Numerical Optimization

Graduate Course

Constrained smooth optimization

Part II: Linear programming

Wen Huang

School of Mathematical Sciences
Xiamen University

Compiled on March 22, 2022

Linear Programming

Linear Programming

- Standard form
- Geometry
- Simplex method
- Interior point method

Linear program in the standard form:

$$\min_{x \in \mathbb{R}^n} c^T x$$
$$\text{s.t. } Ax = b, x \geq 0$$

Linear program:

$$\min c^T x, \text{ subject to } Ax \le b, Bx = f$$

Add slack variables z.

$$\min c^T x, \text{ subject to } Ax + z = b, Bx = f, z \ge 0$$

Define $x^+ = \max(x, 0) \ge 0$ and $x^- = \max(-x, 0) \ge 0$. $x = x^+ - x^-$.

$$\min \begin{bmatrix} c \\ -c \\ 0 \end{bmatrix}^T \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix}$$

$$\text{s.t. } \begin{bmatrix} A & -A & I \end{bmatrix} \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix} = b, \begin{bmatrix} B & -B & 0 \end{bmatrix} \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix} = f, \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix} \ge 0.$$

Definition 1

The set

$$H(\gamma, c) = \{x \in \mathbb{R}^n : c^T x = \gamma\}$$

is a hyperplane when $\gamma \in \mathbb{R}$ and $c \in \mathbb{R}^n$.

Example 2

$H(0, c)$ is the set of all vectors orthogonal to c. $H(0, c)$ is an $n - 1$ dimensional subspace of $\mathbb{R}^n$.

The family of hyperplances $\{H(\gamma, c) : \gamma \in \mathbb{R}\}$ given c can be generated easily.

$$H(\gamma, c) = \hat{x} + H(0, c)$$

where $\hat{x}$ is such that $c^T \hat{x} = \gamma$.
Typically choose $\hat{x} = \beta c$ where

$$\beta = \frac{\gamma}{c^T c}$$

Note: The hyperplanes in the family $H(\gamma, c)$ are parallel to one another.
Moving in the direction of c increases γ.

Definition 3

The set $H^- = \{x \mid c^T x \leq \gamma\}$ is the half space defined by $H(\gamma, c)$. (c always points out of H^-)

Definition 4

$x \in \mathcal{S}$, where $\mathcal{S}$ is convex, is an extreme point of $\mathcal{S}$ if there do not exist two **distinct** points $x_1, x_2 \in \mathcal{S}$ and $\alpha \in \mathbb{R}$ with $0 < \alpha < 1$ such that

$$x = \alpha x_1 + (1 - \alpha) x_2$$

Lemma 5

Hyperplanes and half spaces are convex sets.

Definition 6

The intersection of a finite number of closed half spaces is called a convex polytope. A nonempty bounded convex polytope is a convex polyhedron.

Theorem 7 (Geometric form)

If $\mathcal{K}$ is a convex polyhedron, then, a linear function, $c^T x$ on $\mathcal{K}$ achieves its minimum at an extreme point of $\mathcal{K}$.

Linear program ($c \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$):

$$\min_{x\in\mathbb{R}^n} c^T x$$
$$\text{s.t. } Ax = b, x \geq 0$$

- $c^T x$ defines a family of hyperplanes.
- $x \geq 0$ is convex
- Each row of A and the associated element of b define a hyperplane, i.e., let $e_i^T A = a_i^T$ and $e_i^T b = \beta_i$, then

$$Ax = b \Longrightarrow a_i^T x = \beta_i, i = 1, \ldots m \text{ are } m \text{ hyperplanes}$$

Definition 8

The set

$$\mathcal{F} = \{x : x \geq 0, \;\; Ax = b\}$$

is the set of feasible points for the linear program and is a convex set.

