AKADEMIA GÓRNICZO-HUTNICZA

im. Stanisława Staszica w Krakowie

WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

Mechatronic System Identification – project classes

Cepstral Analysis Matlab GUI App – final report

Bartłomiej Jargut
Imię i nazwisko

Mechatronic Engineering
Kierunek studiów

Contents

1. INTRODUCTION

This report concerns the implementation of a Matlab GUI application for the purpose of computing power and complex cepstra. The application allows the user to upload a file with .mat format containing a signal, choose the type of cepstrum, choose high pass or low pass option, adjust liftering and finally compute and display the computed cepstrum.

2. THEORETICAL BACKGROUND

Cepstral analysis or cepstrum was introduced by Bruce P. Bogert, M. J. R. Healy and John W. Tukeyt from Bell Telephone Laboratories and Princeton University in an article titled *"The Quefrency Alanysis of Time Series for Echos: Cepstrum, Pseudo_Autocovariance, Cross-Cepstrum and Saphe Cracking"* as a tool for investigating periodic structures in frequency spectra, such as echoes, reflections or harmonic frequencies such as partials or overtones. They also coined the terms like "quefrency", "cepstrum" "alanysis" and "saphe" as equivalent of their counterparts in spectral analysis. In general, cepstrum is the sequence of the following mathematical operations:

- Transformation of signal from time domain to the frequency domain via Fourier transform
- Computation of the logarithm of the spectrum
- Transformation to frequency domain.

Cepstrum has many variants, but in this project we are concerned about power and complex cepstra.
Cepstrum was originally invented for characterizing seismic echoes from earthquakes and bomb explosions and determining the fundamental frequency of human speech and to analyze radar signal returns. In speech processing, cepstrum allows us to separate the vocal excitation (pitch) and vocal tract (formants). This is due to the fact that these effects are additive in the logarithm of power spectrum and can be separated. Nowadays, cepstrum finds applications in reflection inference (for example radars or sonars), speech processing, analyzing brain waves or detection of gearbox or turbine blade defects based on harmonic patterns, and most recently in electromyography in detection of neuromuscular abnormalities.

3. IMPLEMENTATION

3.1. GUI

The initial step was to create GUI elements so that we can use the values returned by these elements in the backend.

```
function cepstralAnalysisApp
    % Create the GUI figure
    f = figure('Name', 'Cepstral Analysis App', ...
        'Position', [100 100 1000 600]);

    % UI components
    uicontrol('Style', 'pushbutton', 'String', 'Load Audio', ...
        'Position', [20 550 100 30], 'Callback', @loadAudioCallback);

    uicontrol('Style', 'text', 'Position', [150 555 80 20], ...
        'String', 'Lifter Type:');
    lifterPopup = uicontrol('Style', 'popupmenu', ...
        'String', {'None','Low-pass','High-pass'}, ...
        'Position', [230 550 100 30]);

    uicontrol('Style', 'text', 'Position', [350 555 80 20], ...
        'String', 'Cutoff:');
    cutoffSlider = uicontrol('Style', 'slider', ...
        'Position', [420 550 150 30], 'Min', 1, 'Max', 200, ...
        'Value', 40);

    uicontrol('Style', 'text', 'Position', [600 555 100 20], ...
        'String', 'Cepstrum Type:');
    cepstrumTypePopup = uicontrol('Style', 'popupmenu', ...
        'String', {'Power Cepstrum','Complex Cepstrum'}, ...
        'Position', [700 550 150 30]);

    uicontrol('Style', 'pushbutton', 'String', 'Compute Cepstrum', ...
        'Position', [870 550 100 30], 'Callback', @computeCepstrumCallback);

    % Axes
    waveformAx = axes('Parent', f, 'Position', [0.07 0.6 0.4 0.3]);
    spectroAx = axes('Parent', f, 'Position', [0.57 0.6 0.4 0.3]);
    cepstrumAx = axes('Parent', f, 'Position', [0.07 0.1 0.9 0.35]);
```

3.2. UPLOADING A FILE

Next, the function that allows user to load a data file and displays the waveform was created. As mentioned before, the application accepts a mat file as input. The file has to contain two variables: actual signal (data) and the sampling frequency.

```
function loadAudioCallback(~, ~)
    [file, path] = uigetfile('*.mat', 'Select a MAT file');
    if isequal(file, 0)
        return;
    end
    data = load(fullfile(path, file)); % Load the .mat file
```

