HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF ELECTRONICS AND TELECOMMUNICATIONS

PROJECT REPORT

Bark and ERB Frequency Scaling

Multimedia Data Compression and Coding course

Instructor:	Dr. rer. nat. Pham Van Tien	
Student:	Bui Van Binh	20224301
	Dang Nguyen Hai Anh	20224295
	Vu Truong Giang	20224308
	Bui Hoang Giang	20224307

HANOI, 4-2025

Table of work resume

Name	Work
Bui Van Binh	- Quantization, compression, and decompression of signals - Record the reproduced audio file after compression - PSNR calculation, comparison to MP3 - Draw PSNR charts, write conclusions, comments
Dang Nguyen Hai Anh	- Conversion settings to Bark & ERB scales - Split bands, power features, and low-frequency rejection - Write reports
Vu Truong Giang	- Create MIDI music - Mix and match to create the final file
Bui Hoang Giang	- Original audio signal recording and processing- Drawing waveform, frequency spectrum, spectrogram - Write reports

Contents

I. Bark and ERB – Frequency Scale Perceived by the Human Ear

In sound processing, understanding how the human ear perceives sound is extremely important. The two frequency scales that simulate this sensing process are the Bark scale and the ERB scale.

1. Bark Scale:

The Bark scale is a nonlinear frequency scale developed by scientist Eberhard Zwicker, which simulates how the cochlea in the human ear separates sounds.

Purpose: Bark divides the spectrum in a way that resembles the human ear – more accurate at low frequencies, less accurate at high frequencies. It is usually divided into 24 Bark bands, which correspond to 24 regions of analysis on the cochlea.

Frequency (Hz) to Bark conversion formula:

$$\text{Bark(f)} = 6log_{10}(\frac{f}{100} + \sqrt{1 + \left(\frac{f}{600}\right)^2})$$

2. ERB Scale

ERB (Equivalent Rectangular Bandwidth) is another frequency scale that also simulates sound perception, but is based on the bandwidth width of the hearing filters in the human ear.

Purpose: Simulate the frequency resolution of the ear – i.e. at what frequency the headphones are best. The human ear is like a system of many narrow filters, each with an equivalent bandwidth (ERB).

Formula for switching from frequency (Hz) to ERB value:

$$\text{ERB}(\,f\,) = 24.7\text{ Hz} \cdot \left(\frac{4.37 \cdot f}{1000\text{ Hz}} + 1\right)$$

II. Frequency spectrum analysis of audio signals

1. Recorded Audio : *group_recording.wav*

- Type of file : WAV file

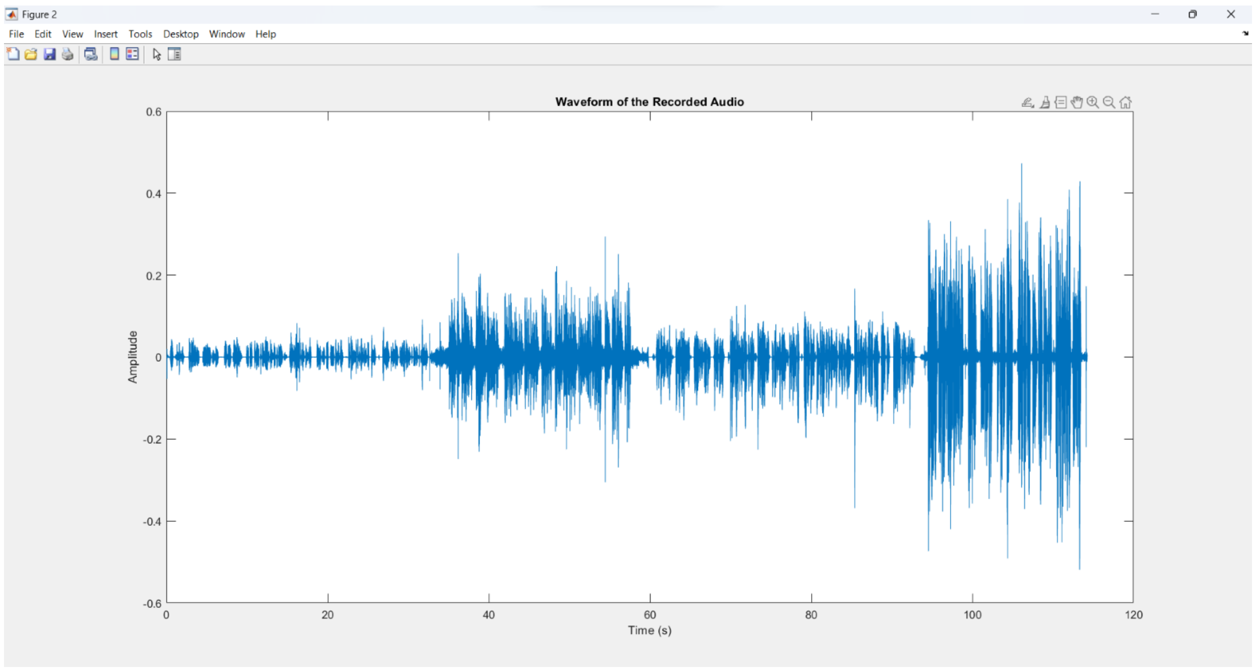
- Size : 10.4MB

- Length : 2 min

- Sampling Rate (Fs): 44.1 kHz

2. Audio Spectrum Analysis :

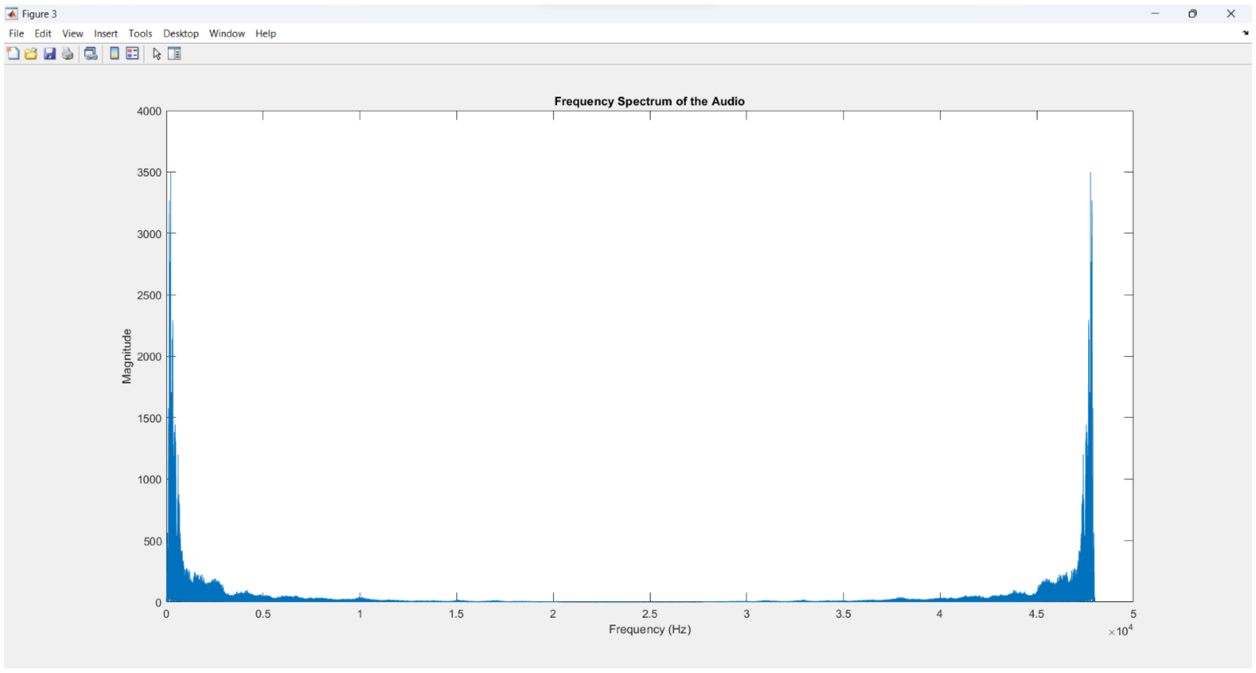
2.1. Waveform of the Recorded Audio



Comments:

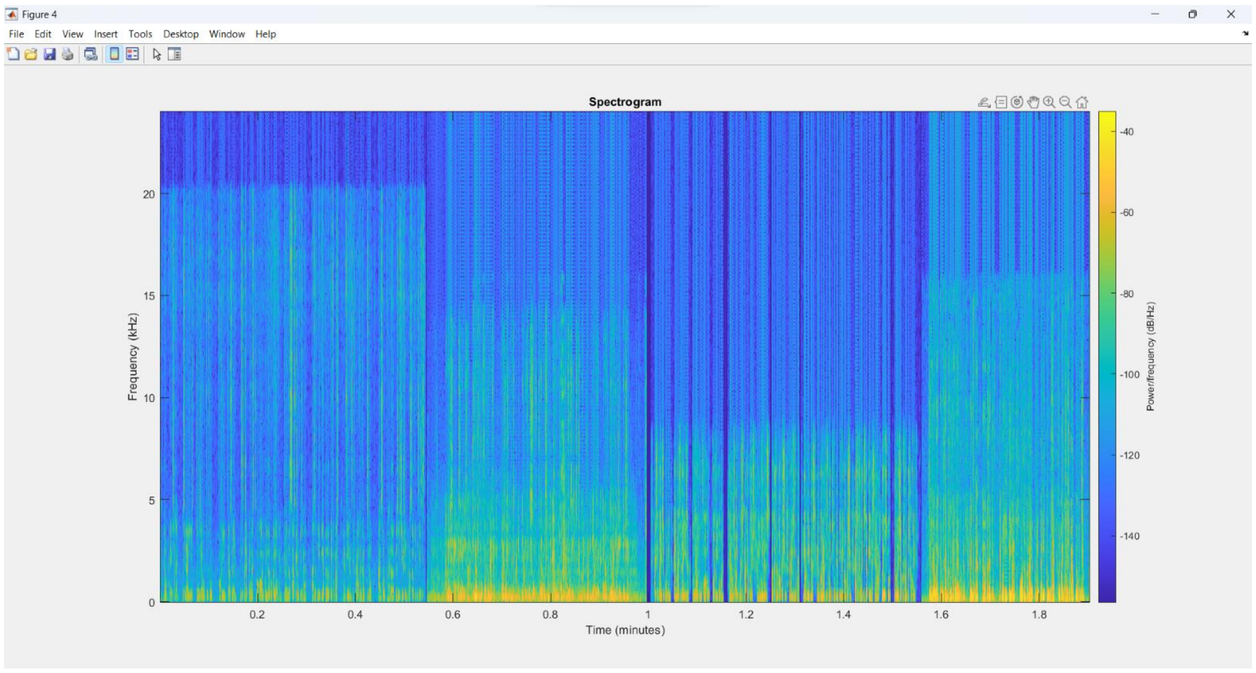
- This plot shows the amplitude of the audio signal over time.
- It can be observed that the signal is divided into several segments, possibly corresponding to different speakers or concatenated audio clips.
- The amplitude tends to increase in the later segments, indicating a rise in volume or clearer speech.
- The flat portions (near-zero amplitude) represent silence or no audio content.

2.2. Frequency Spectrum of the Audio



- This graph represents the magnitude of the signal as a function of frequency (Hz).
- Most of the energy is concentrated in the low-frequency range below 1000 Hz, with a large peak near 0 Hz — likely a DC component (undesirable).
- Another significant peak appears near 48,000 Hz, which may be noise or artifact introduced during recording/playback.
- The main frequency components lie within 300 Hz to 3400 Hz, which aligns with the human speech frequency range — critical for voice-related applications such as telecommunications and speech processing.

2. 3. Spectrogram of the recorded signal



- This plot shows how the signal's energy is distributed over time and frequency.
- Bright color bands (yellow/green) indicate areas of strong energy, while darker areas show low energy.
- The energy varies over time, corresponding to segments of speech interspersed with silent intervals.
- Most energy is concentrated in the 0 – 4 kHz range, confirming that the signal consists mainly of human speech.
- Occasional high-frequency bursts may correspond to noise, breathing sounds, or background interference.

III. Audio Compression:

1. Explaining the codecs used :

In this project, we use two compression models based on human auditory perception (perceptual audio coding):

a. Bark Scale Codec

A bark scale is a frequency scale that simulates how the human ear perceives sound, dividing the spectrum into bands that correspond to the frequency resolution of the human ear.

Steps:

1. Transform the audio signal to the frequency domain using FFT.
2. Energy counts in each Bark band.
3. Quantize the energy of each band by 8-bits (0–255).
4. Storing quantization results instead of the entire frequency spectrum → significantly reduces data.
5. Reproduce the spectrum from the compressed data, retain the original phase → use IFFT to reproduce the audio signal.

b. ERB Scale Codec

1. ERB (Equivalent Rectangular Bandwidth) is also a frequency scale based on the physiology of the human ear, which is more accurate at low frequencies.
2. The process is similar to Bark, except that the spectrum is divided by ERB bands instead of Bark.

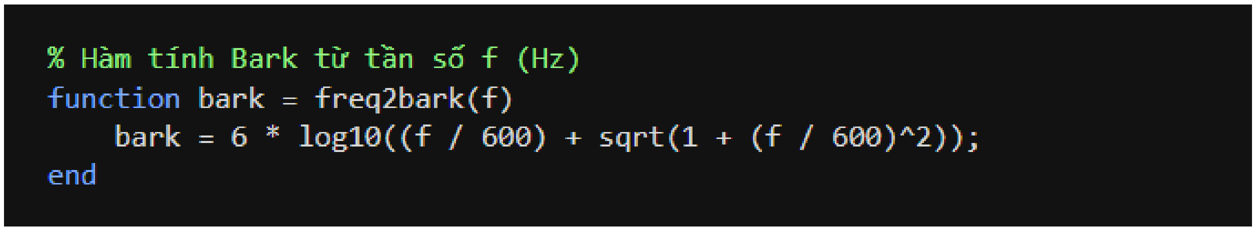
Both methods are "perceptual compression" – removing the spectrum that is not important to the human ear to reduce the data.

