UNIVERSITÄT BERN

Applied Optimization
Exercise 10
Inequality Constrained Optimization

December 6, 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.

Active set method for convex quadratic problem (6 pts)

Consider the following 2-dimensional minimization problem

$$\begin{aligned} \text{min} \quad & (x-1)^2 + (y-2.5)^2 \\ \text{subject to:} \quad & x - 2y + 2 \geq 0, \quad (C_1) \\ & -x - 2y + 6 \geq 0, \quad (C_2) \\ & -x + 2y + 2 \geq 0, \quad (C_3) \\ & x \geq 0, \quad (C_4) \\ & y \geq 0 \quad (C_5) \end{aligned}$$

use the active set method for convex QPs starting from the following points:

1. $(x^0, y^0) = (1, 1/2)^T$ for $\mathcal{A}^0 = \emptyset$
2. $(x^0, y^0) = (3, 1/2)^T$ for $\mathcal{A}^0 = \{3\}$
3. $(x^0, y^0) = (0, 1/2)^T$ for $\mathcal{A}^0 = \{4\}$

Sketch the feasible set and for every starting point:

1. Sketch the path of the active set method.
2. Verify the path of the active set method computationally.
   Note for each step the following states in the table below:
   - the primal variables before the step (x_k, y_k)
   - the active set before the step A_k
   - the primal variables after the step (x_{k+1}, y_{k+1})
   - the dual variables after the step λ_{k+1}, e.g. $(0, 1, -1, 0, 0)$
   - the active set after the step A_{k+1}
   - whether the KKT conditions are satisfied for $(x_{k+1}, y_{k+1}, \lambda_{k+1})$

k	(x_k, y_k)	A_k	(x_{k+1}, y_{k+1})	λ_{k+1}	A_{k+1}	KKT sat.?
0	$(1, 1/2)$	$\emptyset$				
1						
2						
3						

[Hint: See Nocedal example 16.4 p. 475. and exercise 16.10]

”Phase I” method for initial feasible point (2 pts)

The active set method used in the task above requires an initial feasible point. Use the ”Phase I” approach as described in the lecture and the book [Nocedal, Section 16.5] to compute an initial feasible point for the problem above, starting from $x = (1, 2.5)^T$. You can use a computer algebra system if needed. [Hint: formulate the ”Phase I” problem by adding slack variables for violated constraints at the start point. Solve the active set subproblem with active set method or gradient descent (projected newon) in combination with constraint elimination].

”Big M” method for initial feasible point (2 bonus pts)

Consider now the ”Big M” method and use it to compute the initial point also starting from $x = (1, 2.5)^T$ with $M = 100$.

Programming (5 pts)

Augmented Lagrangian Method

Implement the Augmented Lagrangian Method as discussed in the last lecture by providing source code for:

- `AugmentedLagrangian::solve(...)` in `AugmentedLagrangian.hh`
- ToDo functions in file `AugmentedLagrangianProblem.hh`
- ToDo functions in file `CircleConstraint2D.hh`
- ToDo functions in file `CircleConstraintSquared2D.hh`

The new class `AugmentedLagrangianProblem` is a new layer to convert the given objective function and two sets of constraints (h_i and h_i^2) into an augmented Lagrangian objective function.

In this exercise, the boundary graph nodes are quadratically constrained to be on a circle that is centered at the coordinate $(\frac{m}{2.0}, \frac{n}{2.0})$ with radius $r = \frac{m+n}{2.0}$, where m and n are the dimensions of the $m \times n$ spring graph. See `add_boundary_constraints()` in the file `MassSpringSystemT_impl.hh` file for more details. Try both spring element functions, see how the system behaves under the boundary condition.
Note: This time, the local-to-global indexing is handled *inside* the constraint functions. See the code for the details.

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