Because the signal is loaded as one big column, the matrix it's assigned to has to be transposed.

```
    if isfield(data, 'signal') && isfield(data, 'fs')
        audioData = data.signal;
        fs = data.fs;
    else
        msgbox('MAT file must contain "signal" and "fs" variables.');
        return;
    end

    if isrow(audioData)
        audioData = audioData';
    end
```

Next, the waveform is plotted in time domain and the signal spectrogram is displayed.

```
    % Plot waveform
    t = (0:length(audioData)-1)/fs;
    plot(waveformAx, t, audioData);
    title(waveformAx, 'Waveform'); xlabel(waveformAx, 'Time (s)');
    ylabel(waveformAx, 'Amplitude');

    % Plot spectrogram
    axes(spectroAx);
```

```
        spectrogram(audioData, hamming(256), 200, 512, fs, 'yaxis');
        title('Spectrogram');
    end
```

3.3. COMPUTING CEPSTRUM

First, the function checks if any input file is loaded.

```
function computeCepstrumCallback(~, ~)
        if isempty(audioData)
            msgbox('Please load an audio file first!');
            return;
        end
```

If the file is loaded, the chosen cepstrum is calculated. It should be noted that because some signal values are 0, taking logarithm of any of them it would result in an error in Matlab. This is why inside of abs() function call we also add *eps* which in Matlab is equal to 2e-52. This will prevent an error when taking abs() of zero value.

```
    y = audioData;
            N = length(y);
            q = (0:N-1)/fs;

            % Get cepstrum type
            cepstrumType = cepstrumTypePopup.String{cepstrumTypePopup.Value};
            switch cepstrumType
                case 'Power Cepstrum'
                    Y = fft(y);
                    ceps = abs(ifft(log(abs(Y).^2 + eps)));

                case 'Complex Cepstrum'
                    Y = fft(y);
                    ceps = real(ifft(log(Y + eps)));
            end
```

The function also allows for applying a lifter with selected parameters.

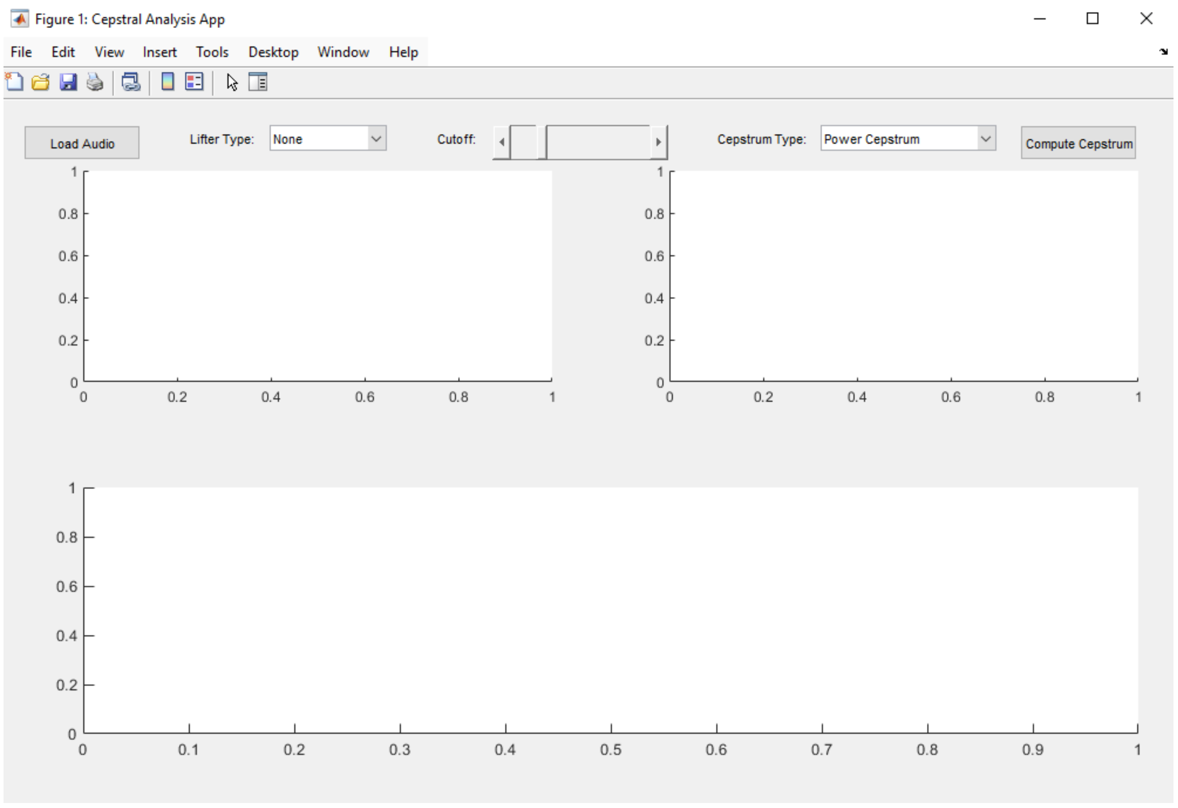
```
    % Get liftering options
            lifterTypeStr = lifterPopup.String{lifterPopup.Value};
            cutoff = round(cutoffSlider.Value);
            switch lifterTypeStr
                case 'Low-pass'
                    lifter = zeros(size(ceps));
                    lifter(1:cutoff) = 1;
                    ceps = ceps .* lifter;
                case 'High-pass'
                    lifter = ones(size(ceps));
                    lifter(1:cutoff) = 0;
                    ceps = ceps .* lifter;
            end
```

