UNIVERSITÀ DEGLI STUDI DELL'AQUILA
Department of Information Engineering, Computer Science and Mathematics

Introduction to Matlab – Spring 2025

Lecturer: Prof. A. Cicone

Final Exam

Project ♯ 73

Build a function `fun` that takes as input two vectors u and v and as output provides the vector w which contains in each entry the max between the same entry position of the two vectors.
Download two audio signals and load them into Matlab. Use the function `fun` applied to the first 0.3 seconds of the first audio channel of the two signals. Plot the original two audio signals and the one obtained using the function in a single figure using subplots.
Generate a report in pdf containing the results obtained, included all the codes created.

Matlab Project #73 report

Long Nguyen

April 20, 2025

1 Project requirements

Build a function `fun` that takes as input two vectors `u` and `v` and as output provides the vector `w` which contains in each entry the max between the same entry position of the two vectors.

Download 2 audio signals and load them into Matlab. Use the function `fun` applied to the first 0.3 seconds of the **first audio channel** of the 2 signals. Plot the original 2 audio signals and the one obtained using the function in a single figure using subplots.

Generate a report in pdf containing the results obtained, including all the codes created.

2 Results

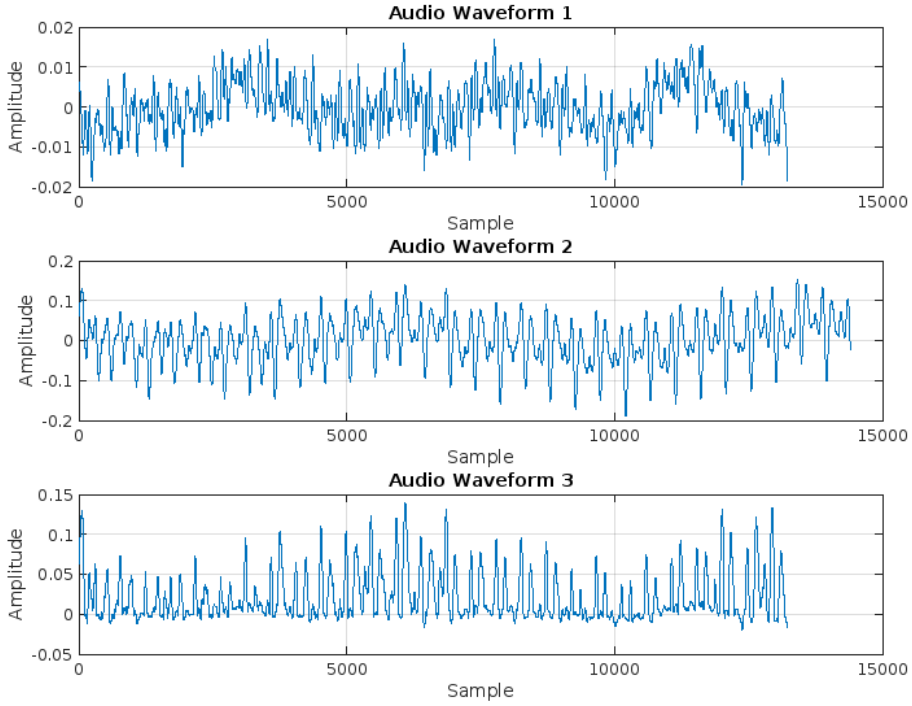
Main function

```
% Add submodule paths
addpath('utils');
addpath('static');

audio1 = read_audio("static/clair-de-lune.mp3", 33, 33.3, false);
audio2 = read_audio("static/moonlight-sonata.mp3", 33, 33.3, false);

blended_audio = fun(audio1, audio2, "truncate");

sound_list = {audio1, audio2, blended_audio};

audio_plot(sound_list);
```

Function output

```
"File Name: "    "clair-de-lune"    ".mp3"

Number of Channels: 2
Sample Rate: 44100
Duration: 0.30002 seconds
    "File Name: "    "moonlight-sonata"    ".mp3"

Number of Channels: 2
Sample Rate: 48000
Duration: 0.30002 seconds
```

