Linear Predictive Speech Synthesizer

EEEM030 - Speech & Audio Processing & Recognition
Assignment 1

Xiaoguang Liang
6844178
xl01339@surrey.ac.uk

December 5, 2024

Contents

Abstract

This report mainly includes the task of modelling, analysing and synthesizing some vowels using the source-filter model of speech production. This project estimates the frequency response spectrums and the formant structure of each vowel directly from real vowel samples by employing a kind of autoregressive (AR) model - linear predictive coding (LPC) and then generates synthesized vowels by passing a periodic impulse train through the all-pole filter obtained.

1 Introduction

This report explores vowel speech by modelling, analysing, and synthesizing them using the source-filter model of speech production. In practice, this project chooses the speech samples - `had_f.wav` (one female vowel) and `had_m.wav` (one male vowel) from sample set.

The source-filter model of speech production is a widely used approach that separates the speech generation process into two parts: a source of sound (such as a periodic impulse train representing the vibration of vocal folds for voiced sounds) and a filter that represents the resonant properties of the vocal tract[1]. A diagram of this model is presented in Figure 1. This model enables the simulation of different speech sounds by manipulating the filter properties while keeping the source consistent.

To estimate the formant structure of each vowel, this project uses an autoregressive (AR) model, implemented through LPC. As a practical method for implementing the source-filter model, LPC is an effective technique in speech processing that models the speech signal by predicting each sample as a linear combination of previous

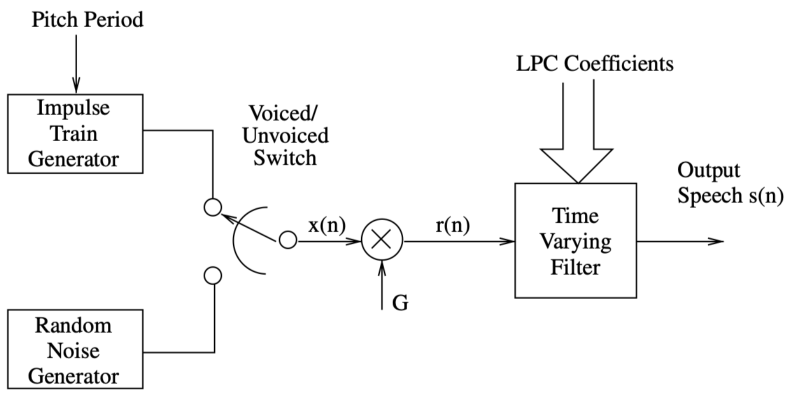


Figure 1: Block diagram of the simplified source-filter model of speech production[1].

samples[1]. This method helps estimate the resonant frequencies (formants) by approximating the vocal tract's response. After obtaining the LPC-based filter (all-pole filter), this project passes a periodic impulse train through it to synthesize each vowel sound. This approach allows us to generate vowels that mimic the natural acoustic properties of real speech samples.

The framework for implementing this project is illustrated in Figure 2. The details of the implementation are organized as following. Firstly, the report will introduce the model estimation for LPC in *Section 2*. Secondly, the speech synthesis progress will be elaborated in *Section 3*. After that, the report introduces the experiments with different AR model orders and segment lengths in *Section 4*. Next, typical conclusions will be figured out respectively in *Section 5*. The implementation of model estimation and synthesis is completed with Matlab and the Matlab codes will be included in the appendix.

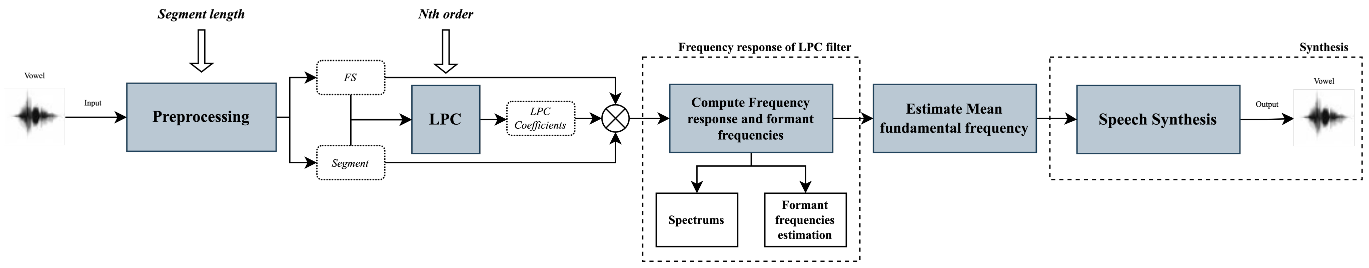

Segment length
Nth order
Vowel
Input
Preprocessing
FS
Segment
LPC
LPC Coefficients
Frequency response of LPC filter
Compute Frequency response and formant frequencies
Spectrums
Formant frequencies estimation
Estimate Mean fundamental frequency
Synthesis
Speech Synthesis
Output
Vowel

Figure 2: The framework of implementation of model estimation and synthesis.

2 Model Estimation

2.1 Preprocessing audio files

The audio files preprocessing part applies Matlab's `audioread` function to read the vowel audio file, which returns the sampled data in Y and the sample rate Fs, in $Hertz$. Then, based on the specified segment duration and sampling rate, calculate the number of samples needed to extract.

Here, the sample start point is set to 1; the segment length is set to 0.1, as specified by the assignment, to obtain a quasi-stationary segment of approximately $100ms$ in duration. In *Section 4*, the effects of different segment lengths on model estimation and the generated audio will be discussed.

2.2 Estimate the LPC coefficients

This section directly uses Matlab's `lpc` function to calculate the LPC coefficients. The input parameter for LPC order is set to 25, and experiments with different order values will be conducted and discussed in the Experiments section.

Since the dynamic range of actual speech signals is large, directly quantizing these signals would require a large amount of bits, leading to a high coding rate. To reduce the coding rate while maintaining good speech quality, we can try to decrease the dynamic range of the input signal to the encoder. LPC predicts new sample values based on past samples, and then calculates an error signal by subtracting the predicted value from the actual sample value. The dynamic range of this error signal is significantly smaller than that of the original speech signal. By quantizing and encoding the error signal, the required bit rate is greatly reduced, thus lowering the coding rate[1].

The mathematical representation of the LPC filter is as follows[1]:

$$H(z) = \frac{1}{(1 + \sum_{i=1}^{p} \alpha_i z^{-i})} \tag{1}$$

According to the principles of linear predictive analysis, to solve for the linear predictive coefficients, the mean square prediction error of the speech signal must be minimized. There are two classic methods for this: the autocorrelation method and the covariance method. Since Matlab's `lpc` function uses the autocorrelation method, only this method is discussed here.

The LPC coefficients are determined by solving a set of linear equations called the Yule-Walker equations[2]:

$$\begin{bmatrix} R_n(0) & R_n(1) & \dots & R_n(p-1) \\ R_n(1) & R_n(0) & \dots & R_n(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_n(p-1) & R_n(p-2) & \dots & R_n(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \vdots \\ R_n(p) \end{bmatrix}$$

here $R_n(j)$ represents the short-time autocorrelation function[2]:

$$R_n(j) = \sum_{m=0}^{N-1-j} s_n[m] s_n[m+j]; \quad j = 0, 1, \dots, p. \tag{2}$$

2.3 Frequency response of the LPC filter

2.3.1 Compute frequency response and formant frequencies

In this section, this project begin by calculating the frequency response of the LPC filter, followed by the computation of the formant frequencies and the N-point DFT of the original segment. These calculation results will then be used to plot the signal in the frequency domain and to compare and analyse the frequency response of the LPC filter with the original signal.

