MATLAB Practical session IV: Flow control

Exercise 1: BMI

This exercise will make a classification of patients based on the BMI index. The BMI classes are underweight (<18.5), normal range (18.5-25), overweight (25-30) and obese (>30).

Step 1: Scatter plot of weight vs. height

Open the excel-file *'Height_weight_BMI_stats.xlsx'*. This file contains data on the height, weight, gender and BMI of 100 subjects. Make a scatter plot of the weight vs. the height of the person. Make a distinction between male and female, which is defined by the gender column (1 for male, 2 for female). How many males and females where included in the study?

- Import the data from the excel file named *'Weight_weight_BMI_stats.xlsx'*. Remember that this file has been made in exercise session II on data import and export.
- Extract the data from the Gender, Height (cm), Weight (kg) and BMI column using logical indexing and the string compare command ('*strcmp*').
- Find the indexes corresponding to males and those corresponding to the females, using the 'find' command (1 for male, 2 for female).
- Answer the question: How many males and females were included in the study?
- Make a scatter plot of the weight against the height. Use a different marker symbol and marker color ('filled') for the males and females. Add also a meaningful title, labels and a legend to the plot.
- Save the plot under the name *'weight_vs_height.jpg'*.

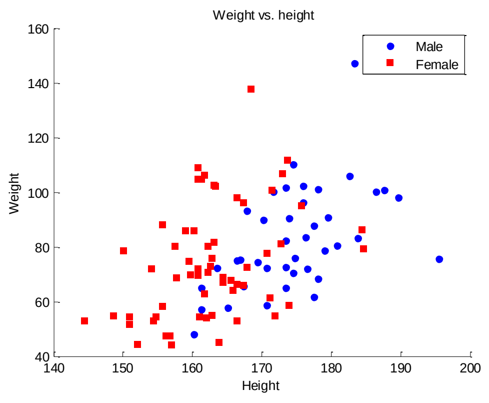

Figure 1 Scatter plot weight versus height for male and female subjects

Step 2: BMI classes

Classify the subjects according the BMI. The classes are: underweight (<18.5), Normal range (18.5-25), overweight (25-30) and obese (>30). Save the number of classes in the excel file as a separated column.

- Construction of zero-vector classes to define the class per subject.
- Use a FOR loop over all subjects. Define the classes using IF/ELSEIF/END statement. There are four classes in total numbered 1, 2, 3, and 4 for underweight, normal range, overweight and obese respectively.
- Add the constructed vector to the rest of the imported data, and add a column header 'Classes'.
- Write everything in a new excel file, called *'Height_weight_BMI_classes.xlsx'*.

Step 3: Stacked bar plot BMI classes

Construct a stacked bar plot, two separated one for the females and males. The height of the bar plot indicates the percentage of male and females in the different BMI classes. Use therefor the classes as determined in previous step.

- Construct a zeros-matrix for the bar plot with two rows (two groups: male and female) and 4 columns (4 BMI classes - underweight, normal range, overweight and obese)
- Fill the bar plot matrix with the number of subjects in a certain class and gender. Use nested FOR loop to run over all gender and all classes. For each combination (gender and classes), determine the number of corresponding subjects (using the sum and relational operators).
- Normalize the bar plot matrix using the number of females and males, such that the total of all classes in 100%.
- Plot the matrix containing the normalized bar plot matrix using the 'bar' plot function, using a stacked layout. If everything is alright, the largest Y-value will be 100%. Add a legend to the plot, which states the different classes.
- Save the figure under the name *'BMI_classification.png'*.

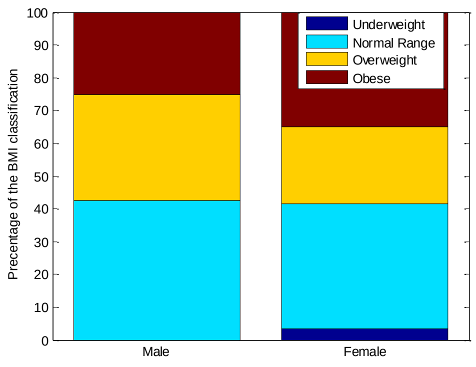

Figure 2 Stacked bar plot containing the percentage of male and female in a certain BMI class.

