ASHESI UNIVERSITY

EE320: Signals and Systems

Gender Recognition from audio signals

ADWOA AGYEIWAA MARFO

BRDGET NANA BERNIAH KWOFIE

MCKAYLA NYAMEDOR GYAMERAH

MAXWELL BOSIAKO ANTWI

OBED KPAKPO ALLOTEY BABBINGTON

Table of Contents

I. Introduction 2

II. Objectives 2

III. Description of Project: 3

IV. Description of each module: 3

V. Project Code: A Detailed Overview 7

VI. Results and Discussion…………………………………………………………….15

VII. Class of Concepts / Key Concepts 18

VIII. Limitations and further improvements 21

IX. Lessons Learnt 22

X. Conclusion 22

I. Introduction

In the modern era of digital communication, gender identification through voice recognition has emerged as a crucial area of study and application. Voice recognition technology has found numerous uses in various sectors including healthcare diagnostics, security, telecommunications, marketing, and scientific research, accentuating its importance in contemporary times. The expanding interest of scientists and engineers has led to diverse ways of formulating effective approaches regarding voice recognition, which highlights its critical role in digital signal processing (DSP). Fundamentally, DSP deals with the analysis, alteration, as well as interpretation of voice signals.

By first acquiring the analogue signal and digitalizing it, the signal undergoes filtering by the Butterworth function in MATLAB, which is able to remove noise, without distorting the integrity signal. Thereafter, the phase of feature extraction follows, utilizing the Fast Fourier Transform (FFT) function to transform the signal from the time domain to the frequency domain. This transition to the frequency domain allows for frequency analysis to be initiated, whereby the signal is deconstructed into numerous frames with unique fundamental frequencies, which are then averaged to compare to a threshold frequency, allowing the distinction between female and male voice samples.

This project employs MATLAB as the principal tool for the processing and analysis of the data. MATLAB's Fast Fourier transform, Butterworth function, as well its graphical user interface (GUI), makes it a more than suitable platform for analysing, testing, as well as simulating the outcome of the project.

This paper aims to explain the technical complexities and applications of voice recognition, in digital signal processing, pinpointing the interconnectedness between the theoretical and practical aspects in the field of science and engineering.

II. Objectives

1. **Voice Gender Classification:** The primary goal is to accurately classify audio samples as either male or female based on their acoustic features.

2. **Accuracy Evaluation:** Assess the accuracy and performance of the gender recognition system through testing and validation using a dataset with known gender labels.

3. **Optimization Evaluation:** To compare and evaluate the accuracy and performance of the system to that of an in-built MATLAB function as a measure of optimization.

III. Description of Project:

This project is an automatic gender classification system based on a person’s voice using the fast Fourier transform (FFT). The project uses two main methods for classification. The first method is the built-in pitch function which extracts the fundamental frequency of an audio signal and directly compares it to a predetermined frequency to determine gender. This threshold frequency is 165Hz, below which is a male voice, and above which is a female. The user-defined function employs a user-defined Butterworth filter function to generate the relevant frequency components in the audio signal, analyzes the fundamental frequency, and classifies the voice as male or female based on a predefined threshold. The system is divided into a front-end and a back-end system, with the front-end responsible for feature extraction, including the use of the Fast Fourier Transform (FFT) algorithm to analyse the power spectrum and extract the required frequencies. Meanwhile, the back end, acting as the classifier, creates a gender model based on these extracted frequencies and compares them to known values for gender identification.

IV. Description of each module:

Module 1: Audio file upload

This module handles the task of uploading audio files into the system for gender recognition analysis. It's the first step in the process and involves providing functionality for users to select and upload audio files from their local storage. In the MATLAB app created for this project, this module includes components such as buttons or file input fields, allowing users to browse their file system and select the audio files they want to analyse. Upon selection, the chosen audio file is read into the system for further processing.

The MATLAB app's interface includes several components to facilitate this process which include:

- Upload Button: This button opens the file selection dialog, allowing users to browse their local storage and select an audio file for analysis.File Input Field: Users input the name of the audio file, including the .wav extension, into this field. It provides a convenient way for users to specify the audio file they want to upload.

Module 2: Audio Playback and Stop

This module is responsible for providing functionality to play and stop the uploaded audio file within the MATLAB app. It enhances the user experience by allowing them to listen to the uploaded audio sample to verify its content and quality.

Key components and functionalities of this module include:

- Listen Audio Button: Once the audio file is uploaded, users can click this button to listen to the uploaded audio sample. It enhances user experience by enabling them to verify the selected audio file.
- Stop Button: This button stops the playback of the audio file, providing users with control over the listening experience.
- Audio Playback Functionality: The **ListenAudioButtonPushed** function associated with the "Listen Audio" button handle the playback of the audio file. This function utilizes MATLAB's audio playback capabilities to play the uploaded audio sample through the system's audio output.
- Stop Functionality: Similarly, functions associated with the "Stop" button trigger the system to cease audio playback when clicked by the user. This functionality ensures that users can stop playback whenever needed.

Module 3: Audio Signal Graph

This module is responsible for displaying a graphical representation of the uploaded audio signal within the MATLAB app. It provides users with a visual representation of the audio waveform, allowing them to observe its characteristics and waveform patterns.

Key components and functionalities of this module include:

- Generate Graph Button: This button triggers the generation and display of the audio signal graph when clicked by the user. It initiates the process of plotting the audio waveform on a graphical axis within the MATLAB app's interface.

 Audio Signal Plotting Functionality: The ‘**GenerateGraphButtonPushed’** function associated with the "Generate Graph" button handles the plotting of the audio signal. These functions utilize MATLAB's plotting capabilities, such as ‘plot’ to create a graphical representation of the audio waveform based on the data stored in the uploaded audio file.
 Graphical Axis (UIAxes): The generated audio signal graph is displayed within a graphical axis component (`UIAxes`) of the MATLAB app's interface. This axis provides a dedicated area for plotting and visualizing the audio waveform.
 Graph Customization: The plotted audio signal graph includes parameters such as time (in seconds) on the x-axis and amplitude on the y-axis.

Module 4: FFT Graph

This module is designed to generate and display the Fast Fourier Transform (FFT) graph of the uploaded audio signal within the MATLAB app. The FFT graph provides insights into the frequency components present in the audio signal, allowing users to analyse its spectral characteristics.

