EC3093D - Digital Signal Processing Lab

PROJECT TITLE :

Melspectrogram Generation for Speech Recognition

Submitted by

ADHYUTH NARAYAN (B210650EC)

ADWAYITH K S (B210664EC)

ADITYA TUPPAD (B210038EC)

DEPARTMENT OF

ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY CALICUT

KERALA, INDIA 673601

Objective :

Audio signals are one dimensional - amplitude over time signals, as a result they are harder to process.

We cannot give a simple audio signal for purposes like speech recognition etc, in such cases we use mel-spectrogram, it captures the frequency components of the speech signal in a way which aligns more closely with human perception.

Our objective in this project is to implement Mel spectrogram, a transformation that details the frequency composition of the signal over time.

In this project we will be manually implementing STFT in MATLAB and designing the Mel-Filterbanks. After that we will be testing the output of our Mel spectrogram with the inbuilt function in MATLAB.

Theoretical Background :

1. Audio Signals and Spectrograms

Audio signals are representations of sound waves in a digital format, which are sampled at discrete time intervals. They are 1D signals, with the amplitude of the signal varying with time. Understanding the frequency components present in these signals helps in various audio processing tasks.

Spectrograms - They are visual representations of audio signals which provide insight into their frequency content vs time. This is obtained by applying the Fourier Transform to short segments of the signal, the spectrogram breaks down the signal into its constituent frequencies, and helps in the analysis of change in frequency over time.

2. Short-Time Fourier Transform (STFT)

The Short-Time Fourier Transform (STFT) is a fundamental technique in audio signal processing for analyzing the time-varying frequency components. Unlike the standard Fourier Transform, which operates on the entire signal at once, the STFT divides the signal into short overlapping segments and applies the Fourier Transform to each segment individually.

This process results in a time-frequency representation of the signal, where each point in the spectrogram represents the magnitude of the signal's frequency components at a specific time. The use of overlapping segments helps capture transient features in the signal while maintaining temporal resolution.
Steps for finding STFT :

- First we decide upon which window function to use.
- After that we decide the values for window size and overlap amount, these values can be given at the time of calling the function.
- After that we iterate on the elements of the given window size by applying the window function and then taking DFT which is given by,

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}$$

3. Mel-Scale and Mel-Filterbanks

The Mel scale is a perceptual scale of pitch that reflects the nonlinear relationship between frequency and perceived pitch in human auditory perception. It is based on psychoacoustic studies that show how humans perceive differences in pitch.

Mel Filterbanks are used to convert the linear frequency axis of a spectrogram into the Mel scale. These are designed to mimic the response of the human auditory system, by emphasizing more on lower frequencies than higher ones. Each filter in the bank corresponds to a specific frequency range on the Mel scale; it has overlapping regions which helps to ensure smooth transitions between frequency bands.

4. Mel-Spectrogram

The Mel spectrogram is a representation of the frequency content of an audio signal over time, with the frequency axis scaled as per the Mel scale, it is obtained by multiplying Mel Filterbanks (as a matrix) to the output of the STFT, it results in a spectrogram

Mel spectrograms are widely used in speech processing tasks, as they provide a more relevant representation of the frequency content of speech signals compared to traditional spectrograms.

Steps for Mel-spectrogram conversion

- Find upper and lower mel values.
- Insert equally spaced points in between
- Convert them into corresponding frequency values.

‘Mels’ in the Mel-scale and Hertz is related by the following formula, where m represents Mels and f represents frequency in Hertz. Conversion equations are purely experimental and here we use the Shaughnessy equation given by,