First, the project uses the `nextpow2` function to compute the number of frequency points. This value controls the resolution between 0 and the Nyquist frequency (half the sampling rate), where a higher provides finer frequency resolution, and lower results in a coarser response. The `nextpow2` function is particularly useful for optimizing efficiency in Fast Fourier Transform (FFT) computations, as it ensures is a power of 2.

Next, the freqz function is used to calculate the N-point complex frequency response (response) and the N-point frequency vector W. In this context, `freqz` takes four inputs: 1 (numerator coefficients), $lpcCoeffs$ (LPC coefficients as denominator coefficients), N (number of frequency points), and Fs (sample rate). The 1 is numerator coefficients and $lpcCoeffs$ is the denominator coefficients in the equation used in `freqz` function. The equation of `freqz` function is like this[3]:

$$H(e)^{jw} = \frac{B(e)^{jw}}{A(e)^{jw}} = \frac{b(1) + b(2)e^{-jw} + \cdots + b(m+1)e^{-jmw}}{a(1) + a(2)e^{-jw} + \cdots + a(n+1)e^{-jnw}} \tag{3}$$

As shown in the equation, the given numerator 1 and denominator coefficients $lpcCoeffs$ are in vectors B and A respectively.

Following this, the `findpeaks` function is applied to locate local peaks in the N-point complex frequency response. These peaks help identify formant frequencies, which are extracted from the the N-point frequency vector W. The formant frequencies will be used to analyse the frequency response of LPC and the estimate the first three formant frequencies.

Finally, the `fft` function is used to compute the N-point discrete Fourier transform (DFT) of the original segment vector, then calculate the frequency vector of original segment. These results provide a basis for spectral analysis of the original segment and the frequency response of LPC .

2.3.2 LPC response and speech amplitude spectrum

In this Section, this project plots the amplitude spectrum of the original segement and the LPC filter frequency response spectrum. To make it easier to compare the spectrums, this project puts the curves on one graph and zoom in the graph to easily observe the spectrums with the function `xlim`.

Figure 3 and Figure 4 are the spectrums of `had_f.wav` (one female vowel) and `had_m.wav` (one male vowel) repectively. According to both of the graphs, it is clear that LPC frequecy response can provide a very good fit to the spectral peaks but is worse fitting to the spectral valleys. That's because this project is using an all-pole filter model to model the spectrum, but don't use any zero-pole to model the spectral valleys[2].

2.4 Estimate first three formant frequencies

The first formant frequencies on the female and male vowel segments are shown in Table 1 and they are also plotted on 3 and Figure 4. All the data of the formant frequencies is produced under the conditions of segment length is $100ms$ and the order of LPC is 25.

Table 1: First three formant frequencies for female and male

Symbol for vowel	Typical word	Audio file	F1 (Hz)	F2 (Hz)	F1 (Hz)
AE	had	`had_f.wav`	251.95	972.66	1936.52
AE	had	`had_m.wav`	706.05	1754.88	2504.88

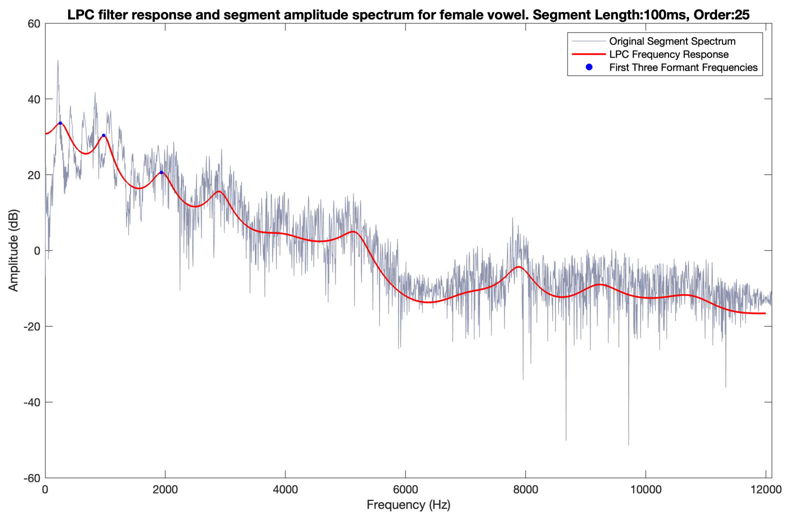


Figure 3: LPC response and segment amplitude spectrum for female vowel.

2.5 Estimate mean fundamental frequency

The fundamental frequency, commonly called the fundamental and abbreviated as $f0$ or $f1$, is the lowest frequency in a periodic waveform. It typically appears as the first vertical line in the frequency spectrum and represents the first harmonic.

In this section, the project first calculates the auto-correlation sequence using the cross-correlation function `xcorr`, then identifies peaks within the expected lag range using the `findpeaks` function. Following this, the project computes the lag values for both female and male vowels corresponding to their respective $F0$ ranges, because adult males typically have a fundamental frequency between 90 and $155Hz$, while adult females range from 165 to $255Hz$[4].

The results show in the talbe 2.

Table 2: Mean fundamental frequency for female and male

Audio file	Segment length	Mean fundamental frequency (Hz)
`had_f.wav`	100ms	201.89
`had_m.wav`	100ms	117.22

3 Synthesis

In this section, this project generates some synthesis audios of about one second in length by reconstructing the speech signal based on the LPC coefficients. Because the vowel segment is the voiced speech, this project needs to compute the simulated periodic impulse train. The simulated periodic impulse train is the excitation signals which can correspond to the vocal cord. Then this project convolves these excitation signals with LPC coefficients and then passes through the all-pole filter using `filter` function to get the synthesized signal output[2]. This progress is shown in the diagram Figure 5. Finally, by normalizing the synthesis signal, this project can improve the clarity

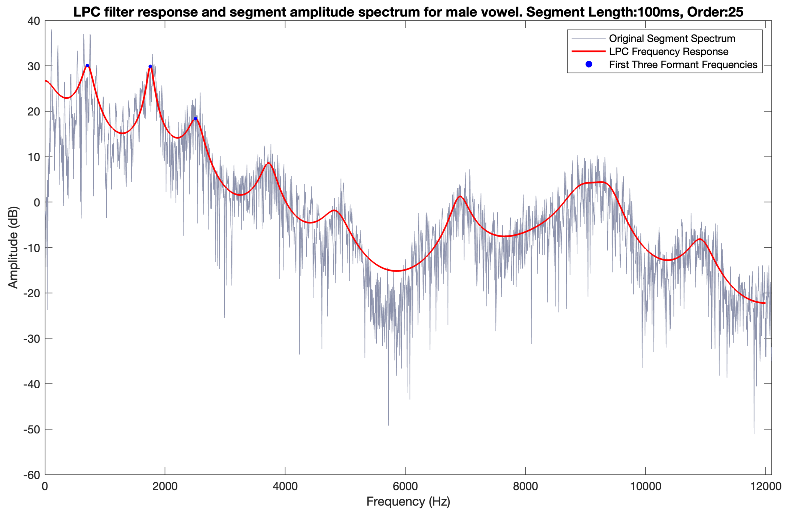


Figure 4: LPC response and segment amplitude spectrum for male vowel.

of the synthesis signal and make the output easier to listen to.

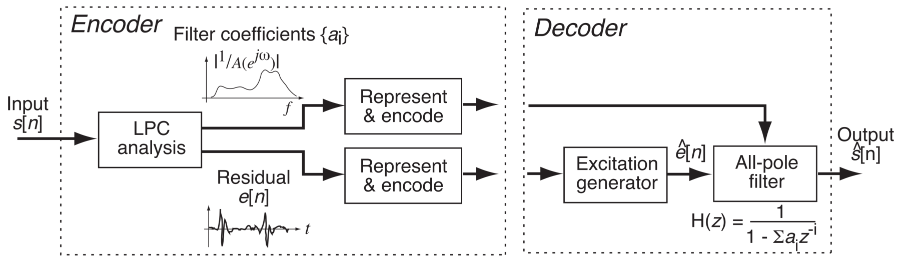


Figure 5: Re-synthesize original speech with LPC coefficients[5].

