Signal Analysis

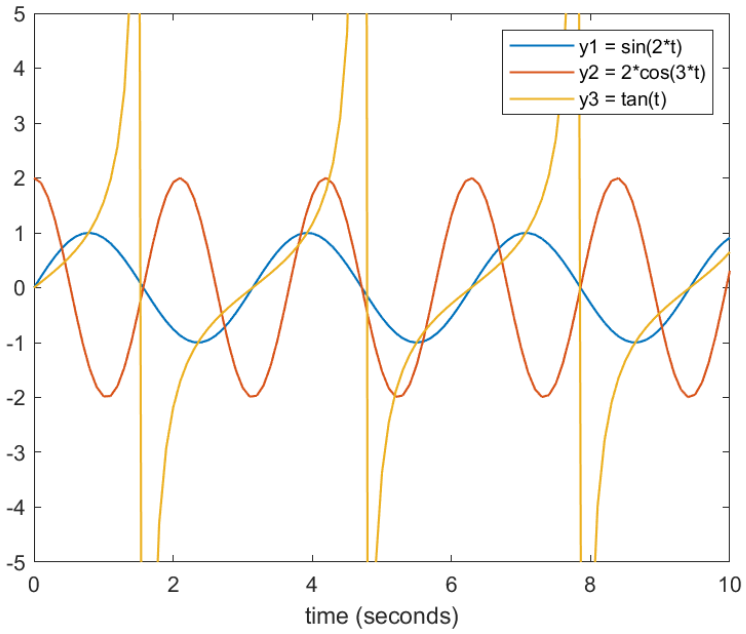
Assignment 1

Dimitar Dimitrov
Radboud University

S1018291
28/Jan/2021

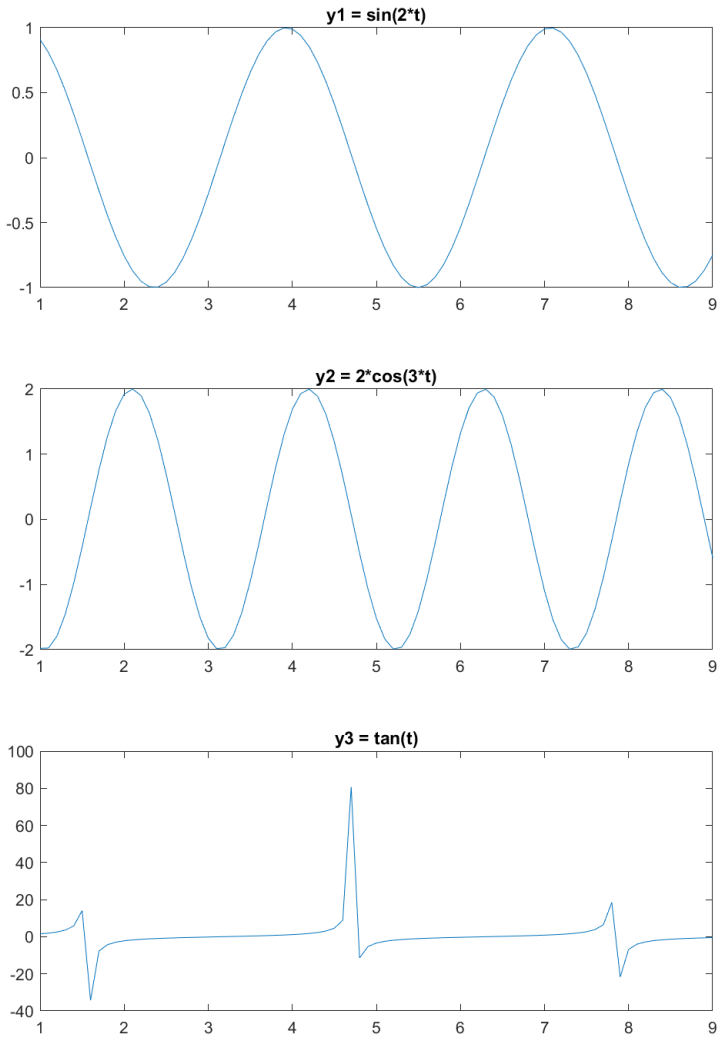
1.8 y1y2y3

```
% plot t against y1, y2 and y3, set the width of the function lines to 1
plot(t,[y1;y2;y3], 'LineWidth', 1)
% limit the y-axis to display only values between -5 and 5
ylim([-5 5])
% show a legend describing each function
legend('y1 = sin(2*t)', 'y2 = 2*cos(3*t)', 'y3 = tan(t)')
% show label for the x-axis
xlabel('time (seconds)')
% save the graph as an image
saveas(gcf,'y1y2y3.png')
```




```
% populate first plot out of a (3,1) subplot space
subplot(3,1,1)
% plot t against y1
plot(t,y1)
% limit the x-axis to show only values between 1 and 9
xlim([1 9])
% add a title
title('y1 = sin(2*t)')
% populate second plot out of a (3,1) subplot space
subplot(3,1,2)
```

```
% plot t against y2
plot(t,y2)
% limit the x-axis to show only values between 1 and 9
xlim([1 9])
% add a title for the subplot describing the function
title('y2 = 2*cos(3*t)')
% populate third plot out of a (3,1) subplot space
subplot(3,1,2)
% add a title for the subplot describing the function
plot(t,y3)
% limit the x-axis to show only values between 1 and 9
xlim([1 9])
% add a title for the subplot describing the function
title('y3 = tan(t)')
% save graph as an image
saveas(gcf, 'separate.png')
```



