# MS-E2122 - Nonlinear Optimization
# Lecture X

Fernando Dias

Department of Mathematics and Systems Analysis

Aalto University
School of Science

November 8, 2023

# Outline of this lecture

## Methods of feasible directions

- Feasible direction methods
- Conditional gradient: the Frank-Wolfe method
- Sequential quadratic programming - SQP

# Outline of this lecture

Methods of feasible directions

- Feasible direction methods
- Conditional gradient: the Frank-Wolfe method
- Sequential quadratic programming - SQP

# The concept of feasible direction

Algorithms of this type progress taking into account two aspects:

1. $x_k + \lambda d$ is feasible
2. $f(x_k + \lambda d_k) \leq f(x_k)$.

Since primal feasibility is observed, these methods are also called primal methods.

However, some variants do not necessarily retain feasibility during the iterations.

# The concept of feasible direction

Algorithms of this type progress taking into account two aspects:

1. $x_k + \lambda d$ is feasible
2. $f(x_k + \lambda d_k) \leq f(x_k)$.

Since primal feasibility is observed, these methods are also called primal methods.

However, some variants do not necessarily retain feasibility during the iterations.

We will discuss 2 main types:

1. Conditional gradient: Frank-Wolfe method;
2. Sequential quadratic programming - SQP.

# Obtaining improving feasible directions

Let us first revisit the definition of an improving feasible direction.

**Definition 1**

Consider the problem min. $\{f(x) : x \in S\}$ with $f : \mathbb{R}^n \to \mathbb{R}$ and $\emptyset \neq S \subseteq \mathbb{R}^n$. A vector $d$ is a feasible direction at $x \in S$ if exists $\delta > 0$ such that $x + \lambda d \in S$ for all $\lambda \in (0, \delta)$. Moreover, $d$ is an improving feasible direction at $x \in S$ if there exists a $\delta > 0$ such that $f(x + \lambda d) < f(x)$ and $x + \lambda d \in S$ for $\lambda \in (0, \delta)$.

# Obtaining improving feasible directions

Let us first revisit the definition of an improving feasible direction.

### Definition 1

Consider the problem min. $\{f(x) : x \in S\}$ with $f : \mathbb{R}^n \to \mathbb{R}$ and $\emptyset \neq S \subseteq \mathbb{R}^n$. A vector $d$ is a feasible direction at $x \in S$ if exists $\delta > 0$ such that $x + \lambda d \in S$ for all $\lambda \in (0, \delta)$. Moreover, $d$ is an improving feasible direction at $x \in S$ if there exists a $\delta > 0$ such that $f(x + \lambda d) < f(x)$ and $x + \lambda d \in S$ for $\lambda \in (0, \delta)$.

Feasible direction methods work as follows. Given $x^k \in S$

1. Obtain an improving feasible direction $d^k$ and a step size $\lambda^k$;
2. Make $x^{k+1} = x^k + \lambda^k d^k$.

# Obtaining improving feasible directions

Let us first revisit the definition of an improving feasible direction.

**Definition 1**

Consider the problem min. $\{f(x) : x \in S\}$ with $f : \mathbb{R}^n \to \mathbb{R}$ and $\emptyset \neq S \subseteq \mathbb{R}^n$. A vector $d$ is a feasible direction at $x \in S$ if exists $\delta > 0$ such that $x + \lambda d \in S$ for all $\lambda \in (0, \delta)$. Moreover, $d$ is an improving feasible direction at $x \in S$ if there exists a $\delta > 0$ such that $f(x + \lambda d) < f(x)$ and $x + \lambda d \in S$ for $\lambda \in (0, \delta)$.

Feasible direction methods work as follows. Given $x^k \in S$

1. Obtain an improving feasible direction $d^k$ and a step size $\lambda^k$;
2. Make $x^{k+1} = x^k + \lambda^k d^k$.

**Remark:** Obtaining $d^k$ and $\lambda^k$ have to be easier to solve than the original problem for the method to make sense.