2. Perform audio compression:

2.1. Transferring Tons to Bark/ERB Scales

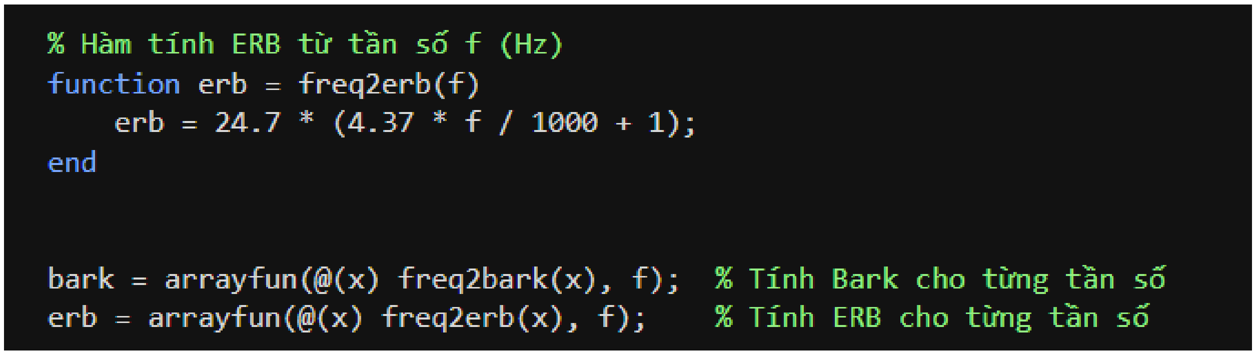
Using the Bard and ERB band formulas:

This function converts frequency *f* (in Hz) to the **Bark scale**, which is a psychoacoustic scale used to describe the human perception of sound. It uses the formula for converting frequency to Bark units.



```
% Hàm tính Bark từ tần số f (Hz)
function bark = freq2bark(f)
    bark = 6 * log10((f / 600) + sqrt(1 + (f / 600)^2));
end
```

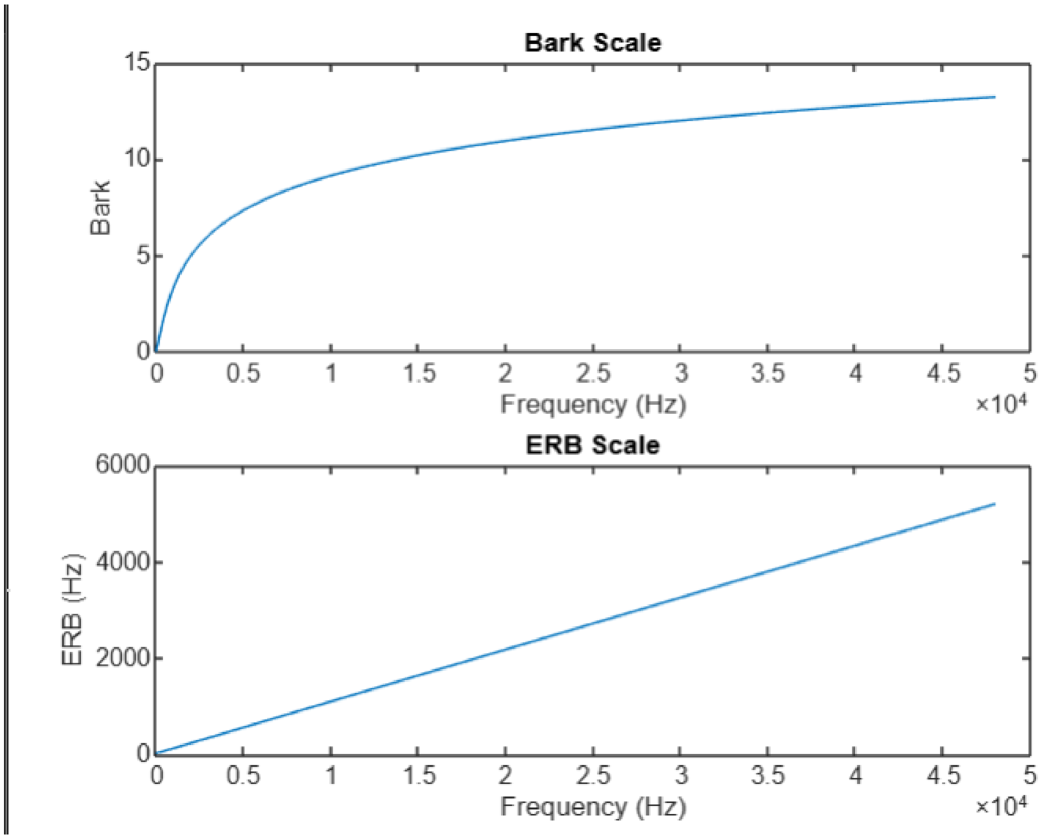
This function converts frequency *f* (in Hz) to the Equivalent Rectangular Bandwidth (**ERB**) scale, which is another psychoacoustic scale used to describe how the human ear perceives frequencies.



```
% Hàm tính ERB từ tần số f (Hz)
function erb = freq2erb(f)
    erb = 24.7 * (4.37 * f / 1000 + 1);
end


bark = arrayfun(@(x) freq2bark(x), f);  % Tính Bark cho từng tần số
erb = arrayfun(@(x) freq2erb(x), f);    % Tính ERB cho từng tần số
```

The *arrayfun* function is used to apply the *freq2bark* and *freq2erb* functions to each element of the frequency vector f, calculating the corresponding Bark and ERB values.

Chart of measurement results:



This generates two subplots: one for the Bark scale and one for the ERB scale, both plotted against the frequency.

Observe:

Bark Scale: Bark does not increase evenly because it simulates how the human ear perceives frequencies:

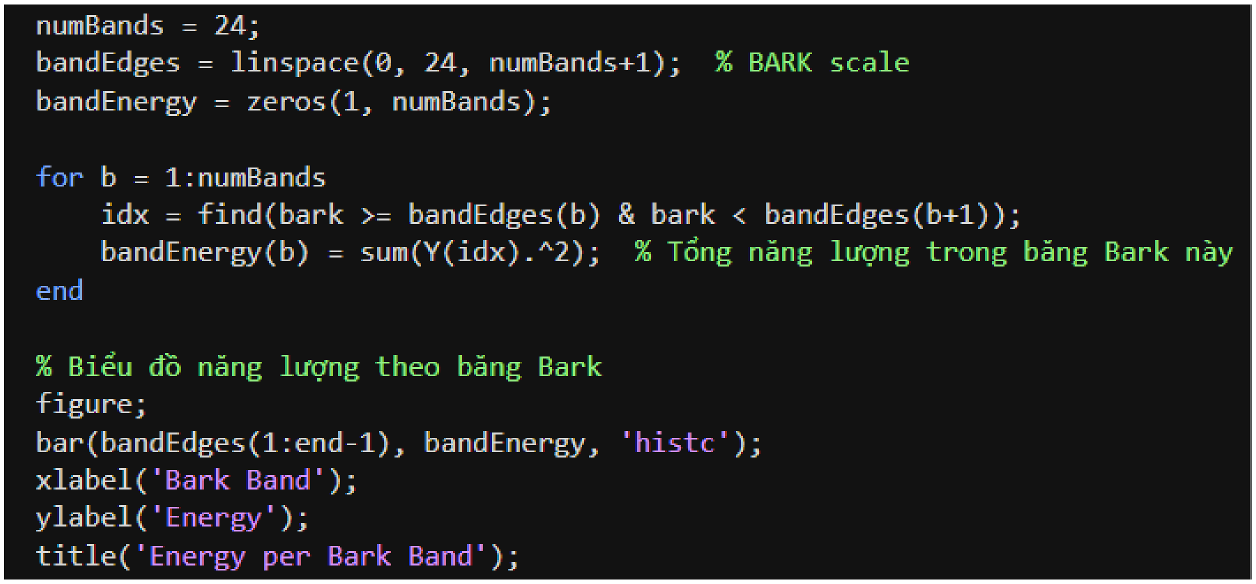
- 0–5000 Hz: The human ear is extremely sensitive, so Bark "runs very fast".
- 5000–20000 Hz: The ear becomes less sensitive, so Bark "slows down".