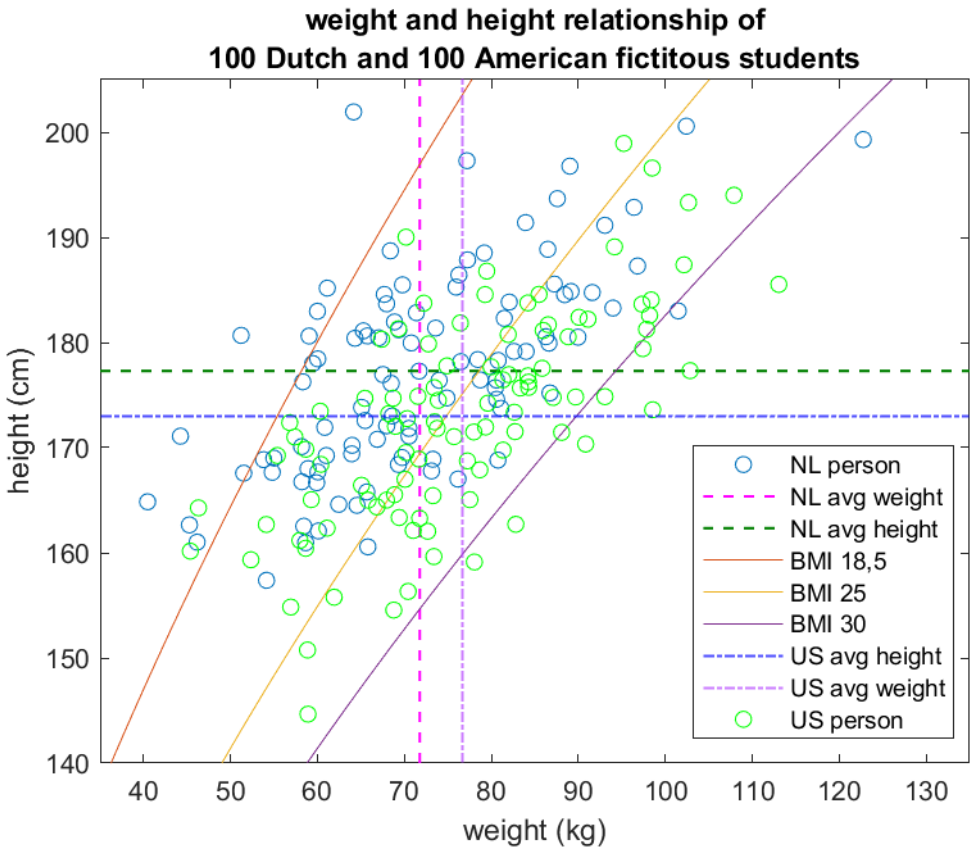
1.9 Students BMI

```
% load the dataset
load StudentsNL
% plot weightNL against lengthNL, use circles
plot(weightNL, lengthNL, 'o')
% add labels and a title
xlabel('weight (kg)')
ylabel('height (cm)')
title('weight and height of 100 fictitious Dutch students')
% make sure we don't lose the plot once we start plotting other things
hold on
% add lines to show the average weight and height
xline(mean(weightNL), 'm--', 'LineWidth', 1) %magenta ---
yline(mean(lengthNL), 'g--', 'LineWidth', 1) %green ---
% create a height array between 140 and 210 in steps of 0.5
height = 140:0.5:210
% create BMI arrays corresponding to the values in the height array
BMI18_5 = height.^2/100^2*18.5
BMI25 = height.^2/100^2*25
BMI30 = height.^2/100^2*30
% plot the BMIs
plot([BMI18_5; BMI25; BMI30], height, 'LineWidth', 1)
% add a neat legend in the bottom right corner so it obstructs less data
legend('person', 'average weight', 'average height', 'BMI 18,5', 'BMI 25', 'BMI
↪  30', 'Location', 'southeast')
% sum the boolean vector of people with BMI>30 to get nr of obese dutch people
sum((weightNL./((lengthNL./100).^2))>30)
    ans =
            2
```

