Project 2

NUMN21/FMNN25: Advanced Numerical Algorithms in Python

Robert Klöfkorn and Andreas Langer

This describes the second bigger programming project in the course, devoted to some classical optimization algorithms.

Task 1

Design an optimization problem class in Python. Its constructor should take an objective function as input, optionally also its gradient can be provided.

Task 2

Design a general optimization method class. It should be generic for special kinds of Quasi-Newton methods. Derive from this method class classes for special methods in the next tasks.

Task 3

Implement classical Newton's method for finding a minimum of the objective function. Approximate the Hessian by finite differences and a symmetrizing step: $G := \frac{1}{2}(\bar{G} + \bar{G}^{\mathrm{T}})$, if necessary.

Task 4

Provide Newton's method with exact line search method.

Task 5

Test the performance of this method on the Rosenbrock function

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

Demonstrate its behavior also in a way, comparable to the right picture in Fig. 1.

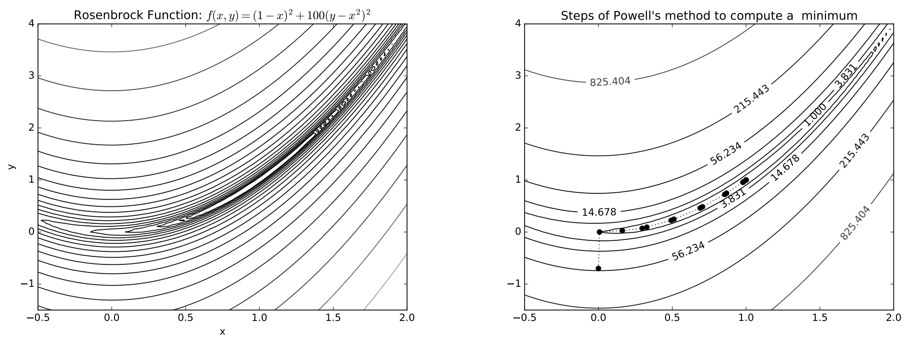

Figure 1: A contour plot of Rosenbrock's function and steps towards its minimum.

Task 6

Write an inexact line search method based on the Goldstein/Wolfe conditions (Algorithm: Fletcher pp.34ff in the additional course material)

Task 7

Test this seperately from an optimization method on Rosenbrock's function and use the parameters given on p.37 in the book mentioned above.

Task 8

Provide Newton's method with the inexact line search as an alternative to the exact one.

Task 9

Derive from Newton's method five classes of Quasi Newton methods:

- Simple Broyden rank-1 update of G and applying Sherman-Morisson's formula ("good Broyden")
- Simple Broyden rank-1 update of $H = G^{-1}$ ("bad Broyden")
- Symmetric Broyden update
- DFP rank-2 update
- BFGS rank-2 update

Task 10

Download from the course's webpage the testexample `chebyquad_problem.py`. It contains the objective function `chebyquad` and its gradient.

The objective function in mathematical terms reads:

$$f(x) = \sum_{i=0}^{n} \left(\frac{1}{n} \sum_{j=0}^{n} T_i(2x_j - 1) - \int_0^1 T_i(2\xi - 1) \mathrm{d}\xi \right)^2 \quad x = [x_0, \dots, x_n] \ \ x_j \in [0, 1] \tag{1}$$

where $T_i(t)$ is the i-th degree Chebyshev polynomial and $t \in [-1, 1]$.

Task 11

Minimizing `chebyquad` corresponds to finding a set of optimal points x_j such that the mean value in (1) approximates best the corresponding integrals. Use your code to compute these points for $n = 4, 8, 11$. Compare your results with those obtained from `scipy.optimize.fmin_bfgs`.

Task 12

The matrix $H^{(k)}$ of the BFGS method should approximate $G(x^{(k)})^{-1}$, where G is the Hessian of the problem. Study the quality of the approximation with growing k.

Lycka till!