Output graphs



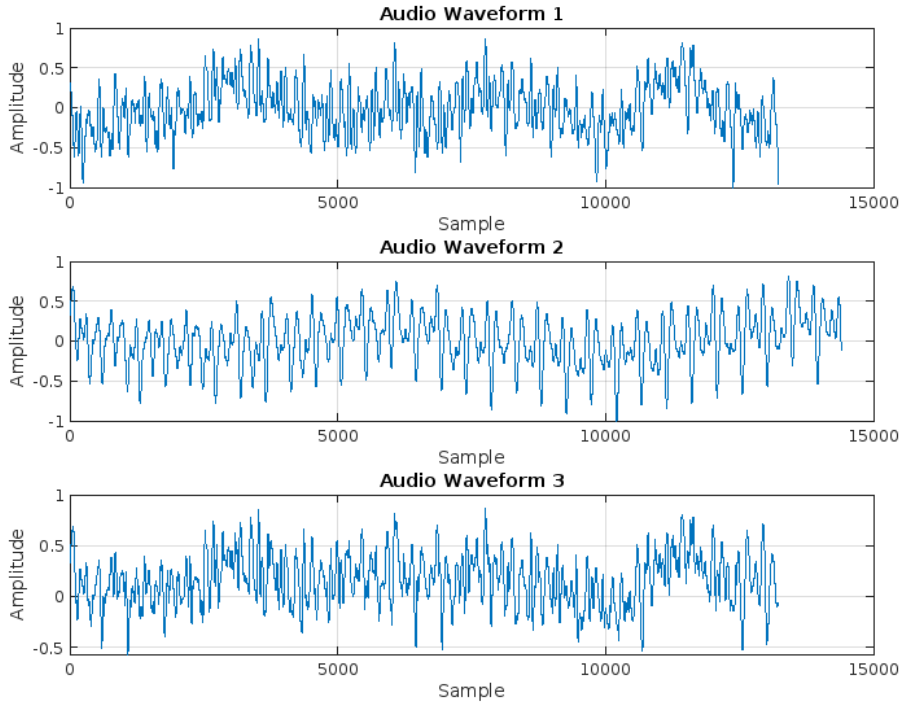
3 Extra funcitonalities

3.1 Function read_audio

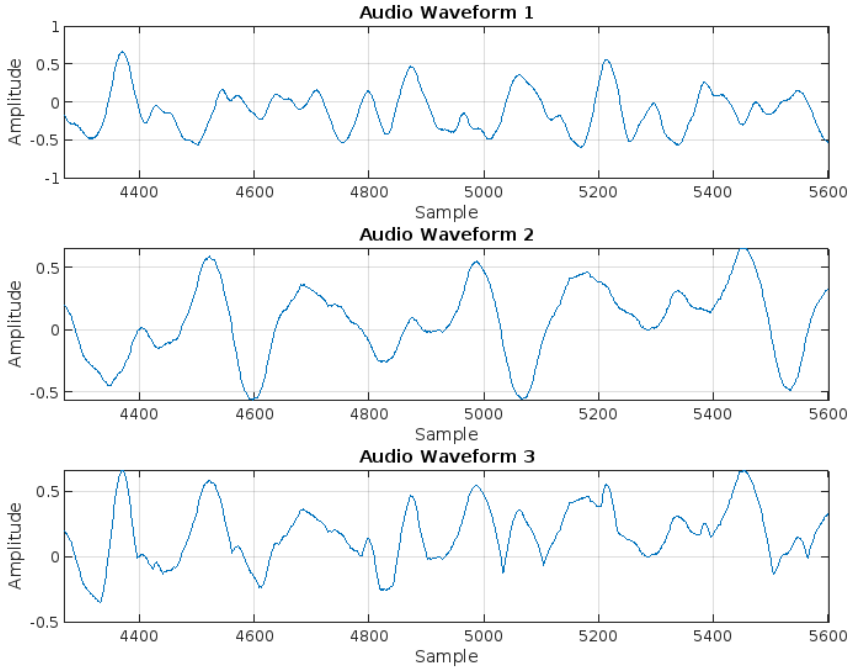
This function has the following extra functionalities:

- The start and stop time to process the audio can be adjusted. In the main file, I am getting data from second 33 till 33.3 of both audio files.
- The last parameter is for audio normalization if specified as true.

Ouput of the function if normalization is set to true



And when zoomed in:



3.2 Source control

Soucre control is enabled for this project and it is located at
https://github.com/longieee/univaq-intro-to-matlab-final-project

3.3 Unit tests

Unit tests are also written for the functions of this project. The unit test code is located inside the `tests` folder.

3.4 Github workflow

A simple Github workflow is also created for this project. The workflow is located at `.github/workflows/matlab_test.yml` and it is triggered on every push to the main branch to automatically run the unit tests.

A possible real-life application: The workflow runs the unit tests and if they pass, it will create a new release with the version number incremented by 1.

