Constrained Nonlinear Optimization Algorithms

Andreas Wächter

Department of Industrial Engineering and Management Sciences
Northwestern University
waechter@iems.northwestern.edu

Institute for Mathematics and its Applications
University of Minnesota

August 4, 2016

Constrained Nonlinear Optimization Problems

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & f(x) \\ \text{s.t. } & c_E(x) = 0 \\ & c_I(x) \le 0 \end{aligned} \qquad \text{(NLP)}$$

$$\begin{aligned} f &: \mathbb{R}^n \longrightarrow \mathbb{R} \\ c_E &: \mathbb{R}^n \longrightarrow \mathbb{R}^{m_E} \\ c_I &: \mathbb{R}^n \longrightarrow \mathbb{R}^{m_I} \end{aligned}$$

- We assume that all functions are twice continuously differentiable.

Constrained Nonlinear Optimization Problems

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & f(x) \\ \text{s.t.} \; & c_E(x) = 0 \\ & c_I(x) \le 0 \end{aligned} \qquad \text{(NLP)}$$

$$\begin{aligned} f &: \mathbb{R}^n \longrightarrow \mathbb{R} \\ c_E &: \mathbb{R}^n \longrightarrow \mathbb{R}^{m_E} \\ c_I &: \mathbb{R}^n \longrightarrow \mathbb{R}^{m_I} \end{aligned}$$

- We assume that all functions are twice continuously differentiable.
- No is convexity required.

Constrained Nonlinear Optimization Problems

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & f(x) \\ \text{s.t.} \; & c_E(x) = 0 \\ & c_I(x) \le 0 \end{aligned} \qquad \text{(NLP)}$$

$$\begin{aligned} f &: \mathbb{R}^n \longrightarrow \mathbb{R} \\ c_E &: \mathbb{R}^n \longrightarrow \mathbb{R}^{m_E} \\ c_I &: \mathbb{R}^n \longrightarrow \mathbb{R}^{m_I} \end{aligned}$$

- We assume that all functions are twice continuously differentiable.
- No is convexity required.
- Most algorithms for NLP have
  - theoretical convergence guarantee only to stationary points;

Constrained Nonlinear Optimization Problems

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c_E(x) = 0 \\ & c_I(x) \leq 0 \end{aligned} \qquad \text{(NLP)}$$

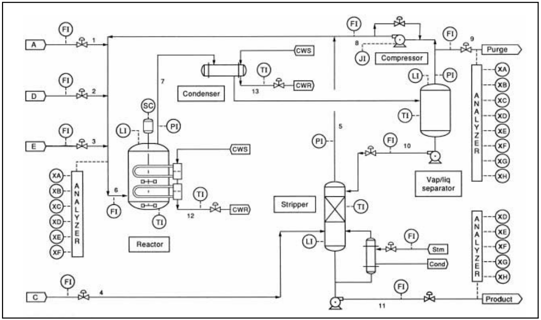
$$\begin{aligned} f &: \mathbb{R}^n \longrightarrow \mathbb{R} \\ c_E &: \mathbb{R}^n \longrightarrow \mathbb{R}^{m_E} \\ c_I &: \mathbb{R}^n \longrightarrow \mathbb{R}^{m_I} \end{aligned}$$

- We assume that all functions are twice continuously differentiable.
- No is convexity required.
- Most algorithms for NLP have
  - theoretical convergence guarantee only to stationary points;
  - ingredients that steer towards local minimizers.

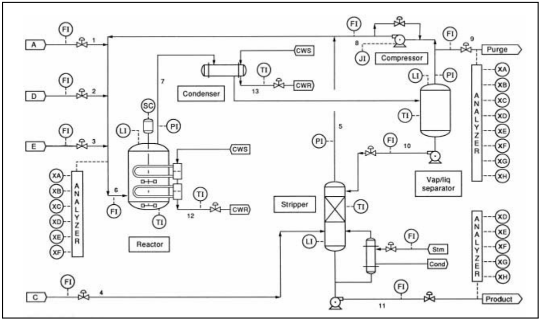
Table of Contents

Table of Contents

Design and Operation of Chemical Plant

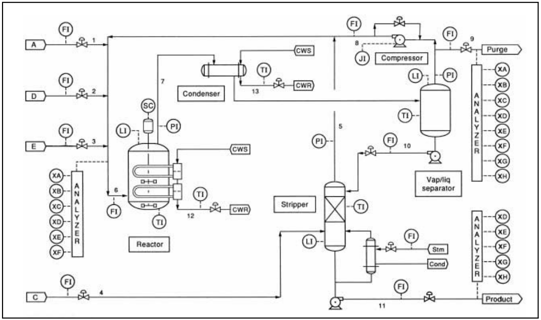


Design and Operation of Chemical Plant



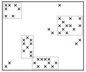
- Minimize "Costs − Profit"
- Variables: Physical quantities
- Constraints: physical relationships
  (conservation laws; themodyn. rel.)
- Limits (physical and operational)
- $< 10^5$ variables; few degrees of freedom

Design and Operation of Chemical Plant



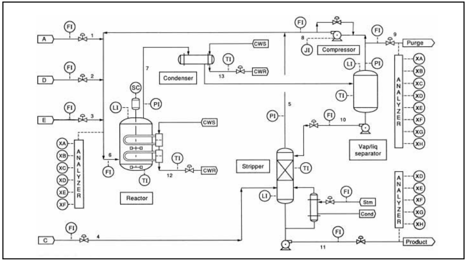
- Minimize "Costs − Profit"
- Variables: Physical quantities
- Constraints: physical relationships
  (conservation laws; themodyn. rel.)
- Limits (physical and operational)
- $< 10^5$ variables; few degrees of freedom

Constraint Jacobian

$\nabla c(x)^T =$ 

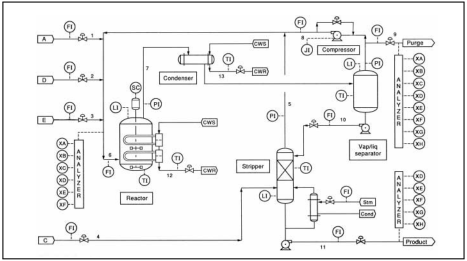
is sparse and structured

Design Under Uncertainty

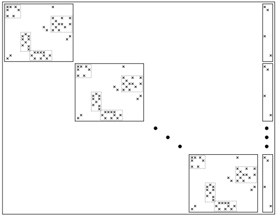


- Scenario parameters: $F_{in}^{l}, x_{in}^{l}, T_{env}^{l}, p_{env}^{l}, \ldots$ (given) (defining scenarios $l = 1, \ldots, L$)
- Design variables: $V_{react}, D_{dist}, h_{tank}, \ldots$
- Control variables: $Q_{heat}^{l}, r_{refl}^{l}, v_{valve}^{l}, \ldots$
- State variables: $x_{str}^{l}, F_{str}^{l}, L_{tank}^{l}, T^{l}, p^{l}, \ldots$

Design Under Uncertainty

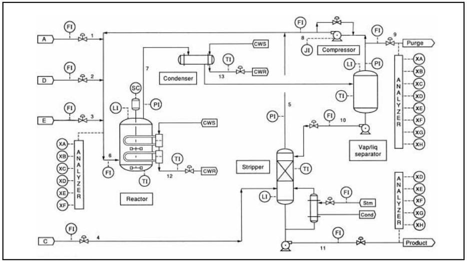


$$\nabla c(x)^T =$$



- Scenario parameters: $F^l_{in}, x^l_{in}, T^l_{env}, p^l_{env}, \ldots$ (given) (defining scenarios $l = 1, \ldots, L$)
- Design variables: $V_{react}, D_{dist}, h_{tank}, \ldots$
- Control variables: $Q^l_{heat}, r^l_{refl}, v^l_{valve}, \ldots$
- State variables: $x^l_{str}, F^l_{str}, L^l_{tank}, T^l, p^l, \ldots$

Optimal Control / Dynamic Optimization

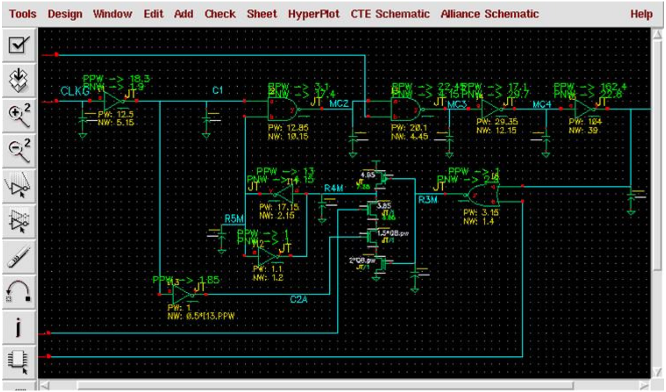


$$
\begin{aligned}
\min_{z,y,u,p} \quad & f(z(t_f), y(t_f), u(t_f), p) \\
\text{s.t.} \quad & F(\dot{z}(t), z(t), y(t), u(t), p) = 0 \\
& G(z(t), y(t), u(t), p) = 0 \\
& z(0) = z_{init} \\
& \underline{\text{bound constraints}}
\end{aligned}
$$

$u : [0, t_f] \rightarrow \mathbb{R}^{n_u}$	control variables
$z : [0, t_f] \rightarrow \mathbb{R}^{n_z}$	differentiable state variables
$y : [0, t_f] \rightarrow \mathbb{R}^{n_y}$	algebraic state variables
$p \in \mathbb{R}^{n_p}$	time-independent parameters
z_{init}	initial conditions
t_f	final time

- Large-scale NLPs arise from discretization.

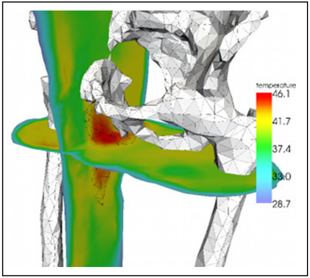
Circuit Tuning



- Model consists of network of gates.
- Gate delays computed by simulation (expensive, noisy).
- Model has many variables (up to $1,000,000$).
- Implemented in IBM's circuit tuning tool EinsTuner.

Hyperthermia Treatment Planning

$$
\begin{aligned}
\min_{T(x),u} \quad & \frac{1}{2}\int_{\Omega} (T(x) - T_{\text{target}}(x))^2 \, dx \\
\text{s.t.} \quad & -\Delta T(x) - w(T(x) - T_{\text{blood}}) = u^* M(x) u && \text{in } \Omega \\
& \nabla T(x) \cdot n = T_{\text{exterior}} - T(x) && \text{on } \partial\Omega \\
& T(x) \le T_{\text{max}} && \text{in } \Omega \setminus \Omega_{\text{Tumor}}
\end{aligned}
$$



- Heat tumors with microwaves (support chemo- and radio-therapy).
- Model is a PDE with
  - controls: Application u of microwave antennas.
  - states: Temperature $T(x)$ defined over domain Ω.
- Finite-dimensional problem obtained by discretization.
  - e.g., finite differences, finite elements
- Resulting NLP is usually very large.

Table of Contents

Quadratic Programming

$$
\begin{aligned}
\min_{x\in\mathbb{R}^n} \quad & \frac{1}{2}x^T Q x + g^T x \\
\text{s.t.} \quad & A_E x + b_E = 0 \\
& A_I x + b_I \le 0
\end{aligned}
\qquad \text{(QP)}
$$

$$
\begin{aligned}
& Q \in \mathbb{R}^{n\times n} \text{ symmetric} \\
& A_E \in \mathbb{R}^{m^E\times n} \qquad b_E \in \mathbb{R}^{m_E} \\
& A_I \in \mathbb{R}^{m^I\times n} \qquad b_I \in \mathbb{R}^{m_I}
\end{aligned}
$$

Quadratic Programming

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \tfrac{1}{2}x^T Q x + g^T x \\ \text{s.t.} \quad & A_E x + b_E = 0 \\ & A_I x + b_I \le 0 \end{aligned} \qquad \text{(QP)}$$

$$Q \in \mathbb{R}^{n\times n} \text{ symmetric}$$
$$A_E \in \mathbb{R}^{m^E\times n} \qquad b_E \in \mathbb{R}^{m_E}$$
$$A_I \in \mathbb{R}^{m^I\times n} \qquad b_I \in \mathbb{R}^{m_I}$$

- Many applications (e.g., portfolio optimization, optimal control).

Quadratic Programming

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \tfrac{1}{2}x^TQx + g^Tx \\ \text{s.t.} \quad & A_Ex + b_E = 0 \\ & A_Ix + b_I \le 0 \end{aligned} \qquad \text{(QP)}$$

$$\begin{array}{cc} \multicolumn{2}{c}{Q \in \mathbb{R}^{n\times n} \text{ symmetric}} \\ A_E \in \mathbb{R}^{m^E\times n} & b_E \in \mathbb{R}^{m_E} \\ A_I \in \mathbb{R}^{m^I\times n} & b_I \in \mathbb{R}^{m_I} \end{array}$$

- Many applications (e.g., portfolio optimization, optimal control).
- Important building block for methods for general NLP.

Quadratic Programming

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \;& \tfrac{1}{2} x^T Q x + g^T x \\ \text{s.t. } & A_E x + b_E = 0 \\ & A_I x + b_I \leq 0 \end{aligned} \qquad \text{(QP)}$$

$$\begin{aligned} & Q \in \mathbb{R}^{n \times n} \text{ symmetric} \\ & A_E \in \mathbb{R}^{m^E \times n} \quad b_E \in \mathbb{R}^{m_E} \\ & A_I \in \mathbb{R}^{m^I \times n} \quad b_I \in \mathbb{R}^{m_I} \end{aligned}$$

- Many applications (e.g., portfolio optimization, optimal control).
- Important building block for methods for general NLP.
- Algorithms:
  - Active-set methods
  - Interior-point methods

Quadratic Programming

$$\min_{x\in\mathbb{R}^n} \ \frac{1}{2}x^T Q x + g^T x$$
$$\text{s.t. } A_E x + b_E = 0$$
$$A_I x + b_I \le 0$$

(QP)

$Q \in \mathbb{R}^{n\times n}$ symmetric

$A_E \in \mathbb{R}^{m^E\times n}$ $\quad b_E \in \mathbb{R}^{m_E}$

$A_I \in \mathbb{R}^{m^I\times n}$ $\quad b_I \in \mathbb{R}^{m_I}$

- Many applications (e.g., portfolio optimization, optimal control).
- Important building block for methods for general NLP.
- Algorithms:
  - Active-set methods
  - Interior-point methods
- Let's first consider equality-constrained case.
- Assume: all rows of A_E are linearly independent.

Equality-Constrained QP

$$\boxed{\begin{aligned} \min_{x \in \mathbb{R}^n} \ & \tfrac{1}{2} x^T Q x + g^T x \\ \text{s.t.} \ & Ax + b = 0 \end{aligned}} \qquad \text{(EQP)}$$

Equality-Constrained QP

$$\boxed{\begin{aligned} \min_{x \in \mathbb{R}^n} \ & \tfrac{1}{2} x^T Q x + g^T x \\ \text{s.t. } & Ax + b = 0 \end{aligned}} \qquad \text{(EQP)}$$

First-order optimality conditions:

$$\begin{aligned} Qx + g + A^T \lambda &= 0 \\ Ax + b &= 0 \end{aligned}$$

Equality-Constrained QP

$$\boxed{\begin{aligned} \min_{x\in\mathbb{R}^n} \ & \tfrac{1}{2}x^TQx + g^Tx \\ \text{s.t. } & Ax + b = 0 \end{aligned}} \qquad \text{(EQP)}$$

First-order optimality conditions:

$$\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}$$

Find stationary point (x^*, λ^*) by solving the linear system

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = -\begin{pmatrix} g \\ b \end{pmatrix}.$$

KKT System of QP

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = - \begin{pmatrix} g \\ b \end{pmatrix}$$

- When is (x^*, λ^*) indeed a solution of (EQP)?

KKT System of QP

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = - \begin{pmatrix} g \\ b \end{pmatrix}$$

- When is (x^*, λ^*) indeed a solution of (EQP)?
- Recall the second-order optimality conditions:
  - Let the columns of $Z \in \mathbb{R}^{n \times (n-m)}$ be a basis of the null-space of A, so $AZ = 0$ ("null-space matrix").
  - Then x^* is a strict local minimizer of (EQP) if $Z^T Q Z \succ 0$.

KKT System of QP

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = - \begin{pmatrix} g \\ b \end{pmatrix}$$

- When is (x^*, λ^*) indeed a solution of (EQP)?
- Recall the second-order optimality conditions:
  - Let the columns of $Z \in \mathbb{R}^{n \times (n-m)}$ be a basis of the null-space of A, so $AZ = 0$ ("null-space matrix").
  - Then x^* is a strict local minimizer of (EQP) if $Z^T QZ \succ 0$.
- On the other hand:
  - If $Z^T QZ$ has negative eigenvalue, then (EQP) is unbounded below.

KKT System of QP

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = - \begin{pmatrix} g \\ b \end{pmatrix}$$

- When is (x^*, λ^*) indeed a solution of (EQP)?
- Recall the second-order optimality conditions:
  - Let the columns of $Z \in \mathbb{R}^{n \times (n-m)}$ be a basis of the null-space of A, so $AZ = 0$ ("null-space matrix").
  - Then x^* is a strict local minimizer of (EQP) if $Z^T QZ \succ 0$.
- On the other hand:
  - If $Z^T QZ$ has negative eigenvalue, then (EQP) is unbounded below.
- There are different ways to solve the KKT system
  - Best choice depends on particular problem

Direct Solution of the KKT System

$$\underbrace{\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix}}_{=:K} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = - \begin{pmatrix} g \\ b \end{pmatrix}$$

- How can we verify that x^* is local minimizer without computing Z?

Direct Solution of the KKT System

$$\underbrace{\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix}}_{=:K} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = - \begin{pmatrix} g \\ b \end{pmatrix}$$

- How can we verify that x^* is local minimizer without computing Z?

Definition

Let n_+, n_-, n_0 be the number of positive, negative, and zero eigenvalues of a matrix M. Then $\text{In}(M) = (n_+, n_-, n_0)$ is the inertia of M.

Direct Solution of the KKT System

$$\underbrace{\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix}}_{=:K} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = - \begin{pmatrix} g \\ b \end{pmatrix}$$

- How can we verify that x^* is local minimizer without computing Z?

Definition

Let n_+, n_-, n_0 be the number of positive, negative, and zero eigenvalues of a matrix M. Then $\mathrm{In}(M) = (n_+, n_-, n_0)$ is the inertia of M.

Theorem

Suppose that A has full rank. Then: $\mathrm{In}(K) = \mathrm{In}(Z^T QZ) + (m, m, 0)$.

Direct Solution of the KKT System

$$\underbrace{\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix}}_{=:K} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = - \begin{pmatrix} g \\ b \end{pmatrix}$$

- How can we verify that x^* is local minimizer without computing Z?

Definition

Let n_+, n_-, n_0 be the number of positive, negative, and zero eigenvalues of a matrix M. Then $\text{In}(M) = (n_+, n_-, n_0)$ is the inertia of M.

Theorem

Suppose that A has full rank. Then: $\text{In}(K) = \text{In}(Z^T Q Z) + (m, m, 0)$.

Corollary

If $\text{In}(K) = (n, m, 0)$, *then* x^* *is the unique global minimizer.*

Computing the Inertia

$$\underbrace{\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix}}_{=:K} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = - \begin{pmatrix} g \\ b \end{pmatrix}$$

- Symmetric indefinite factorization $PKP^T = LBL^T$
  - P: permutation matrix
  - L: unit lower triangular matrix
  - B: block diagonal matrix with 1×1 and 2×2 diagonal blocks

Computing the Inertia

$$\underbrace{\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix}}_{=:K} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = - \begin{pmatrix} g \\ b \end{pmatrix}$$

- Symmetric indefinite factorization $PKP^T = LBL^T$
  - P: permutation matrix
  - L: unit lower triangular matrix
  - B: block diagonal matrix with 1×1 and 2×2 diagonal blocks
- Can be computed efficiently, exploits sparsity.

Computing the Inertia

$$\underbrace{\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix}}_{=:K} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = -\begin{pmatrix} g \\ b \end{pmatrix}$$

- Symmetric indefinite factorization $PKP^T = LBL^T$
  - P: permutation matrix
  - L: unit lower triangular matrix
  - B: block diagonal matrix with 1×1 and 2×2 diagonal blocks
- Can be computed efficiently, exploits sparsity.
- Obtain inertia simply from counting eigenvalues of the blocks in B.

Computing the Inertia

$$\underbrace{\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix}}_{=:K} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = - \begin{pmatrix} g \\ b \end{pmatrix}$$

- Symmetric indefinite factorization $PKP^T = LBL^T$
  - P: permutation matrix
  - L: unit lower triangular matrix
  - B: block diagonal matrix with 1×1 and 2×2 diagonal blocks
- Can be computed efficiently, exploits sparsity.
- Obtain inertia simply from counting eigenvalues of the blocks in B.
- Used also to solve the linear system.

Computing the Inertia

$$\underbrace{\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix}}_{=:K} \begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = -\begin{pmatrix} g \\ b \end{pmatrix}$$

- Symmetric indefinite factorization $PKP^T = LBL^T$
  - P: permutation matrix
  - L: unit lower triangular matrix
  - B: block diagonal matrix with 1×1 and 2×2 diagonal blocks
- Can be computed efficiently, exploits sparsity.
- Obtain inertia simply from counting eigenvalues of the blocks in B.
- Used also to solve the linear system.
- Will be important later when we need to “convexify” QPs
  ($Q \leftarrow Q + \gamma I$).

Schur-Complement Method

$$\boxed{\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}}$$

- Assume, Q is positive definite. Then $AQ^{-1}A^T$ is nonsingular.

Schur-Complement Method

$$\boxed{\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}}$$

- Assume, Q is positive definite. Then $AQ^{-1}A^T$ is nonsingular.
- Pre-multiply first equation by AQ^{-1}.

