Optimization and Optimal Control (WS 22/23) Computer Lab 1

Prof. Dr.–Ing. habil. Thomas Meurer, Chair of Automatic Control

Exercise 1 (Line search). Use the provided `Matlab` files [1] which implement a simple line search method using steepest descent combined with a Wolfe condition to familiarize yourself with the general setup by executing the `oocLab1` script. Herein, the Rosenbrock problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^2} f(\boldsymbol{x}) = 100\big(x_2 - x_1^2\big)^2 + \big(x_1 - 1\big)^2 \tag{1.1}$$

is solved using two (anonymous) functions `rosenbrock` and `gradRosen`, a function `lineSearch`, and `wolfe`. Note that $\boldsymbol{x}^* = [1, 1]^T$ is a local minimizer. Based on this and [2] address the following tasks :

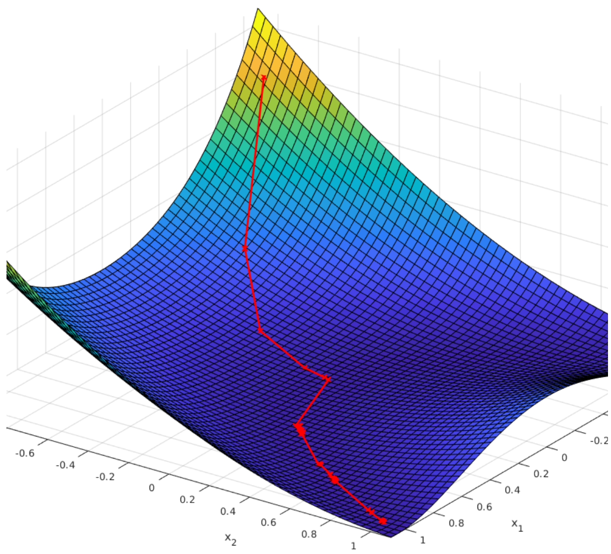

Figure 1.1: Illustration of a line search solution.

(i) Graphically illustrate the successive iterations (after the line search finished) using `plot3`, `ezsurf`, `hold`, and `grid` (see `Matlab` documentation and examples) so that you get something similar to fig. 1.1.

(ii) Suitably extend the provided template(s) to obtain the number of function evaluations and gradient computations. Use this feature in the following to compare the different algorithms. Additionally, modify the starting values and parameters for the step length computation used in `wolfe` in order to check your understanding of the method.
Hint: You can use `global` variables to do that or, alternatively, pass another parameter to the functions.

~~(iii)~~ Implement a function `quadraticInterpolation` to determine the step length α_k. Apply your results to the Rosenbrock problem and compare the number of iterations/function evaluations with the `wolfe` method (with the search direction obtained by steepest descent).

(iv) Implement a function `conjugatedGradient` for the determination of the search direction s_k and compare your results when β_k is obtained using either the *Fletcher–Reeves formula* or the *Polak–Ribière formula*.
Hint: append this as an option `betaMethod` to the `param` structure in the `Matlab oocLab1` script file and distinguish between the cases in the `Matlab lineSearch` function file similar to the options `stepLengthMethod` and `searchDirectionMethod`.

(v) Implement and compare the *Newton* and *Quasi-Newton* method (with DFP and BFGS update, see (2.55),(2.56) in [2] using additional options) for the determination of the search direction s_k. Note that for the Newton method, you need the Hessian of the Rosenbrock function (which should be passed as an additional anonymous function handle to the `handles` structure).

(vi) Review the stopping criterion `while( k < maxIter && fDist > epsf )` in `lineSearch`. What does `fDist` actually measure? Under which conditions would the minimization fail using this approach? Adapt the example to provide a suitable stopping criterion.

References

[1] J. Andrej, D. Siebelts, and S. Helling. *oocLab1*. https://cau-git.rz.uni-kiel.de/ACON/opt/optimization-and-optimal-control. 2020 (cit. on p. 1).

[2] T. Meurer. *Optimization and Optimal Control WS 20/21*. https://www.control.tf.uni-kiel.de/en/teaching/winter-term/optimization-and-optimal-control/fileadmin/opt_ws2021_full. 2020 (cit. on pp. 1, 2).