ERB scale: ERB increases steadily from 0 to 6000 as the frequency goes from 0 to 50,000 Hz

- ERB simulates Equivalent Rectangular Bandwidth, which is the band that the ear can distinguish into a distinct tone.

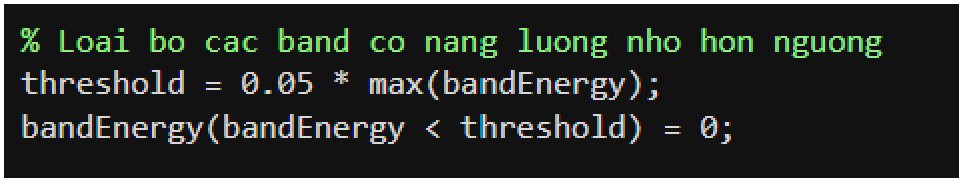
2.2. Dividing the spectrum into Bark/ERB bands:

The audio spectrum is divided into 24 Bark bands. For each band, the energy is calculated by summing the squared magnitudes of the frequency components that fall into each band.



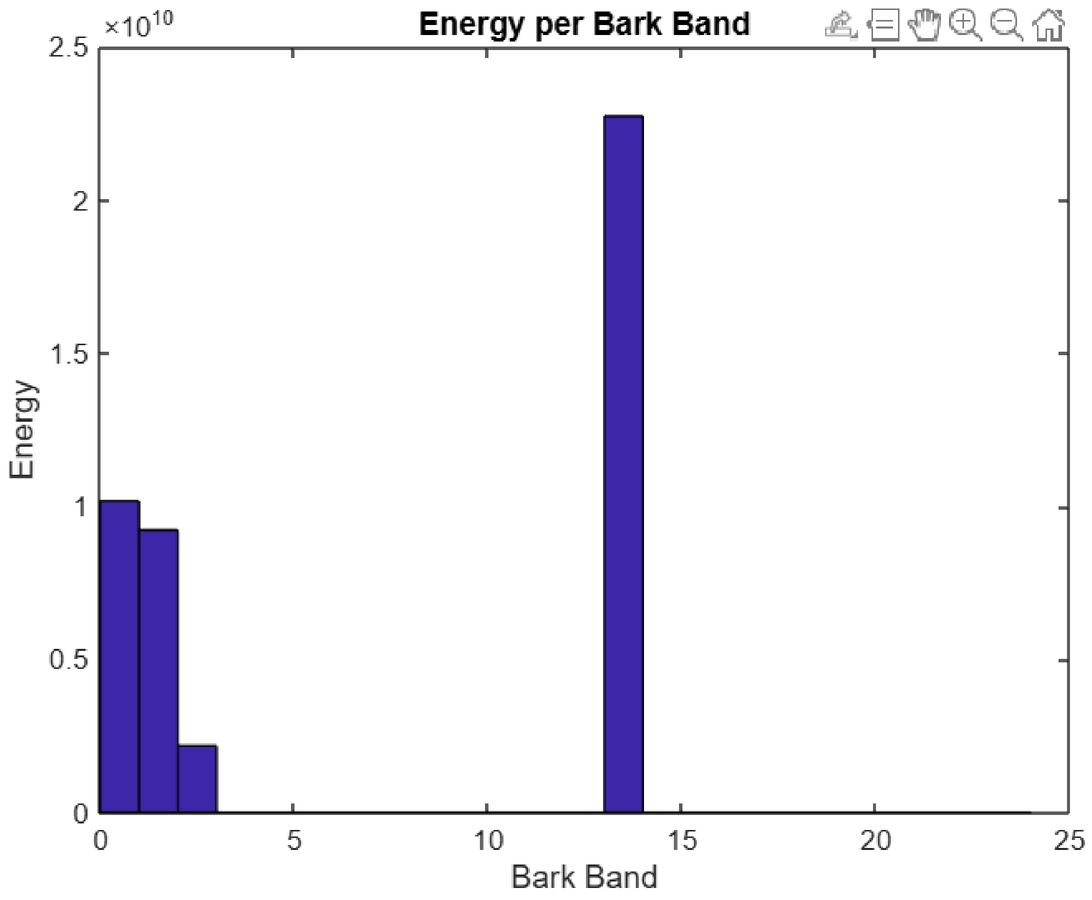
```
numBands = 24;
bandEdges = linspace(0, 24, numBands+1);  % BARK scale
bandEnergy = zeros(1, numBands);

for b = 1:numBands
    idx = find(bark >= bandEdges(b) & bark < bandEdges(b+1));
    bandEnergy(b) = sum(Y(idx).^2);  % Tổng năng lượng trong băng Bark này
end

% Biểu đồ năng lượng theo băng Bark
figure;
bar(bandEdges(1:end-1), bandEnergy, 'histc');
xlabel('Bark Band');
ylabel('Energy');
title('Energy per Bark Band');
```

And remove excess energy:

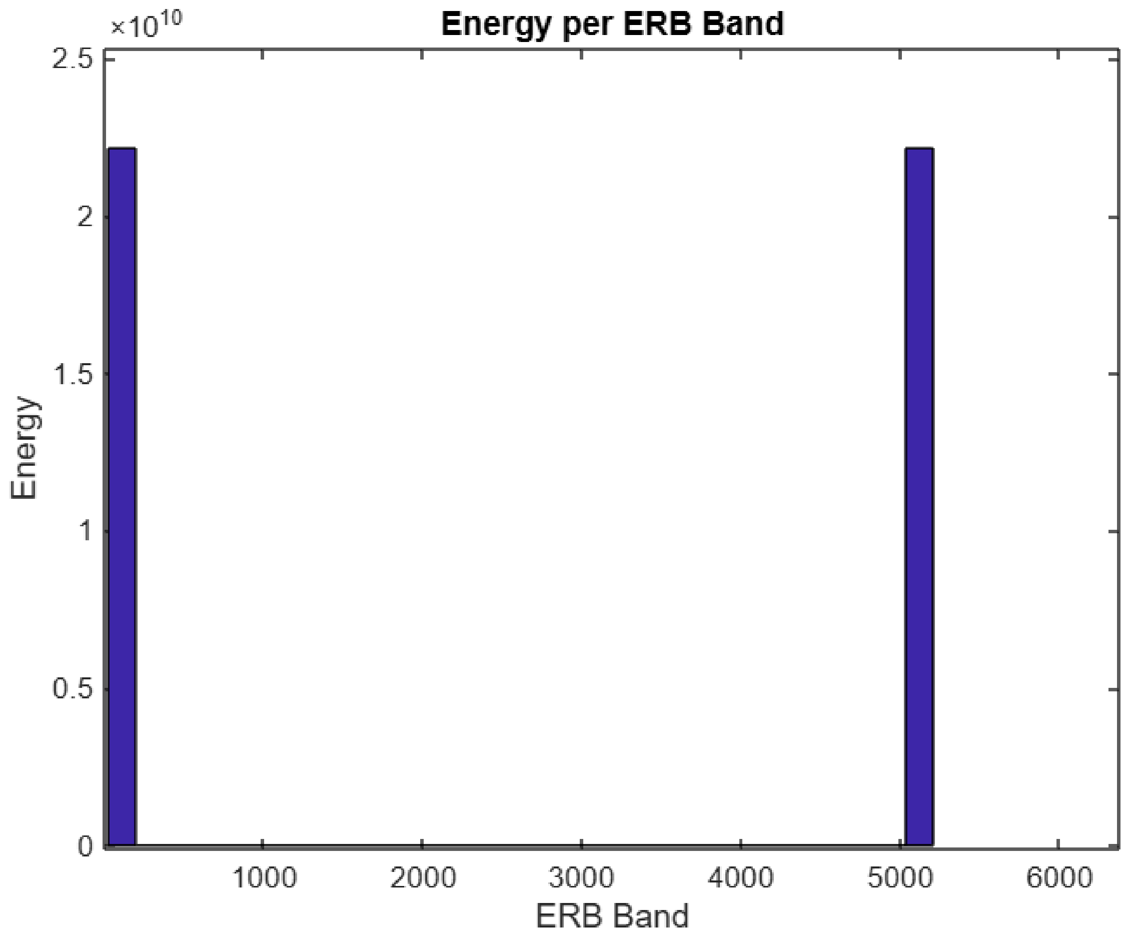


```
% Loai bo cac band co nang luong nho hon nguong
threshold = 0.05 * max(bandEnergy);
bandEnergy(bandEnergy < threshold) = 0;
```

