

# L1. Procesarea semnalelor audio folosind functiile Matlab

---

## 1. Semnale audio

Semnalele audio se refera, în general, la semnale care pot fi percepute de om. Semnalele audio, de obicei, provin dintr-o sursă de sunet care vibrează în gama de frecvențe audio (20Hz-20kHz). Vibrațiile pun în mișcare aerul pentru a forma „valuri de presiune” care se propaga cu aproximativ 340 m/s. Urechile pot primi aceste semnale de presiune și le trimit creierului pentru recunoașterea acestora.

Există numeroase moduri de a clasifica semnalele audio. Dacă luăm în considerare **sursa de semnale audio**, le putem clasifica în două categorii:

- **Sunete produse de vietuitoare:** voci umane, latrat de caine, mieunat de pisica, orăcăitul broastelor etc. În particular, bioacustica este o știință inter-disciplinara, care investigheaza producerea sunetelor de catre vietuitoare și recepția lor de catre acestea.
- **Sunete de la non-vietuitoare:** Sunete de la motoarele auto, tunete, trântitul uși, instrumente muzicale etc.

Dacă luăm în considerare modul de repetare a semnalelor audio, acestea se pot clasifica în două categorii:

- **Sunete cvasi-periodice:** formele de undă sunt aproape periodice, astfel încât putem detecta perioada de repetitie (pitch). Exemple de astfel de sunete includ redarea monofonica a majoritatii instrumentelor muzicale (cum ar fi pianul, vioara, chitara etc) și vorbirea în anumite zone sau cantatul uman.
- **Sunete aperiodice:** formele de undă nu sunt formate din tipare evidente repetate (forme periodice), astfel că nu putem percepe o frecventa de repetitie stabila. Exemple de astfel de sunete includ tunete, batut din palme, parti nesonore în rostirea umana șamd.

În principiu, putem clasifica fiecare segment scurt de vorbire (cunoscut și sub numele de cadru, cu o lungime de aproximativ 20 ms) în două tipuri:

- **Segment sonor:** Acestea sunt produse de vibrația periodica a corzilor vocale, deci pot fi observate perioadele fundamentale într-un cadru. Mai mult decât atât, ca urmare a existenței perioadei fundamentale, poate fi estimata valoarea sa.
- **Segment nesonor:** Acestea nu sunt produse de vibrația corzilor vocale ci de fluxul rapid de aer expulzat prin intermediul tractului vocal. Deoarece aceste sunete sunt produse de un zgomot, cum ar fi fluxul de aer rapid, perioada fundamentala nu poate fi observata și nici o frecventa stabila nu poate fi detectata.

Semnalele audio reprezintă, de fapt, variatia presiunii aerului ca o funcție de timp, care este continua atât în timp cât și în amplitudine. Când se dorește achiziția semnalelor pentru stocarea într-un calculator, există mai multi parametri care trebuie luati în considerare:

1. **Rata de eșantionare:** aceasta reprezinta numărul de puncte de eșantionare pe secundă, în unitatea de Hertz (Hz). O rată de esantionare mai mare indica o calitate mai bună a sunetului, dar spațiul de stocare necesar este de asemenea mai mare. Ratele utilizate în mod obișnuit în eșantionare sunt prezentate în continuare:

- 8 kHz: calitatea vocii pentru telefoane și jucării;

- 16 KHz: Frecvent utilizate pentru recunoașterea vorbirii;
- 44,1 KHz: calitate de CD;

2. **Rezoluția:** Numărul de biți folosiți pentru a reprezenta fiecare eșantion din semnalul audio. Rezoluțiile utilizate în mod obișnuit sunt :

- 8-bit: Gama corespunzătoare este 0 - 255 sau + 127... -128 sau
- 16-bit: Gama corespunzătoare este -32768 - + 32767.