# The Frank-Wolfe method

Recall that, if $\nabla f(x^k)$ is a feasible descent direction, then

$$\nabla f(x^k)^\top (x - x^k) < 0 \text{ for } x \in S.$$

A straightforward way to obtain improving feasible directions $d = (x - x^k)$ is by solving the direction search problem $DS$.

$$(DS): \quad \text{min.} \quad \left\{ \nabla f(x^k)^\top (x - x^k) : x \in S \right\}.$$

# The Frank-Wolfe method

Recall that, if $\nabla f(x^k)$ is a feasible descent direction, then

$$\nabla f(x^k)^\top (x - x^k) < 0 \text{ for } x \in S.$$

A straightforward way to obtain improving feasible directions $d = (x - x^k)$ is by solving the direction search problem $DS$.

$$(DS): \;\; \text{min.} \;\; \left\{ \nabla f(x^k)^\top (x - x^k) : x \in S \right\}.$$

Letting $\overline{x}^k = \arg\min_{x \in S} \left\{ \nabla f(x^k)^\top (x - x^k) \right\}$ and obtaining $\overline{\lambda}^k \in (0, 1]$, the method iterates making

$$x^{k+1} = x^k + \lambda^k (\overline{x}^k - x^k).$$

**Remark:** for convex $S$, $\lambda^k \in (0, 1]$ guarantees feasibility.

# The Frank-Wolfe method

**Algorithm** Frank-Wolfe method

1: **initialise.** $\epsilon > 0, x^0 \in S, k = 0.$
2: **while** $|\nabla f(x)^\top d^k| > \epsilon$ **do**
3: $\quad \overline{x}^k = \arg\min \left\{ \nabla f(x^k)^\top d : x \in S \right\}.$
4: $\quad d^k = \overline{x}^k - x^k$
5: $\quad \lambda^k = \arg\min_\lambda \left\{ f(x^k + \lambda d^k) : 0 \leq \lambda \leq \overline{\lambda} \right\}.$
6: $\quad x^{k+1} = x^k + \lambda^k d^k; k = k + 1.$
7: **end while**
8: **return** $x^k$.

# The Frank-Wolfe method

**Algorithm** Frank-Wolfe method

1: **initialise.** $\epsilon > 0, x^0 \in S, k = 0.$
2: **while** $|\nabla f(x)^\top d^k| > \epsilon$ **do**
3: $\quad \overline{x}^k = \arg\min \left\{\nabla f(x^k)^\top d : x \in S\right\}.$
4: $\quad d^k = \overline{x}^k - x^k$
5: $\quad \lambda^k = \arg\min_\lambda \left\{f(x^k + \lambda d^k) : 0 \leq \lambda \leq \overline{\lambda}\right\}.$
6: $\quad x^{k+1} = x^k + \lambda^k d^k; k = k + 1.$
7: **end while**
8: **return** $x^k$.

**Remarks:**

1. For $f(x)$ nonlinear and a polyhedral feasible region $S$, the subproblems $DS$ are linear programming problems.
2. Can be employed with Armijo to ease the line search for challenging $f(x)$.

# The Frank-Wolfe method

**Example:** min. $\{e^{-(x_1-3)/2} + e^{(4x_2+x_1-20)/10} + e^{(-4x_2+x_1)/10} : 2x_1 + 3x_2 \leq 8, x_1 + 4x_2 \leq 6\}$. The first iteration...

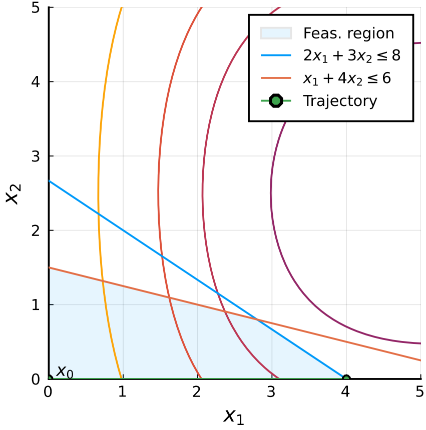



Figure: The FW method with exact line search.

# The Frank-Wolfe method

**Example:** min. $\{e^{-(x_1-3)/2} + e^{(4x_2+x_1-20)/10} + e^{(-4x_2+x_1)/10}$ : $2x_1 + 3x_2 \leq 8, x_1 + 4x_2 \leq 6\}$. All iterations.

