

## EXERCISE 2 - SOLUTION

### **Problem 2.1.** (Image Denoising via Convex Optimization)

We will investigate some of the properties of the Chambolle-Pock algorithm for structured convex optimization problem in this exercise.

- (i) Download the classic  $256 \times 256$ -pixel black-and-white testpiece [image of a cameraman](#) posted on our website for this exercise.
- (ii) Load the image and add some noise to it.
- (iii) Implement a rudimentary version of the Chambolle-Pock algorithm for a fixed number of iterations.
- (iv) Implement the remaining components for using the Chambolle-Pock algorithm for denoising the noisy image, run the algorithm, compare the results and investigate the influence of the algorithmic parameters on the algorithm.

### **Solution.**

Most of the work in this exercise is simply done by programming.

In the results we observe a few things:

- (i) The denoised image manages to retain sharp edges along larger neighboring areas but loses some details in smaller edge parts of the image, like the tower in the back.
- (ii) The first couple of iterations seem to be the most important ones in decreasing the norm of the difference (pixelwise) of the original and the denoised picture.

- (iii) At some point during the algorithm, the norm of the difference of the original and the denoised image starts increasing again. This is classical behavior for optimization based on penalized tracking functionals, as we do not only have the tracking term, but also the regularization influencing the solution. After all, finding the original image from the noisy one is an ill-posed problem.
- (iv) The parameters  $\tau$  and  $\sigma$  play a large role in convergence of the algorithm. The smaller  $\tau\sigma$ , the more regularized the subproblems will be, retaining less information of the original problem, meaning slower progress for the overall optimization algorithm. Larger  $\tau\sigma$  generally mean faster progress, but choosing  $\tau\sigma$  larger than  $1/8$  (e. g.  $1/2$ ) will lead to nonconvergence of the algorithm.