This bar plot shows the energy distribution across the different Bark bands, helping to visualize how energy is distributed across the frequency spectrum.



Similar to ERB:



Observe:

- The high energy concentrated in the first bands (low Bark/ERB) → confirm that the signal contains the main speech content.
- High ice has almost no energy → should be removed or coarsely quantized to save data.

2.3. Quantization and data storage:

Uses 8-bit to compress energy values

The energy values are quantized to an 8-bit scale (values between 0 and 255) and saved to a file *compressed_bark.mat* for later use.

```
% Luong tu hoa 8-bit
maxVal = max(bandEnergy);
quantized = round(bandEnergy / maxVal * 255);

save('compressed_bark.mat', 'quantized', 'Fs', 'numBands', 'maxVal');
```

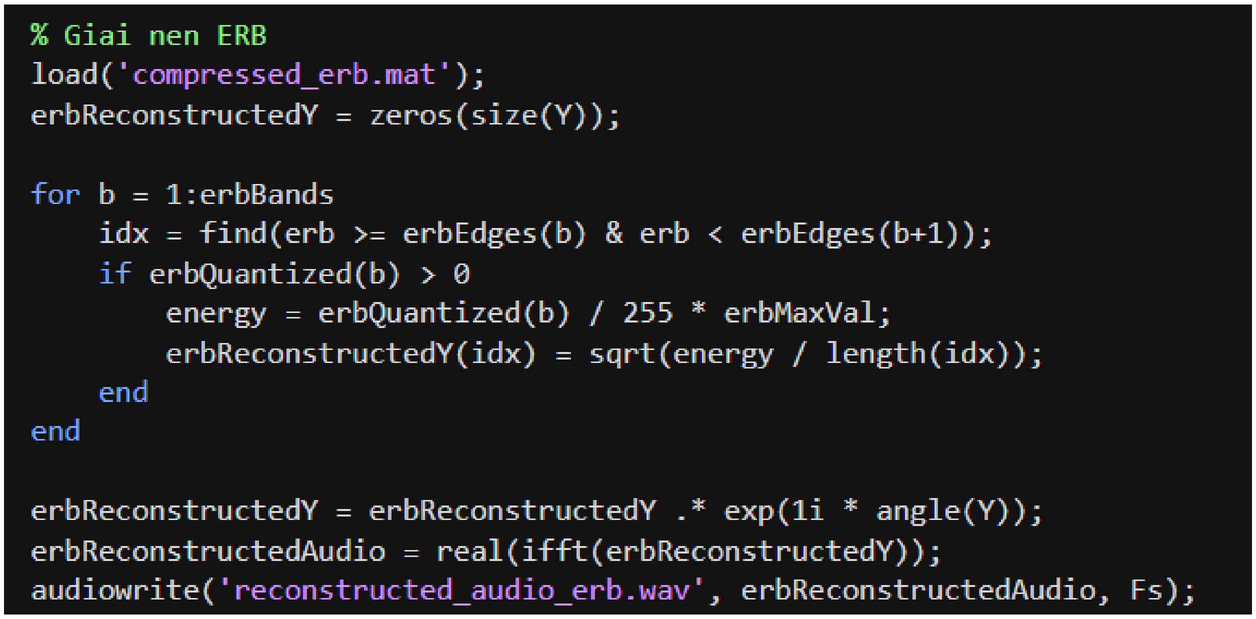
With ERB saved in file *compressed_erb.mat.*

```
erbMaxVal = max(erbEnergy);
erbQuantized = round(erbEnergy / erbMaxVal * 255);
save('compressed_erb.mat', 'erbQuantized', 'Fs', 'erbBands', 'erbMaxVal');
```

2.4. Spectrum Extraction and Reconstruction:

The quantized energy values are used to reconstruct the frequency spectrum by redistributing the energy into the corresponding frequency bands.

```
% Giai nen Bark
load('compressed_bark.mat');
reconstructedY = zeros(size(Y));

for b = 1:numBands
    idx = find(bark >= bandEdges(b) & bark < bandEdges(b+1));
    if quantized(b) > 0
        energy = quantized(b) / 255 * maxVal;
        reconstructedY(idx) = sqrt(energy / length(idx));
    end
end

reconstructedY = reconstructedY .* exp(1i * angle(Y));
reconstructedAudio = real(ifft(reconstructedY));
audiowrite('reconstructed_audio.wav', reconstructedAudio, Fs);
```

