_freq = linspace(-Fs/2,Fs/2,N);

    Norm_factor =1/length(signal);
    FFT_amp =(Norm_factor*yf_singal);
%   FFT_amp =20*log10((Norm_factor*yf_singal));
    if display_plot==1
        figure('Name','Fast Fourier Transform')
        plot(FFT_freq,FFT_amp)
        xlim([-8e3,8e3])
    end
end
```

Published with MATLAB® R2020b