Key components and functionalities of this module include:

 Generate Graph Button: Similar to the audio signal graph module, this button generates and displays the FFT graph when clicked by the user. It initiates the process of computing the FFT of the audio signal and plotting its frequency spectrum.
 FFT Computation: the function ‘FFT” is associated with the "Generate Graph" button which computes the FFT (Fast Fourier Transform) of the uploaded audio signal. This process converts the audio signal from the time domain to the frequency domain.
 FFT Plotting Functionality: MATLAB functions such as ‘FFT’ and ‘plot’ are utilized to compute the FFT and plot the frequency spectrum graph, respectively. The FFT graph typically displays frequency (in Hertz) on the x-axis and magnitude on the y-axis.
 Graphical Axis (UIAxes): Similar to the audio signal graph module, the FFT graph is displayed within a graphical axis component (‘UIAxes’) of the app's interface. This axis provides a dedicated area for plotting and visualizing the FFT graph.

- Graph Customization: The plotted FFT graph may include features such as axis labels (frequency in Hz), titles, and gridlines to enhance readability and interpretation.

Module 5: Gender Classification

The Gender Classification module is designed for automatic gender identification based on speech signals. It employs signal processing techniques to extract relevant features from the uploaded audio sample and uses classification algorithms to determine the gender of the speaker.

Key components and functionalities of this module include:

- Feature Extraction: The module extracts relevant features from the speech signal, such as fundamental frequency (pitch), energy, or spectral characteristics. These features serve as discriminative attributes for differentiating between male and female voices.
- Classifier: A classification algorithm is employed to analyze the extracted features and classify the gender of the speaker. Common classifiers used in gender classification tasks include threshold-based methods, support vector machines (SVM), artificial neural networks (ANN), or decision trees.
- Thresholding Technique: In some cases, a simple thresholding technique based on predefined thresholds for specific features (e.g., fundamental frequency) may be used for gender classification. For example, if the fundamental frequency of the speech signal exceeds a certain threshold, it may be classified as a female voice; otherwise, it may be classified as a male voice.
- Comparison with Known Values: The extracted features from the uploaded audio sample are compared with known values or models derived from a training dataset containing labeled male and female voice samples. This comparison enables the classifier to make informed decisions about the gender classification of the speaker. Output Display: The module displays the classification result, indicating whether the speaker is classified as male or female. This information is presented to the user through the MATLAB app's interface, providing immediate feedback on the gender identification process.

Module 6: Audio Play of gender results

The "Audio Play of Gender Results" module in the MATLAB app enhances user interaction by providing an auditory confirmation of the gender classification results. After the gender classification process, the module allows users to listen to the audio sample with the identified gender label. This feature offers a more engaging and intuitive way for users to validate the accuracy of the gender classification.

Key functionalities of this module include:

- Play Audio: Upon completion of the gender classification process, users can click a button to initiate playback of the audio sample. The audio is played through the computer's speakers or headphones, allowing users to hear the voice of the speaker.
- Gender Label: Alongside the audio playback button, the gender label corresponding to the classification result (e.g., "Male" or "Female") is displayed on the app interface. This label informs users of the gender identified by the classification algorithm.
- Auditory Confirmation: By listening to the audio sample, users can confirm whether the identified gender aligns with their perception. This auditory confirmation provides additional confidence on the accuracy of the gender classification and enhances the overall user experience.

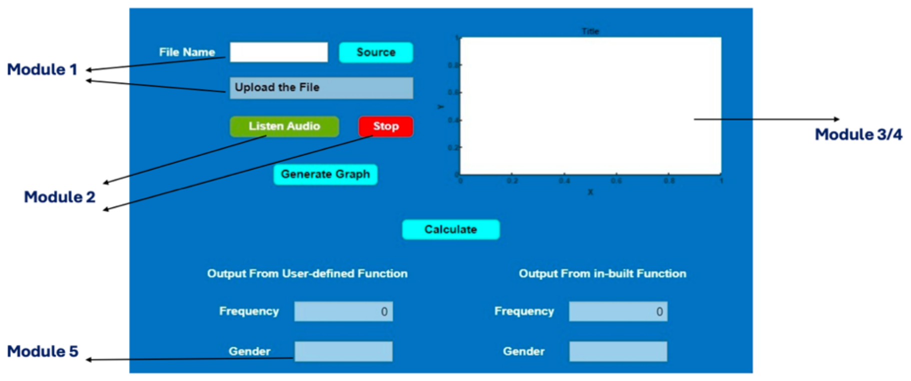


Fig 1.0

An illustration of the different modules

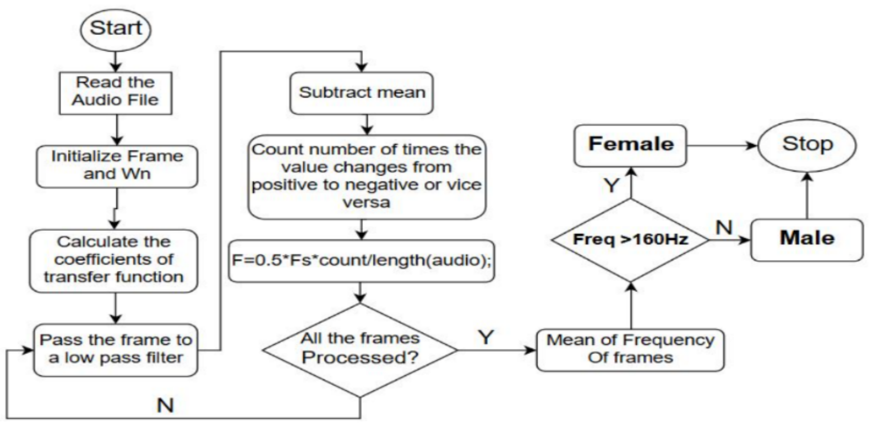
V. Project Code: A Detailed Overview

The MATLAB code required to execute this project can be of itself classified into two major functional parts:

1. **Frontend**: An interactive Graphical User Interface (G.U.I.) was created using the MATLAB App Designer, to provide an interface for users to navigate seamlessly through the different features and functionalities that characterize the project's modules.
2. **Backend**: A set of programs written and tested within MATLAB scripts, designed to execute the project's modules from the upload of audio files through to the classification of gender. Owing to the project's optimization objective, two main scripts are written for the backend functionality: a **User-Defined Implementation**, and an **In-Built Function Implementation** saved as *Male_Female_Voice_Recognizer.m* and *Male_Female_inbuilt_pitch.m* respectively. These would be the focus of this report's overview of the code used within the project.

