Examination and Coding Guide

When we examine your project in the course TSRT04 Introduction in Matlab, we check for functionality and style. Passing the functional requirements is not enough for the project to pass. It is also required that the code is easy to understand and maintain.

Comments
We expect your code to be commented in a reasonable way. It should be possible to understand what a particular segment of code does by simply reading the comments – one should never have to ask the programmer. Since the teachers that handle the examination might come from abroad, you need to write code and comments in English (as it is done at most companies and universities).

Modularization
Build your code using functions and write a short example script that calls these functions to solve the problems stated in the project. This script should be seen as *one* way of calling these functions, and the teaching assistants might ask you to make changes in the input parameters. The functions should print out written errors or warnings if the input parameters are unreasonable or clearly wrong (e.g,. negative instead of positive values, or wrong matrix dimensions).

Function and variable names
Use descriptive names such as `months` and `yearlysalary`, instead of generic letters such as `x` and `y`. By reading the variable or function name, it should be possible to understand what purpose it has. If you use the MATLAB editor to write your code, you can change the name of a variable and then press Shift+Enter to change the name everywhere the variable is used in the same m-file. This can be used to improve the descriptiveness of already existing variable names.

Don't repeat yourself
Don't copy code segments repeatedly and change small parts. Try to put the code in a function that can be called with different inputs, or create a loop.

Understand when to use *for* and when to use *while*
for-loops are used when you know *a priori* the number of iterations that will be performed. while-loops are used if the number of iterations depends on computations performed inside the loop.

Efficiency
We do not expect you to write highly computationally optimized code — in fact, it is often better to make the code easy to read than to save a few milliseconds in run time. It is however problematic if your code is so slow that it takes us several minutes to run. To find out why your code is slow, use the profiler (e.g., press the button "Run and Time" in the MATLAB editor). A typical reason is that calculations are done inside loops (even if they could have been placed outside) or that the dimensions of a matrix is changed inside a loop (it is better to define the matrix with `zeros` before the loop starts).