

Exercise 3: Unconstrained Newton-type Optimization and Globalization Strategies

(to be completed during exercise session on Nov 30, 2018 or sent by email to
messerer@tf.uni-freiburg before Dec 7, 2018)

Prof. Dr. Moritz Diehl, Dimitris Kouzoupis, Andrea Zanelli and Florian Messerer

Aim of this exercise is to become familiar with different Newton-type methods and learn their characteristics in practice. You will then combine a Quasi-Newton-method with globalization strategies to write your own solver for the hanging chain problem.

Exercise Tasks

1. **Regularization:** Prove that a regularized Newton-type step $x_{k+1} = x_k - (B_k + \alpha I)^{-1} \nabla f(x_k)$ with B_k a Hessian approximation, α a positive scalar and I the identity matrix of suitable dimensions, converges to a small gradient step $x_{k+1} = x_k - \frac{1}{\alpha} \nabla f(x_k)$ as $\alpha \rightarrow \infty$.
(2 points)
2. **Unconstrained minimization:** In this task we will implement different Newton-type methods that minimize the nonlinear function

$$f(x, y) = \frac{1}{2}(x - 1)^2 + \frac{1}{2}(10(y - x^2))^2 + \frac{1}{2}y^2. \quad (1)$$

You can use the provided MATLAB script to get an idea of the shape of the function.

- (a) Derive, first on paper, the gradient and Hessian matrix of the function in (1). Then, re-write it in the form $f(x, y) = \frac{1}{2} \|R(x, y)\|_2^2$ where $R : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ is the residual function. Derive the Gauss-Newton Hessian approximation and compare it with the exact one. When do the two matrices coincide?
(2 points)
- (b) Implement your own Newton method with exact Hessian information and full steps. Start from the initial point $(x_0, y_0) = (-1, 1)$ and use as termination condition $\|\nabla f(x_k, y_k)\|_2 \leq 10^{-3}$. Keep track of the iterates (x_k, y_k) and add them to the provided contour plot.
(2 points)
- (c) Update your code to use the Gauss-Newton Hessian approximation instead. Plot the difference between exact and approximate Hessian as a function of the iterations (evaluate both the exact and the Gauss-Newton Hessian at the iterates generated by the Gauss-Newton algorithm). Use the MATLAB function `norm` to measure this difference.
(2 points)
- (d) Check how the steepest descent method performs on this example. Your Hessian approximation now is αI where α is a positive scalar and I the identity matrix. Try $\alpha = 100, 200$ and 500 . For which values does your algorithm converge?
(1 point)
- (e) Compare the performance of the implemented methods. Consider the iteration path (x_k, y_k) , the number of iterations and the run time. You can use MATLAB's `tic toc` to measure time.
(1 point)

3. **Lifted Newton method:** Consider the scalar nonlinear function $F(w) = w^{16} - 2$.

- (a) Implement in MATLAB the Newton method in order to numerically find a root of $F(w)$. Use $\|F(w)\|_2 < 10^{-12}$ as convergence criterion. Plot how the residuals evolve. Test the algorithm for different initial guesses and analyze the behaviour of the algorithm.

(1 point)

- (b) Implement now a Newton-type algorithm that exploits a fixed approximation of the Jacobian

$$w^{k+1} = w^k - M^{-1}F(w^k),$$

where $M = \nabla^\top F(w_0)$ is the Jacobian of F at the initial guess w_0 . Use the conditions for local Newton-Type convergence (Theorem 8.4) to derive a bound on the convergence region. Test numerically for which region(s) of initial values w_0 the algorithm converges (in 10^4 iterations or less).

(2 points)

- (c) An equivalent problem to (a) can be obtained by *lifting* the original one to a higher dimensional space

$$\tilde{F}(w) = \begin{bmatrix} w_2 & - & w_1^2 \\ w_3 & - & w_2^2 \\ w_4 & - & w_3^2 \\ -2 & + & w_4^2 \end{bmatrix}.$$

Implement the Newton method for this lifted problem and compare the convergence of the two algorithms. Use $w_0 = 100$. Initialize all variables w_i of the lifted method at this value.

(1 point)

- (d) Show that the Newton method is guaranteed to converge to a root (if it exists) for the root finding problem $F(x) = 0$, where F is any monotonically increasing convex differentiable function $F : \mathbb{R} \rightarrow \mathbb{R}$.

(1 point)

4. **Convergence of damped Newton's method:** Let f be a twice continuously differentiable function satisfying $LI \succeq \nabla^2 f(x) \succeq mI$ for some $L > m > 0$ and let x^* be the unique minimizer of f over \mathbb{R}^n .

- (a) Show that for any $x \in \mathbb{R}^n$:

$$f(x) - f(x^*) \geq \frac{m}{2} \|x - x^*\|^2,$$

(1 point)

- (b) Let $\{x_k\}_{k \geq 0}$ be the sequence generated by the damped Newton's method with constant stepsize $t_k = \frac{m}{L}$. Show that:

$$f(x_k) - f(x_{k+1}) \geq \frac{m}{2L} \nabla f(x_k)^\top (\nabla^2 f(x_k))^{-1} \nabla f(x_k).$$

(2 points)

- (c) Show that $x_k \rightarrow x^*$ as $k \rightarrow \infty$.

(2 points)

5. **Hanging chain, revisited:** In this task you will solve the unconstrained minimization problem of the hanging chain using the BFGS method in combination with back tracking and the Armijo condition. Consider the non-convex case where the $N + 1$ springs can be both compressed and expanded from their rest length $L_i = L/(N + 1)$. Recall that in this case the objective function is given by:

$$V_{\text{chain}}(y, z) = \frac{1}{2} \sum_{i=0}^N D(\sqrt{(y_i - y_{i+1})^2 + (z_i - z_{i+1})^2} - L_i)^2 + g_0 \sum_{i=0}^{N+1} m z_i. \quad (2)$$

Note that the indices range from 0 to $N+1$. This is in order to fix the chain ends without formulating an equality constraint, i.e. $y_0 = -2$, $y_{N+1} = 2$ and $z_0 = z_{N+1} = 1$ are treated as parameters. Choosing the indices like this we still have decision variables y_1, \dots, y_N and z_1, \dots, z_N . The provided MATLAB function `[F] = hc_obj(x, param)` returns the value of this nonlinear function for a given set of parameters defined in the data structure `param` and a point `x` ordered as $x = [y_1, z_1, \dots, y_N, z_N]^T$.

- (a) Write your own MATLAB function `[F, J] = finite_difference(fun, x, param)` that calculates the function value and the Jacobian of function `fun` at `x` using finite differences. Note that the argument `fun` is a *function handle*. You can then call your function using the syntax `[F, J] = finite_difference(x, @hc_fun, param)` to evaluate the Jacobian of our objective at `x`. Calling `eps` in MATLAB returns floating point precision.

(2 points)

- (b) Complete the template file `main.m` to find the rest position of the hanging chain using the steepest descent method with backtracking and the Armijo condition.

(2 points)

- (c) Now update your code to perform BFGS updates on your Hessian approximation how many iterations does your new scheme need to converge?

Remark: The BFGS Hessian approximation is guaranteed to be positive-definite if and only if the curvature condition $s^T y > 0$ holds. A common workaround to ensure that the search direction is always a descent direction is to check whether this condition holds or not and to skip the BFGS update in case positive-definiteness is not guaranteed.

(2 points)

This sheet gives in total 26 points.