Open Ended Lab

Digital Signal Processing (CS-419)

Fourth Year-Computer and Information Systems Engineering

Batch: 2022

Group Members:

Ms. Amina Shahzad.......................................(CS-22019)

Ms. Jaweria Anwer.......................................(CS-22057)

Ms. Shiza Ejaz .... (CS-22060)

Ms. Fatima Mehdvi.......................................(CS-22065)

Submitted to: Miss Fakhra Aftab

Contents

1. Abstract

This document summarizes the updated Speech Cleaner application developed in MATLAB App Designer. The application enables loading speech audio, adding Gaussian noise, applying digital filters (low-pass, high-pass, band-stop), visualizing signals in time and frequency domains, and listening to original, noisy, and filtered audio.

The project fulfills CLO-3: “**Practice use of modern tools and techniques for digital signal processing.**” This report documents design methodology, implementation details, corrected algorithms, GUI layout, results, and recommendations. Screenshots and figure placeholders are included.

2. Introduction

2.1 Background

Digital signal processing enhances speech by reducing environmental noise, improving intelligibility in real-world applications like telecom, recording, and hearing aids. Using MATLAB App Designer, this project demonstrates noise addition, reversal, and removal with interactive filters and visualization.

2.2 Project Objective

Primary objectives:

- Load speech audio files
- Add noise (Gaussian)
- Design and apply digital filters (low-pass, high-pass, band-stop)
- Visualize signals in time and frequency domains

2.3 Scope of the Project

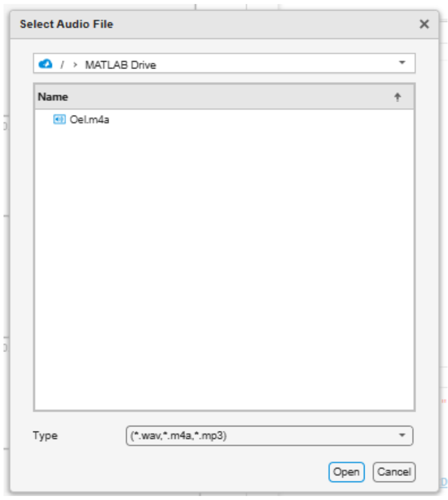
Focus areas include filter design and application, noise simulation, GUI-based interaction (slider and dropdown control), and visual/audible comparison of processing results.

3. System Design

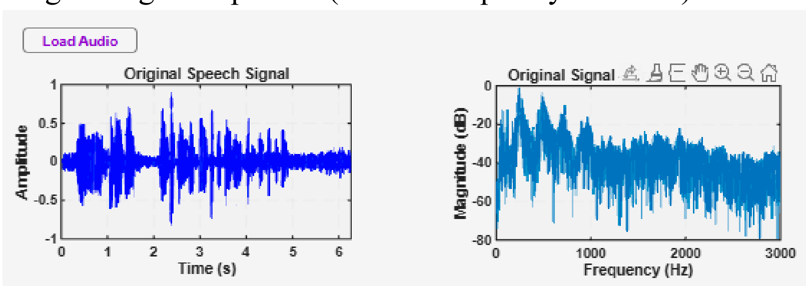
3.1 Overall Workflow

Workflow sequence:

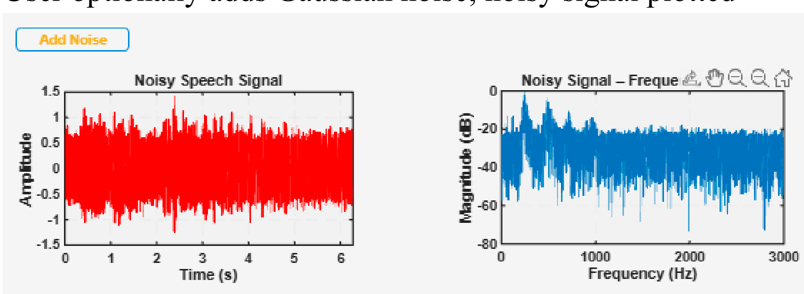
- User loads/records a speech signal



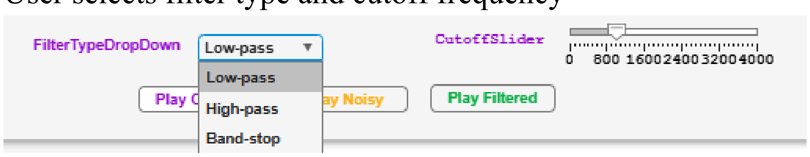
➢ Original signal is plotted (time & frequency domains)



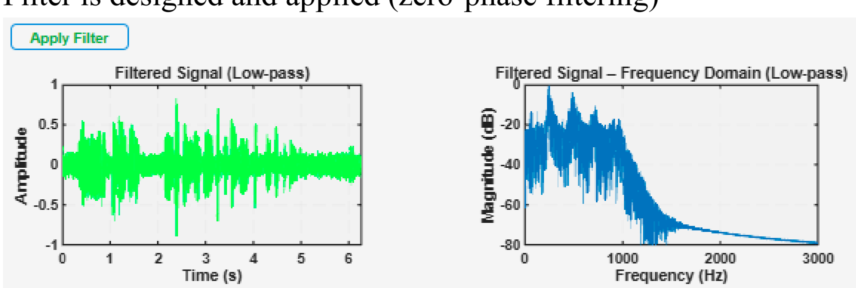
➢ User optionally adds Gaussian noise; noisy signal plotted



