深蓝学院
shenlanxueyuan.com

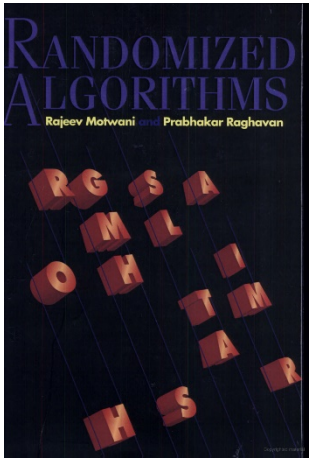
Numerical Optimization in Robotics

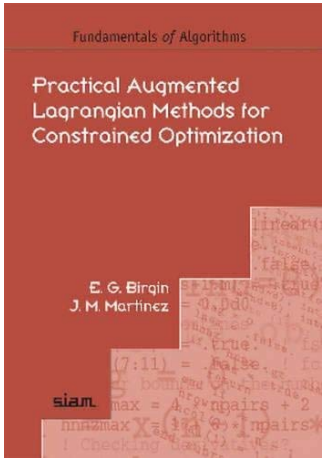
Lecture 3: Constrained Optimization

主讲人 Zhepei Wang

Ph.D. in Robotics
Zhejiang University

Recommended Reading List





The best optimization scheme is problem-specific!

1. Nemirovski A. Advances in convex optimization: conic programming. In International Congress of Mathematicians 2007 May 15 (Vol. 1, pp. 413-444).
2. Seidel R. Small-dimensional linear programming and convex hulls made easy. Discrete & Computational Geometry. 1991 Sep;6(3):423-34.
3. Rockafellar RT. A dual approach to solving nonlinear programming problems by unconstrained optimization. Mathematical Programming. 1973 Dec;5(1):354-73.

Constrained Optimizaiton

Constrained Optimization

optimization problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

$x = (x_1, \cdots, x_n) \in \mathbb{R}^n$: optimization variables

f: $\mathbb{R}^n \mapsto \mathbb{R}$: objective function

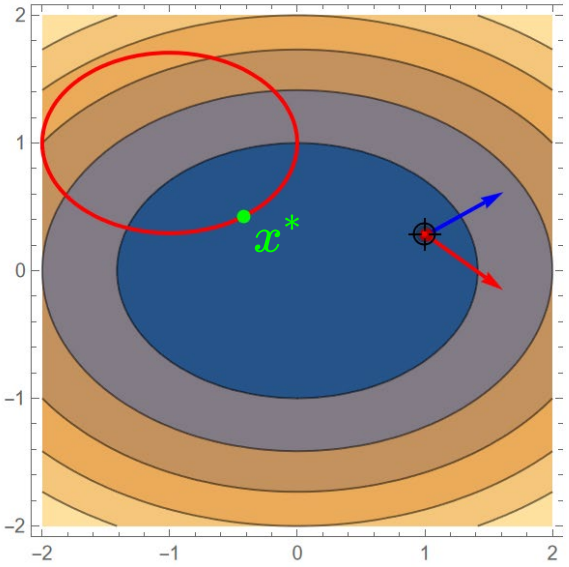
g: $\mathbb{R}^n \mapsto \mathbb{R}^{m_g}$: inequality constraint functions

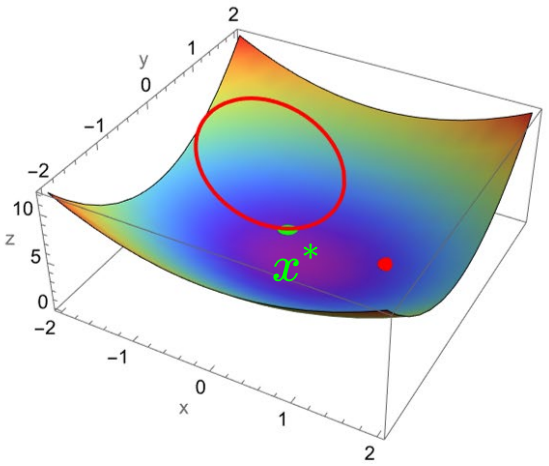
h: $\mathbb{R}^n \mapsto \mathbb{R}^{m_h}$: equality constraint functions

optimal solution x^* has smallest value f among all vectors that satisfy the constraints.

Constrained Optimization

$$\begin{aligned} \min \ & f(x) \\ \text{s.t.} \ & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$



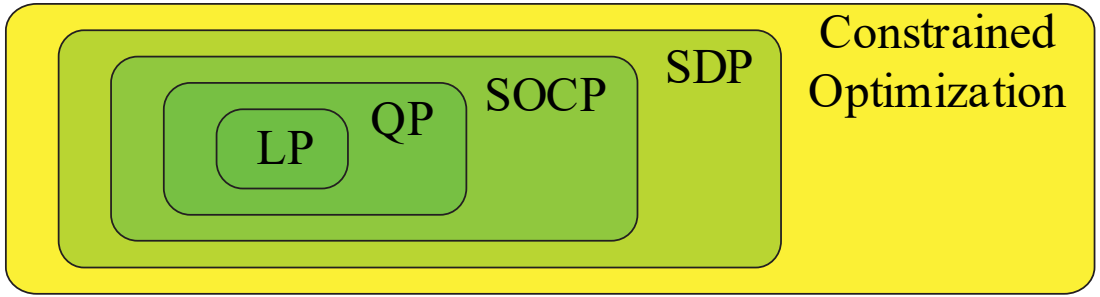


Example:

$$f(x) = x_1^2 + 2x_2^2$$

$$g(x) = x_1^2 + 2x_2^2 - 4x_2 + 3$$

Constrained Opt. : Category and Complexity



Linear Programming (LP)

$$\begin{aligned} \min \quad & c^{\mathrm{T}}x + d \\ \text{s.t.} \quad & Ax \leq b \\ & Gx = h \end{aligned}$$

Second-Order Conic Programming (SOCP)

$$\begin{aligned} \min \quad & f^{\mathrm{T}}x \\ \text{s.t.} \quad & \|A_i x + b_i\|_2 \leq c_i^T x + d_i, \quad i = 1, \ldots, m \\ & Gx = h \end{aligned}$$

Quadratic Programming (QP)

$$\begin{aligned} \min \quad & \tfrac{1}{2}x^{\mathrm{T}}Qx + p^{\mathrm{T}}x + r \\ \text{s.t.} \quad & Ax \leq b \\ & Gx = h \end{aligned}$$

Semi-Definite Programming (SDP)

$$\begin{aligned} \min \quad & c^{\mathrm{T}}x \\ \text{s.t.} \quad & x_1 A_1^i + \cdots + x_n A_n^i + B^i \preceq 0, \; i = 1, \ldots, m \\ & Gx = h \end{aligned}$$

Constrained Opt. : Category and Complexity

Worst-Case Complexity of Some Constrained Programs

Linear Programming (LP)

$$\begin{aligned} \min \quad & c^{\mathrm{T}}x + d \\ \text{s.t.} \quad & Ax \leq b \\ & Gx = h \end{aligned}$$

$O((m+n)^{3/2}n^2L)$

m: number of constraints
n: dimension of decision variables
L: accuracy digits

Second-Order Conic Programming (SOCP)

$$\begin{aligned} \min \quad & f^T x \\ \text{s.t.} \quad & \|A_i x + b_i\|_2 \leq c_i^T x + d_i, \quad i = 1, \ldots, m \\ & Gx = h \end{aligned}$$

$O(m^{1/2}n(mn + n^2 + \sum_i k_i)L)$

m: number of constraints
n: dimension of decision variables
k_i: dimension of cone
L: accuracy digits

Semi-Definite Programming (SDP)

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & x_1 A_1^i + \cdots + x_n A_n^i + B^i \preceq 0, \quad i = 1, \ldots, m \\ & Gx = h \end{aligned}$$

$O(\sqrt{\sum_i k_i}n(n^2 + n\sum_i k_i + \sum_i k_i^3)L)$

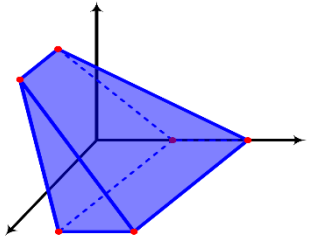
m: number of constraints
n: dimension of decision variables
k_i: row number of matrix
L: accuracy digits

Complexities from polynomial-time interior-point methods (IPMs) under mild conditions.

Constrained Opt. : Category and Complexity

Worst-Case Complexity of Some Constrained Programs

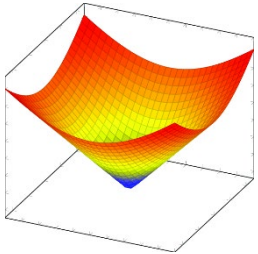
Linear Programming (LP):



$$O((m+n)^{3/2}n^2L)$$

m: number of constraints
n: dimension of decision variables
L: accuracy digits

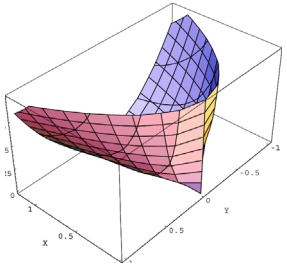
Second-Order Conic Programming (SOCP):



$$O(m^{1/2}n(mn+n^2+\sum_i k_i)L)$$

m: number of constraints
n: dimension of decision variables
k_i: dimension of cone
L: accuracy digits

Semi-Definite Programming (SDP):



$$O(\sqrt{\sum_i k_i}n(n^2+n\sum_i k_i+\sum_i k_i^3)L)$$

m: number of constraints
n: dimension of decision variables
k_i: row number of matrix
L: accuracy digits

Complexities from polynomial-time interior-point methods (IPMs) under mild conditions.