Finally, we can plot the cepstrum.

```
% Plot cepstrum
    plot(cepstrumAx, q, abs(ceps));
    title(cepstrumAx, ['Cepstrum - ', cepstrumType, ' | ', lifterTypeStr]);
    xlabel(cepstrumAx, 'Quefrency (s)');
    ylabel(cepstrumAx, 'Magnitude');
    xlim(cepstrumAx, [0 0.02]); % Zoom in to typical pitch range
```

4. RESULTS AND CONCLUSIONS

The final look of the application is displayed in the figure below.



Lifter Type: None

Cutoff:

Cepstrum Type: Power Cepstrum

Figure 1

For the purpose of testing, I have sourced two audio files. First is a 3 second sample of read human speech taken from open source database CMU arctic. I have used this sample to calculate the power cepstrum in order to detect a peak at the start of the sentence when vowel *'I'* is pronounced.

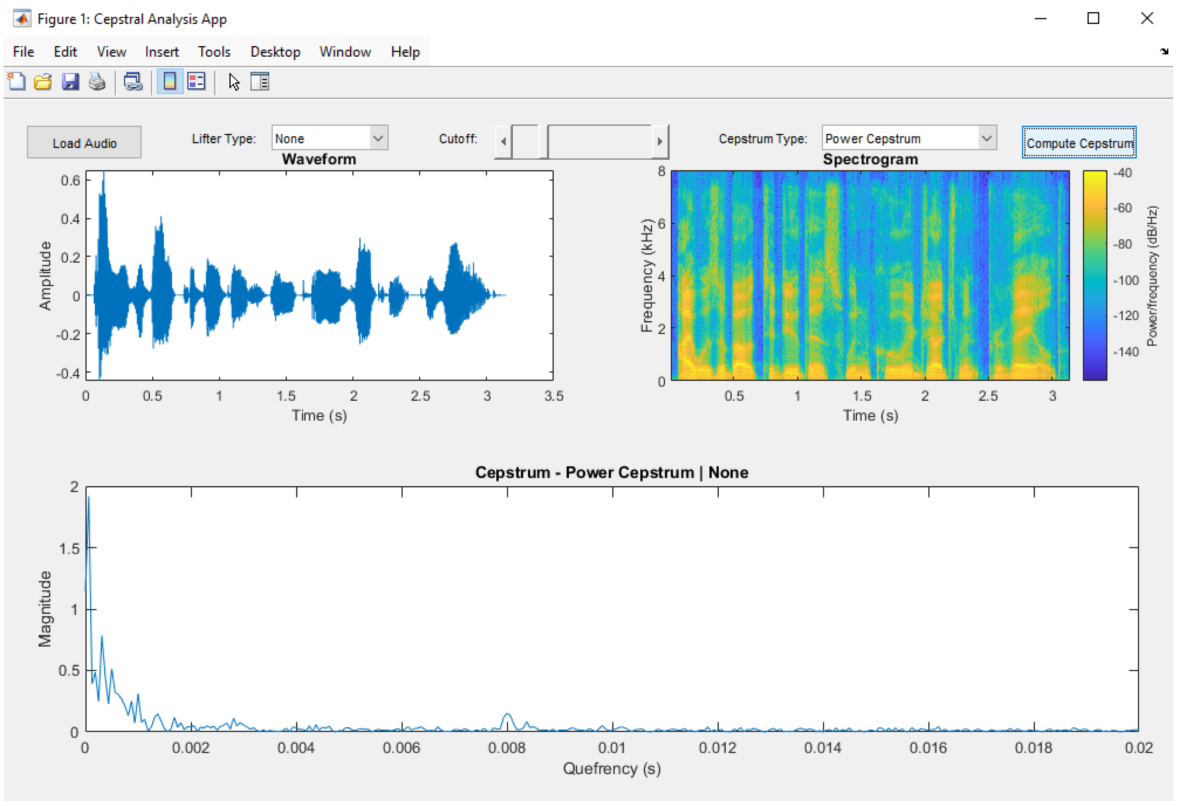


Figure 2

For additional testing and demonstration of complex cepstrum, I have sourced samples of gearbox monitoring data with both normal and faulty operations from Case Western Reserve University Bearing Data Center. In Figure 3, we can see the waveform and high pass cepstrum of a normally operating gearbox audio file. In Figure 4, the waveform and again high pass cepstrum of the faulty gearbox. As we can see, in the low quefrencies, the plots look similar, but in mid and high quefrencies in faulty gearbox we can notice more frequency compoments which indicate some sort of defect, for example repeated impacts.