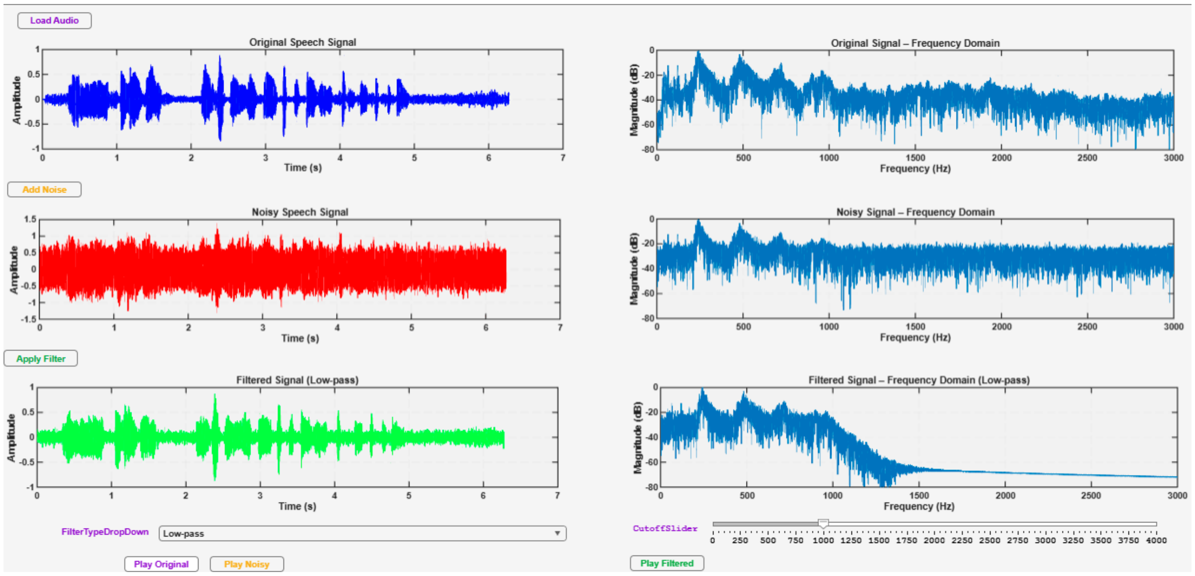
➢ User selects filter type and cutoff frequency



➢ Filter is designed and applied (zero-phase filtering)



➢ Filtered signal is plotted and available for playback



4. MATLAB App Designer Interface

GUI components:

- Buttons: Load Audio, Record, Add Noise, Apply Filter, Play Original, Play Noisy, Play Filtered
- Dropdown: Filter Type (Low-Pass, High-Pass, Band-Stop)
- Slider/Spinner: Noise Level (SNR), Cutoff Frequency
- Axes: Time-domain plot(s), Frequency-domain (magnitude) plot(s)
- Status labels and instructions

5. DSP Concepts and Implementation

5.1 Speech Signal Loading

Signals are loaded using audioread and normalized to avoid clipping.

```matlab
% Button pushed function: LoadAudioButton, PlayOriginalButton
function LoadAudioButtonPushed(app, event)
   [file, path] = uigetfile({'*.wav;*.m4a;*.mp3'}, 'Select Audio File');
    if isequal(file,0)
        return;
    end

    filePath = fullfile(path,file);
    [sig, app.fs] = audioread(filePath);

    % Convert to mono if stereo and ensure column
    if size(sig,2) > 1
        sig = mean(sig,2);
    end
    sig = sig(:);

    app.originalSignal = sig;


    t = (0:length(app.originalSignal)-1) ./app.fs;

    plotTime(app, app.OriginalAxes, t, app.originalSignal, 'b', ...
             'Original Speech Signal');

    plotFreq(app, app.OriginalFreqAxes, app.originalSignal, ...
             'Original Signal - Frequency Domain', 1000);
    sound(app.originalSignal, app.fs);

end
```

5.2 Noise Addition

White (Gaussian) noise is added with user-controlled level. Use an SNR-based approach to make the slider meaningful:

```matlab
% Button pushed function: AddNoiseButton, PlayNoisyButton
function AddNoiseButtonPushed(app, event)

    if isempty(app.originalSignal)
        uialert(app.UIFigure,'Please load audio first!','Error');
        return;
    end

    % Add Gaussian noise (scale chosen empirically)
    app.noisySignal = app.originalSignal + 0.2*randn(size(app.originalSignal));

    t = (0:length(app.noisySignal)-1) ./app.fs;

    plotTime(app, app.NoisyAxes, t, app.noisySignal, 'r', ...
             'Noisy Speech Signal');

    plotFreq(app, app.NoisyFreqAxes, app.noisySignal, ...
             'Noisy Signal - Frequency Domain', 1000);
    sound(app.noisySignal, app.fs);
end
```

