Exercises for Lecture Course on Numerical Optimization (NUMOPT)
Albert-Ludwigs-Universität Freiburg – Winter Term 2022-2023

Exercise 3: Unconstrained Newton-type Optimization, Globalization Strategies

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

Exercise Tasks

1. **Unconstrained minimization:** In this task we will implement different Newton-type methods for solving the problem

$$\min_{x,\,y\,\in\,\mathbb{R}} \quad \underbrace{\frac{1}{2}(x-1)^2+\frac{1}{2}(10(y-x^2))^2+\frac{1}{2}y^2}_{=:f(x,y)}. \tag{1a}$$

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 $f(x,y)$. Then, rewrite it in the form $f(x,y)=\frac{1}{2}||R(x,y)||_2^2$ where $R:\mathbb{R}^2\to\mathbb{R}^3$ is the residual function. Derive the Gauss-Newton Hessian approximation and compare it with the exact one. When do the two matrices coincide?

$$\nabla f(x,y)=\begin{bmatrix}200x^3-200xy+x-1\\ 101y-100x^2\end{bmatrix} \qquad \nabla^2 f(x,y)=\begin{bmatrix}1-200y+600x^2 & -200x\\ -200x & 101\end{bmatrix}$$

$$R(x,y):=\begin{bmatrix}x-1 & 10(y-x^2) & y\end{bmatrix}^\top \qquad \nabla R(x,y)^\top=J(x,y)=\begin{bmatrix}1 & 0\\ -20x & 10\\ 0 & 1\end{bmatrix}$$

$$B_{\mathrm{GN}}(x,y):=J(x,y)^\top J(x,y)=\begin{bmatrix}1+400x^2 & -200x\\ -200x & 101\end{bmatrix}$$

$$y=x^2 \Rightarrow \nabla^2 f(x,y)=B_{\mathrm{GN}}(x,y)$$

(b) Implement your own Newton method with exact Hessian information and full steps. Start from the initial guess $(x_0,y_0)=(-1,1)$ and use as termination condition $\|\nabla f(x_k,y_k)\|_\infty\le 10^{-3}$. Keep track of the iterates (x_k,y_k) and add them to the provided contour plot.

(c) Update by adding a Newton-type method that uses the Gauss-Newton Hessian approximation instead. Add the resulting iterates (x_k,y_k) to the plot.

(d) Check how the steepest descent method performs on this example. Your Hessian approximation is now αI, where α is a positive scalar and I the identity matrix of appropriate size. Try $\alpha=100,200$ and 500. Add the resulting iterates (x_k,y_k) to the plot. For which values does your algorithm converge?

(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 `timeit.default_timer()` in Python, `tic toc` in MATLAB to measure time.

2. **Lifted Newton method:** Consider the scalar nonlinear function $F : \mathbb{R} \to \mathbb{R}$, $F(w) = w^{16} - 2$.

   (a) In Matlab, implement Newton's 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 convergence behaviour of the algorithm.

   (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 the superscript $[k]$ denotes the iteration index and $M = \nabla F(w^{[0]})^\top$ 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 $w^{[k]}$ the algorithm converges (in 10^4 iterations or less).
   Consider $w_* = 2^{\frac{1}{16}}$. Local convergence if and only if

   $$\rho\left(I - \nabla F(w_0)^{-1}\nabla F(w_*)\right) = \rho(1 - \frac{w_*^{15}}{w_0^{15}}) = |1 - \frac{w_*^{15}}{w_0^{15}}| < 1$$

   Find w_0 for which we get equality. Since $w_* > 0$ we immediately see that for $w_0 < 0$ we always have $|1 - \frac{w_*^{15}}{w_0^{15}}| > 1$, and for $w_0 = 0$ a singularity. If we thus restrict the analysis to $w_0 > 0$, it always holds that $1 - \frac{w_*^{15}}{w_0^{15}} < 1$. To find the boundary of the convergence region, we therefore need to solve $1 - \frac{w_*^{15}}{w_0^{15}} = -1 \Leftrightarrow w_0 = 2^{-\frac{1}{15}} w_* \approx 0.9971$. We have local (and therefore global) convergence only if w_0 is larger than this value. For $w_0 \gg 0.9971$ we always have $\rho < 1$. Nonetheless we also have $\rho \approx 1$. Therefore convergence for large w_0 convergence becomes painfully slow.
   Finally: Be aware that all of this analysis is only about *local* convergence. We do not say anything about global convergence.