Cu alte cuvinte, fiecare punct esantionat este reprezentat de un număr întreg de 8 sau 16 biți. Cu toate acestea, în MATLAB, toate semnalele audio sunt transformate în virgulă mobilă în intervalul [-1, 1], pentru o manipulare mai eficientă. Dacă se dorește revenirea valorilor inițiale întregi, este nevoie ca valorile în virgula mobilă să fie înmultite cu  $2^{n_{bits}-1}$ , unde  $n_{bits}$  este rezoluția în biți.

3. **Nr. de Canale:** Mono pentru un singur canal și stereo pentru 2 canalele stereo.

## 2. Caracteristici acustice de baza ale semnalului vocal

Când sunt analizate semnalele audio, este adoptată, de obicei, metoda de analiză pe termen scurt, deoarece semnale audio sunt mai mult sau mai puțin stabile într-o perioadă scurtă de timp, de aproximativ 10-30 ms. Când se face analiza cadru cu cadru, pot exista suprapuneri între cadrele vecine pentru a surprinde schimbarea fină în semnale audio. De reținut este că fiecare cadru este unitatea de bază pentru analiza făcută. În fiecare cadru, se pot observa/defini următoarele caracteristici acustice.

- **Volumul:** Această caracteristică reprezintă intensitatea semnalului audio, care este corelat cu amplitudinea semnalelor. Uneori se referă, de asemenea, ca fiind energia sau intensitatea semnalelor audio.
- **Pitch:** Această caracteristică (înălțimea) reprezintă frecvența de repetiție a semnalelor audio, care poate fi reprezentată de frecvența fundamentală (FF sau F0), corespunzător perioadei fundamentale a semnalelor audio sonore.
- **Timbrul:** este un atribut multidimensional care depinde de mai multe variabile fizice. De multe ori timbrul este definit într-o manieră pur negativă ca "tot ceea ce nu este intensitate, pitch sau percepție spațială". Timbrul este definit ca "acel atribut de senzație auditivă, în care un ascultător poate judeca că două sunete prezentate în mod similar și având aceiași intensitate și același pitch sunt diferite". Există, în primul rând, conținutul în frecvență și profilul spectral al sunetului. Deoarece urechea umană are putere de rezoluție limitată în frecvență, vectorul compoziției spectrale poate fi adesea redus la un vector care reprezintă cantitatea de energie acustică instantanee în fiecare bandă critică (Plomp 1970), fără prea mari pierderi de informații perceptuale. În cele din urmă, anvelopa temporală a unui sunet instrumental, inclusiv atacul, caderea și modularea porțiunii stabile, influențează timbrul perceput într-o asemenea măsură care poate face sunetul unui instrument de nerecunoscut (Berger 1964).

Procedura de bază la extragerea de caracteristici acustice poate fi următoarea:

1. Se face împartirea pe cadre, astfel încât o secvență a semnalului audio este împartită într-un set de cadre. Durata de timp pentru fiecare cadru este de aproximativ 10-30 ms. Dacă durata cadrului este prea mare, nu putem surprinde caracteristicile rapid variabile în timp ale semnalului audio. Pe de altă parte, dacă durata cadrului este prea mică, atunci nu putem extrage caracteristici acustice valide. În general, un cadru ar trebui să conțină mai multe perioade fundamentale ale semnalului audio. De obicei, dimensiunea cadrului (în număr de eșantionate) este egală cu puterile lui 2 (cum ar fi 256, 512, 1024 etc), astfel că este potrivită pentru transformata Fourier rapidă.

2. Dacă se dorește reducerea diferențelor dintre cadre vecine, se permite suprapunerea acestora. De obicei, suprapunerea este de obicei între  $1/2$  și  $2/3$  din cadru original. Cu cât suprapunerea este mai mare, cu atât numărul calculelor crește.
3. Presupunând că semnalele audio într-un cadru sunt cvasistaționare, putem extrage diferiți parametri sau caracteristici cum ar fi numărul de treceri prin zero, volum, pitch, MFCC, LPC etc.

Lucrarea de față își propune o prezentare succintă a funcțiilor din MATLAB care pot fi folosite pentru preluarea, stocarea sau redarea semnalului audio, cât și a metodelor de bază pentru procesarea semnalului audio. Funcțiile Matlab descrise vor fi însoțite de exemple.