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

Similarly the inverse can be calculated by,

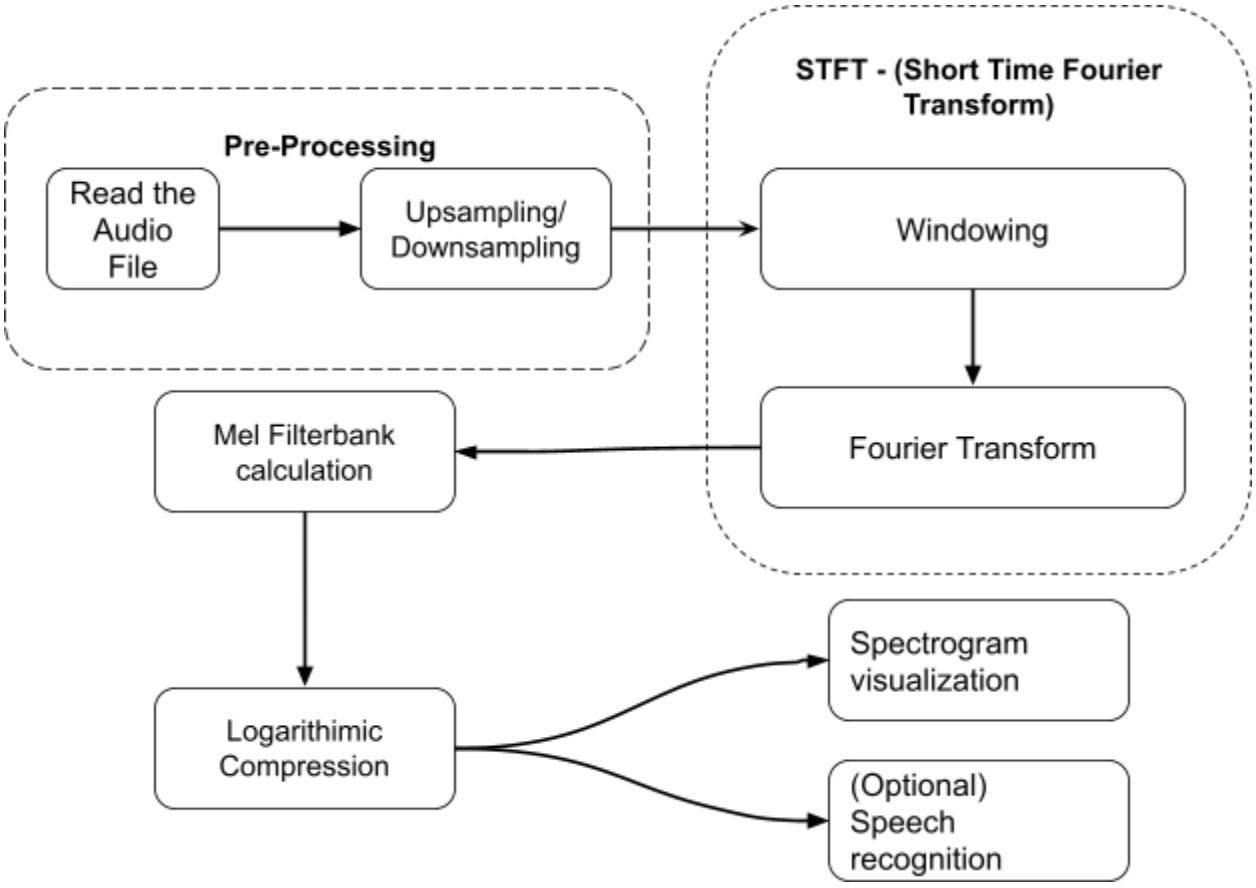
$$f = 700\left(10^{\frac{m}{2595}} - 1\right) = 700\left(e^{\frac{m}{1127}} - 1\right)$$

5. Applications of Mel Spectrograms

Mel spectrograms find applications in various audio processing tasks like automatic speech recognition, speaker identification, music analysis, and environmental sound classification.

Compared to traditional spectrograms they provide more insightful “features” from the data. Features here refers to Mel-frequency cepstral coefficients which are obtained by taking a Discrete Cosine transform.

Block diagram of the implementation :



1. Preprocessing:

   In this step we read the audio file and do some upsampling/downsampling if it is necessary.

2. STFT :

   In this step we make a self defined function in MATLAB, which helps us in getting the short time fourier transform of the input audio signal. The 2 steps involved in this are :

   a. Windowing

      Here we define a window function to be applied on the audio signal for eg: hamming window.

b. Fourier Transform

In this step we take the fourier transform of the windowed audio signal.

