# Submission Guidelines for Homework 1

**VU Numerical Algorithms, SoSe 2019**

*due date: 23.4.2019, 18:00*

## Prerequisites

1. Basics:
   - Please use Octave[1] *version 4.4* or higher and indicate the Octave version in your report. Your submission will be evaluated.
   - Do not import additional packages and do not use global variables.
   - Pay attention to the interface definitions, i.e., use the specified terms. In/output parameters must be in the specified order.
   - Your routines should always check the number and types of input arguments.
   - Do not plot results in predefined routines! Plot results in scripts or self defined routines only.
   - Do not exploit any special structure in the input data. Your routines must be generic and have to work for all $n > 1$.
   - Do not use any existing code which you did not write yourself!
   - You can define your own routines in order to write modular code but please stay consistent with the predefined interface.
2. Interface:
   - Mandatory for **all Parts**:
     a) Create a file *accuracy.m* of the following form:

```
[z] = accuracy(X, Y)
```

     - Input: $X$ and $Y$ are either both $n \times n$ matrices or both vectors of size $n$.

[1] Octave download page: https://www.gnu.org/software/octave/download.html

- Output: scalar $z$ with

$$z := \frac{||X - Y||_1}{||Y||_1}.$$

**Remark**: Use this routine to verify the correctness of your $LU$ factorization in **Part I** and to compute the *relative residual* and *relative forward error* in **Part II** resp. **Part III**.

- Mandatory for **Part I**:
  a) Create a file *plu.m* for the $LU$ factorization with partial pivoting:

```
[A, P] = plu(A, n)
```

  - Input: $n \times n$ matrix $A, n$
  - Output: $n \times n$ matrices $L$ and $U$ stored in the array $A$ ($A = P^T LU$) and the permutation matrix $P$.

  b) Write a script *part1.m* to test your routines and plot your results.

- Mandatory for **Part II**:
  a) Create two files *solveL.m* / *solveU.m* for solving a lower / upper triangular system:

```
[x] = solveL(B, b, n)
[x] = solveU(B, b, n)
```

  - Input: $n \times n$ matrix $B$, the right hand side vector $b$ of size $n$, $n$
  - Output: the solution vector $x$.

  **Remark**: The input matrix $B$ has special structure $B = L + U - I$, where $L$ (with fixed ones in the diagonal) and $U$ are lower resp. upper triangular matrices and $I$ is the Identity.

  b) Write a script *part2.m* to test your routines.

- Mandatory for **Part III**:
  a) Create a file *linSolve.m* for solving a linear system:

```
[x] = linSolve(A, b, n)
```

  - Input: $n \times n$ nonsingular matrix $A$, the right hand side vector $b$ of size $n$, $n$.
  - Output: the solution vector $x$.

**Remark**: This routine must incorporate **plu.m**, **solveL.m** and **solveU.m** from previous Parts!

b) Write a script *part3.m* to test your routines and plot your results.

3. Submission:
   - Upload a single zip archive with all your source code files and your report (as a single PDF file named *report.pdf* with all plots and discussions of results) on the course page in Moodle.
   - Name your archive `a<matriculation number>_<last name>.zip` (e.g. *a01234567_mustermann.zip*)
   - Directories in the archive are not allowed.
   - A complete submission should include the following files:
     a) Routines: *accuracy.m, linSolve.m, plu.m, solveL.m, solveU.m*, self defined routines (optional)
     b) Scripts: *part1.m, part2.m, part3.m*
     c) Documentation: *report.pdf*

## Octave cheat sheet

Below is a list of recommendations that might help with your implementation:

- You should at least be aware of the following routines provided by Octave:

  *validateattributes*, *nargin*, *norm*, *tril*, *triu*, *transpose*, *semilogy*, *abs*, *min*, *max*, *eye*, *zeros*, *ones*, *rand*, *sum*, *length*, ...

  You can lookup all Octave routines here:
  https://octave.sourceforge.io/list_functions.php

- We recommend vectorization to simplify and optimize your code (i.e., eliminate for loops whenever possible):

  https://octave.org/doc/v4.0.1/Basic-Vectorization.html

- For efficiency, the LU factorization algorithm usually operates on a permutation vector (instead of a whole matrix) during the factorization process and creates the permutation matrix at the very end. You might consider using a permutation vector as well (although, this is not mandatory). Creating permutation matrices is explained here:

  https://octave.org/doc/v4.4.1/Creating-Permutation-Matrices.html