Schur-Complement Method

$$\boxed{\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}}$$

- Assume, Q is positive definite. Then $AQ^{-1}A^T$ is nonsingular.
- Pre-multiply first equation by AQ^{-1}.
- Then solve

$$[AQ^{-1}A^T]\lambda^* = b - AQ^{-1}g$$

Schur-Complement Method

$$\boxed{\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}}$$

- Assume, Q is positive definite. Then $AQ^{-1}A^T$ is nonsingular.
- Pre-multiply first equation by AQ^{-1}.
- Then solve

$$\begin{aligned} [AQ^{-1}A^T]\lambda^* &= b - AQ^{-1}g \\ Qx &= -g - A^T\lambda^* \end{aligned}$$

Schur-Complement Method

$$\boxed{\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}}$$

- Assume, Q is positive definite. Then $AQ^{-1}A^T$ is nonsingular.
- Pre-multiply first equation by AQ^{-1}.
- Then solve

$$\begin{aligned} [AQ^{-1}A^T]\lambda^* &= b - AQ^{-1}g \\ Qx &= -g - A^T\lambda^* \end{aligned}$$

- Requirements:
  - Solutions with Q can be done efficiently

Schur-Complement Method

$$\boxed{\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}}$$

- Assume, Q is positive definite. Then $AQ^{-1}A^T$ is nonsingular.
- Pre-multiply first equation by AQ^{-1}.
- Then solve

$$\begin{aligned} [AQ^{-1}A^T]\lambda^* &= b - AQ^{-1}g \\ Qx &= -g - A^T\lambda^* \end{aligned}$$

- Requirements:
  - Solutions with Q can be done efficiently
  - Need to compute $[AQ^{-1}A^T]$ and solve linear system with it

Schur-Complement Method

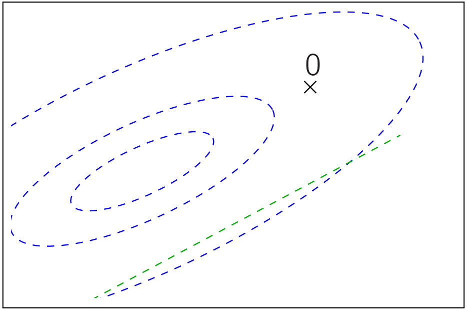
$$\boxed{\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}}$$

- Assume, Q is positive definite. Then $AQ^{-1}A^T$ is nonsingular.
- Pre-multiply first equation by AQ^{-1}.
- Then solve

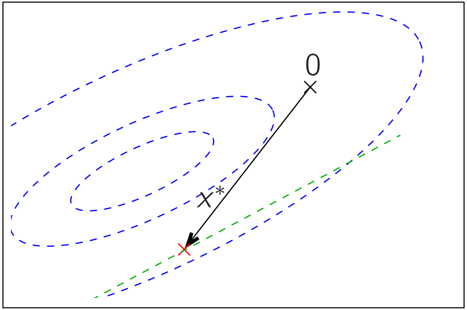
$$\begin{aligned} [AQ^{-1}A^T]\lambda^* &= b - AQ^{-1}g \\ Qx &= -g - A^T\lambda^* \end{aligned}$$

- Requirements:
  - Solutions with Q can be done efficiently
  - Need to compute $[AQ^{-1}A^T]$ and solve linear system with it
  - Works best if m is small

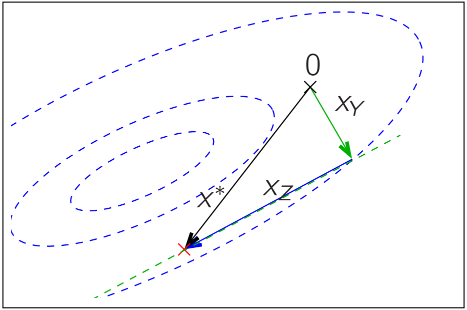
Step Decomposition



Step Decomposition

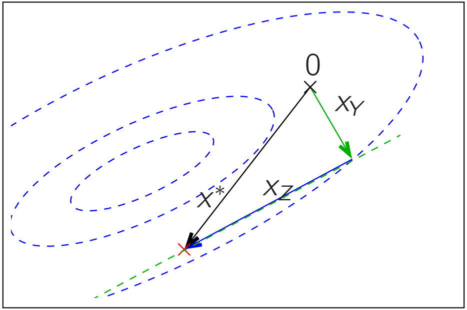


Step Decomposition



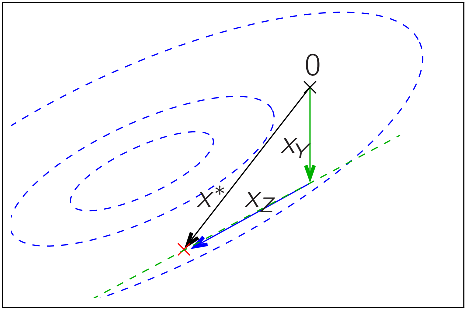
- Decompose $x^* = x_Y + x_Z$ into two steps:
  - "range-space step" x_Y: step into constraints
  - "null-space step" x_Z: optimize within null space

Step Decomposition



- Decompose $x^* = x_Y + x_Z$ into two steps:
  - "range-space step" x_Y: step into constraints
  - "null-space step" x_Z: optimize within null space
- $x_Y = Yp_Y$ and $x_Z = Zp_Z$
  - where $[Y\ Z]$ is basis of $\mathbb{R}^n$ and Z is null space matrix for A.

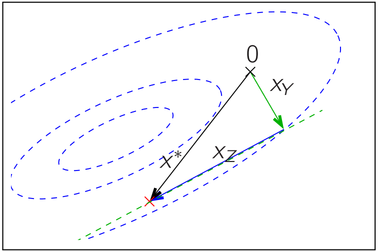
Step Decomposition



- Decompose $x^* = x_Y + x_Z$ into two steps:
  - "range-space step" x_Y: step into constraints
  - "null-space step" x_Z: optimize within null space
- $x_Y = Yp_Y$ and $x_Z = Zp_Z$
  - where $[Y\ Z]$ is basis of $\mathbb{R}^n$ and Z is null space matrix for A.
- Decomposition depends on choice of Y and Z

Step Decomposition

$$\boxed{\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}}$$

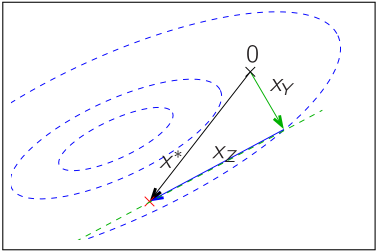


- $x_Y = Yp_Y$ is a step into the constraints:

$$0 = Ax + b = AYp_Y + AZp_Z + b$$

Step Decomposition

$$\boxed{\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}}$$

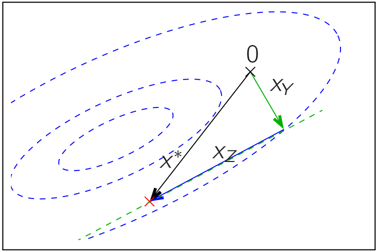


- $x_Y = Yp_Y$ is a step into the constraints:

$$0 = Ax + b = AYp_Y + AZp_Z + b \quad \Longrightarrow \quad p_Y = -[AY]^{-1}b$$

Step Decomposition

$$\boxed{\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}}$$



- $x_Y = Yp_Y$ is a step into the constraints:

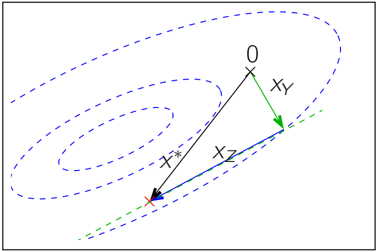
$$0 = Ax + b = AYp_Y + AZp_Z + b \quad \Longrightarrow \quad p_Y = -[AY]^{-1}b$$

- $x_Z = Zp_Z$ optimizes in the null space

$$p_Z = -[Z^TQZ]^{-1}Z^T(g + QYp_Y)$$

Step Decomposition

$$\boxed{\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}}$$



- $x_Y = Yp_Y$ is a step into the constraints:

$$0 = Ax + b = AYp_Y + AZp_Z + b \quad \Longrightarrow \quad p_Y = -[AY]^{-1}b$$

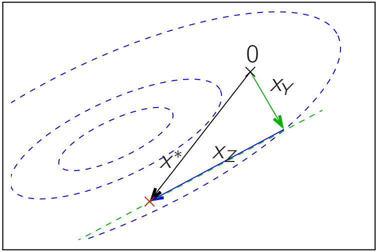
- $x_Z = Zp_Z$ optimizes in the null space

$$p_Z = -[Z^TQZ]^{-1}Z^T(g + QYp_Y)$$

  - Solves $\min_{p_Z} \ \frac{1}{2}p_Z^T[Z^TQZ]p_Z + (g + QYp_Y)^TZp_Z$ ("reduced QP")

Step Decomposition

$$\boxed{\begin{aligned} Qx + g + A^T\lambda &= 0 \\ Ax + b &= 0 \end{aligned}}$$



- $x_Y = Yp_Y$ is a step into the constraints:

$$0 = Ax + b = AYp_Y + AZp_Z + b \quad \Longrightarrow \quad p_Y = -[AY]^{-1}b$$

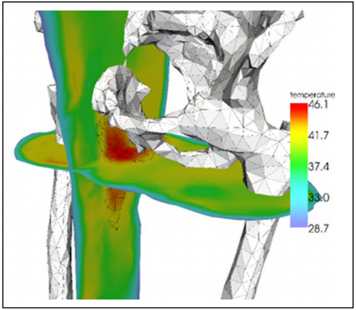
- $x_Z = Zp_Z$ optimizes in the null space

$$p_Z = -[Z^TQZ]^{-1}Z^T(g + QYp_Y)$$

  - Solves $\min_{p_Z} \ \frac{1}{2}p_Z^T[Z^TQZ]p_Z + (g + QYp_Y)^TZp_Z$ ("reduced QP")
- $\lambda = -[AY]^{-T}Y(Qx^* + g)$

Example: PDE-Constrained Optimization

$$
\begin{aligned}
\min_{T,u}\ & \frac{1}{2}\int (T(z) - \hat{T}(z))^2 dz + \frac{\alpha}{2}\|u\|^2 \\
\text{s.t.}\ & -\Delta T(z) = \sum_{i=1}^{n_u} k_i(z) u_i \text{ on } \Omega \\
& T(z) = b(z) \text{ on } \partial\Omega
\end{aligned}
$$

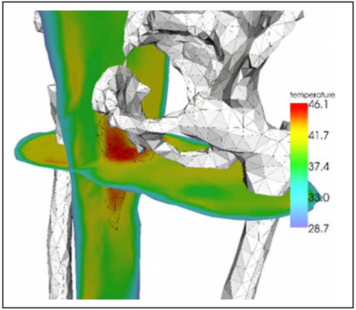


Example: PDE-Constrained Optimization

$$\min_{T,u} \frac{1}{2}\int (T(z) - \hat{T}(z))^2 dz + \frac{\alpha}{2}\|u\|^2$$

$$\text{s.t.} \quad -\Delta T(z) = \sum_{i=1}^{n_u} k_i(z) u_i \text{ on } \Omega$$

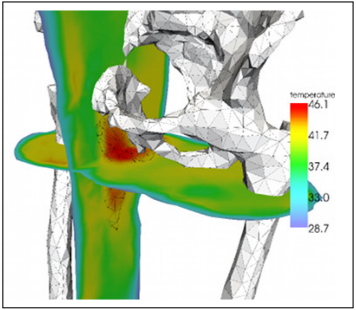
$$T(z) = b(z) \text{ on } \partial\Omega$$



- Given the (independent) control variable u:
  - (Dependent) state T is solution of PDE
  - Can use well-established solution techniques

Example: PDE-Constrained Optimization

$$
\begin{aligned}
\min_{T,u}\ & \frac{1}{2}\int (T(z)-\hat{T}(z))^2 dz + \frac{\alpha}{2}\|u\|^2 \\
\text{s.t.}\ & -\Delta T(z) = \sum_{i=1}^{n_u} k_i(z) u_i \text{ on } \Omega \\
& T(z) = b(z) \text{ on } \partial\Omega
\end{aligned}
$$



- Given the (independent) control variable u:
  - (Dependent) state T is solution of PDE
  - Can use well-established solution techniques
- We have only n_u degrees of freedom

Discretized PDE-Constrained Problem

$$\begin{aligned}\min_{T,u}\ & \frac{1}{2}\int (T(z)-\hat{T}(z))^2 dz + \frac{\alpha}{2}\|u\|^2 \\ \text{s.t.}\ & -\Delta T(z) = \sum_{i=1}^{n_u} k_i(z)u_i \text{ on } \Omega \\ & T(z) = b(z) \text{ on } \partial\Omega\end{aligned}$$

$$\begin{aligned}\min_{t,u}\ & \sum_{i=1}^{n}(t_i-\hat{t}_i)^2 + \sum_{i=1}^{n_u} u_i^2 \\ \text{s.t.}\ & \overline{D}\,t + \overline{K}u + \overline{b} = 0\end{aligned}$$

- Discretized state variables $t \in \mathbb{R}^N$
- Discretized non-singular(!) differential operator $\overline{D} \in \mathbb{R}^{N\times N}$

Discretized PDE-Constrained Problem

$$\begin{aligned} \min_{T,u}\ & \frac{1}{2}\int (T(z) - \hat{T}(z))^2 dz + \frac{\alpha}{2}\|u\|^2 \\ \text{s.t.}\ & -\Delta T(z) = \sum_{i=1}^{n_u} k_i(z) u_i \text{ on } \Omega \\ & T(z) = b(z) \text{ on } \partial\Omega \end{aligned}$$

$$\begin{aligned} \min_{t,u}\ & \sum_{i=1}^{n} (t_i - \hat{t}_i)^2 + \sum_{i=1}^{n_u} u_i^2 \\ \text{s.t.}\ & \overline{D}\, t + \overline{K} u + \overline{b} = 0 \end{aligned}$$

- Discretized state variables $t \in \mathbb{R}^N$
- Discretized non-singular(!) differential operator $\overline{D} \in \mathbb{R}^{N\times N}$
  - Given controls u, the state variables can be computed from

$$t = -\overline{D}^{-1}(\overline{K} u + \overline{b}).$$

Discretized PDE-Constrained Problem

$$\min_{T,u} \frac{1}{2}\int (T(z) - \hat{T}(z))^2 dz + \frac{\alpha}{2}\|u\|^2$$
$$\text{s.t.} \; -\Delta T(z) = \sum_{i=1}^{n_u} k_i(z) u_i \text{ on } \Omega$$
$$T(z) = b(z) \text{ on } \partial\Omega$$

$$\min_{t,u} \sum_{i=1}^{n} (t_i - \hat{t}_i)^2 + \sum_{i=1}^{n_u} u_i^2$$
$$\text{s.t.} \; \overline{D}\, t + \overline{K} u + \overline{b} = 0$$

- Discretized state variables $t \in \mathbb{R}^N$
- Discretized non-singular(!) differential operator $\overline{D} \in \mathbb{R}^{N \times N}$
  - Given controls u, the state variables can be computed from

$$t = -\overline{D}^{-1}(\overline{K} u + \overline{b}).$$

- We could just eliminate t and solve lower-dimensional problem in u

Generalization: Elimination of Variables

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & \tfrac{1}{2}(x_B^T \ x_N^T) Q \begin{pmatrix} x_B \\ x_N \end{pmatrix} + g_B^T x_B + g_N^T x_N \\ \text{s.t. } & B x_B + N x_N + b = 0 \end{aligned}$$

$$Y = \begin{bmatrix} I \\ 0 \end{bmatrix}$$

$$Z = \begin{bmatrix} -B^{-1} N \\ I \end{bmatrix}$$

Generalization: Elimination of Variables

$$
\begin{aligned}
\min_{x\in\mathbb{R}^n} \quad & \tfrac{1}{2}(x_B^T \; x_N^T)Q\begin{pmatrix} x_B \\ x_N \end{pmatrix} + g_B^T x_B + g_N^T x_N \\
\text{s.t.} \quad & B x_B + N x_N + b = 0
\end{aligned}
$$

$$
Y = \begin{bmatrix} I \\ 0 \end{bmatrix} \qquad Z = \begin{bmatrix} -B^{-1}N \\ I \end{bmatrix}
$$

$$
\begin{aligned}
p_Y &= -[AY]^{-1}d && = -B^{-1}b \\
p_Z &= -[Z^TQZ]^{-1}Z^T(g + QYp_Y) && \\
\lambda &= -[AY]^{-T}Y(Qx^* + g) && = -B^{-T}Y(Qx^* + g)
\end{aligned}
$$

- Can use existing implementations of operator B^{-1}:
  - Compute Z and p_Z (assuming N has few columns).
  - Compute λ^* (assuming that we have implementation for B^{-T}).

Generalization: Elimination of Variables

$$\min_{x\in\mathbb{R}^n} \ \tfrac{1}{2}(x_B^T \ x_N^T) Q \begin{pmatrix} x_B \\ x_N \end{pmatrix} + g_B^T x_B + g_N^T x_N$$

$$\text{s.t. } B x_B + N x_N + b = 0$$

$$Y = \begin{bmatrix} I \\ 0 \end{bmatrix} \qquad Z = \begin{bmatrix} -B^{-1}N \\ I \end{bmatrix}$$

$$\begin{aligned} p_Y &= -[AY]^{-1} d &&= -B^{-1} b \\ p_Z &= -[Z^T Q Z]^{-1} Z^T (g + QYp_Y) && \\ \lambda &= -[AY]^{-T} Y (Qx^* + g) &&= -B^{-T} Y (Qx^* + g) \end{aligned}$$

- Can use existing implementations of operator B^{-1}:
  - Compute Z and p_Z (assuming N has few columns).
  - Compute λ^* (assuming that we have implementation for B^{-T}).
- Tailored implementation for "simulation" often already exist.

Generalization: Elimination of Variables

$$\begin{aligned}\min_{x\in\mathbb{R}^n} \quad & \tfrac{1}{2}(x_B^T \; x_N^T) Q \begin{pmatrix} x_B \\ x_N \end{pmatrix} + g_B^T x_B + g_N^T x_N \\ \text{s.t.} \quad & B x_B + N x_N + b = 0\end{aligned}$$

$$Y = \begin{bmatrix} I \\ 0 \end{bmatrix} \qquad Z = \begin{bmatrix} -B^{-1}N \\ I \end{bmatrix}$$

$$\begin{aligned} p_Y &= -[AY]^{-1} d && = -B^{-1} b \\ p_Z &= -[Z^T Q Z]^{-1} Z^T (g + QYp_Y) && \\ \lambda &= -[AY]^{-T} Y(Qx^* + g) && = -B^{-T} Y(Qx^* + g)\end{aligned}$$

- Can use existing implementations of operator B^{-1}:
  - Compute Z and p_Z (assuming N has few columns).
  - Compute λ^* (assuming that we have implementation for B^{-T}).
- Tailored implementation for "simulation" often already exist.
- Exploit problem structure!

Solution of EQP Summary

- Direct method:
  - Factorize KKT matrix
  - If $L^T BL$ factorization is used, we can determine if x^* is indeed a minimizer
  - Easy general purpose option

Solution of EQP Summary

- Direct method:
  - Factorize KKT matrix
  - If $L^T BL$ factorization is used, we can determine if x^* is indeed a minimizer
  - Easy general purpose option
- Schur-complement Method
  - Requires that Q is positive definite and easy to solve (e.g., diagonal)
  - Number of constraints m should not be large

Solution of EQP Summary

- Direct method:
  - Factorize KKT matrix
  - If $L^T BL$ factorization is used, we can determine if x^* is indeed a minimizer
  - Easy general purpose option
- Schur-complement Method
  - Requires that Q is positive definite and easy to solve (e.g., diagonal)
  - Number of constraints m should not be large
- Null-space method
  - Step decomposition into range-space step and null-space step
  - Permits exploitation of constraint matrix structure
  - Number of degrees of freedom $(n - m)$ should not be large

Table of Contents

Inequality-Constrained QPs

$$\boxed{\begin{aligned} \min_{x\in\mathbb{R}^n} \ & \frac{1}{2}x^TQx + g^Tx \\ \text{s.t. } & a_i^Tx + b_i = 0 \text{ for } i \in \mathcal{E} \\ & a_i^Tx + b_i \leq 0 \text{ for } i \in \mathcal{I} \end{aligned}}$$

Inequality-Constrained QPs

$$\begin{aligned}\min_{x\in\mathbb{R}^n}\ & \tfrac{1}{2}x^TQx + g^Tx \\ \text{s.t. } & a_i^Tx + b_i = 0 \text{ for } i \in \mathcal{E} \\ & a_i^Tx + b_i \le 0 \text{ for } i \in \mathcal{I}\end{aligned}$$

$$\begin{aligned}Qx + g + \sum_{i\in\mathcal{E}\cup\mathcal{I}} a_i\lambda_i = 0& \\ a_i^Tx + b_i = 0 \text{ for } i \in \mathcal{E}& \\ a_i^Tx + b_i \le 0 \text{ for } i \in \mathcal{I}& \\ \lambda_i \ge 0 \text{ for } i \in \mathcal{I}& \\ (a_i^Tx + b_i)\lambda_i = 0 \text{ for } i \in \mathcal{I}&\end{aligned}$$

Inequality-Constrained QPs

$$
\begin{aligned}
\min_{x \in \mathbb{R}^n} \ & \tfrac{1}{2} x^T Q x + g^T x \\
\text{s.t. } & a_i^T x + b_i = 0 \text{ for } i \in \mathcal{E} \\
& a_i^T x + b_i \le 0 \text{ for } i \in \mathcal{I}
\end{aligned}
$$

$$
\begin{aligned}
Qx + g + \sum_{i \in \mathcal{E} \cup \mathcal{I}} a_i \lambda_i &= 0 \\
a_i^T x + b_i &= 0 \text{ for } i \in \mathcal{E} \\
a_i^T x + b_i &\le 0 \text{ for } i \in \mathcal{I} \\
\lambda_i &\ge 0 \text{ for } i \in \mathcal{I} \\
(a_i^T x + b_i)\lambda_i &= 0 \text{ for } i \in \mathcal{I}
\end{aligned}
$$

- Assume:
  - Q is positive definite;
  - $\{a_i\}_{i \in \mathcal{E}}$ are linearly independent.

Inequality-Constrained QPs

$$\begin{aligned}\min_{x\in\mathbb{R}^n}\ & \tfrac{1}{2}x^TQx + g^Tx \\ \text{s.t. } & a_i^Tx + b_i = 0 \text{ for } i \in \mathcal{E} \\ & a_i^Tx + b_i \le 0 \text{ for } i \in \mathcal{I}\end{aligned}$$

$$\begin{aligned}Qx + g + \sum_{i\in\mathcal{E}\cup\mathcal{I}} a_i\lambda_i &= 0 \\ a_i^Tx + b_i &= 0 \text{ for } i \in \mathcal{E} \\ a_i^Tx + b_i &\le 0 \text{ for } i \in \mathcal{I} \\ \lambda_i &\ge 0 \text{ for } i \in \mathcal{I} \\ (a_i^Tx + b_i)\lambda_i &= 0 \text{ for } i \in \mathcal{I}\end{aligned}$$

- Assume:
  - Q is positive definite;
  - $\{a_i\}_{i\in\mathcal{E}}$ are linearly independent.
- Difficulty: Decide, which inequality constraints are active.