Constrained Opt. : Category and Complexity

Worst-Case Complexity of Some Constrained Programs

Linear Programming (LP)

$$O((m+n)^{3/2}n^2L)$$

Second-Order Conic Programming (SOCP)

$$O(m^{1/2}n(mn+n^2+\sum_i k_i)L)$$

Semi-Definite Programming (SDP)

$$O(\sqrt{\sum_i k_i}n(n^2+n\sum_i k_i+\sum_i k_i^3)L)$$

Features of IPMs

- IPM typically needs $O(\sqrt{m})$ iterations
- IPM gives a promise of computational complexity
- IPM has quadratic/cubic cost per iteration (dense)
- IPM extracts solutions or infeasibilities for LP/SOCP/SDP
- IPM has numerous stable implementations (MOSEK, Gurobi)

Does IPM suit every problem?

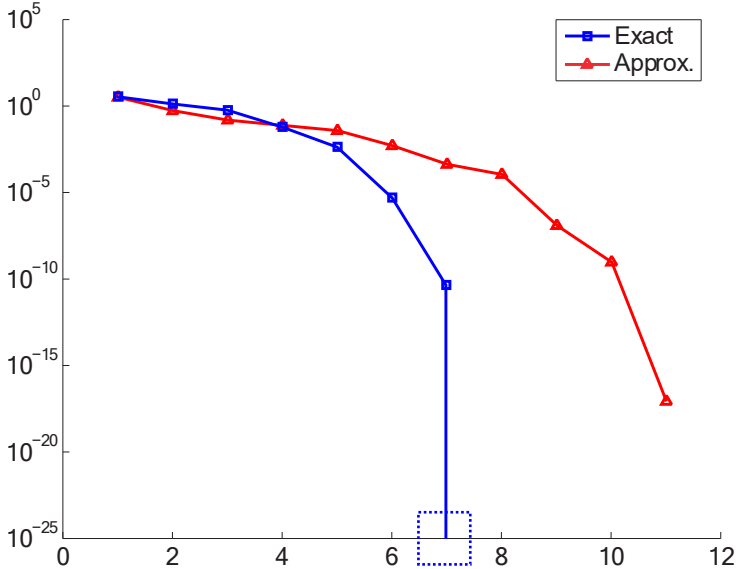
- What if the dimension or constraint size is really large?
- What if Hessian info is unavailable?
- What if solution precision is less important?
- What if exact solution is needed?
- What if small problems are frequently solved?
- …

Constrained Opt. : Category and Complexity

Optimization algorithms: Approximation vs. Exact

Approximation: algorithm will eventually produce a solution with some guarantees.

Exact: algorithm will eventually provide an optimal exact solution.



Exact opt gives exact solutions after finite iterations, while approx. opt applies to more problems.

Constrained Opt. : Category and Complexity

The most suitable opt algorithm is case-by-case.

- Need exact solution or not?
- Very large variable dimension or not?
- Very large constraint number or not?
- Hessian is available or not?
- Overhead is negligible or not?
- Need guaranteed complexity or not?
- …

Low-Dimensional LP: Seidel's Alg.

Low-Dimensional Linear Programming

Problem Statement

Maximize the objective function

$$f(x_1, x_2 \dots x_d) = c_1 x_1 + c_2 x_2 + \cdots + c_d x_d$$

under the constraints

$$\begin{aligned} a_{1,1} x_1 + \cdots + a_{1,d} x_d &\leqslant b_1 \\ a_{2,1} x_1 + \cdots + a_{2,d} x_d &\leqslant b_2 \\ \vdots \qquad\qquad\quad & \vdots \quad \vdots \\ a_{n,1} x_1 + \cdots + a_{n,d} x_d &\leqslant b_n \end{aligned}$$

This is linear programming in dimension *d*.

Equivalently, the goal could be to minimize *f*.

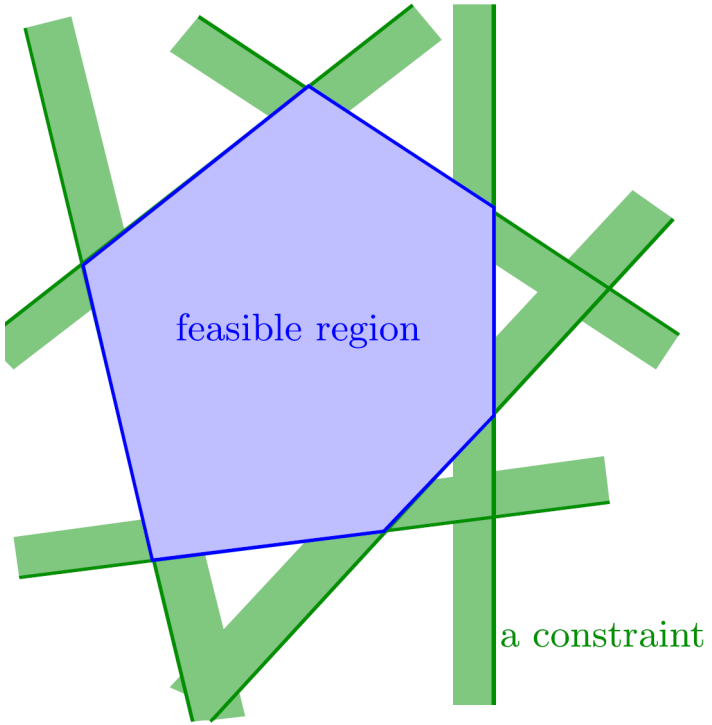
Low-Dimensional Linear Programming

Geometric Interpretation

Each constraint represents a half-space in $\mathbb{R}^d$.

Intersection of half-spaces forms the feasible region.

The feasible region is a convex polytope in $\mathbb{R}^d$.

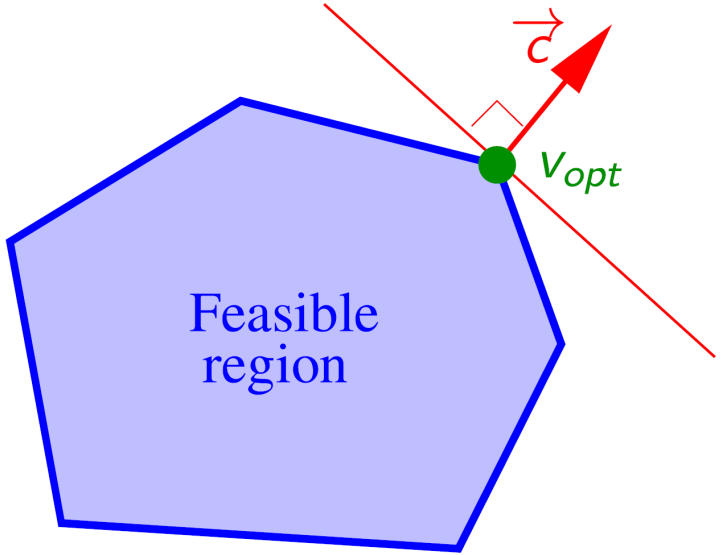


Low-Dimensional Linear Programming

Geometric Interpretation

Let $\vec{c} = (c_1, c_2, \ldots c_d)$.

We want to find a point $\mathrm{v}_{\mathrm{opt}}$ of the feasible region such that $\vec{c}$ is an outer normal at $\mathrm{v}_{\mathrm{opt}}$, if there is one.



Low-Dimensional Linear Programming

Complexity Comparison

Many optimization problems in engineering are linear programs.

A practical algorithm: The simplex algorithm.

- Exact but exponential time in the worst case.

There are polynomial time algorithms: Interior point methods.

- Polynomial time in the worst case but inexact.

Many problems in robot navigation with small dimension and a large number of constraints require efficient and exact solutions.

Seidel's LP Algorithm

One-dimensional case

Maximize the objective function

$$f(x) = cx$$

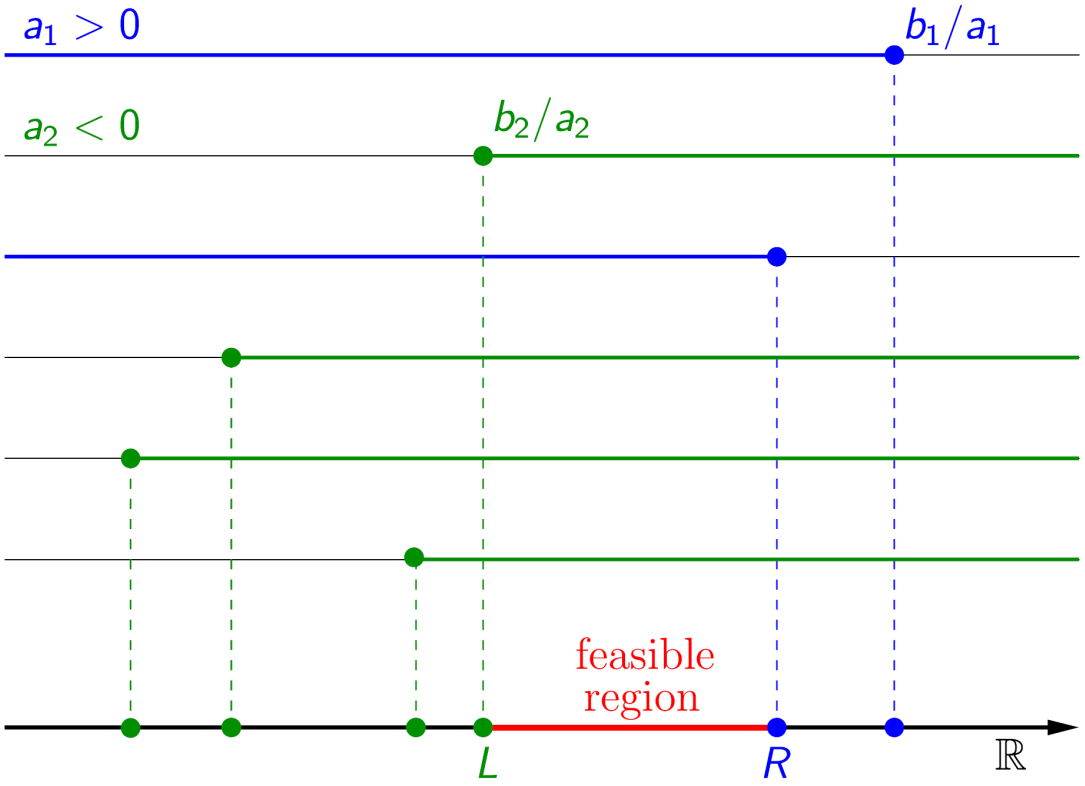
under the constraints

$$\begin{aligned} a_1 x &\leqslant b_1 \\ a_2 x &\leqslant b_2 \\ \vdots \quad &\quad \vdots \\ a_n x &\leqslant b_n \end{aligned}$$

It is trivial to solve this problem in linear time.