4 Experiments

4.1 Experimental design

This experiment uses the variable-controlling approach to examine the impact of different order values and segment lengths on the LPC frequency response and the synthesis speech quality.

Segment length. In the experiments, this project determines that the sample rate of the segment length should be greater than 0 and less than 2.5. Therefore, the sample rate of the segment length is set to range from 0.05 to 0.2, with a step size of 0.05.

Order. In practice, LPC filter modelling is typically used to represent the formant structure, making the spectral details from higher orders unnecessary. An LPC order of around 20-30 is generally sufficient to model the first few formants in speech[2]. Therefore, the LPC order value ranges from 15 to 35, with a step size of 5.

In the code implementation, to improve efficiency, all combinations of order values and segment lengths are firstly iterated over, with the results stored in a struct data structure, which includes all generated audio outputs. Then, using the data in the struct, spectrum plots are generated in batches for combinations with the same order and different segment lengths, as well as those with the same segment length and different order values, as shown in Figure 6.

It should be noted that different segment lengths produce different original segment spectrums. If all original segment spectrums were plotted in the figure, it would make the chart difficult to interpret clearly. Therefore, in the case of the same order with different segment length combinations, the original segment spectrums are not plotted in the figure. Due to space constraints, Figure 6 only displays 8 of the images from the experiments.

4.2 Comapre the spectrums

By observing all the graphs in Figure 6, we can obtain the following inferences based on this experiment:

- For the female vowel, with a fixed segment length, increasing LPC order allows LPC frequency response to more accurately align with the spectral peaks. This pattern is similarly observed for the male vowel.
- For experiments with the same order but different segment lengths, it is difficult to compare how LPC frequency response fits the original segment spectrum across different segment lengths, as it is hard to distinguish when plotting all the original segment spectra on the same graph. However, we can compare the general trends between the different curves. For the same order with different segment lengths, the frequency response shows little difference in the low-frequency region, but significant differences in the high-frequency region. For different orders, the higher the order value, the greater the differences between the frequency response curves.
- For the female vowel, with the same LPC order but different segment lengths, a longer segment enables LPC frequency response to more accurately match the spectral peaks.

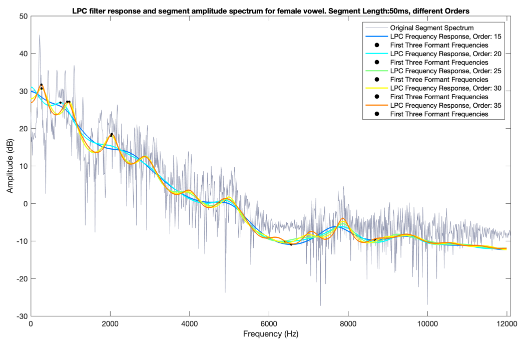
4.3 Informal subjective assessment for speech synthesis

The following provides an informal subjective assessment of the generated vowels compared to the original vowels, focusing on three specific aspects.

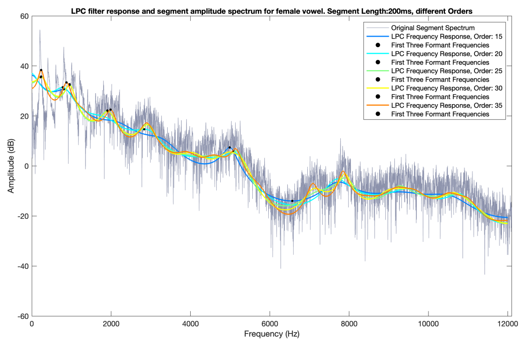
Pitch. Due to the short duration of the original audio, from a subjective listening perspective, the pitch of all the generated audios are not significantly different from the original audios, whether for male or female audio.

Comparison of the generated audios under different order values and segment length values:

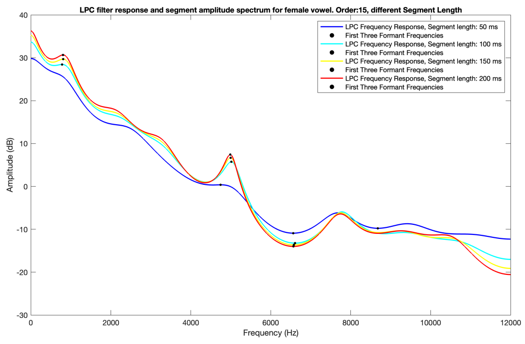
- Under the condition of the same order value but different segment lengths, the pitch increases noticeably as the segment length value increases.
- Under the condition of the same segment length but different order values, the pitch increases slightly as the order value increases, but the change is not very noticeable.



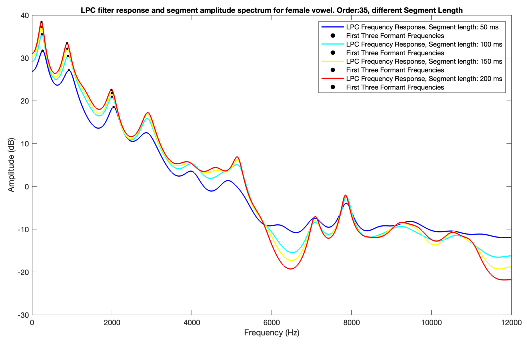
(a) Spectrum for female, segment length:50ms.



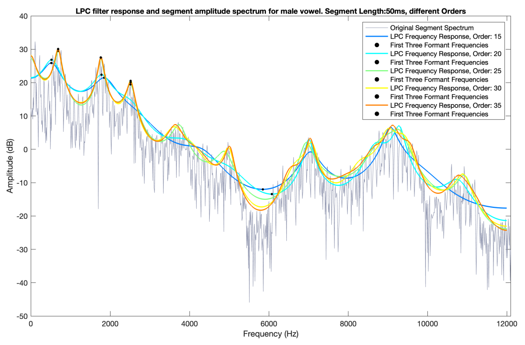
(b) Spectrum for female, segment length:200ms.



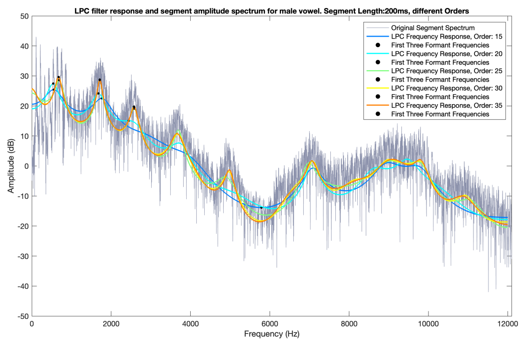
(c) Spectrum for female, order:15.



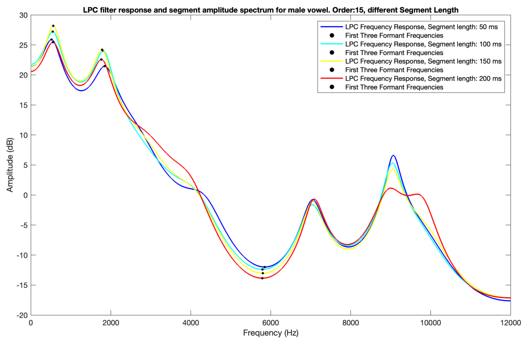
(d) Spectrum for female, order:35.