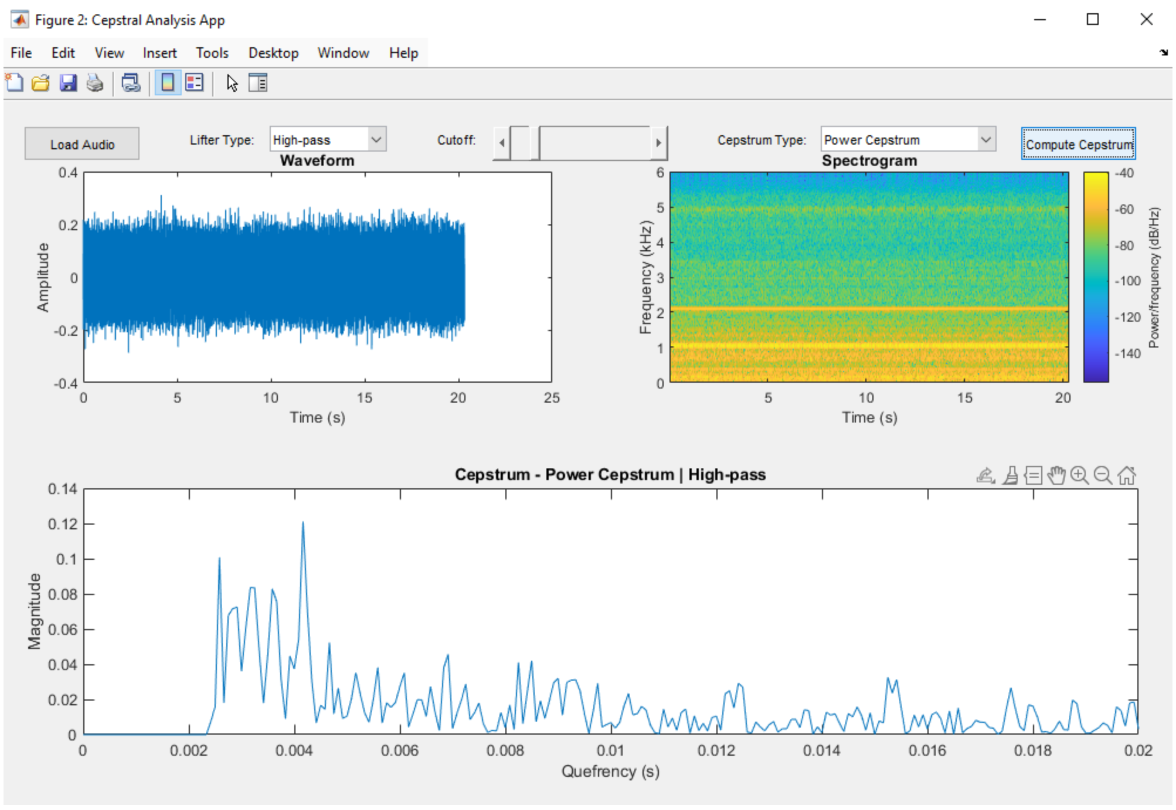


Figure 3

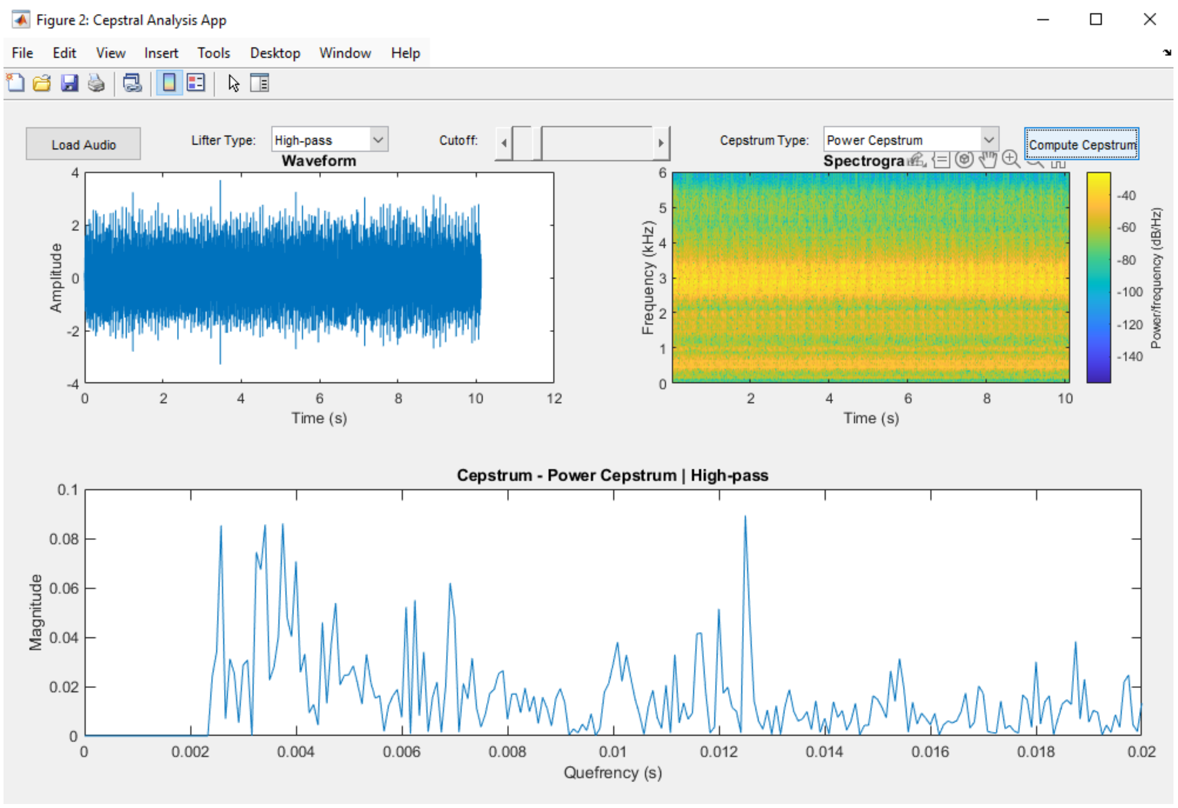


Figure 4

Next, for the purposes of testing the complex cepstrum, I have used a short sample of a female singing voice inside Nashville First Baptist Church from OpenAir dataset. I have chosen this file for complex spectrum because it's a good tool for analyzing reverberated speech samples – here, voice will echo from the church surfaces. We are expecting a noisier plot with a big number of peaks.