User-Defined Implementation

The user-defined implementation code is guided by a set of processes represented within the flow chart below:


Start
Read the Audio File
Initialize Frame and Wn
Calculate the coefficients of transfer function
Pass the frame to a low pass filter
N
Subtract mean
Count number of times the value changes from positive to negative or vice versa
F=0.5*Fs*count/length(audio);
All the frames Processed?
Y
Mean of Frequency Of frames
Freq >160Hz
Y
N
Female
Male
Stop

Process 1: *Read the Audio File*

% Read Audio Sample
[y, Fs] = audioread('samples/m2.wav');

This code line reads an audio file named *m2.wav* in the sample's directory. It loads the audio data into the variable y, and it also stores the sampling frequency of the audio in the variable Fs.

Process 2: *Initialize Frame*

% Set Frame Rate

frame = 3500;

This line of code initializes a variable named frame to a value of 3500. In the context of audio processing, this variable represents the length of each frame in the analysis.

Frames are used to segment audio signals for analysis because audio signals are typically non-stationary, meaning their properties change over time. By dividing the signal into frames, we can analyze smaller, more stationary segments of the signal.

Process 3: *Calculate the coefficients of the transfer function (for Butterworth filter)*

% Preprocessing audio signal
[b0, a0] = mybutter(350/(Fs/2)); % Get the coefficient of the filter matrix
This line of code begins the process of designing a Butterworth filter by retrieving the filter coefficients from the *mybutter* function. In MATLAB, a Butterworth filter is typically defined by two sets of coefficients: **b** - numerator coefficients and **a -** denominator coefficients. In this case, **b0** represents the numerator coefficients, and **a0** represents the denominator coefficients.

The argument 350/(Fs/2) specifies the cutoff frequency of the filter relative to the Nyquist frequency (half of the sampling frequency Fs). This cutoff frequency determines the range of frequencies that the filter allows to pass through relatively unaffected, while attenuating frequencies outside this range.

Process 3.5: *Inner workings of* `mybutter` *Function*

```
function [B, A] = mybutter(W)
      % Input:
            % - W: Normalized cutoff frequency
      % Output:
            % - B: Numerator coefficients of the Butterworth filter
            % - A: Denominator coefficients of the Butterworth filter
      % Convert normalized cutoff frequency to analog domain
      V = tan (W * 1.5707963267948966); % Calculate V using tan mapping
      % Calculate squared magnitude of V
      Sg = V ^ 2;

      % Calculate poles of analog Butterworth filter using bilinear transformation
      Sp = V * [-1-1i, -1+1i] / sqrt(2);

      % Transform poles from s-plane to z-plane
      P = (1 + Sp)./ (1 - Sp);
```

```
        % Calculate gain for scaling
        G = real (Sg / prod(1 - Sp));

        % Construct numerator coefficients based on gain
        B = G * [1, 2, 1];
        % Construct denominator coefficients using roots of transformed poles
        A = real(poly(P));
end
```

The function starts by initializing variables V, Sg, and Sp. V represents the mapped cutoff frequency in the analog domain, Sg represents the squared magnitude of V, and Sp represents the poles of the analog Butterworth filter.

The poles Sp are transformed from the analog s-plane to the z-plane using the bilinear transformation. This transformation ensures stability in the digital domain. Poles P and gain G are calculated from the transformed poles Sp. The numerator coefficients B are constructed based on the gain G, while the denominator coefficients A are computed from the transformed poles P using the poly function. The function returns the numerator coefficients B and the denominator coefficients A, which represent the digital Butterworth filter.

Process 4: *Passing the Frame through a low pass [butterworth] filter*

```
%% Identify the frequency of each frame
for i = 1:length(y)/frame

% Extract current frame
x = y(1+(i-1)*frame:i*frame);

% Compute absolute value of the frame
xin = abs(x);

% Filter the frame using Butterworth filter
xin = myfilter(b0, a0, xin);
```

In this portion of code, the loop iterates over the audio signal `y` in frames of length `frame`. Inside the loop, it extracts the current frame `x` from the audio signal. It then calculates the absolute value of the frame `x` to focus on its amplitude characteristics, discarding phase information. The absolute value of the frame is then filtered using a Butterworth filter defined by coefficients b0 and a0 obtained earlier. This filtering helps attenuate high-frequency components (noise) in the frame, enhancing gender-relevant features.

Process 4.5: *Inner workings of* `myfilter` *Function*

```
function filtered = myfilter(b, a, raw)
% Input:
% - b: Numerator coefficients of the filter (feedforward coefficients)
% - a: Denominator coefficients of the filter (feedback coefficients)
% - raw: Input signal to be filtered
% Output:
% - filtered: Filtered signal

% Initialize filtered signal
filtered = zeros(size(raw));

% Apply filter to the input signal
for n = 3:size(raw,1)
      % Compute the output of the filter using the difference equation
      filtered(n,1) = b(1)*raw(n,1) + b(2)*raw(n-1,1) + b(3)*raw(n-2,1) ...
      - a(1)*filtered(n,1) - a(2)*filtered(n-1,1) - a(3)*filtered(n-2,1);
end
end
```

This function *`myfilter`* applies a digital filter to the input signal raw using the difference equation. It takes the numerator coefficients `b` and the denominator coefficients `a` of the filter as input and computes the filtered signal using a loop over the input samples. The output of the filter is stored in the variable `filtered`.

Process 5: Subtracting Mean (Removing DC Component) and Computing Zero Crossing Rate

```
% Remove DC component
xin = xin - mean(xin);

% Store filtered frame
x_out(1+(i-1)*frame:i*frame,1) = xin;

% Compute Zero Crossing Rate
x2 = zeros(length(xin),1);
x2(1:length(x)-1) = xin(2:length(x));
zc = length(find((xin > 0 & x2 < 0) | (xin < 0 & x2 > 0)));
```

By subtracting the mean of the signal (`xin`), the code removes any constant offset or DC component. This helps focus on variations or fluctuations in the signal that may contain gender-relevant information. The filtered signal (`xin`) is stored in the variable `x_out`. This allows for further analysis or processing of the filtered frames. The zero-crossing rate (ZCR) is calculated for each frame. ZCR represents the rate at which the signal changes its sign, which can be indicative of speech characteristics such as pitch or voicing.