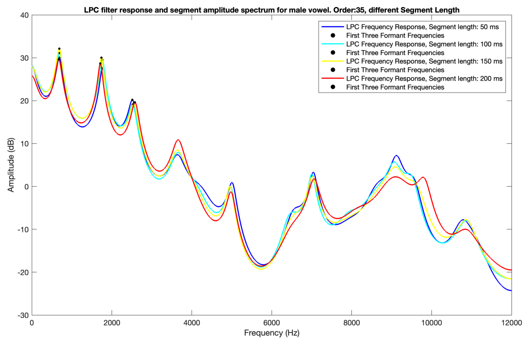
(e) Spectrum for male, segment length:50ms.



(f) Spectrum for male, segment length:200ms.



(g) Spectrum for male, order:15.



(h) Spectrum for male, order:35.

Figure 6: Spectrums for experiments with different orders values and segment lengths.

Timbre. Overall, the timbre of all the generated audios differs significantly from the original audios. However, in the generated audios, the timbre of the male and female speech differs significantly, making them easy to distinguish.

It's probable that the synthesized speech lacked the richness and depth of natural human voices. Some of the higher-order harmonics were either missing or not accurately represented, causing the voice to sound somewhat flat and nasal. LPC filter did a reasonable job of modelling the basic shape of the spectrum, but it struggled with the finer details that give voices their unique character. This resulted in speech that was intelligible but not particularly lifelike.

Comparison of the generated audios under different order values and segment length values:

- Under the condition of the same order value but different segment lengths, the timbre shows slight differences, but they are not significant.
- Under the condition of the same segment length but different order values, there is almost no difference in timbre.

Loudness. The generated audio is generally louder than the original audio. the loudness levels of the synthesized speech are consistent but do not fully match the dynamic range of the original audio. The output audios sometimes sound either too soft or unnaturally loud, depending on the segment.

Comparison of the generated audios under different order values and segment length values:

- The loudness of all the audios is almost the same, indicating that the order value and segment length have little effect on the loudness.

5 Conclusion

This report demonstrates how to create a source-filter model using an LPC filter to model, synthesize, analyse, and evaluate speech signals. In this project, two parameters need to be adjusted: LPC order and segment length. Variations in these two parameters ultimately lead to different results in both the model analysis and speech synthesis stages. Through experiments, a relatively suitable parameter combination can be identified, resulting in a series of synthesized vowel signals. However, based on informal subjective evaluation, it was found that due to the limited performance of the LPC filter, the final synthesized vowel speech has some noticeable differences from the original audio. In the future, more autoregressive (AR) models and synthesis methods can be explored for vowel speech synthesis.

References

[1] A.M. Kondoz. *Digital Speech: Coding for Low Bit Rate Communication Systems*, volume 4-5, pages 65–66. Wiley, 2005.

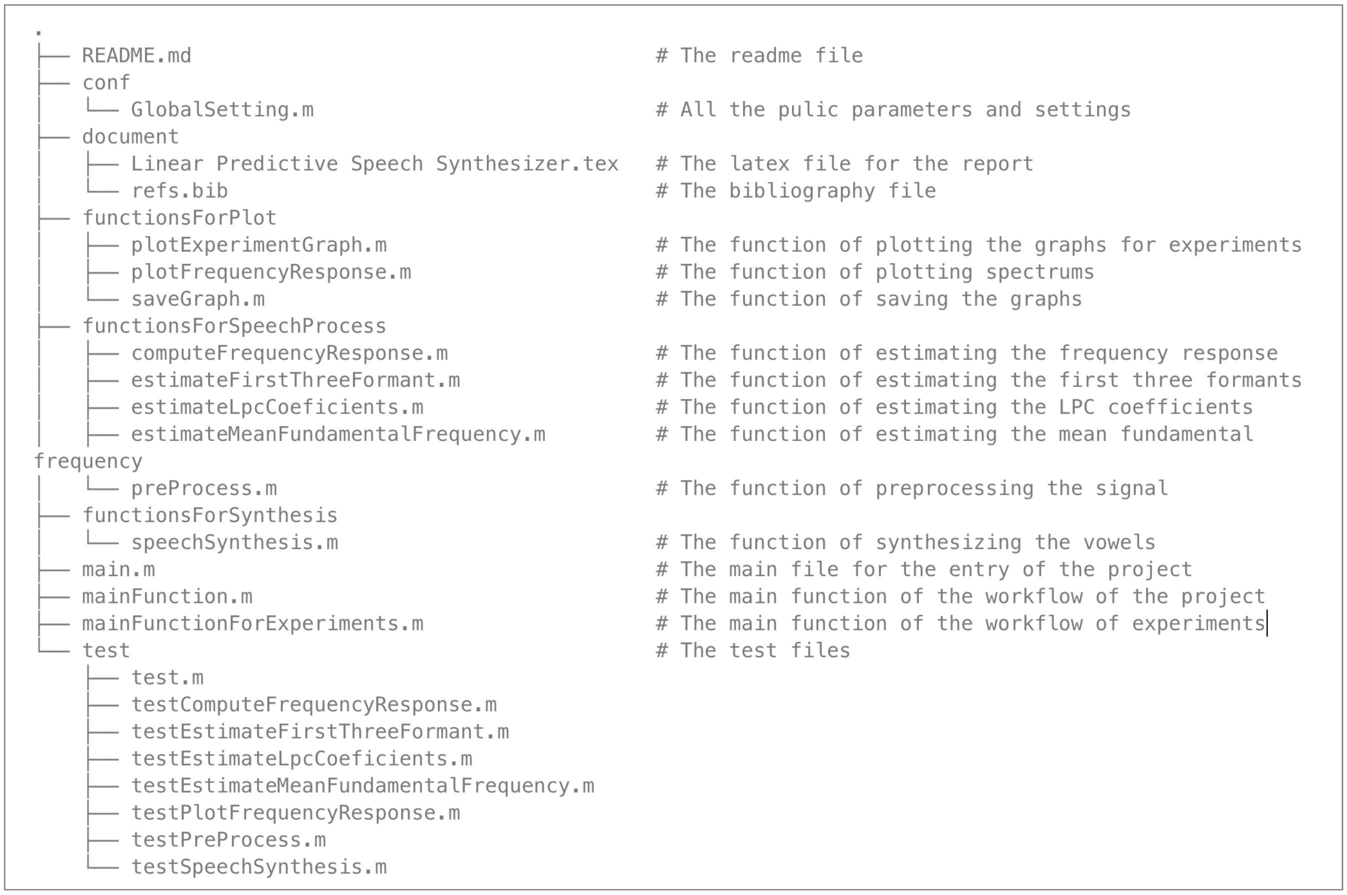
[2] Wenwu Wang. Speech and audio processing and recognition: speech and audio processing (part 3). [Lecture], 2024.

[3] The MathWorks Inc. Matlab version: 24.2.0.2712019 (r2024b), 2024.

[4] R.J. Baken and R.F. Orlikoff. *Clinical Measurement of Speech and Voice*, page 381. Speech Science. Singular Thomson Learning, 2000.

[5] Dan Ellis. Linear prediction (lpc). [Lecture], October 2013.

A Appendix: Structure of the codes



```
.
├── README.md                                # The readme file
├── conf
│   └── GlobalSetting.m                      # All the pulic parameters and settings
├── document
│   ├── Linear Predictive Speech Synthesizer.tex   # The latex file for the report
│   └── refs.bib                             # The bibliography file
├── functionsForPlot
│   ├── plotExperimentGraph.m                # The function of plotting the graphs for experiments
│   ├── plotFrequencyResponse.m              # The function of plotting spectrums
│   └── saveGraph.m                          # The function of saving the graphs
├── functionsForSpeechProcess
│   ├── computeFrequencyResponse.m           # The function of estimating the frequency response
│   ├── estimateFirstThreeFormant.m          # The function of estimating the first three formants
│   ├── estimateLpcCoeficients.m             # The function of estimating the LPC coefficients
│   ├── estimateMeanFundamentalFrequency.m   # The function of estimating the mean fundamental
frequency
│   └── preProcess.m                         # The function of preprocessing the signal
├── functionsForSynthesis
│   └── speechSynthesis.m                    # The function of synthesizing the vowels
├── main.m                                   # The main file for the entry of the project
├── mainFunction.m                           # The main function of the workflow of the project
├── mainFunctionForExperiments.m             # The main function of the workflow of experiments
└── test                                     # The test files
    ├── test.m
    ├── testComputeFrequencyResponse.m
    ├── testEstimateFirstThreeFormant.m
    ├── testEstimateLpcCoeficients.m
    ├── testEstimateMeanFundamentalFrequency.m
    ├── testPlotFrequencyResponse.m
    ├── testPreProcess.m
    └── testSpeechSynthesis.m
```

Figure 7: Structure of the codes.

