The Chinese University of Hong Kong, Shenzhen
SDS · School of Data Science
Xiao Li · Andre Milzarek · Zizhuo Wang · Spring Semester 2023

MAT 3007 — Optimization

Exercise Sheet 8

Problem 1 (A One-Dimensional Problem): (approx. *20* pts)
Consider the minimization problem

$$\min_{x\in\mathbb{R}} \ f(x) \qquad \text{s.t.} \qquad x \in [0,1],$$

where f is given by $f(x) := e^{-x} - \cos(x)$. Solve this problem using the bisection **or** the golden section method. Compare the number of iterations required to recover a solution in $[0,1]$ with accuracy less or equal than 10^{-5}.

Problem 2 (Descent Directions): (approx. *20* pts)
Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function and consider $x \in \mathbb{R}^n$ with $\nabla f(x) \neq 0$. Verify the following statements:

a) Set $d = -(\nabla f(x)_j)\cdot e_j = -\frac{\partial f}{\partial x_j}(x)\cdot e_j$, where $e_j \in \mathbb{R}^n$ is the j-th unit vector and $j \in \{1,\dots,n\}$ is an index satisfying

$$\left|\frac{\partial f}{\partial x_j}(x)\right| = \max_{1\le i\le n} \left|\frac{\partial f}{\partial x_i}(x)\right| = \|\nabla f(x)\|_\infty.$$

Then, d is a descent direction of f at x.

b) The direction $d = -\nabla f(x)/\|\nabla f(x)\|$ is a descent direction of f at x.

c) Let f be twice continuously differentiable and define $d_i = -(\nabla f(x)_i)/\max\{\nabla^2 f(x)_{ii}, \varepsilon\}$ for all $i \in \{1,\dots,n\}$ and for some $\varepsilon > 0$. Then, d is well-defined (we do not divide by zero) and it is a descent direction of f at x.

Problem 3 (The Gradient Method): (approx. *35* pts)
In this exercise, we want to solve the optimization problem

$$\min_{x\in\mathbb{R}^2} \ f(x) := x_1^4 + 2(x_1 - x_2)x_1^2 + 4x_2^2 \tag{1}$$

via the gradient descent method. (This is problem 1 discussed in sheet 6).

Implement the gradient method that was presented in the lecture in `MATLAB` or `Python`.

The following input functions and parameters should be considered:

- `obj`, `grad` – function handles that calculate and return the objective function $f(x)$ and the gradient $\nabla f(x)$ at an input vector $x \in \mathbb{R}^n$. You can treat these handles as functions or fields of a class or structure `f` or you can use f and ∇f directly in your code. (For example, your function can have the form `gradient_method(obj,grad,...)`).
- x^0 – the initial point.

- `tol` – a tolerance parameter. The method should stop whenever the current iterate x^k satisfies the criterion $\|\nabla f(x^k)\| \leq \texttt{tol}$.

We want to investigate the performance of the gradient method for different step size strategies. In particular, we want to test and compare backtracking and exact line search. The following parameters will be relevant for these strategies:

- $\sigma, \gamma \in (0,1)$ – parameters for backtracking and the Armijo condition. (At iteration k, we choose α_k as the largest element in $\{1, \sigma, \sigma^2, \dots\}$ satisfying the condition $f(x^k - \alpha_k \nabla f(x^k)) - f(x^k) \leq -\gamma\alpha_k \cdot \|\nabla f(x^k)\|^2$).
- You can use the golden section method to determine the exact step size α_k. The parameters for the golden section method are: `maxit` (maximum number of iterations), `tol` (stopping tolerance), $[0, \texttt{a}]$ (the interval of the step size).

You can organize the latter parameters in an appropriate `options` class or structure. It is also possible to implement separate algorithms for backtracking and exact line search. The method(s) should return the final iterate x^k that satisfies the stopping criterion.

a) Apply the gradient method with backtracking and parameters $(\sigma, \gamma) = (0.5, 0.1)$ and exact line search ($\texttt{maxit} = 100$, $\texttt{tol} = 10^{-6}$, $\texttt{a} = 2$) to solve the problem $\min_x f(x)$.

   The algorithms should use the stopping tolerance $\texttt{tol} = 10^{-5}$. Test the methods using the initial point $x^0 = (3, -3)^\top$ and report the performance of the methods, i.e., compare the number of iterations and the point to which the different gradient methods converged.

b) Let us define the set of ten initial points

$$\mathcal{X}^0 := \left\{ \begin{pmatrix} -4 \\ -4 \end{pmatrix}, \begin{pmatrix} -4 \\ 0 \end{pmatrix}, \begin{pmatrix} -4 \\ 4 \end{pmatrix}, \begin{pmatrix} -2 \\ -4 \end{pmatrix}, \begin{pmatrix} -2 \\ 4 \end{pmatrix}, \begin{pmatrix} 2 \\ -4 \end{pmatrix}, \begin{pmatrix} 2 \\ 4 \end{pmatrix}, \begin{pmatrix} 4 \\ -4 \end{pmatrix}, \begin{pmatrix} 4 \\ 0 \end{pmatrix}, \begin{pmatrix} 4 \\ 4 \end{pmatrix} \right\}.$$