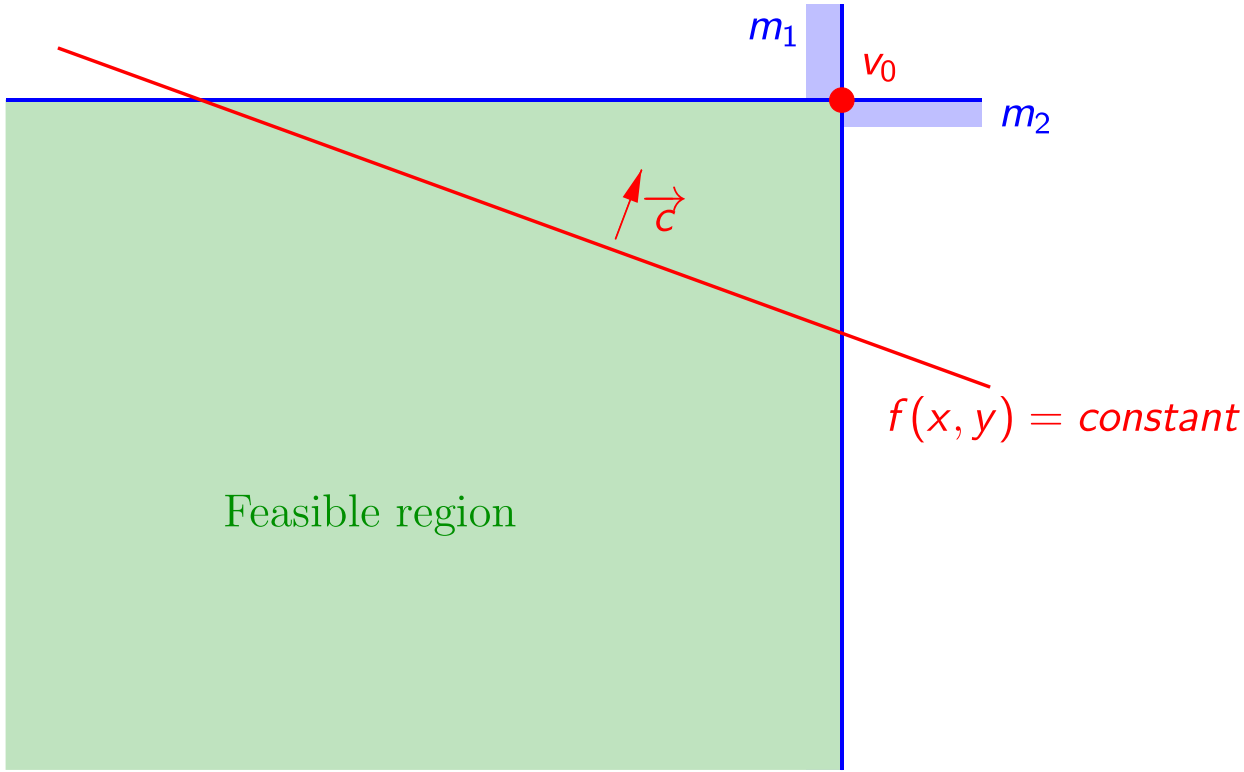
Seidel's LP Algorithm

One-dimensional case



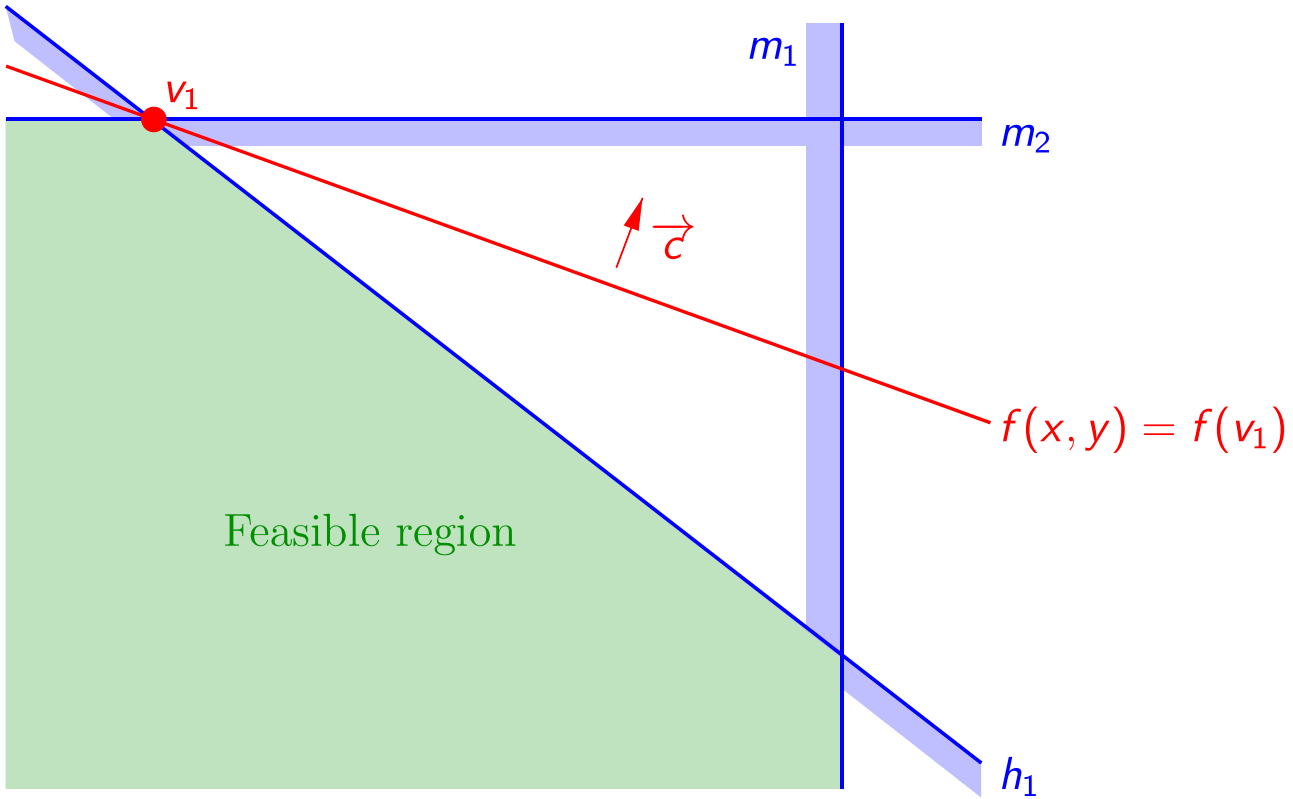
Seidel's LP Algorithm

Two-dimensional case



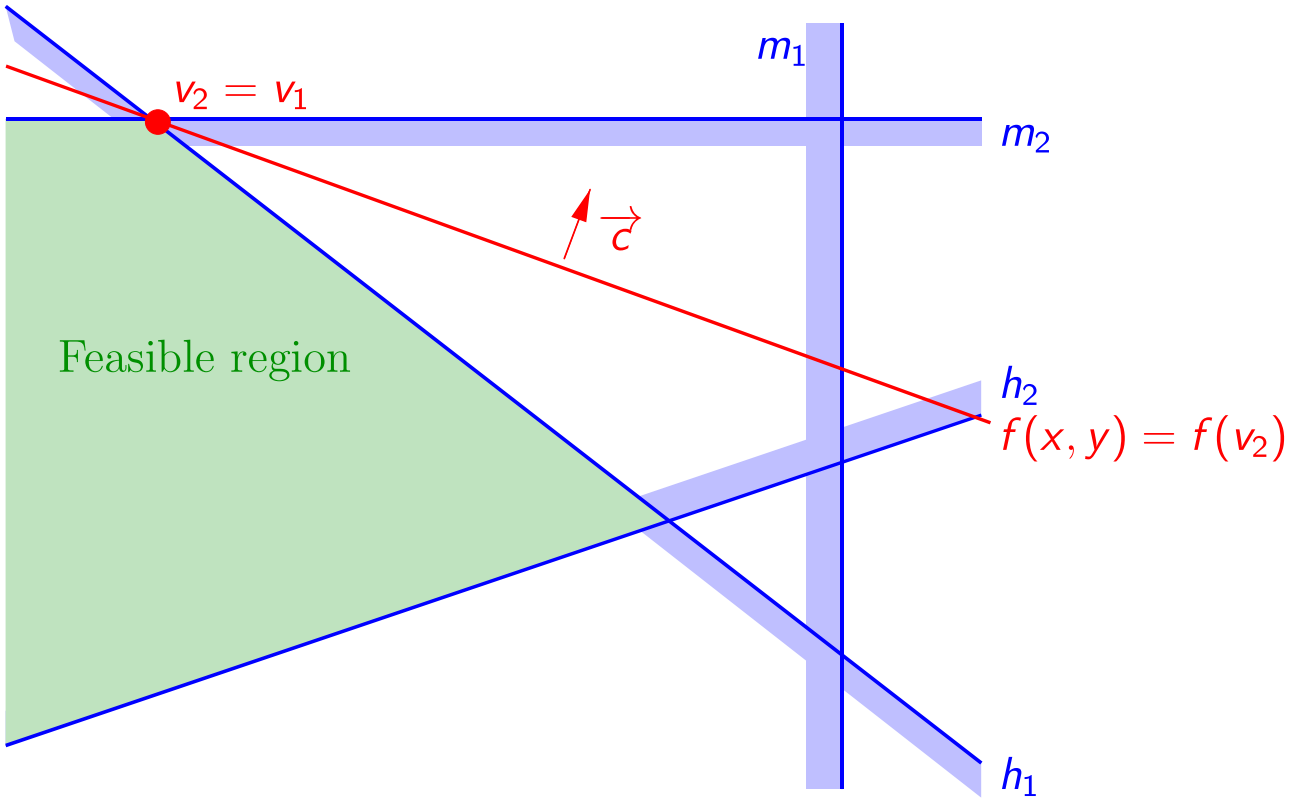
Seidel's LP Algorithm

Two-dimensional case



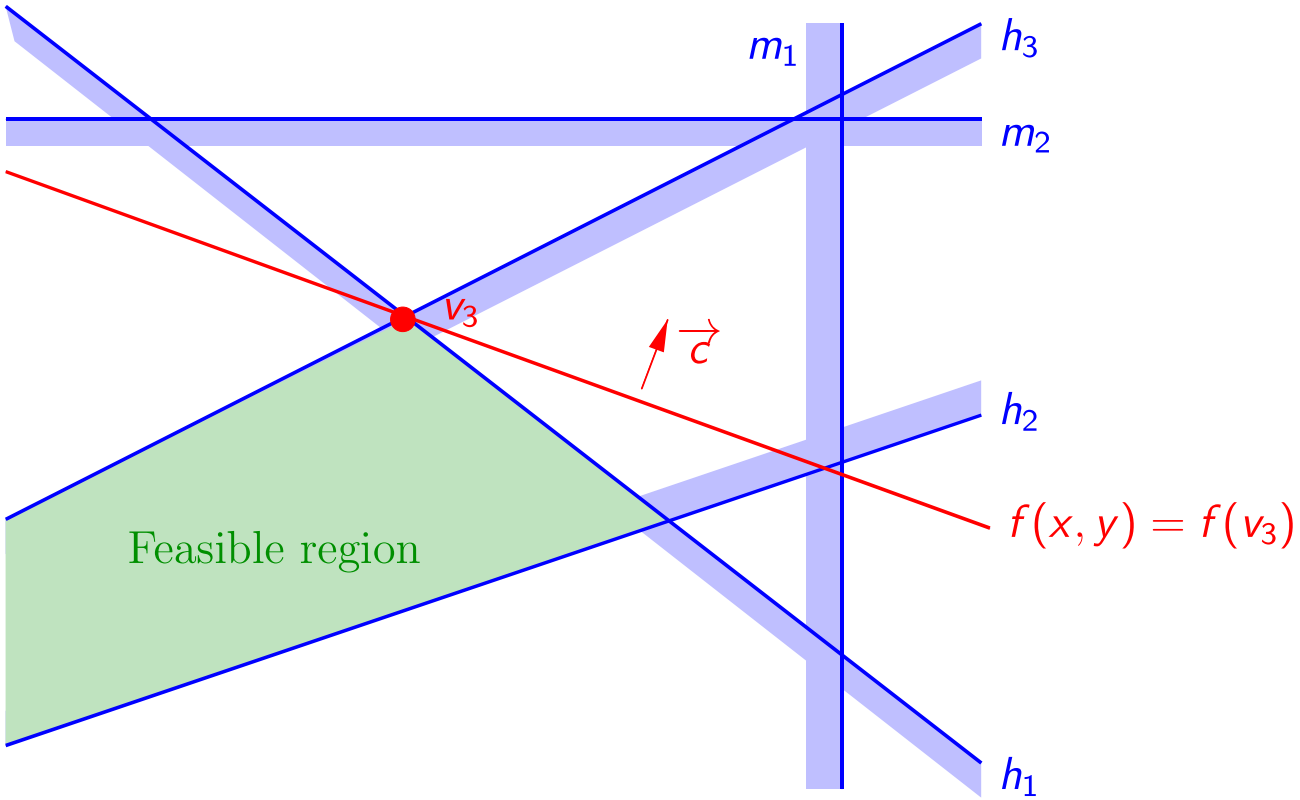
Seidel's LP Algorithm

Two-dimensional case



