Practical Course MATLAB/Simulink

Data Handling and Visualization

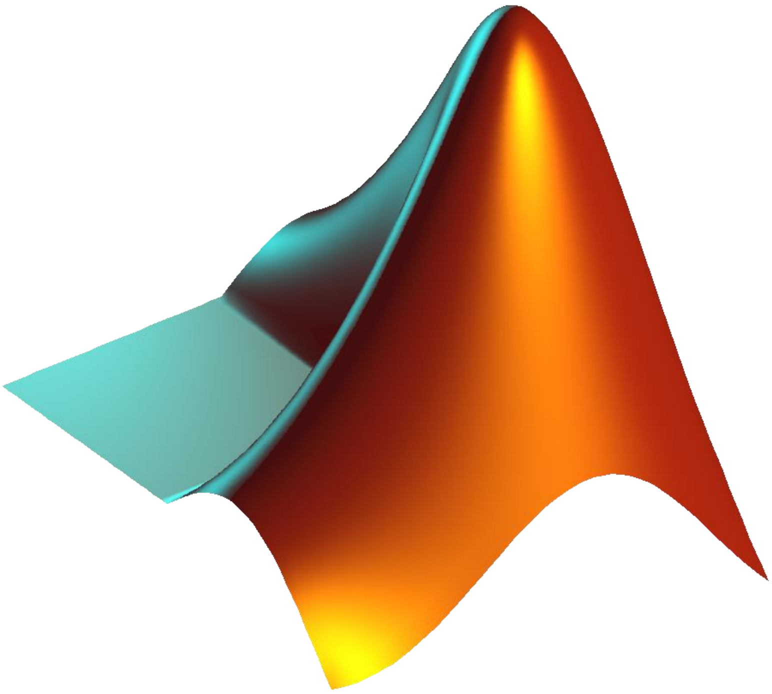


Table of Contents

List of Tables

Table of Figures

0 General Information and Advice

The following exercises cover Data Handling and Visualization. The Tasks will cover importing and exporting data, efficient memory management in MATLAB and creating and customizing graphics programmatically and interactively.

It is recommended that you write the MATLAB code you produce during this session into an M-File (MATLAB script). This makes it easier for your supervisor to help you in case of problems. You can also save and keep the file for your records.

At the beginning of each exercise, delete all existing variables.

1 Importing and Exporting Data

In this exercise you will import and export data to and from different formats. The data covers the development of the German population from 1960 to 2014 and can be accessed at http://data.worldbank.org/country/germany. Please go to your Moodle account to download the data.

Exercise

(1) Copy the file PopulationData.dat to your current working directory, open the interactive import tool and load the data into your workspace as a table data type. Be sure to set the right delimiter and data range.

(2) Investigate your data using the `summary` command.

(3) Add new columns to the population data table containing the overall number of female and male population for each year.

(4) Use the `csvwrite` command to create a comma separated data file called GermanPopulation.csv containing all data from the table. Create a variable of a numeric data type first that contains all numeric data from the table.

(5) Save only the population data table to a .mat-file and name it GermanPopulation.mat. Check the file using the Details window to verify that only the table is contained by the file.

(6) Perform a low level import of the data from PopulationData.dat.
   - Use the `textscan` command to read the header
   - Then use `fscanf` to read all data into a numeric array.

   Hit F1 or use the doc command to find out about the syntax of `textscan` and `fscanf`.

(7) Use the fprintf command to print all numeric data to a text file. Use a format specification string to format the output
   - Each column is 10 characters wide
   - Year and overall population are formatted as integers
   - all other data is formatted to display 3 digits right of the decimal point

2 Memory Management

Memory Management is an important part of MATLAB when huge data has to be processed or fast computation time has to be achieved. In this exercise some of the features of memory management are demonstrated.

Exercise

(1) The first task will demonstrate the difference between local, global and persistent variables. Create three functions that increment a variable:

- The first function inputs the previous counter value, increments it by one and outputs it as the new counter value. The variables used in this function exist locally in the function's workspace and are deleted once the function is terminated.
- The second function has no input arguments; however, it declares a persistent variable. This variable has to be initialized when it is called for the first time, i.e. when it is empty. The function outputs the new counter value.
- The third function has neither input nor output arguments. A global variable is declared within the function, as well as the script calling the function. This variable is initialized as 0 in the script and incremented every time the function is called.