4 Listing of all source code

Folder structure

```
.
|____LICENSE
|____tests
| |____test_read_audio.m
| |____test_extendVectors.m
| |____test_fun.m
|____utils
| |____fun.m
| |____read_audio.m
| |____extendVectors.m
| |____audio_plot.m
|____README.md
|____report.tex
|____rendered
| |____audio_result_no_normalize.png
| |____audio_result_normalized.png
| |____project-description.pdf
| |____audio_result_normalized_zoomed.png
| |____report.synctex.gz
| |____report.pdf
| |____results.pdf
| |____coverage.xml
|____.gitignore
|____main.m
|____static
| |____moonlight-sonata.mp3
| |____clair-de-lune.mp3
|____.github
| |____workflows
| | |____matlab_tests.yml
```

Main file (main.m)

```
% Add submodule paths
addpath('utils');
addpath('static');
```

```
4
5  audio1 = read_audio("static/clair-de-lune.mp3", 33, 33.3, false);
6  audio2 = read_audio("static/moonlight-sonata.mp3", 33, 33.3, false);
7
8  blended_audio = fun(audio1, audio2, "truncate");
9
10 sound_list = {audio1, audio2, blended_audio};
11
12 audio_plot(sound_list);
```

Utilitiy functions

Function fun

```
function w = fun(u,v,method)
    %FUN takes as input two vectors ‘u‘ and ‘v‘ and as output provides
    % the vector ‘w‘ which contains in each entry the max between
    % the same entry position of the two vectors.
    %   In the case the length of the 2 input vectors are different, the
    %   argument ‘method‘ will decide how to deal with the situation
    %
    %   - If method="truncate", then the output's length is the length o
    %   the shorter vector
    %   - If method="fill", then the output's length is the length of th
    %   longer vector, and the missing part of the shorter vector is
    %   considered 0
    %
    %   Inputs:
    %       v1 - First input vector (row or column)
    %       v2 - Second input vector (row or column)
    %   Output:
    %       vector ‘w‘ as described above

    % Input validation
    if nargin < 2
        error('fun:NotEnoughInputs', 'At least TWO input argument is req
    end

    % Case non-vectors
    if ~isvector(u)
        error('fun:InvalidInput', 'Input argument must be a vector, rece
    end

    if ~isvector(v)
        error('fun:InvalidInput','Input argument must be a vector, recei
    end

    % Case empty vectors
    if isempty(u) || isempty(v)
    error('fun:InvalidInput', 'Input arguments must not be empty');
    end

    % Provide default value for method argument
    if nargin < 3
```

```
41          method="truncate";
42      end
43      % and then check its validity
44      if method~="truncate" && method~="fill"
45          error('fun:InvalidMethod', 'Method must be one of {truncate, fil
46      end
47
48      % Main logic
49      % Easy case: Same length
50      if length(u) == length(v)
51          w = max(u,v);
52      % 2 vectors are not of the same length
53      else
54          if method=="truncate"
55              min_length = min(length(u),length(v));
56              w = max( ...
57                  u(1:min_length), ...
58                  v(1:min_length) ...
59                  );
60          else
61              [u2,v2] = extendVectors(u,v);
62              w = max(u2,v2);
63          end
64      end
65  end
```

Function read_audio

```
1   function channel1 = read_audio(file_path, start, stop, normalize)
2       %READ_AUDIO Read an MP3 file and return the first channel
3       %   channel1 = read_audio(file_path, start, stop, normalize) reads t
4       %   by file_path from the start sample to the stop sample and return
5       %   If normalize is true, the audio data is normalized to the range
6       %
7       %   Inputs:
8       %       file_path - Path to the MP3 file
9       %       start - Optional start sample (default: 1)
10      %       stop - Optional stop sample (default: end of file)
11      %       normalize - Optional boolean to indicate if normalization is
12      %   Outputs:
13      %       channel1 - First channel of the audio data
14      %   Example usage:
```

```
15      %       channel1 = read_audio('path/to/audio.mp3', 1000, 5000, true)
16
17      % Handle optional arguments
18      if nargin < 2 || isempty(start)
19          start = 1;
20      end
21
22      if nargin < 3 || isempty(stop)
23          info = audioinfo(file_path);
24          stop = info.TotalSamples;
25      end
26
27      if nargin < 4
28          normalize = false;
29      end
30
31      % Read audio with specified range
32      % Convert time in seconds to sample indices if needed
33      info = audioinfo(file_path);
34      fs = info.SampleRate;
35
36      % Convert start form seconds to samples
37      start_sample = max(1, round(start * fs));
38      % Convert stop from seconds to samples
39      stop_sample = min(info.TotalSamples, round(stop * fs));
40
41      [audio_data, fs] = audioread(file_path, [start_sample, stop_sample]);
42      channel1 = audio_data(:, 1);
43
44      if normalize
45          % Normalize the audio data to the range [-1, 1]
46          channel1 = channel1 / max(abs(channel1));
47      end
48
49      % Display the file name
50      [~, file_name, ext] = fileparts(file_path);
51      disp(['File Name: ', file_name, ext]);
52      % Display the number of channels
53      num_channels = size(audio_data, 2);
54      disp(['Number of Channels: ', num2str(num_channels)]);
55      % Display the sample rate
56      disp(['Sample Rate: ', num2str(fs)]);
57      % Display the duration of the audio
```

```
    duration = length(channel1) / fs;
    disp(['Duration: ', num2str(duration), ' seconds']);
end
```