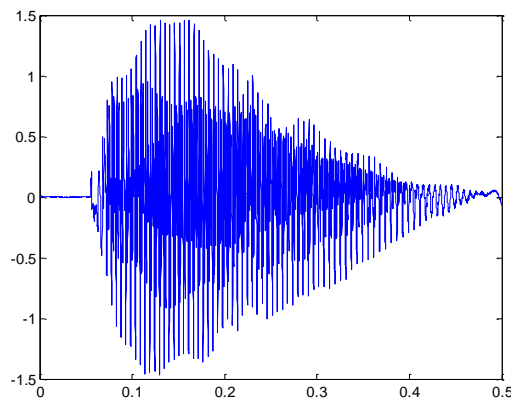
### 3. Procesarea semnalului audio folosind funcțiile Matlab

Acest capitol introduce câteva dintre cele mai importante funcții din cadrul mediului Matlab, funcții ce sunt utilizate pentru procesarea semnalului audio. Astfel, vor fi prezentate următoarele funcții pentru:

1. Citirea și afișarea fișierelor audio;
2. Redarea și afișarea semnalelor audio;
3. Înregistrarea de la microfon;
4. Salvarea fișierelor .wav.

#### 3.1 Citirea fișierelor .wav

În MATLAB se pot citi fișiere de tip .wav prin intermediul comenzii "wavread". Următorul exemplu citește fișierul "boy.wav" și afișează forma de undă al acestuia.



Forma de undă a rostirii "boy"

```
[y, fs]=audioread('boy.wav');
sound(y, fs);           % Reda semnalul audio
time=(1:length(y))/fs; % Vectorul timp pe axa X
plot(time, y);         % Afișează forma de undă în timp
```

În exemplul de mai sus, "fs" este frecvența de eșantionare, care este 16000, în acest caz. Acest lucru indică faptul că există 16000 de eșantioane pe secundă atunci când sunetul a fost înregistrat. Vectorul "y" este un vector coloană care conține eșantioanele semnalului vocal. Putem folosi comanda "sound(y, fs)" pentru a reda semnalul audio citit din fișier.

"time" este un vector de timp, în care fiecare element corespunde la momentul fiecărui eșantion. Prin urmare, putem afișa "y" vs. "t" adică  $y(t)$ , pentru a vizualiza forma de undă directă.

Cele mai multe semnale audio sunt digitizate cu o rezoluție de 8 sau 16 biți. Dacă vrem să știm rezoluția de biți a fișierului de intrare .wav, putem folosi parametri suplimentari de ieșire la funcția "audioread" pentru a obține informații, cum ar fi:

```
[y, fs, nbits]=audioread('girl.wav');
```

TABELUL 1: Formatele audio disponibile pentru a fi citite prin comanda Matlab audioread

Audio File Format	Description	File extension
WAVE	Raw audio	.wav
OGG	OGG vorbis	.ogg
FLAC	Lossless audio compression	.flac
AU	Raw audio	.au
AIFF	Raw audio	.aiff,.aif
AIFC	Raw audio	.aifc
MP3	MPEG1 Layer 3, lossy compressed audio	.mp3
MPEG4 AAC	MPEG4, lossy compressed audio	.m4a, .mp4

Mai mult decât atât, dacă vrem să știm durata unui fișier audio, putem folosi direct funcția "length(y)/fs". Următorul exemplu poate obține cele mai importante informații dintr-un fișier .wav.

```
>> information = audioinfo('girl.wav');
```

```
>> information = audioinfo('girl.wav');
>> information
```

```
information =
```

```

      Filename: 'C:\Users\eugen\Documents\MATLAB\Colea\girl.wav'
CompressionMethod: 'Uncompressed'
      NumChannels: 1
      SampleRate: 16000
    TotalSamples: 8000
      Duration: 0.5000
         Title: []
        Comment: []
         Artist: []
    BitsPerSample: 16
```