Use a for loop to increment a counter 10 time using each of the functions. Be careful to specify the correct input and output arguments. Display the results in the command window.

Run the script containing the test loop several times. Observe that the persistent variable continues counting because it is not reset. Use the clear “FunctionName” command to reset the persistent variable and try again.

(2) Memory allocation can be very important to improve speed of MATLAB programs. Using the tic and toc functions, time assigning the quadratic numbers from 1 thru 1000 to a numeric array with and without pre-allocating the memory. Recall that MATLAB has to find a new memory block each time a variable's size is increased.

(3) Create a large variable called A containing a 20000 by 20000 magic array. Open the task manager and observe the memory used on your machine. Assign B to A and observe the change in memory usage. Change the first entry of B to zero and check your memory again.

3 Graphics

In this Exercise you will learn to visualize data. Firstly, 2D plot will be created from the data imported and processed in Exercise 1. Subsequently 3D plots will be created from 3 dimensional data.

(1) Load the table containing the German population data you saved in Exercise 1. Observe how variables are loaded to the workspace from mat-files. Extract the table and store it in a new variable.
(2) Plot the total population number over the years contained in the data.
(3) Use the plot tools to interactively add a title to the plot and labels to each axis. Add a grid to the axis.
(4) Add a legend to the plot. Use the `gca` command to retrieve the handle of the current axis.
(5) Add two more graphs for the male and female population and set the display name of each graph. Update the legend by calling the legend command again with all lines contained within the axis object (Children of the axis object).
(6) Create two new figures to plot the female and male population using an area graph and a pie chart. Annotate both graphs.
(7) Use subplots to create a figure containing 4 different visualizations of Cleve Moler's L-shaped membrane. You can create the data using the `membrane` command.