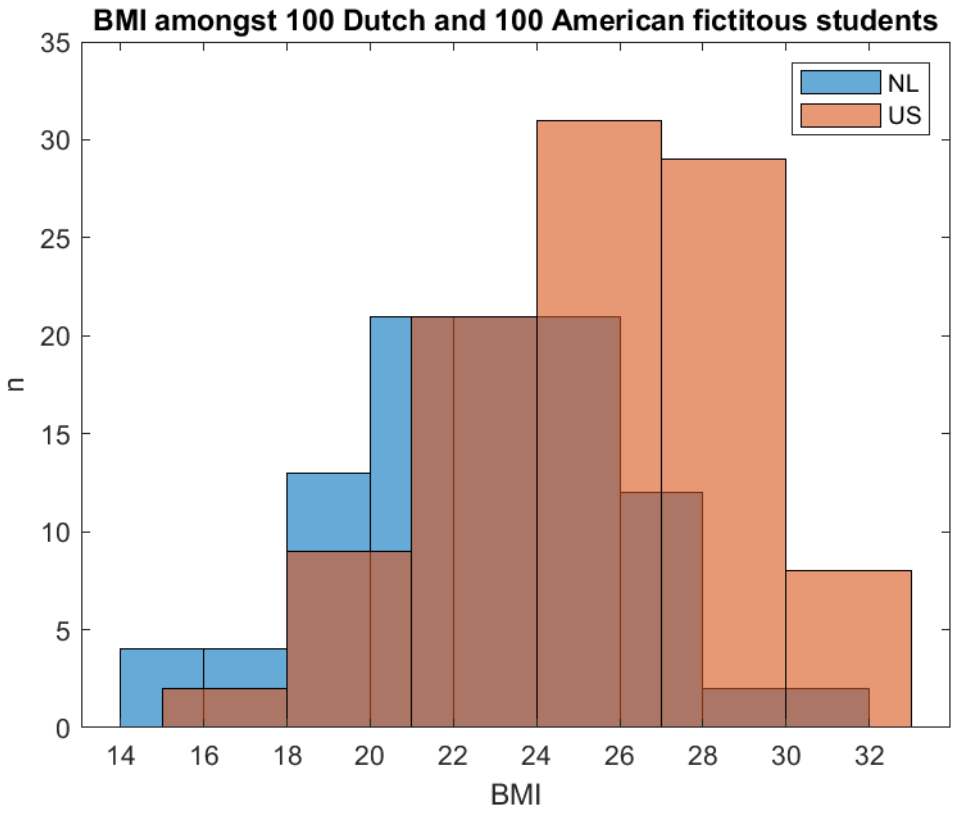
- there are 2 obese dutch students

```
% now do all of these things but for the american database
load StudentsVS
plot(weightVS, lengthVS, 'go') % green circles
xline(mean(weightVS), 'c-.', 'LineWidth', 1) % cyan -.-.
yline(mean(lengthVS), 'k-.', 'LineWidth', 1) % black -.-.
legend('NL person', 'NL avg weight', 'NL avg height', 'BMI 18,5', 'BMI 25', 'BMI
↪  30', 'US person', 'US avg weight', 'US avg height', 'Location', 'southeast')
title({'weight and height relationship of','100 Dutch and 100 American fictitous
↪  students'})
sum((weightVS./((lengthVS./100).^2))>30)
    ans =
            8
```