Process 6: *Computing Fundamental Frequency per Frame*

```
% Fundamental Frequency Formula
F0(i) = 0.5 * Fs * zc / length(x);
```

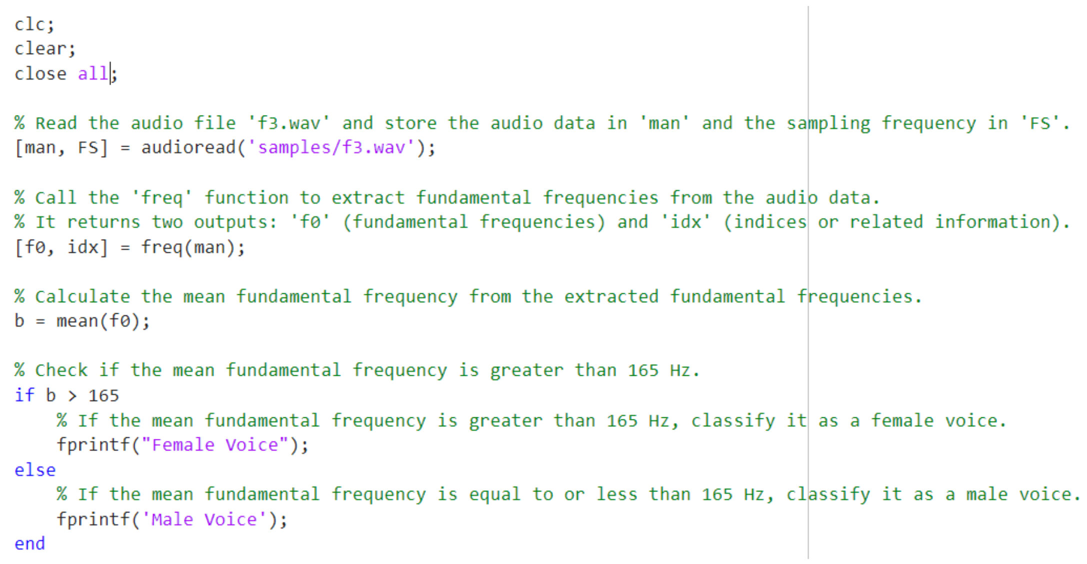
By dividing the number of zero crossings by the length of the frame and multiplying by the sampling frequency (`Fs`), this line effectively estimates the fundamental frequency of the signal within the frame.

Process 7: *Computing mean fundamental frequency and classifying the gender*

```
% Compute Mean Frequency
Fx = mean(F0); % Take mean of all the frequency for each frame

%% Display the output frequency
fprintf('Estimated frequency is %3.2f Hz.\n', Fx);

%% Display the final Gender
if Fx > 160 % Set the threshold
      fprintf('Female Voice\n');
else
      fprintf('Male Voice\n');
end
```

This code culminates the user-defined implementation by computing the mean frequency of the audio signal frames, comparing it to a threshold value (160 Hz) to determine the likely gender, and displaying the result. In the MATLAB App GUI, the result would be displayed as both texts, and played audio recording indicating the classified gender.

In-Built Function Implementation



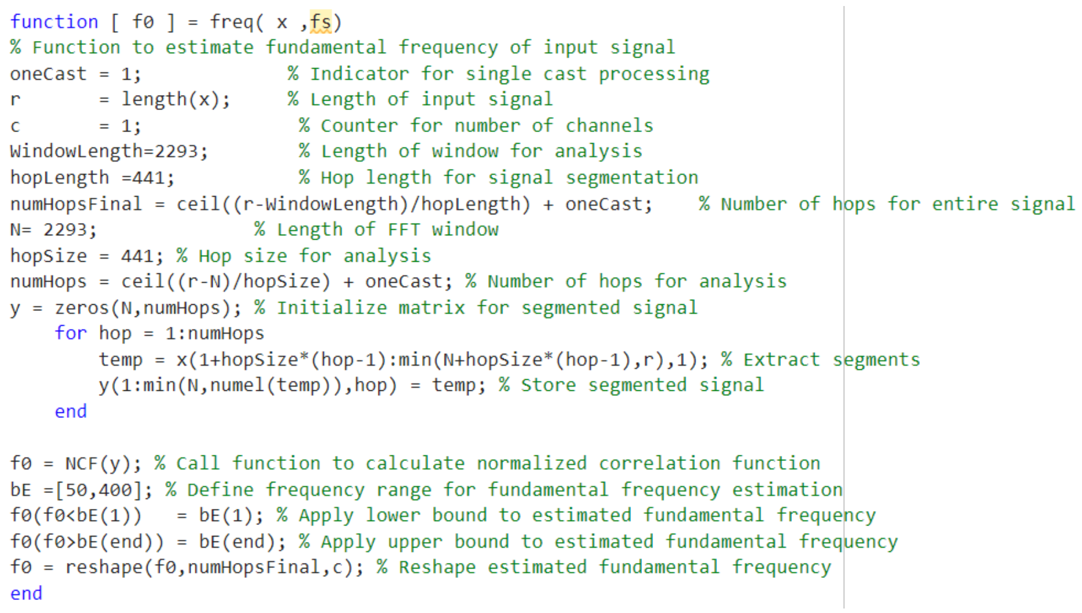
```
clc;
clear;
close all;

% Read the audio file 'f3.wav' and store the audio data in 'man' and the sampling frequency in 'FS'.
[man, FS] = audioread('samples/f3.wav');

% Call the 'freq' function to extract fundamental frequencies from the audio data.
% It returns two outputs: 'f0' (fundamental frequencies) and 'idx' (indices or related information).
[f0, idx] = freq(man);

% Calculate the mean fundamental frequency from the extracted fundamental frequencies.
b = mean(f0);

% Check if the mean fundamental frequency is greater than 165 Hz.
if b > 165
    % If the mean fundamental frequency is greater than 165 Hz, classify it as a female voice.
    fprintf("Female Voice");
else
    % If the mean fundamental frequency is equal to or less than 165 Hz, classify it as a male voice.
    fprintf('Male Voice');
end
```

The above code is designed to determine whether a given audio sample represents a male or female voice based on its fundamental frequency. The code starts by loading an audio file named 'f3.wav' using the *'audioread'* function. The audio sample is stored in the variable `man`, and the sampling frequency (in Hz) is stored in the variable 'FS'. The *'freq'* function (a sub function) is assumed to calculate the fundamental frequency of the input audio signal. The function returns two outputs: **'f0'**, which likely contains the fundamental frequency values, and 'idx', which may correspond to indices, or some information related to the frequencies**.** The code then calculates the mean of the fundamental frequency values stored in **'f0'** using the *'mean'* function, and the result is stored in the variable 'b'.