Function extendVectors

```
function [v1, v2] = extendVectors(v1, v2)
    % EXTENDVECTORS Extends the shorter vector to match the longer one.
    %
    %   [v1, v2] = EXTENDVECTORS(v1, v2) takes two vectors and extends t
    %   shorter one by padding with zeros to match the length of the lon
    %   The function preserves the original orientation (row or column).
    %
    %   Inputs:
    %       v1 - First input vector (row or column)
    %       v2 - Second input vector (row or column)
    %
    %   Outputs:
    %       v1 - First vector, extended if necessary
    %       v2 - Second vector, extended if necessary
    %
    %   Example usage:
    %       v1 = [1, 2, 3];
    %       v2 = [4, 5, 6, 7, 8];
    %       [v1_ext, v2_ext] = extendVectors(v1, v2);
    %       % v1_ext: [1 2 3 0 0], v2_ext: [4 5 6 7 8]
    %
    %       v1 = [1; 2];
    %       v2 = [3; 4; 5; 6];
    %       [v1_ext, v2_ext] = extendVectors(v1, v2);
    %       % v1_ext: [1; 2; 0; 0], v2_ext: [3; 4; 5; 6]

    % Validate inputs - consider empty arrays as valid vectors
    if (~isempty(v1) && ~isvector(v1)) || (~isempty(v2) && ~isvector(v2)
        error('Both inputs must be vectors.');
    end

    % Determine the orientation (row or column) for empty vectors
    if isempty(v1)
        % If v1 is empty, use the orientation of v2, or default to row
        if isrow(v2)
            v1 = zeros(1, 0);  % Empty row vector
```

```
        elseif iscolumn(v2)
            v1 = zeros(0, 1);  % Empty column vector
        else
            v1 = zeros(1, 0);  % Default to row vector
        end
    end

    if isempty(v2)
        % If v2 is empty, use the orientation of v1, or default to row
        if isrow(v1)
            v2 = zeros(1, 0);
        elseif iscolumn(v1)
            v2 = zeros(0, 1);
        else
            v2 = zeros(1, 0);
        end
    end

    % Get lengths
    len1 = length(v1);
    len2 = length(v2);
    maxLen = max(len1, len2);

    % Extend the shorter vector while maintaining shape
    if len1 < maxLen
        if isrow(v1) || (isempty(v1) && isrow(v2))
            v1(1, maxLen) = 0;  % Extend row vector
        else
            v1(maxLen, 1) = 0;  % Extend column vector
        end
    end

    if len2 < maxLen
        if isrow(v2) || (isempty(v2) && isrow(v1))
            v2(1, maxLen) = 0;  % Extend row vector
        else
            v2(maxLen, 1) = 0;  % Extend column vector
        end
    end
end
```

Function audio_plot

```
function audio_plot(sound_list)
    %AUDIO_PLOT Takes in a list of audio files and plots their waveforms
    % This function is intended to plot the waveforms before and after t
    % written inside the rest of the script.
    % The "audios" are essentially vectors of the first channel of the a
    % data. The function takes in a list of audio files or just a single
    % plots their waveforms.
    %   Inputs:
    %       sound_list - A cell array of strings, each string is a path
    %   Outputs:
    %       None, but the function will display the waveforms of the aud
    %   Example usage:
    %       audio_plot({vector1, vector2, vector3});

    % Check if the cell array is empty
    if isempty(sound_list)
        error('audio_plot:InvalidInput', 'Input cell array is empty.');
    end

    % If the input is just 1 vector, convert it to a cell array
    if ~iscell(sound_list) && isvector(sound_list)
        sound_list = {sound_list};
    end

    % Check if the input is a cell array
    if ~iscell(sound_list)
        error('audio_plot:InvalidInput', 'Input must be a cell array of
    end

    % Determine number of audio files
    cell_size = size(sound_list);
    num_sounds = cell_size(2);

    % Create a figure with subplots for each audio file
    figure('Name', 'Audio Waveforms', 'NumberTitle', 'off');

       % Create a figure with subplots for each audio file
    figure('Name', 'Audio Waveforms', 'NumberTitle', 'off');
    % Initialize an array to store axes handles
    ax = zeros(num_sounds, 1);

    % Loop through each audio file and plot its waveform
```

```
    for i = 1:num_sounds
        ax(i) = subplot(num_sounds, 1, i);

        % Get the current audio data
        audio_data = sound_list{1,i};

        % Plot the waveform
        plot(audio_data);

        % Add title and labels
        title(['Audio Waveform ' num2str(i)]);
        xlabel('Sample');
        ylabel('Amplitude');

        % Add grid for better visualization
        grid on;
    end

    % Link all axes to share control
    linkaxes(ax, 'x');

    % Adjust the spacing between subplots
    set(gcf, 'Position', [100, 100, 800, 200*num_sounds]);
end
```