Inequality-Constrained QPs

$$
\begin{aligned}
\min_{x\in\mathbb{R}^n}\ & \tfrac{1}{2}x^T Qx + g^T x \\
\text{s.t. } & a_i^T x + b_i = 0 \text{ for } i \in \mathcal{E} \\
& a_i^T x + b_i \le 0 \text{ for } i \in \mathcal{I}
\end{aligned}
$$

$$
\begin{aligned}
Qx + g + \sum_{i\in\mathcal{E}\cup\mathcal{I}} a_i\lambda_i = 0 & \\
a_i^T x + b_i = 0 \text{ for } i \in \mathcal{E} \\
a_i^T x + b_i \le 0 \text{ for } i \in \mathcal{I} \\
\lambda_i \ge 0 \text{ for } i \in \mathcal{I} \\
(a_i^T x + b_i)\lambda_i = 0 \text{ for } i \in \mathcal{I}
\end{aligned}
$$

- Assume:
  - Q is positive definite;
  - $\{a_i\}_{i\in\mathcal{E}}$ are linearly independent.
- Difficulty: Decide, which inequality constraints are active.
- We know how to solve equality-constrained QPs.
  - Can we use that here?

Working Set

Choose working set $\mathcal{W} \subseteq \mathcal{I}$ and solve

$$\boxed{\begin{aligned} \min_{x \in \mathbb{R}^n} \ & \tfrac{1}{2} x^T Q x + g^T x \\ \text{s.t. } & a_i^T x + b_i = 0 \text{ for } i \in \mathcal{E} \\ & a_i^T x + b_i = 0 \text{ for } i \in \mathcal{W} \end{aligned}}$$

Working Set

Choose working set $\mathcal{W} \subseteq \mathcal{I}$ and solve

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & \tfrac{1}{2} x^T Q x + g^T x \\ \text{s.t. } & a_i^T x + b_i = 0 \text{ for } i \in \mathcal{E} \\ & a_i^T x + b_i = 0 \text{ for } i \in \mathcal{W} \end{aligned}$$

$$\begin{aligned} Qx + g + \sum_{i \in \mathcal{E} \cup \mathcal{W}} a_i \lambda_i &= 0 \\ a_i^T x + b_i &= 0 \text{ for } i \in \mathcal{E} \\ a_i^T x + b_i &= 0 \text{ for } i \in \mathcal{W} \end{aligned}$$

Working Set

Choose working set $\mathcal{W} \subseteq \mathcal{I}$ and solve

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \tfrac{1}{2} x^T Q x + g^T x \\ \text{s.t.} \quad & a_i^T x + b_i = 0 \text{ for } i \in \mathcal{E} \\ & a_i^T x + b_i = 0 \text{ for } i \in \mathcal{W} \end{aligned}$$

$$\begin{aligned} Qx + g + \sum_{i \in \mathcal{E} \cup \mathcal{W}} a_i \lambda_i &= 0 \\ a_i^T x + b_i &= 0 \text{ for } i \in \mathcal{E} \\ a_i^T x + b_i &= 0 \text{ for } i \in \mathcal{W} \end{aligned}$$

Set missing multipliers $\lambda_i = 0$ for $i \in \mathcal{I} \setminus \mathcal{W}$ and verify

$$a_i^T x + b_i \overset{?}{\leq} 0 \text{ for } i \in \mathcal{I} \setminus \mathcal{W}$$

$$\lambda_i \overset{?}{\geq} 0 \text{ for } i \in \mathcal{I}$$

Working Set

Choose working set $\mathcal{W} \subseteq \mathcal{I}$ and solve

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & \tfrac{1}{2} x^T Q x + g^T x \\ \text{s.t. } & a_i^T x + b_i = 0 \text{ for } i \in \mathcal{E} \\ & a_i^T x + b_i = 0 \text{ for } i \in \mathcal{W} \end{aligned}$$

$$\begin{aligned} Qx + g + \sum_{i \in \mathcal{E} \cup \mathcal{W}} a_i \lambda_i &= 0 \\ a_i^T x + b_i &= 0 \text{ for } i \in \mathcal{E} \\ a_i^T x + b_i &= 0 \text{ for } i \in \mathcal{W} \end{aligned}$$

Set missing multipliers $\lambda_i = 0$ for $i \in \mathcal{I} \setminus \mathcal{W}$ and verify

$$\begin{aligned} a_i^T x + b_i &\overset{?}{\leq} 0 \text{ for } i \in \mathcal{I} \setminus \mathcal{W} \\ \lambda_i &\overset{?}{\geq} 0 \text{ for } i \in \mathcal{I} \end{aligned}$$

- If satisfied, (x, λ) is the (unique) optimal solution

Working Set

Choose working set $\mathcal{W} \subseteq \mathcal{I}$ and solve

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \ & \tfrac{1}{2}x^TQx + g^Tx \\ \text{s.t. } & a_i^Tx + b_i = 0 \text{ for } i \in \mathcal{E} \\ & a_i^Tx + b_i = 0 \text{ for } i \in \mathcal{W} \end{aligned}$$

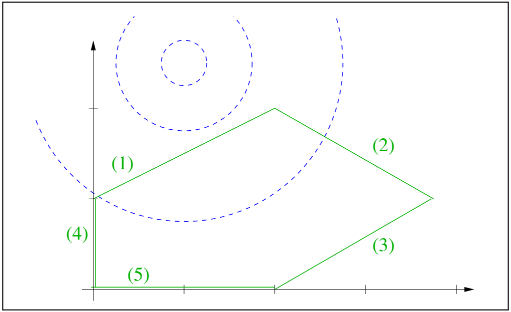
$$\begin{aligned} Qx + g + \sum_{i\in\mathcal{E}\cup\mathcal{W}} a_i\lambda_i &= 0 \\ a_i^Tx + b_i &= 0 \text{ for } i \in \mathcal{E} \\ a_i^Tx + b_i &= 0 \text{ for } i \in \mathcal{W} \end{aligned}$$

Set missing multipliers $\lambda_i = 0$ for $i \in \mathcal{I} \setminus \mathcal{W}$ and verify

$$\begin{aligned} a_i^Tx + b_i &\overset{?}{\leq} 0 \text{ for } i \in \mathcal{I} \setminus \mathcal{W} \\ \lambda_i &\overset{?}{\geq} 0 \text{ for } i \in \mathcal{I} \end{aligned}$$

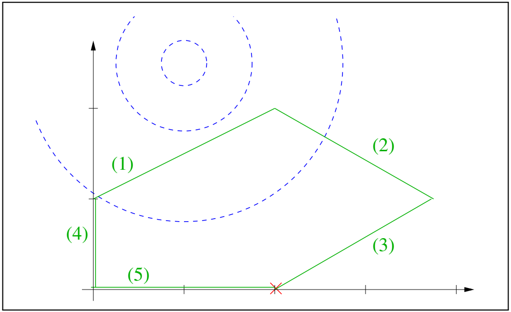
- If satisfied, (x, λ) is the (unique) optimal solution
- Otherwise, let's try a different working set

Example QP



$$
\begin{aligned}
\min\ & (x_1 - 1)^2 + (x_2 - 2.5)^2 \\
\text{s.t.}\ & -x_1 + 2x_2 - 2 \le 0 \ (1) \qquad -x_1 \le 0 \ (4) \\
& x_1 + 2x_2 - 6 \le 0 \ (2) \qquad -x_2 \le 0 \ (5) \\
& x_1 - 2x_2 - 2 \le 0 \ (3)
\end{aligned}
$$

Primal Active-Set QP Solver Iteration 1

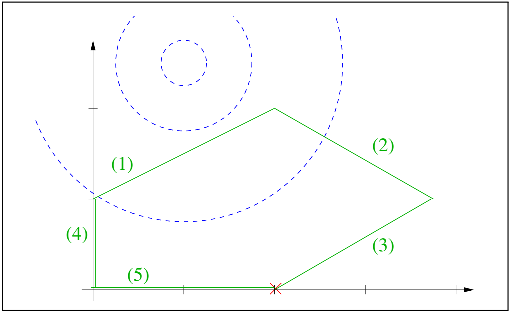


Initialization:

Choose feasible starting iterate x

$x = (0, 2)$

Primal Active-Set QP Solver Iteration 1



$$\mathcal{W} = \{3, 5\}$$
$$x = (0, 2)$$

Initialization:

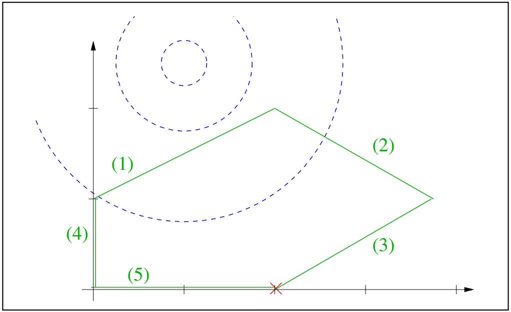
Choose feasible starting iterate x

Choose working set $\mathcal{W} \subseteq \mathcal{I}$ with

- $i \in \mathcal{W} \Longrightarrow a_i^T x + b_i = 0$
- $\{a_i\}_{i \in \mathcal{E} \cup \mathcal{W}}$ are linear independent

(Algorithm will maintain these properties)

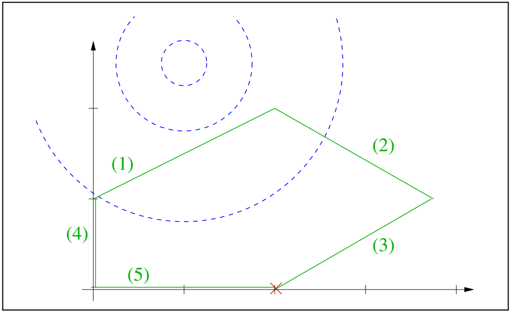
Primal Active-Set QP Solver Iteration 1



$$\mathcal{W} = \{3, 5\}$$
$$x = (0, 2)$$
$$x^{\text{EQP}} = (0, 2)$$
$$\lambda_3 = -2$$
$$\lambda_5 = -1$$

Solve (EQP)

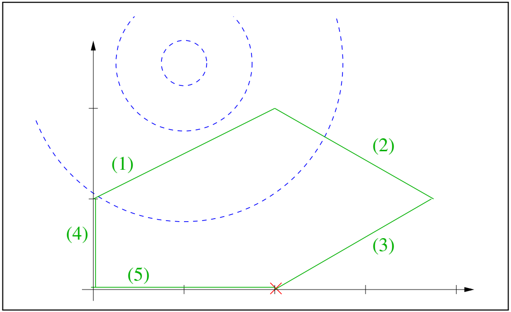
Primal Active-Set QP Solver Iteration 1



Status: Current iterate is optimal for (EQP).

$$\mathcal{W} = \{3, 5\}$$
$$x = (0, 2)$$
$$x^{\mathsf{EQP}} = (0, 2)$$
$$\lambda_3 = -2$$
$$\lambda_5 = -1$$

Primal Active-Set QP Solver Iteration 1



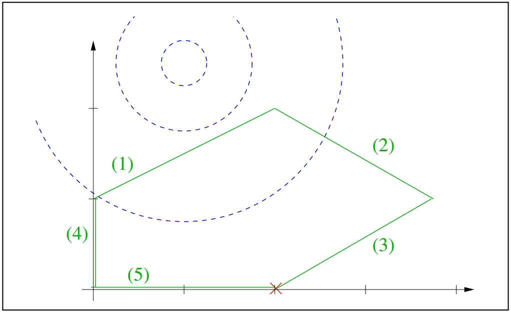
$$\mathcal{W} = \{3, 5\}$$
$$x = (0, 2)$$
$$x^{\text{EQP}} = (0, 2)$$
$$\lambda_3 = -2$$
$$\lambda_5 = -1$$

Status: Current iterate is optimal for (EQP).

Release Constraint:

- Pick constraint i with $\lambda_i < 0$.

Primal Active-Set QP Solver Iteration 1



$$\mathcal{W} = \{3, 5\}$$
$$x = (0, 2)$$
$$x^{\mathsf{EQP}} = (0, 2)$$
$$\lambda_3 = -2$$
$$\lambda_5 = -1$$

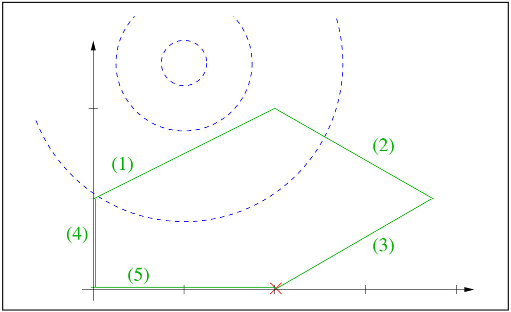
Status: Current iterate is optimal for (EQP).

<u>Release Constraint:</u>

- Pick constraint i with $\lambda_i < 0$.
- Remove i from working set:

$$\mathcal{W} \leftarrow \mathcal{W} \setminus \{3\} = \{5\}$$

Primal Active-Set QP Solver Iteration 1



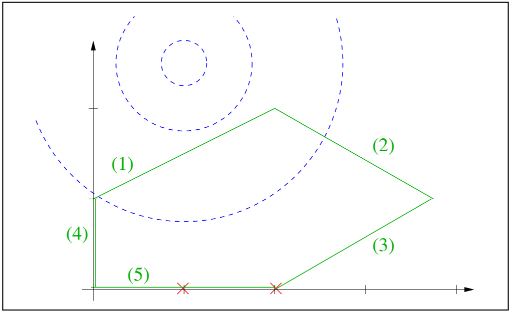
$$\mathcal{W} = \{3, 5\}$$
$$x = (0, 2)$$
$$x^{\mathrm{EQP}} = (0, 2)$$
$$\lambda_3 = -2$$
$$\lambda_5 = -1$$

Status: Current iterate is optimal for (EQP).

Release Constraint:

- Pick constraint i with $\lambda_i < 0$.
- Remove i from working set:
$$\mathcal{W} \leftarrow \mathcal{W} \setminus \{3\} = \{5\}$$
- Keep iterate $x = (0, 2)$.

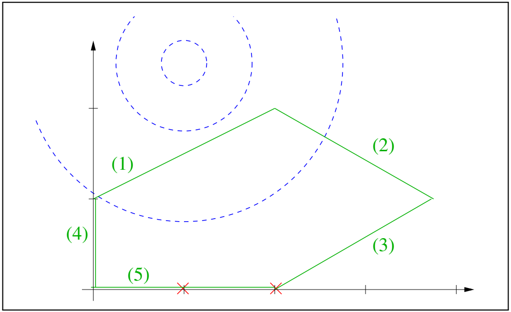
Primal Active-Set QP Solver Iteration 2



$$\mathcal{W} = \{5\}$$
$$x = (2, 0)$$
$$x^{\mathrm{EQP}} = (1, 0)$$
$$\lambda_5 = -5$$

Solve (EQP)

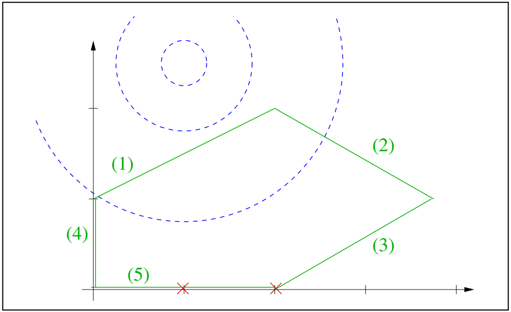
Primal Active-Set QP Solver Iteration 2



Status: Current iterate is not optimal for (EQP).

$$\mathcal{W} = \{5\}$$
$$x = (2, 0)$$
$$x^{\mathsf{EQP}} = (1, 0)$$
$$\lambda_5 = -5$$

Primal Active-Set QP Solver Iteration 2



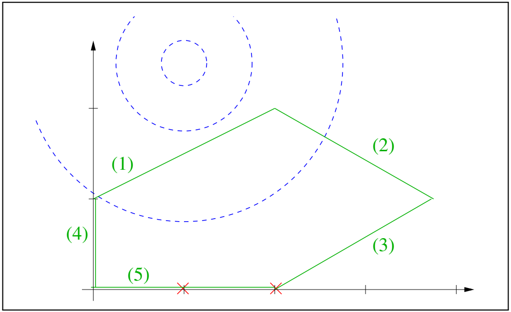
$$\mathcal{W} = \{5\}$$
$$x = (2,0)$$
$$x^{\mathsf{EQP}} = (1,0)$$
$$\lambda_5 = -5$$

Status: Current iterate is not optimal for (EQP).

Take step (x^{EQP} is feasible):

- Update iterate $x \leftarrow x^{\mathsf{EQP}}$

Primal Active-Set QP Solver Iteration 2



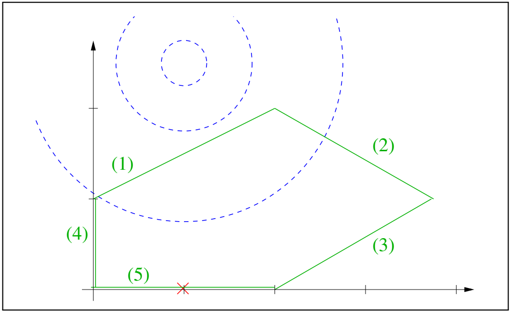
$$\mathcal{W} = \{5\}$$
$$x = (2, 0)$$
$$x^{\mathsf{EQP}} = (1, 0)$$
$$\lambda_5 = -5$$

Status: Current iterate is not optimal for (EQP).

Take step (x^{EQP} is feasible):

- Update iterate $x \leftarrow x^{\mathsf{EQP}}$
- Keep $\mathcal{W}$

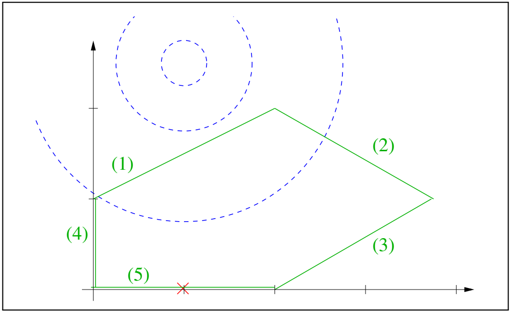
Primal Active-Set QP Solver Iteration 3



$$\mathcal{W} = \{5\}$$
$$x = (1, 0)$$
$$x^{\text{EQP}} = (1, 0)$$
$$\lambda_5 = -5$$

Solve (EQP)

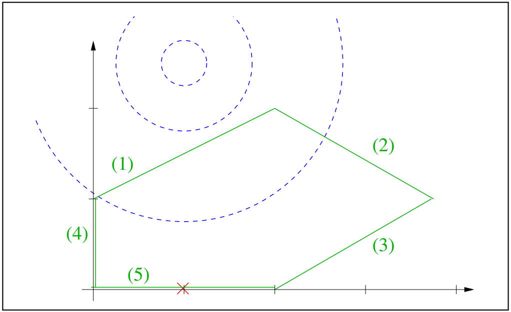
Primal Active-Set QP Solver Iteration 3



Status: Current iterate is optimal for (EQP)

$$\mathcal{W} = \{5\}$$
$$x = (1, 0)$$
$$x^{\mathsf{EQP}} = (1, 0)$$
$$\lambda_5 = -5$$

Primal Active-Set QP Solver Iteration 3



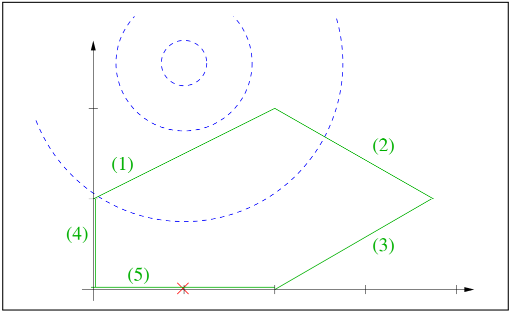
$$\mathcal{W} = \{5\}$$
$$x = (1, 0)$$
$$x^{\text{EQP}} = (1, 0)$$
$$\lambda_5 = -5$$

Status: Current iterate is optimal for (EQP)

Release Constraint:

- Pick constraint i with $\lambda_i < 0$.

Primal Active-Set QP Solver Iteration 3



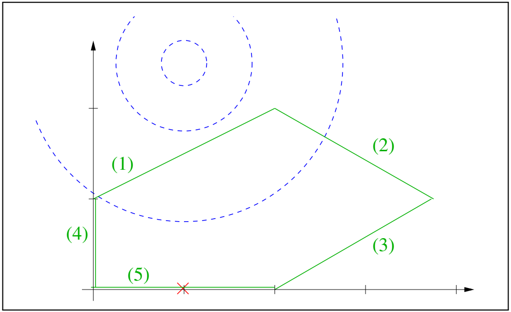
$$\mathcal{W} = \{5\}$$
$$x = (1, 0)$$
$$x^{\mathrm{EQP}} = (1, 0)$$
$$\lambda_5 = -5$$

Status: Current iterate is optimal for (EQP)

Release Constraint:

- Pick constraint i with $\lambda_i < 0$.
- Remove i from working set:
  $$\mathcal{W} \leftarrow \mathcal{W} \setminus \{5\} = \emptyset$$

Primal Active-Set QP Solver Iteration 3



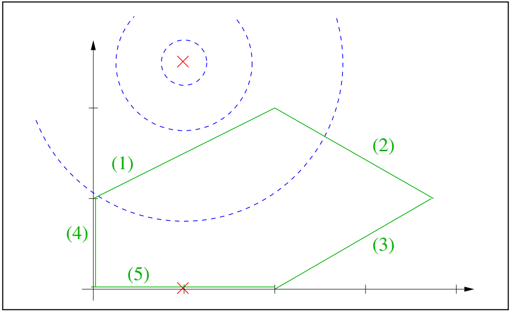
Status: Current iterate is optimal for (EQP)

$$\mathcal{W} = \{5\}$$
$$x = (1, 0)$$
$$x^{\text{EQP}} = (1, 0)$$
$$\lambda_5 = -5$$

Release Constraint:

- Pick constraint i with $\lambda_i < 0$.
- Remove i from working set:
$$\mathcal{W} \leftarrow \mathcal{W} \setminus \{5\} = \emptyset$$
- Keep iterate $x = (1, 0)$.