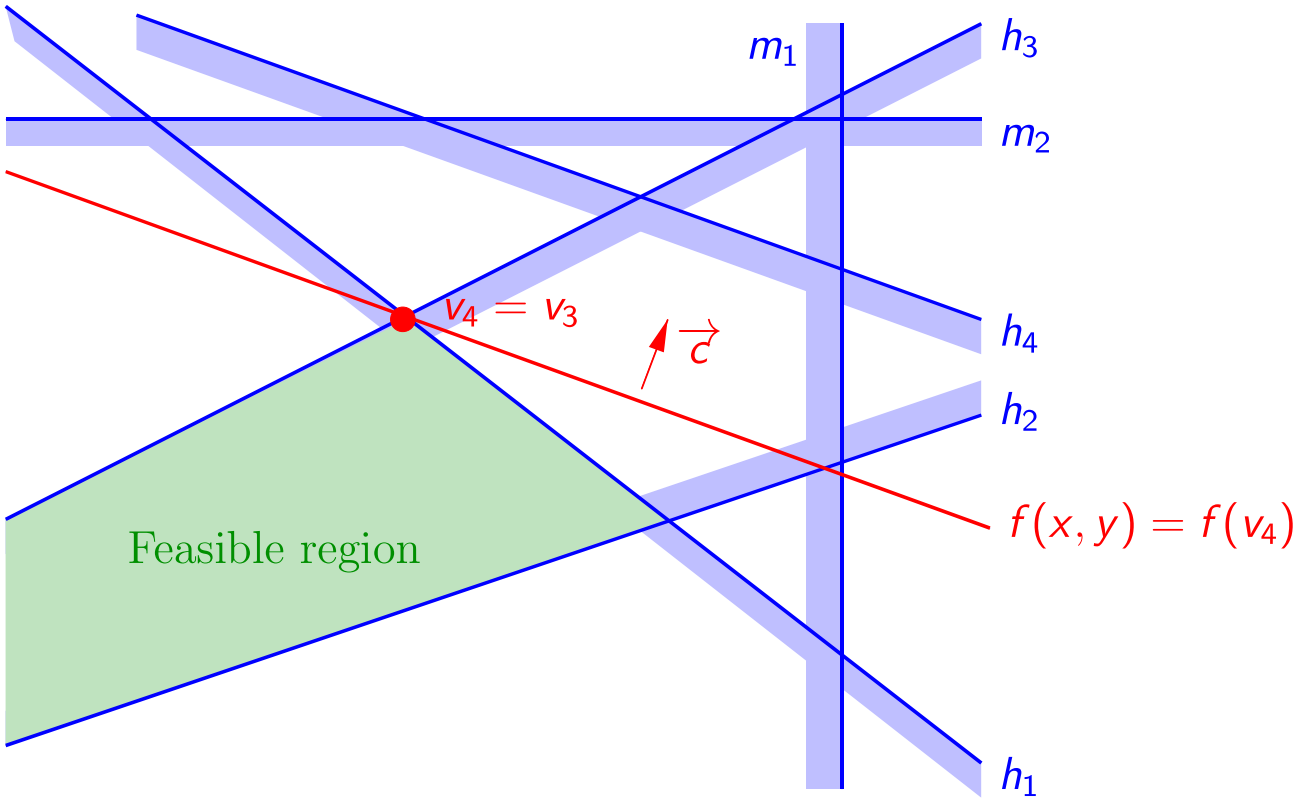
Seidel's LP Algorithm

Two-dimensional case



Seidel's LP Algorithm

Two-dimensional case

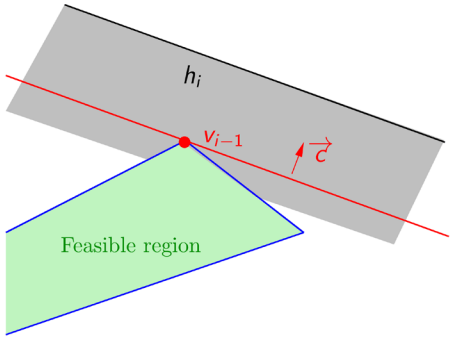


Seidel's LP Algorithm

Two-dimensional case

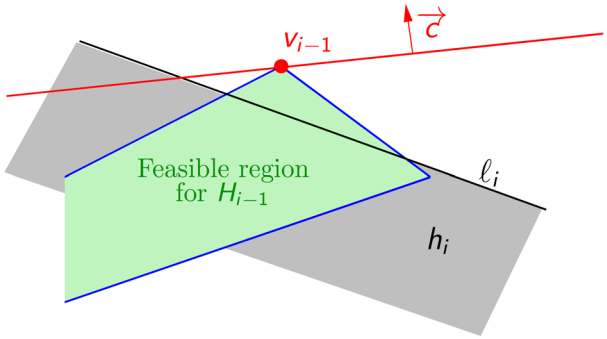
Incremental Algorithm:
Before inserting h_i, we only assume that we know v_{i-1}.
How can we find v_i?