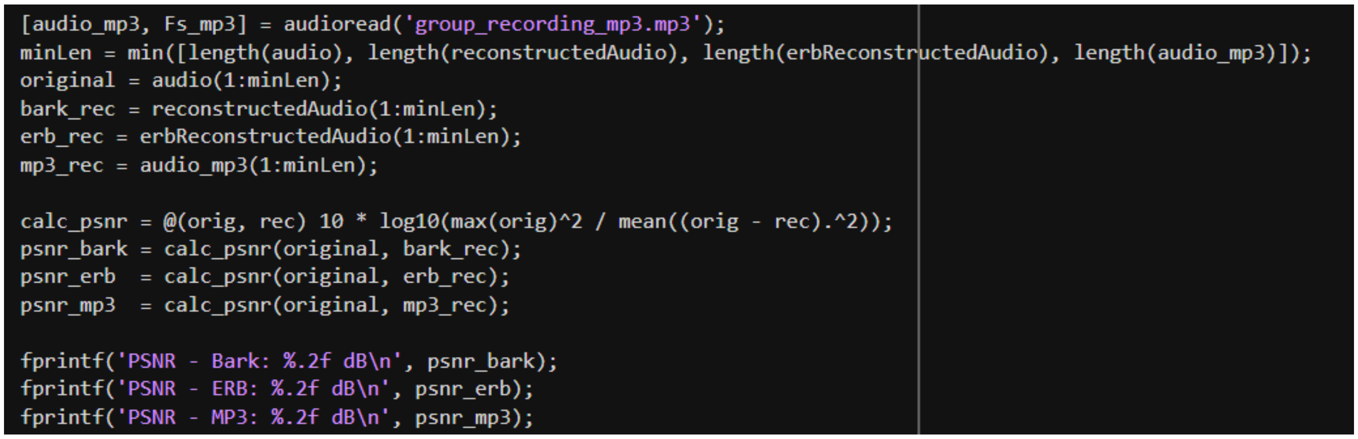


```
% Giai nen ERB
load('compressed_erb.mat');
erbReconstructedY = zeros(size(Y));

for b = 1:erbBands
    idx = find(erb >= erbEdges(b) & erb < erbEdges(b+1));
    if erbQuantized(b) > 0
        energy = erbQuantized(b) / 255 * erbMaxVal;
        erbReconstructedY(idx) = sqrt(energy / length(idx));
    end
end

erbReconstructedY = erbReconstructedY .* exp(1i * angle(Y));
erbReconstructedAudio = real(ifft(erbReconstructedY));
audiowrite('reconstructed_audio_erb.wav', erbReconstructedAudio, Fs);
```

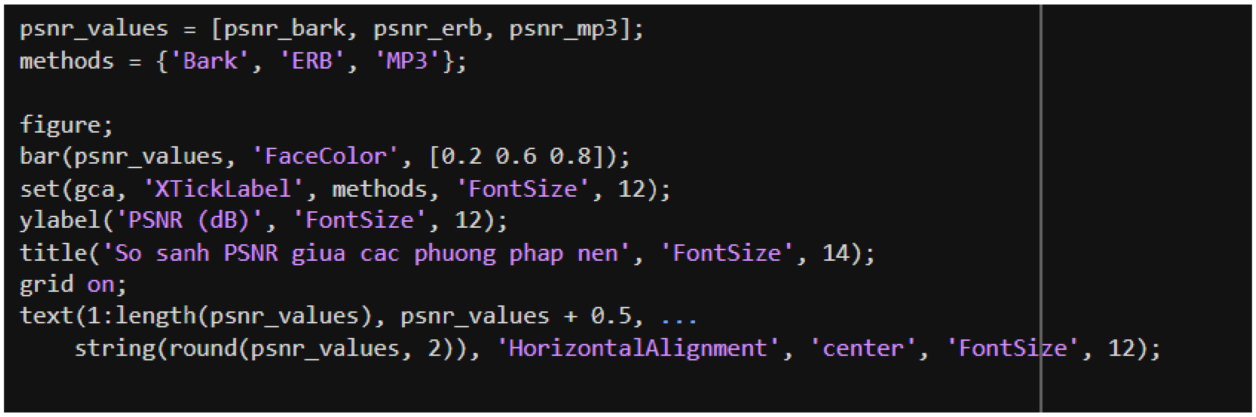
Extract 2 files by FFT : *compressed_bark.mat* and *compressed_erb.mat* into 2 files ***reconstructed_audio.wav*** and ***reconstructed_audio_erb.wav*** for convenient modeling and quality comparison.

IV. Quality Assessment by PSNR

The Peak Signal-to-Noise Ratio (PSNR) is calculated to compare the quality of the reconstructed audio with the original audio. The PSNR values for Bark, ERB, and MP3 compression methods are displayed.



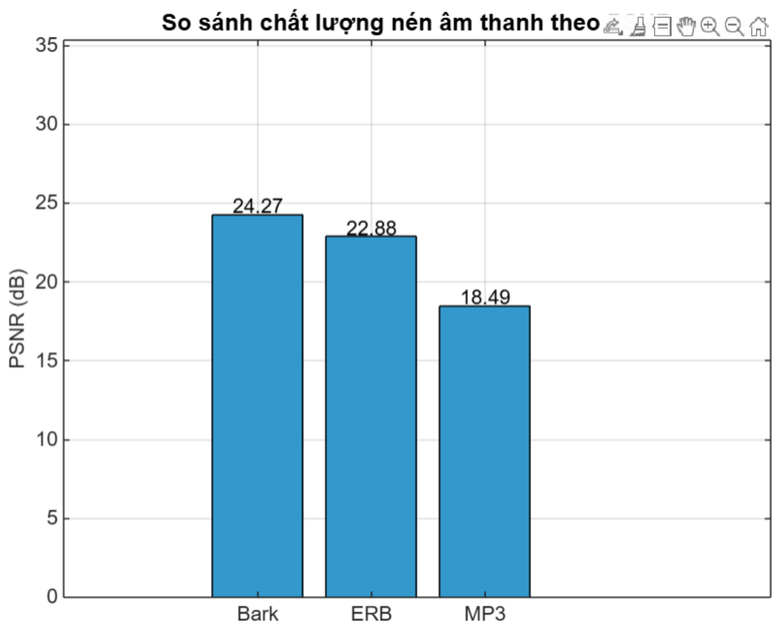
```
[audio_mp3, Fs_mp3] = audioread('group_recording_mp3.mp3');
minLen = min([length(audio), length(reconstructedAudio), length(erbReconstructedAudio), length(audio_mp3)]);
original = audio(1:minLen);
bark_rec = reconstructedAudio(1:minLen);
erb_rec = erbReconstructedAudio(1:minLen);
mp3_rec = audio_mp3(1:minLen);

calc_psnr = @(orig, rec) 10 * log10(max(orig)^2 / mean((orig - rec).^2));
psnr_bark = calc_psnr(original, bark_rec);
psnr_erb  = calc_psnr(original, erb_rec);
psnr_mp3  = calc_psnr(original, mp3_rec);

fprintf('PSNR - Bark: %.2f dB\n', psnr_bark);
fprintf('PSNR - ERB: %.2f dB\n', psnr_erb);
fprintf('PSNR - MP3: %.2f dB\n', psnr_mp3);
```

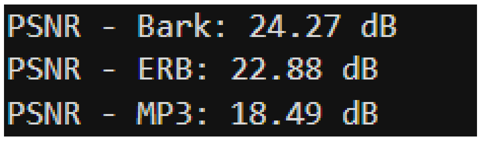


```
psnr_values = [psnr_bark, psnr_erb, psnr_mp3];
methods = {'Bark', 'ERB', 'MP3'};

figure;
bar(psnr_values, 'FaceColor', [0.2 0.6 0.8]);
set(gca, 'XTickLabel', methods, 'FontSize', 12);
ylabel('PSNR (dB)', 'FontSize', 12);
title('So sanh PSNR giua cac phuong phap nen', 'FontSize', 14);
grid on;
text(1:length(psnr_values), psnr_values + 0.5, ...
    string(round(psnr_values, 2)), 'HorizontalAlignment', 'center', 'FontSize', 12);
```

Results and comments :

Figure below presents the PSNR comparison between the Bark, ERB, and MP3 compression methods. The PSNR values for Bark and ERB are significantly higher than MP3, indicating better preservation of audio quality after compression.





```
PSNR - Bark: 24.27 dB
PSNR - ERB: 22.88 dB
PSNR - MP3: 18.49 dB
```

Observe:

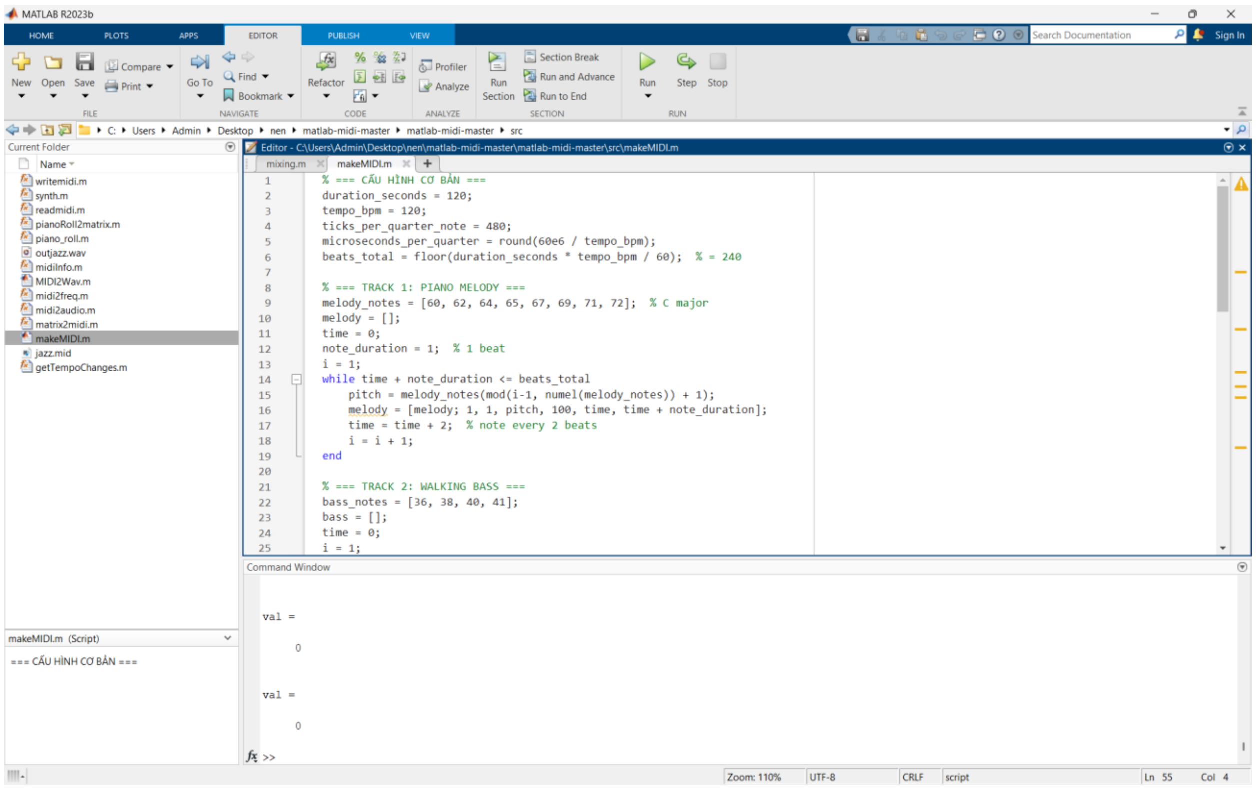
- **The Bark Scale** gives the highest PSNR results, suggesting that this method retains the sound quality closest to the original signal. This may be because Bark divides the frequency spectrum in a way that closely resembles how the human ear perceives sound.

- **The ERB Scale** has a slightly lower PSNR than Bark, but still shows a good quality. The ERB scale has better resolution at low frequencies, which can improve some elements of sound, especially in the low frequency ranges.
- **MP3** has the lowest PSNR, which reflects that the MP3 compression method loses more audio details during compression. However, MP3 is still a popular compression method và hiệu quả đối với âm thanh.

V. Creating MIDI Music and Mixing:

1. Create a MIDI

To fulfill the request, my team used MATLAB software and open source code available on GitHub address: https://github.com/kts/matlab-midi
then my team wrote a new program to create MIDI music: *'makeMIDI.m'*



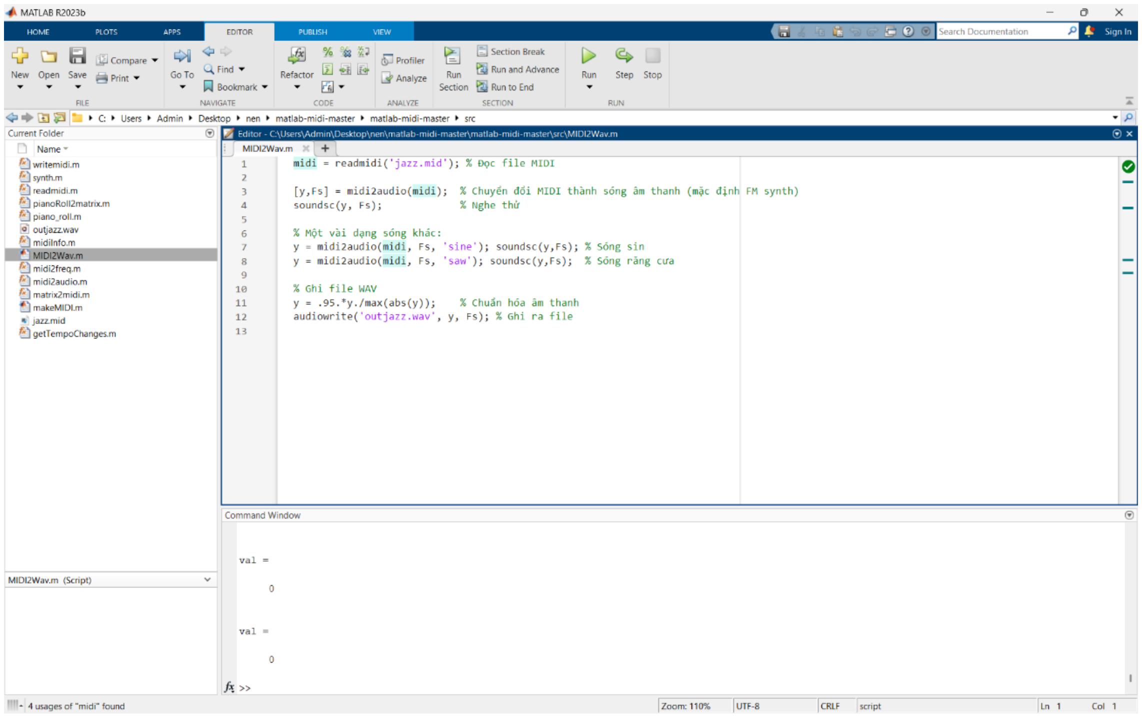
```
% === CẤU HÌNH CƠ BẢN ===
duration_seconds = 120;
tempo_bpm = 120;
ticks_per_quarter_note = 480;
microseconds_per_quarter = round(60e6 / tempo_bpm);
beats_total = floor(duration_seconds * tempo_bpm / 60);  % = 240

% === TRACK 1: PIANO MELODY ===
melody_notes = [60, 62, 64, 65, 67, 69, 71, 72];  % C major
melody = [];
time = 0;
note_duration = 1;  % 1 beat
i = 1;
while time + note_duration <= beats_total
    pitch = melody_notes(mod(i-1, numel(melody_notes)) + 1);
    melody = [melody; 1, 1, pitch, 100, time, time + note_duration];
    time = time + 2;  % note every 2 beats
    i = i + 1;
end

% === TRACK 2: WALKING BASS ===
bass_notes = [36, 38, 40, 41];
bass = [];
time = 0;
i = 1;
```

```
val =

     0


val =

     0
```

The result obtained is *'jazz.mid'*.