Primal Active-Set QP Solver Iteration 4

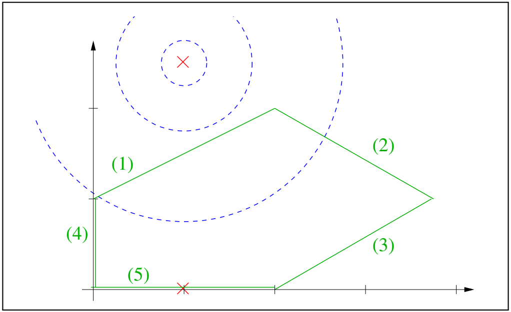


$$\mathcal{W} = \emptyset$$

$$x = (1, 0) \qquad \text{Solve (EQP)}$$

$$x^{\text{EQP}} = (1, 2.5)$$

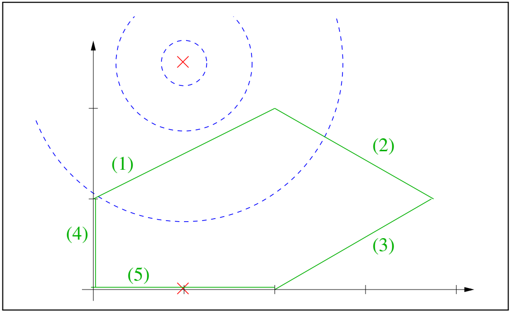
Primal Active-Set QP Solver Iteration 4



Status: Current iterate not optimal for (EQP)

$$\mathcal{W} = \emptyset$$
$$x = (1, 0)$$
$$x^{\text{EQP}} = (1, 2.5)$$

Primal Active-Set QP Solver Iteration 4



Status: Current iterate not optimal for (EQP)

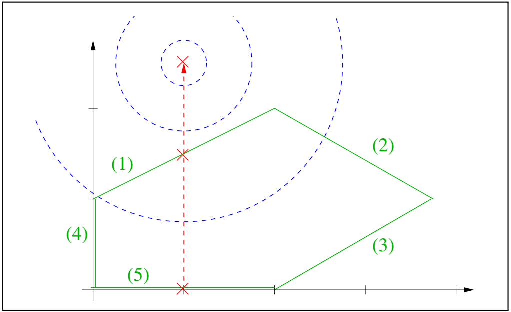
<u>Take step (x^{EQP} not feasible):</u>

$$\mathcal{W} = \emptyset$$

$$x = (1, 0)$$

$$x^{\mathsf{EQP}} = (1, 2.5)$$

Primal Active-Set QP Solver Iteration 4



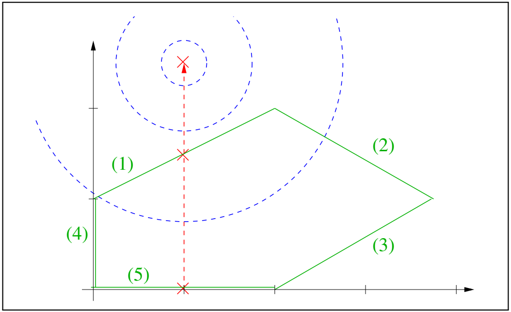
Status: Current iterate not optimal for (EQP)

Take step (x^{EQP} not feasible):

- Largest $\alpha \in [0, 1]$: $x + \alpha(x^{\mathsf{EQP}} - x)$ feasible

$$\mathcal{W} = \emptyset$$
$$x = (1, 0)$$
$$x^{\mathsf{EQP}} = (1, 2.5)$$

Primal Active-Set QP Solver Iteration 4



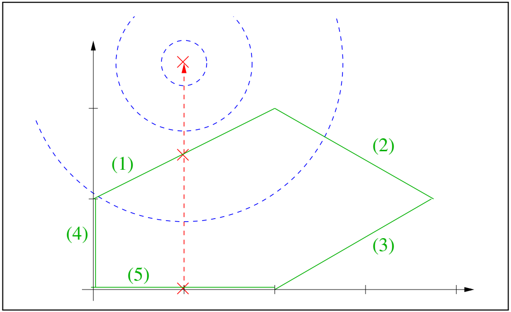
Status: Current iterate not optimal for (EQP)

Take step (x^{EQP} not feasible):

- Largest $\alpha \in [0, 1]$: $x + \alpha(x^{\mathrm{EQP}} - x)$ feasible
- Update iterate $x \leftarrow x + \alpha(x^{\mathrm{EQP}} - x)$

$$\mathcal{W} = \emptyset$$
$$x = (1, 0)$$
$$x^{\mathrm{EQP}} = (1, 2.5)$$

Primal Active-Set QP Solver Iteration 4



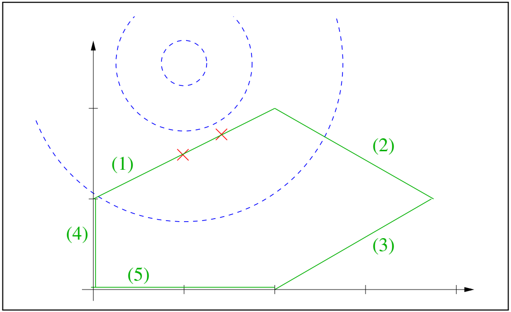
Status: Current iterate not optimal for (EQP)

Take step (x^{EQP} not feasible):

$$\mathcal{W} = \emptyset$$
$$x = (1, 0)$$
$$x^{\mathsf{EQP}} = (1, 2.5)$$

- Largest $\alpha \in [0, 1]$: $x + \alpha(x^{\mathsf{EQP}} - x)$ feasible
- Update iterate $x \leftarrow x + \alpha(x^{\mathsf{EQP}} - x)$
- Update $\mathcal{W} \leftarrow \mathcal{W} \cup \{i\} = \{1\}$
  - where constraint $i = 1$ is "blocking"

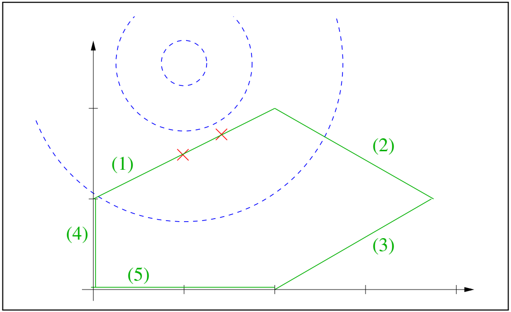
Primal Active-Set QP Solver Iteration 5



$$\mathcal{W} = \{1\}$$
$$x = (1, 1.5)$$
$$x^{\text{EQP}} = (1.4, 1.7)$$
$$\lambda_1 = 0.8$$

Solve (EQP)

Primal Active-Set QP Solver Iteration 5



$$\mathcal{W} = \{1\}$$

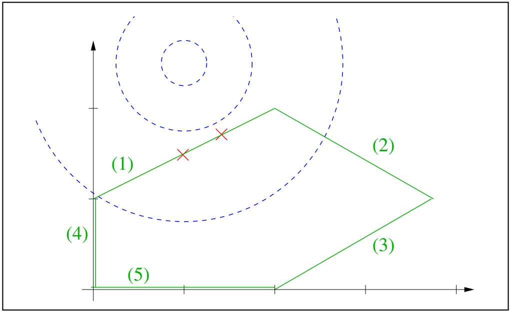
$$x = (1, 1.5)$$

$$x^{\mathsf{EQP}} = (1.4, 1.7)$$

$$\lambda_1 = 0.8$$

Status: Current iterate is not optimal for (EQP).

Primal Active-Set QP Solver Iteration 5



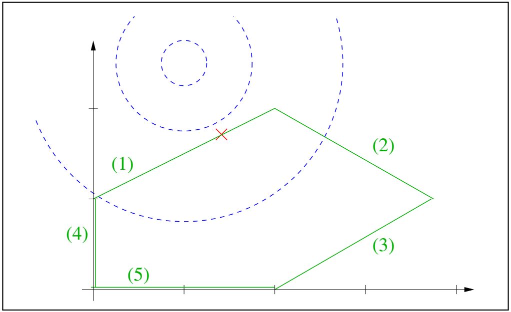
$$\mathcal{W} = \{1\}$$
$$x = (1, 1.5)$$
$$x^{\mathsf{EQP}} = (1.4, 1.7)$$
$$\lambda_1 = 0.8$$

Status: Current iterate is not optimal for (EQP).

Take step (x^{EQP} feasible):

- Update iterate $x \leftarrow x^{\mathsf{EQP}}$.
- Keep $\mathcal{W}$.

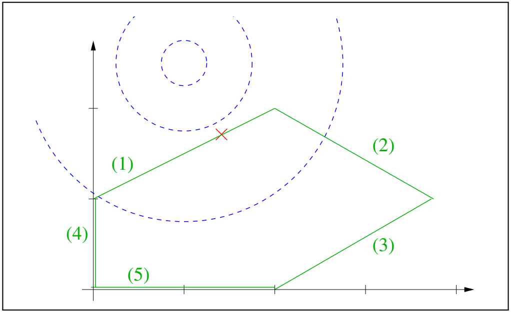
Primal Active-Set QP Solver Iteration 6



$$\mathcal{W} = \{1\}$$
$$x = (1.4, 1.7)$$
$$x^{\mathrm{EQP}} = (1.4, 1.7)$$
$$\lambda_1 = 0.8$$

Solve (EQP)

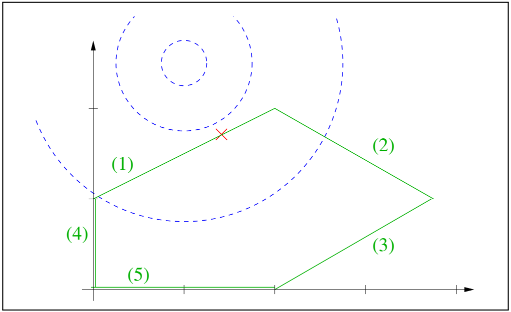
Primal Active-Set QP Solver Iteration 6



$$\mathcal{W} = \{1\}$$
$$x = (1.4, 1.7)$$
$$x^{\mathsf{EQP}} = (1.4, 1.7)$$
$$\lambda_1 = 0.8$$

Status: Current iterate is optimal for (EQP)

Primal Active-Set QP Solver Iteration 6

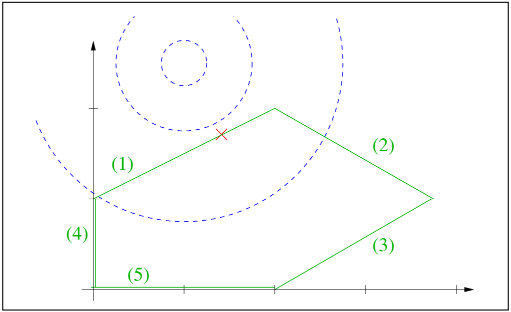


$$\mathcal{W} = \{1\}$$
$$x = (1.4, 1.7)$$
$$x^{\text{EQP}} = (1.4, 1.7)$$
$$\lambda_1 = 0.8$$

Status: Current iterate is optimal for (EQP)

- $\lambda_i \geq 0$ for all $i \in \mathcal{W}$.

Primal Active-Set QP Solver Iteration 6



$$\mathcal{W} = \{1\}$$
$$x = (1.4, 1.7)$$
$$x^{\text{EQP}} = (1.4, 1.7)$$
$$\lambda_1 = 0.8$$

Status: Current iterate is optimal for (EQP)

- $\lambda_i \geq 0$ for all $i \in \mathcal{W}$.

Declare Optimality!

Primal Active-Set QP Method

1. Select feasible x and $\mathcal{W} \subseteq \mathcal{I} \cap \mathcal{A}(x)$.

Primal Active-Set QP Method

1. Select feasible x and $\mathcal{W} \subseteq \mathcal{I} \cap \mathcal{A}(x)$.
2. Solve (EQP) to get x^{EQP} and λ^{EQP}.

Primal Active-Set QP Method

1. Select feasible x and $\mathcal{W} \subseteq \mathcal{I} \cap \mathcal{A}(x)$.
2. Solve (EQP) to get x^{EQP} and λ^{EQP}.
3. If $x = x^{\mathsf{EQP}}$:

Primal Active-Set QP Method

1. Select feasible x and $\mathcal{W} \subseteq \mathcal{I} \cap \mathcal{A}(x)$.
2. Solve (EQP) to get x^{EQP} and λ^{EQP}.
3. If $x = x^{\text{EQP}}$:
   - If $\lambda^{\text{EQP}} \geq 0$: Done!

Primal Active-Set QP Method

1. Select feasible x and $\mathcal{W} \subseteq \mathcal{I} \cap \mathcal{A}(x)$.
2. Solve (EQP) to get x^{EQP} and λ^{EQP}.
3. If $x = x^{\text{EQP}}$:
   - If $\lambda^{\text{EQP}} \geq 0$: Done!
   - Otherwise, select $\lambda_i^{\text{EQP}} < 0$ and set $\mathcal{W} \leftarrow \mathcal{W} \setminus \{i\}$.

Primal Active-Set QP Method

1. Select feasible x and $\mathcal{W} \subseteq \mathcal{I} \cap \mathcal{A}(x)$.
2. Solve (EQP) to get x^{EQP} and λ^{EQP}.
3. If $x = x^{\mathsf{EQP}}$:
   - If $\lambda^{\mathsf{EQP}} \geq 0$: Done!
   - Otherwise, select $\lambda_i^{\mathsf{EQP}} < 0$ and set $\mathcal{W} \leftarrow \mathcal{W} \setminus \{i\}$.
4. If $x \neq x^{\mathsf{EQP}}$:

Primal Active-Set QP Method

1. Select feasible x and $\mathcal{W} \subseteq \mathcal{I} \cap \mathcal{A}(x)$.
2. Solve (EQP) to get x^{EQP} and λ^{EQP}.
3. If $x = x^{\text{EQP}}$:
   - If $\lambda^{\text{EQP}} \geq 0$: Done!
   - Otherwise, select $\lambda_i^{\text{EQP}} < 0$ and set $\mathcal{W} \leftarrow \mathcal{W} \setminus \{i\}$.
4. If $x \neq x^{\text{EQP}}$:
   - Compute step $p = x^{\text{EQP}} - x$.

Primal Active-Set QP Method

1. Select feasible x and $\mathcal{W} \subseteq \mathcal{I} \cap \mathcal{A}(x)$.
2. Solve (EQP) to get x^{EQP} and λ^{EQP}.
3. If $x = x^{\text{EQP}}$:
   - If $\lambda^{\text{EQP}} \geq 0$: Done!
   - Otherwise, select $\lambda_i^{\text{EQP}} < 0$ and set $\mathcal{W} \leftarrow \mathcal{W} \setminus \{i\}$.
4. If $x \neq x^{\text{EQP}}$:
   - Compute step $p = x^{\text{EQP}} - x$.
   - Compute $\alpha = \arg\max\{\alpha \in [0, 1] : x + \alpha p \text{ is feasible}\}$.

Primal Active-Set QP Method

1. Select feasible x and $\mathcal{W} \subseteq \mathcal{I} \cap \mathcal{A}(x)$.
2. Solve (EQP) to get x^{EQP} and λ^{EQP}.
3. If $x = x^{\text{EQP}}$:
   - If $\lambda^{\text{EQP}} \geq 0$: Done!
   - Otherwise, select $\lambda_i^{\text{EQP}} < 0$ and set $\mathcal{W} \leftarrow \mathcal{W} \setminus \{i\}$.
4. If $x \neq x^{\text{EQP}}$:
   - Compute step $p = x^{\text{EQP}} - x$.
   - Compute $\alpha = \arg\max\{\alpha \in [0,1] : x + \alpha p \text{ is feasible}\}$.
   - If $\alpha < 1$, pick $i \in \mathcal{I} \setminus \mathcal{W}$ with $a_i^T p > 0$ and $a_i^T(x + \alpha p) + b_i = 0$, and set $\mathcal{W} \leftarrow \mathcal{W} \cup \{i\}$.

Primal Active-Set QP Method

1. Select feasible x and $\mathcal{W} \subseteq \mathcal{I} \cap \mathcal{A}(x)$.
2. Solve (EQP) to get x^{EQP} and λ^{EQP}.
3. If $x = x^{\text{EQP}}$:
   - If $\lambda^{\text{EQP}} \geq 0$: Done!
   - Otherwise, select $\lambda_i^{\text{EQP}} < 0$ and set $\mathcal{W} \leftarrow \mathcal{W} \setminus \{i\}$.
4. If $x \neq x^{\text{EQP}}$:
   - Compute step $p = x^{\text{EQP}} - x$.
   - Compute $\alpha = \arg\max\{\alpha \in [0,1] : x + \alpha p \text{ is feasible}\}$.
   - If $\alpha < 1$, pick $i \in \mathcal{I} \setminus \mathcal{W}$ with $a_i^T p > 0$ and $a_i^T(x + \alpha p) + b_i = 0$, and set $\mathcal{W} \leftarrow \mathcal{W} \cup \{i\}$.
   - Update $x \leftarrow x + \alpha p$.
5. Go to step 2.

Active-Set QP Algorithms

- Primal active-set method:
  - Keeps all iterates feasible.
  - Changes $\mathcal{W}$ by at most one constraint per iteration.
  - $\{a_i\}_{i\in\mathcal{E}\cup\mathcal{W}}$ remain linearly independent.

Active-Set QP Algorithms

- Primal active-set method:
  - Keeps all iterates feasible.
  - Changes $\mathcal{W}$ by at most one constraint per iteration.
  - $\{a_i\}_{i\in\mathcal{E}\cup\mathcal{W}}$ remain linearly independent.
- Convergence
  - Finite convergence:
    - Finitely many options for $\mathcal{W}$.
    - Objective decreases with every step;

Active-Set QP Algorithms

- Primal active-set method:
  - Keeps all iterates feasible.
  - Changes $\mathcal{W}$ by at most one constraint per iteration.
  - $\{a_i\}_{i\in\mathcal{E}\cup\mathcal{W}}$ remain linearly independent.
- Convergence
  - Finite convergence:
    - Finitely many options for $\mathcal{W}$.
    - Objective decreases with every step; as long as $\alpha > 0$!
  - Special handling of degeneracy necessary ($\alpha = 0$ steps)

Active-Set QP Algorithms

- Primal active-set method:
  - Keeps all iterates feasible.
  - Changes $\mathcal{W}$ by at most one constraint per iteration.
  - $\{a_i\}_{i\in\mathcal{E}\cup\mathcal{W}}$ remain linearly independent.
- Convergence
  - Finite convergence:
    - Finitely many options for $\mathcal{W}$.
    - Objective decreases with every step; as long as $\alpha > 0$!
  - Special handling of degeneracy necessary ($\alpha = 0$ steps)
- Efficient solution of (EQP)
  - Update the factorization when $\mathcal{W}$ changes.

Active-Set QP Algorithms

- Primal active-set method:
  - Keeps all iterates feasible.
  - Changes $\mathcal{W}$ by at most one constraint per iteration.
  - $\{a_i\}_{i\in\mathcal{E}\cup\mathcal{W}}$ remain linearly independent.
- Convergence
  - Finite convergence:
    - Finitely many options for $\mathcal{W}$.
    - Objective decreases with every step; as long as $\alpha > 0$!
  - Special handling of degeneracy necessary ($\alpha = 0$ steps)
- Efficient solution of (EQP)
  - Update the factorization when $\mathcal{W}$ changes.
- Complexity
  - Fast convergence if good estimate of optimal working set is given.
  - Worst case exponential complexity.
  - Alternative: Interior-point QP solvers (polynomial complexity).

Active-Set QP Algorithms

- Primal active-set method:
  - Keeps all iterates feasible.
  - Changes $\mathcal{W}$ by at most one constraint per iteration.
  - $\{a_i\}_{i\in\mathcal{E}\cup\mathcal{W}}$ remain linearly independent.
- Convergence
  - Finite convergence:
    - Finitely many options for $\mathcal{W}$.
    - Objective decreases with every step; as long as $\alpha > 0$!
  - Special handling of degeneracy necessary ($\alpha = 0$ steps)
- Efficient solution of (EQP)
  - Update the factorization when $\mathcal{W}$ changes.
- Complexity
  - Fast convergence if good estimate of optimal working set is given.
  - Worst case exponential complexity.
  - Alternative: Interior-point QP solvers (polynomial complexity).
- There are variants that allow Q to be indefinite.

Table of Contents

Equality-Constrained Nonlinear Problems

$$\boxed{\begin{aligned} \min_{x \in \mathbb{R}^n} \; & f(x) \\ \text{s.t. } & c(x) = 0 \end{aligned}}$$

Equality-Constrained Nonlinear Problems

$$\boxed{\begin{aligned} \min_{x\in\mathbb{R}^n} \;& f(x) \\ \text{s.t. } & c(x) = 0 \end{aligned}} \quad \longrightarrow \quad \boxed{\begin{aligned} \nabla f(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \end{aligned}}$$

Equality-Constrained Nonlinear Problems

$$\boxed{\begin{aligned} \min_{x\in\mathbb{R}^n} \ & f(x) \\ \text{s.t. } & c(x) = 0 \end{aligned}} \quad \longrightarrow \quad \boxed{\begin{aligned} \nabla f(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \end{aligned}}$$

- System of nonlinear equations in $(x, \lambda) \in \mathbb{R}^n \times \mathbb{R}^m$

Equality-Constrained Nonlinear Problems

$$\boxed{\begin{aligned} \min_{x \in \mathbb{R}^n} \; & f(x) \\ \text{s.t. } & c(x) = 0 \end{aligned}} \quad \longrightarrow \quad \boxed{\begin{aligned} \nabla f(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \end{aligned}}$$

- System of nonlinear equations in $(x, \lambda) \in \mathbb{R}^n \times \mathbb{R}^m$
- Apply Newton's method: Fast local convergence!

Equality-Constrained Nonlinear Problems

$$\boxed{\begin{aligned} \min_{x \in \mathbb{R}^n} \ & f(x) \\ \text{s.t.} \ & c(x) = 0 \end{aligned}} \quad \longrightarrow \quad \boxed{\begin{aligned} \nabla f(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \end{aligned}}$$

- System of nonlinear equations in $(x, \lambda) \in \mathbb{R}^n \times \mathbb{R}^m$
- Apply Newton's method: Fast local convergence!
- Issues:
  - Guarantees only (fast) local convergence.

Equality-Constrained Nonlinear Problems

$$\boxed{\begin{aligned} \min_{x \in \mathbb{R}^n} \; & f(x) \\ \text{s.t.} \; & c(x) = 0 \end{aligned}} \quad \longrightarrow \quad \boxed{\begin{aligned} \nabla f(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \end{aligned}}$$

- System of nonlinear equations in $(x, \lambda) \in \mathbb{R}^n \times \mathbb{R}^m$
- Apply Newton's method: Fast local convergence!
- Issues:
  - Guarantees only (fast) local convergence.
  - We would like to find local minima and not just any kind of stationary point.