Definition 9

The set of basic solutions is defined as:

$$\mathfrak{B} = \left\{ x : Ax = b, \text{ and } \exists P \text{ with } Px = \begin{bmatrix} x_B^T & 0 \end{bmatrix}^T \right\}$$

where $x_B \in \mathbb{R}^m$ and P is a permutation matrix, and the matrix of first m columns of AP^T is nonsingular. The index of B is denoted by $\mathcal{B}$ and called a basis.

Definition 10

The set $\mathcal{S} = \mathfrak{B} \cap \mathcal{F}$ is called the set of basic feasible solutions, i.e., basic solutions with nonnegative components.

Theorem 11

$x \in \mathcal{S}$ if and only if $x \in \mathcal{F}$ is an extreme point.

See Luenberger and Ye p. 23.

Definition 10

The set $\mathcal{S} = \mathfrak{B} \cap \mathcal{F}$ is called the set of basic feasible solutions, i.e., basic solutions with nonnegative components.

Theorem 11

$x \in \mathcal{S}$ if and only if $x \in \mathcal{F}$ is an extreme point.

See Luenberger and Ye p. 23.

Simplex method:

only consider $x \in \mathcal{S}$ when solving the Linear program.

Let x be a basic feasible solution. Define

$$AP^T = \begin{bmatrix} B & N \end{bmatrix}, Px = \begin{bmatrix} x_B \\ x_N \end{bmatrix}, Pc = \begin{bmatrix} c_B \\ c_N \end{bmatrix}$$

with $B \in \mathbb{R}^{m \times m}$ full rank and $x_N = 0$.

Moving to another extreme point x^+:

- Choose a component of x_B to make 0 and a 0 component of x_N to make positive, i.e., swap a column in B with a column in N
- The update is based on considering the cost contribution of each element of x

$$\left.\begin{array}{r} Bx_B + Nx_N = b \\ Bx_B^+ + Nx_N^+ = b \end{array}\right\} \Longrightarrow x_B^+ + B^{-1}Nx_N^+ = x_B + B^{-1}Nx_N$$
$$\Longrightarrow x_B^+ = x_B - B^{-1}Nx_N^+. \tag{1}$$

$$\begin{aligned} f(x^+) =& c^T x^+ = c_B^T x_B^+ + c_N^T x_N^+ = c_B^T(x_B - B^{-1}Nx_N^+) + c_N^T x_N^+ \quad \text{(by (1))} \\ =& c_B^T x_B + (c_N^T - c_B^T B^{-1}N)x_N^+ \\ =& f(x) + r^T x_N^+ \end{aligned}$$

where $r^T = c_N^T - c_B^T B^{-1} N$ is called the reduced cost vector. If all components in r are nonnegative, then the cost cannot go down.

- Choose the component of x_N the corresponding to the most negative value in r
- Let i denote the index of the component, i.e., $x_N^+ = \alpha e_i$
- Choose α such that the smallest value of $x_B^+ = x_B - B^{-1}N(\alpha e_i)$ is zero
- Let j denote the index of the component, i.e., $e_j^T x_B^+ = 0$
- Swap j-th column of B and i-th column of N to produce B^+ and N^+

- Choose the component of x_N the corresponding to the most negative value in r
- Let i denote the index of the component, i.e., $x_N^+ = \alpha e_i$
- Choose α such that the smallest value of $x_B^+ = x_B - B^{-1}N(\alpha e_i)$ is zero
- Let j denote the index of the component, i.e., $e_j^T x_B^+ = 0$
- Swap j-th column of B and i-th column of N to produce B^+ and N^+

Note that such α may be zero.

(We assume that x and A are permuted appropriately at every step and that the indices are relative to the positions in x_B and x_N.)

1. Solve $Bx_B = b$ and $B^T z = c_B$
2. Evaluate $r = c_N - N^T z$
3. entering variable $i = argmin_{1 \leq j \leq n-m} \; e_j^T r$, for $e_j^T r < 0$
4. Solve $Bw = Ne_i$
5. leaving variable $k = argmin_{1 \leq j \leq m} \; \frac{e_j^T x_B^{(old)}}{e_j^T w}$ for $e_j^T w > 0$.
6. Exchange k-th column of B with i-th column of N to produce new B and N.