Din forma de undă de mai sus data ca exemplu, se poate observa că semnalul audio este normalizat, amplitudinea având valori între -1 și 1. Cu toate acestea, fiecare eșantion este reprezentat de un număr întreg pe 8/16 biți. Cum sunt corelate între ele? Mai întâi de toate, trebuie să știm următoarea convenție:

- a) Dacă un fișier .wav are o rezoluție de 8 biți, atunci fiecare esantion este stocat ca un întreg fara semn între 0 și 255 ( $2^8-1$ ).
- b) Dacă un fișier .wav are o rezoluție de 16 biți, atunci fiecare punct esantionat este stocat ca un întreg cu semn între -32768 ( $2^{15}$ ) și 32767 ( $2^{15}-1$ ).

Din moment ce aproape toate variabilele din MATLAB au tipul de date de "double", prin urmare, toate eșantioanele sunt convertite într-un număr în virgulă mobilă între [-1 și 1), pentru manipulare/procesare usoara. Prin urmare, pentru a prelua valorile originale întregi ale semnalelor audio, se poate proceda după cum urmează.

- a) Pentru rezoluție de 8-biti, se multiplica "y" (valoarea obținută prin wavread) cu 128 și apoi, se aduna 128.
- b) Pentru 16-bit rezoluție, se multiplica "y" (valoarea obținută prin wavread) cu 32768.

Se poate folosi, de asemenea, comanda "wavread" pentru a citi un fișier .wav stereo. Variabila returnata va fi o matrice de 2 coloane, fiecare conținând semnalele audio de la un singur canal.

### 3.2 Redarea semnalului audio (playback)

Odată ce putem citi fișierul audio în MATLAB, putem începe procesarea semnalelor audio prin modificarea intensităților lor, sau modificarea ratelor de eșantionare a acestora, șamd. După ce semnalele audio sunt procesate, este nevoie de redarea lor pentru o inspecție audio. Comanda de baza pentru redarea semnalelor audio este "sound sau play". Următorul exemplu poate încărca un flux de date audio din fisierul "handel.mat" și reda imediat semnalul audio incarcat.

```
load handel.mat % Incarca semnale stocate in handel.mat
p = audioplayer(y, Fs); % creează un obiect audioplayer pentru semnalul Y, folosind rata de eșantionare l
play(y, Fs); % Reda semnalul audio;
```

```
[y, fs]=audioread('do.wav');
sound(y, fs); % Reda semnalul cu amplitudinea originala
% fara parametrul Fs se reda la 8kHz
```

Deoarece volumul de redare este determinat de amplitudinea semnalelor audio, putem schimba amplitudinea pentru a modifica volumul, după cum urmează.

```
sound(3*y, fs); % Reda semnalul amplificat de 3 ori
```

În exemplul de mai sus, deși avem creștere în amplitudine cu un factor de 3, intensitatea percepută de către urechea umana nu este de factorul 3. Acest lucru servește pentru a ilustra faptul că percepția volumului nu este proporțională liniar cu amplitudinea. De fapt, aceasta este proporțională cu logaritmul amplitudinii.

Când semnalele de intrare sunt în afara acestui interval (prea mare sau prea mic), putem auzi "sunetul intrerupt". Se poate încerca acest lucru executand comanda "sound(1000\*y,fs)" pentru a auzi rezultatul.

Dacă vom schimba frecvența de eșantionare în timpul redării, aceasta va afecta durata de timp, precum și frecvența pitch perceputa. În următorul exemplu, vom crește frecvența de eșantionare treptat, astfel încât se va auzi un sunet scurt, cu o frecvența pitch, similar cu sunetul personajului din desenele animate Disney Donald Duck.

```
[y, fs]=audioread('church.wav');
p=audioplayer(y,0.5*fs)
play(p); % Reda semnalul la ½ din viteza normala
p=audioplayer(y,2*fs)
play(y, 2.0*fs); % Reda semnalul la o viteza de 2 ori mai mare
```