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

   $$\widetilde{F}(\omega) = \begin{bmatrix} \omega_2 & - & \omega_1^2 \\ \omega_3 & - & \omega_2^2 \\ \omega_4 & - & \omega_3^2 \\ -2 & + & \omega_4^2 \end{bmatrix},$$

   such that for any root $\bar{\omega}$ with $\widetilde{F}(\bar{\omega}) = 0$, it holds that $F(\bar{\omega}_1) = 0$, i.e., ω_1 corresponds to w, and $\bar{w} = \bar{\omega}_1$ is a root of F if $\bar{\omega}$ is a root of $\widetilde{F}$. This can be seen via

   $$\tilde{F}(\bar{\omega}) = 0 \Leftrightarrow \bar{\omega}_2 = \bar{\omega}_1^2,\ \bar{\omega}_3 = \bar{\omega}_2^2,\ \bar{\omega}_4 = \bar{\omega}_3^2,\ \bar{\omega}_4^2 - 2 = 0. \tag{2}$$

   $$0 = -2 + \bar{\omega}_4^2 = -2 + (\bar{\omega}_3^2)^2 = \ldots = -2 + \bar{\omega}_1^{16} = F(\bar{\omega}_1). \tag{3}$$

   While $F(w)$ is a strongly nonlinear function with unbounded curvature, $\widetilde{F}(\omega)$ – though still nonlinear – is linear quadratic and therefore has bounded curvature (actuall even a constant Hessian.)
   Implement the Newton method for this lifted problem and compare the convergence of the two algorithms. Use $w^{[0]} = 100$, and correspondingly $\omega_1^{[0]} = w^{[0]}$, $\omega_2^{[0]} = (\omega_1^{[0]})^2, \ldots$.
   Also note that for the lifted method you have more flexibility of initializing. Try what happens if you initialize all components as $\omega_i^{[0]} = 100$ for $i = 1, \ldots, 4$.

3. **Convergence of damped Newton's method:** Let $f : \mathbb{R}^n \to \mathbb{R}$ 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 $x \in \mathbb{R}^n$:

$$f(x) - f(x^*) \geq \frac{m}{2}\|x - x^*\|_2^2,$$

Hint: Use the Taylor rest term formula (cf. proof of Thrm. 3.2, last equation)

Taylor rest term formula with $\varphi = x^* + \theta(x - x^*)$ for some $\theta \in [0, 1]$:

$$\begin{aligned} f(x) &= f(x^*) + \nabla f(x^*)^\top (x - x^*) + \frac{1}{2}(x - x^*)^\top \nabla^2 f(\varphi)(x - x^*) \\ f(x) - f(x^*) &= \underbrace{\nabla f(x^*)^\top}_{=0}(x - x^*) + \frac{1}{2}(x - x^*)^\top \underbrace{\nabla^2 f(\varphi)}_{\succeq mI}(x - x^*) \\ &\geq \frac{1}{2}(x - x^*)^\top mI(x - x^*) = \frac{m}{2}\|x - x^*\|_2^2 \end{aligned}$$

(b) Let $\{x_k\}_{k\geq 0}$ be the sequence generated by the damped Newton's method with constant stepsize $t_k = \frac{m}{L}$, such that $x_{k+1} = x_k + t_k p_k$, where p_k is the full Newton step. Show that:

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

Damped newton step: $\Delta x_k = -t_k \nabla^2 f(x_k)^{-1} \nabla f(x_k)$ where $t_k = \frac{m}{L}$

Taylor rest term formula with $\xi = x_k + \bar{\theta}(x_{k+1} - x_k)$ for some $\bar{\theta} \in [0, 1]$:

$$\begin{aligned} f(x_k) - f(x_{k+1}) &= -\nabla f(x_k)^\top \Delta x_k - \frac{1}{2}\Delta x_k^\top \nabla^2 f(\xi) \Delta x_k \\ &= \frac{m}{L}\nabla f(x_k)^\top \nabla^2 f(x_k)^{-1} \nabla f(x_k) - \frac{m^2}{2L^2}\nabla f(x_k)^\top \left(\nabla^2 f(x_k)\right)^{-1} \nabla^2 f(\xi) \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) \\ &= \frac{m}{2L}\nabla f(x_k)^\top \left(\nabla^2 f(x_k)\right)^{-\frac{1}{2}} \left(2I - \frac{m}{L}\left(\nabla^2 f(x_k)\right)^{-\frac{1}{2}} \underbrace{\nabla^2 f(\xi)}_{\preceq LI} \left(\nabla^2 f(x_k)\right)^{-\frac{1}{2}}\right) \left(\nabla^2 f(x_k)\right)^{-\frac{1}{2}} \nabla f(x_k) \\ &\geq \frac{m}{2L}\nabla f(x_k)^\top \left(\nabla^2 f(x_k)\right)^{-\frac{1}{2}} \left(2I - m\underbrace{\left(\nabla^2 f(x_k)\right)^{-1}}_{\preceq \frac{1}{m} I}\right) \left(\nabla^2 f(x_k)\right)^{-\frac{1}{2}} \nabla f(x_k) \\ &\geq \frac{m}{2L}\nabla f(x_k)^\top \left(\nabla^2 f(x_k)\right)^{-\frac{1}{2}} \left(2I - I\right) \left(\nabla^2 f(x_k)\right)^{-\frac{1}{2}} \nabla f(x_k) \\ &= \frac{m}{2L}\nabla f(x_k)^\top \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) \end{aligned}$$

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

From $\nabla^2 f(x) \succeq mI$ and $m > 0$ follows that $\nabla^2 f(x) \succ 0$. Therefore SOSC holds and $\nabla f(x) = 0 \Rightarrow x = x^*$. Further $\nabla^2 f(x)^{-1} \succ 0$.

We have that $f(x_k) - f(x_{k+1}) \geq \frac{m}{2L}\nabla f(x_k)^\top \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) > 0$ as long as $\nabla f(x_k) \neq 0$ in which case we would have found the minimum. Therefore $f(x_{k+1}) < f(x_k)$ for $x_k \neq x^*$. Further we have $f(x_k) \geq f(x^*)$. From this follows that x_k converges to some $\bar{x}$ with $f(\bar{x}) \geq f(x^*)$ and $f(x_k) \to f(\bar{x})$. Therefore we get $f(x_k) - f(x_{k+1}) \to f(\bar{x}) - f(\bar{x}) = 0$ which implies that $\nabla f(\bar{x}) = 0$ and therefore $\bar{x} = x^*$.

4. **Hanging chain, revisited:** We revisit the hanging chain problem from the first exercise sheet. So far, our problem formulation uses the assumption that the springs have a rest length $L = 0$, which is not very realistic. A more realistic model includes the rest length L in the potential energy of the string in the following way:

$$V_{\text{el}}(y_i, y_{i+1}, z_i, z_{i+1}) = \frac{1}{2} D \left(\sqrt{(y_i - y_{i+1})^2 + (z_i - z_{i+1})^2} - L\right)^2, \quad i = 1, \ldots, N-1, \tag{4}$$

where $L = l/(N-1)$ and l the length of the chain. Note that setting $L = 0$ we obtain the same expression as in Exercise sheet 1.

In this task you will solve the unconstrained minimization problem of the hanging chain using the BFGS method in combination with backtracking and the Armijo condition. The objective function is given by

$$V_{\text{chain}}(y, z) = \frac{1}{2} \sum_{i=0}^{N} D(\sqrt{(y_i - y_{i+1})^2 + (z_i - z_{i+1})^2} - L_i)^2 + g_0 \sum_{i=0}^{N+1} m\, z_i. \tag{5}$$

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 $y_1, \ldots, y_N$ and $z_1, \ldots, z_N$. The provided 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 $\mathtt{x} = [y_1, z_1, \ldots, y_N, z_N]^\top$.

(a) Write your own 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(hc_fun,x,param)` in Python,
`[F,J] = finite_difference(@hc_fun,x,param)` in MATLAB (note the @ before the function handle argument) to evaluate the Jacobian of our objective at `x`.
Hint: Calling `numpy.finfo(float).eps` or `sys.float_info.epsilon` in Python, `eps` in MATLAB returns floating point precision.

(b) Complete the template file `main.m` to find the rest position of the hanging chain using the constant Hessian approximation $B_k = I$ (i.e., steepest descent) with backtracking and the Armijo condition to ensure convergence.

(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 weather this condition holds or not and to skip the BFGS update in case positive-definiteness is not guaranteed.