Lab - Signals

Group Members	Tamas Orban, Stefan Hududui, Stefan Neciu

Exercise 1

File preview

Let's go back to our vector y, and fill it with numbers from 10 to 100, with a step of 10 between them.

```
>> y = [10 20 30 40 50 60 70 80 90 100] ;
```

Exercise 1. Can you think of another way to generate this vector?

Solution

```
y = (10:10:100)
```

```
>> y = (10:10:100)

y =

    10    20    30    40    50    60    70    80    90   100

>>
```

Exercise 2

Exercise 2. Investigate how the MATLAB function `rand` works. Then, use it to create a vector filled with 10 random numbers, and plot it.
Hint: By typing
`>>help xxx`
you can find out how any MATLAB function (in this example, `xxx`) works and how it can be used. Another way to get more information about a MATLAB function is to type the command name at the MATLAB prompt, then press the F1 key.

Help Source : `help rand`

- `rand(N)` - returns an **N-by-N matrix containing** pseudorandom values drawn from the standard uniform distribution on the open `interval(0,1).`
- `rand(M,N)` or `rand([M,N])` returns an **M-by-N matrix.**
- `rand(M,N,P,...)` or `rand([M,N,P,...])` returns an **M-by-N-by-P-by-... array.**
- `rand` returns a scalar.
- `rand(SIZE(A))` returns an array the same size as A

Solution

```
y = rand(10,1)
```

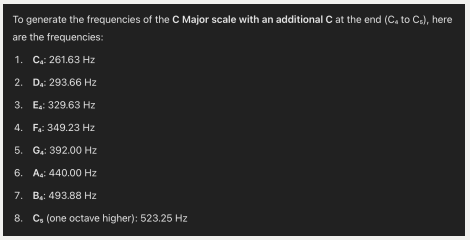
```
>> y = rand(10,1)

y =

    0.8116
    0.5328
    0.3507
    0.9390
    0.8759
    0.5502
    0.6225
    0.5870
    0.2077
    0.3012
```

Exercise 3

Exercise 3. Write a MATLAB program that uses the note_gen function to play a **C-major** scale ascending. To play the **C major** scale ascending, start with the root note C, and play the notes in order: C - D - E - F - G - A - B - C.

Source

- **Chat GPT**



Solution

- `note_gen` **function**
  - `f` : frequency
  - `fs` : sampling frequency
  - `Td` : duration

```
function [s]=note_gen(f,fs,Td)

    t=0:1/fs:Td;
    s=sin(2*pi*f*t);
    sound(s,fs);

end
```

1. Make a vector with `asc` `order`

```
C = 262;
D = 294;
E = 330;
F = 349;
G = 392;
A = 440;
B = 494;
C2 = 524;

y = [C,D,E,F,G,A,B,C2];
```

2. For Loop in `y` with `0.5 duration` , `8000 sampling freq` for each
   - `Problem` : it was playing it at the same time with no pause

- `Solution` : pause() function

🚫 **Source : MATLAB Documentation**

- `help pause`
- `help for`

```
for i = 1:length(y)

    note = note_gen(y(i), 8000, 0.5);
    pause(0.5);

end
```

Exercise 4

> **Exercise 4:** Utter a certain word three times, and record yourself as you do so. Cut the signal into 3 segments, and plot the resulting signals in a mosaic of 3 windows - one window for each segment, placed one above each other - using the MATLAB function `subplot`.