Equality-Constrained Nonlinear Problems

$$\boxed{\begin{aligned} \min_{x\in\mathbb{R}^n} \ & f(x) \\ \text{s.t.} \ & c(x) = 0 \end{aligned}} \quad \longrightarrow \quad \boxed{\begin{aligned} \nabla f(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \end{aligned}}$$

- System of nonlinear equations in $(x, \lambda) \in \mathbb{R}^n \times \mathbb{R}^m$
- Apply Newton's method: Fast local convergence!
- Issues:
  - Guarantees only (fast) local convergence.
  - We would like to find local minima and not just any kind of stationary point.
- Need:
  - Globalization scheme (for convergence from any starting point)
  - Mechanisms that encourage convergence to local minima

Newton's Method

$$\boxed{\begin{aligned} \nabla f(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \end{aligned}}$$

At iterate (x_k, λ_k) compute step p_k, p_k^λ from

Newton's Method

$$\boxed{\begin{aligned} \nabla f(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \end{aligned}}$$

At iterate (x_k, λ_k) compute step p_k, p_k^λ from

$$\boxed{\begin{bmatrix} H_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ p_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla f_k + \nabla c_k \lambda_k \\ c_k \end{pmatrix}}$$

$$\nabla f_k := \nabla f(x_k) \qquad \nabla c_k := \nabla c(x_k) \qquad c_k := c(x_k)$$

Newton's Method

$$\boxed{\begin{aligned} \nabla f(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \end{aligned}}$$

At iterate (x_k, λ_k) compute step p_k, p_k^λ from

$$\boxed{\begin{bmatrix} H_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ p_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla f_k + \nabla c_k \lambda_k \\ c_k \end{pmatrix}}$$

$$\nabla f_k := \nabla f(x_k) \qquad \nabla c_k := \nabla c(x_k) \qquad c_k := c(x_k)$$

$$\mathcal{L}(x, \lambda) := f(x) + \sum_{j=1}^{m} c_j(x)\lambda_j \qquad H_k := \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)$$

Newton's Method

$$\boxed{\begin{aligned} \nabla f(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \end{aligned}}$$

At iterate (x_k, λ_k) compute step p_k, p_k^λ from

$$\boxed{\begin{bmatrix} H_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ p_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla f_k + \nabla c_k \lambda_k \\ c_k \end{pmatrix}}$$

- Update iterate $(x_{k+1}, \lambda_{k+1}) = (x_k, \lambda_k) + (p_k, p_k^\lambda)$

$$\nabla f_k := \nabla f(x_k) \qquad \nabla c_k := \nabla c(x_k) \qquad c_k := c(x_k)$$

$$\mathcal{L}(x, \lambda) := f(x) + \sum_{j=1}^{m} c_j(x)\lambda_j \qquad H_k := \nabla_{xx}^2 \mathcal{L}(x_k, \lambda_k)$$

Sequential Quadratic Programming

$$\boxed{\begin{bmatrix} H_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ p_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla f_k + \nabla c_k \lambda_k \\ c_k \end{pmatrix}}$$

Sequential Quadratic Programming

$$\boxed{\begin{bmatrix} H_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \lambda_k + p_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla f_k \cancel{+ \nabla c_k \lambda_k} \\ c_k \end{pmatrix}}$$

Sequential Quadratic Programming

$$\boxed{\begin{bmatrix} H_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = -\begin{pmatrix} \nabla f_k \\ c_k \end{pmatrix}}$$

$$\tilde{\lambda}_{k+1} = \lambda_k + p_k^\lambda$$

Sequential Quadratic Programming

$$\boxed{\begin{bmatrix} H_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k \\ c_k \end{pmatrix}}$$

These are the optimality conditions of

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f_k^T p + f_k \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}}$$

with multipliers $\tilde{\lambda}_{k+1} = \lambda_k + p_k^\lambda$

Sequential Quadratic Programming

$$\boxed{\begin{bmatrix} H_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = -\begin{pmatrix} \nabla f_k \\ c_k \end{pmatrix}}$$

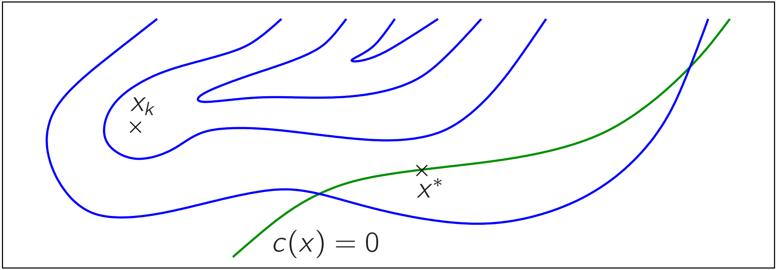
These are the optimality conditions of

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \; & \frac{1}{2} p^T H_k p + \nabla f_k^T p + f_k \\ \text{s.t.} \; & \nabla c_k^T p + c_k = 0 \end{aligned}}$$

with multipliers $\tilde{\lambda}_{k+1} = \lambda_k + p_k^\lambda$

- Newton step can be interpreted as solution of a local QP model!

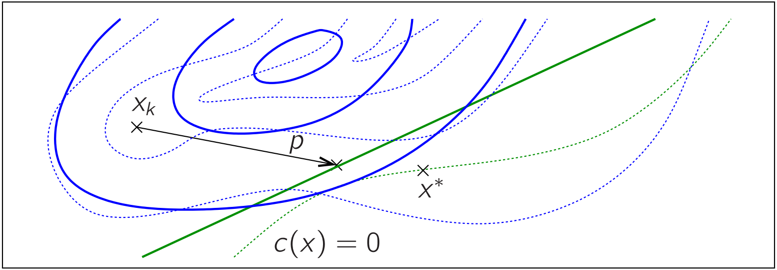
Local QP Model



Original Problem

$$\min_{x \in \mathbb{R}^n} f(x)$$
$$\text{s.t. } c(x) = 0$$

Local QP Model



Original Problem

$$\min_{x \in \mathbb{R}^n} \; f(x)$$
$$\text{s.t.} \; c(x) = 0$$

Local QP model (QP$_k$)

$$\min_{p \in \mathbb{R}^n} \; f_k + \nabla f_k^T p + \frac{1}{2} p^T H_k p$$
$$\text{s.t.} \; c_k + \nabla c_k^T p = 0$$

Exact Penalty Function

- Need tool to facilitate convergence from any starting point.

Exact Penalty Function

- Need tool to facilitate convergence from any starting point.
- Here, we have two (usually competing) goals:

Exact Penalty Function

- Need tool to facilitate convergence from any starting point.
- Here, we have two (usually competing) goals:

Optimality	Feasibility
$\boxed{\min f(x)}$	$\boxed{\min \|c(x)\|}$

Exact Penalty Function

- Need tool to facilitate convergence from any starting point.
- Here, we have two (usually competing) goals:

Optimality	Feasibility
$\min f(x)$	$\min \|c(x)\|$

- Combined in (non-differentiable) exact penalty function:

$$\phi_\rho(x) = f(x) + \rho \|c(x)\|_1 \qquad (\rho > 0)$$

Exact Penalty Function

- Need tool to facilitate convergence from any starting point.
- Here, we have two (usually competing) goals:

Optimality	Feasibility
$\boxed{\min f(x)}$	$\boxed{\min \|c(x)\|}$

- Combined in (non-differentiable) exact penalty function:

$$\phi_\rho(x) = f(x) + \rho\|c(x)\|_1 \qquad (\rho > 0)$$

Lemma

Suppose, x^* is a local minimizer of (NLP) with multipliers λ^* and LICQ holds. Then x^* is a local minimizer of ϕ_ρ if $\rho > \|\lambda^*\|_\infty$.

Exact Penalty Function

- Need tool to facilitate convergence from any starting point.
- Here, we have two (usually competing) goals:

Optimality	Feasibility
$\boxed{\min f(x)}$	$\boxed{\min \|c(x)\|}$

- Combined in (non-differentiable) exact penalty function:

$$\phi_\rho(x) = f(x) + \rho\|c(x)\|_1 \qquad (\rho > 0)$$

Lemma

Suppose, x^* is a local minimizer of (NLP) with multipliers λ^* and LICQ holds. Then x^* is a local minimizer of ϕ_ρ if $\rho > \|\lambda^*\|_\infty$.

- We can use decrease in ϕ_ρ as a measure of progress towards a local minimizer of (NLP).

Line Search

$$\boxed{\phi_\rho(x) = f(x) + \rho \|c(x)\|_1}$$

Line Search

$$\boxed{\phi_\rho(x) = f(x) + \rho \|c(x)\|_1}$$

- Backtracking line search: Try $\alpha_k \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ until

Line Search

$$\boxed{\phi_\rho(x) = f(x) + \rho\|c(x)\|_1}$$

- Backtracking line search: Try $\alpha_k \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ until

$$\phi_\rho(x_k + \alpha_k p_k) \leq \phi_\rho(x_k) + \eta\alpha_k D\phi_\rho(x_k; p_k). \qquad (\eta \in (0,1))$$

Line Search

$$\boxed{\phi_\rho(x) = f(x) + \rho\|c(x)\|_1}$$

- Backtracking line search: Try $\alpha_k \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ until

$$\phi_\rho(x_k + \alpha_k p_k) \le \phi_\rho(x_k) + \eta\alpha_k D\phi_\rho(x_k; p_k). \qquad (\eta \in (0,1))$$

$D\phi_\rho(x_k; p_k)$: Directional derivative of ϕ_ρ at x_k in direction p_k.

Line Search

$$\boxed{\phi_\rho(x) = f(x) + \rho\|c(x)\|_1}$$

- Backtracking line search: Try $\alpha_k \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ until

$$\phi_\rho(x_k + \alpha_k p_k) \le \phi_\rho(x_k) + \eta\alpha_k D\phi_\rho(x_k; p_k). \qquad (\eta \in (0,1))$$

$D\phi_\rho(x_k; p_k)$: Directional derivative of ϕ_ρ at x_k in direction p_k.

Lemma

Let p_k be an optimal solution of (QP_k). Then

$$D\phi_\rho(x_k; p_k) \le -p_k^T H_k p_k - (\rho - \|\tilde{\lambda}_{k+1}\|_\infty)\|c_k\|_1.$$

Line Search

$$\boxed{\phi_\rho(x) = f(x) + \rho\|c(x)\|_1}$$

- Backtracking line search: Try $\alpha_k \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ until

$$\phi_\rho(x_k + \alpha_k p_k) \leq \phi_\rho(x_k) + \eta\alpha_k D\phi_\rho(x_k; p_k). \qquad (\eta \in (0,1))$$

$D\phi_\rho(x_k; p_k)$: Directional derivative of ϕ_ρ at x_k in direction p_k.

Lemma

Let p_k be an optimal solution of (QP_k). Then

$$D\phi_\rho(x_k; p_k) \leq -p_k^T H_k p_k - (\rho - \|\tilde{\lambda}_{k+1}\|_\infty)\|c_k\|_1.$$

- So, p_k is a descent direction for ϕ_ρ if

Line Search

$$\boxed{\phi_\rho(x) = f(x) + \rho\|c(x)\|_1}$$

- Backtracking line search: Try $\alpha_k \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ until

$$\phi_\rho(x_k + \alpha_k p_k) \le \phi_\rho(x_k) + \eta\alpha_k D\phi_\rho(x_k; p_k). \qquad (\eta \in (0,1))$$

$D\phi_\rho(x_k; p_k)$: Directional derivative of ϕ_ρ at x_k in direction p_k.

Lemma

Let p_k be an optimal solution of (QP_k). Then

$$D\phi_\rho(x_k; p_k) \le -p_k^T H_k p_k - (\rho - \|\tilde{\lambda}_{k+1}\|_\infty)\|c_k\|_1.$$

- So, p_k is a descent direction for ϕ_ρ if $H_k \succ 0$ and $\rho > \|\tilde{\lambda}_{k+1}\|_\infty$.

Basic SQP Algorithm

1. Choose x_1, λ_1, $\rho_0 > 0$. Set $k \leftarrow 1$.

Basic SQP Algorithm

1. Choose x_1, λ_1, $\rho_0 > 0$. Set $k \leftarrow 1$.
2. Solve (QP_k) to get p_k and $\tilde{\lambda}_{k+1}$.

Basic SQP Algorithm

1. Choose x_1, λ_1, $\rho_0 > 0$. Set $k \leftarrow 1$.
2. Solve (QP_k) to get p_k and $\tilde{\lambda}_{k+1}$.
3. Update penalty parameter: ($\beta > 0$)

$$\rho_k = \begin{cases} \rho_{k-1} & \text{if } \rho_{k-1} \geq \|\tilde{\lambda}_{k+1}\|_\infty + \beta \\ \|\tilde{\lambda}_{k+1}\|_\infty + 2\beta & \text{otherwise.} \end{cases}$$

Basic SQP Algorithm

1. Choose x_1, λ_1, $\rho_0 > 0$. Set $k \leftarrow 1$.
2. Solve (QP_k) to get p_k and $\tilde{\lambda}_{k+1}$.
3. Update penalty parameter: $(\beta > 0)$

$$\rho_k = \begin{cases} \rho_{k-1} & \text{if } \rho_{k-1} \geq \|\tilde{\lambda}_{k+1}\|_\infty + \beta \\ \|\tilde{\lambda}_{k+1}\|_\infty + 2\beta & \text{otherwise.} \end{cases}$$

4. Perform backtracking line search:
   Find largest $\alpha_k \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ with

$$\phi_{\rho_k}(x_k + \alpha_k p_k) \leq \phi_\rho(x_k) + \eta \alpha_k D\phi_{\rho_k}(x_k; p_k).$$

Basic SQP Algorithm

1. Choose x_1, λ_1, $\rho_0 > 0$. Set $k \leftarrow 1$.
2. Solve (QP$_k$) to get p_k and $\tilde{\lambda}_{k+1}$.
3. Update penalty parameter: $(\beta > 0)$

$$\rho_k = \begin{cases} \rho_{k-1} & \text{if } \rho_{k-1} \geq \|\tilde{\lambda}_{k+1}\|_\infty + \beta \\ \|\tilde{\lambda}_{k+1}\|_\infty + 2\beta & \text{otherwise.} \end{cases}$$

4. Perform backtracking line search:
   Find largest $\alpha_k \in \{1, \frac{1}{2}, \frac{1}{4}, \dots\}$ with

$$\phi_{\rho_k}(x_k + \alpha_k p_k) \leq \phi_\rho(x_k) + \eta \alpha_k D\phi_{\rho_k}(x_k; p_k).$$

5. Update iterate $x_{k+1} = x_k + \alpha_k p_k$ and $\lambda_{k+1} = \tilde{\lambda}_{k+1}$.

Basic SQP Algorithm

1. Choose x_1, λ_1, $\rho_0 > 0$. Set $k \leftarrow 1$.
2. Solve (QP_k) to get p_k and $\tilde{\lambda}_{k+1}$.
3. Update penalty parameter: $(\beta > 0)$

$$\rho_k = \begin{cases} \rho_{k-1} & \text{if } \rho_{k-1} \geq \|\tilde{\lambda}_{k+1}\|_\infty + \beta \\ \|\tilde{\lambda}_{k+1}\|_\infty + 2\beta & \text{otherwise.} \end{cases}$$

4. Perform backtracking line search:
   Find largest $\alpha_k \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ with

$$\phi_{\rho_k}(x_k + \alpha_k p_k) \leq \phi_\rho(x_k) + \eta \alpha_k D\phi_{\rho_k}(x_k; p_k).$$

5. Update iterate $x_{k+1} = x_k + \alpha_k p_k$ and $\lambda_{k+1} = \tilde{\lambda}_{k+1}$.
6. Set $k \leftarrow k + 1$ and to go Step 2.

Convergence Result for Basic SQP Algorithm

Assumptions

- *f and c are twice continuously differentiable.*

Convergence Result for Basic SQP Algorithm

Assumptions

- *f and c are twice continuously differentiable.*
- *The matrices H_k are bounded and uniformly positive definite.*

Convergence Result for Basic SQP Algorithm

Assumptions

- *f and c are twice continuously differentiable.*
- *The matrices H_k are bounded and uniformly positive definite.*
- *The smallest singular value of ∇c_k is uniformly bounded away from zero.*

Convergence Result for Basic SQP Algorithm

Assumptions

- *f and c are twice continuously differentiable.*
- *The matrices H_k are bounded and uniformly positive definite.*
- *The smallest singular value of ∇c_k is uniformly bounded away from zero.*

Theorem

Under these assumptions, we have

$$\lim_{k\to\infty} \left\| \begin{pmatrix} \nabla f_k + \nabla c_k \tilde{\lambda}_{k+1} \\ c_k \end{pmatrix} \right\| = 0.$$

Convergence Result for Basic SQP Algorithm

Assumptions

- *f and c are twice continuously differentiable.*
- *The matrices H_k are bounded and uniformly positive definite.*
- *The smallest singular value of ∇c_k is uniformly bounded away from zero.*

Theorem

Under these assumptions, we have

$$\lim_{k\to\infty} \left\| \begin{pmatrix} \nabla f_k + \nabla c_k \tilde{\lambda}_{k+1} \\ c_k \end{pmatrix} \right\| = 0.$$

In other words, each limit point of $\{x_k\}$ is a stationary point for (NLP).

Choice of Hessian H_k

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n} \ & \frac{1}{2}p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\mathrm{QP}_k)$$

- For fast local convergence, want to choose $H_k = \nabla^2_{xx}\mathcal{L}_k$.

Choice of Hessian H_k

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\mathrm{QP}_k)$$

- For fast local convergence, want to choose $H_k = \nabla^2_{xx} \mathcal{L}_k$.
- $\nabla^2_{xx} \mathcal{L}_k$ is positive definite, if f and c are convex.

Choice of Hessian H_k

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\mathrm{QP}_k)$$

- For fast local convergence, want to choose $H_k = \nabla^2_{xx} \mathcal{L}_k$.
- $\nabla^2_{xx} \mathcal{L}_k$ is positive definite, if f and c are convex.
- In general, $\nabla^2_{xx} \mathcal{L}_k$ might be indefinite.

Choice of Hessian H_k

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\mathrm{QP}_k)$$

- For fast local convergence, want to choose $H_k = \nabla^2_{xx} \mathcal{L}_k$.
- $\nabla^2_{xx} \mathcal{L}_k$ is positive definite, if f and c are convex.
- In general, $\nabla^2_{xx} \mathcal{L}_k$ might be indefinite.
  - (QP_k) might be unbounded.

Choice of Hessian H_k

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\mathrm{QP}_k)$$

- For fast local convergence, want to choose $H_k = \nabla^2_{xx} \mathcal{L}_k$.
- $\nabla^2_{xx} \mathcal{L}_k$ is positive definite, if f and c are convex.
- In general, $\nabla^2_{xx} \mathcal{L}_k$ might be indefinite.
  - (QP_k) might be unbounded.
  - p_k might not be a descent direction for ϕ_ρ.

Choice of Hessian H_k

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n}\ & \frac{1}{2}p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\text{QP}_k)$$

- For fast local convergence, want to choose $H_k = \nabla^2_{xx}\mathcal{L}_k$.
- $\nabla^2_{xx}\mathcal{L}_k$ is positive definite, if f and c are convex.
- In general, $\nabla^2_{xx}\mathcal{L}_k$ might be indefinite.
  - (QP_k) might be unbounded.
  - p_k might not be a descent direction for ϕ_ρ.
  - Would like $Z_k^T H_k Z_k \succ 0$ (Z_k null-space matrix for ∇c_k^T).

Choice of Hessian H_k

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\mathrm{QP}_k)$$

- For fast local convergence, want to choose $H_k = \nabla^2_{xx} \mathcal{L}_k$.
- $\nabla^2_{xx} \mathcal{L}_k$ is positive definite, if f and c are convex.
- In general, $\nabla^2_{xx} \mathcal{L}_k$ might be indefinite.
  - (QP_k) might be unbounded.
  - p_k might not be a descent direction for ϕ_ρ.
  - Would like $Z_k^T H_k Z_k \succ 0$ (Z_k null-space matrix for ∇c_k^T).
- $H_k =$ BFGS approximation of $\nabla^2_{xx} \mathcal{L}(x_k, \lambda_k)$.

Choice of Hessian H_k

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\mathrm{QP}_k)$$

- For fast local convergence, want to choose $H_k = \nabla^2_{xx} \mathcal{L}_k$.
- $\nabla^2_{xx} \mathcal{L}_k$ is positive definite, if f and c are convex.
- In general, $\nabla^2_{xx} \mathcal{L}_k$ might be indefinite.
  - (QP_k) might be unbounded.
  - p_k might not be a descent direction for ϕ_ρ.
  - Would like $Z_k^T H_k Z_k \succ 0$ (Z_k null-space matrix for ∇c_k^T).
- $H_k =$ BFGS approximation of $\nabla^2_{xx} \mathcal{L}(x_k, \lambda_k)$.
  - Potentially slow local convergence, since $\nabla^2_{xx} \mathcal{L}(x^*, \lambda^*)$ may be indefinite.

Regularization

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n} \ & \frac{1}{2}p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}}$$

Recall optimality conditions

$$\underbrace{\begin{bmatrix} H_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix}}_{=:K_k} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k \\ c_k \end{pmatrix}$$

Regularization

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n}\ & \frac{1}{2}p^T(H_k+\gamma I)p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}}$$

Recall optimality conditions

$$\underbrace{\begin{bmatrix} H_k+\gamma I & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix}}_{=:K_k} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = -\begin{pmatrix} \nabla f_k \\ c_k \end{pmatrix}$$

- Choose $\gamma \geq 0$

Regularization

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n} \ & \frac{1}{2}p^T(H_k+\gamma I)p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}}$$

Recall optimality conditions

$$\underbrace{\begin{bmatrix} H_k+\gamma I & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix}}_{=:K_k} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = -\begin{pmatrix} \nabla f_k \\ c_k \end{pmatrix}$$

- Choose $\gamma \geq 0$ so that K_k has inertia $(n, m, 0)$.
- Then $Z_k^T H_k Z_k$ is positive definite.

Regularization

$$\boxed{\begin{aligned}&\min_{p\in\mathbb{R}^n}\ \frac{1}{2}p^T(H_k+\gamma I)p+\nabla f_k^T p\\ &\text{s.t. } \nabla c_k^T p + c_k = 0\end{aligned}}$$

Recall optimality conditions

$$\underbrace{\begin{bmatrix} H_k+\gamma I & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix}}_{=:K_k} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = -\begin{pmatrix} \nabla f_k \\ c_k \end{pmatrix}$$

- Choose $\gamma \geq 0$ so that K_k has inertia $(n, m, 0)$.
    - E.g.: Trial and error, computing inertia via factorization
- Then $Z_k^T H_k Z_k$ is positive definite.

Regularization

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n}\ & \frac{1}{2}p^T(H_k+\gamma I)p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}}$$

Recall optimality conditions

$$\underbrace{\begin{bmatrix} H_k+\gamma I & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix}}_{=:K_k} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = -\begin{pmatrix} \nabla f_k \\ c_k \end{pmatrix}$$

- Choose $\gamma \geq 0$ so that K_k has inertia $(n, m, 0)$.
  - E.g.: Trial and error, computing inertia via factorization
- Then $Z_k^T H_k Z_k$ is positive definite.
- No regularization necessary close to second-order sufficient solution.