   Run the methods:

   - Gradient descent method with backtracking and $(\sigma, \gamma) = (0.5, 0.1)$,
   - Gradient method with exact line search and $\texttt{maxit} = 100$, $\texttt{tol} = 10^{-6}$, $\texttt{a} = 2$,
   - General descent method using the direction d from **Problem 2 a)** as descent direction (you can either use backtracking $(\sigma, \gamma) = (0.5, 0.1)$ or exact line search ($\texttt{maxit} = 100$, $\texttt{tol} = 10^{-6}$, $\texttt{a} = 2$) to determine the step sizes)

   for every initial point in the set $\mathcal{X}^0$ using the tolerance $\texttt{tol} = 10^{-5}$. For each algorithm/step size strategy create a single figure that contains all of the solution paths generated for the different initial points. The initial points and limit points should be clearly visible. Add a contour plot of the function f in the background of each figure.

Problem 4 (Globalized Newton's Method): (approx. *25* pts)
Implement the globalized Newton method with backtracking that was presented in the lecture as a function `newton_glob` in `MATLAB` or `Python`.

The pseudo-code for the full Newton method is given below. The following input functions and parameters should be considered:

- x^0 – the initial point.
- `tol` – a tolerance parameter. The method should stop whenever the current iterate x^k satisfies the criterion $\|\nabla f(x^k)\| \leq \texttt{tol}$.

Algorithm 1: The Globalized Newton Method

```
1 Initialization:   Select an initial point $x^0 \in \mathbb{R}^n$ and parameter $\gamma, \gamma_1, \gamma_2, \sigma \in (0,1)$ and tol.
  for $k = 0, 1, \dots$ do
2     If $\|\nabla f(x^k)\| \le$ tol, then STOP and $x^k$ is the output.
3     Compute the Newton direction $s^k$ as solution of the linear system of equations:
          $\nabla^2 f(x^k) s^k = -\nabla f(x^k).$
4     If $-\nabla f(x^k)^\top s^k \ge \gamma_1 \min\{1, \|s^k\|^{\gamma_2}\}\|s^k\|^2$, then accept the Newton direction and set $d^k = s^k$.
       Otherwise set $d^k = -\nabla f(x^k)$.
5     Choose a step size $\alpha_k$ by backtracking and calculate $x^{k+1} = x^k + \alpha_k d^k$.
```

- `obj`, `grad`, `hess` – function handles that calculate and return the objective function $f(x)$, the gradient $\nabla f(x)$, and the Hessian $\nabla^2 f(x)$ at an input vector $x \in \mathbb{R}^n$. You can treat these handles as functions or fields of a class or structure `f` or you can use f, ∇f, and $\nabla^2 f$ from part a) and b) directly in the algorithm. (For example, your function can have the form `newton_glob(obj,grad,hess,...)`).
- $\gamma_1, \gamma_2 > 0$ – parameters for the Newton condition.
- $\sigma, \gamma \in (0,1)$ – parameters for backtracking and the Armijo condition.

You can again organize the latter parameters in an appropriate `options` class or structure. You can use the backslash operator `A\b` in `MATLAB` or `numpy.linalg.solve(A,b)` to solve the linear system of equations $Ax = b$. If the computed Newton step $s^k = -\nabla^2 f(x^k)^{-1}\nabla f(x^k)$ is a descent direction and satisfies

$$-\nabla f(x^k)^\top s^k \ge \gamma_1 \min\{1, \|s^k\|^{\gamma_2}\}\|s^k\|^2,$$

we accept it as next direction d^k. Otherwise, the gradient direction $d^k = -\nabla f(x^k)$ is chosen. The method should return the final iterate x^k that satisfies the stopping criterion.

a) Test your approach on the Rosenbrock function $f : \mathbb{R}^2 \to \mathbb{R}$

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

with initial point $x^0 = (-1, -0.5)^\top$ and parameter $(\sigma, \gamma) = (0.5, 10^{-4})$ and $(\gamma_1, \gamma_2) = (10^{-6}, 0.1)$. (Notice that γ is smaller here). Besides the globalized Newton method also run the gradient method with backtracking ($(\sigma, \gamma) = (0.5, 10^{-4})$) on this problem and compare the performance of the two approaches using the tolerance `tol` $= 10^{-7}$.

Does the Newton method always utilize the Newton direction? Does the method always use full step sizes $\alpha_k = 1$?

b) Repeat the performance test from **Problem 3 b)** for problem (1) with the globalized Newton method. You can use $(\sigma, \gamma) = (0.5, 0.1)$, $(\gamma_1, \gamma_2) = (10^{-6}, 0.1)$, and `tol` $= 10^{-5}$.

Plot all of the solution paths obtained by Newton's method for the different initial points in $\mathcal{X}^0$ in one figure (with a contour plot of f in the background).

Sheet 8 is due on **April, 28th**. Submit your solutions before **April, 28th**, **12:00 pm (noon)**.