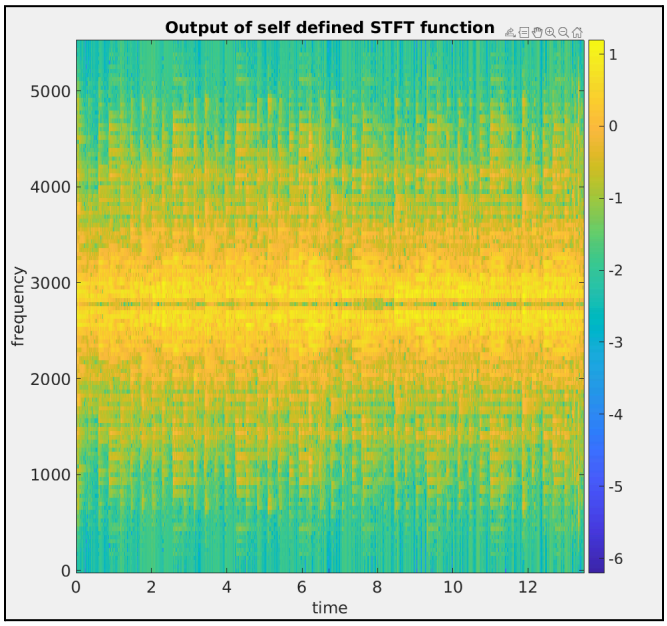
These steps are done iteratively with a given window size and overlap length (these parameters should be defined while calling the function) and at the end we get the STFT of the given audio signal.

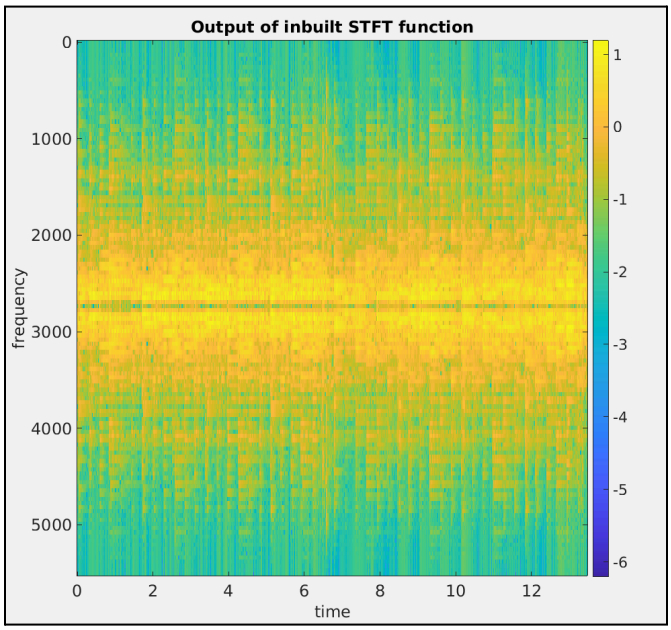
3. Mel Filterbank calculation and Logarithmic Compression:
    a. First, the minimum and maximum frequencies in the Mel scale are estimated based on the provided sampling frequency and window length.
    b. Calculation of width of each filter in the Mel scale and their star and end indices. The filters are half-overlapping. Values are assigned to generate the triangular filters.
4. Spectrogram Visualization
    a. The output spectrogram has 3 axes - frequency, time and power. Generally the x-axis represents time, frequency on the y-axis and a color scale represents the power.
5. Speech Recognition

    As a bonus objective we would like to apply the result on a pre-trained speech recognition model like DeepSpeech from Mozilla or a sound classification model like YAMnet.

Demonstration of Results

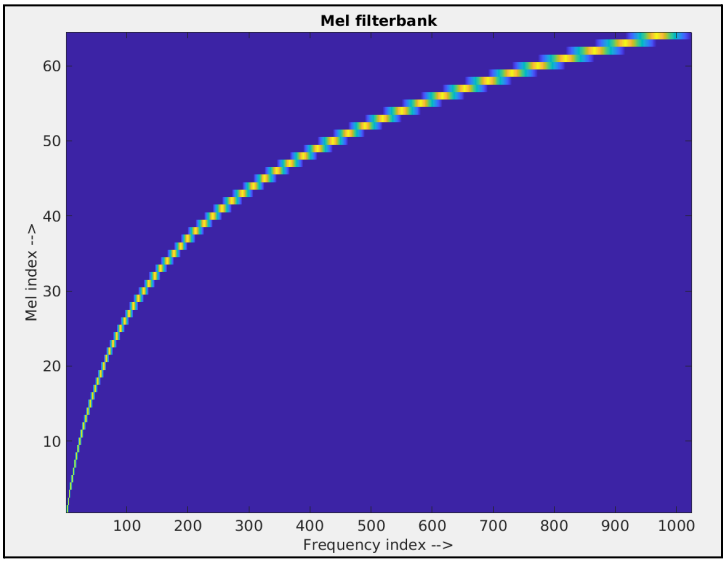
Demonstration of the output of our self defined STFT function and its comparison with the inbuilt STFT function :