Step Decomposition Revisited

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\mathrm{QP}_k)$$

Decomposition $p_k = Y_k p_{Y,k} + Z_k p_{Z,k}$

Step Decomposition Revisited

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\mathrm{QP}_k)$$

Decomposition $p_k = Y_k p_{Y,k} + Z_k p_{Z,k}$

- Range space step

$$p_{Y,k} = -[\nabla c_k^T Y_k]^{-1} \nabla f_k$$

Step Decomposition Revisited

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n} \ & \frac{1}{2}p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\mathrm{QP}_k)$$

Decomposition $p_k = Y_k p_{Y,k} + Z_k p_{Z,k}$

- Range space step

$$p_{Y,k} = -[\nabla c_k^T Y_k]^{-1} \nabla f_k$$

- Reduced space QP

$$\min_{p_Z} \ \frac{1}{2} p_Z^T [Z_k^T H_k Z_k] p_Z + (\nabla f_k + H_k Y_k p_{Y,k})^T Z_k p_Z$$

Step Decomposition Revisited

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\mathrm{QP}_k)$$

Decomposition $p_k = Y_k p_{Y,k} + Z_k p_{Z,k}$

- Range space step

$$p_{Y,k} = -[\nabla c_k^T Y_k]^{-1} \nabla f_k$$

- Reduced space QP

$$\min_{p_Z} \ \frac{1}{2} p_Z^T [Z_k^T H_k Z_k] p_Z + (\nabla f_k + H_k Y_k p_{Y,k})^T Z_k p_Z$$

- Make sure $Z_k^T H_k Z_k$ is positive definite

Step Decomposition Revisited

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n} \ & \frac{1}{2}p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \end{aligned}} \qquad (\text{QP}_k)$$

Decomposition $p_k = Y_k p_{Y,k} + Z_k p_{Z,k}$

- Range space step

$$p_{Y,k} = -[\nabla c_k^T Y_k]^{-1} \nabla f_k$$

- Reduced space QP

$$\min_{p_Z} \ \frac{1}{2} p_Z^T [Z_k^T H_k Z_k] p_Z + (\nabla f_k + H_k Y_k p_{Y,k})^T Z_k p_Z$$

- Make sure $Z_k^T H_k Z_k$ is positive definite (e.g., BFGS)

Trust-Region SQP Method

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n}\ & \frac{1}{2}p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0, \quad \|p\| \le \Delta_k \end{aligned}} \qquad (\mathrm{QP}_k)$$

- Trust-region radius Δ_k, updated throughout iterations

Trust-Region SQP Method

$$
\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n}\ & \frac{1}{2}p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0, \quad \|p\| \le \Delta_k \end{aligned}} \qquad (\mathrm{QP}_k)
$$

- Trust-region radius Δ_k, updated throughout iterations
- No positive-definiteness requirements for H_k

Trust-Region SQP Method

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0, \quad \|p\| \le \Delta_k \end{aligned}} \qquad (\mathrm{QP}_k)$$

- Trust-region radius Δ_k, updated throughout iterations
- No positive-definiteness requirements for H_k
- Step p_k is accepted if $\frac{\mathrm{pred}_k}{\mathrm{ared}_k} \ge \eta$ with $(\eta \in (0,1))$

$$\mathrm{pred}_k = m_k(0) - m_k(p_k), \qquad \mathrm{ared}_k = \phi_\rho(x_k) - \phi_\rho(x_k + p_k)$$

Trust-Region SQP Method

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n} \ & \frac{1}{2}p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0, \quad \|p\| \le \Delta_k \end{aligned}} \qquad (\mathrm{QP}_k)$$

- Trust-region radius Δ_k, updated throughout iterations
- No positive-definiteness requirements for H_k
- Step p_k is accepted if $\frac{\text{pred}_k}{\text{ared}_k} \ge \eta$ with $(\eta \in (0,1))$

$$\text{pred}_k = m_k(0) - m_k(p_k), \qquad \text{ared}_k = \phi_\rho(x_k) - \phi_\rho(x_k + p_k)$$

- Piece-wise quadratic model of $\phi_\rho(x) = f(x) + \rho\|c(x)\|_1$:

$$m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2}p^T H_k p + \rho\|c_k + \nabla c_k^T p\|_1$$

Trust-Region SQP Method

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n} \ & \frac{1}{2}p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0, \quad \|p\| \le \Delta_k \end{aligned}} \qquad (\mathrm{QP}_k)$$

- Trust-region radius Δ_k, updated throughout iterations
- No positive-definiteness requirements for H_k
- Step p_k is accepted if $\frac{\text{pred}_k}{\text{ared}_k} \ge \eta$ with $\quad (\eta \in (0,1))$

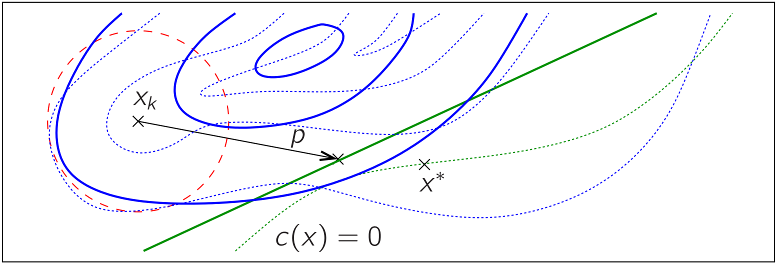
$$\text{pred}_k = m_k(0) - m_k(p_k), \qquad \text{ared}_k = \phi_\rho(x_k) - \phi_\rho(x_k + p_k)$$

- Piece-wise quadratic model of $\phi_\rho(x) = f(x) + \rho\|c(x)\|_1$:

$$m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2}p^T H_k p + \rho\|c_k + \nabla c_k^T p\|_1$$

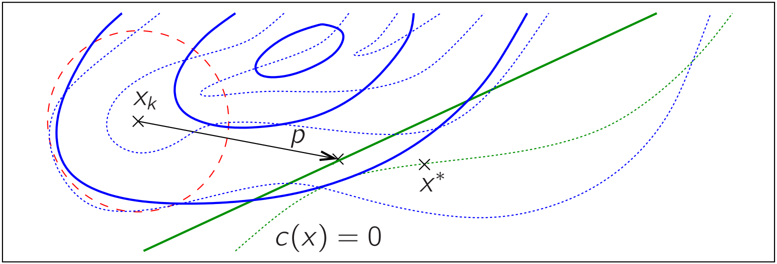
- Otherwise, decrease Δ_k

Inconsistent QPs



- If x_k is not feasible and Δ_k small, (QP$_k$) might not be feasible

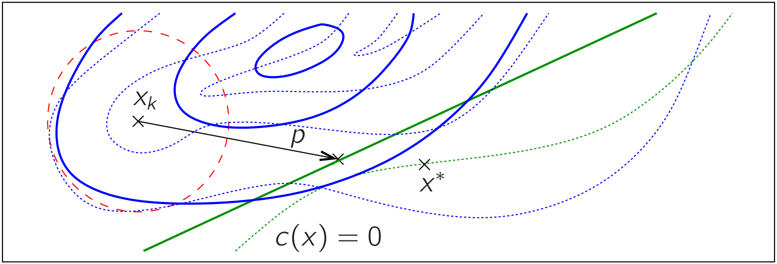
Inconsistent QPs



- If x_k is not feasible and Δ_k small, (QP_k) might not be feasible
- One remedy: Penalize constraint violation

$$\begin{aligned} \min_{p \in \mathbb{R}^n} \quad & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_k^T p + c_k = 0 \\ & \|p\| \leq \Delta_k \end{aligned}$$

Inconsistent QPs



- If x_k is not feasible and Δ_k small, (QP$_k$) might not be feasible
- One remedy: Penalize constraint violation

$$\begin{aligned} \min_{p \in \mathbb{R}^n; t,s \in \mathbb{R}^m} \quad & \frac{1}{2} p^T H_k p + \nabla f_k^T p + \rho \sum_{j=1}^{m} (s_j + t_j) \\ \text{s.t. } & \nabla c_k^T p + c_k = s - t \\ & \|p\| \le \Delta_k, \quad s, t \ge 0 \end{aligned}$$

Fletcher's Sℓ_1QP

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n;t,s\in\mathbb{R}^m} \ & \frac{1}{2}p^T H_k p + \nabla f_k^T p + \rho\sum_{j=1}^m (s_j + t_j) \\ \text{s.t. } & \nabla c_k^T p + c_k = s - t \\ & \|p\| \le \Delta_k, \quad s,t \ge 0 \end{aligned}}$$

Fletcher's Sℓ_1QP

$$
\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n;t,s\in\mathbb{R}^m} \ & \frac{1}{2}p^T H_k p + \nabla f_k^T p + \rho \sum_{j=1}^m (s_j + t_j) \\ \text{s.t. } & \nabla c_k^T p + c_k = s - t \\ & \|p\| \le \Delta_k, \quad s,t \ge 0 \end{aligned}}
$$

is equivalent to

$$
\boxed{\min_{p\in\mathbb{R}^n} \ m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2}p^T H_k p + \rho \|c_k + \nabla c_k^T p\|_1}
$$

Fletcher's Sℓ_1QP

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n;t,s\in\mathbb{R}^m} \quad & \frac{1}{2}p^T H_k p + \nabla f_k^T p + \rho \sum_{j=1}^m (s_j + t_j) \\ \text{s.t.} \quad & \nabla c_k^T p + c_k = s - t \\ & \|p\| \le \Delta_k, \quad s,t \ge 0 \end{aligned}}$$

is equivalent to

$$\boxed{\min_{p\in\mathbb{R}^n} \; m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2}p^T H_k p + \rho \|c_k + \nabla c_k^T p\|_1}$$

- Natural algorithm for minimizing $\phi_\rho(x)$.

Fletcher's Sℓ_1QP

$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n;t,s\in\mathbb{R}^m} \; & \frac{1}{2}p^T H_k p + \nabla f_k^T p + \rho \sum_{j=1}^m (s_j + t_j) \\ \text{s.t. } & \nabla c_k^T p + c_k = s - t \\ & \|p\| \le \Delta_k, \quad s, t \ge 0 \end{aligned}}$$

is equivalent to

$$\boxed{\min_{p\in\mathbb{R}^n} \; m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2}p^T H_k p + \rho \|c_k + \nabla c_k^T p\|_1}$$

- Natural algorithm for minimizing $\phi_\rho(x)$.
- Difficulty: Selecting sufficiently large value of ρ.

Fletcher's Sℓ_1QP

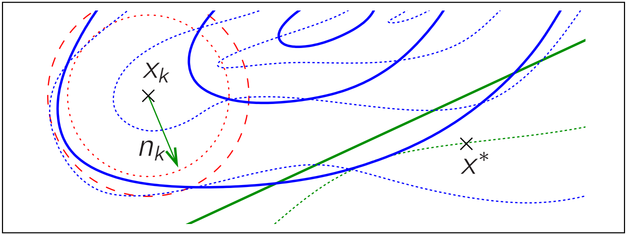
$$\boxed{\begin{aligned} \min_{p\in\mathbb{R}^n;t,s\in\mathbb{R}^m} \quad & \frac{1}{2}p^T H_k p + \nabla f_k^T p + \rho \sum_{j=1}^m (s_j + t_j) \\ \text{s.t.} \quad & \nabla c_k^T p + c_k = s - t \\ & \|p\| \le \Delta_k, \quad s, t \ge 0 \end{aligned}}$$

is equivalent to

$$\boxed{\min_{p\in\mathbb{R}^n} \; m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2}p^T H_k p + \rho\|c_k + \nabla c_k^T p\|_1}$$

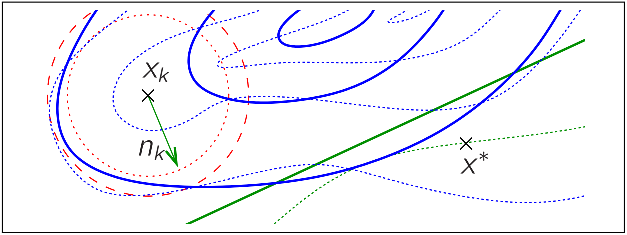
- Natural algorithm for minimizing $\phi_\rho(x)$.
- Difficulty: Selecting sufficiently large value of ρ.
  - This motivated the invention of *filter methods*.

Byrd-Omojokun Trust-Region Algorithm



- Decompose step $p_k = n_k + t_k$

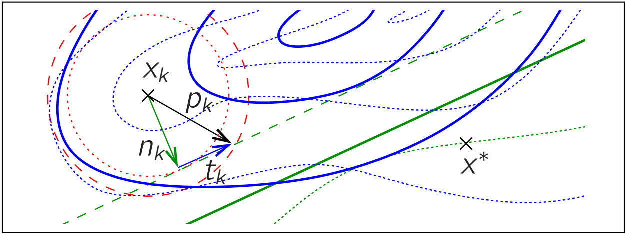
Byrd-Omojokun Trust-Region Algorithm



- Decompose step $p_k = n_k + t_k$

Normal component n_k
towards feasibility

Byrd-Omojokun Trust-Region Algorithm

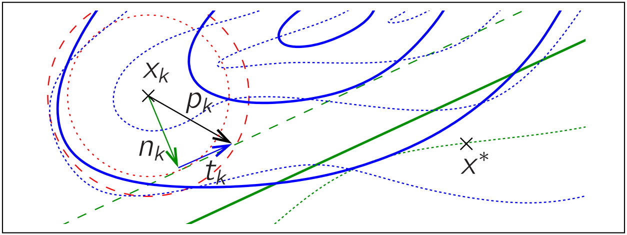


- Decompose step $p_k = n_k + t_k$

Normal component n_k
towards feasibility

Tangential component t_k
towards optimality

Byrd-Omojokun Trust-Region Algorithm



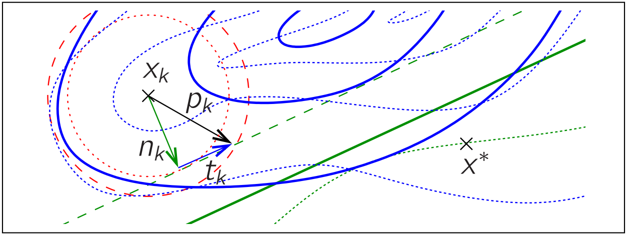
- Decompose step $p_k = n_k + t_k$

Normal component n_k
towards feasibility

$$\boxed{\begin{array}{c} \min\limits_{n} \; \|\nabla c_k^T n + c_k\|_2^2 \\ \text{s.t. } \|n\|_2 \leq 0.8\Delta_k \end{array}}$$

Tangential component t_k
towards optimality

Byrd-Omojokun Trust-Region Algorithm



- Decompose step $p_k = n_k + t_k$

Normal component n_k
towards feasibility

$$\min_n \ \|\nabla c_k^T n + c_k\|_2^2$$
$$\text{s.t. } \|n\|_2 \le 0.8\Delta_k$$

Tangential component t_k
towards optimality

$$\min_t \ \frac{1}{2}(n_k+t)^T H_k (n_k+t) + \nabla f_k^T (n_k+t)$$
$$\text{s.t. } \nabla c_k^T t = 0$$
$$\|t\|_2 \le \sqrt{\Delta_k^2 - \|n_k\|_2^2}$$

Byrd-Omojokun Trust-Region Algorithm

Normal component n_k
towards feasibility

$$\boxed{\begin{aligned} &\min_n \; \|\nabla c_k^T n + c_k\|_2^2 \\ &\text{s.t. } \|n\|_2 \le 0.8\Delta_k \end{aligned}}$$

Tangential component t_k
towards optimality

$$\boxed{\begin{aligned} &\min_t \; \frac{1}{2}(n_k+t)^T H_k (n_k+t) + \nabla f_k^T (n_k+t) \\ &\text{s.t. } \nabla c_k^T t = 0 \\ &\qquad \|t\|_2 \le \sqrt{\Delta_k^2 - \|n_k\|_2^2} \end{aligned}}$$

- Subproblems can be solved inexactly

Byrd-Omojokun Trust-Region Algorithm

Normal component n_k
towards feasibility

$$\boxed{\begin{aligned} &\min_n \; \|\nabla c_k^T n + c_k\|_2^2 \\ &\text{s.t. } \|n\|_2 \le 0.8\Delta_k \end{aligned}}$$

Tangential component t_k
towards optimality

$$\boxed{\begin{aligned} &\min_t \; \frac{1}{2}(n_k+t)^T H_k (n_k+t) + \nabla f_k^T (n_k+t) \\ &\text{s.t. } \nabla c_k^T t = 0 \\ &\qquad \|t\|_2 \le \sqrt{\Delta_k^2 - \|n_k\|_2^2} \end{aligned}}$$

- Subproblems can be solved inexactly
  - Normal problem: Dogleg method.

Byrd-Omojokun Trust-Region Algorithm

Normal component n_k
towards feasibility

$$\boxed{\begin{aligned} &\min_n \ \|\nabla c_k^T n + c_k\|_2^2 \\ &\text{s.t. } \|n\|_2 \le 0.8\Delta_k \end{aligned}}$$

Tangential component t_k
towards optimality

$$\boxed{\begin{aligned} &\min_t \ \frac{1}{2}(n_k+t)^T H_k(n_k+t) + \nabla f_k^T(n_k+t) \\ &\text{s.t. } \nabla c_k^T t = 0 \\ &\qquad \|t\|_2 \le \sqrt{\Delta_k^2 - \|n_k\|_2^2} \end{aligned}}$$

- Subproblems can be solved inexactly
  - Normal problem: Dogleg method.
  - Tangential problem: Conjugate-gradients method in null space.

Byrd-Omojokun Trust-Region Algorithm

Normal component n_k
towards feasibility

$$\boxed{\begin{aligned} &\min_n \ \|\nabla c_k^T n + c_k\|_2^2 \\ &\text{s.t. } \|n\|_2 \le 0.8\Delta_k \end{aligned}}$$

Tangential component t_k
towards optimality

$$\boxed{\begin{aligned} &\min_t \ \frac{1}{2}(n_k+t)^T H_k (n_k+t) + \nabla f_k^T (n_k+t) \\ &\text{s.t. } \nabla c_k^T t = 0 \\ &\qquad \|t\|_2 \le \sqrt{\Delta_k^2 - \|n_k\|_2^2} \end{aligned}}$$

- Subproblems can be solved inexactly
  - Normal problem: Dogleg method.
  - Tangential problem: Conjugate-gradients method in null space.
- ℓ_2-norm penalty function $\phi_\rho(x) = f(x) + \rho\|c(x)\|_2$.

Byrd-Omojokun Trust-Region Algorithm

Normal component n_k
towards feasibility

$$\begin{aligned} \min_n \; & \|\nabla c_k^T n + c_k\|_2^2 \\ \text{s.t. } & \|n\|_2 \le 0.8\Delta_k \end{aligned}$$

Tangential component t_k
towards optimality

$$\begin{aligned} \min_t \; & \frac{1}{2}(n_k+t)^T H_k (n_k+t) + \nabla f_k^T (n_k+t) \\ \text{s.t. } & \nabla c_k^T t = 0 \\ & \|t\|_2 \le \sqrt{\Delta_k^2 - \|n_k\|_2^2} \end{aligned}$$

- Subproblems can be solved inexactly
  - Normal problem: Dogleg method.
  - Tangential problem: Conjugate-gradients method in null space.
- ℓ_2-norm penalty function $\phi_\rho(x) = f(x) + \rho\|c(x)\|_2$.
- Strong convergence result:

Byrd-Omojokun Trust-Region Algorithm

Normal component n_k towards feasibility

$$\begin{aligned} \min_n \ & \|\nabla c_k^T n + c_k\|_2^2 \\ \text{s.t. } & \|n\|_2 \leq 0.8\Delta_k \end{aligned}$$

Tangential component t_k towards optimality

$$\begin{aligned} \min_t \ & \frac{1}{2}(n_k+t)^T H_k (n_k+t) + \nabla f_k^T (n_k+t) \\ \text{s.t. } & \nabla c_k^T t = 0 \\ & \|t\|_2 \leq \sqrt{\Delta_k^2 - \|n_k\|_2^2} \end{aligned}$$

- Subproblems can be solved inexactly
  - Normal problem: Dogleg method.
  - Tangential problem: Conjugate-gradients method in null space.
- ℓ_2-norm penalty function $\phi_\rho(x) = f(x) + \rho\|c(x)\|_2$.
- Strong convergence result:
  - If (NLP) is infeasible, limit points of $\{x_k\}$ are stationary points for infeasibility minimization problem $\min_x \|c(x)\|_2^2$.

Maratos Effect

- Even arbitrarily close to solution, full step $\alpha = 1$ might be rejected because the non-smooth merit function ϕ_ρ increases.
- Degrades fast local convergence.
- Remedies: Second-order correction steps or "watchdog" method.