Pe de altă parte, dacă am reduce rata de eșantionare treptat, vom avea sunete mai lungi și cu frecvența fundamentală mai scăzută, iar în cele din urmă acesta va suna ca un muget de vacă.

```
[y, fs]=audioread('church.wav');
sound(y, 1.0*fs); % Reda semnalul la viteza normala
sound(y, 0.5*fs); % Reda semnalul la o viteza 0.5 din fs
```

Dacă am inversa semnalul audio prin înmulțirea sa cu -1, percepția va fi exact la fel ca originalul. Acest lucru, de asemenea, servește pentru a demonstra că *percepția umană a semnalului audio nu este afectată de fază*. Cu toate acestea, în cazul inversării semnalului audio pe axa timpului, atunci acesta va suna ca o limbă necunoscută ca în următorul exemplu.

```
[y, fs]=audioread('hall.wav');
P=audioplayer(-y, fs) ;
play(y); % Reda semnalul inversat pe axa Y (sus-jos)
p=audioplayer(flipud(y), fs); % Reda semnalul oglindit dreapta-stanga
%("reverse speech")
```

### Subeșantionarea unui fișier audio.

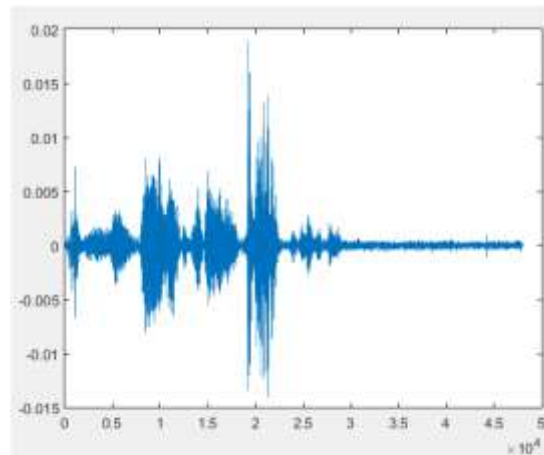
Downsampling cu un factor de doi se obține cu un fișier audio prin eliminarea fiecărei al 2-lea eșantion. Un exemplu ilustrativ de implementare a subeșantionării cu un factor de 2 este prezentat mai jos

```
[y,fs]=audioread('hall.wav');
sound(y,Fs);
y2 = downsample(y,2);
sound(y2,Fs);
subplot(2,1,2);plot(y2);
subplot(2,1,1);plot(y);
linkaxes;
```

### 3.3 Inregistrarea sunetelor de la microfon

Acest exemplu arată cum se înregistrează intrarea microfonului, se redă înregistrarea și se stochează semnalul audio înregistrat într-o matrice numerică. Mai întâi trebuie să conectați un microfon la sistem.

În exemplul de mai jos, "durata\*fs" este numărul de eșantioane care urmează să fie înregistrate. Eșantioanele înregistrate sunt stocate în variabila "y", care este un vector de dimensiune 48000x1. Tipul de date al lui "y" este dublu și spațiul de memorie ocupat de "y" este 256Kbytes. În acest exemplu numărul de canale este 1 și tipul de date pentru punctele eșantionate este double..



```

fs=16000;           % Frecventa de esantionare
nbits=16;
nChannels=1;
duration=3;        % Durata de inregistrare
arObj=audiorecorder(fs, nbits, nChannels);
fprintf('Apasa orice tasta pentru a incepe inregistrarea a %g secunde...',
duration); pause
fprintf('Se inregistreaza...');
recordblocking(arObj, duration);
fprintf('Inregistrare terminata.\n');
fprintf('Apasa orice tasta pentru a reda inregistrarea...'); pause;
fprintf('\n');
play(arObj);
fprintf('Se afiseaza forma de unda a semnalului inregistrat\n');
y=getaudiodata(arObj); % Ia datele audio esantionate
plot(y);           % Afiseaza forma de unda

```

Apasa orice tasta pentru a incepe inregistrarea a 3 secunde...Se inregistreaza...Inregistrare terminata.

Apasa orice tasta pentru a reda inregistrarea...

Se afiseaza forma de unda a semnalului inregistrat.