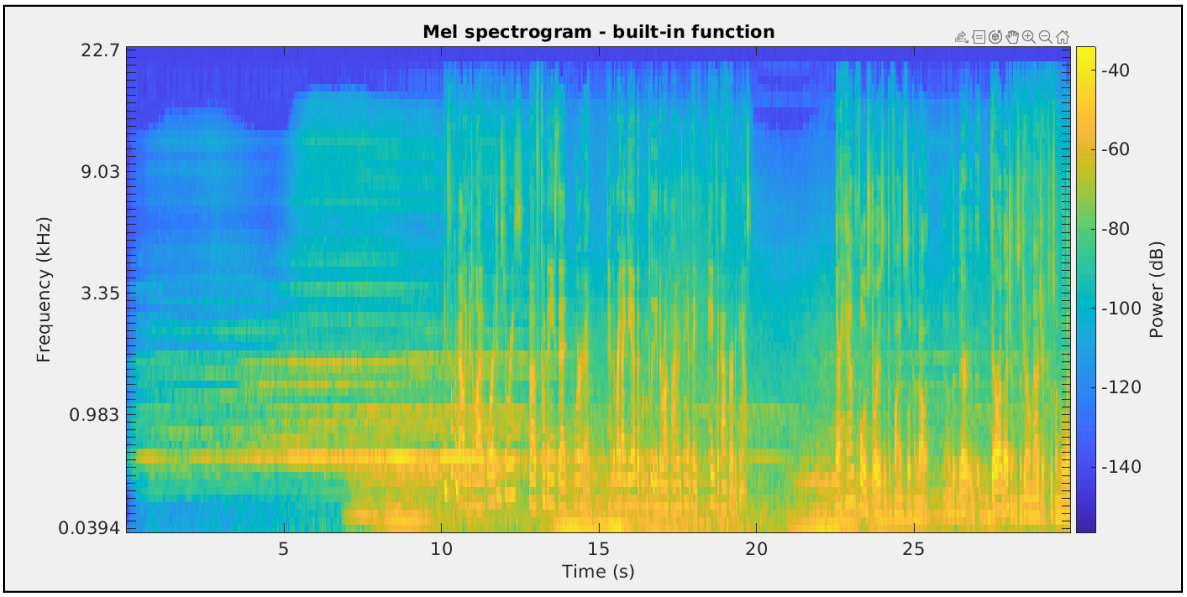


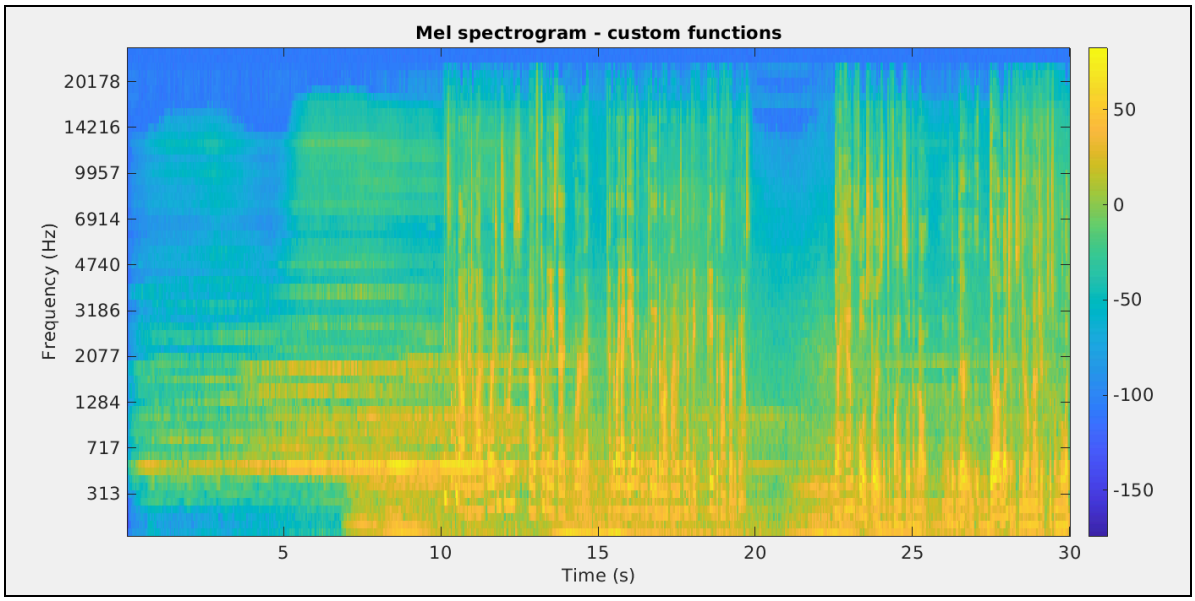
We can see that the output of the inbuilt stft function matches exactly with that of the self-defined function.