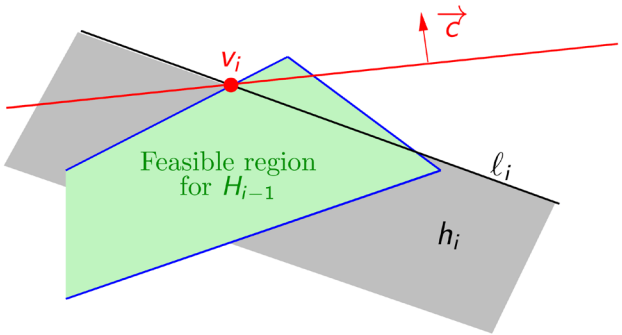
First case: $v_{i-1} \in h_i$



Then $v_i = v_{i-1}$

Second case: $v_{i-1} \notin h_i$





Then $v_i \neq v_{i-1}$, but we have $v_i \in \ell_i$, thus resorting to **one-dimensional case**.

Seidel's LP Algorithm

Complexity Analysis

First case is done in $O(1)$ time: Just check whether $v_{i-1} \in h_i$.

Second case in $O(i)$ time: One dimensional LP with $O(i)$ constraints.

We define a random variable X_i assuming input to be a random permutation:

- $X_i = 0$ in the first case ($v_i = v_{i-1}$).
- $X_i = 1$ in the second case ($v_i \neq v_{i-1}$).

So the expected running time is

$$\begin{aligned} E[T(n)] &= O\left(\sum_{i=1}^{n} 1 + i \cdot E[X_i]\right) \\ &= O\left(\sum_{i=1}^{n} 1 + i \cdot \frac{2}{i}\right) \\ &= O(n) \end{aligned}$$

We randomize the input order to achieve linear complexity w.r.t constraint number.

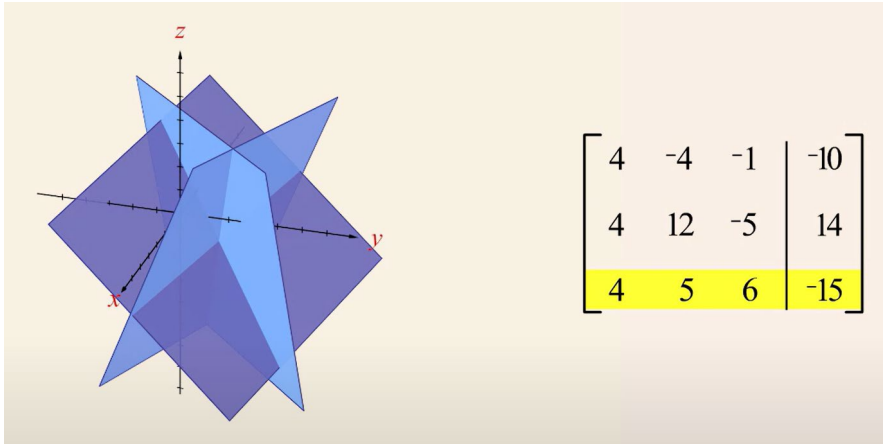
Seidel's LP Algorithm

Seidel's Exact LP Algorithm for d-dimensional case

SeidelLP$(\mathcal{H}, c)$
  if $\dim(c) = 1$
    $x \leftarrow \text{OneDimLP}(\mathcal{H}, c)$
  end if
  $\mathcal{I} \leftarrow \{\}$
  for $h \in \mathcal{H}$ in a random order
    if $x \notin h$
      $\mathcal{H}' \leftarrow \text{GaussElim}(\mathcal{I}, h)$
      $c' \leftarrow \text{GaussElim}(c, h)$
      $x \leftarrow \text{SeidelLP}(\mathcal{H}', c')$
    end if
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{h\}$
  end for
return x

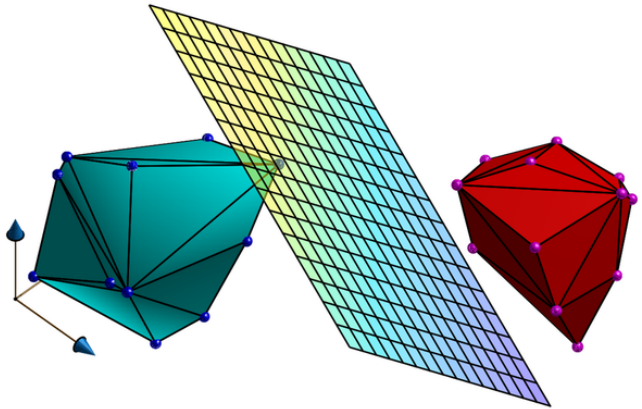
Complexity: $O(d!n)$

If dimension is fixed as $d \in \{1, \ldots, 9\}$,
the complexity is $O(n)$, i.e., linear in constraint number.

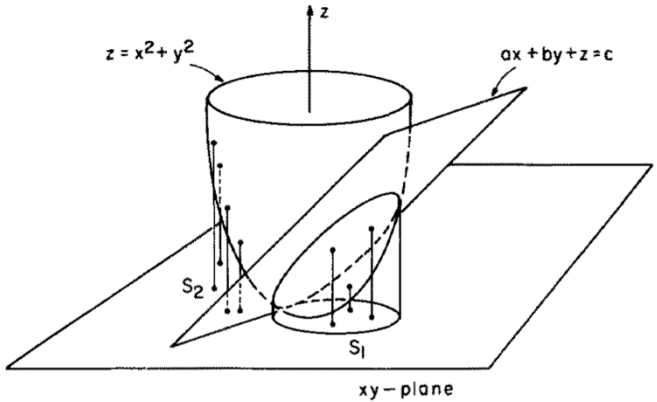


Gaussian elimination reduce 1 dim before recursion.
The dim with largest abs. coeff. ensures stability (Pivoted LU).

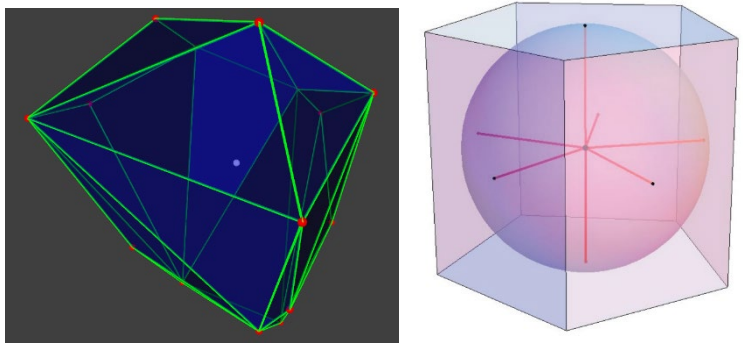
Seidel's LP Algorithm: Applications



Linear Separability



Circular Separability



Chebyshev Center

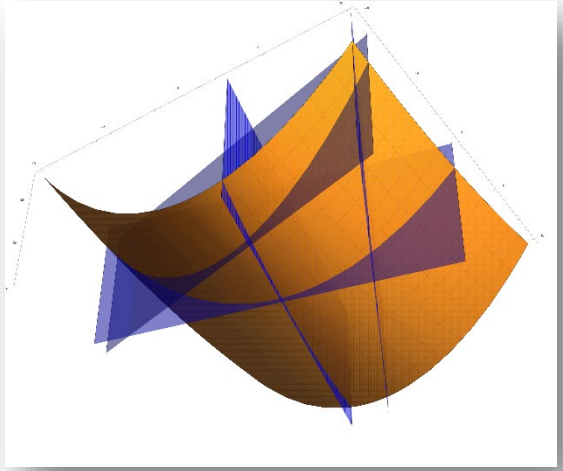
$$\min_{\bar{x}\in\mathbb{R}^{n+1}} -\bar{x}^{\mathrm{T}} e_{n+1},\ \ s.t.\ \ (\mathbf{A},\mathbf{1})\,\bar{x} \preceq b$$

Low-Dimensional QP

Low-Dimensional QP

Consider the strictly convex low-dimensional QP

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}x^{\mathrm{T}}M_{\mathcal{Q}}x + c_{\mathcal{Q}}^{\mathrm{T}}x, \text{ s.t. } A_{\mathcal{Q}}x \le b_{\mathcal{Q}}$$



Strict convexity implies $M_{\mathcal{Q}} \succ 0$ whose Cholesky factorization exists.

$$M_{\mathcal{Q}} = L_{\mathcal{Q}}L_{\mathcal{Q}}^{\mathrm{T}}$$

This QP is equivalent to a minimum-norm problem for $y \in \mathbb{R}^n$ via

$$y = L_{\mathcal{Q}}^{\mathrm{T}}x + L_{\mathcal{Q}}^{-1}c_{\mathcal{Q}} \quad \text{or} \quad x = L_{\mathcal{Q}}^{-\mathrm{T}}y - \left(L_{\mathcal{Q}}L_{\mathcal{Q}}^{\mathrm{T}}\right)^{-1}c_{\mathcal{Q}}$$

Thus the original QP is equivalent to a much simpler problem below:

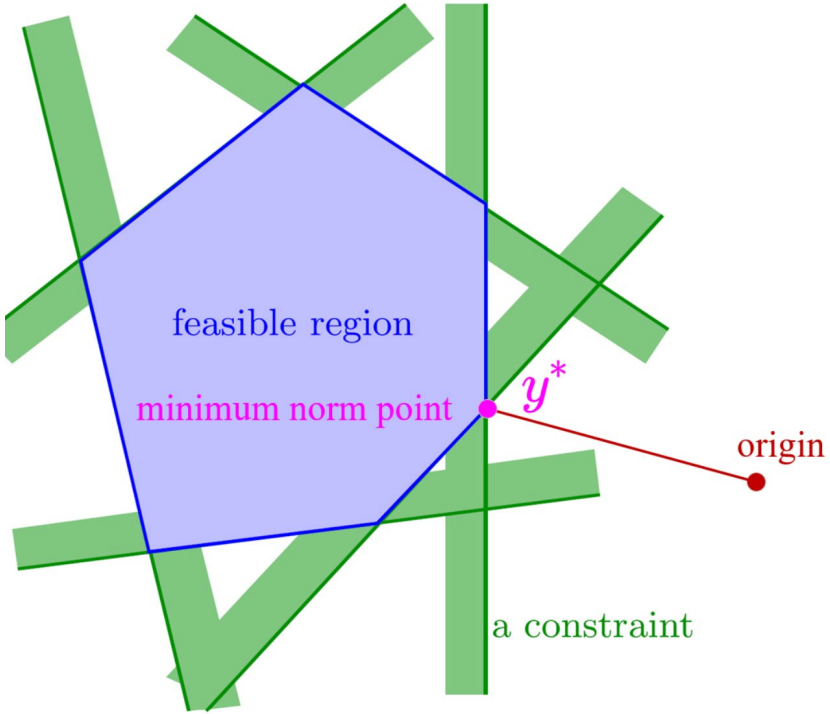
$$\min_{y\in\mathbb{R}^n} \frac{1}{2}y^{\mathrm{T}}y, \text{ s.t. } Ey \le f \quad \text{where} \quad E = A_{\mathcal{Q}}L_{\mathcal{Q}}^{-\mathrm{T}},\ f = A_{\mathcal{Q}}\left(L_{\mathcal{Q}}L_{\mathcal{Q}}^{\mathrm{T}}\right)^{-1}c_{\mathcal{Q}} + b_{\mathcal{Q}}$$

Find a point in the polytope that is closest to the origin.

Low-Dimensional QP

Find a point in the polytope that is closest to the origin.

$$\min_{y \in \mathbb{R}^n} \frac{1}{2} y^{\mathrm{T}} y, \text{ s.t. } Ey \leq f$$



Low-Dimensional QP

One-dimensional case

Minimize the objective function

$$f(x) = \frac{1}{2}x^2$$

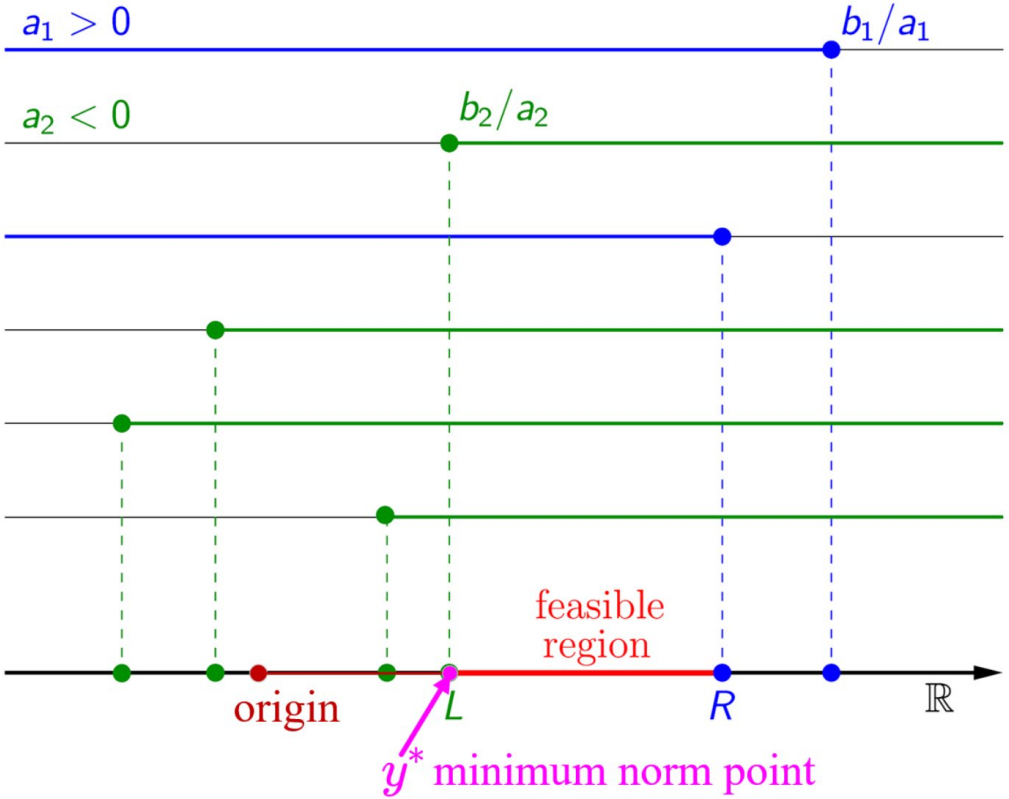
under the constraints

$$\begin{aligned} a_1 x &\leqslant b_1 \\ a_2 x &\leqslant b_2 \\ \vdots \quad &\quad \vdots \\ a_n x &\leqslant b_n \end{aligned}$$

It is also trivial to solve this problem in linear time.