5.3 Filter Design

Supported filters: Low-pass, High-pass, Band-stop.

```
% Button pushed function: ApplyFilterButton, PlayFilteredButton
function ApplyFilterButtonPushed(app, event)

        if isempty(app.noisySignal)
            uialert(app.UIFigure,'Add noise first!','Error');
            return;
        end

        % Read filter type from dropdown
        filterType = app.FilterTypeDropDown.Value;

        % Read cutoff from slider
        cutoff = app.CutoffSlider.Value;

        % Design filter based on selection
        switch filterType
            case 'Low-pass'
                d = designfilt('lowpassiir','FilterOrder',8, ...
                     'HalfPowerFrequency',cutoff/(app.fs/2));

            case 'High-pass'
                d = designfilt('highpassiir','FilterOrder',8, ...
                     'HalfPowerFrequency',cutoff/(app.fs/2));

            case 'Band-stop'
                f1 = (cutoff - 500) / (app.fs/2);
                f2 = (cutoff + 500) / (app.fs/2);
                d = designfilt('bandstopiir','FilterOrder',8, ...
                    'HalfPowerFrequency1',max(f1,0.001), ...
                    'HalfPowerFrequency2',min(f2,0.999));
        end
```

5.4 Filtering Operation

Zero-phase filtering using filtfilt prevents phase distortion, which is important for speech quality. After filtering, visualize time-domain and frequency-domain results.

```
        % Apply filtering
        app.filteredSignal = filtfilt(d, app.noisySignal);

        % Create time axis
        t = (0:length(app.filteredSignal)-1) / app.fs;

        % ==== TIME DOMAIN PLOT ====
        plotTime(app, app.FilteredAxes, t, app.filteredSignal, 'g', ...
            ['Filtered Signal (' filterType ')']);

        % ==== FREQUENCY DOMAIN PLOT ====
        switch filterType
            case {'Low-pass','High-pass'}
                plotFreq(app, app.FilteredFreqAxes, app.filteredSignal, ...
                    ['Filtered Signal - Frequency Domain (' filterType ')'], cutoff);

            case 'Band-stop'
                plotFreq(app, app.FilteredFreqAxes, app.filteredSignal, ...
                    ['Filtered Signal - Frequency Domain (Band-stop)'], []);
                hold(app.FilteredFreqAxes,'on');
                xline(app.FilteredFreqAxes, cutoff-500,'--r','LineWidth',1.5);
                xline(app.FilteredFreqAxes, cutoff+500,'--r','LineWidth',1.5);
                hold(app.FilteredFreqAxes,'off');
        end
        sound(app.filteredSignal, app.fs);

end
```

6. Results and Discussion

Expected outcomes

- Time-domain plots reveal how noise and filtering affect waveform clarity.

- Frequency-domain plots show suppression or preservation of spectral components depending on filter choice.
- Playback allows subjective evaluation of filtering impact.

7. Conclusion

The updated Speech Cleaner application integrates core DSP techniques within an interactive GUI. It enables loading, noisy simulation, filter design/application, visualization, and listening to results, making it a practical teaching and experimentation tool.

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-419
Course Title: Digital Signal Processing
Open Ended Lab
BE Batch 2022, Fall Semester 2025
Grading Rubric

Roll#	**Roll#**	**Roll#**	**Roll#**

Project Title: ____

Criteria	Description	Excellent (5)	Good (4/3)	Basic (2)	NI (1/0)
Understanding of DSP Concepts	Demonstrated grasp of underlying DSP principles relevant to the chosen project (filtering, sampling, system analysis, etc.).	Shows deep understanding; accurately applies theory to implementation.	Shows good understanding with minor conceptual gaps.	Demonstrates basic understanding but with limited explanation.	Misinterprets key DSP ideas or applies them incorrectly.
Functionality of the App	App performs all required functions correctly (signal processing, plotting, playback, etc.).	Fully functional, no errors, all features implemented correctly.	Most functions work, minor bugs present.	Some core features missing or partially working.	Major functions incomplete or incorrect.
User Interface Design (App Designer)	Clarity, organization, and interactivity of GUI (use of buttons, sliders, plots, labels).	GUI is intuitive, well-labeled, and visually appealing; smooth interactivity.	GUI is clear but slightly cluttered or missing some labels.	Basic GUI; minimal interactivity or confusing layout.	Poorly designed interface; difficult to use or incomplete.
Analysis, Visualization, and Interpretation	Quality of signal visualization (time-domain, frequency-domain, pole-zero plots, etc.) and interpretation of results.	Plots are clear, correctly scaled, and well-labeled; insightful interpretation.	Plots correct but interpretation basic.	Some plots unclear or missing labels; limited discussion.	Incorrect or missing analysis; no interpretation.
Report & Presentation	Clarity, completeness, and professionalism of written and/or oral presentation.	Concise, well-structured report; clear explanation of design choices and results.	Report mostly clear, with minor issues in organization or analysis.	Report covers main points but lacks detail or clarity.	Report missing sections or poorly written; unclear explanations.

Total Marks Obtained: _________ out of 25

Teacher’s Signature: ________