Generation of Mel Filterbanks - A Mel Filter bank consisting of 64 filters is generated. Below is a pictorial representation of the generated matrix. It can be seen that each filter is half-overlapping. Y-axis indicates the n-th filter and x-axis corresponds to the frequency bins.



Generation of Mel Spectrogram - An 30 sec Audio clip containing a conversation is used as the input. Fow the window function we chose hanning window.





It is observed that the time, and frequency values are correct and agree perfectly with the observations using the built in MATLAB function. However the power values are comparatively different thus affecting the colors.

CONCLUSION

After conducting a comparison between our custom function and MATLAB's built-in function for Mel-Spectrogram generation, it can be concluded that the outputs closely matched the expected outcome. The analysis included generation of the Mel-Spectrogram using 64 Mel-Filter banks. The comparison revealed similarities in shape of the graph, range of frequencies but some disagreements in the computation of power. Overall, the results demonstrate the effectiveness of our custom function in producing outputs. Further enhancements could focus on optimizing the code to make it more efficient and try to implement this in real world applications like speech recognition and classification.

References

1. Spectrograms - https://www.youtube.com/watch?v=_FatxGN3vAM
2. Mel Spectrograms
    - https://www.youtube.com/watch?v=SJo7vPgRlBQ&pp=ygUIbWVsIHNwZWM%3D
    - https://www.youtube.com/watch?v=9GHCiiDLHQ4&t=116s&pp=ygUIbWVsIHNwZWM%3D
    - https://en.wikipedia.org/wiki/Mel_scale
3. Speech/Sound detection
    - https://github.com/matlab-deep-learning/deepspeech/
4. General reference

- https://www.mathworks.in/help/matlab/
- Audio and Speech Processing with MATLAB Paul Hill, University of Bristol

Code :

```
audio = audioread("resources/audio.wav");
sampling_frequency = 44100;
window_length = 2^nextpow2(0.04*sampling_frequency);
window = hamming(window_length);
number_mels = 128;
audio_stft = fftshift(mystft(audio,sampling_frequency, window, 2));
mel_filterbank =
myMelFilterBank(sampling_frequency,window_length,number_mels);
audio_spectrogram = abs(audio_stft(2:length(window)/2+1,:));
mel_spectrogram = mel_filterbank*audio_spectrogram;
audio_mfcc = dct(log(mel_spectrogram+eps));
figure
imagesc(db(mel_spectrogram))
colormap('jet')
axis xy
```

Code of associated inbuilt functions:

1. myMelFilterBank.m

```
function mel_filterbank = myMelFilterBank(sampling_frequency,
window_length, number_filters)
            % Compute the minimum and maximum mels
            minimum_melfrequency =
2595*log10(1+(sampling_frequency/window_length)/700);
            maximum_melfrequency = 2595*log10(1+(sampling_frequency/2)/700);

            % Derive the width of the half-overlapping filters in the mel
scale (constant)
            filter_width =
2*(maximum_melfrequency-minimum_melfrequency)/(number_filters+1);

            % Compute the start and end indices of the overlapping filters
in the mel scale (linearly spaced)
            filter_indices =
minimum_melfrequency:filter_width/2:maximum_melfrequency;
```

```
            % Derive the indices of the filters in the linear frequency
scale (log spaced)
            filter_indices =
round(700*(10.^(filter_indices/2595)-1)*window_length/sampling_frequency);

            % Initialize the mel filterbank
            mel_filterbank = zeros(number_filters,window_length/2);

            % Loop over the filters
            for i = 1:number_filters

                % Compute the left and right sides of the triangular filters
                % (this is more accurate than creating triangular filters
directly)
                mel_filterbank(i,filter_indices(i):filter_indices(i+1)) ...
                    = linspace(0,1,filter_indices(i+1)-filter_indices(i)+1);
                mel_filterbank(i,filter_indices(i+1):filter_indices(i+2))
...
                    =
linspace(1,0,filter_indices(i+2)-filter_indices(i+1)+1);
            end

            % Make the mel filterbank sparse
            mel_filterbank = sparse(mel_filterbank);

end
```

