SIGNALS IN 1 DIMENSION (1D).

Signals in Matlab are handled with vectors (arrays).
Signal values are accessed using indexes of the array.

Synthetic signals.

Task 1:
Simple signals.
See how we can create a signal named "Constant" with 10 values (= samples) where all samples have the same value 1. The signal is displayed in a figure.

```
>> Constant = ones (1,10); % create signal with command "ones"
>> figure (1); subplot (3,1,1); stem (0: 9,Constant); % plot the signal
```

Let´s create a ramp signal named "Ramp" with the values 1,2,3,4,5,6,7,8,9,10. Plot the signal in the same figure as the signal before, but in a separate sub-window.

```
>> Ramp = 1:1:10; % create signal
>> figure(1); subplot(3,1,2); stem(0:9,Ramp); % plot the signal in another sub-window
```

Let´s create a new signal called ConstantRamp that consists of the "Constant" followed by the "Ramp" signals created above.

```
>> ConstantRamp = [Constant, Ramp]; %concatenate the two signals
>> figure (1); subplot (3,1,3); stem (0: length (ConstantRamp) -1, ConstantRamp);
```

Read values of the signal "ConstantRamp" in positions 0, 5 and 13.
Also read and display the values from position 8 to 14.

```
>> ConstantRamp (1) % position 0
>> ConstantRamp (6) %position 5
>> ConstantRamp (14) % position 13
>> ConstantRamp (9 :15) % positions 8-14
```

Did you get the right values?

How does indexing work the Matlab?
Note that we cannot access to the index 0 of a vector in Matlab

****

task 1.1
Create a "double-ramp" signal called DoubleRamp with values:
0,1,2,3,4,5,6,7,8,9,10,9,8,7,6,5,4,3,2,1,0

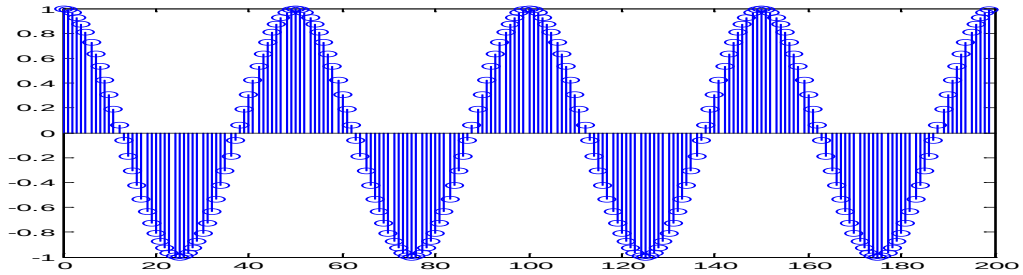
Create a new signal called NewRamp from the signal DoubleRamp using indexing.
NewRamp should have values 7,8,9,10,9,8,7.

Plot the two signals in the same figure.

****

Task 2:
Cosine / Sine signals.
Create a cosine signal with a length of 200 samples named “Cos1”, with amplitude = 1, frequency = 200 Hz, phase angle nf = 0, which is sampled with a sampling frequency of 10 kHz. The signal is periodic with period N = 50

```
>> A = 1; %amplitude
>> F = 200; %frequency
>> Fs = 10000; % sampling frequency
>> nf = 0; % phase angle, integer in the range [0, 50]
>> n = 0: 199; % index of samples (200 samples)
>> Cos1=A*cos(2*pi*(F/Fs)*(n + nf)); % cosine signal
>> figure(2); subplot(3,1,1); stem(0:199,Cos1)
```



****

task 2.1
Changing frequency of the cosine signal.

Halve the frequency to f = 100 Hz and plot with subplot (3,1,2) in Figure 2.

Double the frequency f = 400 Hz and plot with subplot (3,1,3) in Figure 2.

How does the appearance of the cosine signal change when decreasing or increasing the frequency?

Now change the phase angle nf.

Plot the cosine signal with phase angle nf = 0 in figure 21 with the subplot (3,1,1), then a cosine signal with phase angle nf = -8 with subplot (3,1,2) and a cosine signal with phase angle nf = + 8 with subplot (3,1,3) all in the same figure 21

How does the appearance of the cosine signal change when the phase angle is changed?

