MATH-329 Nonlinear optimization
Exercise session 5: Trust-region with Cauchy step

Instructor: Nicolas Boumal
TAs: Andrew McRae, Andreea Musat

Document compiled on October 2, 2024

Exercises marked with $(*)$ will be used in later exercises or in the homeworks: you might want to prioritize those.

1. The Cauchy step. Consider the quadratic model for $f\colon \mathbb{R}^n \to \mathbb{R}$ at $x \in \mathbb{R}^n$ given by

$$m(v) = f(x) + \langle \nabla f(x), v\rangle + \frac{1}{2}\langle v, Hv\rangle$$

for some symmetric matrix $H \in \mathbb{R}^{n\times n}$. We let $\Delta > 0$ be a radius and we let $g = \nabla f(x)$ denote the gradient.

1. Remember that the Cauchy step is defined by

$$u^C = -t^C \cdot g \qquad \text{with} \qquad t^C \in \operatorname*{arg\,min}_{0\leq t\leq \frac{\Delta}{\|g\|}} m(-t\cdot g).$$

   Show that the step size is given by

$$t^C = \begin{cases} \min\left(\frac{\|g\|^2}{\langle g, Hg\rangle}, \frac{\Delta}{\|g\|}\right) & \text{if } \langle g, Hg\rangle > 0, \\ \frac{\Delta}{\|g\|} & \text{otherwise.} \end{cases}$$

2. Show that the Cauchy step leads to the following decrease in model value

$$m(0) - m(u^C) \geq \frac{1}{2}\min\left(\Delta, \frac{\|g\|}{\|H\|}\right)\|g\|.$$

2. $(*)$ Implementing the TR algorithm.

1. Implement the trust-region algorithm (see lecture notes). Use $H_k = \nabla^2 f(x_k)$ for the quadratic model and the Cauchy step to approximately solve the trust-region subproblem.

2. Consider the n-dimensional Rosenbrock function (see the definition at the end of the exercise sheet). We provide files on Moodle to compute the function value, the gradient and the Hessian. Run your implementation of the TR algorithm with $n = 10$ and $x_0 =$ `randn(n,1)`. You may choose parameters $\bar{\Delta} = \sqrt{n}$, $\Delta_0 = \bar{\Delta}/8$ and $\rho' = 0.1$.

3. Compare the performance with the line-search gradient descent algorithm. Both algorithms should have approximately the same convergence speed; can you guess why? Can you see pros and cons for TR with Cauchy steps rather than line-search gradient descent? Soon we will see how to exploit second-order information better to greatly improve the convergence speed.

Multidimensional Rosenbrock function. We generalize the Rosenbrock function in n dimensions as

$$f(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (1 - x_i)^2 \right].$$

The vector of ones is the unique global minimum (because the function is non-negative and is zero if and only if all entries are ones). The gradient at $x \in \mathbb{R}^n$ is given by

$$\nabla f(x)_i = \begin{cases} -2(1 - x_1) - 400x_1(x_2 - x_1^2) & \text{if } i = 1 \\ 200(x_i - x_{i-1}^2) - 2(1 - x_i) - 400x_i(x_{i+1} - x_i^2) & \text{if } 1 < i < n \\ 200(x_n - x_{n-1}^2) & \text{if } i = n. \end{cases}$$

The Hessian at x is a symmetric tridiagonal $n \times n$ matrix. The main diagonal and the first diagonal above are given by

$$\begin{bmatrix} 2 + 1200x_1^2 - 400x_2 \\ 202 + 1200x_2^2 - 400x_3 \\ \vdots \\ 202 + 1200x_{n-1}^2 - 400x_n \\ 200 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -400x_1 \\ \vdots \\ -400x_{n-1} \end{bmatrix}$$

respectively. In practice we never build the full matrix but solely compute matrix/vector products. This can be done efficiently because the matrix is sparse.