**DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING**

**COLLEGE OF E&ME, NUST, RAWALPINDI**

**Subject Name**

**Digital Signal Processing**

**Lab Number**

**3**

**SUBMITTED TO:**

**LE Sundas Ashraf**

**SUBMITTED BY:**

**Student Name**

1. Wahaaj Nasir

**Reg#413238**

**DE- 44 Dept C&SE**

**Objectives:**

Processing in MATLab

**Related Topic/Chapter in theory class:**

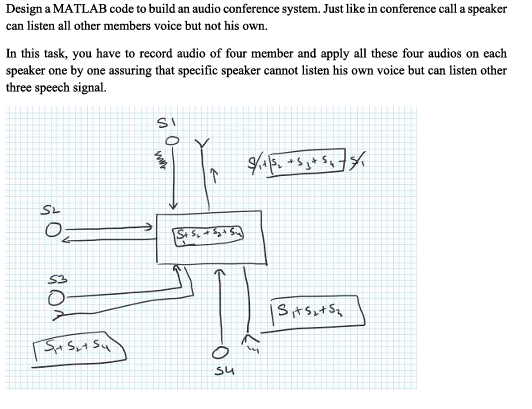
Basics Of Digital Signal Processing

**Hardware/Software required:**

Hardware: PC

Software Tool: MATLab

**Task :**

**Solution:**

classdef Conference < matlab.apps.AppBase

% Properties that correspond to app components

properties (Access = public)

UIFigure matlab.ui.Figure

Play4Button matlab.ui.control.Button

Record4Button matlab.ui.control.Button

Play3Button matlab.ui.control.Button

Record3Button matlab.ui.control.Button

Play2Button matlab.ui.control.Button

Record2Button matlab.ui.control.Button

Play1Button matlab.ui.control.Button

Record1Button matlab.ui.control.Button

UIAxes matlab.ui.control.UIAxes

end

% Callbacks that handle component events

methods (Access = private)

% Button pushed function: Record1Button

function Record1ButtonPushed(app, event)

recorder = audiorecorder(8000, 8, 1);

disp("Start Recording");

recordblocking(recorder, 4);

disp("Stop Recording");

pause(3)

record1 = getaudiodata(recorder);

audiowrite('Record1.wav', record1, 8000)

plot(app.UIAxes, record1)

xlabel(app.UIAxes, 'Time')

ylabel(app.UIAxes, 'Amplitude')

title(app.UIAxes, 'Record 1')

end

% Button pushed function: Record2Button

function Record2ButtonPushed(app, event)

recorder = audiorecorder(8000, 8, 1);

disp("Start Recording");

recordblocking(recorder, 4);

disp("Stop Recording");

pause(3)

record2 = getaudiodata(recorder);

audiowrite('Record2.wav', record2, 8000)

plot(app.UIAxes, record2)

xlabel(app.UIAxes, 'Time')

ylabel(app.UIAxes, 'Amplitude')

title(app.UIAxes, 'Record 2')

end

% Button pushed function: Record3Button

function Record3ButtonPushed(app, event)

recorder = audiorecorder(8000, 8, 1);

disp("Start Recording");

recordblocking(recorder, 4);

disp("Stop Recording");

pause(3)

record3 = getaudiodata(recorder);

audiowrite('Record3.wav', record3, 8000)

plot(app.UIAxes, record3)

xlabel(app.UIAxes, 'Time')

ylabel(app.UIAxes, 'Amplitude')

title(app.UIAxes, 'Record 3')

end

% Button pushed function: Record4Button

function Record4ButtonPushed(app, event)

recorder = audiorecorder(8000, 8, 1);

disp("Start Recording");

recordblocking(recorder, 4);

disp("Stop Recording");

pause(3)

record4 = getaudiodata(recorder);

audiowrite('Record4.wav', record4, 8000)

plot(app.UIAxes, record4)

xlabel(app.UIAxes, 'Time')

ylabel(app.UIAxes, 'Amplitude')

title(app.UIAxes, 'Record 4')

end

% Button pushed function: Play1Button

function Play1ButtonPushed(app, event)

[record2, Fs] = audioread('Record2.wav');

[record3, Fs] = audioread('Record3.wav');