Exercise 2: Kinematics

Step 1: Creating subplots with for statement

Open the file 'KS_motion_trial62.mot' with the kinematics values for the simulated motion and plot the ankle, hip flexion, hip adduction and the knee angles in four different subplots on 1 figure. Instead of doing this as performed in exercise session II on plotting, try to do this as efficient as possible, by using the FOR in combination with the SWITCH command. Only one time the subplot command may be written.

- Import data from *'KS_motion_trial62.mot'*
- In total we want to have four different subplots e.g. hip flexion, hip adduction, ankle flexion and knee flexion. Use the FOR statement to create the subplots. The SWITCH command is used to determine the name of the column header to plot. For this dataset we are interested in the following column names: 'hip_flexion_r', 'hip_adduction_r', 'ankle_angle_r', 'knee_angle_r'.
    - Find the column number by searching the matching column header in the imported column header data
    - Create the subplot and plot the data

Step 2: Add titles and labels to the plots

Copy the previous code and adapt it, such that a title is added to each subplot, which state the name of the variable (Hip flexion, Hip adduction, Knee flexion and Ankle flexion). Use an IF statement to add the x-label ('Time (s)') to the lower subplots and the y-label ('Angle (°)') to the most outer left subplots. The output is the same figure as in the Plotting exercise session, but now implemented more efficient.

Step 3: Add another dataset to the figure

Repeat this workflow to open a second data file and plot the same kinematic values on the corresponding subplot. Use a different color for the different files. Implement this also as efficient as possible, using a nested for loop. Only the number in the filename is changed for the second file (*'KS_motion_trial69.mot'*, '69 instead of 62). Use this information to efficiently implement the FOR loop. Use the SWITCH command to change the filename and the color of the line plot.

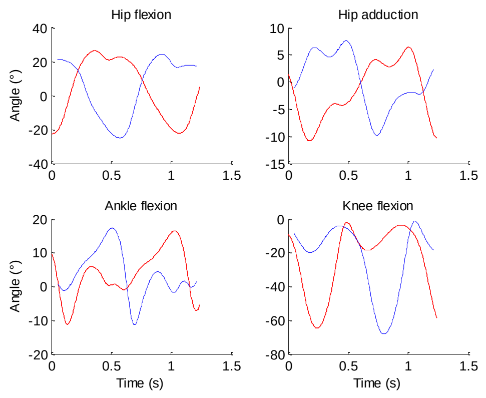

Figure 3 Kinematica data from '*KS_motion_trial69.mot*' and '*KS_motion_trial62.mot*'

Exercise 3: Pressure

Create a plantar pressure movie from a dynamical plantar pressure measurement.

Sensor information

Total_width = 64;

Total_height = 64;

length_active_area = 0.488;

width_active_area = 0.325;

delta_y = length_active_area/Total_height; 64 scanning lines

delta_x = width_active_area/Total_width; 64 scanning lines

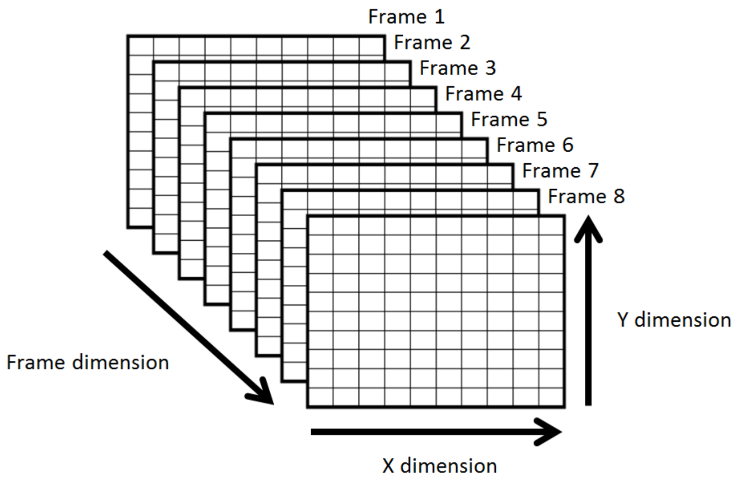

Figure 4 Sensor information

Step 1: Import dynamical pressure measurement

Load the data from a dynamical pressure measurement, stored in *'Jos_4 - RS_L - Dynamic Roll off.xls'*. In this file a matrix is given for each time frame of the measurements. This matrix contains the pressure values for each sensor in the matrix. You can compare it with the data from an exercise from Practical Session 2. Read each frame and save this information in a large 3D-matrix: the 2 first dimensions for the dimensions of the matrix, and the third one is the frame dimension. Do this as efficient as possible using a WHILE loop.