2. mystft.m

```
function [s, f, t] = mystft(x, fs, window, overlap)
   % x: input signal    % fs: sampling frequency
   % window: window function    % overlap: no.of overlap segments
   N = length(window);
   hop_size = round(N - overlap);
   num_frames = 1 + fix((length(x) - N) / hop_size);
   s = zeros(N, num_frames);
   for i = 1:num_frames
       start_idx = (i - 1) * hop_size + 1;
       end_idx = start_idx + N - 1;
       x_frame = x(start_idx:end_idx);
       x_windowed = x_frame .* window;
       X_frame = fft(x_windowed);
       s(:, i) = X_frame;
   end
   f = (0:N/2) * fs / N;
   t = (0:num_frames - 1) * hop_size / fs;
end
```

3. myMelshow.m

```
function
myMelshow(mel_spectrogram,number_samples,sampling_frequency,window_length,x
tick_step)

            xtick_step = 5;

            % Get the number of mels and time frames
            [number_mels,number_times] = size(mel_spectrogram);

            % Derive the number of seconds and the number of time frames
per second
            number_seconds = number_samples/sampling_frequency;
            time_resolution = number_times/number_seconds;

            % Derive the minimum and maximum mel
```

```
            minimum_mel =
2595*log10(1+(sampling_frequency/window_length)/700);
            maximum_mel = 2595*log10(1+(sampling_frequency/2)/700);

            % Compute the mel scale (linearly spaced)
            mel_scale = linspace(minimum_mel, maximum_mel, number_mels);

            % Derive the Hertz scale (log spaced)
            hertz_scale = 700*(10.^(mel_scale/2595) - 1);

            % Prepare the tick locations and labels for the x-axis
            xtick_locations =
xtick_step*time_resolution:xtick_step*time_resolution:number_times;
            xtick_labels = xtick_step:xtick_step:number_seconds;

            % Prepare the tick locations and labels for the y-axis
            ytick_locations = 0:6:number_mels;
            ytick_labels = round(hertz_scale(1:6:end));

            % Display the mel spectrogram in dB, seconds, and Hz
            imagesc(db(mel_spectrogram))
            axis xy
            colormap(jet)
            xticks(xtick_locations)
            xticklabels(xtick_labels)
            yticks(ytick_locations)
            yticklabels(ytick_labels)
            xlabel('Time (s)')
            ylabel('Frequency (Hz)')

End
```

5. Implementation

```
[audio, sampling_frequency]= audioread("resources/audios.wav");

window_length = 2^nextpow2(0.04*sampling_frequency);
window = hanning(window_length);    % window function to use
number_mels = 64;       % Number of filters to use
audio_stft = mystft(audio,sampling_frequency, window, 1024);
mel_filterbank =
myMelFilterBank(sampling_frequency,window_length,number_mels);
audio_spectrogram = (abs(audio_stft(2:length(window)/2+1,:))).^2; % Squaring
for power calculation
mel_spectrogram = mel_filterbank*audio_spectrogram;

% Pictorial representation of the filterbank
myMelshow(mel_spectrogram, length(audio), sampling_frequency,
window_length)
colormap("parula")
colorbar
title('Mel spectrogram - custom functions');
fontsize(16,"points")

% Custom function
figure
imagesc(mel_filterbank)
axis xy
colormap("parula")
title('Mel filterbank')
xlabel('Frequency index')
ylabel('Mel index')
fontsize(16,"points")

% MATLAB built-in function
figure
melSpectrogram(audio,sampling_frequency, ...
            'Window',hanning(window_length,'periodic'), ...
            'OverlapLength',1024, ...
            'NumBands',64, ...
            'MelStyle','oshaughnessy');

title('Mel spectrogram - built-in function');
fontsize(16,"points")
```