College of Electrical and Mechanical Engineering
National University of Sciences and Technology

Digital Signal Processing Spring 2023

Open Lab: Linear Predictive Coding

Submission: 25 May 2023

AMINA QADEER

CE-42-A

359607

Objective:

In this lab, you will be building upon your knowledge gained in the previous labs and build upon them to implement Linear Predictive Coding and use it to compress speech audio.

CODE:

```
% LPC Coding Function
function [coeff, pitch, gain] = lpcCoding(audio, frameSize, overlap, order)
    % Frame parameters
    frameShift = frameSize - overlap;
    numFrames = floor((length(audio) - overlap) / frameShift);

    % Initialize variables to store coefficients, pitch, and gain
coeff = zeros(order+1, numFrames);    pitch = zeros(1, numFrames);
gain = zeros(1, numFrames);

    %   Loop   over   frames
index = 1;
    for  frame  =  1:numFrames
% Extract current frame
        frameData = audio(index:index+frameSize-1);

        % Apply LPC analysis
        [a, pitch(frame), err] = myLPC(frameData, order);

        % Store coefficients, pitch, and gain
coeff(:, frame) = a;          gain(frame) =
sqrt(mean(err));
```

```
        % Update index for the next frame
index = index + frameShift;     end end

% Synthesis Function
function synthesizedAudio = synthesis(coeff, pitch, gain, frameSize, overlap)
    % Frame parameters
    frameShift    =    frameSize    -    overlap;
numFrames = size(coeff, 2);

    % Initialize variables for synthesized audio
    synthesizedAudio = zeros(1, (numFrames-1)*frameShift + frameSize);

    %   Loop   over   frames
index = 1;
    for  frame  =  1:numFrames            %
Initialize frame signal        frameSignal
= zeros(1, frameSize);

        % Generate  initial  signal  based  on  pitch
if pitch(frame) == 0
            frameSignal  =  randn(size(frameSignal));  %  Unvoiced  frame
else
            pulses = pulstran((0:frameSize-1), pitch(frame)*(1/frameSize),
@rectpuls); % Voiced frame
            frameSignal  =  frameSignal  +  pulses;
end

        % Apply LPC synthesis filter
        frameSignal = filter(1, coeff(:, frame), frameSignal);

        % Scale frame signal by gain
        frameSignal = frameSignal * gain(frame);

        %       Add       frame       signal       to       synthesized       audio
synthesizedAudio(index:index+frameSize-1)                                          =
synthesizedAudio(index:index+frameSize-1) + frameSignal;

        % Update index for the next frame
index = index + frameShift;     end end %


% Custom LPC analysis function
function [a, pitch, err] = myLPC(frameData, order)
    % Compute autocorrelation     r
= xcorr(frameData, frameData);

    % Apply  Levinson-Durbin  recursion
a = levinson(r, order);
```

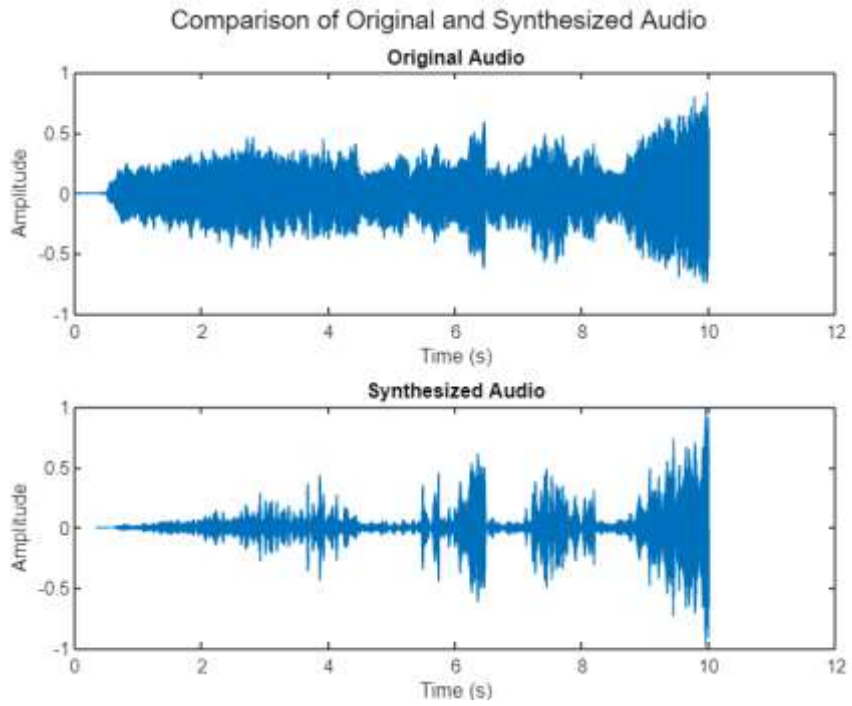
```
    % Estimate pitch
    [rxx, lags] = xcorr(frameData);
    [~, locs] = findpeaks(rxx(lags > 0)); % Find local peaks
[~, sortedIndex] = sort(rxx(locs), 'descend');     sortedLocs
= locs(sortedIndex);

    if length(sortedLocs) >= 2 && sortedLocs(2) - sortedLocs(1) > 100
pitch = sortedLocs(2) - sortedLocs(1);     else
        pitch   =   0;
end

    %  Compute  prediction  error
err  =  filter(a,  1,  frameData);
err = err(order+1:end); end
```