One situation that the algorithm is not well-defined

- B is nonsingular but B^+ is singular (excluded by nondegeneracy)

Two situations that function value does not decrease in a step of the simplex method

- $r \geq 0$ (implies finding a minimizer)
- $\alpha = 0$ (excluded by nondegeneracy)

One situation that the algorithm is not well-defined

- B is nonsingular but B^+ is singular (excluded by nondegeneracy)

Two situations that function value does not decrease in a step of the simplex method

- $r \geq 0$ (implies finding a minimizer)
- $\alpha = 0$ (excluded by nondegeneracy)

Definition 12

A basis $\mathcal{B}$ is said to be degenerate if $x_i = 0$ for some $i \in \mathcal{B}$, where x is the basic feasible solution corresponding to $\mathcal{B}$. A linear program is said to be degenerate if it has at least one degenerate basis.

Theorem 13

Given a Linear program that is bounded and not degenerate, $r \geq 0$ in the Simplex Method implies the current x is optimal.

See detailed proofs in [NW06, Theorem 13.4].

- Basic feasible solutions have **at most** m nonzeros.
- It is possible and not at all rare to have x_B contain **less** than m nonzeros, the so-called degenerate basic feasible solutions.
- In most cases this is not a problem and the simplex method moves away from these basic feasible solutions in a normal fashion.
- It is possible however, to construct problems where the degeneracy causes the entering variable to be such that the step returns to the same basic feasible solution, i.e., the method cycles.
- The Simplex folklore says this is rare. Many codes do not worry about it, but to be completely robust this cycling should be monitored. See [LY08, Exercises 15-17 and 31] and [NW06, Section 13.5].

Simplex Method

Remarks

- Usually takes $2m$ to $3m$ iterations
- Can visit every single extreme points theoretically, i.e., exponential complexity
- Polynomial complexity algorithm: interior point methods

Linear program:

$$\min_{x \in \mathbb{R}^n} c^T x$$
$$\text{s.t. } Ax = b, x \geq 0$$

where $A \in \mathbb{R}^{m \times n}$ is full row rank.

- Lagrangian function:

$$\mathcal{L} = c^T x - (Ax - b)^T \lambda - x^T s$$

  where $\lambda \in \mathbb{R}^m$ and $s \in \mathbb{R}^n$ are Lagrange multipliers.
- KKT conditions:

$$\begin{aligned} A^T \lambda + s &= c \\ Ax &= b \\ x_i s_i &= 0, i = 1, \ldots, n \\ (x, s) &\geq 0 \end{aligned}$$

KKT conditions:

$$F : \mathbb{R}^{2n+m} \to \mathbb{R}^{2n+m} : (x, \lambda, s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XS\mathbf{1} \end{bmatrix} = 0 \tag{2}$$

$$(x, s) \geq 0, \tag{3}$$

where $X = \mathrm{diag}(x_1, \dots, x_n)$ and $S = \mathrm{diag}(s_1, \dots, s_n)$.

- Find x, λ, and s that satisfy (2) and (3)?
- Newton method for (2)
- Without (3), many spurious solutions exist

Solving a system of a nonlinear equation

Newton's method

Let $r : \mathbb{R}^n \to \mathbb{R}^n : x \mapsto r(x)$ be a vector function. Solve

$$r(x) = 0.$$

Algorithm 1 Newton's method for solving a nonlinear system

Input: An initial iterate x_0;
Output: Initial iterate (x_0, λ_0, s_0);

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: Calculate a solution p_k to the Newton equations:

$$J(x_k)p_k = -r(x_k);$$

where $J(x_k)$ is the Jacobi matrix of r at x_k;
3: $x_{k+1} \leftarrow x_k + p_k$;
4: **end for**

Local superlinear convergence rate

Newton method for nonlinear equation:

$$J_F(x, \lambda, s)[\Delta x; \Delta\lambda; \Delta s] = -F(x, \lambda, s)$$
$$\Longrightarrow \begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r^c \\ -r^b \\ -XS\mathbf{1} \end{bmatrix}$$

where $r^c = A^T\lambda + s - c$ and $r^b = Ax - b$.