Now change the amplitude of the cosine signal to 2 and plot in figure 22 the two cosine signals with amplitude 1 and amplitude 2 (all other parameters the same for both)

How is the signal affected by changes in the amplitude?

****

Remember: To get help for functions in Matlab, type: `help + function name` (e.g. >> `help stem`)

Listen to cosine signals with different amplitude and frequency.
Use the following code for this purpose (make sure that you have the audio volume on):

```
>> n=0:5000;
>> Fs=8192;
>> F1=500; %low frequency
>> F2=1000; %high frequency
>> A1=0.1; %low amplitude
>> A2=1; %high amplitude
>> S1=A1*cos(2*pi*(F1/Fs)*n);%low amplitude and low frequency
>> S2=A2*cos(2*pi*(F1/Fs)*n);%high amplitude and low frequency
>> S3=A1*cos(2*pi*(F2/Fs)*n);% low amplitude and high frequency
>> sound([S1 S2],Fs); play signal; variation in amplitude
>> sound([S1 S3],Fs);% playback signal; variation in frequency
```

****

task 2.2
Generate more signals and play with them using different amplitudes and frequencies (the amplitude must be <1, and the frequency <4000).

What does represent the tone and amplitude of the cosine signals?

Why the frequency must be approx. below 4000?

****

Task 3:
How can you compare if two signals are similar or different?
It can be made with the normalized scalar product (correlation).
The scalar product of two signals is calculated by:

1) Element-wise multiplication of the values of the two signals, and then
2) sum of these products.

To normalize the scalar product to the range [-1,1], the result of the above operation is divided by the norm (length) of the two input signals.

Example in Matlab:

```
>> Signal1=[1 -1 1 -1 1 -1];
>> Signal2=[-1 1 -1 1 -1 1];
>> Produkter=Signal1.*Signal2; % operation 1) above
>> %(the operator .* represents element-wise multiplication of the two vectors)
>> Produkter
>> SkalProd=sum(Produkter);  % operation 2) above
>> SkalProd
>> norm1=norm(Signal1);  %length signal1
>> norm2=norm(Signal2);  %length signal2
>> NormSkalProd=SkalProd/norm1/norm2; % or SkalProd/(norm1*norm2)
>> NormSkalProd
```

Remember: To get help for functions in Matlab, type: `help + function name` (e.g. >> `help stem`)

****

task 3.1

Verify manually the values obtained in Produkter and SkalProd

Calculate manually the scalar product of "Signal1 with itself" and check the result with Matlab.

How do you interpret the values of the normalized scalar products that you obtain in each case? Which value of the normalized scalar product corresponds to "the most similar" signals, and which to the "the most different" signals?

****

Create a cosine signal named "Cosinus" with A = 1, F = 200, Fs = 10000, nf = 0 which is 200 samples long. The signal period is N = 50, i.e. the signal repeats itself after 50 samples.

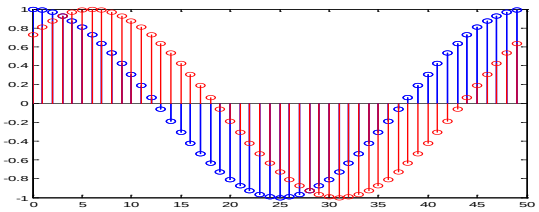
```
>> n=0:199;
>> Cosinus=1*cos(2*pi*(200/10000)*n);
```

From this signal create a new signal called "HelPeriod" that includes one period of the cosine signal. Plot the signal.

```
>> HelPeriod=Cosinus (1:50); % one period of signal Cos
>> figure(3);stem(0:49,HelPeriod); % plot signal HelPeriod
```

Create a period of the cosine signal with a different phase angle (offset).
Plot the signal HelPeriod and the new signal with offset in the same graph and calculate the similarity measure between the two signals using the normalized scalar product.

```
>> nf=6; % offset in the number of samples in the interval [0, 50]
>> CosF=1*cos(2*pi*(200/10000)*(n-nf)); %
>> CosF=CosF(1:50); %one period
>> figure(31);  stem(0:49,HelPeriod); hold on; stem(0:49,CosF,'r'); hold off;
>> NormSkalProd=sum(HelPeriod.*CosF)/norm(HelPeriod)/norm(CosF)
```



****

task 3.2
Compute a similarity measure at different shifts: 0,6,12,18,24,30,36,42 and 48

For each displacement, plot HelPeriod and the shifted signal in the same graph.