B Appendix: Codes

Listing 1: main.m

```matlab
%% Author: Xiaoguang Liang (PG/T - Comp Sci & Elec Eng)
%% University of Surrey, United Kingdom
%% Email address: xl01339@surrey.ac.uk
%% Time: 29/10/2024 21:16

clc;
close all;
clear;

% Female vowel phoneme sample, had_f.wav
femaleFile=GlobalSetting.femaleFile;
% Male vowel phoneme sample, had_m.wav
```

```
maleFile=GlobalSetting.maleFile;

% Create a struct for sample files of the female or male
fileStruct = struct('female', femaleFile, 'male', maleFile);

% Set the Experiment Mode. 1: run experiment mode for experiments; 0: run default
    mode for mainFunction with default parameters
ExperimentMode = 1;

% Create a struct for the output arguments
experimentArgs = struct('gender', {}, 'segmentLen', {}, 'NthOrder', {}, 'H', {}, 'W
    ', {}, 'formantFrequencies', {}, 'Y', {}, 'frequencyVector', {});
if ExperimentMode == 0
    fprintf('>>> Run default mode \n');
    %% 1 Model Estimation & 2 Speech Synthesis
    % default length of segment: 0.1, 100ms
    segmentLen = GlobalSetting.segmentLen;
    % Default Order of the LPC filter, an Nth order forward linear predictor
    NthOrder = GlobalSetting.NthOrder;

    % Start processing the vowels
    mainFunction(fileStruct,segmentLen,NthOrder,ExperimentMode,experimentArgs);
else
    fprintf('>>> Run experiment mode \n');
    %% 3 Experiment with different AR model orders and segment lengths. Notice:
        Part 1 and 2 are included in the mainFunction()

    % The lengths of segment range from 50ms to 200ms
    segmentLenArray=0.05:0.05:0.2;
    % segmentLenArray=0.05:0.05:0.1;
    % Orders of the LPC filter, an Nth order forward linear predictor, range from
        15 to 35
    NthOrdersArray=15:5:35;
    % NthOrdersArray=15:5:20;

    % Start experiments for processing the vowels
    mainFunctionForExperiments(fileStruct,segmentLenArray,NthOrdersArray,
        ExperimentMode,experimentArgs);
end
```

Listing 2: mainFunction.m

```
 function experimentArgs = mainFunction(fileStruct, segmentLen, NthOrder,
    ExperimentMode, experimentArgs)
 % MAINFUNCTION Summary of this function goes here
 %
```

```matlab
% [OUTPUTARGS] = MAINFUNCTION(INPUTARGS) This function processes vowel files to
    estimate LPC coefficients, formant frequencies, and the mean fundamental
    frequency.
% It also synthesizes speech based on the estimated parameters and saves the
    results.
%
% Examples:
% fileStruct = struct('male', 'male_vowel.wav', 'female', 'female_vowel.wav');
% segmentLen = 0.1; % Segment length in seconds
% NthOrder = 10; % Order of LPC coefficients
% ExperimentMode = 0; % 0 for plotting, 1 for saving results
% experimentArgs = [];
% mainFunction(fileStruct, segmentLen, NthOrder, ExperimentMode, experimentArgs);
%
% See also: preProcess, estimateLpcCoeficients, computeFrequencyResponse,
    estimateFirstThreeFormant, plotFrequencyResponse,
    estimateMeanFundamentalFrequency, speechSynthesis

% Author: Xiaoguang Liang, University of Surrey
% Date: 2024/11/02 16:42:01
% Revision: 0.1

%% 1 Model Estimation
% Get the field names of the struct
fields = fieldnames(fileStruct);

%% Iterate over the files to process the vowel files
for i = 1:length(fields)
    gender = fields{i};
    file = fileStruct.(gender);
    fprintf('>>> Start processing %s vowel file: %s\n', gender, file);

    %% 1.1 Pre-processing audio files
    % Choose the start point in the segment
    sampleStart = 1;
    fprintf('1.1 Pre-processing audio files\n');
    fprintf('Segment start point: %d; Segment length: %f\n', sampleStart,
        segmentLen);
    [~, Fs, segment] = preProcess(file, sampleStart, segmentLen);

    %% 1.2 Estimate the LPC coefficients
    fprintf('1.2 Estimate the LPC coefficients\n');
    fprintf('The Nth order of LPC is: %d\n', NthOrder);
    lpcCoeffs = estimateLpcCoeficients(segment, NthOrder);

    %% 1.3 Plot the frequency response of the LPC filter
    % 1.3.1 Compute frequency response and formant frequencies
    fprintf('1.3.1 Compute frequency response and formant frequencies\n');
    [H, W, formantFrequencies, Y, frequencyVector] = computeFrequencyResponse(
        lpcCoeffs, segment, Fs);

    % Get the first three formant frequencies of the vowel and plot them
    formantFrequencies = estimateFirstThreeFormant(formantFrequencies);

    % Save the output arguments
    if ExperimentMode == 1
        experimentArgs(end+1).gender = gender;
        experimentArgs(end).segmentLen = segmentLen;
        experimentArgs(end).NthOrder = NthOrder;
        experimentArgs(end).H = H;
        experimentArgs(end).W = W;
        experimentArgs(end).formantFrequencies = formantFrequencies;
        experimentArgs(end).Y = Y;
        experimentArgs(end).frequencyVector = frequencyVector;
    end

    strSegmentLen = num2str(segmentLen * 1000);
    strNthOrder = num2str(NthOrder);
    % Only on experiment mode, plot the frequency response of the LPC filter
    if ExperimentMode == 0
        % 1.3.2 Plot the frequency response of the LPC filter
        fprintf("1.3.2 Plot the frequency response of the LPC filter\n");
        plotFrequencyResponse(H, W, formantFrequencies, Y, frequencyVector,
            strSegmentLen, strNthOrder, gender);
    end

    %% 1.4 Estimate the first three formant frequencies of the vowel
    fprintf("1.4 Estimate the first three formant frequencies of the vowel\n");
    firstThreeFormants = formantFrequencies;

    %% 1.5 Estimate the mean fundamental frequency.
    fprintf("1.5 Estimate the mean fundamental frequency\n");
    meanFundamentalFrequency = estimateMeanFundamentalFrequency(segment, Fs,
        gender);
    fprintf('The mean fundamental frequency is: %d\n', meanFundamentalFrequency);

    %% 2 Synthesis
    %% 2.1 Generate a periodic impulse train with the same fundamental frequency
    fprintf("2.1 Generate a periodic impulse train with the same fundamental
        frequency\n");
    % Make the save file for the synthesis audios
    if strcmp(gender, 'female')
        saveFile = ['synthesized_speech_female_when_segmentLength_',
            strSegmentLen, '_Order_', strNthOrder, '.wav'];
    else
        saveFile = ['synthesized_speech_male_when_segmentLength_', strSegmentLen,
            '_Order_', strNthOrder, '.wav'];
    end
    % Generate impulse train with the same fundamental frequency as the original
        vowel segment
    speechSynthesis(Fs, meanFundamentalFrequency, lpcCoeffs, saveFile);
end

end
```

