קוד מאט לאב הסבר מפורט:

**📁 שלב 1: בחירת קובץ אודיו**

[fileName, filePath] = uigetfile({'\*.mp3;\*.wav', 'Audio Files (\*.mp3, \*.wav)'}, 'SELECT קובץ אודיו');

* פותח **חלון בחירה של קובץ** (GUI).
* המשתמש יכול לבחור קובץ עם סיומת .mp3 או .wav.
* הפלט הוא:
  + fileName – שם הקובץ בלבד (למשל: 'song.mp3')
  + filePath – הנתיב שבו נמצא הקובץ (למשל: 'C:\Users\...')

if isequal(fileName, 0)

disp('לא נבחר קובץ.');

return;

end

* בודק אם המשתמש לחץ על "Cancel" ולא בחר קובץ.
* אם כן:
  + מציג הודעה: **"לא נבחר קובץ."**
  + מפסיק את הריצה של הסקריפט בעזרת return.

**🎵 שלב 2: טעינת קובץ האודיו**

[audioData, sampleRate] = audioread(fullfile(filePath, fileName));

* קורא את קובץ האודיו באמצעות audioread.
* fullfile(filePath, fileName) מאחד את הנתיב והשם – יוצר מחרוזת מלאה של הנתיב לקובץ.
* הפלטים:
  + audioData – מטריצה עם הערכים של האודיו (אם סטריאו, יהיו 2 עמודות).
  + sampleRate – קצב הדגימה (מספר הדגימות לשנייה, למשל 44100 Hz).

**🎚️ שלב 3: טיפול בסטריאו**

if size(audioData, 2) > 1

audioData = audioData(:, 1); % בחר את הערוץ הראשון (שמאלי)

end

* אם audioData הוא **סטריאו** (יותר מעמודה אחת) – בוחרים רק את **הערוץ השמאלי** (העמודה הראשונה).
* כך ממשיכים עם סיגנל **חד-ערוצי (mono)** לעיבוד פשוט יותר.

**🔁 שלב 4: המרת MP3 ל-WAV**

if strcmpi(fileName(end-2:end), 'mp3')

* בודק אם 3 האותיות האחרונות של שם הקובץ הן 'mp3' (לא תלוי רישיות – בגלל strcmpi).
* כלומר: האם הקובץ הוא בפורמט MP3.

outputFileName = fullfile(filePath, 'audio.wav');

audiowrite(outputFileName, audioData, sampleRate);

disp('המרה ל-WAV בוצעה.');

* אם כן, יוצר קובץ בשם 'audio.wav' באותו נתיב.
* משתמש ב־audiowrite כדי לכתוב את הנתונים שנטענו קודם לקובץ WAV.
* מציג הודעה: **"המרה ל-WAV בוצעה."**

else

disp('הקובץ כבר בפורמט WAV.');

end

* אם הקובץ כבר בפורמט WAV, מודפסת הודעה מתאימה.

**✨ סיכום:**

הקטע הזה:

1. פותח חלון לבחירת קובץ MP3 או WAV.
2. טוען את הקובץ.
3. ממיר אותו לחד-ערוצי אם צריך.
4. ואם הקובץ בפורמט MP3 – ממיר אותו ל-WAV.

יופי, עכשיו אנחנו נכנסים לניתוח הספקטרלי של האודיו. הנה הסבר שורה-שורה למה שקורה כאן:

**🎛️ שלב 3: חישוב ספקטרוגרמה (Spectrogram)**

window = 1024; % גודל החלון

* מגדיר את גודל החלון (מספר דגימות בכל קטע של האודיו לניתוח).
* ככל שהחלון קטן יותר – מקבלים **רזולוציה טובה בזמן** אבל פחות מדויקת בתדרים, ולהפך.
* כאן: כל חלון מכיל 1024 דגימות מהאודיו.

overlap = 512; % חפיפות

* מספר הדגימות **שחופפות** בין חלון לחלון.
* חפיפה של 50% (כלומר: 1024 חלון, 512 חפיפה) היא ערך מקובל – עוזרת ברציפות ובדיוק.

nfft = 2048; % מספר חישובי ה-FFT

* מספר נקודות לחישוב FFT (Fast Fourier Transform).
* בדרך כלל שווה או גדול מגודל החלון.
* משפיע על רזולוציית התדרים: ככל ש־nfft גדול יותר → מקבלים תדרים יותר "צפופים".

