University of Mannheim
Scientific Computing, B6 26, C306, 68131 Mannheim
Dr. Andreas Sommer (ansommer@mail.uni-mannheim.de)

Nonlinear Optimization (FSS 2023)

Exercise Sheet #6

Due on 10.04.2023 (before 13:00).

1. **Stopping criterion for algorithms** [4 points]

   Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable, and $(x^k) \subset \mathbb{R}^n$ be a convergent sequence with limit $\tilde{x} \in \mathbb{R}^n, \nabla f(\tilde{x}) = 0$ and $\nabla^2 f(\tilde{x})$ nonsingular. Show that there exist $k_0 \in \mathbb{N}$ and $\beta > 0$, such that

   $$\|\nabla f(x^k)\| \geq \beta \|x^k - \tilde{x}\|$$

   for all $k \geq k_0$.

2. **Convergence of local Newton method** [4 points]

   Let $f : \mathbb{R} \to \mathbb{R}, f(x) = |x|^p$ for $p > 2$ and $x^0 > 0$. Apply the local (full-step) Newton method (Algorithm (5)) to determine the global minimum of f at $\tilde{x} = 0$, starting from initial guess x^0.

   Show that the local Newton method (Algorithm 5) converges linearly to $\tilde{x}$ and give the convergence rate. Show that the convergence is not super-linearly.

   Why is this no contradiction to the local convergence result from the lecture?

3. **Programming assignment: Globalized Newton Method** [6 points]

   Implement the globalized Newton method (Algorithm (6)). Use a function header

   ```
   function X = globalnewton(f, gradf, hessf, x0, e, maxit, beta, gamma, a1, a2, p)
   ```

   Here, `f`, `gradf` and `hessf` are functions which return for given x the functional value $f(x)$, the gradient $\nabla f(x)$ and the Hessian $\nabla^2 f(x)$, and further `x0` denotes the starting point. The iteration should be terminated if $\|\nabla f(x^k)\| \leq \epsilon$ for a given tolerance `e` $= \epsilon > 0$ or if a maximum number of iterations `maxit` has been reached. Parameters `beta` and `gamma` are β and γ from the Armijo rule, and `a1`, `a2`, `p` are the parameters α_1, α_2, p from the search direction condition (7.1) in line 4 of Algorithm (6). The return value `X` shall contain all iterates x^k as a matrix.

   Display in every iteration by `N` or `G`, whether the Newton direction or the Gradient direction has been chosen, i.e. Newton: `fprintf('N')`, gradient: `fprintf('G')`. Also display after each step size calculation whether a full step F or a reduced step `r` has been chosen, such that your program emits information like `GFNrNFGFGFGFNrNrNFNF` while progressing.

   Test your program on the following function [e is Euler's number here; e^x in Matlab: `exp(x)`]:

   $$f(x) = log(e^{x_1} + e^{-x_1}) + x_2^2$$

   using $x_a^0 = (1.0, -0.5)^\top$, $x_b^0 = (-1.2, 1.0)^\top$, $x_c^0 = (10.0, 12.0)^\top$, and $x_d^0 = (33.0, -62.0)^\top$, as initial guesses. Choose as parameters: `e` $= \varepsilon = 10^{-6}$, `maxiter`=100, $\beta = 0.5$, $\gamma = 10^{-4}$, $\alpha_1 = \alpha_2 = 10^{-6}$, and $p = 0.1$.

   Compare the results to the local Newton method from the previous exercise sheet.

Put all files into a single zip-archive, named by the lexicographically ordered family names of all group members separated by a hyphen, and send the zip-file to ansommer@mail.uni-mannheim.de.
Please add PDF files containing code and output (text and visual) to your submissions.
Comment your code intensely. Use a complete header that describes input and output arguments and also comment the implementation where appropriate (see the examples at the course web site). Avoid obvious inefficiencies like repeated evaluation of identical/unchanged expressions.