### 3.4 Salvarea/ scrierea fisierelor audio

Putem scrie fisiere audio utilizând comanda MATLAB "audiowrite".

Comanda este: **audiowrite(FILENAME,Y,FS)**

scrie datele Y intr-un fișier audio specificat prin numele de fișier FILENAME, cu o frecvență de eșantionare de FS Hz.

Datele stereo trebuie să fie specificate ca o matrice cu două coloane. Datele multicanal trebuie specificate ca o matrice cu N coloane.

În acest exemplu, vom stoca datele audio cu tipul "uint8" in fișierul "test.wav". Vom invoca apoi aplicația corespunzătoare pentru redarea fișierului. Deoarece variabila "y" pentru comanda "wavwrite" ar trebui să fie de tipul double în intervalul [-1, 1], este nevoie de executarea unor conversii în cazul în care datele înregistrate sunt de alte tipuri de date, cum ar fi "single", "int16", sau "uint8". Aici este tabelul pentru conversie.

```

fs=11025;           % Frecventa de esantionare
nbits=16;
nChannels=1;
duration=2;        % Durata de inregistrare

```

```

arObj=audiorecorder(fs, nbits, nChannels);
waveFile='test.wav'; % Fisierul wav care va fi salvat
fprintf('Apasa orice tasta pentru a incepe inregistrarea a %g secunde...', duration);
pause;
fprintf('Se inregistreaza...');
recordblocking(arObj, duration);
fprintf('Inregistrare terminata.\n');
fprintf('Apasa orice tasta pentru a salva inregistrarea in %s...', waveFile); pause;
fprintf('\n');
play(arObj);
fprintf('Se afiseaza forma de unda a semnalului inregistrat\n');
y=getaudiodata(arObj); % Ia datele audio esantionate
plot(y); % Afiseaza forma de unda
audiowrite(waveFile,y, fs);
fprintf('S-a terminat scrierea fisierului %s\n', waveFile);
fprintf('Apasa orice tasta pentru a reda semnalul scris %s...\n', waveFile);
dos(['start ', waveFile]); % Porneste aplicatia pentru fisierul .wav

```

### 3.5. Controlul volumului in fisierele audio

Intensitatea sonoră a semnalelor audio este cea mai proeminentă caracteristică pentru percepția auditivă umană. În general, există mai mulți termeni similari care sunt utilizați în mod obișnuit pentru a descrie **tăria** semnalelor audio: **volum, intensitate sau energie**. Din motive de coerență, aici vom folosi termenul de "volum" pentru a descrie intensitatea. Practic, volumul este o caracteristică acustică, care este corelată cu amplitudinile eșantioanelor dintr-un cadru. Pentru a defini volumul cantitativ, putem folosi două metode pentru a calcula volumul unui cadru dat:

1. Suma esantioanelor in valoare absoluta din fiecare cadru:

$$volume = \sum_{i=1}^n |s_i|$$

unde  $s(i)$  este al  $i$ -lea esantion într-un cadru, iar  $n$  este dimensiunea cadrului. Această metodă necesită doar operații întregi și este potrivită pentru sisteme low-end, cum ar fi microcontrolerele.

2. Sau cel dat de formula:

$$volume = 10 * \log_{10} \sum_{i=1}^n s_i^2$$

Această metodă necesită calcule în virgulă mobilă, dar este (mai mult sau mai puțin) liniar corelată cu percepția umană a intensității semnalelor audio. (<http://www.phys.unsw.edu.au/~jw/dB.html>)

Câteva dintre caracteristicile intensității sonore sunt prezentate în cele ce urmează:

- Pentru înregistrarea într-un loc liniștit folosind un microfon uni-directional, volumul sunetelor sonore este de obicei mai mare decât cel al sunetelor nesonore, iar volumul sunetelor nesonore este de obicei mai mare decât cea a zgomotului ambiental (cu toate acestea, acest lucru nu este aplicabil înregistrării prin intermediul microfonului omni-directional);
- Volumul este foarte mult influențat de setările microfonului, mai ales câștigul microfonului;



- Volumul este de obicei folosit pentru detectarea regiunilor de activitate a vocii (voice activity detection).