- there are 8 obese american students




```
% calculate dutch students' BMIs
NL_BMI= weightNL./(lengthNL./100).^2
% calculate american students' BMIs
US_BMI= weightVS./(lengthVS./100).^2
% produce a histogram of dutch students' BMIs
histogram(NL_BMI)
% make sure we don't lose the previous histogram
hold on
% produce a histogram of american students' BMIs
histogram(US_BMI)
% add labels, legend, title
xlabel('BMI')
ylabel('n')
legend('NL','US')
title('BMI amongst 100 Dutch and 100 American fictitous students')
% save image
saveas(gcf, 'bmi.png')
```



1.10 age & initials

```
% initialize age array
age = [14;32;23;18;20;45;16;50]
% older than 26
selectionA = find(age<26)
% younger than 22 or older than 30
selectionB = find(age<22 | age>30)
% isn't 23
selectionC = find(~age==23)
% is 50
selectionD = find(age==50)
% initialize initials array
initials = ['RT';'MK';'LH';'MW';'ST';'TH';'VB';'GF']
% first initial is 'L'
selectionE = find(initials(:,1) == 'L')
% is older than 20 and first initial is 'M'
selectionF = find(age>20 & initials(:,1) == 'M')
```

- note: if we want to grab the actual values for age we can just use `age(selectionA)`, but if we want to grab the actual values for initials we have to add `,:` to make sure we grab the entire string and not just the first character - `initials(selectionE)` returns only `L` but `initials(selectionE,:)` returns `LH`

1.11 faces

- correct data

```
load perception
% fix incorrect data by correcting the values in the scores matrix that
↪  correspond to the face stimuli
Scores(find(Stimulus==3),2) = 30
Scores(find(Stimulus==5),2) = 62
Scores(find(Stimulus==17),2) = 35
```

- compute mean, maximum and minimum scores per participant

```
mean_scores = mean(Scores, 1)
maximum_scores = max(Scores,[],1)
minimum_scores = min(Scores, [], 1)
```

  - this gives us the following values:

	participant 1	participant 2	participant 3
mean	46.8000	41.2500	39.0500
max	69	75	88
min	22	21	11

- preference of participant 1 towards male or female faces:

```
% extract participant 1 for easier use of data
p1 = Scores(:,1)
% grab the indices of male faces
male = ismember(Gender, 1)
% grab the indices of corresponding stimuli
male_stim = Stimulus(male)
% grab participant 1's scores of those stimuli
p1_male = p1(male_stim)
% compute the mean score for male faces for participant 1
p1_maleAVG = mean(p1_male)
% do the same for female faces but faster:
p1_femaleAVG = mean(p1(Stimulus ( ismember( Gender, 2))))
% compare avg scores
p1_maleAVG > p1_femaleAVG
    ans = logical 0 % therefore 'p1 prefers female faces'
```

  - participant 1 scored male faces lower on average than female faces (43.4 vs 50.2 respectively), therefore they may have a preference for female faces

- preference of participant 3 towards symmetric or asymmetric faces:

```
% extract participant 3 for easier use of data
p3 = Scores(:,3)
% first get the unique values used to score symmetry using
unique(Sym) % 1 2 3 4 5
% i've decided to separate symmetry classification in 2 groups:
low_sym = [1 : 3]
high_sym = [3 : 5]
% get only the faces that have low symmetry (between 1 and 3)
low_symmetry = ismember(Sym, low_sym)
```

```
% get the stimuli indices
low_sym_stims = Stimulus(low_symmetry)
% extract the actual trials that used those stimuli
p3_low_sym = p3(low_sym_stims)
% compute the mean of the low symmetry face scores
p3_asymAVG = mean(p3_low_sym)
% now do the same for high symmetry faces (between 3 and 5)
p3_symAVG = mean(p3( Stimulus ( ismember ( Sym, high_sym))))
% compare avg scores
p3_symAVG > p3_asymAVG
    ans = logical 1 % therefore p3 prefers symmetric faces
```