Table of Contents

SQP For Inequality-Constrained Nonlinear Problems

$$\boxed{\begin{aligned} \min_{x \in \mathbb{R}^n} \ & f(x) \\ \text{s.t. } & c_E(x) = 0 \end{aligned}}$$

Compute p_k from local QP model

$$\boxed{\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f(x_k)^T p \\ \text{s.t. } & \nabla c_E(x_k)^T p + c_E(x_k) = 0 \end{aligned}} \qquad (\mathrm{QP}_k)$$

SQP For Inequality-Constrained Nonlinear Problems

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & f(x) \\ \text{s.t.} \ & c_E(x) = 0 \\ & c_I(x) \leq 0 \end{aligned}$$

Compute p_k from local QP model

$$\begin{aligned} \min_{p \in \mathbb{R}^n} \ & \frac{1}{2} p^T H_k p + \nabla f(x_k)^T p \\ \text{s.t.} \ & \nabla c_E(x_k)^T p + c_E(x_k) = 0 \end{aligned} \qquad (\text{QP}_k)$$

SQP For Inequality-Constrained Nonlinear Problems

$$\begin{aligned}\min_{x\in\mathbb{R}^n} \quad & f(x)\\ \text{s.t.} \quad & c_E(x) = 0\\ & c_I(x) \le 0\end{aligned}$$

Compute p_k from local QP model

$$\begin{aligned}\min_{p\in\mathbb{R}^n} \quad & \frac{1}{2}p^T H_k p + \nabla f(x_k)^T p\\ \text{s.t.} \quad & \nabla c_E(x_k)^T p + c_E(x_k) = 0\\ & \nabla c_I(x_k)^T p + c_I(x_k) \le 0\end{aligned} \qquad (\mathrm{QP}_k)$$

Local Behavior

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \;& f(x) \\ \text{s.t. } & c_i(x) = 0 \quad i \in \mathcal{E} \\ & c_i(x) \leq 0 \quad i \in \mathcal{I} \end{aligned}$$

Local Behavior

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \ & f(x) \\ \text{s.t. } & c_i(x) = 0 \quad i \in \mathcal{E} \\ & c_i(x) \le 0 \quad i \in \mathcal{A}_*^{\text{NLP}} \\ & c_i(x) \le 0 \quad i \in \overline{\mathcal{A}}_*^{\text{NLP}} \end{aligned}$$

$$\mathcal{A}_*^{\text{NLP}} = \{i \in \mathcal{I} : c_i(x^*) = 0\} \qquad \overline{\mathcal{A}}_*^{\text{NLP}} = \{i \in \mathcal{I} : c_i(x^*) < 0\}$$

Local Behavior

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & f(x) \\ \text{s.t.} \ & c_i(x) = 0 \quad i \in \mathcal{E} \\ & c_i(x) = 0 \quad i \in \mathcal{A}_*^{\text{NLP}} \\ & \cancel{c_i(x) \leq 0} \quad i \in \overline{\mathcal{A}}_*^{\text{NLP}} \end{aligned}$$

$$\mathcal{A}_*^{\text{NLP}} = \{i \in \mathcal{I} : c_i(x^*) = 0\} \qquad \overline{\mathcal{A}}_*^{\text{NLP}} = \{i \in \mathcal{I} : c_i(x^*) < 0\}$$

Local Behavior

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c_i(x) = 0 \qquad i \in \mathcal{E} \\ & c_i(x) = 0 \qquad i \in \mathcal{A}_*^{\text{NLP}} \\ & \cancel{c_i(x) \le 0} \qquad i \in \overline{\mathcal{A}}_*^{\text{NLP}} \end{aligned}$$

$$\begin{aligned} \min_{p\in\mathbb{R}^n} \quad & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t.} \quad & \nabla c_{k,i}^T p + c_{k,i} = 0 \qquad i \in \mathcal{E} \\ & \nabla c_{k,i}^T p + c_{k,i} = 0 \qquad i \in \mathcal{A}_*^{\text{QP}_k} \\ & \cancel{\nabla c_{k,i}^T p + c_{k,i} \le 0} \qquad i \in \overline{\mathcal{A}}_*^{\text{QP}_k} \end{aligned}$$

$$\mathcal{A}_*^{\text{NLP}} = \{ i \in \mathcal{I} : c_i(x^*) = 0 \} \qquad \overline{\mathcal{A}}_*^{\text{NLP}} = \{ i \in \mathcal{I} : c_i(x^*) < 0 \}$$

Local Behavior

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c_i(x) = 0 \qquad i \in \mathcal{E} \\ & c_i(x) = 0 \qquad i \in \mathcal{A}_*^{\text{NLP}} \\ & \cancel{c_i(x) \le 0} \qquad i \in \overline{\mathcal{A}}_*^{\text{NLP}} \end{aligned}$$

$$\begin{aligned} \min_{p\in\mathbb{R}^n} \quad & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t.} \quad & \nabla c_{k,i}^T p + c_{k,i} = 0 \qquad i \in \mathcal{E} \\ & \nabla c_{k,i}^T p + c_{k,i} = 0 \qquad i \in \mathcal{A}_*^{\text{QP}_k} \\ & \cancel{\nabla c_{k,i}^T p + c_{k,i} \le 0} \qquad i \in \overline{\mathcal{A}}_*^{\text{QP}_k} \end{aligned}$$

$$\mathcal{A}_*^{\text{NLP}} = \{ i \in \mathcal{I} : c_i(x^*) = 0 \} \qquad \overline{\mathcal{A}}_*^{\text{NLP}} = \{ i \in \mathcal{I} : c_i(x^*) < 0 \}$$

Lemma

Suppose x^* is a local minimizer satisfying the sufficient second-order optimality conditions, at which LICQ and strict optimality hold. Then $\mathcal{A}_*^{\text{NLP}} = \mathcal{A}_*^{\text{QP}_k}$ for all x_k sufficiently close to x_*.

Back to Newton's Method

$$\begin{aligned}\min_{x\in\mathbb{R}^n}\ & f(x) \\ \text{s.t. } & c_i(x) = 0 \quad & i\in\mathcal{E} \\ & c_i(x) = 0 \quad & i\in\mathcal{A}_*^{\text{NLP}} \\ & \cancel{c_i(x) \le 0} \quad & i\in\overline{\mathcal{A}}_*^{\text{NLP}}\end{aligned}$$

$$\begin{aligned}\min_{p\in\mathbb{R}^n}\ & \frac{1}{2}p^T H_k p + \nabla f_k^T p \\ \text{s.t. } & \nabla c_{k,i}^T p + c_{k,i} = 0 \quad & i\in\mathcal{E} \\ & \nabla c_{k,i}^T p + c_{k,i} = 0 \quad & i\in\mathcal{A}_*^{\text{NLP}} \\ & \cancel{\nabla c_{k,i}^T p + c_{k,i} \le 0} \quad & i\in\overline{\mathcal{A}}_*^{\text{NLP}}\end{aligned}$$

- When x_k is close to x^*, (QP$_k$) produces the same steps as SQP for equality-constrained NLP.

Back to Newton's Method

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c_i(x) = 0 \qquad i \in \mathcal{E} \\ & c_i(x) \color{red}{= 0} \qquad i \in \mathcal{A}_*^{\text{NLP}} \\ & \color{red}{\cancel{c_i(x) \le 0}} \qquad i \in \overline{\mathcal{A}}_*^{\text{NLP}} \end{aligned}$$

$$\begin{aligned} \min_{p\in\mathbb{R}^n} \quad & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\ \text{s.t.} \quad & \nabla c_{k,i}^T p + c_{k,i} = 0 \qquad i \in \mathcal{E} \\ & \nabla c_{k,i}^T p + c_{k,i} \color{red}{= 0} \qquad i \in \mathcal{A}_*^{\text{NLP}} \\ & \color{red}{\cancel{\nabla c_{k,i}^T p + c_{k,i} \le 0}} \qquad i \in \overline{\mathcal{A}}_*^{\text{NLP}} \end{aligned}$$

- When x_k is close to x^*, (QP_k) produces the same steps as SQP for equality-constrained NLP.
- We are back to Newton's method...

Back to Newton's Method

$$
\begin{aligned}
\min_{x\in\mathbb{R}^n} \quad & f(x) \\
\text{s.t.} \quad & c_i(x) = 0 \qquad i \in \mathcal{E} \\
& c_i(x) = 0 \qquad i \in \mathcal{A}_*^{\text{NLP}} \\
& \cancel{c_i(x) \le 0} \qquad i \in \overline{\mathcal{A}}_*^{\text{NLP}}
\end{aligned}
$$

$$
\begin{aligned}
\min_{p\in\mathbb{R}^n} \quad & \frac{1}{2} p^T H_k p + \nabla f_k^T p \\
\text{s.t.} \quad & \nabla c_{k,i}^T p + c_{k,i} = 0 \qquad i \in \mathcal{E} \\
& \nabla c_{k,i}^T p + c_{k,i} = 0 \qquad i \in \mathcal{A}_*^{\text{NLP}} \\
& \cancel{\nabla c_{k,i}^T p + c_{k,i} \le 0} \qquad i \in \overline{\mathcal{A}}_*^{\text{NLP}}
\end{aligned}
$$

- When x_k is close to x^*, (QP_k) produces the same steps as SQP for equality-constrained NLP.
- We are back to Newton's method...
- Fast local convergence!

Global Convergence

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c_E(x) = 0 \\ & c_I(x) \leq 0 \end{aligned}$$

Methods for equality constraints can be generalized.

- For example, penalty function

$$\phi_\rho(x) = f(x) + \rho \|c_E(x)\|_1 + \rho \| \max\{c_I(x), 0\} \|_1.$$

Global Convergence

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \ & f(x) \\ \text{s.t.} \ & c_E(x) = 0 \\ & c_I(x) \le 0 \end{aligned}$$

Methods for equality constraints can be generalized.

- For example, penalty function

$$\phi_\rho(x) = f(x) + \rho\|c_E(x)\|_1 + \rho\|\max\{c_I(x), 0\}\|_1.$$

$$\begin{aligned} \min_{p\in\mathbb{R}^n; t,s\in\mathbb{R}^{m_E}; r\in\mathbb{R}^{m_I}} \ & \frac{1}{2}p^T H_k p + \nabla f_k^T p + \rho\sum_{j=1}^{m_E}(s_j + t_j) + \rho\sum_{j=1}^{m_I} r_j \\ \text{s.t.} \ & \nabla c_{E,k}^T p + c_{E,k} = s - t \\ & \nabla c_{E,k}^T p + c_{E,k} \le r \\ & \|p\| \le \Delta_k, \quad s, t, r \ge 0 \end{aligned}$$

Table of Contents

Barrier Problem

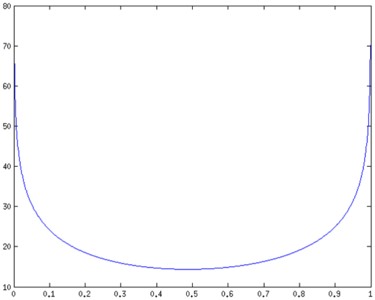
$$\boxed{\begin{aligned} \min_{x\in\mathbb{R}^n} \; & f(x) \\ \text{s.t.} \; & c(x) = 0 \\ & x \geq 0 \end{aligned}}$$

Barrier Problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & \cancel{x \geq 0} \end{aligned} \qquad \longrightarrow \qquad \begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) - \mu \sum_{i=1}^{n} \log(x_i) \\ \text{s.t.} \quad & c(x) = 0 \end{aligned}$$

Barrier Problem

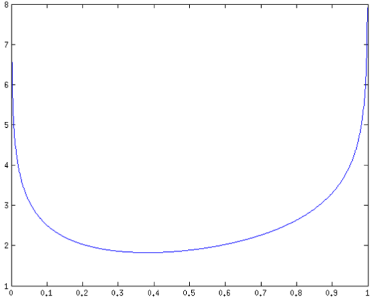
$$\boxed{\begin{aligned} &\min_{x\in\mathbb{R}} \; f(x) \\ &\text{s.t.} \; \cancel{x \geq 0} \\ &\qquad \cancel{x \leq 10} \end{aligned}} \longrightarrow \boxed{\min_{x\in\mathbb{R}} \; f(x) - \mu \log(x) - \mu \log(10 - x)}$$



$$\mu = 10$$

Barrier Problem

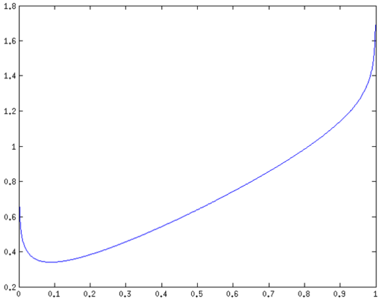
$$\min_{x\in\mathbb{R}} f(x) \quad \text{s.t. } \cancel{x \geq 0},\ \cancel{x \leq 10} \quad \longrightarrow \quad \boxed{\min_{x\in\mathbb{R}} f(x) - \mu \log(x) - \mu \log(10-x)}$$



$\mu = 1$

Barrier Problem

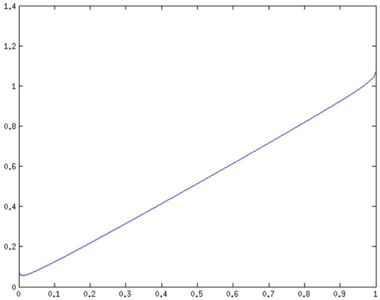
$$\boxed{\begin{aligned} &\min_{x\in\mathbb{R}} \; f(x) \\ &\text{s.t.} \;\; \cancel{x \geq 0} \\ &\qquad\;\; \cancel{x \leq 10} \end{aligned}} \longrightarrow \boxed{\min_{x\in\mathbb{R}} \; f(x) - \mu\log(x) - \mu\log(10-x)}$$



$\mu = 0.1$

Barrier Problem

$$\begin{aligned}\min_{x\in\mathbb{R}} \;& f(x)\\ \text{s.t.}\;& \cancel{x \geq 0}\\ & \cancel{x \leq 10}\end{aligned} \quad\longrightarrow\quad \boxed{\min_{x\in\mathbb{R}} f(x) - \mu\log(x) - \mu\log(10-x)}$$

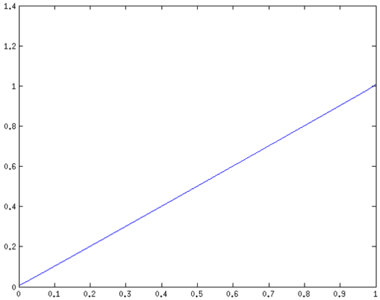


$\mu = 0.01$

Barrier Problem

$$\min_{x \in \mathbb{R}} f(x) \quad \text{s.t.} \quad \cancel{x \geq 0}, \quad \cancel{x \leq 10}$$

$$\longrightarrow \quad \min_{x \in \mathbb{R}} f(x) - \mu \log(x) - \mu \log(10 - x)$$



$$\mu = 0.001$$

Barrier Method

$$\boxed{\begin{aligned} \min_{x\in\mathbb{R}^n}\ & f(x) \\ \text{s.t. }\ & c(x) = 0 \\ & \cancel{x \geq 0} \end{aligned}} \longrightarrow \boxed{\begin{aligned} \min_{x\in\mathbb{R}^n}\ & f(x) - \mu \sum_{i=1}^{n} \log(x_i) \\ \text{s.t. }\ & c(x) = 0 \end{aligned}} \quad (\mathrm{BP}_\mu)$$

Basic Algorithm:

1. Choose $x_0 \in \mathbb{R}^n$, $\mu_0 > 0$, $\epsilon_0 > 0$. Set $k \leftarrow 0$.

Barrier Method

$$\boxed{\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & \cancel{x \geq 0} \end{aligned}} \longrightarrow \boxed{\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) - \mu \sum_{i=1}^{n} \log(x_i) \\ \text{s.t.} \quad & c(x) = 0 \end{aligned}} \quad (\mathrm{BP}_\mu)$$

Basic Algorithm:

1. Choose $x_0 \in \mathbb{R}^n$, $\mu_0 > 0$, $\epsilon_0 > 0$. Set $k \leftarrow 0$.
2. Starting from x_0, solve (BP_{μ_k}) to tolerance ϵ_k and obtain x_{k+1}.

Barrier Method

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & f(x) \\ \text{s.t. } & c(x) = 0 \\ & \cancel{x \geq 0} \end{aligned} \quad \longrightarrow \quad \begin{aligned} \min_{x \in \mathbb{R}^n} \ & f(x) - \mu \sum_{i=1}^{n} \log(x_i) \\ \text{s.t. } & c(x) = 0 \end{aligned} \quad (\mathrm{BP}_\mu)$$

Basic Algorithm:

1. Choose $x_0 \in \mathbb{R}^n$, $\mu_0 > 0$, $\epsilon_0 > 0$. Set $k \leftarrow 0$.
2. Starting from x_0, solve (BP_{μ_k}) to tolerance ϵ_k and obtain x_{k+1}.
3. Decrease $\mu_{k+1} < \mu_k$ and $\epsilon_{k+1} < \epsilon_k$; set $k \leftarrow k + 1$; go to 2.

   (Ensure $\mu_k \to 0$ and $\epsilon_k \to 0$.)

Barrier Method

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & f(x) \\ \text{s.t. } & c(x) = 0 \\ & \cancel{x \geq 0} \end{aligned} \quad \longrightarrow \quad \begin{aligned} \min_{x \in \mathbb{R}^n} \; & f(x) - \mu \sum_{i=1}^{n} \log(x_i) \\ \text{s.t. } & c(x) = 0 \end{aligned} \quad (\mathrm{BP}_\mu)$$

Basic Algorithm:

1. Choose $x_0 \in \mathbb{R}^n$, $\mu_0 > 0$, $\epsilon_0 > 0$. Set $k \leftarrow 0$.
2. Starting from x_0, solve (BP_{μ_k}) to tolerance ϵ_k and obtain x_{k+1}.

   $\rightarrow$ Use SQP techniques
3. Decrease $\mu_{k+1} < \mu_k$ and $\epsilon_{k+1} < \epsilon_k$; set $k \leftarrow k+1$; go to 2.

   (Ensure $\mu_k \rightarrow 0$ and $\epsilon_k \rightarrow 0$.)

SQP Techniques for Solving the Barrier Problem

$$\min_{x\in\mathbb{R}^n} \varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \log(x_i)$$
$$\text{s.t. } c(x) = 0$$

Can re-use SQP techniques:

- Step computation
  - KKT system with regularization
  - Decomposition

SQP Techniques for Solving the Barrier Problem

$$\min_{x \in \mathbb{R}^n} \varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \log(x_i)$$
$$\text{s.t. } c(x) = 0$$

Can re-use SQP techniques:

- Step computation
  - KKT system with regularization
  - Decomposition
- Step acceptance
  - Line search
  - Trust region

SQP Techniques for Solving the Barrier Problem

$$\min_{x \in \mathbb{R}^n} \varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \log(x_i)$$
$$\text{s.t. } c(x) = 0$$

Can re-use SQP techniques:

- Step computation
  - KKT system with regularization
  - Decomposition
- Step acceptance
  - Line search
  - Trust region
- Measuring progress
  - Exact penalty function
  - Filter method

Barrier Term Considerations

$$\min_{x \in \mathbb{R}^n} \varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \log(x_i)$$
$$\text{s.t. } c(x) = 0$$

- Variables must stay positive

Barrier Term Considerations

$$\boxed{\begin{aligned} \min_{x\in\mathbb{R}^n} \; & \varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \log(x_i) \\ \text{s.t. } \; & c(x) = 0 \end{aligned}}$$

- Variables must stay positive
  - Fraction-to-the-boundary rule ($\tau \in (0,1)$, e.g., $\tau = 0.99$)

$$\alpha_k^{\max} = \arg\max \left\{\alpha \in (0,1] : x_k + \alpha p_k \geq (1-\tau)x_k\right\}$$

Barrier Term Considerations

$$\min_{x \in \mathbb{R}^n} \varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \log(x_i)$$
$$\text{s.t. } c(x) = 0$$

- Variables must stay positive
  - Fraction-to-the-boundary rule $(\tau \in (0,1)$, e.g., $\tau = 0.99)$

$$\alpha_k^{\max} = \arg\max \{\alpha \in (0,1] : x_k + \alpha p_k \geq (1-\tau) x_k\}$$

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k + \mu X_k^{-2} & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k - \mu X_k^{-1} e \\ c_k \end{pmatrix}$$

Barrier Term Considerations

$$\min_{x \in \mathbb{R}^n} \varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \log(x_i)$$
$$\text{s.t. } c(x) = 0$$

- Variables must stay positive
  - Fraction-to-the-boundary rule $(\tau \in (0,1)$, e.g., $\tau = 0.99)$

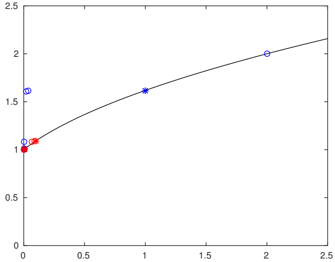
$$\alpha_k^{\max} = \arg\max \{\alpha \in (0,1] : x_k + \alpha p_k \geq (1-\tau) x_k\}$$

- Ill-conditioning in linear system

$$\begin{bmatrix} \nabla^2_{xx} \mathcal{L}_k + \mu X_k^{-2} & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k - \mu X_k^{-1} e \\ c_k \end{pmatrix}$$

Example

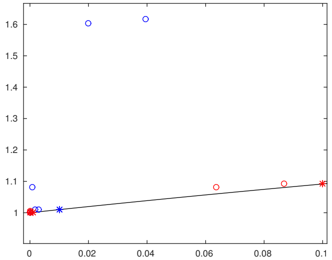
$$\min_{x \in \mathbb{R}^2} \; x_1 + (x_2 - 1)^2$$
$$\text{s.t. } x \geq 0$$




```
 k |     mu_k |      f_k | (  x_k(1),  x_k(2)) | (   p_k(1),   p_k(2)) | err_barr |    alpha
------------------------------------------------------------------------------------------------
 0 | 1.00e+00 | 2.50e+00 | (2.00e+00,2.00e+00) | ( 0.00e+00, 0.00e+00) | 7.07e-01 | 0.00e+00
 1 | 1.00e+00 | 2.02e-01 | (2.00e-02,1.60e+00) | (-2.00e+00,-4.00e-01) | 2.04e+00 | 9.90e-01
 2 | 1.00e+00 | 2.31e-01 | (3.96e-02,1.62e+00) | ( 1.96e-02, 1.40e-02) | 2.41e-02 | 1.00e+00
 3 | 1.00e-01 | 6.69e-02 | (6.35e-02,1.08e+00) | ( 2.39e-02,-5.36e-01) | 5.36e-01 | 1.00e+00
 4 | 1.00e-01 | 9.09e-02 | (8.67e-02,1.09e+00) | ( 2.32e-02, 9.33e-03) | 2.50e-02 | 1.00e+00
 5 | 1.00e-02 | 4.15e-03 | (8.67e-04,1.08e+00) | (-6.65e-01,-8.18e-02) | 6.70e-01 | 1.29e-01
 6 | 1.00e-02 | 1.71e-03 | (1.66e-03,1.01e+00) | ( 7.92e-04,-7.12e-02) | 7.12e-02 | 1.00e+00
 ...
12 | 3.16e-05 | 2.70e-05 | (2.70e-05,1.00e+00) | ( 7.45e-06, 2.95e-11) | 7.45e-06 | 1.00e+00
13 | 1.78e-07 | 4.98e-10 | (4.81e-12,1.00e+00) | (-4.09e-03,-3.14e-05) | 4.09e-03 | 6.62e-03
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  1.300371e-16.
14 | 1.78e-07 | 9.63e-12 | (9.62e-12,1.00e+00) | ( 4.81e-12,-3.12e-05) | 3.12e-05 | 1.00e+00
15 | 1.78e-07 | 1.93e-11 | (1.92e-11,1.00e+00) | ( 9.62e-12, 9.44e-17) | 9.62e-12 | 1.00e+00
16 | 7.50e-11 | 3.35e-11 | (3.35e-11,1.00e+00) | ( 1.43e-11,-1.78e-07) | 1.78e-07 | 1.00e+00
```

Example

$$\min_{x\in\mathbb{R}^2} \; x_1 + (x_2 - 1)^2$$
$$\text{s.t. } x \geq 0$$




```
 k |     mu_k |      f_k | (  x_k(1),  x_k(2)) | (   p_k(1),   p_k(2)) | err_barr |    alpha
-------------------------------------------------------------------------------------------------
 0 | 1.00e+00 | 2.50e+00 | (2.00e+00,2.00e+00) | ( 0.00e+00, 0.00e+00) | 7.07e-01 | 0.00e+00
 1 | 1.00e+00 | 2.02e-01 | (2.00e-02,1.60e+00) | (-2.00e+00,-4.00e-01) | 2.04e+00 | 9.90e-01
 2 | 1.00e+00 | 2.31e-01 | (3.96e-02,1.62e+00) | ( 1.96e-02, 1.40e-02) | 2.41e-02 | 1.00e+00
 3 | 1.00e-01 | 6.69e-02 | (6.35e-02,1.08e+00) | ( 2.39e-02,-5.36e-01) | 5.36e-01 | 1.00e+00
 4 | 1.00e-01 | 9.09e-02 | (8.67e-02,1.09e+00) | ( 2.32e-02, 9.33e-03) | 2.50e-02 | 1.00e+00
 5 | 1.00e-02 | 4.15e-03 | (8.67e-04,1.08e+00) | (-6.65e-01,-8.18e-02) | 6.70e-01 | 1.29e-01
 6 | 1.00e-02 | 1.71e-03 | (1.66e-03,1.01e+00) | ( 7.92e-04,-7.12e-02) | 7.12e-02 | 1.00e+00
 ...
12 | 3.16e-05 | 2.70e-05 | (2.70e-05,1.00e+00) | ( 7.45e-06, 2.95e-11) | 7.45e-06 | 1.00e+00
13 | 1.78e-07 | 4.98e-10 | (4.81e-12,1.00e+00) | (-4.09e-03,-3.14e-05) | 4.09e-03 | 6.62e-03
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  1.300371e-16.
14 | 1.78e-07 | 9.63e-12 | (9.62e-12,1.00e+00) | ( 4.81e-12,-3.12e-05) | 3.12e-05 | 1.00e+00
15 | 1.78e-07 | 1.93e-11 | (1.92e-11,1.00e+00) | ( 9.62e-12, 9.44e-17) | 9.62e-12 | 1.00e+00
16 | 7.50e-11 | 3.35e-11 | (3.35e-11,1.00e+00) | ( 1.43e-11,-1.78e-07) | 1.78e-07 | 1.00e+00
```