First we declared an audio recorder with:

- 8000 is sampling frequency
- 16 bits per sample
- 1 channel

```
rec = audiorecorder(8000,16,1)
```

Start a recording for 4 secounds and storing it to `rec`

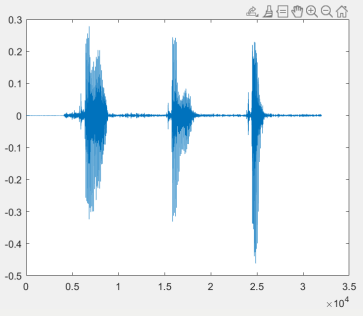
```
recordblocking(rec, 4);
```

Getting recorded audio into speech variable

```
speech = getaudiodata(rec);
```

This is a recording of us saying “hi” 3x times

```
plot(speech);
```



Segmentation into 3 segments

- **Get the full lenght of the audio so we can create segmentation**

```
n = lenght(speech) % we got 32000
```

```
segment_1 = speech3(1:10666);
```

```
segment_2 = speech3(10667:21200);

segment_3 = speech3(21201:32000);
```

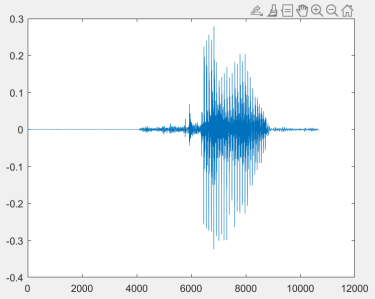
🚫 MATLAB function `subplot` - documentation

```
subplot Create axes in tiled positions.
   H = subplot(m,n,p), or subplot(mnp), breaks the Figure window
   into an m-by-n matrix of small axes, selects the p-th axes for
   the current plot, and returns the axes handle.  The axes are
   counted along the top row of the Figure window, then the second
   row, etc.  For example,

       subplot(2,1,1), PLOT(income)
       subplot(2,1,2), PLOT(outgo)

   plots income on the top half of the window and outgo on the
   bottom half. If the CurrentAxes is nested in a uipanel the
   panel is used as the parent for the subplot instead of the
   current figure.
```

Subplotting each

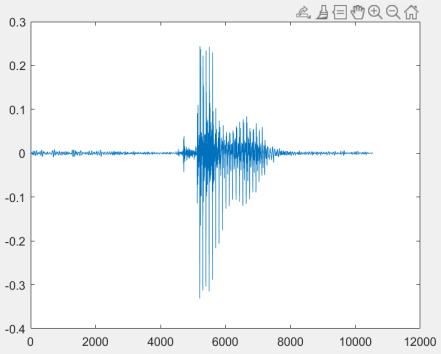
- **Segment 1**

```
plot(segment_1);
```



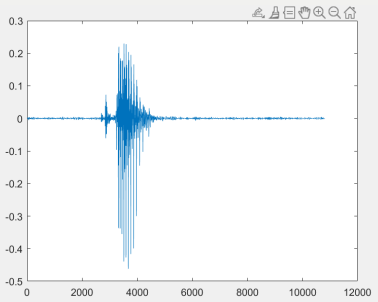
- **Segment 2**

```
plot(segment_2);
```



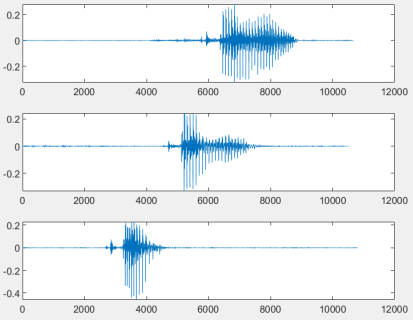
- **Segment 3**

```
plot(segment_3);
```



Subplot

```
subplot(3,1,1);
subplot(3,1,2);
subplot(3,1,3);

plot(segment3);
```



🚫 Exercise 5

Exercise 5a: Try to record a sentence with a low sampling rate (undersampling) and play back the result. Observe and describe the effect. Explain why this effect happens.

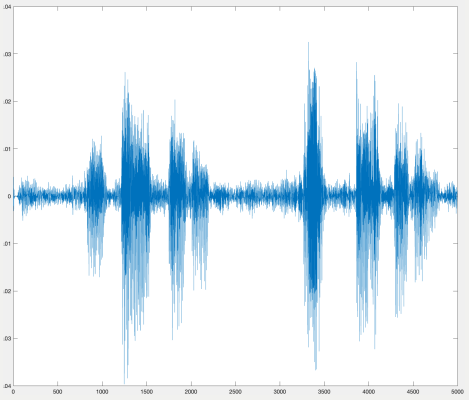
Exercise 5b: Try to record a sentence, and then increase or reduce the playback rate and observe and describe the effect. Explain why this effect happens.