Unit tests

Unit test for fun

```
classdef test_fun < matlab.unittest.TestCase
    % TEST_FUN Unit tests for the function `fun.m`.
    %
    % This test suite verifies the correctness of the function `fun` usi
    % the MATLAB Unit Testing Framework.

    properties (TestParameter)
        % Define test parameters for vector inputs
        u = { [1, 3, 5], [1; 3; 5], [1, 3, 5], [1; 3; 5], [], [5] };
        v = { [2, 2, 6], [2; 2; 6], [2, 2], [2; 2], [1, 2, 3], [10] };
        method = { "truncate", "fill" };
    end

    methods (Test)
        function testEqualLengthVectors(testCase)
            % Test case where vectors have the same length
            u = [1, 3, 5];
            v = [2, 2, 6];
            expected = [2, 3, 6];
            actual = fun(u, v);
            testCase.verifyEqual(actual, expected);
        end

        function testEqualLengthColumnVectors(testCase)
            % Test case for equal-length column vectors
            u = [1; 3; 5];
            v = [2; 2; 6];
            expected = [2; 3; 6];
            actual = fun(u, v);
            testCase.verifyEqual(actual, expected);
        end

        function testTruncateMode(testCase)
            % Test truncate mode with different length vectors
            u = [1, 3, 5];
            v = [2, 2];
            expected = [2, 3];
            actual = fun(u, v, "truncate");
            testCase.verifyEqual(actual, expected);
        end
```

```
41
42          function testFillMode(testCase)
43              % Test fill mode with different length vectors
44              u = [1, 3, 5];
45              v = [2, 2];
46              expected = [2, 3, 5]; % Missing parts filled with zero
47              actual = fun(u, v, "fill");
48              testCase.verifyEqual(actual, expected);
49          end
50
51          function testFillModeColumnVectors(testCase)
52              % Test fill mode with column vectors
53              u = [1; 3; 5];
54              v = [2; 2];
55              expected = [2; 3; 5];
56              actual = fun(u, v, "fill");
57              testCase.verifyEqual(actual, expected);
58          end
59
60          function testInvalidInputNonVector(testCase)
61              % Test for non-vector input, should throw an error
62              testCase.verifyError(@() fun([1 2; 3 4], [5 6]), 'fun:Invali
63          end
64
65          function testInvalidMethodArgument(testCase)
66              % Test for invalid method argument, should throw an error
67              testCase.verifyError(@() fun([1, 2, 3], [4, 5, 6], "wrong_me
68          end
69
70          function testSingleElementVectors(testCase)
71              % Test case with single-element vectors
72              u = [5];
73              v = [10];
74              expected = [10];
75              actual = fun(u, v);
76              testCase.verifyEqual(actual, expected);
77          end
78
79          function testEmptyVectorFillMode(testCase)
80              % Test empty vector with fill mode
81              testCase.verifyError(@() fun([], [5 6]), 'fun:InvalidInput')
82          end
83      end
```

```
84 end
```