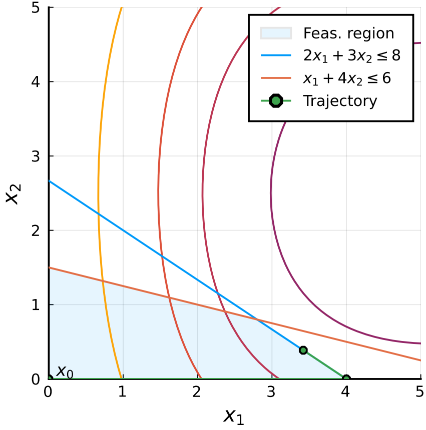



Figure: Total of 2 iterations are required for $e = 10^{-4}$.

# The Frank-Wolfe method

**Example:** min. $\{e^{-(x_1-3)/2} + e^{(4x_2+x_1-20)/10} + e^{(-4x_2+x_1)/10} : 2x_1 + 3x_2 \leq 8, x_1 + 4x_2 \leq 6\}$. All iterations with Armijo.

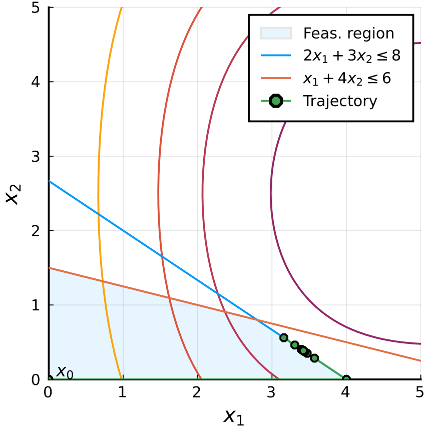



Figure: Total of 15 iterations are required for $e = 10^{-4}$.

# Sequential quadratic programming - SQP

SQP is inspired on the idea of employing Newton's method to solve the KKT system directly.

# Sequential quadratic programming - SQP

SQP is inspired on the idea of employing Newton's method to solve the KKT system directly.

Let $P = \text{min. } \{f(x) : h_i(x) = 0, i = 1, \dots, l\}$. The KKT conditions for $P$ are

$$W(x, v) = \begin{cases} \nabla f(x) + \sum_{i=1}^{l} v_i \nabla h_i(x) = 0 \\ h_i(x) = 0, i = 1, \dots, l. \end{cases}$$

# Sequential quadratic programming - SQP

SQP is inspired on the idea of employing Newton's method to solve the KKT system directly.

Let $P = \text{min. } \{f(x) : h_i(x) = 0, i = 1, \ldots, l\}$. The KKT conditions for $P$ are

$$W(x,v) = \begin{cases} \nabla f(x) + \sum_{i=1}^{l} v_i \nabla h_i(x) = 0 \\ h_i(x) = 0, i = 1, \ldots, l. \end{cases}$$

Using Newton(-Raphson) to solve $W(x,v)$ at $(x^k, v^k)$, we obtain

$$W(x^k, v^k) + \nabla W(x^k, v^k) \begin{bmatrix} x - x^k \\ v - v^k \end{bmatrix} = 0. \tag{1}$$

# Sequential quadratic programming - SQP

Let $L(x, v) = f(x) + v^\top h(x)$ be the Lagrangian function and

$$\nabla^2 L(x^k, v^k) = \nabla^2 f(x^k) + \sum_{i=1}^{l} v_i^k \nabla^2 h_i(x^k)$$

its Hessian at $x^k$. Thus

$$\nabla W(x^k, v^k) = \begin{bmatrix} \nabla^2 L(x^k, v^k) & \nabla h(x^k)^\top \\ \nabla h(x^k) & 0 \end{bmatrix}.$$

# Sequential quadratic programming - SQP

Let $L(x, v) = f(x) + v^\top h(x)$ be the Lagrangian function and

$$\nabla^2 L(x^k, v^k) = \nabla^2 f(x^k) + \sum_{i=1}^{l} v_i^k \nabla^2 h_i(x^k)$$

its Hessian at $x^k$. Thus

$$\nabla W(x^k, v^k) = \begin{bmatrix} \nabla^2 L(x^k, v^k) & \nabla h(x^k)^\top \\ \nabla h(x^k) & 0 \end{bmatrix}.$$

Setting $d = (x - x^k)$, we can rewrite (1) as

$$\nabla^2 L(x^k, v^k) d + \nabla h(x^k)^\top v = -\nabla f(x^k) \quad (2)$$

$$\nabla h(x^k) d = -h(x^k), \quad (3)$$

which can be repeatedly solved until $||(x^k, v^k) - (x^{k-1}, v^{k-1})|| = 0$, i.e., convergence, is observed. Then, $(x^k, v^k)$ is a KKT point.