Listing 3: mainFunctionForExperiments.m

```
function mainFunctionForExperiments(fileStruct, segmentLenArray, NthOrdersArray,
    ExperimentMode, experimentArgs)
% MAINFUNCTIONFOREXPERIMENTS Summary of this function goes here
%
% [OUTPUTARGS] = MAINFUNCTIONFOREXPERIMENTS(INPUTARGS) This function runs
    multiple experiments by looping over different segment lengths and LPC orders.
% It calls the `mainFunction` for each combination of segment length and LPC
    order, and collects the results.
% Finally, it plots all the experimental results.
%
% Examples:
% fileStruct = struct('male', 'male_vowel.wav', 'female', 'female_vowel.wav');
% segmentLenArray = [0.1, 0.2]; % Array of segment lengths in seconds
% NthOrdersArray = [10, 15]; % Array of LPC orders
% ExperimentMode = 0; % 0 for plotting, 1 for saving results
% experimentArgs = [];
% mainFunctionForExperiments(fileStruct, segmentLenArray, NthOrdersArray,
    ExperimentMode, experimentArgs);
%
% See also: mainFunction, plotExperimentGraph

% Author: Xiaoguang Liang, University of Surrey
% Date: 2024/11/06 09:45:15
% Revision: 0.1

% Loop over the segment lengths and orders, and run the mainFunction to plot
    graphs and generate the synthesis audios
for i = 1:length(segmentLenArray)
    segmentLen = segmentLenArray(i);
```

```matlab
        for j = 1:length(NthOrdersArray)
            NthOrder = NthOrdersArray(j);
            fprintf('>>> Experiment when Segment Length: %f and Order: %d\n',
                segmentLen, NthOrder);
            experimentArgs = mainFunction(fileStruct, segmentLen, NthOrder,
                ExperimentMode, experimentArgs);
        end
    end

    % Plot all the graphs
    plotExperimentGraph(experimentArgs);

    end
```

Listing 4: preProcess.m

```matlab
function [y,Fs,segment]=preProcess(filepath,sampleStart,segmentLen)
% preProcess Preprocesses an audio file to extract a specified duration segment
%
% [y,Fs,segment] = preProcess(filepath, sampleStart, segmentLen) Reads an audio
    file and extracts a segment of specified duration starting from a given sample
     point.
%
% Parameters:
%   - filepath: String, path to the audio file
%   - sampleStart: Integer, starting sample point for the segment extraction
%   - segmentLen: Float, duration of the segment to extract (in seconds)
%
% Returns:
%   - y: Array, original audio signal
%   - Fs: Integer, sampling rate of the audio
%   - segment: Array, extracted segment of the specified duration
%
% Examples:
%   [y,Fs,segment] = preProcess('vowel.wav', 10, 0.1);
%   Extracts a 0.1-second segment from 'vowel.wav' starting at the 10th sample
    point.
%
% See also: audioread

% Author: Xiaoguang Liang, University of Surrey
% Date: 2024/10/30 11:10:20
% Revision: 0.1

% Read the audio file
[y,Fs]=audioread(filepath);
```

```

% Calculate the number of samples for the segment
segmentSpan = round(segmentLen * Fs);

% Extract the specified duration segment from the speech signal
segment = y(sampleStart:sampleStart+segmentSpan);

end
```

Listing 5: estimateLpcCoeficients.m

```
function lpcCoeffs = estimateLpcCoeficients(segment, NthOrder)
% ESTIMATELPCCOEFICIENTS Summary of this function goes here
%
% [OUTPUTARGS] = ESTIMATELPCCOEFICIENTS(INPUTARGS) This function estimates the
    Linear Predictive Coding (LPC) coefficients for a given speech segment.
% It uses the `lpc` function to compute the coefficients and ignores the
    prediction error.
%
% Examples:
% segment = [0.1, 0.2, 0.3, 0.4, 0.5];
% NthOrder = 4;
% lpcCoeffs = estimateLpcCoeficients(segment, NthOrder);
%
% See also: computeFrequencyResponse, plotFrequencyResponse

% Author: Xiaoguang Liang, University of Surrey
% Date: 2024/10/31 21:57:11
% Revision: 0.1

% Using LPC to estimate formant frequencies in both speech

% The LPC coefficients are stored in the first variable in the vector.
% For now, I am ignoring the second variable which is the prediction error.
[lpcCoeffs, ~] = lpc(segment, NthOrder);

end
```

Listing 6: computeFrequencyResponse.m

```
function [H, W, formantFrequencies, Y, frequencyVector] =
    computeFrequencyResponse(lpcCoeffs, segment, Fs)
% COMPUTEFREQUENCYRESPONSE Summary of this function goes here
%
% [OUTPUTARGS] = COMPUTEFREQUENCYRESPONSE(INPUTARGS) This function computes the
    frequency response of an LPC filter and the FFT of a given segment.
% It also identifies the formant frequencies from the LPC spectrum.
```

```matlab
%
% Examples:
% [H, W, formantFrequencies, Y, frequencyVector] = computeFrequencyResponse(
    lpcCoeffs, segment, 16000);
%
% See also: plotFrequencyResponse

% Author: Xiaoguang Liang, University of Surrey
% Date: 2024/10/31 23:09:35
% Revision: 0.1

%% Compute frequency response for LPC filter
% Using nextpow2 to calculate the number of frequency points
N = 2^nextpow2(length(segment));

% Compute the N-point complex frequency response of the LPC filter
% Got the N-point complex frequency response: H, and the N-point frequency vector
    : W
[H, W] = freqz(1, lpcCoeffs, N, Fs);
% Converts the magnitude to a decibel (dB) scale
H = 20 * log10(abs(H));

%% Find peaks in the LPC spectrum that correspond to formants
[~, LOCS] = findpeaks(abs(H));
% Get formant frequencies
formantFrequencies = W(LOCS);

%% Compute the FFT of the signal, returns the n-point DFT
Y = fft(segment, N);
% Converts the magnitude to a decibel (dB) scale
Y = 20 * log10(abs(Y));

% Compute frequency Vector for the FFT Result
Ylen = length(Y);
% Make the scales from 0 Hz up to the sampling frequency Fs and provides the
    frequency for each point in Y.
frequencyVector = (0:Ylen-1) * Fs / Ylen;

end
```

Listing 7: plotFrequencyResponse.m

```matlab
function plotFrequencyResponse(H, W, formantFrequencies, Y, frequencyVector,
    strSegmentLen, strNthOrder, gender)
% PLOTFREQUENCYRESPONSE Summary of this function goes here
%
```

```
% [OUTPUTARGS] = PLOTFREQUENCYRESPONSE(INPUTARGS) This function plots the LPC
    frequency response and the amplitude spectrum of a given segment.
% It also marks the formant frequencies on the plot.
%
% Examples:
% plotFrequencyResponse(rand(1, 100), 0:0.01:1, [1000, 2000, 3000], rand(1, 100),
     0:0.01:1, '20', '10', 'male');
%
% See also: saveGraph

% Author: Xiaoguang Liang, University of Surrey
% Date: 2024/11/02 15:23:18
% Revision: 0.1

% Adjust the axes position for margins
figure('Position', [100 100 800 500], 'Visible', 'off');

% Get current axes
ax = gca;
% Position: [left, bottom, width, height]
ax.Position = [0.1, 0.1, 0.8, 0.8];

% Set the paper position mode
set(gcf, 'PaperPositionMode', 'auto');

% Set the font format of axis
set(gca, 'Fontname', 'Times New Roman', 'Fontsize', 10);

% Plot the amplitude spectrum of the original segment
plot(frequencyVector, Y, "Color", "#8C92AC");
% Zooms in the plot to easily observe the spectrums
maxW = max(W);
xlim([0 maxW + 100]);
hold on;

% Plot LPC frequency response on the graph
plot(W, H, 'r', 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Amplitude (dB)');

% Get the magnitude of the response at each formant frequency
formantMag = zeros(length(formantFrequencies), 1);
for i = 1:length(formantFrequencies)
    freqIndex = find(W >= formantFrequencies(i), 1);
    formantMag(i) = H(freqIndex);
```

```
end

% Plot the formant frequencies points
sz = 10;
scatter(formantFrequencies, formantMag, sz, "filled", "o", "MarkerFaceColor", "b
    ");

titleStr = ['LPC filter response and segment amplitude spectrum for ', gender, '
    vowel. ', 'Segment Length:', strSegmentLen, 'ms, Order:', strNthOrder];
title(titleStr, 'FontSize', 12);

legend('Original Segment Spectrum', 'LPC Frequency Response', 'First Three
    Formant Frequencies');
hold off;

% Save graph
graphName = ["LPC_response_and_segment_amplitude_spectrum_", gender, '
    _segment_length_', strSegmentLen, 'ms_Order_', strNthOrder];
saveGraph(gcf, graphName);

% Close the invisible figure
close(figure);
end
```