[S, F, T] = spectrogram(audioData, window, overlap, nfft, sampleRate);

זוהי השורה המרכזית:  
מחשבת את הספקטרוגרמה של audioData.

**מה כל פרמטר עושה:**

* audioData – האות האודיו (חד-ערוצי כפי שהכנת קודם).
* window – גודל כל חלון ניתוח.
* overlap – כמה דגימות חופפות בין חלון לחלון.
* nfft – כמה נקודות בתדר לחשב (גודל ה-FFT).
* sampleRate – קצב הדגימה (כדי לדעת מהו תדר אמיתי לכל bin).

**מה מוחזר:**

* S – מטריצה קומפלקסית של ספקטרוגרמה:
  + שורות: תדרים
  + עמודות: חלונות בזמן
  + כל ערך מייצג את **העוצמה** של תדר מסוים בזמן מסוים
* F – וקטור תדרים (Hz) – כלומר, מה הערך של כל שורה ב־S.
* T – וקטור זמנים (שניות) – כלומר, מה הזמן של כל עמודה ב־S.

**🔍 מה זה ספקטרוגרמה בעצם?**

זוהי דרך להסתכל על תדרים לאורך זמן:

* ציר X: זמן
* ציר Y: תדר
* צבע או ערך: עוצמת התדר באותו רגע

זה אחד הכלים הכי חשובים בניתוח אותות קוליים או מוזיקליים.

יצירת סינטזה לקובץ:  
**חישוב התדרים הדומיננטיים:**

[~, maxIndex] = max(abs(S)); % מציאת המגניטודה המקסימלית

בשלב הזה אתה מוצא את המגניטודה המקסימלית של הספקטרוגרמה S. המגניטודה היא הערך האבסולוטי של כל עמודה, כלומר ערך האורכי של התדרים ברגע נתון. הפונקציה max מחפשת את המגניטודה הגבוהה ביותר בכל עמודה. ערך maxIndex מחזיר את המיקום של התדר הדומיננטי ביותר בכל רגע זמן.

dominantFrequencies = F(maxIndex); % התדרים הדומיננטיים

כאן אתה משתמש ב-maxIndex כדי למצוא את התדרים הדומיננטיים לפי המיקומים שהתקבלו בשלב הקודם במטריצה F (ששומרת את התדרים). כך אתה מקבל את התדר הדומיננטי עבור כל עמודה בספקטרוגרמה.

**אתחול האות הסינתטי:**

synthesizedAudio = zeros(size(audioData)); % אתחול האות

כאן אתה יוצר וקטור ריק באותו הגודל של audioData, כלומר, יצירת מקום לאות הסינתטי החדש שיתבצע ממנו חישוב בהמשך.

dt = 1 / sampleRate; % מרווח דגימה

dt הוא מרווח הדגימה, כלומר הזמן שעובר בין כל דגימה ודגימה בקובץ האודיו שלך. הפונקציה sampleRate מחזירה את קצב הדגימה (למשל, 44100 דגימות לשנייה), ולכן חישוב dt הוא הפוך לכך.

**יצירת הגל הסינוסי לכל מקטע:**

for k = 1:length(T)-1

% זמן למקטע

t = T(k):dt:T(k+1)-dt;

בשלב הזה אתה מתחיל ללולאה על כל הזמן (במקטעים של T). כל מקטע בזמן יתחלק לגבולות T(k) ו-T(k+1), ויוצר את הזמן של המקטע t.

% יצירת גל סינוסי בתדר הדומיננטי

freq = dominantFrequencies(k); % התדר הדומיננטי במקטע

synthesizedAudio(round(T(k)\*sampleRate):round(T(k+1)\*sampleRate)-1) = ...

sin(2 \* pi \* freq \* t);

end

כאן אתה יוצר גל סינוסי בתדר הדומיננטי עבור כל מקטע. הפונקציה sin(2 \* pi \* freq \* t) יוצרת גל סינוסי בתדר freq לכל ערך זמן במקטע. אתה שומר את הערכים האלה במערך synthesizedAudio על פי המקטעים הרלוונטיים.

**הצגת הגל הסינתטי עם תדרים דומיננטיים:**

figure;

פה אתה פותח חלון גרפי חדש כדי להציג את הגרפים שיבואו אחריו.

