Assignment 10

Problem 1: The QP approximation

NLP:

$$\min_{x} \; f(x) \qquad \text{s.t} \quad c(x) = 0 \qquad (I)$$

a) Starting from the NLP problem in (I). The Lagrangian function for the problem is

$$\mathcal{L}(x, \lambda) = f(x) - \lambda^T c(x)$$

Since $f(x)$, $c(x)$ and thus $\mathcal{L}(x, \lambda)$ is in general non-linear, we want to approximate them with Taylor-expansions around the point x_k, for a small perturbation $x_{k+1} = x_k + p$

For the constraints we use a 1st order approximation:

$$c(x_{k+1}) = c(x_k + p)$$

$$\approx c(x_k) + \frac{dc}{dx}(x_k) \cdot (\cancel{x_k} + p - \cancel{x_k})$$

$$\approx c(x_k) + \frac{dc}{dx}(x_k) p$$

$$\frac{dc}{dx}(x) = \begin{bmatrix} \nabla_x c_1(x)^T \\ \nabla_x c_2(x)^T \\ \vdots \\ \nabla_x c_m(x)^T \end{bmatrix} = A(x)$$

$$\longrightarrow c(x_{k+1}) \approx c(x_k) + A(x_k) p \overset{!}{=} 0$$

For the Lagrangian we use a 2nd order approximation:

$$\mathcal{L}(x_{k+1}; \lambda_k)$$

$$= \mathcal{L}(x_k + p, \lambda_k)$$

$$\approx \mathcal{L}(x_k, \lambda_k) + \frac{d\mathcal{L}}{dx}(x_k, \lambda_k)(\cancel{x_k} + p - \cancel{x_k})$$

$$+ \frac{1}{2}(\cancel{x_k} + p - \cancel{x_k})^T \frac{d^2\mathcal{L}}{dx^2}(x_k, \lambda_k)(\cancel{x_k} + p - \cancel{x_k})$$

$$\approx \mathcal{L}(x_k, \lambda_k) + \nabla_x \mathcal{L}(x_k, \lambda_k)^T p + \frac{1}{2} p^T \nabla_{xx} \mathcal{L}(x_k, \lambda_k) p$$

$$\Longrightarrow \mathcal{L}(x_{k+1}, \lambda_k)$$

$$\approx \mathcal{L}(x_k, \lambda_k) + \nabla_x \mathcal{L}(x_k, \lambda_k)^T p + \frac{1}{2} p^T \nabla_{xx} \mathcal{L}(x_k, \lambda_k) p$$

$$\nabla_x \mathcal{L}(x, \lambda)$$

$$= \frac{d\mathcal{L}}{dx}(x, \lambda) = \frac{d}{dx}\left(f(x) - \lambda^T c(x)\right)$$

$$= \nabla_x f(x) - \left(\frac{dc}{dx}(x)\right)^T \lambda$$

$$= \nabla_x f(x) - A(x)^T \lambda$$

$$\longrightarrow \mathcal{L}(x_{k+1}, \lambda_k)$$

$$\approx f(x_k) - \lambda_k^T c(x_k) + \left(\nabla_x f(x_k) - A(x_k)^T \lambda_k\right)^T p$$

$$+ \frac{1}{2} p^T \nabla_{xx} \mathcal{L}(x_k, \lambda_k) p$$

$$\approx f(x_k) + \lambda_k^T c(x_k) + \nabla_x f(x_k)^T p$$

$$- \lambda_k^T A(x_k) p + \frac{1}{2} p^T \nabla_{xx} \mathcal{L}(x_k, \lambda_k) p$$

$$\approx f(x_k) - \lambda_k^T \underbrace{\left(c(x_k) + A(x_k)p\right)}_{=0}$$

$$+ \nabla_x f(x_k)^T p + \frac{1}{2} p^T \nabla_{xx} \mathcal{L}(x_k, \lambda_k) p$$

$$\Longrightarrow \mathcal{L}(x_{k+1}, \lambda_k)$$

$$\approx \underline{f(x_k) + \nabla_x f(x_k)^T p + \frac{1}{2} p^T \nabla_{xx} \mathcal{L}(x_k, \lambda_k) p}$$

$$\min \quad f(x_k) + \nabla_x f(x_k)^T p + \frac{1}{2} p^T \nabla_{xx} \mathcal{L}(x_k, \lambda_k) p$$

$$\text{s.t.} \quad c(x_k) + A(x_k) p = 0$$

b)

