UNIVERSITÄT BERN

Applied Optimization
Exercise 8
Trust Region and Equality Constrained Optimization

November 11, 2021

Hand-in instructions:

Please hand-in **only one** compressed file named after the following convention: `Exercise`n`-GroupMemberNames.zip`, where n is the number of the current exercise sheet. This file should contain:

- The complete code folder except for the `build` subfolder. Make sure that your code submission compiles on your machine. Zip the code folder.
- A `readme.txt` file containing a description on how you solved each exercise (use the same numbers and titles) and the encountered problems.
- Other files that are required by your `readme.txt` file. For example, if you mention some screenshot images in `readme.txt`, these images need to be submitted too.
- Submit your solutions to ILIAS before the submission deadline.

Trust Region Subproblem: Exact Solution (6 pts) + (6 pt bonus)

Consider the following trust region subproblem in two dimensions, $v \in \mathbb{R}^2$:

$$\begin{aligned} &\text{minimize } m(v) = g^T v + \frac{1}{2} v^T B v \\ &\text{subject to: } \|v\| \leq \Delta \end{aligned}$$

as discussed in [Nocedal, Sec 4.3] and the slides. In every problem, given the inputs B, g and Δ, solve the task and plot the quadratic model and the trust region with the tool.

Problem 1:

$$B = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}, \quad g = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \Delta = 1/2$$

Compute an approximation of v^* with Cauchy point method.

Problem 2:

$$B = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}, \quad g = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \Delta = 1/2$$

Compute an approximation of v^* with the Dogleg method.

Problem 3:

$$B = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}, \quad g = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \Delta = 1$$

Compute the λ^* and v^* which satisfy the KKT optimality conditions.

Problem 4:

$$B = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}, \quad g = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \Delta = 1/2$$

Compute the λ^* and v^* which satisfy the KKT optimality conditions.

Problem 5 (bonus):

$$B = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}, \quad g = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \quad \Delta = 1/2$$

Compute the λ^* and v^* which satisfy the KKT optimality conditions.

Problem 6: (bonus)

Consider now the the indefinite matrix

$$\tilde{B} = \begin{pmatrix} 4 & 0 \\ 0 & -2 \end{pmatrix}, \quad g = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \Delta = 1$$

Compute the λ^* and v^* which satisfy the KKT optimality conditions.

Problem 7: (bonus)

Same parameters as in problem 6, but now the gradient is zero $g = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

Programming (4 pts)

Equality Constrained Newton Method

In previous exercises, we solve the mass spring system with constrained spring elements by adding some penalty terms to the objective function

$$E = \sum_{e(a,b)} E_{a,b} + \sum_{n \in C} \frac{1}{2} penalty_n ||(\mathbf{x_n} - \mathbf{p_n})||^2$$

In this exercise, we instead solve an equality constrained optimization problem to achieve the same result. The problem can be described as following:

$$\begin{aligned} \text{minimize} \quad & E = \sum_{e(a,b)} E_{a,b} \\ \text{subject to} \quad & Ax = b \end{aligned}$$

It can be solved by the equality constrained Newton method. The newton step Δx can be computed by solving the KKT system as follows:

$$\begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \nu \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ 0 \end{bmatrix}$$

You could refer to the lecture slides for the pseudo code "Newton's Method with Equality Constraints".

The four corner nodes in the spring graph are constrained to certain positions by specifying the equality constraints $Ax = b$. The code is provided in `setup_linear_equality_constraints(...)` function in the `MassSpringSystemT_impl.hh` file. Implement the algorithm in the `solve_equality_constrained(...)` function in the `NewtonMethods.hh` file.

Note that the equality constrained Newton works only when it starts from a feasible point which means the equality constraint is satisfied. Otherwise a projection operation that projects an infeasible point to the feasible set is needed. Implement it in the `project_on_affine(...)` function.
Hint: You can project your infeasible starting point x_0 to the hyperplane $Ax = b$ by finding a δx such that $A(x_0 + \delta x) = b$. Ask yourself what system you could solve to find such a δx

You can test your implementation by running the `EqualityConstrainedNewton` executable on random mass spring systems and visualize the results on the usual website. As usual, you will find helpful unit tests alongside the main executable.