[record4, Fs] = audioread('Record4.wav');

sil = zeros(Fs \* 1, size(record2, 2));

final\_voice = [record2; sil; record3; sil; record4];

size(final\_voice, 1)

sound(final\_voice, 8000)

plot(app.UIAxes, final\_voice)

xlabel(app.UIAxes, 'Time')

ylabel(app.UIAxes, 'Amplitude')

title(app.UIAxes, 'Record 2 + 3 + 4')

end

% Button pushed function: Play2Button

function Play2ButtonPushed(app, event)

[record1, Fs] = audioread('Record1.wav');

[record3, Fs] = audioread('Record3.wav');

[record4, Fs] = audioread('Record4.wav');

sil = zeros(Fs \* 1, size(record1, 2));

final\_voice = [record1; sil; record3; sil; record4];

sound(final\_voice, 8000)

size(final\_voice, 1)

plot(app.UIAxes, final\_voice)

xlabel(app.UIAxes, 'Time')

ylabel(app.UIAxes, 'Amplitude')

title(app.UIAxes, 'Record 1 + 3 + 4')

end

% Button pushed function: Play3Button

function Play3ButtonPushed(app, event)

[record1, Fs] = audioread('Record1.wav');

[record2, Fs] = audioread('Record2.wav');

[record4, Fs] = audioread('Record4.wav');

sil = zeros(Fs \* 1, size(record2, 2));

final\_voice = [record1; sil; record2; sil; record4];

sound(final\_voice, 8000)

size(final\_voice, 1)

plot(app.UIAxes, final\_voice)

xlabel(app.UIAxes, 'Time')

ylabel(app.UIAxes, 'Amplitude')

title(app.UIAxes, 'Record 1 + 2 + 4')

end

% Button pushed function: Play4Button

function Play4ButtonPushed(app, event)

[record1, Fs] = audioread('Record1.wav');

[record2, Fs] = audioread('Record2.wav');

[record3, Fs] = audioread('Record3.wav');

sil = zeros(Fs \* 1, size(record2, 2));

final\_voice = [record1; sil; record2; sil; record3];

sound(final\_voice, 8000)

size(final\_voice, 1)

plot(app.UIAxes, final\_voice)

xlabel(app.UIAxes, 'Time')

ylabel(app.UIAxes, 'Amplitude')

title(app.UIAxes, 'Record 1 + 2 + 3')

end

end

% Component initialization

methods (Access = private)

% Create UIFigure and components

function createComponents(app)

% Create UIFigure and hide until all components are created

app.UIFigure = uifigure('Visible', 'off');

app.UIFigure.Position = [100 100 640 480];

app.UIFigure.Name = 'MATLAB App';

% Create UIAxes

app.UIAxes = uiaxes(app.UIFigure);

title(app.UIAxes, 'Title')

xlabel(app.UIAxes, 'X')

ylabel(app.UIAxes, 'Y')

zlabel(app.UIAxes, 'Z')

app.UIAxes.Position = [171 149 300 185];

% Create Record1Button

app.Record1Button = uibutton(app.UIFigure, 'push');

app.Record1Button.ButtonPushedFcn = createCallbackFcn(app, @Record1ButtonPushed, true);

app.Record1Button.Position = [204 394 100 23];

app.Record1Button.Text = 'Record 1';

% Create Play1Button

app.Play1Button = uibutton(app.UIFigure, 'push');

app.Play1Button.ButtonPushedFcn = createCallbackFcn(app, @Play1ButtonPushed, true);

app.Play1Button.Position = [342 394 100 23];

app.Play1Button.Text = 'Play 1';

% Create Record2Button

app.Record2Button = uibutton(app.UIFigure, 'push');

app.Record2Button.ButtonPushedFcn = createCallbackFcn(app, @Record2ButtonPushed, true);

app.Record2Button.Position = [35 279 100 23];

app.Record2Button.Text = 'Record 2';

% Create Play2Button

app.Play2Button = uibutton(app.UIFigure, 'push');

app.Play2Button.ButtonPushedFcn = createCallbackFcn(app, @Play2ButtonPushed, true);

app.Play2Button.Position = [35 205 100 23];

app.Play2Button.Text = 'Play 2';