  - participant 3 scored symmetrical faces higher on average than asymmetric faces(39 vs 38.8333 respectively), therefore they may have a preference for symmetrical faces (the difference is pretty small though)

- calculate mean difference between participant 1 and 2

```
p1p2diff = mean(abs(Scores(:,1) - Scores(:,2)))
```

  - the mean difference of participant 1 and participant 2's scores is 23.2500

- extract old faces

```
old_faces = [4,10,12,18]
% get scores of old faces
old_scores = Scores ( find ( ismember( Stimulus, old_faces)),:)
% calculate mean of each old face ac cross participants
old_scores_faceAVG = mean(old_scores,2)
% calculate mean of each participant's score of old faces
old_scores_faceAVG = mean(old_scores,1)
% check if the average for all faces is higher than for old faces
Avg > old_scores_faceAVG
    ans = 0 0 1
```

  - the average score of older faces is higher than the general average score for participant 1 and 2, so they may have a preference for older faces, the opposite is true for participant 3

- sort the scores based on face number

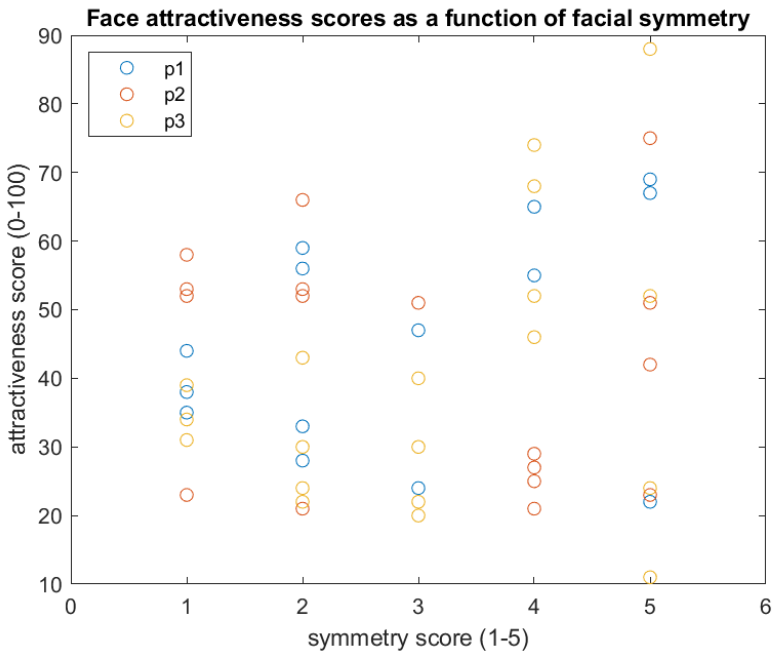
```
% we need the Stimulus vector in the matrix so we can sort on it
sortedScores = sortrows([Stimulus Scores])
% then we can get rid of the unnecessary first column (ex Stimulus vector)
sortedScores(:,1) = []
```

  - the sorted matrix looks like this:

22	51	11
33	52	24
30	51	30
58	52	31
56	66	22
24	51	20
35	58	31
28	53	43
38	53	39
67	75	52
47	22	22
44	23	34
65	25	52
65	27	74
47	30	40
42	23	24
52	21	46
59	21	30
55	29	68
69	42	88

- plot scores as a function of symmetry

```
% plot the symmetry of each stimulus against the scores
plot(Sym(Stimulus), Scores, 'o')
% add labels n stuff
ylabel('attractiveness score (0-100)')
xlabel('symmetry score (1-5)')
legend('p1','p2','p3', 'Location', 'northwest')
title('Face attractiveness scores as a function of facial symmetry')
% change the x-axis for a better overview
xlim([0 6])
```



  - It doesn't make sense to connect them with a line, as each score is independent of the next. If we computed the average for each symmetry score per participant it might make sense to connect the averages to visualize each participant's regression line.