כמו כן, בשלב זה יש לך את הגל הסינתטי שנוצר ואת התדרים הדומיננטיים, תוכל להציג אותם בגרפים כמו שמופיע בהמשך הקוד.

**מציאת התווים:**בוא נבין כל שורה בקוד:

**1. יצירת מילון של תדרים ותווים:**

note\_frequencies = [261.63, 277.18, 293.66, 311.13, 329.63, 349.23, 369.99, 392.00, 415.30, 440.00, ...

466.16, 493.88, 523.25, 554.37, 587.33, 622.25, 659.25, 698.46, 739.99, ...

783.99, 830.61, 880.00, 932.33, 987.77, 1046.50, 1108.73, 1174.66, ...

1244.51, 1318.51];

במשתנה note\_frequencies אתה שומר את **תדרי המוזיקה** שמייצגים את התווים השונים במוזיקה המערבית. למשל, התדר של התו C הוא 261.63 Hz, C# הוא 277.18 Hz, וכן הלאה.

notes\_names = {'C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B', 'C5', 'C#5', 'D5', ...

'D#5', 'E5', 'F5', 'F#5', 'G5', 'G#5', 'A5', 'A#5', 'B5', 'C6', 'C#6', 'D6', ...

'D#6', 'E6'};

כאן אתה יוצר **מערך של שמות התווים** המתאימים לתדרים שמופיעים במערך note\_frequencies. כל שם תו תואם לתדר בסדר מסוים, לדוגמה, התו C5 יהיה תואם לתדר 523.25 Hz.

note\_dict = containers.Map(note\_frequencies, notes\_names);

אתה יוצר **מילון** (HashMap) בין התדרים (note\_frequencies) לבין השמות שלהם (notes\_names). המילון הזה יאפשר לך לחפש את השם של התו בהתבסס על תדר.

**2. הגדרת רף למגניטודה:**

magnitude\_threshold = 0.01; % רף מינימלי לעוצמת האות

המשתנה magnitude\_threshold מגדיר את **הרף המינימלי** לעוצמת המגניטודה של התדר. כלומר, אם המגניטודה בתדר כלשהו נמוכה יותר מ-0.01, לא תתבצע זיהוי של תו עבור אותו תדר. זה עוזר לסנן רעשים או תדרים חלשים שלא שייכים לצלילים המרכזיים של המוזיקה.

**3. זיהוי תווים בהתבסס על תדרים ומגניטודות:**

detected\_notes = cell(size(dominantFrequencies));

for i = 1:length(dominantFrequencies)

if max(abs(S(:, i))) > magnitude\_threshold % אם המגניטודה גבוהה מספיק

[~, index] = min(abs(note\_frequencies - dominantFrequencies(i))); % מצא את התדר הקרוב ביותר

detected\_notes{i} = note\_dict(note\_frequencies(index)); % שמור את התו

end

end

* detected\_notes = cell(size(dominantFrequencies));  
  אתה יוצר מערך של תאים (cell array) כדי לאחסן את התווים שזוהו עבור כל תדר דומיננטי.
* בלולאת for אתה עובר על כל אחד מהתדרים הדומיננטיים (שנמצאו קודם בתהליך), ובודק את **העוצמה של התדר**.
  + max(abs(S(:, i))) מחשב את העוצמה המקסימלית עבור המקטע הנוכחי של התדר.
  + אם העוצמה גבוהה יותר מרף המגניטודה שהגדרת, כלומר אם מדובר בתדר חזק מספיק, אתה מחפש את **התדר הקרוב ביותר** מתוך התדרים המוגדרים במילון שלך:
    - min(abs(note\_frequencies - dominantFrequencies(i))) מחשב את ההבדלים בין התדר הדומיננטי לבין כל אחד מהתדרים במילון ומוצא את התדר הקרוב ביותר.
* לאחר שמצאת את התדר הקרוב ביותר, אתה **שומר את השם של התו** (ממילון note\_dict) במערך detected\_notes.

**מה עושה הקוד הזה?**

הקוד הזה מטרתו לזהות את התווים שהתאימו לתדרים הדומיננטיים שנמצאו בקובץ האודיו. בעזרת המילון שיצרת, אתה מקשר כל תדר דומיננטי שהתגלתה לו עוצמה גבוהה לתו המוזיקלי המתאים.