Unit test for read_audio

```
1  classdef test_read_audio < matlab.unittest.TestCase
2      % TEST_READ_AUDIO Unit tests for the function 'read_audio.m'.
3      %
4      % This test suite verifies the correctness of the function 'read_aud
5      % using the MATLAB Unit Testing Framework.
6
7      properties
8          TestFile
9          TempDir
10     end
11
12     methods(TestMethodSetup)
13         function createTestFile(testCase)
14             % Create a temporary directory and synthetic audio file for
15             testCase.TempDir = tempname;
16             mkdir(testCase.TempDir);
17
18             % Create a simple sine wave audio signal
19             fs = 44100;  % Sample rate
20             t = 0:1/fs:2;  % 2 seconds
21             y = sin(2*pi*440*t)';  % 440 Hz sine wave
22             stereo_y = [y, y*0.5];  % Create stereo by duplicating with
23
24             % Save as a temporary WAV file
25             testCase.TestFile = fullfile(testCase.TempDir, 'sample.wav')
26             audiowrite(testCase.TestFile, stereo_y, fs);
27         end
28     end
29
30     methods(TestMethodTeardown)
31         function cleanupTestFile(testCase)
32             % Clean up temporary files
33             if exist(testCase.TestFile, 'file')
34                 delete(testCase.TestFile);
35             end
36             if exist(testCase.TempDir, 'dir')
37                 rmdir(testCase.TempDir, 's');
38             end
```

```
39          end
40      end
41
42      methods(Test)
43          function testBasicReading(testCase)
44              % Test basic reading functionality
45              channel1 = read_audio(testCase.TestFile);
46              testCase.verifyTrue(isvector(channel1), 'Output should be a
47              testCase.verifyGreaterThan(length(channel1), 0, 'Output shou
48          end
49
50          function testNormalization(testCase)
51              % Test normalization option
52              % Using 0-0.1 seconds instead of sample indices
53              channel1 = read_audio(testCase.TestFile, 0, 0.1, true);
54              testCase.verifyLessThanOrEqual(max(abs(channel1)), 1.0, 'Nor
55
56              % Check if it's actually normalized to max amplitude
57              if ~isempty(channel1) && any(abs(channel1) > 0)
58                  testCase.verifyEqual(max(abs(channel1)), 1.0, 'Output sh
59              end
60          end
61
62          end
63
64          function testTimeBasedInput(testCase)
65              % Test time-based input
66              % Using audioread directly with samples to compare with read
67              info = audioinfo(testCase.TestFile);
68              fs = info.SampleRate;
69
70              % Test with specific time range
71              time_start = 0.5; % 0.5 seconds
72              time_end = 1.0; % 1.0 seconds
73
74              % Calculate corresponding sample range
75              sample_start = round(time_start * fs);
76              sample_end = round(time_end * fs);
77
78              % Get audio directly using sample indices with audioread
79              [audio_data, ~] = audioread(testCase.TestFile, [sample_start
80              expected_channel1 = audio_data(:, 1);
81
```

```
82              % Get audio using read_audio with time-based parameters
83              actual_channel1 = read_audio(testCase.TestFile, time_start,
84
85              testCase.verifyEqual(actual_channel1, expected_channel1, 'Ti
86          end
87
88          function testChannelExtraction(testCase)
89              % Test that only first channel is returned
90              % Time-based parameters (0 to 0.02 seconds)
91              time_start = 0;
92              time_end = 0.02;
93              fs = audioinfo(testCase.TestFile).SampleRate;
94
95              % Calculate sample range
96              sample_start = max(1, round(time_start * fs));
97              sample_end = round(time_end * fs);
98
99              % Get expected first channel directly
100             [audio_data, ~] = audioread(testCase.TestFile, [sample_start
101             expected_channel1 = audio_data(:, 1);
102
103             % Compare with read_audio output
104             actual_channel1 = read_audio(testCase.TestFile, time_start,
105
106             testCase.verifyEqual(actual_channel1, expected_channel1, 'Fi
107         end
108     end
109 end
```

Unit test for extendVectors

```
1  classdef test_extendVectors < matlab.unittest.TestCase
2      % TEST_EXTENDVECTORS Unit tests for the function `extendVectors.m`.
3      %
4      % This test suite verifies the correctness of the function `extendVe
5      % using the MATLAB Unit Testing Framework.
6
7      methods (Test)
8          function testRowVectors(testCase)
9              % Test with row vectors where first is shorter
10             v1 = [1, 2, 3];
11             v2 = [4, 5, 6, 7, 8];
```

```
12              [v1_ext, v2_ext] = extendVectors(v1, v2);
13
14              testCase.verifyEqual(length(v1_ext), length(v2_ext), 'Vector
15              testCase.verifyEqual(v1_ext, [1, 2, 3, 0, 0], 'First vector
16              testCase.verifyEqual(v2_ext, [4, 5, 6, 7, 8], 'Second vector
17              testCase.verifyTrue(isrow(v1_ext), 'Extended vector should m
18          end
19
20          function testColumnVectors(testCase)
21              % Test with column vectors where second is shorter
22              v1 = [1; 2; 3; 4];
23              v2 = [5; 6];
24              [v1_ext, v2_ext] = extendVectors(v1, v2);
25
26              testCase.verifyEqual(length(v1_ext), length(v2_ext), 'Vector
27              testCase.verifyEqual(v1_ext, [1; 2; 3; 4], 'First vector sho
28              testCase.verifyEqual(v2_ext, [5; 6; 0; 0], 'Second vector sh
29              testCase.verifyTrue(iscolumn(v2_ext), 'Extended vector shoul
30          end
31
32          function testMixedOrientations(testCase)
33              % Test with mixed orientations (row and column)
34              v1 = [1, 2, 3];
35              v2 = [4; 5; 6; 7];
36              [v1_ext, v2_ext] = extendVectors(v1, v2);
37
38              testCase.verifyEqual(length(v1_ext), length(v2_ext), 'Vector
39              testCase.verifyTrue(isrow(v1_ext), 'First vector should main
40              testCase.verifyTrue(iscolumn(v2_ext), 'Second vector should
41          end
42
43          function testEqualLengthVectors(testCase)
44              % Test with vectors of equal length
45              v1 = [1, 2, 3];
46              v2 = [4, 5, 6];
47              [v1_ext, v2_ext] = extendVectors(v1, v2);
48
49              testCase.verifyEqual(v1_ext, v1, 'Equal length vectors shoul
50              testCase.verifyEqual(v2_ext, v2, 'Equal length vectors shoul
51          end
52
53          function testEmptyVectors(testCase)
54              % Test with one empty vector
```

```
55              v1 = [];
56              v2 = [1, 2, 3];
57              [v1_ext, v2_ext] = extendVectors(v1, v2);
58
59              testCase.verifyEqual(length(v1_ext), length(v2_ext), 'Vector
60              testCase.verifyEqual(v1_ext, [0, 0, 0], 'Empty vector should
61              testCase.verifyEqual(v2_ext, [1, 2, 3], 'Non-empty vector sh
62          end
63
64          function testNonVectorInput(testCase)
65              % Test with non-vector input, should throw an error
66              testCase.verifyError(@() extendVectors([1, 2; 3, 4], [5, 6])
67          end
68
69          function testSingleElementVectors(testCase)
70              % Test with single-element vectors
71              v1 = [5];
72              v2 = [10];
73              [v1_ext, v2_ext] = extendVectors(v1, v2);
74
75              testCase.verifyEqual(v1_ext, v1, 'Single-element vectors sho
76              testCase.verifyEqual(v2_ext, v2, 'Single-element vectors sho
77          end
78      end
79  end
```