4. Voice Gender Determination: Based on the calculated mean fundamental frequency (`b`), the code decides whether the voice is male or female. It does this by comparing the mean fundamental frequency value to a threshold value of 165 Hz. If the mean fundamental frequency is greater than 165 Hz, it prints "Female Voice"; otherwise, it prints "Male Voice".

The freq() function from the inbuilt function has two main sub-functions: NCF Function and getCandidates Function

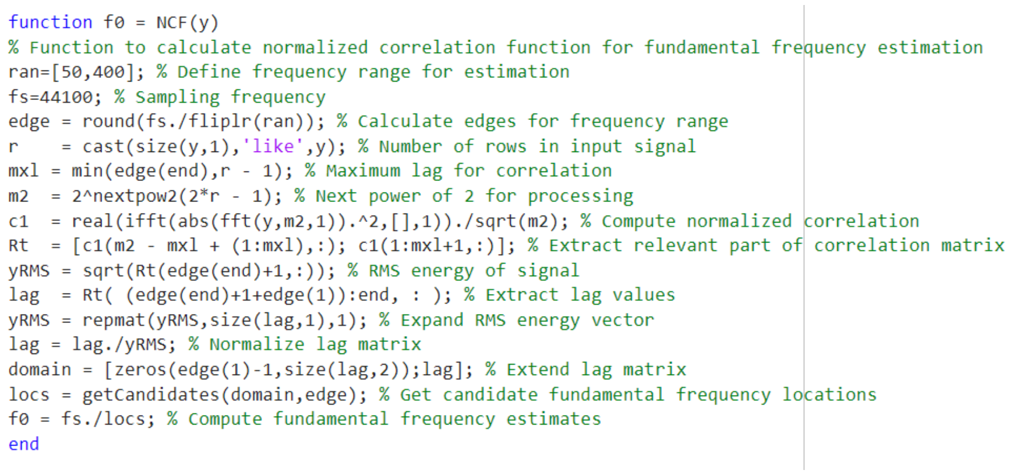
The freq() function



```
function [ f0 ] = freq( x ,fs)
% Function to estimate fundamental frequency of input signal
oneCast = 1;             % Indicator for single cast processing
r       = length(x);     % Length of input signal
c       = 1;              % Counter for number of channels
WindowLength=2293;        % Length of window for analysis
hopLength =441;           % Hop length for signal segmentation
numHopsFinal = ceil((r-WindowLength)/hopLength) + oneCast;    % Number of hops for entire signal
N= 2293;              % Length of FFT window
hopSize = 441; % Hop size for analysis
numHops = ceil((r-N)/hopSize) + oneCast; % Number of hops for analysis
y = zeros(N,numHops); % Initialize matrix for segmented signal
    for hop = 1:numHops
        temp = x(1+hopSize*(hop-1):min(N+hopSize*(hop-1),r),1); % Extract segments
        y(1:min(N,numel(temp)),hop) = temp; % Store segmented signal
    end

f0 = NCF(y); % Call function to calculate normalized correlation function
bE =[50,400]; % Define frequency range for fundamental frequency estimation
f0(f0<bE(1))   = bE(1); % Apply lower bound to estimated fundamental frequency
f0(f0>bE(end)) = bE(end); % Apply upper bound to estimated fundamental frequency
f0 = reshape(f0,numHopsFinal,c); % Reshape estimated fundamental frequency
end
```

This function is used in estimating the fundamental frequency of the input signal. It follows a frame-based approach to analyze the signal. The input signal x is segmented into frames with a specified window length (WindowLength = 2293) and hop size (hopLength = 441). The number of frames (numHopsFinal) is calculated based on the signal length and frame parameters. Each frame undergoes fundamental frequency estimation using the NCF function. The resulting fundamental frequency estimates are adjusted to ensure they fall within a specified frequency range (bE = [50, 400]), which is typically relevant for human voice. Finally, the estimated fundamental frequencies are reshaped to match the format expected by the inbuilt function code which is used in facilitating further processing.

The NCF() function



```
function f0 = NCF(y)
% Function to calculate normalized correlation function for fundamental frequency estimation
ran=[50,400]; % Define frequency range for estimation
fs=44100; % Sampling frequency
edge = round(fs./fliplr(ran)); % Calculate edges for frequency range
r    = cast(size(y,1),'like',y); % Number of rows in input signal
mxl  = min(edge(end),r - 1); % Maximum lag for correlation
m2   = 2^nextpow2(2*r - 1); % Next power of 2 for processing
c1   = real(ifft(abs(fft(y,m2,1)).^2,[],1))./sqrt(m2); % Compute normalized correlation
Rt   = [c1(m2 - mxl + (1:mxl),:); c1(1:mxl+1,:)]; % Extract relevant part of correlation matrix
yRMS = sqrt(Rt(edge(end)+1,:)); % RMS energy of signal
lag  = Rt( (edge(end)+1+edge(1)):end, : ); % Extract lag values
yRMS = repmat(yRMS,size(lag,1),1); % Expand RMS energy vector
lag = lag./yRMS; % Normalize lag matrix
domain = [zeros(edge(1)-1,size(lag,2));lag]; % Extend lag matrix
locs = getCandidates(domain,edge); % Get candidate fundamental frequency locations
f0 = fs./locs; % Compute fundamental frequency estimates
end
```

This function is responsible for computing the normalized correlation function (NCF), a fundamental step in fundamental frequency estimation. It first determines the frequency range for fundamental frequency estimation based on the provided input range (ran). The input signal **y** is processed to calculate the NCF, involving operations such as FFT, autocorrelation, and normalization. The computed NCF provides insights into the periodicity of the signal, aiding in identifying potential fundamental frequency candidates. These candidate fundamental frequencies are further processed to refine the estimation.