Exercise 5c: Modify your script so that the recording is played in reverse. You can search the Internet for a MATLAB function to reverse the sound vector, if you need to.

a.) - Low sampling rate - `undersampling`

1. **Record sentence with `low sampling rate` for `5 secounds` and plot + play**
   - Sampling rate: `500 Hz`

```
rec = audiorecorder(1000,16,1)
recordblocking(rec, 5);
speech = getaudiodata(rec);

plot(speech);
sound(speech,1000);
```



Plot of recording : *"I'm saying a sentence"*

🚫 **Observation: The playback is**

- **not understandable**
- **very deep voice (mumble)**
- **also the playback is very slow**

> *observed effects are primarily due to **aliasing** and **loss of high-frequency content** due to undersampling.*

b.) - Increased playback speed

1. **Record sentence**

```
rec = audiorecorder(8000,16,1)
recordblocking(rec, 5);
speech = getaudiodata(rec);
```

2. **Increase playback speed**

```
sound(speech, 16000);
```

🚫 **Observation:**

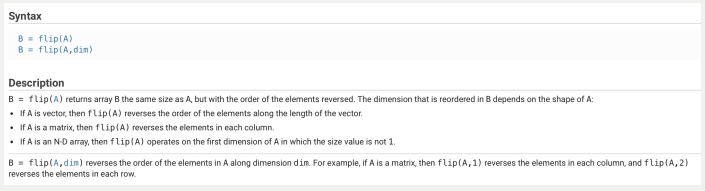
- The playback voice good `very high`
- by `increasing` the sampling rate in `sound()` our **playback speed** `increased`

---

c.) - Play in reverse

1. **Modify script so we play it in reverse**

🚫 **Source:** https://nl.mathworks.com/help/matlab/ref/flip.html



Syntax

```
B = flip(A)
B = flip(A,dim)
```

Description

`B = flip(A)` returns array B the same size as A, but with the order of the elements reversed. The dimension that is reordered in B depends on the shape of A:

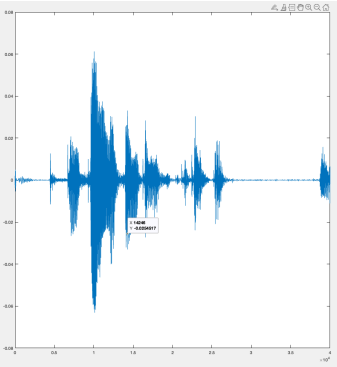
- If A is vector, then `flip(A)` reverses the order of the elements along the length of the vector.
- If A is a matrix, then `flip(A)` reverses the elements in each column.
- If A is an N-D array, then `flip(A)` operates on the first dimension of A in which the size value is not 1.

`B = flip(A,dim)` reverses the order of the elements in A along dimension `dim`. For example, if A is a matrix, then `flip(A,1)` reverses the elements in each column, and `flip(A,2)` reverses the elements in each row.

```
reversed = flip(speech);
sound(reversed);
```

