Universität Konstanz Sommersemester 2016
Fachbereich Mathematik und Statistik
Prof. Dr. Stefan Volkwein
Sabrina Rogg

Optimierung

http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/

Program 3 (6 Points)

Submission by E-Mail: 27.06.2016, 10:00 h

Implementation of a globalized (Quasi-)Newton method

We modify the local Newton method from Program 2 such that it is globally convergent. In addition, we add a switch to a globalized BFGS method if the Hessian matrix of the considered function is not given. The resulting algorithm is defined in Algorithm 1 and will be implemented in the function `globalnewtonmethod`. Use the Matlab function `nargout` to identify if the Hessian is provided or not. Note that the inequality in Line 9 of Algorithm 1 can be interpreted as a generalized angle condition.
Write the function in the form

```
[X] = globalnewtonmethod(fhandle, x0, epsilon, alpha1, alpha2, p, ...
                        , t0, alpha, beta, nmax, amax)
```

Fix the parameters `p = 1/10`, `alpha1 = alpha2 = 1e-6` and test your program as follows:

1. Use the negative cosine function with the parameters from Program 2. Write an additional function file `negativeCosineNoHessian.m` which only returns the function- and gradient value. Use the initial points `x0 = 1.1656, 1.9, atan(-pi)` and `x0 = -3`.

2. Use the function `fun2` together with the parameters from Program 2. Again, implement an additional function `fun2NoHessian.m`. As done on Program sheet 2 plot the values of the objective function in semilog scale for the two methods under consideration and visualize the iterates of both methods along with the contour lines of the objective function.

3. Use the Rosenbrock function $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$, $x = (x_1, x_2)^\top \in \mathbb{R}^2$, with the parameter setting from Program 1 but with `nmax=100`, `epsilon = 1e-5` and starting points `[1;-0.5]` and `[-1.5; -1]`. Write the functions `rosenbrock.m` and `rosenbrockNoHessian.m`.

Compare the two methods under consideration and take a look at the following points: Does the Armijo algorithm have to reduce the (initial) step size 1? In case the exact Hessian is used: When is the algorithm forced to set $d^n = -\nabla f(x^n)$ (Line 10)? In case of BFGS: Is the algorithm forced to reset $H_{n+1} = I$ (Line 21)?

Discuss your observations in the written report and visualize your results in suitable plots.

Algorithm 1

Require: Initial point x^0, stopping tolerance $\varepsilon > 0$, maximal iteration number $n_{\max}$, $\alpha_1, \alpha_2 > 0$, $p > 0$, and (for Armijo) an initial step size t_0, $\alpha \in (0,1)$, $\beta \in (0,1)$, maximal iteration number $a_{\max}$

1: $n = 0$;
2: **if** $\nabla^2 f$ is given **then**
3:     $H_0 = \nabla^2 f(x^0)$
4: **else**
5:     $H_0 = I$
6: **end if**
7: **while** $\|\nabla f(x^n)\| > \varepsilon$ **and** $n < n_{\max}$ **do**
8:     Compute d^n by solving $H_n d^n = -\nabla f(x^n)$;
9:     **if** $\nabla^2 f$ is given and $-\nabla f(x^n)^\top d^n < \min\{\alpha_1, \alpha_2 \|d^n\|^p\}\|d^n\|^2$ **then**
10:         $d^n = -\nabla f(x^n)$
11:     **end if**
12:     Compute a stepsize t_n using Armijo rule (see Program 1);
13:     Set $x^{n+1} = x^n + t_n d^n$;
14:     **if** $\nabla^2 f$ is given **then**
15:         $H_{n+1} = \nabla^2 f(x^{n+1})$
16:     **else**
17:         $s^n = x^{n+1} - x^n$, $y^n = \nabla f(x^{n+1}) - \nabla f(x^n)$
18:         **if** $(y^n)^\top s^n > 0$ **then**
19:             Set $H_{n+1} = H_n + \frac{y^n (y^n)^\top}{(y^n)^\top s^n} - \frac{H_n s^n (H_n s^n)^\top}{(s^n)^\top H_n s^n}$
20:         **else**
21:             Set $H_{n+1} = I$
22:         **end if**
23:     **end if**
24:     Set $n = n + 1$;
25: **end while**