The getCandidates() Function:

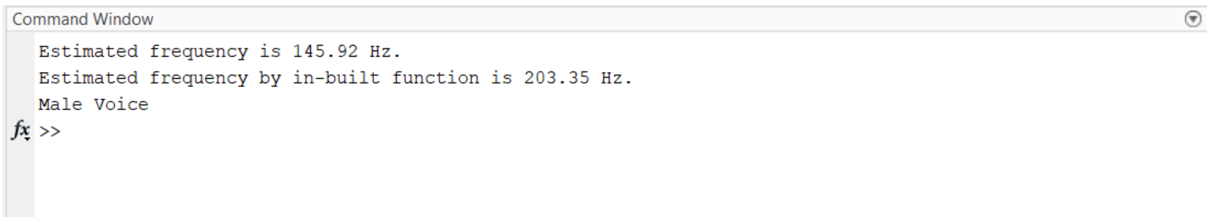
```
function locs = getCandidates(domain,edge)
% Function to extract candidate fundamental frequency locations
numCol = size(domain,2); % Number of columns in domain
locs = zeros(numCol,1,'like',domain); % Initialize vector for locations
lower  = edge(1); % Lower bound of frequency range
upper  = edge(end); % Upper bound of frequency range
assert(upper<192000); % Check upper bound does not exceed Nyquist frequency
domain = domain(lower:upper,:); % Extract relevant portion of domain
for c = 1:numCol
    [~,tempLoc] = max( domain(:,c) ); % Find maximum value in each column
    locs(c) = tempLoc; % Store location of maximum value
end
locs = lower + locs - 1; % Adjust locations to match original domain
end
```

The **getCandidates** identifies candidate fundamental frequency locations by scanning the computed NCF. It extracts peaks from the NCF, which correspond to potential fundamental frequency values. The locations of these peaks are returned as candidate fundamental frequencies. By extracting specific candidate frequencies, this function contributes to determining the likely fundamental frequency of the input signal. The key part of this function involves scanning the domain matrix, which represents the computed NCF, and extracting peaks that correspond to potential fundamental frequency values. This is achieved by finding the maximum value in each column of the domain matrix. The locations of these peaks are then adjusted to match the original domain, and they are returned as candidate fundamental frequencies (locs).

VI. Results and Discussion

Owing to a MATLAB scripting of the backend program files prior to the GUI incorporation, both sets of results may be analysed - the first being the raw values from the script, and the second, the visual process of interfacing with the program's modules.

MATLAB Script Results:

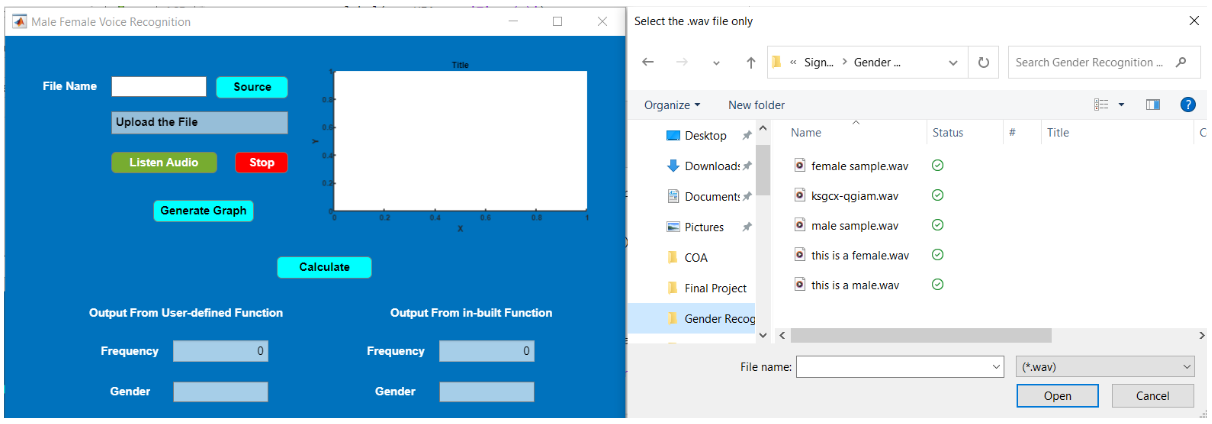


```
Estimated frequency is 145.92 Hz.
Estimated frequency by in-built function is 203.35 Hz.
Male Voice
>>
```

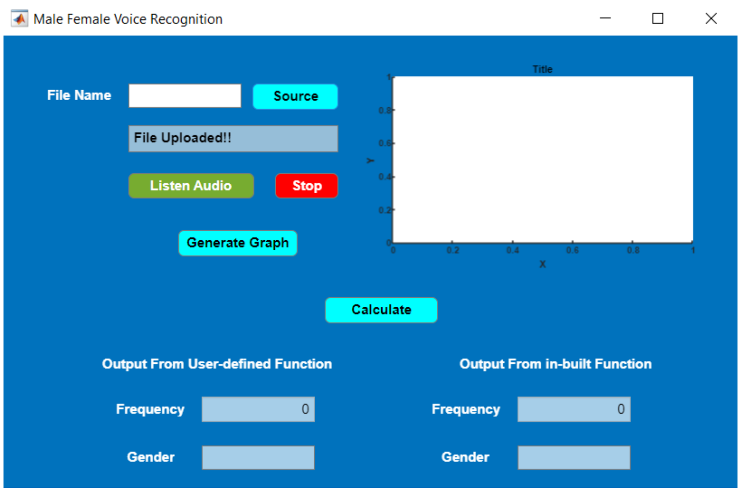
Using both the user-defined and in-built implementations, one observes in the above image the estimated frequencies both implementations produced, and the classified gender of the former (user-defined). In the above, a male sample file was passed through the requisite functions, for which "Male Voice" is aptly displayed as the correct classification.