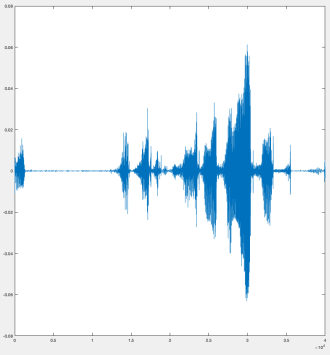
Original

```
plot(speech);
```



Reversed

```
plot(reversed);
```



Exercise 6

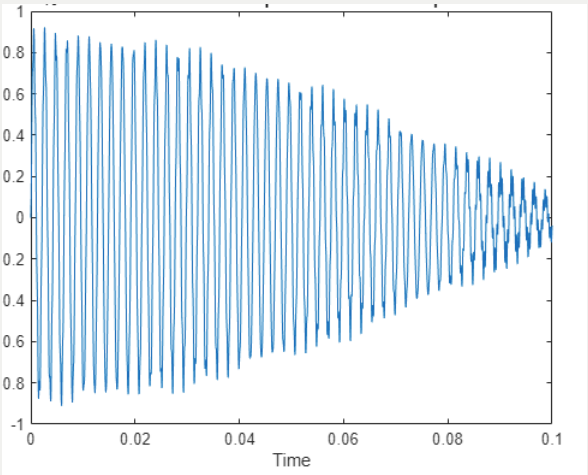
> **Exercise 6.** Change the x-axis so that you can see the time in seconds, then plot the acquired signal against the time axis again. Can you determine the signal frequency from this plot? Compare it with the A note frequency.

1. **First Step :** We need a time vector of `0 sec` to `lenght of rec` divided by the `sampling frequency`

```
n = length(y);
t = (0:n-1) / 8000;
```

2. **Plot**

```
plot(t, rec);
xlabel('Time');
```



3. **Determine frequency**

To find the frequency of the signal, we can `zoom in` on a short segment and `calculate` the `period` by observing the `time between` simular peaks and drops

- We can see every `0.02` `sec` → `9` **idendical period, which results in:**
  - `450 Hz`
  - $T = 0.2/9$ → `0.00222 secound`
  - $f = 1/T = 1/0.00222$ → around `450 Hz`

4. **Comparison**

- The frequency of the **A note is approximately** `440 Hz`.

Note	Octave 0	Octave 4
G	24.50 Hz	392 Hz
G#/Ab	25.96 Hz	415.30 Hz
A	**27.50 Hz**	**440 Hz**
A#/Bb	29.14 Hz	466.16 Hz

- **So in comparison our result has `10 Hz` difference**

Exercise 7

Exercise 7. Write MATLAB code that first generates a square green image, waits 10 seconds, then changes the square colour to blue, waits 10 seconds and changes this blue image to a red one, and finally waits 10 seconds and changes the square to yellow.

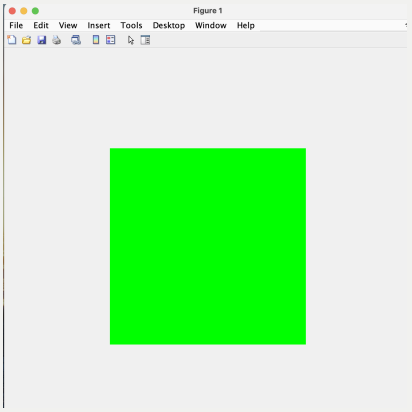
Solution

1. First we initailised a `200 x 200 x 3` **black square**

```
x = zeros(200,200,3);
```

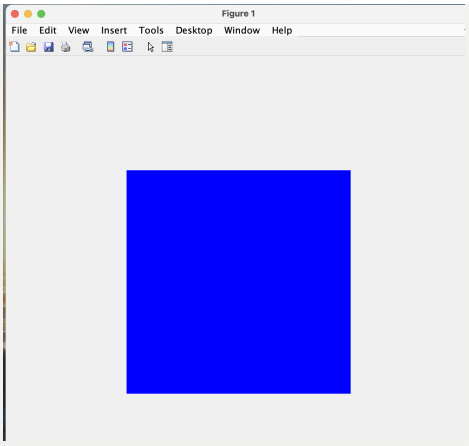
2. **Green Square:** We changed the 2 `row` in the **3D Array** which stands for green and modify it `255`
   a. Display : `imshow()`
   b. Waits 10 seconds : `pause(10)`

```
x(:, :, 2) = 255;
imshow(x);
pause(10);
```