Înainte de a calcula volumul, se va efectua, de obicei, o eliminare a offsetului (prin scăderea simplă a mediei cadrului din fiecare esanțion), pentru a elimina un potențial DC.

Pentru metoda 1, vom aplica, de obicei, scăderea medianei pentru eliminarea offsetului.

Pentru metoda 2, vom aplica, scăderea mediei pentru ajustarea zeroului.

Utilizarea celor două metode pentru calculul volumului este ilustrat de exemplul următor.

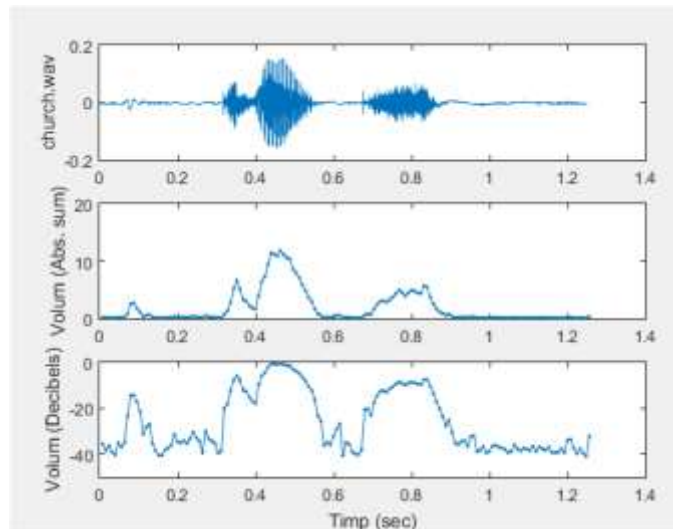
```

waveFile='church.wav';
frameSize=256; overlap=128;
[y, fs]=audioread(waveFile);
fprintf('Durata of %s is %g sec.\n', waveFile, length(y)/fs);
frameMat=buffer(y, frameSize, overlap);
frameNum=size(frameMat, 2);      % Calculeaza volumul cu metoda 1
volumel=zeros(frameNum, 1);
for i=1:frameNum
    frame=frameMat(:,i);
    frame=frame-median(frame); % zero-justified
    volumel(i)=sum(abs(frame)); % metoda 1
end

volumel2=zeros(frameNum, 1);      % Calculeaza volumul cu metoda 2
for i=1:frameNum frame=frameMat(:,i);
    frame=frame-mean(frame);      % zero-justified
    volumel2(i)=10*log10(sum(frame.^2)+realmin); % metoda 2
end
sampleTime=(1:length(y))/fs;
frameTime=((0:frameNum-1)*(frameSize-overlap)+0.5*frameSize)/fs;
subplot(3,1,1);
plot(sampleTime, y);
ylabel(waveFile);
subplot(3,1,2);
plot(frameTime, volumel, '-');
ylabel('Volum (Abs. sum)');
subplot(3,1,3); plot(frameTime, volumel2, '-');
ylabel('Volum (Decibels)'); xlabel('Timp (sec)');

```

Cele două metode de calcul a intensității sunt doar o aproximare a percepției umane. Cu toate acestea, intensitatea sunetului se bazează pe percepția umană și ar putea exista diferențe semnificative între "intensitatea calculată" și "intensitatea percepută". De fapt, intensitatea percepută este foarte mult afectată de frecvența, precum și de timbrul semnalelor audio.



Calculul volumului prin metodele 1 si 2 („church”)

#### 4. Mersul lucrării.

Pe baza celor studiate anterior în cap.1-3 rezolvați problemele și cerințele următoare și realizați raportul cu rezultate, capturi, cod etc.

4.1. Obțineți informații dintr-un fișier audio mono. Scrieți un script MATLAB care poate citi fișierul "church.wav" și va afișa următoarele informații în acest script:

Numărul de puncte de eșantionare.