Main:

```
% Read audio file filename
= 'audio.wav';
[audio, Fs] = audioread(filename);

% LPC parameters
frameSize = round(0.03 * Fs); % 30 ms frame size
overlap = round(0.015 * Fs); % 15 ms overlap order
= 12; % LPC order

% LPC coding
[coeff, pitch, gain] = lpcCoding(audio, frameSize, overlap, order);

% Synthesis
synthesizedAudio = synthesis(coeff, pitch, gain, frameSize, overlap);

% Adjust length of synthesized audio to match original audio if
length(synthesizedAudio) > length(audio)
    synthesizedAudio = synthesizedAudio(1:length(audio)); else
    audio = audio(1:length(synthesizedAudio)); end

% Time axis for plotting t_audio =
(0:length(audio)-1) / Fs;
t_synthesized = (0:length(synthesizedAudio)-1) / Fs;

%   Original   audio   plot
subplot(2,1,1);
plot(t_audio,         audio);
title('Original     Audio');
xlabel('Time          (s)');
ylabel('Amplitude');

% Synthesized audio plot subplot(2,1,2);
```

```
plot(t_synthesized, synthesizedAudio);
title('Synthesized Audio'); xlabel('Time
(s)'); ylabel('Amplitude');

% Adjust plot spacing
sgtitle('LPC Coding: Original Audio vs Synthesized Audio');
```

OUTPUT:



Explanation:

1. LPC Coding Function: The function **lpcCoding** performs LPC analysis on an audio signal. It takes the input audio, frame size, overlap, and LPC order as parameters and returns the LPC coefficients, pitch estimates, and gain values.

2. Synthesis Function: The function **synthesis** synthesizes an audio signal based on the LPC coefficients, pitch estimates, gain values, frame size, and overlap. It generates frames of the synthesized audio signal by applying an LPC synthesis filter and scaling the frame signal by the corresponding gain. If the frame is voiced, it generates pulses based on the pitch estimate; otherwise, it generates random noise. The synthesized frames are then combined to obtain the final synthesized audio signal.

3. Main Script: The main script performs LPC coding and synthesis on an audio file and plots the original and synthesized audio signals. Here's a breakdown of the main script:

- The audio file is read using the **audioread** function, and the sampling frequency **Fs** and audio data **audio** are obtained.

- LPC parameters such as frame size, overlap, and LPC order are defined.
- The LPC coding function **lpcCoding** is called with the audio data and parameters, which returns the LPC coefficients **coeff**, pitch estimates **pitch**, and gain values **gain**.
- The synthesis function **synthesis** is called with the obtained LPC parameters and values, resulting in the synthesized audio signal **synthesizedAudio**.
- If the length of the synthesized audio exceeds the original audio, it is truncated to match the length of the original audio.
- Time axes for plotting are created.
- Two subplots are created: one for the original audio and another for the synthesized audio. The corresponding time axes, titles, and axis labels are set.
- The **sgtitle** function sets a common title for the subplots.
- The script finishes, and the plots of the original and synthesized audio signals are displayed.

Overall, this code implements LPC analysis and synthesis to model and recreate an audio signal using LPC coefficients, pitch estimates, and gain values. The synthesized audio is compared with the original audio through plots.