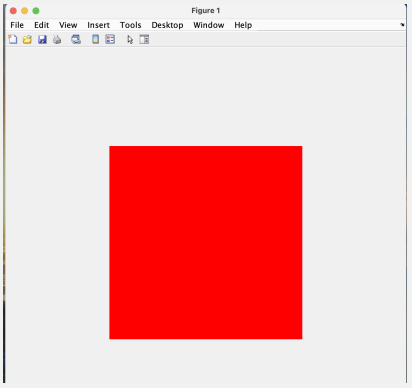
3. **Blue Square:** We changed the 2 `row` in the **3D Array** which stands for green and modify it `255`

```
x(:, :, 2) = 0;
x(:, :, 3) = 255;
imshow(x);
pause(10);
```



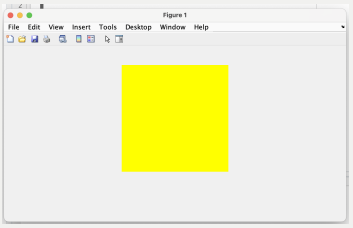
4. **Red Square:** We changed the 2 `row` in the **3D Array** which stands for green and modify it `255`

```
x(:, :, 3) = 0;
x(:, :, 1) = 255;
imshow(x);
pause(10);
```



5. **Yellow Square:** We changed the 2 `row` in the **3D Array** which stands for green and modify it `255`

```
x(:, :, 2) = 255;
imshow(x);
```



Full Code

```
function [] = ex6()

    x = zeros(200,200,3);
    x(:, :, 2) = 255;
    imshow(x);
    pause(10);

    x(:, :, 2) = 0;
    x(:, :, 3) = 255;
    imshow(x);
    pause(10);

    x(:, :, 3) = 0;
    x(:, :, 1) = 255;
    imshow(x);
    pause(10);

    x(:, :, 2) = 255;
    imshow(x);

end
```

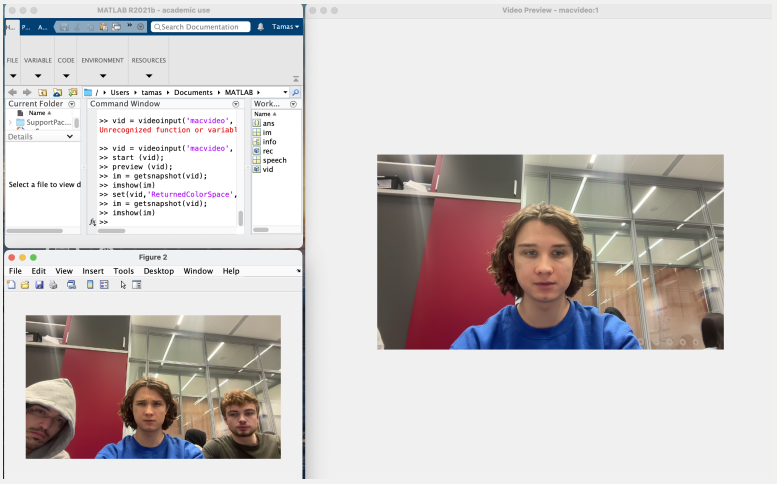
Exercise 8

Exercise 8. Acquire an RGB image using your camera and visualize it. Make some investigations to see how you can convert it to black-and-white image. Make a plot where you show your original image, a grayscale version and a black-and-white version of the same image.

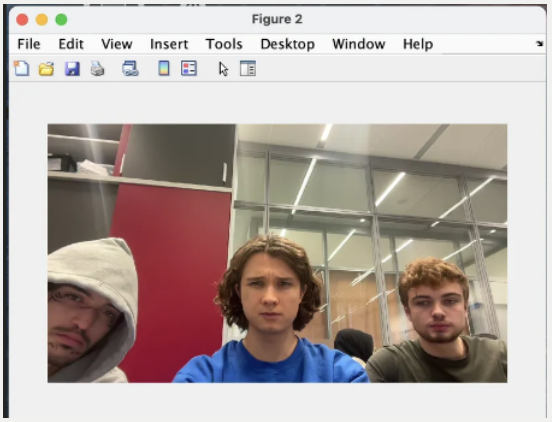
Solution

1. **Acquire Image**

```
vid = videoinput('macvideo', 1);
start (vid);
preview (vid);
```