- Choose x_0, λ_0, s_0 such that $(x_0, s_0) \geq 0$
- For any k,

$$(x_{k+1}, \lambda_{k+1}, s_{k+1}) = (x_k, \lambda_k, s_k) + \alpha_k(\Delta x_k, \Delta\lambda_k, \Delta s_k)$$

  where α_k is sufficient small such that $(x_{k+1}, \lambda_{k+1}, s_{k+1}) \geq 0$
- Potential problem: α small $\Longrightarrow$ slow
- Potential problem: coefficient matrix near singular $\Longrightarrow$ direction unreliable

- x_k and s_k are strictly positive for all k
- KKT conditions $\Longrightarrow$ a variant:

$$\left.\begin{array}{c} F(x, \lambda, s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XS\mathbf{1} \end{bmatrix} = 0 \\ (x, s) \geq 0, \end{array}\right\} \Longrightarrow \left\{\begin{array}{c} \tilde{F}(x, \lambda, s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XS\mathbf{1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \tau\mathbf{1} \end{bmatrix} \\ (x, s) > 0, \end{array}\right. \tag{4}$$

where $\tau \geq 0$.

- We show in the next slide that the solution to (4) is unique

Linear program vs a log-barrier formulation:

$$\begin{array}{c} \min_{x\in\mathbb{R}^n} c^T x \\ \text{s.t. } Ax = b, x \geq 0 \end{array} \quad \text{vs} \quad \begin{array}{c} \min_{x\in\mathbb{R}^n} c^T x - \tau \sum_{i=1}^n \ln x_i \\ \text{s.t. } Ax = b. \end{array}$$

- The log-barrier formulation: strictly convex objective function
- Unique solution
- Lagrangian function of the log-barrier formulation:

$$\mathcal{L} = c^T x - \tau \sum_{i=1}^{n} \ln x_i - (Ax - b)^T \lambda$$

- KKT conditions:

$$\begin{array}{c} \begin{bmatrix} A^T\lambda + \tau./x - c \\ Ax - b \end{bmatrix} = 0 \\ x > 0 \\ \text{Unique solution } x, \lambda \end{array} \quad \xRightarrow{s:=\tau./x} \quad \begin{array}{c} \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XS\mathbf{1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \tau\mathbf{1} \end{bmatrix} \\ (x, s) > 0 \\ \text{Unique solution } x, \lambda, s \end{array}$$

- The central path: $\mathcal{C} = \{(x^{(\tau)}, \lambda^{(\tau)}, s^{(\tau)}) : \tau > 0\}$
- $(x^{(\tau)}, \lambda^{(\tau)}, s^{(\tau)}) \to$ a solution as $\tau \to 0$
- Approximately solve:

$$\left.\begin{array}{c} F(x,\lambda,s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XS\mathbf{1} \end{bmatrix} = 0 \\ (x,s) \geq 0, \end{array}\right\}$$

$$\Longrightarrow \left\{\begin{array}{c} \tilde{F}(x,\lambda,s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XS\mathbf{1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \tau\mathbf{1} \end{bmatrix} \\ (x,s) > 0, \end{array}\right.$$

e.g., a step of Newton method

- Reduce τ appropriately in every iteration

$$\begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XS\mathbf{1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sigma\mu\mathbf{1} \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r^c \\ -r^b \\ -XS\mathbf{1} + \sigma\mu\mathbf{1} \end{bmatrix}$$

- $\mu = x^T s/n$ is duality measure
- $\sigma \in [0, 1]$ is the centering parameter
- σ is typically chosen to be in $(0, 1)$

- The primal-dual strictly feasible set

$$\mathcal{F}^0 = \{(x, \lambda, s) : Ax = b, A^T\lambda + s = c, x, s > 0\}$$

- A neighborhood of $\mathcal{C}$:

$$\mathcal{N}_{-\infty}(\gamma) = \{(x, \lambda, s) \in \mathcal{F}^0 : x_i s_i \geq \gamma\mu, i = 1, \ldots, n\}$$