Other files

Github workflow

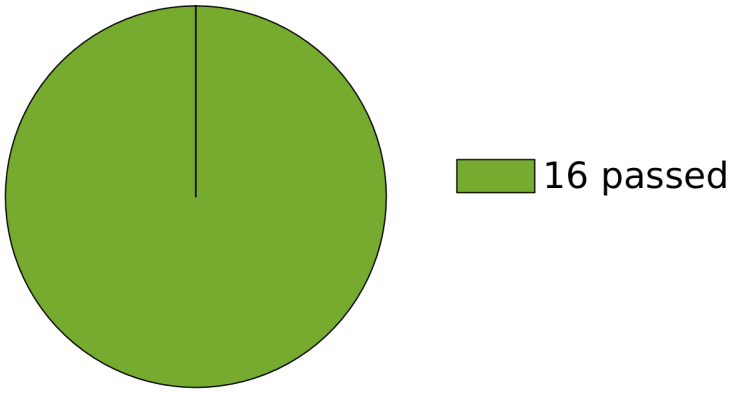
```yaml
name: Run MATLAB Tests with Coverage

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  run-matlab-unittest:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Set up MATLAB
        uses: matlab-actions/setup-matlab@v2
        with:
          release: R2024b

      - name: Run MATLAB tests and collect coverage
        uses: matlab-actions/run-tests@v2
        with:
          source-folder: utils
          select-by-folder: tests
          use-parallel: true
          test-results-pdf: test-results/results.pdf
          code-coverage-cobertura: code-coverage/coverage.xml
          logging-level: detailed
          output-detail: verbose

      - name: Upload Test result Report
        uses: actions/upload-artifact@v4
        with:
          name: test-results
          path: test-results/results.pdf

      - name: Upload Code Coverage Report
        uses: actions/upload-artifact@v4
```

```
41          with:
42            name: code-coverage
43            path: code-coverage/coverage.xml
```

Generated test report

MATLAB® Test Report

Timestamp:	20-Apr-2025 07:49:42
Host:	fv-az1912-101
Platform:	glnxa64
MATLAB Version:	24.2.0.2923080 (R2024b) Update 6
Number of Tests:	16
Testing Time:	0.4500 seconds
Overall Result:	PASSED



Overview

/home/runner/work/univaq-intro-to-matlab-final-project/univaq-intro-to-matlab-final-project/tests/

test_extendVectors ✅✅✅✅✅✅✅	0.3824 seconds
test_fun ✅✅✅✅✅✅✅✅✅	0.0676 seconds

Details

/home/runner/work/univaq-intro-to-matlab-final-project/univaq-intro-to-matlab-final-project/tests/

test_extendVectors

✅ testRowVectors

The test passed.
Duration: 0.2163 seconds

(Overview)

✅ testColumnVectors

The test passed.
Duration: 0.0319 seconds

(Overview)

✅ testMixedOrientations

The test passed.
Duration: 0.0127 seconds

(Overview)

✅ testEqualLengthVectors

The test passed.
Duration: 0.0154 seconds

(Overview)

✅ testEmptyVectors

The test passed.
Duration: 0.0044 seconds

(Overview)

✅ testNonVectorInput

The test passed.
Duration: 0.0991 seconds

(Overview)

testSingleElementVectors

The test passed.
Duration: 0.0026 seconds

(Overview)

test_fun

testEqualLengthVectors

The test passed.
Duration: 0.0076 seconds

(Overview)

testEqualLengthColumnVectors

The test passed.
Duration: 0.0039 seconds

(Overview)

testTruncateMode

The test passed.
Duration: 0.0027 seconds

(Overview)

testFillMode

The test passed.
Duration: 0.0021 seconds

(Overview)

testFillModeColumnVectors

The test passed.
Duration: 0.0034 seconds

(Overview)

testInvalidInputNonVector

The test passed.
Duration: 0.0214 seconds

(Overview)

testInvalidMethodArgument

The test passed.
Duration: 0.0125 seconds

(Overview)

testSingleElementVectors

The test passed.
Duration: 0.0043 seconds

(Overview)

testEmptyVectorFillMode

The test passed.
Duration: 0.0097 seconds

(Overview)

