Inverse problems course — Exercise 1 (January 20-22, 2015)

University of Helsinki
Department of Mathematics and Statistics
Samuli Siltanen and Andreas Hauptmann

Related book sections (Mueller & Siltanen 2012): 2.1.1 and 2.1.2.

Theoretical exercises:

T1. Define a function $g : \mathbb{R} \to \mathbb{R}$ by

$$g(x) = \begin{cases} 1 & \text{for } -0.1 \leq x \leq 0.1, \\ 0 & \text{otherwise.} \end{cases}$$

Compute the function $g * g$ analytically (by hand), where

$$(g * g)(x) = \int_{-\infty}^{\infty} g(x')g(x - x')\, dx'.$$

Outside which interval $[a, b] \subset \mathbb{R}$ is $(g * g)(x) = 0$?

T2. Let the discrete point spread function $p \in \mathbb{R}^3$ and the vector $f \in \mathbb{R}^{10}$ be defined by

$$\begin{aligned} \widetilde{p} &= [\widetilde{p}_{-1}, \widetilde{p}_0, \widetilde{p}_1]^T = [1, 1, 1]^T, \\ f &= [f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}]^T = [0, 0, 0, 0, 1, 1, 0, 0, 0, 0]^T. \end{aligned}$$

Compute the discrete convolution vector $(\widetilde{p} * f) \in \mathbb{R}^{10}$ by

$$(\widetilde{p} * f)_j = \sum_{\ell=-1}^{1} \widetilde{p}_\ell f_{j-\ell}, \qquad 1 \leq j \leq 10,$$

where $f_{j-\ell}$ is defined using periodic boundary conditions for the cases $j-\ell < 1$ and $j - \ell > n$.

T3. Take $\Delta x = \frac{1}{10}$ and compute the normalized point spread function

$$p = \Big(\Delta x \sum_{j=-1}^{1} \widetilde{p}_j\Big)^{-1} \widetilde{p}.$$

Compute the discrete convolution vector $(p * f) \in \mathbb{R}^{10}$ with vector $f \in \mathbb{R}^{10}$ as in exercise T2 except that $f_1 = 2$. Be careful with the periodic boundary condition!

Matlab exercises:

M1. Download the following files from the course webpage:
`target1.m`
`targets_plot.m`
`PSF.m`
`deconv1_cont_comp.m`
`deconv1_cont_plot.m`

(a) Create your own target function by modifying the file `target1.m`. Name your function file `target2.m`.

(b) Replace the silly Riemann sum integration (also known as *midpoint rule*) by the *trapezoidal rule* in the convolution file `deconv1_cont_comp.m`. You can use Matlab's built-in routine `trapz.m`. Can you use less integration points (smaller `Nt`) and still get an accurate convolution result?

(c) Choose different values of parameter a and run `deconv1_cont_comp.m` using your own function `target2.m`. Can you find a suitable value of a such that some information is lost (corners rounded or so), but the main features and forms of your function are still visible?