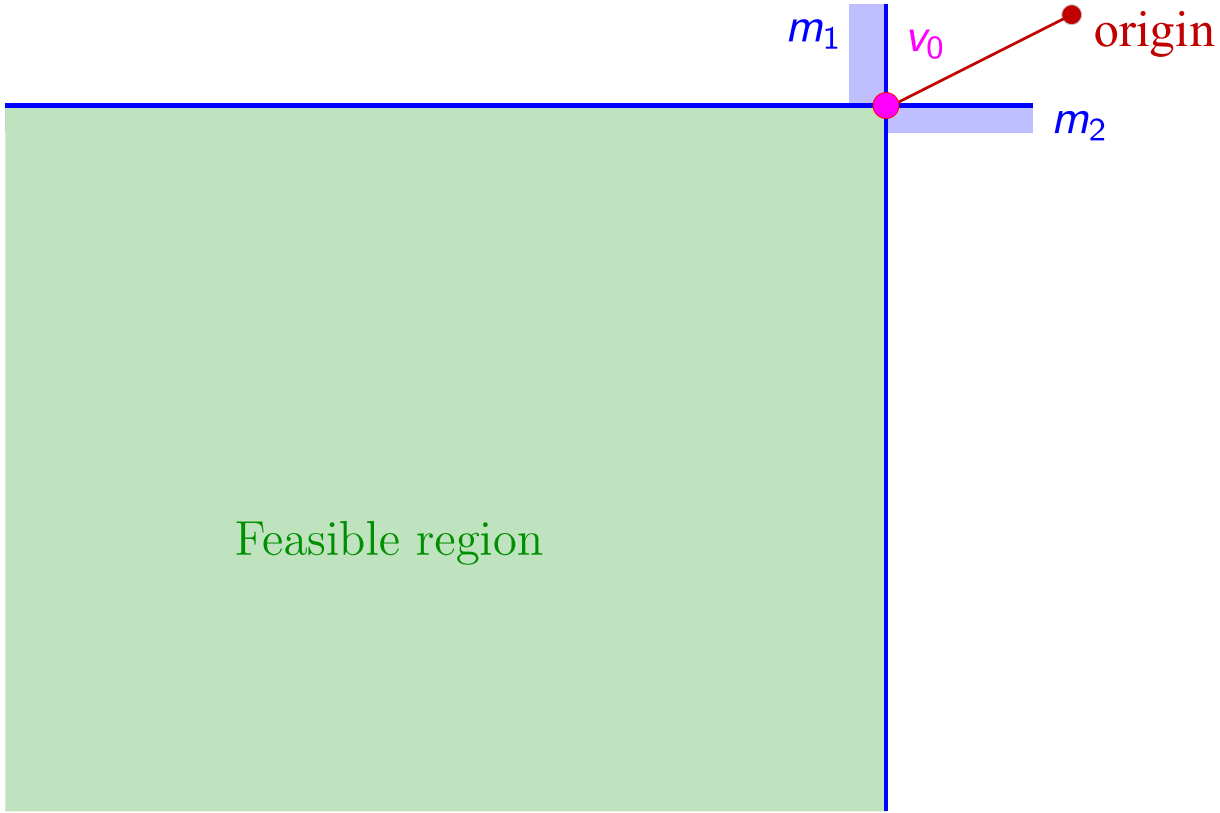
Low-Dimensional QP

One-dimensional case

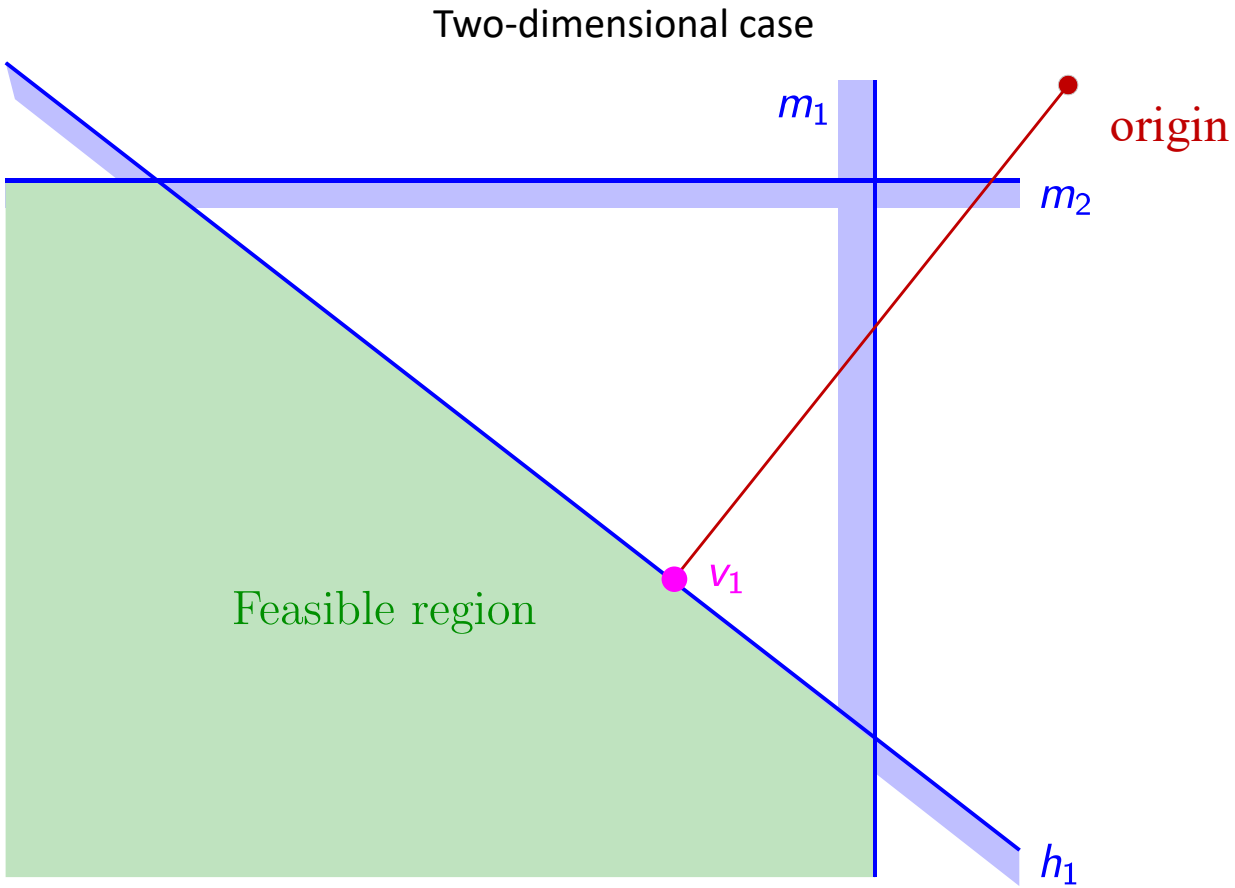


Low-Dimensional QP

Two-dimensional case

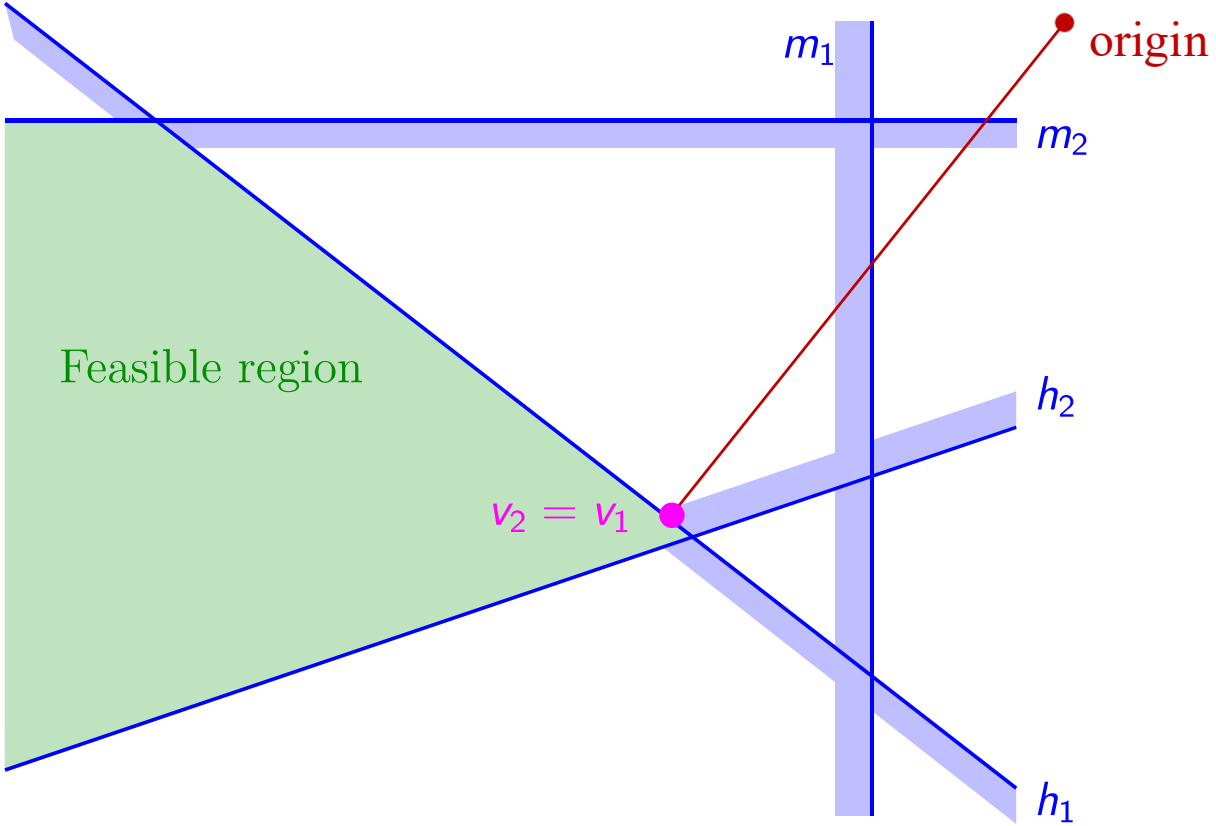


Low-Dimensional QP

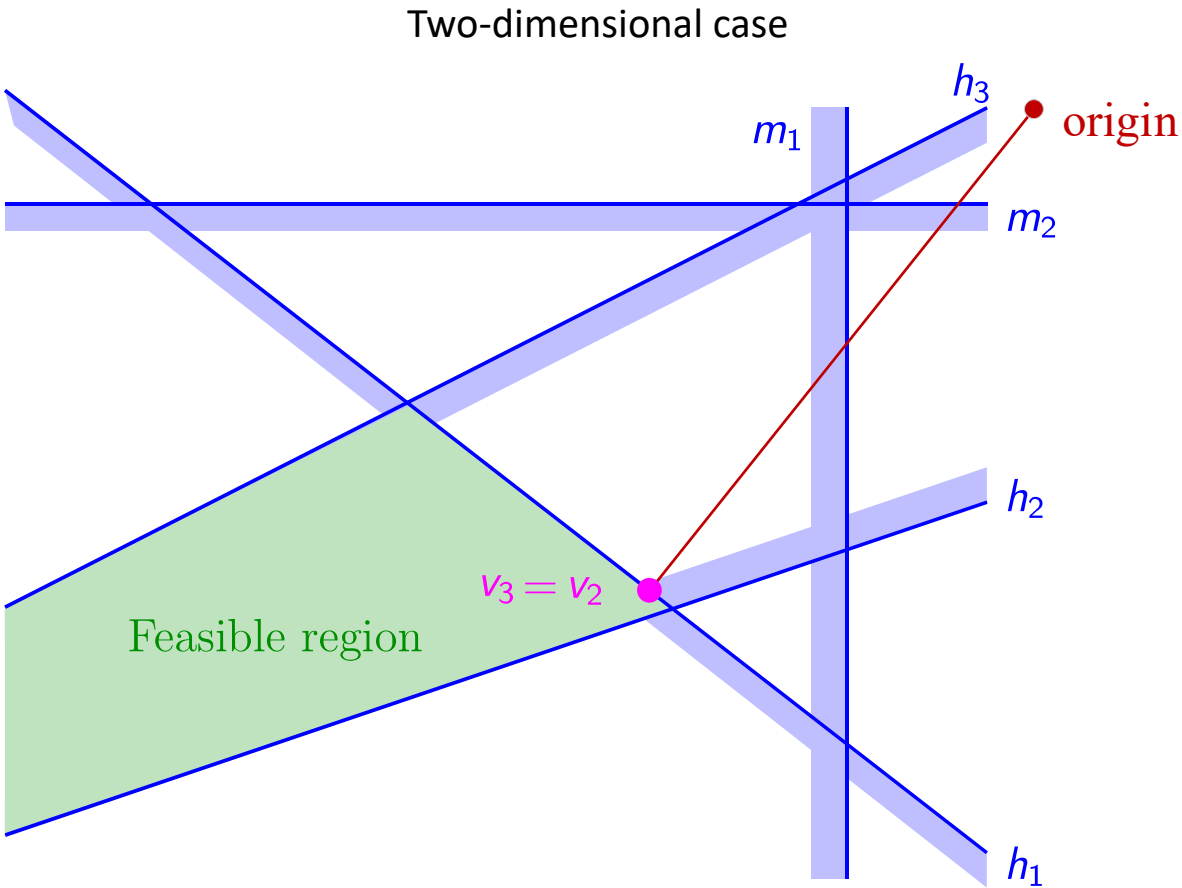


Low-Dimensional QP

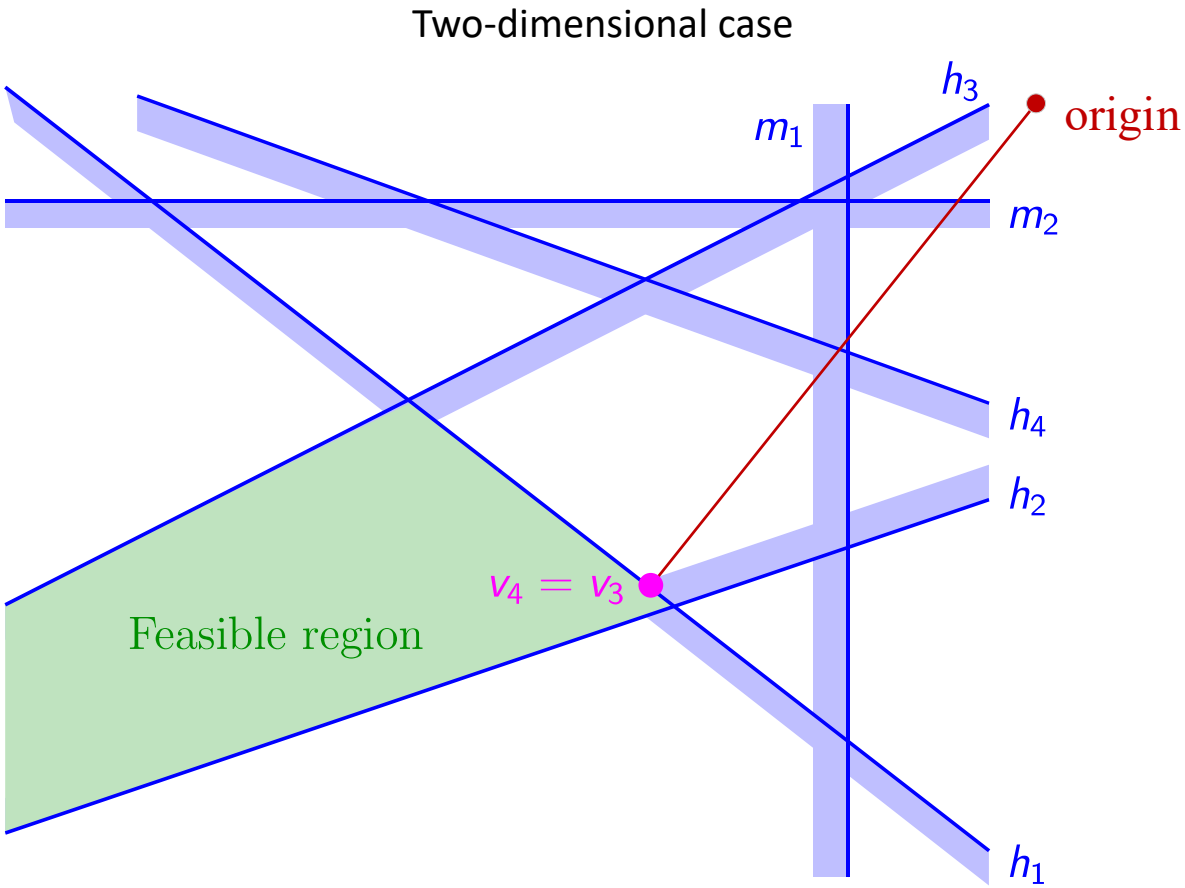
Two-dimensional case



Low-Dimensional QP



Low-Dimensional QP



Low-Dimensional QP

Exact Minimum Norm Algorithm for d-dimensional case

```
LowDimMinNorm(H)
  if dim(H) = 1
    y ← OneDimMinNorm(H)
  end if
  I ← {}
  for h ∈ H in a random order
    if y ∉ h
        {M, v, H'} ← HouseholderProj(I, h)
        y' ← LowDimMinNorm(H')
        y ← My' + v
    end if
    I ← I ∪ {h}
  end for
return y
```

Complexity: $O(d!n)$ again!

If dimension is fixed as $d \in \{1, \ldots, 9\}$,
the complexity is $O(n)$, i.e., linear in constraint number.

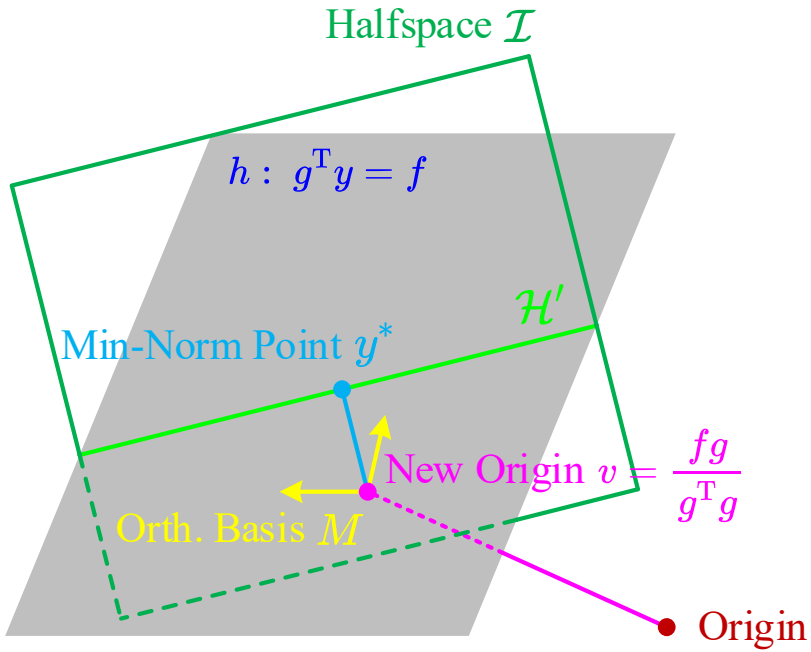
Need to construct a lower dimensional min-norm problem:

h: a hyperplane containing the solution
$v \in \mathbb{R}^d$: the minimum-norm point on h
$M \in \mathbb{R}^{d \times (d-1)}$: an orthonormal basis of h
$\mathcal{H}'$: $(d-1)$-dimensional inequalities on h

Low-Dimensional QP

Exact Minimum Norm Algorithm for d-dimensional case

LowDimMinNorm($\mathcal{H}$)
  if $\dim(\mathcal{H}) = 1$
    $y \leftarrow$ OneDimMinNorm($\mathcal{H}$)
  end if
  $\mathcal{I} \leftarrow \{\}$
  for $h \in \mathcal{H}$ in a random order
    if $y \notin h$
      $\{M, v, \mathcal{H}'\} \leftarrow$ HouseholderProj($\mathcal{I}, h$)
      $y' \leftarrow$ LowDimMinNorm($\mathcal{H}'$)
      $y \leftarrow My' + v$
    end if
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{h\}$
  end for
return y



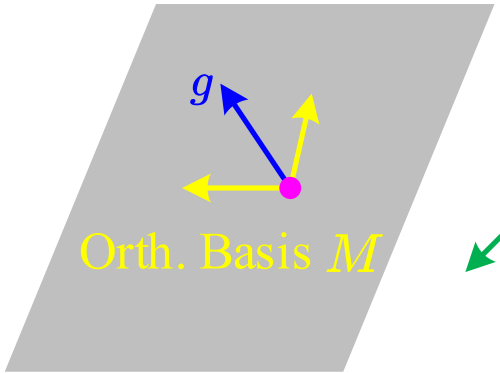
$$\|My'\|^2 + \|v - o\|^2 = \|y'\|^2 + \|v - o\|^2 = \|y^*\|^2$$