Code coverage report

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<coverage branch-rate="NaN" branches-covered="NaN" branches-valid="NaN"
  <sources>
    <source>/home/runner/work/univaq-intro-to-matlab-final-project/univa
  </sources>
  <packages>
    <package branch-rate="NaN" complexity="NaN" line-rate="0.38889" name
      <classes>
        <class branch-rate="NaN" complexity="NaN" filename="audio_plot.m
          <methods/>
          <lines>
            <line hits="0" number="16"/>
            <line hits="0" number="17"/>
            <line hits="0" number="21"/>
            <line hits="0" number="22"/>
            <line hits="0" number="26"/>
            <line hits="0" number="27"/>
            <line hits="0" number="31"/>
            <line hits="0" number="32"/>
            <line hits="0" number="35"/>
            <line hits="0" number="38"/>
            <line hits="0" number="40"/>
            <line hits="0" number="43"/>
            <line hits="0" number="44"/>
            <line hits="0" number="47"/>
            <line hits="0" number="50"/>
            <line hits="0" number="53"/>
            <line hits="0" number="54"/>
            <line hits="0" number="55"/>
            <line hits="0" number="58"/>
            <line hits="0" number="62"/>
            <line hits="0" number="65"/>
          </lines>
        </class>
        <class branch-rate="NaN" complexity="NaN" filename="extendVector
          <methods/>
          <lines>
            <line hits="9" number="28"/>
            <line hits="1" number="29"/>
            <line hits="8" number="33"/>
            <line hits="1" number="35"/>
```

```
42              <line hits="1" number="36"/>
43              <line hits="0" number="37"/>
44              <line hits="0" number="38"/>
45              <line hits="0" number="40"/>
46              <line hits="8" number="44"/>
47              <line hits="0" number="46"/>
48              <line hits="0" number="47"/>
49              <line hits="0" number="48"/>
50              <line hits="0" number="49"/>
51              <line hits="0" number="51"/>
52              <line hits="8" number="56"/>
53              <line hits="8" number="57"/>
54              <line hits="8" number="58"/>
55              <line hits="8" number="61"/>
56              <line hits="3" number="62"/>
57              <line hits="3" number="63"/>
58              <line hits="0" number="65"/>
59              <line hits="8" number="69"/>
60              <line hits="3" number="70"/>
61              <line hits="1" number="71"/>
62              <line hits="2" number="73"/>
63            </lines>
64          </class>
65          <class branch-rate="NaN" complexity="NaN" filename="fun.m" line-
66            <methods/>
67            <lines>
68              <line hits="9" number="21"/>
69              <line hits="0" number="22"/>
70              <line hits="9" number="26"/>
71              <line hits="2" number="27"/>
72              <line hits="7" number="30"/>
73              <line hits="0" number="31"/>
74              <line hits="7" number="35"/>
75              <line hits="0" number="36"/>
76              <line hits="7" number="40"/>
77              <line hits="3" number="41"/>
78              <line hits="7" number="44"/>
79              <line hits="1" number="45"/>
80              <line hits="6" number="50"/>
81              <line hits="3" number="51"/>
82              <line hits="3" number="54"/>
83              <line hits="1" number="55"/>
84              <line hits="1" number="56"/>
```

```
85              <line hits="1" number="57"/>
86              <line hits="1" number="58"/>
87              <line hits="1" number="59"/>
88              <line hits="2" number="61"/>
89              <line hits="2" number="62"/>
90            </lines>
91          </class>
92          <class branch-rate="NaN" complexity="NaN" filename="read_audio.m
93            <methods/>
94            <lines>
95              <line hits="0" number="18"/>
96              <line hits="0" number="19"/>
97              <line hits="0" number="22"/>
98              <line hits="0" number="23"/>
99              <line hits="0" number="24"/>
100             <line hits="0" number="27"/>
101             <line hits="0" number="28"/>
102             <line hits="0" number="33"/>
103             <line hits="0" number="34"/>
104             <line hits="0" number="37"/>
105             <line hits="0" number="39"/>
106             <line hits="0" number="41"/>
107             <line hits="0" number="42"/>
108             <line hits="0" number="44"/>
109             <line hits="0" number="46"/>
110             <line hits="0" number="50"/>
111             <line hits="0" number="51"/>
112             <line hits="0" number="53"/>
113             <line hits="0" number="54"/>
114             <line hits="0" number="56"/>
115             <line hits="0" number="58"/>
116             <line hits="0" number="59"/>
117           </lines>
118         </class>
119       </classes>
120     </package>
121   </packages>
122 </coverage>
```