MATLAB GUI Results:

Module 1: Audio File Upload



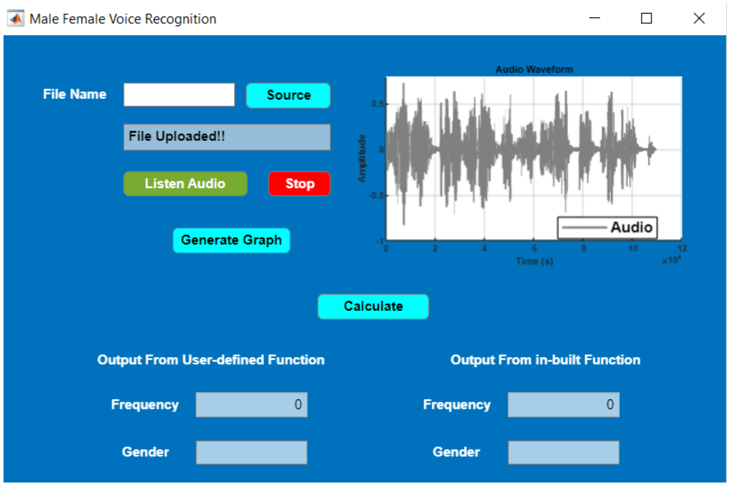
As showcased above, by clicking the 'Source' button, a pop-up window appears, allowing one to select solely .wav files from their device's file collection. For this demonstration, the "male sample.wav" file is chosen. Upon successful upload, the "Upload the File" text display updates to "File Uploaded!!" to validate the success of the upload. This may be observed below:



Module 2: Audio Playback and Stop

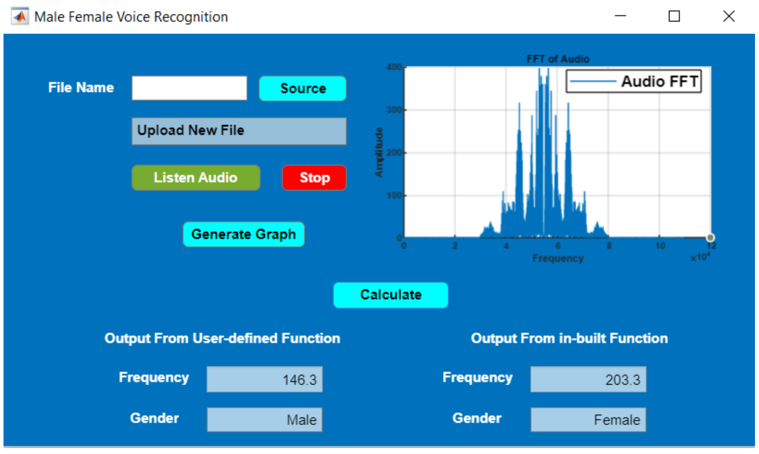
As described in the Module Description section, by clicking on the "Listen Audio" and "Stop" buttons, one can listen to the uploaded audio sample, and stop the playback of the audio file, respectively. Results from this may be heard by an interaction with the uploaded code.

Module 3: Audio Signal Graph



Upon clicking the "Generate Graph" button, a graphical representation of the uploaded audio signal is displayed in the top right region of the interface, shown in the above image. In this demonstration, this waveform is that of a male's voice. However, it is difficult to make relevant classifications upon observations of the raw waveform, necessitating the subsequent modules.

Module 4/5: FFT Graph & Gender Classification



Output From User-defined Function

Frequency 146.3

Gender Male

Output From in-built Function

Frequency 203.3

Gender Female

In the above display, upon clicking "Calculate," two sets of results may be observed: the outputs of both functions, compared side-to-side in the lower half of the interface, as well an FFT graph updating the audio waveform graph, to give the user a visual feel of the frequencies at play within the audio. In this [male sample] demonstration, the user-defined function determines a fundamental frequency of 146.3 Hz, indicating a male's voice, while 203.3 Hz is determined from the in-built implementation, indicating a female's voice. It is from this and other samples that the user-defined implementation was ascertained to be more optimal in gender classification than that more reliant on in-built functionality.

Module 6: Audio Play of Gender Results

After the on-screen outputs of the previous module, an audio recording indicating the determined gender is played, of whose results can also be heard through an interaction with the code.

VII. Class Concepts / Key Concepts

Throughout this project, the key concepts surround Digital Signal Processing, with the main areas being Filtering, Sampling, Feature extraction, and Signal analysis. Below are the explanations of the key concepts included in this project.

Discrete time signals

For any sort of analysis to be initiated, discrete-time signals play a significant role in Digital Signal Processing. At the beginning, the Quantity Under Measurement (QUM), which is the voice signal, is a continuous signal. The transition from continuous-time signal to discrete-time signal is done by sampling and quantization, which entails measuring the amplitude of the continuous-time signal at regular intervals. The rate at which these measurements are taken is known as the sampling rate. Below is an image depicting the transition from continuous-time signal to discrete-time signal.

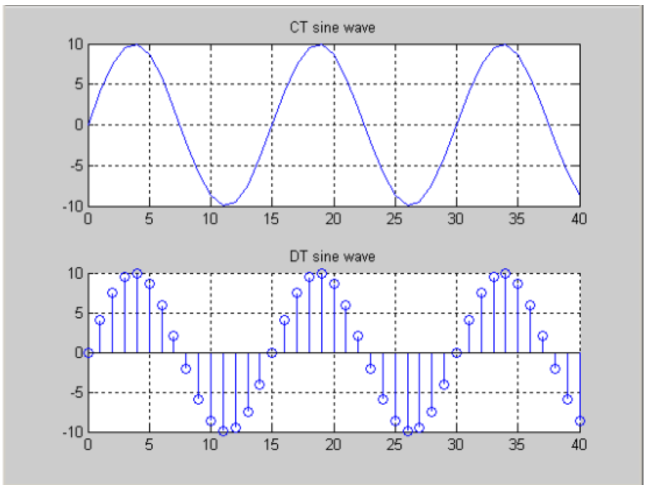


Figure depicting the difference between a continuous-time signal and a discrete-time signal.

Signal Acquisition

In the MATLAB code, there is a function named audioread (), which reads the .wav (Waveform Audio) file. This file is then changed to digital form, by the use of sampling and quantization. This is done to allow the signal to be analysed digitally.