Listing 8: estimateFirstThreeFormant.m

```
function firstThreeFormants = estimateFirstThreeFormant(formantFrequencies)
% ESTIMATEFIRSTTHREEFORMANT Summary of this function goes here
%
% [OUTPUTARGS] = ESTIMATEFIRSTTHREEFORMANT(INPUTARGS) This function estimates the
     first three formant frequencies from the given formant frequencies.
% It selects the first three peaks and prints them.
%
% Examples:
% formantFrequencies = [1000, 2000, 3000, 4000];
% firstThreeFormants = estimateFirstThreeFormant(formantFrequencies);
%
% See also: computeFrequencyResponse, plotFrequencyResponse

% Author: Xiaoguang Liang, University of Surrey
% Date: 2024/11/01 17:52:37
% Revision: 0.1

%% Estimate the first three formant frequencies of the vowel
% Select the first three peaks as the first three formants
firstThreeFormants = zeros(3, 1);
```

```matlab
for i = 1:min(3, length(formantFrequencies))
    firstThreeFormants(i) = formantFrequencies(i);
    fprintf('Formant %d: %.2f Hz\n', i, formantFrequencies(i));
end

end
```

Listing 9: estimateMeanFundamentalFrequency.m

```matlab
function meanFundamentalFrequency = estimateMeanFundamentalFrequency(segment, Fs, gender)
% ESTIMATEMEANFUNDAMENTALFREQUENCY Summary of this function goes here
%
% [OUTPUTARGS] = ESTIMATEMEANFUNDAMENTALFREQUENCY(INPUTARGS) This function estimates the mean fundamental frequency (F0) of a given speech segment.
% It uses the autocorrelation method to find peaks within the expected F0 range based on the gender.
%
% Examples:
% segment = [0.1, 0.2, 0.3, 0.4, 0.5];
% Fs = 16000;
% gender = 'male';
% meanFundamentalFrequency = estimateMeanFundamentalFrequency(segment, Fs, gender);
%
% See also: computeFrequencyResponse, plotFrequencyResponse

% Author: Xiaoguang Liang, University of Surrey
% Date: 2024/10/30 10:52:36
% Revision: 0.1

% Use the correlation function
autocorrSegment = xcorr(segment);

% Find peaks in the autocorrelation within the expected lag range
[~, LOCS] = findpeaks(autocorrSegment);

% Define the expected F0 range, (90-155 Hz) for male and (165-255 Hz) for female
if strcmp(gender, 'female')
    minF0 = 165;
    maxF0 = 255;
else
    minF0 = 90;
    maxF0 = 155;
end

```

```matlab
% Calculate the lag values corresponding to the F0 range
minLag = round(Fs / maxF0);
maxLag = round(Fs / minF0);

% Filter peaks within the expected lag range
expectedPeaks = LOCS(LOCS >= minLag & LOCS <= maxLag);

% Calculate fundamental frequencies in Hz
fundamentalFrequencies = Fs ./ expectedPeaks;

% Calculate the mean F0
meanFundamentalFrequency = mean(fundamentalFrequencies);

end
```

Listing 10: speechSynthesis.m

```matlab
function speechSynthesis(Fs, meanF0, lpcCoeffs, saveFile)
% SPEECHSYNTHESIS Summary of this function goes here
%
% [OUTPUTARGS] = SPEECHSYNTHESIS(INPUTARGS) This function synthesizes speech
    using the given fundamental frequency, LPC coefficients, and sampling rate.
% It generates a periodic impulse train, filters it using the LPC coefficients,
    normalizes the synthesized signal, and saves it to a specified file.
%
% Examples:
% Fs = 16000;
% meanF0 = 120; % Example mean fundamental frequency
% lpcCoeffs = [1, -0.8, 0.3]; % Example LPC coefficients
% saveFile = 'synthesized_speech.wav';
% speechSynthesis(Fs, meanF0, lpcCoeffs, saveFile);
%
% See also: estimateMeanFundamentalFrequency, estimateLpcCoeficients

% Author: Xiaoguang Liang, University of Surrey
% Date: 2024/11/01 20:40:37
% Revision: 0.1

% Define the variable to manipulate the length of the synthesized signal
% Unit of measurement: seconds
desiredSpan = 1;
desiredSamples = round(desiredSpan * Fs);

%% Generate a periodic impulse train with the same fundamental frequency
% as the original vowel segment and of roughly a second in length
impulseTrainPeriod = round(Fs / meanF0);
```

```

impulseTrainSamples = min(desiredSamples, impulseTrainPeriod);
impulseTrain = zeros(1, desiredSamples);
impulseTrain(1:impulseTrainSamples:end) = 1;

% Filter the pulse train using the LPC filter determined above
synthesizedSignal = filter(1, lpcCoeffs, impulseTrain);

% Normalize the synthesized signal
synthesizedSignal = synthesizedSignal / max(abs(synthesizedSignal));

% Play the signal
% sound(synthesizedSignal, Fs);

% Make the save path for the synthesis audio
saveDir = GlobalSetting.SYNTHESIS_PATH;
if ~exist(saveDir, 'dir')
    % Create the new directory
    mkdir(saveDir);
end

savePath = fullfile(saveDir, saveFile);
% Synthesis signal output
audiowrite(savePath, synthesizedSignal, Fs);

end
```

Listing 11: plotExperimentGraph.m

```
function plotExperimentGraph(experimentArgs)
% PLOTEXPERIMENTGRAPH Summary of this function goes here
%
% [OUTPUTARGS] = PLOTEXPERIMENTGRAPH(INPUTARGS) Explain usage here
%
% Examples:
% plotExperimentGraph(struct('gender', {'male', 'female'}, 'segmentLen', {0.02,
   0.03}, ...
%                            'NthOrder', {10, 12}, 'H', {rand(1, 100), rand(1,
   100)}, ...
%                            'W', {0:0.01:1, 0:0.01:1}, 'formantFrequencies', {
   rand(1, 3), rand(1, 3)}, ...
%                            'Y', {rand(1, 100), rand(1, 100)}, 'frequencyVector
   ', {0:0.01:1, 0:0.01:1}));
%
% See also: saveGraph

