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

Sommersemester 2015

Optimierung

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

Program 3 (6 Points)

Submission by E-Mail: 2015/06/29, 10:00 h

Implementation of a globalized (Quasi-)Newton method

Write Part 1 and Part 2 together in a main file `main.m`.

Part 1: Implement the local Newton method for optimization known from the lecture (Algorithm 5.6 with $F = \nabla f$). Write a function

```
function [X] = newtonmethod(fhandle, x0, epsilon, nmax)
```

with input arguments

- `fhandle`: function handle to a function of form `[f,g,H] = functionname(x)` (the output values are the function value, the gradient and the Hessian matrix corresponding to the input argument `x`).
- `x0`: initial point
- `epsilon`: tolerance for the termination condition $\|\nabla f(x^k)\| < \epsilon$
- `nmax` : maximum number of iterations

The program should return a matrix `X = [x0, x1, x2, ...]` containing the whole iterations.

Test your program by using the negative cosine function. Write herefore a function file `ncosH.m` which is of the above form. Use the parameters `epsilon = 1e-5` and `nmax = 50`. As initial points choose `x0 = 1.1655, 1.1656, 1.9, atan(-pi)`. Explain the results you get and use suitable plots for showing `X`. For comparison plot also the iterates you obtain by applying the function `gradmethod` from Program 1 with `t0 = 1`, `alpha = 1e-2`, `beta = 0.5` and `amax = 30`.

Part 2: In this part we modify the local Newton method 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 ankle condition.

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^A, $\alpha^A \in (0,1)$, $\beta^A \in (0,1)$, maximal iteration number $a_{\max}$

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

Write the function in the form

```
[X] = globalnewtonmethod(fhandle, x0, epsilon, alpha1, alpha2, p, ...

                        , t0, alpha, beta, nmax, amax)
```

Test your program as follows:

1. Use the negative cosine function as in Part 1 with additional parameters `p = 1/10` and `alpha1 = alpha2 = 1e-6`. Write herefore an additional function file `ncos.m` which only returns the function- and gradient value.

2. 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]`. Set `alpha1 = alpha2 = 1e-6` and

`p = 1/10`. Use the function file `rosenbrock.m` from Program 1 and write a new one, `rosenbrockH.m`, which additionally returns the Hessian matrix computed in `x`.

Compare the two methods under consideration and take a look at the following: 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)?

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