# Sequential quadratic programming - SQP

Instead of solving a Newton system, SQP relies on successively solving the problem:

$$QP(x^k, v^k) : \text{min.} \quad f(x^k) + \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d \quad (4)$$

$$\text{subject to:} \quad h_i(x^k) + \nabla h_i(x^k)^\top d = 0, i = 1, \dots, l, \quad (5)$$

to which optimality conditions are given by (2) and (3).

# Sequential quadratic programming - SQP

Instead of solving a Newton system, SQP relies on successively solving the problem:

$$QP(x^k, v^k) : \text{min.} \quad f(x^k) + \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d \quad (4)$$

$$\text{subject to: } h_i(x^k) + \nabla h_i(x^k)^\top d = 0, i = 1, \dots, l, \quad (5)$$

to which optimality conditions are given by (2) and (3).

Two alternative ways of interpreting this objective function:

1. a second-order approximation of $f(x)$, also considering a term $(1/2) \sum_{i=1}^{l} v_i^k d^\top \nabla^2 h_i(x^k) d$ representing constraint curvature;

# Sequential quadratic programming - SQP

Instead of solving a Newton system, SQP relies on successively solving the problem:

$$QP(x^k, v^k) : \text{min.} \quad f(x^k) + \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d \quad (4)$$

$$\text{subject to: } h_i(x^k) + \nabla h_i(x^k)^\top d = 0, i = 1, \dots, l, \quad (5)$$

to which optimality conditions are given by (2) and (3).

Two alternative ways of interpreting this objective function:

1. a second-order approximation of $f(x)$, also considering a term $(1/2) \sum_{i=1}^{l} v_i^k d^\top \nabla^2 h_i(x^k) d$ representing constraint curvature;
2. Let $L(x, v) = f(x) + \sum_{i=1}^{l} v_i h_i(x)$. Then, (4) can be seen as the second-order approximation of $L(x, v)$,

$$L(x^k, v^k) + \nabla_x L(x^k, v^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d$$

which explains its alternative name: projected Lagrangian.

# Sequential quadratic programming - SQP

To see (2), notice that

$$L(x, v) \approx L(x^k, v^k) + \nabla_x L(x^k, v^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d =$$

$$f(x_k) + {v^k}^\top h(x^k) + (\nabla f(x^k) + {v^k}^\top \nabla h(x^k))^\top d$$

$$+ \frac{1}{2} d^\top (\nabla^2 f(x^k) + \sum_{i=1}^{l} v_i^k \nabla^2 h_i(x^k)) d$$

and that $\nabla h(x^k)^\top (x - x^k) = 0$ (from (5), as $h(x^k) = 0$).

# Sequential quadratic programming - SQP

To see (2), notice that

$$L(x,v) \approx L(x^k,v^k) + \nabla_x L(x^k,v^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k,v^k) d =$$

$$f(x_k) + {v^k}^\top h(x^k) + (\nabla f(x^k) + {v^k}^\top \nabla h(x^k))^\top d$$

$$+ \frac{1}{2} d^\top (\nabla^2 f(x^k) + \sum_{i=1}^{l} v_i^k \nabla^2 h_i(x^k)) d$$

and that $\nabla h(x^k)^\top (x - x^k) = 0$ (from (5), as $h(x^k) = 0$). For the general case, we have

$$\begin{aligned} QP(x^k,u^k,v^k) : \text{min.} \ & \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k,u^k,v^k) d \\ \text{subject to: } & g_i(x^k) + \nabla g_i(x^k)^\top d \leq 0, i = 1,\dots,m \\ & h_i(x^k) + \nabla h_i(x^k)^\top d = 0, i = 1,\dots,l, \end{aligned}$$

where $L(x,u,v) = f(x) + \sum_{i=1}^{m} u_i g_i(x) + \sum_{i=1}^{l} v_i h_i(x)$.