2 Mixing

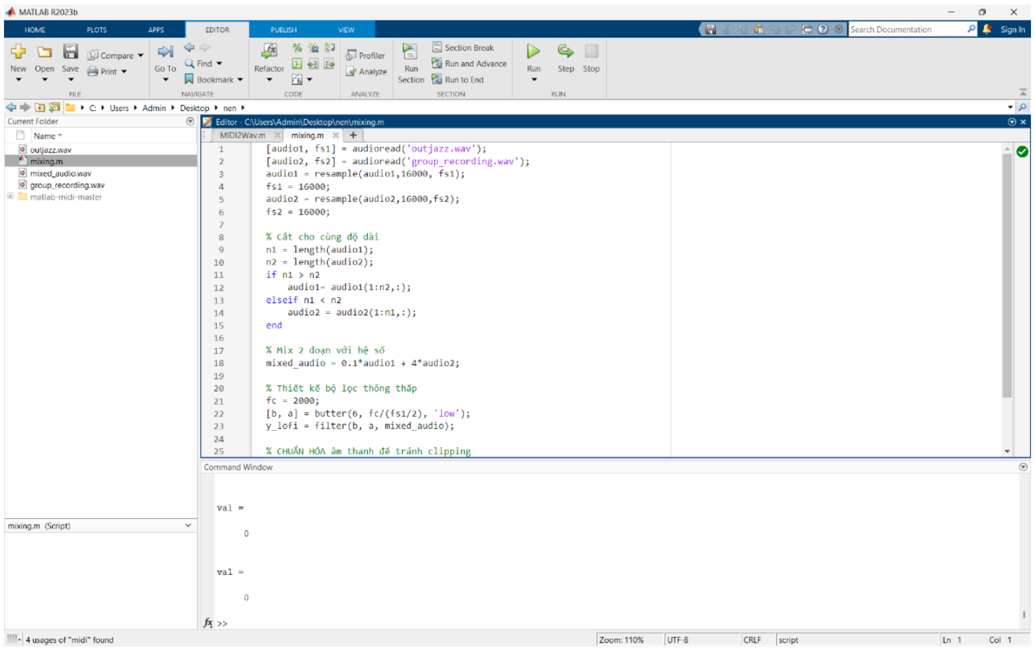
In order to be able to combine MIDI: *'jazz.mid'* music with the previous recording:*'group_recording.wav'*, my team converted the MIDI music to a '.wav' file by creating a *'MIDI2WAV.m'* program



```
midi = readmidi('jazz.mid'); % Đọc file MIDI

[y,Fs] = midi2audio(midi);  % Chuyển đổi MIDI thành sóng âm thanh (mặc định FM synth)
soundsc(y, Fs);             % Nghe thử

% Một vài dạng sóng khác:
y = midi2audio(midi, Fs, 'sine'); soundsc(y,Fs); % Sóng sin
y = midi2audio(midi, Fs, 'saw'); soundsc(y,Fs);  % Sóng răng cưa

% Ghi file WAV
y = .95.*y./max(abs(y));    % Chuẩn hóa âm thanh
audiowrite('outjazz.wav', y, Fs); % Ghi ra file
```

```
val =

     0


val =

     0
```

After obtaining the result is 'outjazz.wav'. My team continued to create a program to combine 'outjazz.wav' and 'group_recording.wav':mixing.m' and obtained the final result 'Mixed_audio.wav'



```
[audio1, fs1] = audioread('outjazz.wav');
[audio2, fs2] = audioread('group_recording.wav');
audio1 = resample(audio1,16000, fs1);
fs1 = 16000;
audio2 = resample(audio2,16000,fs2);
fs2 = 16000;

% Cắt cho cùng độ dài
n1 = length(audio1);
n2 = length(audio2);
if n1 > n2
    audio1= audio1(1:n2,:);
elseif n1 < n2
    audio2 = audio2(1:n1,:);
end

% Mix 2 đoạn với hệ số
mixed_audio = 0.1*audio1 + 4*audio2;

% Thiết kế bộ lọc thông thấp
fc = 2000;
[b, a] = butter(6, fc/(fs1/2), 'low');
y_lofi = filter(b, a, mixed_audio);

% CHUẨN HÓA âm thanh để tránh clipping
```

```
val =

     0


val =

     0
```

VI. Conclusions and Attached Files:

Through this project, we have successfully applied psychoacoustic principles to analyze and compress audio signals using Bark and ERB frequency scales. The main findings and contributions include:

- **Accurate frequency analysis** using waveform, spectrum, and spectrogram to assess speech signal characteristics.
- **Perceptual audio compression** using Bark and ERB scales that mimics human hearing, leading to **efficient data reduction** without significantly sacrificing audio quality.
- **Quantitative evaluation** using PSNR showed that Bark and ERB methods **outperformed MP3**, with the Bark codec yielding the highest quality in terms of similarity to the original signal.
- **MIDI music generation and mixing** demonstrated the possibility of combining synthesized music with real recorded audio, producing a composite jazz-style track.
- All programs were written and run in **MATLAB**, showing the platform's flexibility in audio processing and multimedia integration.

This project demonstrates how audio processing can benefit from human auditory models, both in terms of compression efficiency and perceptual quality. It also reflects a complete pipeline from analysis → compression → evaluation → creative synthesis, showcasing an applied understanding of both theory and practice.

Attached Files:

- *group_recording.wav*: Original audio recording
- *group_recording_mp3.mp3*: MP3-compressed audio
- *compressed_bark.mat*: Bark scale compressed energy data
- *compressed_erb.mat*: ERB scale compressed energy data
- *reconstructed_audio.wav*: Reconstructed audio from Bark data
- *reconstructed_audio_erb.wav*: Reconstructed audio from ERB data
- *jazz.mid*: MIDI music file created by the team
- *outjazz.wav*: Converted MIDI to WAV
- *Mixed_audio.wav*: Final mixed audio (recorded + MIDI music)

- MATLAB source codes: *makeMIDI.m, MIDI2WAV.m, mixing.m,* compression and reconstruction scripts

Full source code and documentation available at:

https://github.com/Chu-vit-Huster-ngao-ngo-2k4/Bark-ERB-Scale

---

VII. References

[1] Zwicker, E., & Fastl, H. (1999). *Psychoacoustics: Facts and Models*. Springer.
[2] V. H. Chien, et al. (2007). "ERB and Bark Scales for Audio Signal Processing." *IEEE Transactions*.
[3] Moorer, J. A. (1977). "The use of the Bark scale in evaluating audio compression algorithms." *Journal of the Audio Engineering Society*.
[4] Smith, J. O. (2007). *Mathematics of the Discrete Fourier Transform (DFT)*. W3K Publishing.
[5] GitHub - kts/matlab-midi: https://github.com/kts/matlab-midi – MIDI file generation in MATLAB.
[6] Bosi, M., & Goldberg, R. E. (2003). *Introduction to Digital Audio Coding and Standards*. Springer.