Primal-Dual System

$$\boxed{\begin{aligned} \min_{x \in \mathbb{R}^n} \ & f(x) \\ \text{s.t. } & c(x) = 0 \\ & x \geq 0 \end{aligned}}$$

Primal-Dual System

$$\boxed{\begin{aligned} \min_{x\in\mathbb{R}^n} \;& f(x) \\ \text{s.t. } & c(x) = 0 \\ & x \geq 0 \end{aligned}}$$

$$\boxed{\begin{aligned} \nabla f(x) + \nabla c(x) - z &= 0 \\ c(x) &= 0 \\ XZe &= 0 \\ x, z &\geq 0 \end{aligned}}$$

Primal-Dual System

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$

$$\begin{aligned} \nabla f(x) + \nabla c(x) - z &= 0 \\ c(x) &= 0 \\ XZe &= \mu e \\ (x, z &\geq 0) \end{aligned}$$

Primal-Dual System

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$

$$\begin{aligned} \nabla f(x) + \nabla c(x) - z &= 0 \\ c(x) &= 0 \\ XZe &= \mu e \\ (x, z &\geq 0) \end{aligned}$$

Newton Steps

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k & \nabla c_k & -I \\ \nabla c_k^T & 0 & 0 \\ Z_k & 0 & X_k \end{bmatrix} \begin{pmatrix} p_k \\ p_k^\lambda \\ p_k^z \end{pmatrix} = - \begin{pmatrix} \nabla f_k + \nabla c_k \lambda_k - z_k \\ c_k \\ X_k Z_k e - \mu e \end{pmatrix}$$

Primal-Dual System

$$\min_{x \in \mathbb{R}^n} \; f(x)$$
$$\text{s.t. } c(x) = 0$$
$$x \geq 0$$

$$\begin{aligned} \nabla f(x) + \nabla c(x) - z &= 0 \\ c(x) &= 0 \\ XZe &= \mu e \\ (x, z &\geq 0) \end{aligned}$$

Newton Steps

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k & \nabla c_k & -I \\ \nabla c_k^T & 0 & 0 \\ Z_k & 0 & X_k \end{bmatrix} \begin{pmatrix} p_k \\ p_k^\lambda \\ p_k^z \end{pmatrix} = - \begin{pmatrix} \nabla f_k + \nabla c_k \lambda_k - z_k \\ c_k \\ X_k Z_k e - \mu e \end{pmatrix}$$

Block elimination

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k + X_k^{-1} Z_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k - \mu X_k^{-1} e \\ c_k \end{pmatrix}$$

Primal-Dual Steps

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k + \Sigma_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k - \mu X_k^{-1} e \\ c_k \end{pmatrix}$$

Primal-Dual Steps

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k + \Sigma_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k - \mu X_k^{-1} e \\ c_k \end{pmatrix}$$

- Barrier Hessian term:
  - $\Sigma_k = X_k^{-2}$: primal
  - $\Sigma_k = X_k^{-1} Z_k$: primal-dual

Primal-Dual Steps

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k + \Sigma_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k - \mu X_k^{-1} e \\ c_k \end{pmatrix}$$

- Barrier Hessian term:
  - $\Sigma_k = X_k^{-2}$: primal
  - $\Sigma_k = X_k^{-1} Z_k$: primal-dual
- Step for dual variables: $p_k^z = \mu X_k^{-1} e - z_k - \Sigma_k p_k$.

Primal-Dual Steps

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k + \Sigma_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k - \mu X_k^{-1} e \\ c_k \end{pmatrix}$$

- Barrier Hessian term:
  - $\Sigma_k = X_k^{-2}$: primal
  - $\Sigma_k = X_k^{-1} Z_k$: primal-dual
- Step for dual variables: $p_k^z = \mu X_k^{-1} e - z_k - \Sigma_k p_k$.
- Can still use SQP-type globalization techniques.

Primal-Dual Steps

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k + \Sigma_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k - \mu X_k^{-1} e \\ c_k \end{pmatrix}$$

- Barrier Hessian term:
  - $\Sigma_k = X_k^{-2}$: primal
  - $\Sigma_k = X_k^{-1} Z_k$: primal-dual
- Step for dual variables: $p_k^z = \mu X_k^{-1} e - z_k - \Sigma_k p_k$.
- Can still use SQP-type globalization techniques.
- Now: Fast local convergence.

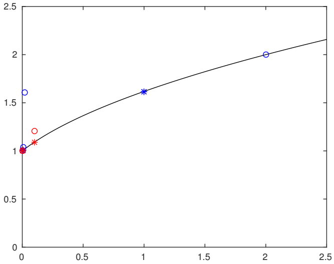
Primal-Dual Steps

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k + \Sigma_k & \nabla c_k \\ \nabla c_k^T & 0 \end{bmatrix} \begin{pmatrix} p_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k - \mu X_k^{-1} e \\ c_k \end{pmatrix}$$

- Barrier Hessian term:
  - $\Sigma_k = X_k^{-2}$: primal
  - $\Sigma_k = X_k^{-1} Z_k$: primal-dual
- Step for dual variables: $p_k^z = \mu X_k^{-1} e - z_k - \Sigma_k p_k$.
- Can still use SQP-type globalization techniques.
- Now: Fast local convergence.
- Ill-conditioning in KKT system is benign for direct linear solvers.

Example Revisited with Primal-Dual Method

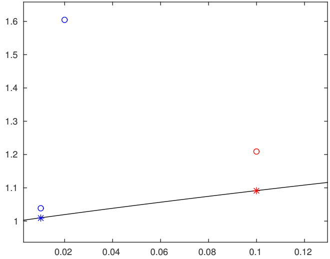
$$\min_{x \in \mathbb{R}^2} \; x_1 + (x_2 - 1)^2$$
$$\text{s.t. } x \geq 0$$



k	mu_k	f_k	(x_k(1), x_k(2))	(p_k(1), p_k(2))	err_barr	alpha
0	1.00e+00	2.50e+00	(2.00e+00,2.00e+00)	(0.00e+00, 0.00e+00)	7.07e-01	0.00e+00
1	1.00e+00	2.02e-01	(2.00e-02,1.60e+00)	(-2.00e+00,-4.00e-01)	4.00e-03	9.90e-01
2	1.00e-01	1.22e-01	(1.00e-01,1.21e+00)	(8.00e-02,-3.94e-01)	4.74e-16	1.00e+00
3	1.00e-02	1.07e-02	(1.00e-02,1.04e+00)	(-9.00e-02,-1.72e-01)	5.55e-17	1.00e+00
4	1.00e-03	1.00e-03	(1.00e-03,1.00e+00)	(-9.00e-03,-3.58e-02)	5.55e-17	1.00e+00
5	3.16e-05	3.16e-05	(3.16e-05,1.00e+00)	(-9.68e-04,-2.24e-03)	3.64e-17	1.00e+00
6	1.78e-07	1.78e-07	(1.78e-07,1.00e+00)	(-3.14e-05,-3.65e-05)	1.10e-16	1.00e+00
7	7.50e-11	7.50e-11	(7.50e-11,1.00e+00)	(-1.78e-07,-1.79e-07)	5.34e-17	1.00e+00

Example Revisited with Primal-Dual Method

$$\min_{x \in \mathbb{R}^2} \; x_1 + (x_2 - 1)^2$$
$$\text{s.t. } x \geq 0$$



k	mu_k	f_k	(x_k(1), x_k(2))	(p_k(1), p_k(2))	err_barr	alpha
0	1.00e+00	2.50e+00	(2.00e+00,2.00e+00)	(0.00e+00, 0.00e+00)	7.07e-01	0.00e+00
1	1.00e+00	2.02e-01	(2.00e-02,1.60e+00)	(-2.00e+00,-4.00e-01)	4.00e-03	9.90e-01
2	1.00e-01	1.22e-01	(1.00e-01,1.21e+00)	(8.00e-02,-3.94e-01)	4.74e-16	1.00e+00
3	1.00e-02	1.07e-02	(1.00e-02,1.04e+00)	(-9.00e-02,-1.72e-01)	5.55e-17	1.00e+00
4	1.00e-03	1.00e-03	(1.00e-03,1.00e+00)	(-9.00e-03,-3.58e-02)	5.55e-17	1.00e+00
5	3.16e-05	3.16e-05	(3.16e-05,1.00e+00)	(-9.68e-04,-2.24e-03)	3.64e-17	1.00e+00
6	1.78e-07	1.78e-07	(1.78e-07,1.00e+00)	(-3.14e-05,-3.65e-05)	1.10e-16	1.00e+00
7	7.50e-11	7.50e-11	(7.50e-11,1.00e+00)	(-1.78e-07,-1.79e-07)	5.34e-17	1.00e+00

Table of Contents

Advantages of Different Algorithms

(very rough guide. . .)

Advantages of Different Algorithms

(very rough guide. . .)

SQP methods

- Very efficient for small- to medium-sized problems
  - up to several thousand variables and constraints

Advantages of Different Algorithms

(very rough guide…)

SQP methods

- Very efficient for small- to medium-sized problems
  - up to several thousand variables and constraints
- Can exploit good estimate of solution (warm starts)
  - branch-and-bound for mixed-integer nonlinear programming
  - real-time optimal control

Advantages of Different Algorithms

(very rough guide...)

SQP methods

- Very efficient for small- to medium-sized problems
  - up to several thousand variables and constraints
- Can exploit good estimate of solution (warm starts)
  - branch-and-bound for mixed-integer nonlinear programming
  - real-time optimal control

Interior-point methods

- Can solve very large problems
  - up to millions of variables and constraints

Advantages of Different Algorithms

(very rough guide...)

SQP methods

- Very efficient for small- to medium-sized problems
  - up to several thousand variables and constraints
- Can exploit good estimate of solution (warm starts)
  - branch-and-bound for mixed-integer nonlinear programming
  - real-time optimal control

Interior-point methods

- Can solve very large problems
  - up to millions of variables and constraints
- Difficult to warm start

Some Optimization Software for NLP

(This is not an exhaustive list!)

Some Optimization Software for NLP

SQP methods (This is not an exhaustive list!)

- SNOPT [Gill, Murray, Sanders]
  - line search with augmented Lagrangian as merit function
  - reduced Hessian BFGS

Some Optimization Software for NLP

SQP methods (This is not an exhaustive list!)

- SNOPT [Gill, Murray, Sanders]
  - line search with augmented Lagrangian as merit function
  - reduced Hessian BFGS
- FilterSQP [Fletcher, Leyffer]
  - Sℓ_1QP with exact Hessian
  - trust-region filter method

Some Optimization Software for NLP

SQP methods (This is not an exhaustive list!)

- SNOPT [Gill, Murray, Sanders]
  - line search with augmented Lagrangian as merit function
  - reduced Hessian BFGS
- FilterSQP [Fletcher, Leyffer]
  - Sℓ_1QP with exact Hessian
  - trust-region filter method

Primal-dual interior-point methods

- Ipopt [Wächter, Biegler]
  - line-search filter method
  - full-space step computation with regularization

Some Optimization Software for NLP

(This is not an exhaustive list!)

SQP methods

- SNOPT [Gill, Murray, Sanders]
  - line search with augmented Lagrangian as merit function
  - reduced Hessian BFGS
- FilterSQP [Fletcher, Leyffer]
  - Sℓ_1QP with exact Hessian
  - trust-region filter method

Primal-dual interior-point methods

- Ipopt [Wächter, Biegler]
  - line-search filter method
  - full-space step computation with regularization
- Knitro [Byrd, Nocedal, Waltz et al.]
  - trust-region with exact penalty function
  - Byrd-Omojokun decomposition
  - other algorithmic options: direct method; SLP-EQP method

Some Optimization Software for NLP

Augmented Lagrangian methods

- Lancelot [Conn, Gould, Toint]
  - trust region
  - gradient projection combined with conjugate gradients

Some Optimization Software for NLP

Augmented Lagrangian methods

- Lancelot [Conn, Gould, Toint]
  - trust region
  - gradient projection combined with conjugate gradients
- Algencan [Birgin, Martinez]
  - trust region
  - spectral projected gradients

Some Optimization Software for NLP

Augmented Lagrangian methods

- Lancelot [Conn, Gould, Toint]
  - trust region
  - gradient projection combined with conjugate gradients
- Algencan [Birgin, Martinez]
  - trust region
  - spectral projected gradients

Based on others algorithmic frameworks

- CONOPT [Arki Consulting]
  - "based on generalize reduced-gradient method"

Some Optimization Software for NLP

Augmented Lagrangian methods

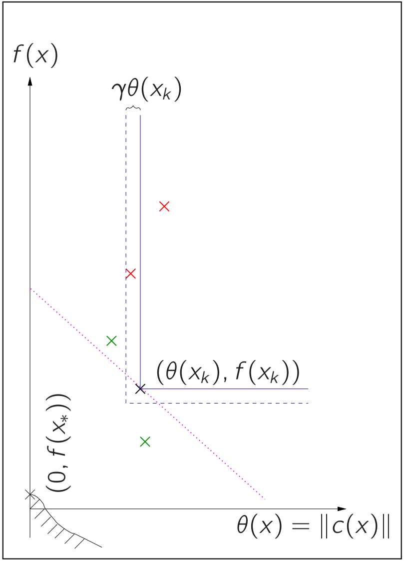
- Lancelot [Conn, Gould, Toint]
  - trust region
  - gradient projection combined with conjugate gradients
- Algencan [Birgin, Martinez]
  - trust region
  - spectral projected gradients

Based on others algorithmic frameworks

- CONOPT [Arki Consulting]
  - "based on generalize reduced-gradient method"
- MINOS [Murtagh, Sanders]
  - linearly-constrained augmented Lagrangian method
  - line search

Thank You!

A Filter Line Search Method

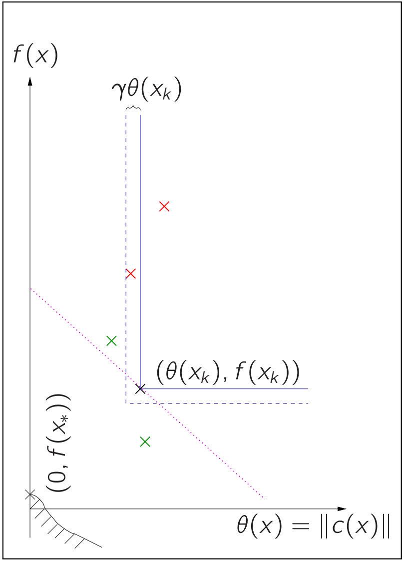


Idea: Bi-objective optimization

$$\boxed{\begin{array}{ll} \min & f(x) \\ \text{s.t.} & c(x) = 0 \end{array}}$$

$$\boxed{\min\ \theta(x)} \qquad \boxed{\min\ f(x)}$$

A Filter Line Search Method



Idea: Bi-objective optimization

$$\boxed{\begin{array}{ll} \min & f(x) \\ \text{s.t.} & c(x) = 0 \end{array}}$$

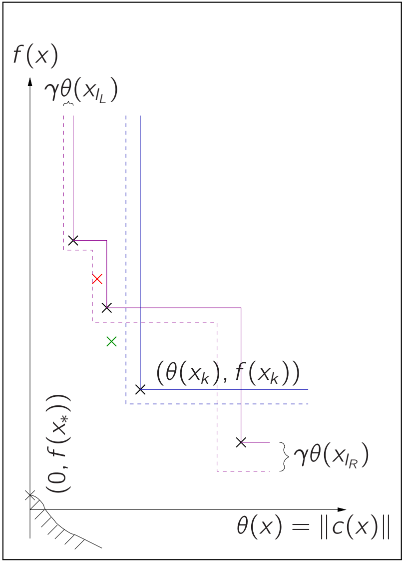
$$\boxed{\min\ \theta(x)} \qquad \boxed{\min\ f(x)}$$

$$\boxed{x_{\text{tr}} = x_k + \alpha d_k^x}$$

Sufficient progress w.r.t. x_k:

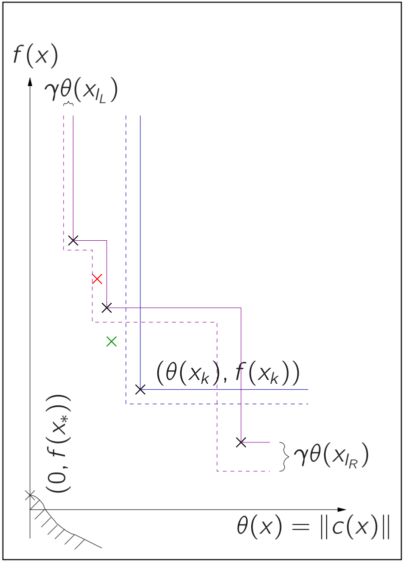
$$\boxed{\begin{array}{lll} f(x_{\text{tr}}) & \leq & f(x_k) - \gamma_f \theta(x_k) \\ \theta(x_{\text{tr}}) & \leq & \theta(x_k) - \gamma_\theta \theta(x_k) \end{array}}$$

A Filter Line Search Method (Filter)



Need to avoid cycling

A Filter Line Search Method (Filter)



Need to avoid cycling

$\Downarrow$

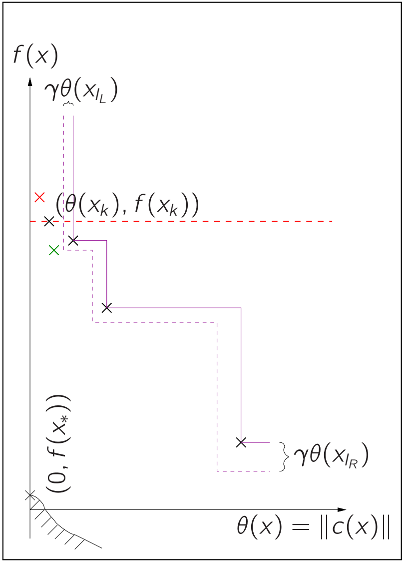
Store some previous $(\theta(x_l), f(x_l))$ pairs in filter $\mathcal{F}_k$

Sufficient progress w.r.t. filter:

$$\begin{aligned} f(x_{\mathrm{tr}}) &\leq f(x_l) - \gamma_f \theta(x_l) \\ \theta(x_{\mathrm{tr}}) &\leq \theta(x_l) - \gamma_\theta \theta(x_l) \end{aligned}$$

for $(\theta(x_l), f(x_l)) \in \mathcal{F}_k$

A Filter Line Search Method ("f-type")



If *switching condition*

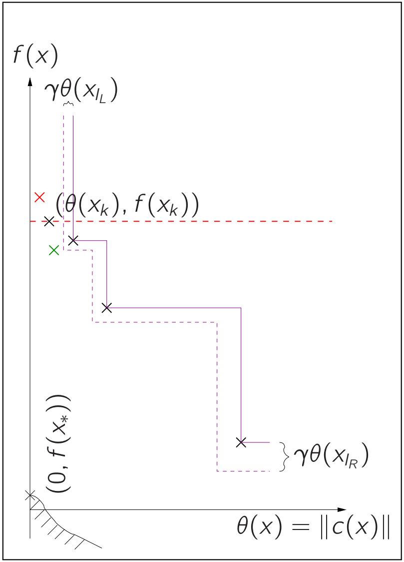
$$\boxed{-\alpha\nabla f(x_k)^T d_k^x > \delta\,[\theta(x_k)]^{s_\theta}}$$

holds ($s_\theta > 1$):

Require Armijo-condition on $f(x)$:

$$\boxed{f(x_{\rm tr}) \le f(x_k) + \eta\alpha\nabla f(x_k)^T d_k^x}$$

A Filter Line Search Method (“f-type”)



If *switching condition*

$$\boxed{-\alpha\nabla f(x_k)^T d_k^x > \delta\,[\theta(x_k)]^{s_\theta}}$$

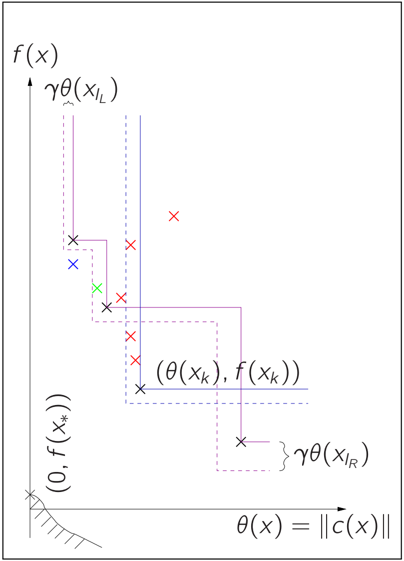
holds ($s_\theta > 1$):

Require Armijo-condition on $f(x)$:

$$\boxed{f(x_{\mathrm{tr}}) \le f(x_k) + \eta\alpha\nabla f(x_k)^T d_k^x}$$

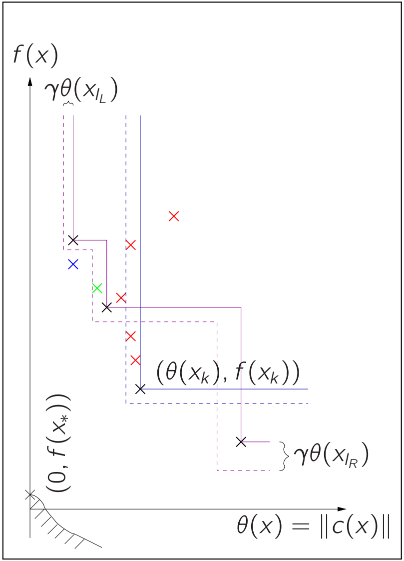
$\Longrightarrow$ Don't augment $\mathcal{F}_k$ in that case

A Filter Line Search Method (Restoration)



If no admissible step size α_k can be found

A Filter Line Search Method (Restoration)



If no admissible step size α_k can be found

$\Downarrow$

Revert to
feasibility restoration phase:

Decrease $\theta(x)$ until

- found acceptable new iterate x_{k+1}, or
- converged to local minimizer of constraint violation