Filtering and noise reduction

After the digitization of the signal, all the noise, as well as unwanted frequencies are removed, using the Butterworth function, which has a flat frequency response and a passband, hence, reducing the distortion to the signal. The Butterworth filter is a type of signal processing filter, characterized by the cutoff frequency and the order of the filter, which is designed to select a specified frequency band. By having a smooth transition between its passband (without ripples) and stopband, as compared to ideal filters, it delivers a flat frequency response, at the expense of a wide pass band. This ensures that the audio signal is not tampered with and distorted, and has a uniform frequency response, allowing the audio signal to sound appealing. The higher the order of a Butterworth filter, the closer it is to an ideal filter. Below are images of Butterworth filters and ideal filters:

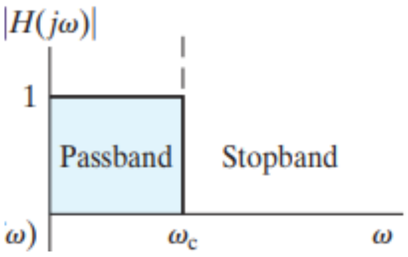


Figure depicting an ideal transition between the passband and stopband of a low- pass filter

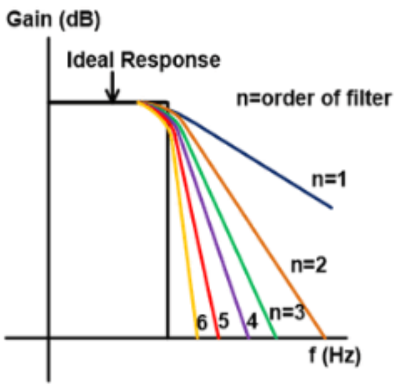


Figure depicting a Butterworth transition between the passband and stopband of a low- pass filter.

Mathematical Representation

Below is the frequency response of a Butterworth filter as a function of angular frequency:

$$H(j\omega) = \frac{1}{\left(1 + \frac{\omega}{\omega_c}\right)^{2N}}$$

Where ω represents frequency, ω_c represents the cutoff frequency and N represents the order of the filter.

Feature extraction, frequency analysis & classification of gender

After the noise and unwanted frequencies are filtered out using the Butterworth function, the Fast Fourier Transform (FFT) is applied, which changes the signal from its time domain to the frequency domain, producing a graph of amplitude against frequency. As a result of this, the voice signal is broken down into various time frames, each having their own fundamental frequencies. Thereafter, an average frequency is calculated, which is compared to the frequency value of 165 Hz. Below 165 Hz is said to be that of a male and above is said to be a female's voice.

VIII. Limitations and further

Assessing our project, we identified some limitations which we hope to work on improving. Some of these limitations are listed below.

Overly reliance on energy-based thresholding

Our gender recognition system overly depends on energy-based thresholds to classify gender. Though energy is a good indicative of gender difference in speech, it does not capture all the relevant characteristics. The limitation to this is that when both genders have similar energy levels within a certain frequency band, it will be difficult to accurately classify them. This can result in false-positive in our system, as both genders might have the same pitch (fundamental frequency). A clear example is when a male talks with a higher pitch which has a frequency greater than the fundamental frequency of 165Hz, it will be detected as female. Similar error occurs when a female talks with a lower frequency below the threshold. This will prompt our system to determine it as a male, since these are the pre-determined ranges for the different genders. Other features such as formants (resonant frequency), duration etc. can be extracted to further enhance the accuracy of the gender classification. These features capture different aspects of speech and can help improve the accuracy of the gender classification system.

Noise Sensitivity

While our system was filtering some noise in the signals, we didn't consider an environment where two or more people are talking at the same time. For instance, when the environment of the speaker is noisier, it can affect the system's performance as the energy level in the speech signal might fluctuate. This environment might influence the filtering part of our system, as it doesn't cater to noise of that sort. We can further improve our filtering method by adopting techniques like the Mel-frequency cepstral coefficient (MFCCs), which approximates the human ear's response to different frequencies and focus more on the power spectrum and less sensitive to the variation in the energy or the threshold. (Mehrish et. al., 2023).

Dependence on Already trained models

The system depended on already trained models for the threshold frequency. While these models are well trained, it does not provide details on extensive validation and evaluation. Meaning, we cannot further train the model with our sampled data. To further improve our system, we can fine-tune the pre-trained model with our own sampled data with variable accents and pitches. This can help us to further improve the system and add more specifics to our project.

IX. Lessons Learnt

Working on this gender recognition project using MATLAB has provided us with practical experiences, applying class concepts. We have gained insights into signal processing techniques particularly in the context of speech recognition and analysis. We realized that understanding how to extract meaningful features from speech signals is very important in various applications beyond gender recognition. While we didn't train our own Machine Learning model, we noticed that using the right classification algorithm has a greater impact on the classification accuracy of our system. This has made we appreciate the needs to train our model with diverse dataset of male and female voices from different age groups, accents, and speech patterns to improve the performance. This has influenced us to look at further algorithms that can be used to improve our system going forward.

X. Conclusion

This project has demonstrated the potential of using voice characteristics for gender recognition, a process that is relevant to various applications, including security measures and personalized user experiences. In today's world where we do a lot of things online, being able to tell who's speaking based on their voice is important. Our approach, which primarily relied on energy estimation as a threshold value, provided a simple yet effective method for distinguishing between male and female voices. The project's primary objectives revolved around utilizing signal processing techniques, particularly the Fast Fourier Transform (FFT), to extract relevant features from audio signals for gender classification. Two main methods were employed: one involving the built-in pitch function to determine fundamental frequency, and the other using user-defined function for sample audio processing using energy-based thresholding techniques. Each module of the project served a distinct purpose, ranging from audio upload and playback functionalities to gender classification and auditory confirmation of results. MATLAB's versatile capabilities allowed the implementation of these modules, providing a user-friendly interface for data processing and analysis. Overall, this project has provided us with practical experience from our signals and systems class, providing us with insights into signal processing and their application.

References

Anand, A., & Shanmugam, R. (2021). Voice Speech and Recognition—An Overview. In *Lecture notes in networks and systems (Online)* (pp. 347–356). https://doi.org/10.1007/978-981-15-9712-1_29

Kanishk-K-U. (n.d.-a). *Kanishk-K-u/gender-recognition-using-MATLAB: Developed a gender recognition system using MATLAB and simulink.* GitHub. https://github.com/Kanishk-K-U/Gender-Recognition-using-MATLAB#:~:text=In%20this%20project%2C%20the%20speaker,female%20produces%20the%20voice%20sample.

Mehrish, A., Majumder, N., Bharadwaj, R., Mihalcea, R., & Poria, S. (2023). A review of deep learning techniques for speech processing. *Information Fusion*, *99*, 101869. https://doi.org/10.1016/j.inffus.2023.101869

Siddiqui, T. (2023, February 3). Voice recognition system on MATLAB for Beginners. Medium. https://bootcamp.uxdesign.cc/voice-recognition-system-on-matlab-for-beginners-bbf280bdfdaa#:~:text=Firstly%2C%20all%20the%20saved%20files,is%20sent%20from%20the%20directory.