Algorithm 2 Long-step path-following interior point method

Input: Given γ, $\sigma_{\min}$, $\sigma_{\max}$ with $\gamma \in (0,1)$, $0 < \sigma_{\min} \leq \sigma_{\max} < 1$ and $(x_0, \lambda_0, s_0) \in \mathcal{N}_{-\infty}(\gamma)$;

1: **for** $i = 0, 1, 2, \ldots$ **do**

2: Choose $\sigma_k \in [\sigma_{\min}, \sigma_{\max}]$ and solve

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S_k & 0 & X_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta \lambda_k \\ \Delta s_k \end{bmatrix} = \begin{bmatrix} -r_k^c \\ -r_k^b \\ -X_k S_k \mathbf{1} + \sigma_k \mu_k \mathbf{1} \end{bmatrix}$$

where $\mu_k = x_k^T s_k / n$

3: Set

$$(x_{k+1}, \lambda_{k+1}, s_{k+1}) \leftarrow (x_k, \lambda_k, s_k) + \alpha_k (\Delta x_k, \Delta \lambda_k, \Delta s_k)$$

where α_k as the largest value of α in $[0,1]$ such that $(x_{k+1}, \lambda_{k+1}, s_{k+1}) \in \mathcal{N}_{-\infty}(\gamma)$

4: **end for**

Theorem 14

Let $\{\mu_k\}$ *denote the sequence of duality measures of Algorithm 2. Then given* $\epsilon \in (0, 1)$, *there is an index* K *with* $K = O(n \log(1/\epsilon))$ *such that*

$$\mu_k \leq \epsilon \mu_0, \qquad \text{for all } k \geq K.$$

See detailed proofs in [NW06, Theorem 14.4].

Algorithm 3 Initial iterate

Input: $A \in \mathbb{R}^{m\times n}$ full row rank, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$;
Output: Initial iterate (x_0, λ_0, s_0);
1: $\tilde{x} \leftarrow A^T(AA^T)^{-1}b$, $\tilde{\lambda} \leftarrow (AA^T)^{-1}Ac$, $\tilde{s} \leftarrow c - A^T\tilde{\lambda}$;
2: $\delta_x \leftarrow \max\left(-\frac{3}{2}\min_i \tilde{x}_i, 0\right)$, $\delta_s \leftarrow \max\left(-\frac{3}{2}\min_i \tilde{s}_i, 0\right)$;
3: $\hat{x} = \tilde{x} + \delta_x\mathbf{1}$, $\hat{s} = \tilde{s} + \delta_s\mathbf{1}$;
4: $\hat{\delta}_x \leftarrow \frac{\hat{x}^T\hat{s}}{2\mathbf{1}^T\hat{s}}$, $\hat{\delta}_s \leftarrow \frac{\hat{x}^T\hat{s}}{2\mathbf{1}^T\hat{x}}$;
5: $x_0 \leftarrow \hat{x} + \hat{\delta}_x\mathbf{1}$, $\lambda_0 \leftarrow \tilde{\lambda}$, $s_0 \leftarrow \hat{s} + \hat{\delta}_s\mathbf{1}$;

- $\tilde{x}$: $\min_x x^Tx$, s.t. $Ax = b$;
- $(\tilde{\lambda}, \tilde{s})$: $\min_{\lambda,s} s^Ts$, s.t. $A^T\lambda + s = c$;
- Adjust $\tilde{x}$ and $\tilde{s}$: $\hat{x}$ and $\hat{s}$ are positive
- Ensure x and s not too close to zero and not too dissimilar: x_0 and s_0

Algorithm 4 Predictor-corrector algorithm Part I

Input: Calculas (x_0, λ_0, s_0) by Algorithm 3; $\{\eta_k \in [0.9, 1)\}$ $\forall k$ and $\eta_k \to 1$;

