

## RECAP (3 points)

Available: 2018-10-13 - 2018-10-18

Duration: 60 min

Topics:

- Applications of CRV
- Architectures of CRV systems
- MATLAB basics (as taught in the Onramp course)

Hint: It's possible to use MATLAB during the test. Therefore, make sure you have MATLAB available during the test. If you see a command (e.g. 'diag') you are not familiar with, you can get some information about it by typing 'help diag' or 'doc diag' into the MATLAB Command Window.

Link: [1]

## PROGRAMMING (7 points)

The goal of this weeks programming exercises is to learn some advanced MATLAB skills like writing a script, implementing a function and the benefits of vectorization. These skills are necessary when working on computer vision tasks in the future exercises.

### 1. **Writing a script** (2 points)

Scripts offer the repetition of command execution instead of typing single commands into the command window every time. Therefore, scripts are mostly useful and extensively used.

- a) Read the MATLAB help about creating a script [2].
- b) Create a script named


`CRV_%exercise%.m`


You should substitute %exercise% with the number of the exercise (here it is 1). The first three lines should look like:

```

1 %% CRV_01_Write a Script
  % name: John Doe
3 % student number: 11235813

```

Of course again  fill in your name and student number!

- c) In your script add a section named 'clean up'. In this section you should clear the workspace, close all figures and clean up the command window.
- d) In your script add a section named 'plotting'. Here you should plot sine and cosine for 300 linearly spaced points within the range  $[-2\pi, 2\pi]$ . The sine should be plotted in red, while cosine is blue. Also add axis labels, a title and a legend. Additionally make sure that just  $x \in [-2\pi, 2\pi]$  is shown. 
- e) Run your script.

## 2. Writing a function (3 points)

Despite using the functions, that MATLAB already offers, we will create our own functions as well. This exercises trains you to do this on your own.

- a) Have a look at the MATLAB help about creating a function. [3], [4].
- b) Implement a function with the header

```

1 function [ f ] = MyFacRec( n )
  %MYFACREC Recursive factorial calculation
3 % This function implements the calculation of the factorial
  for natural numbers.

```

As the header says you should implement a recursive calculation of the factorial. Of course without using MATLAB's built-in function **factorial**!

For any natural number  $n$  the factorial function  $\text{fac} : \mathbb{N} \rightarrow \mathbb{N}$  is defined by

$$\begin{aligned}
 n \mapsto \text{fac}(n) := n! &:= \prod_{i=1}^n i \\
 &= 1 \cdot 2 \cdot \dots \cdot n
 \end{aligned}$$



Remark: For all natural numbers  $n > 1$  it holds:  $n! = n \cdot (n-1)!$ .

- c) Implement a function with the header

```

1 function [ f ] = MyFacIter( n )
  %MYFACITER Iterative factorial calculation
3 % This function implements the calculation of the factorial
  of integers.

```

As the header says you should implement an iterative calculation of the factorial. Of course without using MATLAB's built-in function `factorial`!



- d) Create a script (analog to exercise 1. b) & c)), in which you execute both your factorial functions for  $n \in \{1, 2, 3, 42\}$ .
- e) Extend your script by using the MATLAB functions `tic` and `toc` to evaluate the execution time of your functions.
- f) Compare the results and the execution times of your functions with the ones of the MATLAB built-in function `factorial`.



### 3. Vectorization (2 points)

MATLAB is the MATrix LABoratory. Many functions are optimized to deal with matrices and vectors as input. Vectorization is the art of using functions, which work on vectors and matrices, instead of using loops to work on single elements.

- a) Create a script (analog to exercise 1. b) & c)).
- b) Add a section 'matrix generation'. Generate a square matrix with normally distributed random entries. The size of the matrix should be defined via a variable, which has been initialized with 1000.
- c) Add a section 'squaring with for-loops v1'. Create a copy of the random matrix. Use this copy to calculate the matrix, where every entry has been squared. This should be done for each entry separately here by using two nested for-loops.
- d) Copy your previously created section into 'squaring with for-loops v2' and exchange the loop order.
- e) Create a section 'squaring using MATLAB functions', where you again create a copy of the random matrix. Here the squaring of the entries should be done via the element-wise power function `.^`.
- f) Use `tic` and `toc` to evaluate the runtime of the three variants.



## References

- [1] CRV. *Recap 00*. URL: <https://moodle.uni-due.de/mod/quiz/view.php?id=674300>.
- [2] MathWorks Matlab Help. *Create Scripts*. URL: [https://de.mathworks.com/help/matlab/matlab\\_prog/create-scripts.html](https://de.mathworks.com/help/matlab/matlab_prog/create-scripts.html).
- [3] MathWorks Matlab Help. *Create Functions in Files*. URL: [https://de.mathworks.com/help/matlab/matlab\\_prog/create-functions-in-files.html](https://de.mathworks.com/help/matlab/matlab_prog/create-functions-in-files.html).
- [4] MathWorks Matlab Help. *Functions*. URL: [https://de.mathworks.com/help/matlab/ref/function.html?s\\_tid=doc\\_ta](https://de.mathworks.com/help/matlab/ref/function.html?s_tid=doc_ta).