Exercises 5 & 6

Getting more familiar with concepts of optimization and solution algorithms.

Problem 1

Complement the given m-file `grad_method.m` to an optimization routine, which uses user-supplied gradient, quadratic minimization for line search and BFGS-based quasi Newton method for search direction determination. Test the implementation using the given `exafun.m` and optimization based determination of the mean-value (cf. lecture notes and slides) of randomly generated vectors (the reference method in MATLAB for testing the correctness of the solution is `mean`; notice the appropriate dimensions of the random sample matrix).

Let us define the following cost functionals:

$$\begin{aligned} \mathcal{J}_1(x,y) &= 100 * (y - x^2)^2 + (1-x)^2 \quad \text{(so-called } \textit{Rosenbrock}\text{'s function)}, \\ \mathcal{J}_2(x,y,z) &= 100 * (2 * z - y^2 - x^2)^2 + (1-y)^2 + (1-x)^2, \\ f(x) &= \frac{\sin(x^2)}{x}. \end{aligned}$$

You can probably figure out by yourself the exact minimizers of, at least, functions $\mathcal{J}_1$ and $\mathcal{J}_2$.

Problem 2

Write m-files which can be used to compute the values of the given functions and their gradients also in the case when coordinate matrices are given as inputs (similarly to built-in functions of MATLAB). Using the corresponding m-file, plot function $f(x)$ and its derivative.

Problem 3

Minimize the given functions using your own routine from Problem 1 and the built-in routines `fmin` and `fminunc` in MATLAB. Compare the solutions that your obtain using these three methods with each other to be sure that all routines recover (up to the termination tolerance) the same solution. Use also commands `tic` and `toc` to measure the amount of CPU time that was needed for solving the problems. Compare the solution from `fmin` to the plot of $f(x)$ and its derivative.

Problem 4

Illustrate function $\mathcal{J}_1$ using a surface plot and point out the location of the minimizer in this image (cf. kuva 3.3 in lecture notes and slides). Compute the numerical gradient for function $\mathcal{J}_1$ using the `gradient` command. Find the gradient vector of minimal length and print out the x and y coordinates of the corresponding point. What do you notice?

Problem 5

Add to the illustration in Problem 4 also the search path that the different routines follow while seeking the (local) minimum. Finally, modify this illustration by creating two `subplot`s, which contain the surface plot and the search path history together with the corresponding convergence history of function values during the minimization.