1: **for** $k = 0, 1, 2, \ldots$ **do**

2: Solve

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S_k & 0 & X_k \end{bmatrix} \begin{bmatrix} \Delta x_k^{\mathrm{aff}} \\ \Delta \lambda_k^{\mathrm{aff}} \\ \Delta s_k^{\mathrm{aff}} \end{bmatrix} = \begin{bmatrix} -r_k^c \\ -r_k^b \\ -X_k S_k \mathbf{1} \end{bmatrix}$$

3: Compute

$$\alpha_{\mathrm{aff}}^{\mathrm{pri}} \leftarrow \min\left(1, \min_{i:\left(\Delta x_k^{\mathrm{aff}}\right)_i < 0} -\frac{(x_k)_i}{\left(\Delta x_k^{\mathrm{aff}}\right)_i}\right)$$

$$\alpha_{\mathrm{aff}}^{\mathrm{dual}} \leftarrow \min\left(1, \min_{i:\left(\Delta s_k^{\mathrm{aff}}\right)_i < 0} -\frac{(s_k)_i}{\left(\Delta s_k^{\mathrm{aff}}\right)_i}\right)$$

$$\mu_{\mathrm{aff}} \leftarrow \left(x_k + \alpha_{\mathrm{aff}}^{\mathrm{pri}} \Delta x_k^{\mathrm{aff}}\right)^T \left(s_k + \alpha_{\mathrm{aff}}^{\mathrm{dual}} \Delta s_k^{\mathrm{aff}}\right) / n$$

4: Continue on the next page

Algorithm 5 Predictor-corrector algorithm Part II

1: Compute $\mu_k = x_k^T s_k / n$ and $\sigma_k = (\mu_{\rm aff}/\mu_k)^3$

2: Solve

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S_k & 0 & X_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta\lambda_k \\ \Delta s_k \end{bmatrix} = \begin{bmatrix} -r_k^c \\ -r_k^b \\ -X_k S_k \mathbf{1} - \Delta X_k^{\rm aff} \Delta S_k^{\rm aff} \mathbf{1} + \sigma_k \mu_k \mathbf{1} \end{bmatrix}$$

3: Compute

$$\alpha_{k,\max}^{\rm pri} \leftarrow \min_{i:(\Delta x_k)_i<0} -\frac{(x_k)_i}{(\Delta x_k)_i} \quad \text{and} \quad \alpha_k^{\rm pri} \leftarrow \min(1, \eta_k \alpha_{k,\max}^{\rm pri})$$

$$\alpha_{k,\max}^{\rm dual} \leftarrow \min_{i:(\Delta s_k)_i<0} -\frac{(s_k)_i}{(\Delta s_k)_i} \quad \text{and} \quad \alpha_k^{\rm dual} \leftarrow \min(1, \eta_k \alpha_{k,\max}^{\rm dual})$$

4: Set

$$x_{k+1} \leftarrow x_k + \alpha_k^{\rm pri} \Delta x_k,$$

$$(\lambda_{k+1}, s_{k+1}) \leftarrow (\lambda_k, s_k) + \alpha_k^{\rm dual} (\Delta\lambda_k, \Delta s_k)$$

5: **end for**

- Dominated computational cost
- Solve for v:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r^c \\ -r^b \\ -r^{xs} \end{bmatrix}$$

- Equivalent formulation:

$$\begin{aligned} A(XS^{-1})A^T \Delta\lambda &= -\, r^b - AXS^{-1} r^c + AS^{-1} r^{xs} \\ \Delta s &= -\, r^c - A^T \Delta\lambda \\ \Delta x &= -\, S^{-1} r^{xs} - XS^{-1} \Delta s \end{aligned}$$

- Cholesky decomposition for $A(XS^{-1})A^T$

References I

D. G. Luenberger and Y. Ye.
Linear and nonlinear programming.
Springer, third edition, 2008.

J. Nocedal and S. J. Wright.
Numerical Optimization.
Springer, second edition, 2006.