# Sequential quadratic programming - SQP

A pseudocode for the standard SQP method is presented below.

**Algorithm** SQP method

1: **initialise.** $\epsilon > 0, x^0 \in S, u^0 \geq 0, v^0, k = 0.$
2: **while** $||d^k|| > \epsilon$ **do**
3: $\quad d^k = \arg\min QP(x^k, u^k, v^k)$
4: $\quad$ obtain $u^{k+1}, v^{k+1}$ from $QP(x^k, u^k, v^k)$
5: $\quad x^{k+1} = x^k + d^k, k = k + 1.$
6: **end while**
7: **return** $x^k$.

# Sequential quadratic programming - SQP

A pseudocode for the standard SQP method is presented below.

**Algorithm** SQP method

1: **initialise.** $\epsilon > 0, x^0 \in S, u^0 \geq 0, v^0, k = 0.$
2: **while** $||d^k|| > \epsilon$ **do**
3: $\quad d^k = \arg\min QP(x^k, u^k, v^k)$
4: $\quad$ obtain $u^{k+1}, v^{k+1}$ from $QP(x^k, u^k, v^k)$
5: $\quad x^{k+1} = x^k + d^k, k = k + 1.$
6: **end while**
7: **return** $x^k$.

**Remark:** notice that the step in Line 5 requires dual variable values, which can be trivially recovered from simplex-based solvers.

# Sequential quadratic programming - SQP

Some relevant aspects:

1. Can be used in conjunction with quasi-Newton (BFGS) to approximate $\nabla^2 L(x^k, v^k)$.

# Sequential quadratic programming - SQP

Some relevant aspects:

1. Can be used in conjunction with quasi-Newton (BFGS) to approximate $\nabla^2 L(x^k, v^k)$.
2. Closely mimics convergence properties of Newton's method, i.e., under appropriate conditions, quadratic (superlinear) convergence is observed.

# Sequential quadratic programming - SQP

Some relevant aspects:

1. Can be used in conjunction with quasi-Newton (BFGS) to approximate $\nabla^2 L(x^k, v^k)$.
2. Closely mimics convergence properties of Newton's method, i.e., under appropriate conditions, quadratic (superlinear) convergence is observed.
3. Can exploit efficient (dual) simplex solvers.

# Sequential quadratic programming - SQP

Some relevant aspects:

1. Can be used in conjunction with quasi-Newton (BFGS) to approximate $\nabla^2 L(x^k, v^k)$.
2. Closely mimics convergence properties of Newton's method, i.e., under appropriate conditions, quadratic (superlinear) convergence is observed.
3. Can exploit efficient (dual) simplex solvers.
4. Can consider general nonlinear constraints, using first-order approximations.

# Sequential quadratic programming - SQP

Some relevant aspects:

1. Can be used in conjunction with quasi-Newton (BFGS) to approximate $\nabla^2 L(x^k, v^k)$.
2. Closely mimics convergence properties of Newton's method, i.e., under appropriate conditions, quadratic (superlinear) convergence is observed.
3. Can exploit efficient (dual) simplex solvers.
4. Can consider general nonlinear constraints, using first-order approximations.
5. Line searches cannot be easily performed, because feasibility is only implicitly considered in $QP(x^k, v^k)$

# Sequential quadratic programming - SQP

Some relevant aspects:

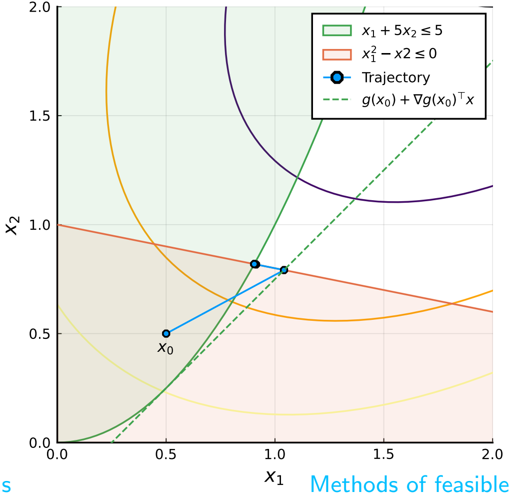
1. Can be used in conjunction with quasi-Newton (BFGS) to approximate $\nabla^2 L(x^k, v^k)$.
2. Closely mimics convergence properties of Newton's method, i.e., under appropriate conditions, quadratic (superlinear) convergence is observed.
3. Can exploit efficient (dual) simplex solvers.
4. Can consider general nonlinear constraints, using first-order approximations.
5. Line searches cannot be easily performed, because feasibility is only implicitly considered in $QP(x^k, v^k)$
6. Might present divergence, in a similar way than Newton's method, if started too far from the optimum.

# Sequential quadratic programming - SQP

**Example:** min. $\{2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 : x_1^2 - x_2 \leq 0,$
$x_1 + 5x_2 \leq 5, x_1 \geq 0, x_2 \geq 0\}$

# Sequential quadratic programming - SQP

**Example:** min. $\{2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 : x_1^2 - x_2 \leq 0,$
$x_1 + 5x_2 \leq 5, x_1 \geq 0, x_2 \geq 0\}$



# Sequential quadratic programming - SQP

The $l_1$-SQP is a variant that addresses divergence issues while presenting superior computational performance.

- Relies on a similar principle of penalty methods, encoding penalisation for infeasibility in the objective function.
- This allows for considering line searches or trust regions, which in turn can guarantee convergence.

# Sequential quadratic programming - SQP

The $l_1$-SQP is a variant that addresses divergence issues while presenting superior computational performance.

- Relies on a similar principle of penalty methods, encoding penalisation for infeasibility in the objective function.
- This allows for considering line searches or trust regions, which in turn can guarantee convergence.

Let us consider the trust-region $l_1$-penalty $QP$ subproblem:

$$\begin{aligned} \text{min.} \ & \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d \\ & + \mu \left[ \sum_{i=1}^{m} [g_i(x^k) + \nabla g_i(x^k)^\top d]^+ + \sum_{i=1}^{l} |h_i(x^k) + \nabla h_i(x^k)^\top d| \right] \\ \text{subject to:} \ & -\Delta^k \leq d \leq \Delta^k, \end{aligned}$$

where $\mu$ is a penalty term, $[\cdot] = \max\{0, \cdot\}$, and $\Delta^k$ is a trust region term.

# Sequential quadratic programming - SQP

$l_1 - QP(x^k, v^k)$ can be recast as a QP with linear constraints:

$$l_1 - QP(x^k, v^k):$$

$$\text{min. } \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d + \mu \left[ \sum_{i=1}^{m} y_i + \sum_{i=1}^{l} (z_i^+ - z_i^-) \right]$$

$$\text{subject to: } -\Delta^k \leq d \leq \Delta^k$$

$$y_i \geq g_i(x^k) + \nabla g_i(x^k)^\top d, i = 1 \ldots, m$$

$$z_i^+ - z_i^- = h_i(x^k) + \nabla h_i(x^k)^\top d, i = 1, \ldots, l$$

$$y, z^+, z^- \geq 0$$

# Sequential quadratic programming - SQP

$l_1 - QP(x^k, v^k)$ can be recast as a QP with linear constraints:

$$
\begin{aligned}
& l_1 - QP(x^k, v^k) : \\
& \quad \text{min. } \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d + \mu \left[ \sum_{i=1}^{m} y_i + \sum_{i=1}^{l} (z_i^+ - z_i^-) \right] \\
& \text{subject to: } -\Delta^k \leq d \leq \Delta^k \\
& \qquad y_i \geq g_i(x^k) + \nabla g_i(x^k)^\top d, i = 1 \dots, m \\
& \qquad z_i^+ - z_i^- = h_i(x^k) + \nabla h_i(x^k)^\top d, i = 1, \dots, l \\
& \qquad y, z^+, z^- \geq 0
\end{aligned}
$$

**Remarks:**

1. $l_1$-SQP is globally convergent (does not diverge) and enjoys superlinear convergence rate.
2. the $l_1$ term is often called a merit function in the literature.