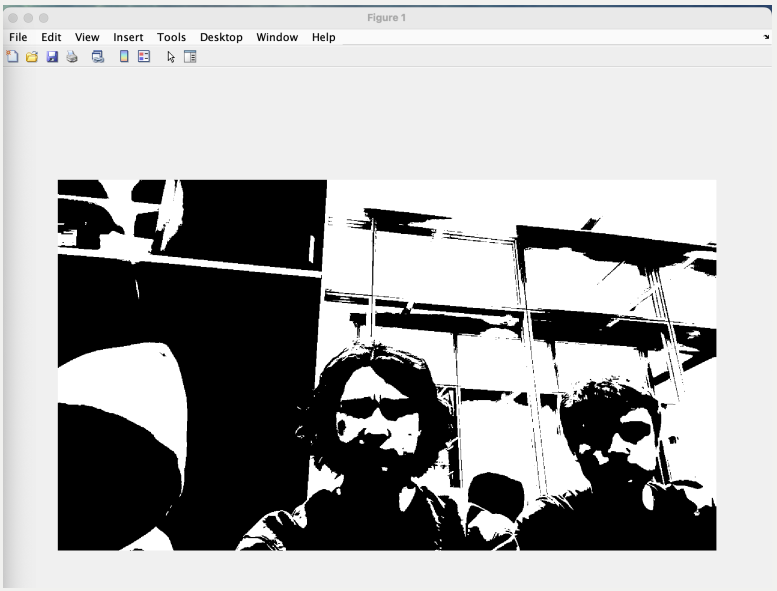
```
im = getsnapshot(vid);
imshow(im);
```



2. **Conversion to *black-and-white***

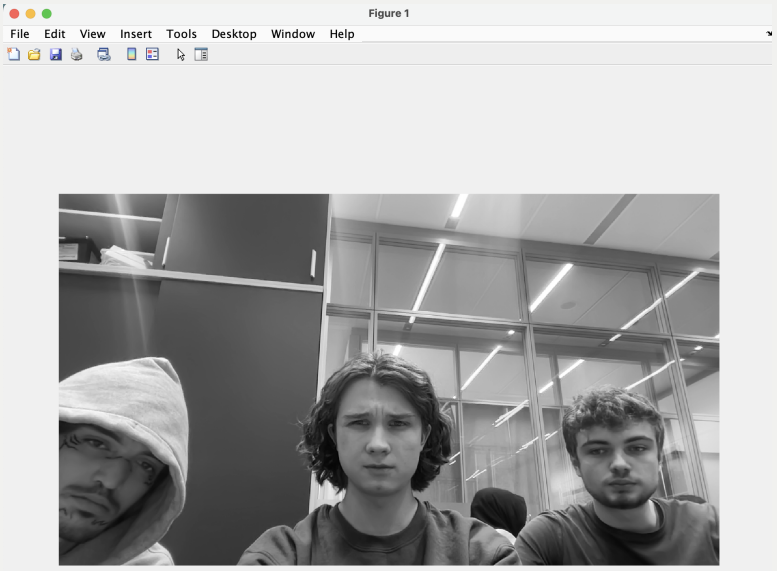
```
binaryImage = imbinarize(im);
```

```
imshow(binaryImage);
```



3. **Conversion to *Grayscale***

```
grayImage = rgb2gray(im);
imshow(grayImage);
```



4. **Original** & **Grayscale** & **Black-and-white**

> **Create a figure so we can subplot all types**
>
> ```
> figure;
> ```

- **Subplot original**

```
subplot(1, 3, 1);
imshow(im);
title('Original Image');
```

- **Subplot black-and-white**

```
subplot(1, 3, 2);
imshow(grayImage);
title('Grayscale Image');12
```

- **Subplot grayscale**

```
subplot(1, 3, 3);
imshow(binaryImage);
title('Black-and-White Image');
```

Final Result