Rata eșantionare

Rezoluție Biti

Numărul de canale

Durata de timp a înregistrării (în secunde)

4.2. Înregistrare Wave. Scrieți un script MATLAB pentru a înregistra 10 de secunde de vorbire, cum ar fi "Numele meu este X....Y.... și sunt un student la master în anul I TM la departamentul de Comunicatii al Universitatii Tehnice din Cluj-Napoca". Salvați înregistrarea ca myVoice.wav. Alți parametri de înregistrare sunt: Frecvența de eșantionare = 16 kHz, rezoluție biți = 16 biți. Script-ul va permite indicarea răspunsurilor la următoarele întrebări, în cadrul ferestrei MATLAB.

Cât spațiu este ocupat cu date audio în spațiul de lucru MATLAB?

Ce tip de date au datele audio?

Cum vă calculați cantitatea de memorie necesară din parametrii de înregistrare?

Care este dimensiunea fișierului myVoice.wav?

Câți octeți sunt utilizați în myVoice.wav pentru a înregistra date, altele decât datele audio în sine?

4.3. Manipularea semnalului audio: Scrieți un script MATLAB pentru a înregistra rostirea ta de "azi e ziua mea de naștere". Încercați să explicați efectul de redare observați după ce încercați următoarele operații asupra semnalelor audio:

Inmultiti semnalele audio cu -1.

Inversati semnalele audio pe axa timpului.

Inmultiti semnalele audio cu 10.

Înlocuiți fiecare esantion cu rădăcină sa pătrată.

Înlocuiți fiecare esantion cu pătratul său.

Taierea/limitarea semnalului, astfel încât eșantioanele din gama [-0.5, 0.5] sunt setate la zero.

Modificati forma de undă astfel încât eșantioanele din intervalul [-0.5, 0.5] sunt setate la zero, iar eșantioanele din afara gamei de [-0.5, 0.5] sunt mutate spre zero, cu cantitatea de 0,5.

4.4. Experimente pe rata de eșantionare: Scrieți un script MATLAB pentru a înregistra dvs. o rostire "numele meu este \*\*\*\*", cu o rată de eșantionare de 16 kHz și 8-biți rezoluție sau folositi un fisier existent. Încercați să reeșantionati semnalele audio prin scăderea ratelor de eșantionare la 11 kHz, 8 kHz, 4 kHz, 2 kHz, 1 kHz, și așa mai departe. La care rata de esantionare începem să avem dificultăți în a înțelege conținutul enunțului?

4.5. Experimente cu adăugarea de zgomot: Scrieți un script MATLAB pentru a înregistra rostirea "numele meu este \*\*\*\*", cu o rată de eșantionare de 8 kHz și 8-biți rezoluție. Putem adăuga zgomot la semnalele audio prin utilizarea secvenței următoare:

```
k = 0.1;
```

```
y2 = y + k*randn(length(y), 1); % Aduna zgomot
```

```
sound(y2, 8000); % Playback/redare
```

```
plot(y2);
```

Creșteti valoarea lui k cu 0,1 de fiecare dată și răspundeți la următoarele întrebări.

a) La ce valoare a lui K începem să avem dificultăți în a înțelege conținutul redat?

b) Se trasează formele de undă la valori diferite ale lui k. La ce valoare a lui k vom începe să avem dificultăți în a identifica perioada fundamentală ?

4.6 Studiați funcția detectSpeech și exersați pentru un exemplu la alegere.

<https://se.mathworks.com/help/audio/ref/detectspeech.html>

## 5. Bibliografie

[https://www.mathworks.com/help/matlab/import\\_export/record-and-play-audio.html](https://www.mathworks.com/help/matlab/import_export/record-and-play-audio.html)

<http://mirilab.org/jang/books/audioSignalProcessing/>

<http://mirilab.org/jang/matlab/toolbox/sap/>

<http://mirilab.org/jang/matlab/toolbox/utility/>

<http://neural.cs.nthu.edu.tw/jang/matlab/toolbox/sap/html/sap/wavReadInt.html>

<https://se.mathworks.com/help/audio/ref/detectspeech.html>