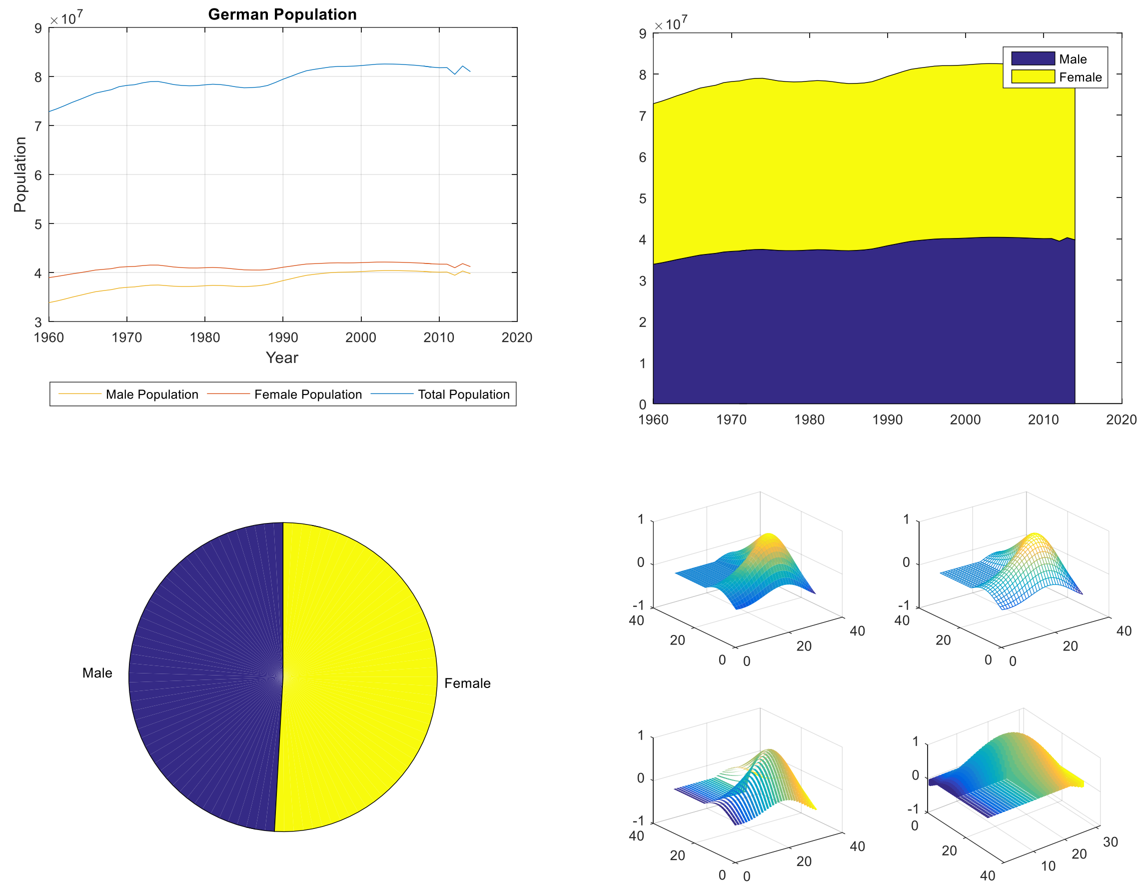


Figure 1: Plots

4 Race Track for Arduino Robot

In the MATLAB/Simulink Practical Course, an Arduino Robot will be programmed which can follow a given track. This track will be generated, processed and displayed within this exercise using MATLAB's input and plot functions.

(1) Create a figure to display the results from each of the three steps of the data processing. Pre-allocate memory for three subplots using the gobjects command. Store a first subplot in the first element of the new variable.

(2) Load the image file melbourne_circuit.jpg into the MATLAB workspace using the appropriate command and store it in the variable melbourne. Display the image in the first subplot. Observe that the race track has already been highlighted. Inspect the data structure by investigating the variable melbourne in the workspace.

(3) Extract the race track from the image data.
- Using logical indexing, replace every pixel that is not blue by a white pixel (RGB: [255;255;255]). Apply the criteria from Table 1. Display the resulting image in the second subplot.
- Pre-allocate memory for an array of data type logical, sized according to the number of pixels in the processed image file. Initialize the array with false.
- Fill the matrix with logical values, depending on whether the pixel belongs to the race track or not. Display the result in the third subplot using the spy command.
  Hint: You can do this by checking the red and green channels for zeros.

Table 1: Criteria for a blue pixel

Color	Minimal value	Maximal value
R (red)	0	0
G (green)	0	0
B (blue)	250	255

(4) Use the find command to retrieve the coordinates of all points on the track. Write the x,y data pairs to a comma separated value file (csv).

http://www.mister-foley.com/images/races/melbourne_circuit.jpg

5 Large Data Analysis

Within this exercise we want to visualize the spread of the COVID19 virus across Germany in MATLAB. Therefore we download the by the RKI institute published data for the spread of the virus over Germany [1]. Next, we calculate the number of cases for each state within Germany and plot the corresponding values as a bubble chart in MATLAB.

For readability the term state is being used from now on for federal state.

1. Download the raw data from Moodle and have a quick look into the .CSV file to get familiar with the information content provided by the data.
2. For the import and analysis of the provided data we want to use datastore objects and tall arrays due to the large size of the provided ***RKI_COVID19.csv*** file.
    a. Create a datastore object for the provided CSV file. After creating the datastore object, change the delimiter to a comma by changing the delimiter property of the datastore object. An example to change the property of the datastore object is shown in the lecture slides.
    b. For the further analysis within the scope of this exercise we do not need every column of the provided file. We need following variables (e.g. columns) with their corresponding column number in the CSV file, their name after import and their import formats:

Column Number in CSV file	**Name**	**Format**
3	State	Categorial, e.g. %C
5	Age Group	Categorial
6	Sex	Categorial
7	Cases	Float
8	Deaths	Float

    Set the respective property of the datastore object to the defined values from above. (Hint: *SelectedVariableNames* and *SelectedFormats* of the datastore object). Finally, display a preview of the created datastore object by means of the command *preview(your_datastore_object).*

    c. Convert the datastore object to a tall array.
    d. Since the data size is large, we want to create a subset of the whole dataset for debugging purposes. Therefore, create a smaller tall array of the existing tall array (Hint: *head).* From now on, continue to work with the smaller array.
    e. Sum the values for the variables *Cases* and *Deaths* and group them for each state in Germany. The commands *gather* and *groupsummary* will be your friend for this task.
    f. Create two pie charts based on the results. One for the cases and one for the deaths in each state in Germany.
    g. In the last task we want to obtain the bubble figure as shown below.