Based on the plot, explain how to interpret the value of the similarity metric when you compare the two signals.

****

Remember: To get help for functions in Matlab, type: `help + function name` (e.g. >> `help stem`)

Real signals.

Task 4.
Now we will record a voice signal via the computer's sound card and store it in the vector "Speech":

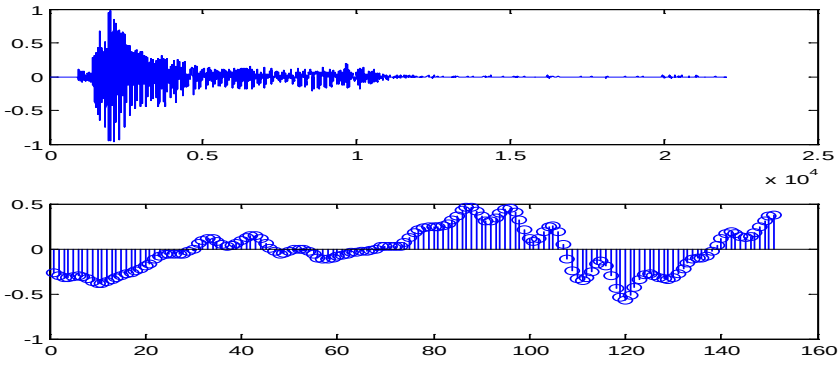
```
>> Fs=22050;
>> bits=8;
>> channels=1;
>> duration=1; %1 second of recording
>> r=audiorecorder(Fs, bits, channels); %set up recording
>> record(r, duration); %record; talk to the microphone now!
>> p=play(r); %play signal
>> Speech =getaudiodata(r); %save signal in vector Speech
```

If you are using a lab computer without a microphone, execute the next commands in Matlab:

```
>> Fs = 12500;
>> Speech = audioread('s1.wav')
```

Plot the voice signal and parts of it, and show single values of the signal Speech.

```
>> figure(4);subplot(2,1,1);plot(Speech);
>> figure(4);subplot(2,1,2);stem(Speech (2500:2650));
>> Speech (2500:2520)
>> Speech (2510)
```



(you will obtain a different figure, based on your own recording)

****

task 4.1

Describe in your own words how a signal looks like in Matlab (this is how a signal looks like in all digital systems). Try that your description contains the word "vector", "values", or "signal duration".

Note the relation between Fs and the x-axis limit of the first subplot

****

Remember: To get help for functions in Matlab, type: `help + function name` (e.g. >> `help stem`)

SIGNALS IN 2 DIMENSIONS (2D): IMAGES.

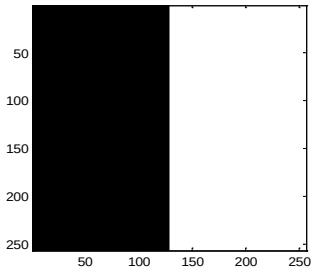
Images in Matlab are handled with matrixes.

Synthetic images.

Task 5:

Create an image of size 256x256 with the left half in black, and the right half in white:

```
>> BlackWhite=ones(256,256)*255;
>> BlackWhite (1:256,1:128)=zeros(256,128);
>> figure(5); imagesc(BlackWhite); colormap(gray); truesize; %plot the image
```



Read the pixel values of the positions (row, column) = (128,50) and = (128,200) in the BlackWhite matrix as:

```
>> BlackWhite (128,50)
>> BlackWhite (128,200)
```

Check the values by invoking the command

```
>> impixelinfo
```

and pointing the mouse in the black and white parts of figure 5. Note that (X,Y)↔(column,row).

Read a 10x10 area in the matrix BlackWhite in the black area, in the border between black and white, and in the white area:

```
>> O1= BlackWhite (123:132,45:54)
>> O2= BlackWhite (123:132,124:133)
>> O3= BlackWhite (123:132,195:204)
```

Which part of the image corresponds to the three matrixes?

Calculate the mean value of the pixels of each area.