% Create Record3Button

app.Record3Button = uibutton(app.UIFigure, 'push');

app.Record3Button.ButtonPushedFcn = createCallbackFcn(app, @Record3ButtonPushed, true);

app.Record3Button.Position = [500 279 100 23];

app.Record3Button.Text = 'Record 3';

% Create Play3Button

app.Play3Button = uibutton(app.UIFigure, 'push');

app.Play3Button.ButtonPushedFcn = createCallbackFcn(app, @Play3ButtonPushed, true);

app.Play3Button.Position = [500 205 100 23];

app.Play3Button.Text = 'Play 3';

% Create Record4Button

app.Record4Button = uibutton(app.UIFigure, 'push');

app.Record4Button.ButtonPushedFcn = createCallbackFcn(app, @Record4ButtonPushed, true);

app.Record4Button.Position = [204 76 100 23];

app.Record4Button.Text = 'Record 4';

% Create Play4Button

app.Play4Button = uibutton(app.UIFigure, 'push');

app.Play4Button.ButtonPushedFcn = createCallbackFcn(app, @Play4ButtonPushed, true);

app.Play4Button.Position = [342 76 100 23];

app.Play4Button.Text = 'Play 4';

% Show the figure after all components are created

app.UIFigure.Visible = 'on';

end

end

% App creation and deletion

methods (Access = public)

% Construct app

function app = Conference

% Create UIFigure and components

createComponents(app)

% Register the app with App Designer

registerApp(app, app.UIFigure)

if nargout == 0

clear app

end

end

% Code that executes before app deletion

function delete(app)

% Delete UIFigure when app is deleted

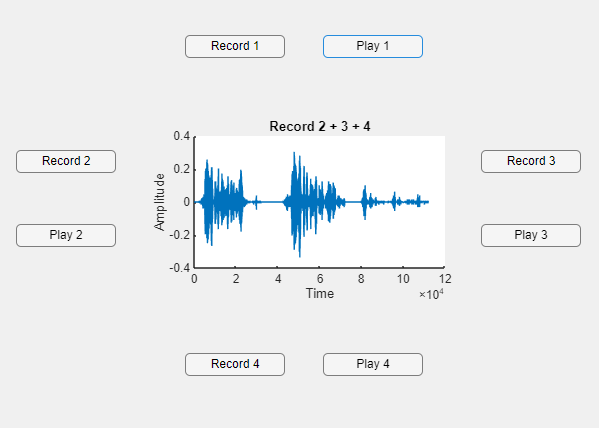
delete(app.UIFigure)