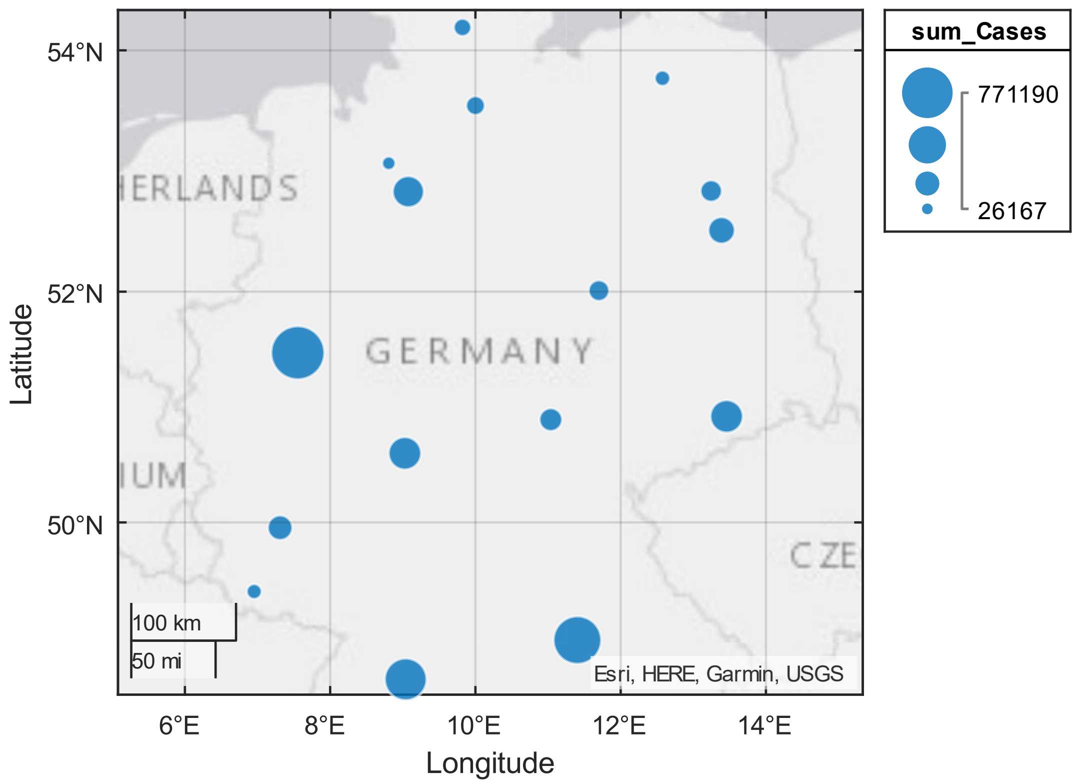
i. Import the Longitudinal and Lateral information for each state with from the provided Excel sheet and assign the imported values to a global variable.

ii. Create a MATLAB table programmatically with the following shape where each row corresponds to a state.

Latitude	**Longitude**	**sum_cases**	**sum_Deaths**

To look up the longitudinal and lateral information for each state create a custom local function with the state name as input and the longitudinal and lateral values as outputs. Within the function you can access the previous defined global variable. To find the matching index, use a *cellfun* in combination with a *strcmp* command.

iii. To visualize the bubble figure use the MATLAB command *geobubble.* Specify the dimension of the bubbles accordingly to the aggregated number of cases.



[1]: https://npgeo-corona-npgeo-de.hub.arcgis.com/datasets/dd4580c810204019a7b8eb3e0b329dd6_0