```

```matlab
% Author: Xiaoguang Liang, University of Surrey
% Date: 2024/11/06 10:48:30
% Revision: 0.1

% Get the field names and data of the struct
genderData = {experimentArgs.gender};
segmentLenData = cellfun(@(x) num2str(x), {experimentArgs.segmentLen}, '
    UniformOutput',0);
NthOrderData = cellfun(@(y) num2str(y), {experimentArgs.NthOrder}, 'UniformOutput
    ',0);
HData = {experimentArgs.H};
WData = {experimentArgs.W};
formantFrequenciesData = {experimentArgs.formantFrequencies};
YData = {experimentArgs.Y};
frequencyVectorData = {experimentArgs.frequencyVector};

% Get unique values for gender, segment length, and order
genderUnique = unique(genderData);
segmentLenUnique = unique(segmentLenData);
NthOrderUnique = unique(NthOrderData);

% Loop through each unique gender
for i = 1:length(genderUnique)
    gender = genderUnique{i};

    % Plot the spectrum with fixed segment length and different orders
    for j = 1:length(segmentLenUnique)
        segmentLen = segmentLenUnique{j};
        % Find the index of the corresponding data
        idxes = find(strcmp(genderData, gender) & strcmp(segmentLenData,
            segmentLen));

        % Adjust the axes position for margins
        figure('Position', [100 100 800 500], 'Visible', 'off');
        ax = gca;
        ax.Position = [0.1, 0.1, 0.8, 0.8];
        set(gcf, 'PaperPositionMode', 'auto');
        set(gca, 'Fontname', 'Times New Roman', 'Fontsize', 10);

        NthOrders = NthOrderData(idxes);
        NthOrders = cellfun(@(x) num2str(x), NthOrders, 'UniformOutput',0);

        colorLen = length(NthOrders);
        colors = jet(colorLen);  % 'jet' colormap with 'numCurves' colors

        legendLabels = {}; % Initialize an empty cell array for legend labels
        l = 1;

        for k = 1:length(idxes)
            idx = idxes(k);
            % Extract the data for the current condition
            H = cell2mat(HData(idx));
            W = cell2mat(WData(idx));
            formantFrequencies = cell2mat(formantFrequenciesData(idx));
            Y = cell2mat(YData(idx));
            frequencyVector = cell2mat(frequencyVectorData(idx));

            if k == 1
                % Plot the amplitude spectrum of the original segment
                h(l) = plot(frequencyVector, Y, "Color", "#8C92AC");
                legendLabels{l} = 'Original Segment Spectrum'; % Add label to the
                    legend array
                l = l + 1;
            end

            % Zoom in the plot to find the formants clearly
            maxW = max(W);
            xlim([0 maxW + 100]);
            hold on;

            % Plot LPC frequency response on the graph
            h(l) = plot(W, H, 'Color', colors(k, :), 'LineWidth', 1.5);
            legendLabels{l} = strjoin(['LPC Frequency Response, Order:' NthOrders
                (k)]); % Add label to the legend array
            l = l + 1;
            xlabel('Frequency (Hz)');
            ylabel('Amplitude (dB)');

            % Get the magnitude of the response at each formant frequency
            formantMag = zeros(2, 1);
            for n = 1:length(formantFrequencies)
                freqIndex = find(W >= formantFrequencies(n), 1);
                formantMag(n) = H(freqIndex);
            end

            % Plot the formant frequencies points
            sz = 10;
            h(l) = scatter(formantFrequencies, formantMag, sz, "filled", "o", "
                MarkerFaceColor", 'k');
            legendLabels{l} = 'First Three Formant Frequencies';
```

```matlab
            l = l + 1;

            hold on;
        end

        strSegmentLen = num2str(str2num(segmentLen) * 1000);
        titleStr = ['LPC filter response and segment amplitude spectrum for ', gender, ' vowel. ', 'Segment Length:', strSegmentLen, 'ms, different Orders'];
        title(titleStr, 'FontSize', 10);

        legend(h, legendLabels);
        hold off;

        % Save graph
        graphName = ["Experiment_LPC_response_and_segment_amplitude_spectrum_", gender, '_segment_length_', strSegmentLen, 'ms_different_Orders'];
        saveGraph(gcf, graphName);

        % Close the invisible figure
        close(figure);
    end

    % Plot the spectrum with fixed order and different segment lengths
    for j = 1:length(NthOrderUnique)
        NthOrder = NthOrderUnique{j};
        % Find the index of the corresponding data
        idxes = find(strcmp(genderData, gender) & strcmp(NthOrderData, NthOrder));

        % Adjust the axes position for margins
        figure('Position', [100 100 800 500], 'Visible', 'off');
        ax = gca;
        ax.Position = [0.1, 0.1, 0.8, 0.8];
        set(gcf, 'PaperPositionMode', 'auto');
        set(gca, 'Fontname', 'Times New Roman', 'Fontsize', 10);

        segmentLens = segmentLenData(idxes);
        segmentLens = cellfun(@(x) num2str(x), segmentLens, 'UniformOutput',0);

        colorLen = length(segmentLens);
        colors = jet(colorLen);  % 'jet' colormap with 'numCurves' colors

        legendLabels = {}; % Initialize an empty cell array for legend labels
        l = 1;

        for k = 1:length(idxes)
            idx = idxes(k);
            % Extract the data for the current condition
            H = cell2mat(HData(idx));
            W = cell2mat(WData(idx));
            formantFrequencies = cell2mat(formantFrequenciesData(idx));

            % Plot LPC frequency response on the graph
            h(l) = plot(W, H, 'Color', colors(k, :), 'LineWidth', 1.5);
            strSegmentLen = segmentLens{k};
            strSegmentLen = num2str(str2num(strSegmentLen) * 1000);
            label_ = {'LPC Frequency Response, Segment length:', strSegmentLen, ' ms'};
            legendLabels{l} = strjoin(label_); % Add label to the legend array
            l = l + 1;
            xlabel('Frequency (Hz)');
            ylabel('Amplitude (dB)');
            hold on;

            % Get the magnitude of the response at each formant frequency
            formantMag = zeros(2, 1);
            for n = 1:length(formantFrequencies)
                freqIndex = find(W >= formantFrequencies(n), 1);
                formantMag(n) = H(freqIndex);
            end

            % Plot the formant frequencies points
            sz = 10;
            h(l) = scatter(formantFrequencies, formantMag, sz, "filled", "o", "MarkerFaceColor", 'k');
            legendLabels{l} = 'First Three Formant Frequencies';
            l = l + 1;

            hold on;
        end

        strNthOrder = NthOrder;
        titleStr = ['LPC filter response and segment amplitude spectrum for ', gender, ' vowel. ', 'Order:', strNthOrder, ', different Segment Length'];
        title(titleStr, 'FontSize', 10);

        legend(h, legendLabels);
        hold off;

        % Save graph
        graphName = ["Experiment_LPC_response_and_segment_amplitude_spectrum_",
            gender, '_Order_', strNthOrder, '_different_segment_lengths'];
        saveGraph(gcf, graphName);

        % Close the invisible figure
        close(figure);
    end
end
end
```

Listing 12: saveGraph.m

```
function saveGraph(gcf, graphName)
% SAVEGRAPH Summary of this function goes here
%
% [OUTPUTARGS] = SAVEGRAPH(INPUTARGS) This function saves the current figure as a
    high-resolution PNG file.
% It ensures that the directory exists and creates it if necessary.
%
% Examples:
% saveGraph(gcf, 'example_graph');
%
% See also: plotFrequencyResponse

% Author: Xiaoguang Liang, University of Surrey
% Date: 2024/11/01 22:38:29
% Revision: 0.1

% Save the graph as a high-resolution PNG file
saveDir = [GlobalSetting.GRAPH_PATH];
if ~exist(saveDir, 'dir')
    % Create the new directory
    mkdir(saveDir);
end

savePath = [saveDir, '/', graphName, '.png'];
savePath = strjoin(savePath, '');

% Save the figure as a PNG file with specified margins
print(gcf, savePath, '-dpng', '-r300');  % '-r300' sets 300 DPI for high
   resolution

end
```