end

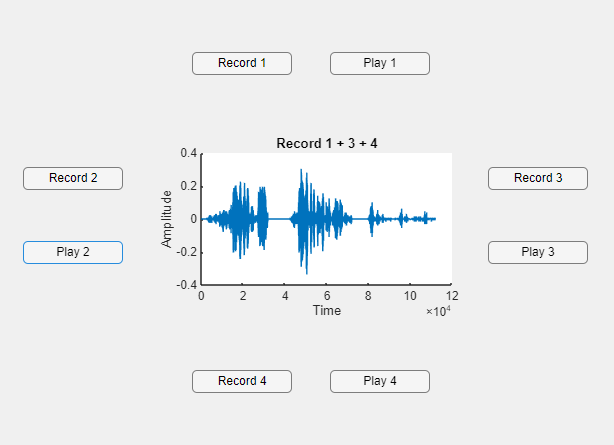
end

end

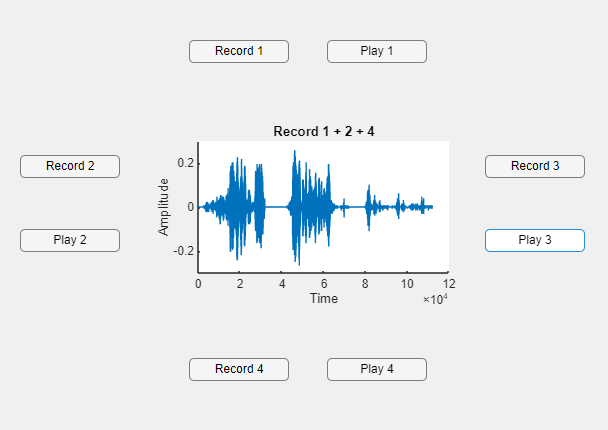
**Pressing Play 1:**



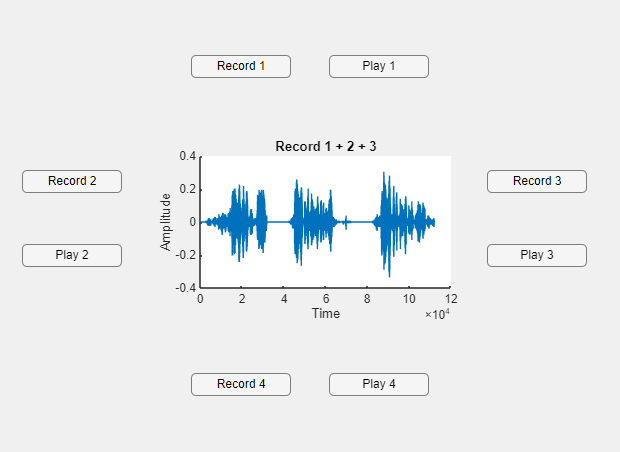
**Play 2:**



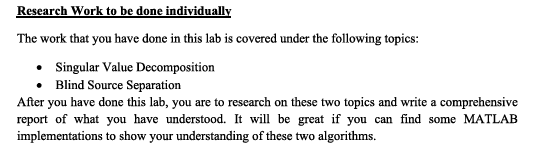
**Play 3:**



**Play 4:**



**Research Work:**

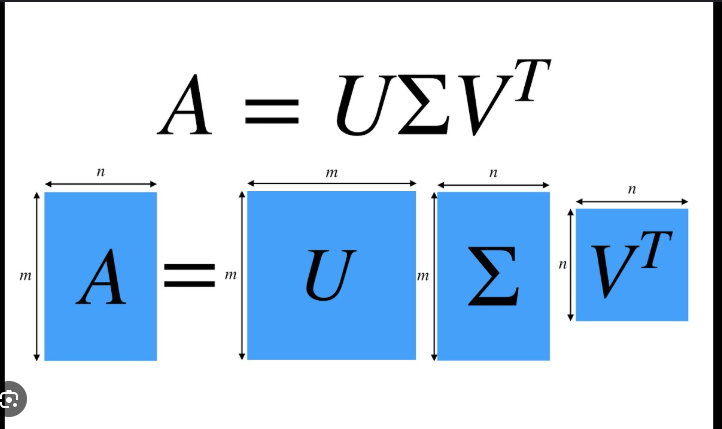
****

**Single Value Decomposition:**

Singular Value Decomposition (SVD) is a mathematical technique used to factorize a matrix into three matrices:

Where:

* U is an m × m orthogonal matrix (left singular vectors) (columns are eigenvectors of )
* Σ is an m × n diagonal matrix containing singular values (diagonal entries, square roots of eigenvalues of )
* is an n × n orthogonal matrix (right singular vectors) (columns are eigenvectors of )



**MATLAB Implementation:**

A = [4 0; 3 -5; 0 2];

% Compute SVD

[U, S, V] = svd(A);

% Display results

disp('U matrix:');

disp(U);

disp('Singular values (S matrix):');

disp(S);

disp('V matrix:');

disp(V);

% Verify the decomposition

A\_reconstructed = U \* S \* V';

disp('Reconstructed A:');

disp(A\_reconstructed);

**Output:**

U matrix:

-0.4059 0.8736 -0.2683

-0.8840 -0.3009 0.3578

0.2318 0.3824 0.8944

Singular values (S matrix):

6.4910 0

0 3.4449

0 0

V matrix:

-0.6587 0.7524

0.7524 0.6587

Reconstructed A:

4.0000 0.0000

3.0000 -5.0000

0.0000 2.0000

**Blind Source Separation:**

Blind Source Separation (BSS) is the process of separating mixed signals into their original independent sources without prior knowledge of the mixing process. The most common technique for BSS is **Independent Component Analysis (ICA)**.

Applications include:

* Audio signal separation (e.g., separating voices in a room)
* Biomedical signal processing (e.g., EEG signal separation)