The mean value is a description (= attribute) of each area of the image: m1 for area 1, m2 for area 2 and m3 for area 3.

```
>> m1=mean(mean(O1))%average of area 1
>> m2=mean(mean(O2))% average of area 2
>> m3=mean(mean(O3))% average of area 3
```

Does the average value of each area seem to be OK?

Task 6:
Classification of an image area.
Any 10x10 area from the image BlackWhite can be determined to belong to (= be of the class of) the reference-areas 1, 2 or 3 with the help of the area mean m (= property of the selected 10x10 area).

The classification is done by comparing the mean value m with the three averages: m1, m2 and m3. The comparison can be done by three distances: d1 = distance between m and m1, d2 = distance between m and m2, and d3 = distance between m and m3.

The minimum distance is the "winner", i.e. if m is closest to m1, the area is labelled as “area 1” or “mostly black”; if is closest to m2, it is labelled as “area 2” or “border”; and so on.

Display help of command MinDistClass (file included with this lab exercise):
>> help MinDistClass

and read the text that pops up on the screen so you understand how to use the command to classify a 10x10 area of an image by using the average of the pixels in the area. The area is classified to belong to the reference areas "black" (area 1), "white" (area 3) or "border" (area 2).

Test various areas of the image using the MinDistClass function.

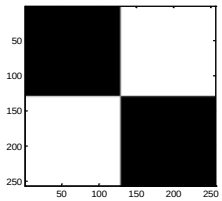
MinDistClass.m is an example of how you can make your own functions in Matlab.

Study the code in the Matlab editor and identify:

- *The description (= property) of the reference areas, "black", "white" and "border".*
- *How the distance is calculated between the selected area and the three reference areas "black", "white" and "border".*
- *How the decision is made.*

****

task 6.1
Create a 256x256 image as:



Test the classification of various areas of the image using the command MinDistClass.

****

Remember: To get help for functions in Matlab, type: `help + function name` (e.g. >> `help stem`)

Real images.

Task 7.
Now we will take a picture via the web-camera (Microsoft LifeCam VX-1000, format RGB24_320x240) using the "Image Acquisition Tool" (parameters: 1 frame, log to memory, 1 trigger and manual) and save in Matlab's working memory (Export Data to MATLAB workspace).

>> imaqtool; % start up the Image Acquisition Tool

Take a picture of your face and save in Matlab's working memory with the variable name: "Face". Verify that the variable is in the "workspace" with format = 240x320x3 uint8.

In the case that you do not have a webcam, import the image attached with this lab to Matlab workspace by introducing the following commands.

>> Face=imread('Face.png'); % import the image file "Face.png" to the workspace

Convert the image to grayscale, and display it in a figure window.

```
>> GI=double(rgb2gray(Face)); %convert image to grayscale
>> figure(1);imagesc(GI);colormap(gray);truesize; %display in figure=1
```

You can also capture a picture with another method, but the camera needs to warm up to reproduce colors in a good way (we recommend using imaqtool as above). This is done with the command "getsnapshot", which immediately returns one single image frame, but it results in poorer color reproduction.

In order to "make contact with" the web-camera, invoke the following commands:

```
>> vid=videoinput('winvideo',1); %set up shooting
>> preview(vid); %test if it works
>> closepreview(vid); %close
```

Take for example a picture of your face with the variable name "Face1" by:

```
>> bild=getsnapshot(vid);
>> Face1=double(bild(:,:,1));% convert to grayscale
>> figure(7);imagesc(Face1);colormap(gray); truesize;
```

****

task 7.1

Cut out the left eye in the picture face. The command looks like this (the range of pixels will be different for your particular picture, you will have to find out which range is appropriate for you):
>> voga= Face (65:85,87:107);

To make it easier, you can invoke the command
>> Impixelinfo;

Remember: To get help for functions in Matlab, type: `help + function name` (e.g. >> `help stem`)

to find the center point of the left eye

Now compute the similarity measure using the normalized scalar product when you compare the left eye with himself, the left eye with the right eye, and the left eye with some other areas of the picture face. Make use of the command NormSC included with this lab exercise (call >> help NormSC to get a description of this command).

Compare the left eye and the right eye, but flip one of them in left/right direction with the Matlab command "fliplr", and see if it makes a difference.

Write down the similarity measure at the various comparisons you make and comment on the results.

****