Step 2: Plot each frame

Use the 3D-matrix from the previous step to plot each frame. Use the knowledge from Practical session 2 in which you plotted the pressure data. Plot the frame on the same figure (without *hold on*), such that feels like a movie, displaying the plantar pressure. Tip: use a FOR loop.

Step 3: Run for 2 different files

Copy the code from step 1 and step 2 and let it run for 2 different files (*'Jos_4 - RS_L - Dynamic Roll off.xls', 'Jos_4 - BMe_R - Dynamic Roll off.xls'*). Do this also as efficient as possible using a FOR loop.

Step 4: Create a movie

EXTRA: Add some code to capture a movie from the plots made. Therefore you need to you the *VideoWriter* object. Try to find more information in the help file.

Exercise 4: Ages

Step 1: Age of subjects

Determine the age of a person based on the birthdate and a reference date. The birthdate of a person is given by three integers *bd, bm, by* representing the day, month and year of birth. Make a script using the IF statement which determines the age in years of a person at a certain reference date. The reference date is represented by three integers *rd, rm, ry* (day, month, year). Remember to take into account that someone may have had a birthday before the reference date. Test the script for multiple dates.

- Create six variables to define the birthdate and reference date.
- Computation of the age. Remember to take into account that someone may have had a birthday before the reference date. To solve this problem use an IF statement. If the reference date has not been passed during the reference year decrease the age with 1. Make sure no answer or zero is created if the reference day is prior to the birthday.

Step 2: Birthday database

- Import the data from the Excel file named *'Birthdays.xls'*. This file contains birthday data of 1000 subjects. Extract the *'day'*, *'month'* and *'year'* of birth using string compare command ('*strcmp*').
- Copy and adapt the previous code to calculate the age of 1000 subjects. To do so add a FOR statement to execute the previous code for all the subjects.

Step 3: Population groups

To visualize the age distribution for our birthday data we have to create age classes. For this exercise we want to define four age groups (0-25, 26-50, 51-75, 76+). Visualize the distribution over the groups using a bar plot. Save the group number and the age in an excel file as a separated column. For all the age groups we want to know the number of subjects included in the age group. Use a FOR loop to execute the statements for all the subjects. To create the groups use the SWITCH statement. For the definition of the case expressions use a cell array.

- Construct a zeros-vector to store the group number for every subject.
- Construct a zeros-vector to include the number of subjects included in the group.
- Use a FOR loop over all subjects. Define the groups using SWITCH statement. There are four groups in total numbered 1, 2, 3, and 4 for 0-25, 26-50, 51-75, 76+ respectively. (It is also possible to create the age groups using an IF statement. As and exercise you could program this as well).
- Construct a bar plot to visualize the distribution of the ages over the four groups. Add a title and labels to the axes of the bar plot.
- Add the constructed vector containing the age group and the computed ages to the rest of the imported data, and add the column headers 'Age' and 'AgeGroup'.
- Write everything in a new excel file, called *'Birthdays_group.xls'*.

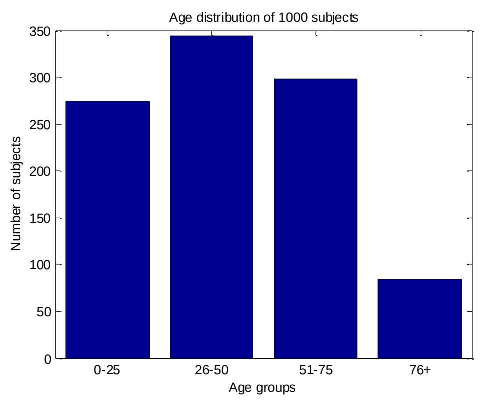

Figure 5 Bar graph showing the age distribution of 1000 subjects

Exercise 5: Metric unit converter

- Write a program to convert metric units to centimeters. Use the SWITCH command to define different cases e.g. mm, cm, dm, m, km. Test your code for different units and values.

  ```
  x = 3.0;          % numeric variable
  units = 'mm';     % string variable
  ```
- Copy the previous code and adapt it to use this program for converting arrays of numbers and units. Use a FOR loop.