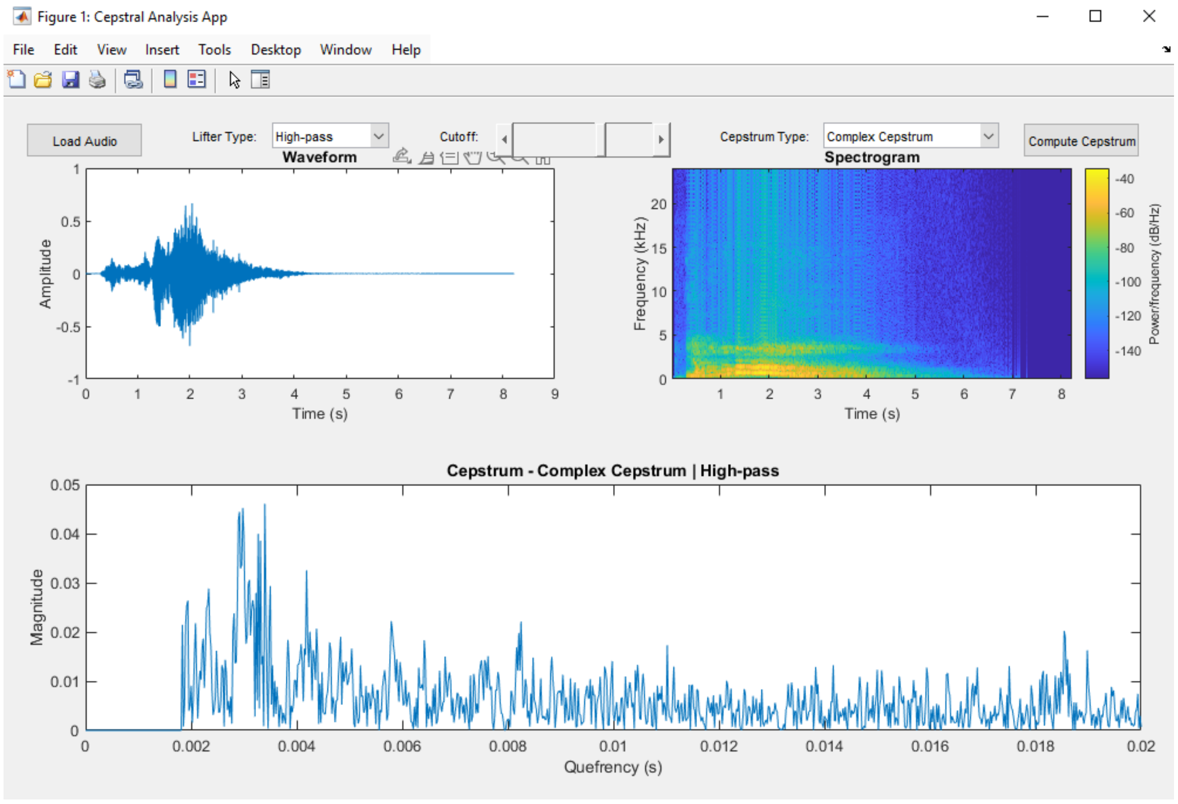


Figure 5

To conclude, the aim of the project was realized. The application allows the user to compute cepstrum for a desired signal with their own chosen configuration.

5. LITERATURE

Bogert, B. P., Healy, M. J. R., & Tukeyf, J. W. (1974). The Guefrency Alanysis of Time Series for. In *Proc. Symp. on Time Set. Anal., M. Rosenblatt, ed, Wiley NY*.

Oppenheim, A. V., & Schafer, R. W. (2004). From frequency to quefrency: A history of the cepstrum. *IEEE signal processing Magazine*, *21*(5), 95-106.

https://en.wikipedia.org/wiki/Cepstrum (date visited: 2025.06.08)
http://festvox.org/cmu_arctic/cmu_arctic/cmu_us_bdl_arctic/wav/
(date visited: 2025.06.08)
https://engineering.case.edu/bearingdatacenter/12k-drive-end-bearing-fault-data
(date visited: 2025:06:08)
https://www.openair.hosted.york.ac.uk/?page_id=406
(date visited: 2025.06.08)