QP problem:

$$\min \quad f(x_k) + \nabla_x f(x_k)^T p + \frac{1}{2} p^T \nabla_{xx}^2 \mathcal{L}(x_k, \lambda_k) p$$
$$\text{s.t.} \quad c(x_k) + A(x_k) p = 0$$

1st. order KKT conditions:

$$\nabla_p \mathcal{L}_g(p, \gamma) = 0$$
$$c_g(p) = c(x_k) + A(x_k) p = 0$$

$$\mathcal{L}_g(p, \gamma)$$
$$= f(x_k) + \nabla_x f(x_k)^T p + \frac{1}{2} p^T \nabla_{xx}^2 \mathcal{L}(x_k, \lambda_k) p$$
$$- \gamma^T (c(x_k) + A(x_k) p)$$

$$\nabla_p \mathcal{L}_g(p, \gamma)$$
$$= \nabla_x f(x_k) + \nabla_{xx}^2 \mathcal{L}(x_k, \lambda_k) p - A(x_k)^T \gamma = 0$$

$$\Longrightarrow \nabla_{xx}^2 \mathcal{L}(x_k, \lambda_k) p - A(x_k)^T \gamma = - \nabla_x f(x_k)$$

Write as a matrix equation, the 1st order KKT conditions become:

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k) & -A(x_k)^T \\ A(x_k) & 0 \end{bmatrix} \begin{bmatrix} p \\ \gamma \end{bmatrix} = \begin{bmatrix} -\nabla_x f(x) \\ -c(x_k) \end{bmatrix}$$

Problem 2: Merit functions

a) Merit functions are used to evaluate whether a trial step

$$x_{k+1} = x_k + \alpha_k p_k$$

should be accepted. In the algorithm the merit functions are placed in a conditional while loop.

b) $\phi_1(x;\mu) = f(x) + \mu \sum_{i \in \mathcal{E}} |c_i(x)| + \mu \sum_{i \in \mathcal{I}} [c_i(x)]^-$

$\phi_2(x;\mu) = f(x) + \mu \|c(x)\|_2$

$\phi_F(x;\mu) = f(x) - \lambda(x)^T c(x) + \frac{1}{2}\mu \sum_{i \in \mathcal{E}} c_i(x)^2$

c) The parameter μ penalizes the constraints by adding a weight to them, thus the feasibility of a solution matters.

Usually, μ is initially small and grows larger for each iteration of the SQP algorithm.

d) An exact merit function $\phi(x;\mu)$ is defined if there exists a positive scalar μ^*, such that for any $\mu > \mu^*$ any local solution of a NLP is a local minimizer of $\phi(x;\mu)$.

ϕ_1, ϕ_2 and ϕ_F are all exact merit functions.

e) The Maratos effect is when optimization algorithms based on merit functions converge slowly due to rejection of steps that would result in significant progress of the optimization problem.

ϕ_1 and ϕ_2 suffers from the Maratos effect.

f) Merit functions usually decrease from one iteration to the next they are used in the line-search of the SQP algorithm. The line-search does not terminate until a decrease has been found.

g) Since a merit function is used to evaluate the step from one iteration to the next and not the objective function, the objective function does not in general decrease from one iteration to the next.

This is different from unconstrained optimization, LP and QP, where we could formulate the algorithms so that the objective function would decrease from one iteration to the next.

Problem 3 : Feasibility and local solutions

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & c_i(x) = 0 , \quad i \in \mathcal{E} \\ & c_i(x) \geq 0 , \quad i \in \mathcal{I} \end{aligned}$$

a) The SQP algorithm does not set any requirements to the feasibility of x_0, since the algorithm uses merit functions to weight constraints.

This is different from the simplex algorithm for LP problems and the active set for QP problems, both of which required that the initial solution x_0 was within the feasible set of solutions.

b) The iterates does not necessarily have to be feasible. This is different from the simplex algorithm for LP problems and active set algorithm for QP problems, where all the iterates were within the feasible set of the problems.

c) The NLP problem can have multiple local solutions if the problem is nonconvex.

d) A simple method for trying to find other solutions would be to use the same SQP algorithm, but with different initial iterates (x_0, λ_0).