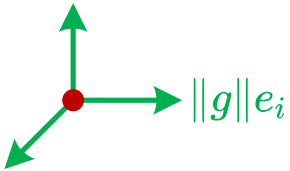
How to find the orthonormal basis on $h: g^{\mathrm{T}}y = f$?

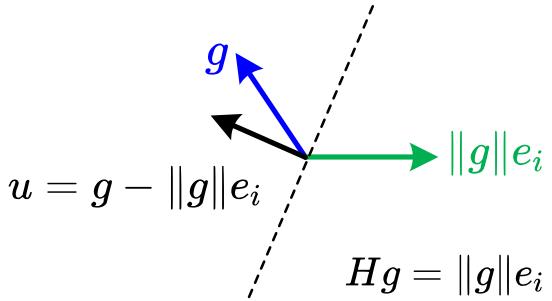
Low-Dimensional QP

How to find the orthonormal basis on $h: g^{\mathrm{T}}y = f$?

All vectors in the orth. basis of h is perpendicular to g.







H is exactly the Householder reflection (orthonormal) matrix
H has $(d-1)$ orthonormal row vectors that are perpendicular to g
All $(d-1)$ column vectors of M are these $(d-1)$ row vectors of H
Choosing $-\operatorname{sgn}(g_i)\|g\|e_i$ and $i = \operatorname{argmax}_k |g_k|$ is numerically stable

$$H = I_d - \frac{2uu^{\mathrm{T}}}{u^{\mathrm{T}}u}$$

Low-Dimensional QP

Exact Minimum Norm Algorithm for n-dimensional case

```
LowDimMinNorm(H)
  if dim(H) = 1
    y ← OneDimMinNorm(H)
  end if
  I ← {}
  for h ∈ H in a random order
    if y ∉ h
      {M, v, H'} ← HouseholderProj(I, h)
      y' ← LowDimMinNorm(H')
      y ← My' + v
    end if
    I ← I ∪ {h}
  end for
return y
```

The solution of QP is thus recovered by

$$x = L_{\mathcal{Q}}^{-\mathrm{T}} y - \left(L_{\mathcal{Q}} L_{\mathcal{Q}}^{\mathrm{T}}\right)^{-1} c_{\mathcal{Q}}$$

The complexity of low-dim QP is also $O(n)$,
i.e., linear in constraint number.

Constrained Optimization via Sequential Unconstrained Optimization

Penalty Method

L_2-Penalty Function: Equality Constrained Case

A optimization problem with only equality constraints:

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

The penalized function of this optimization problem is :

$$P_E(x, \sigma) = f(x) + \frac{1}{2}\sigma \sum_{i \in \mathcal{E}} c_i^2(x)$$

The red part is called quadratic penalty function, where σ is the penalty weight.

Intuitively speaking,

$$\lim_{\sigma \to +\infty} \operatorname{argmin}_x P_E(x, \sigma) = \operatorname{argmin}_x f(x), \text{s.t. } c_i(x) = 0, \ i \in \mathcal{E}$$

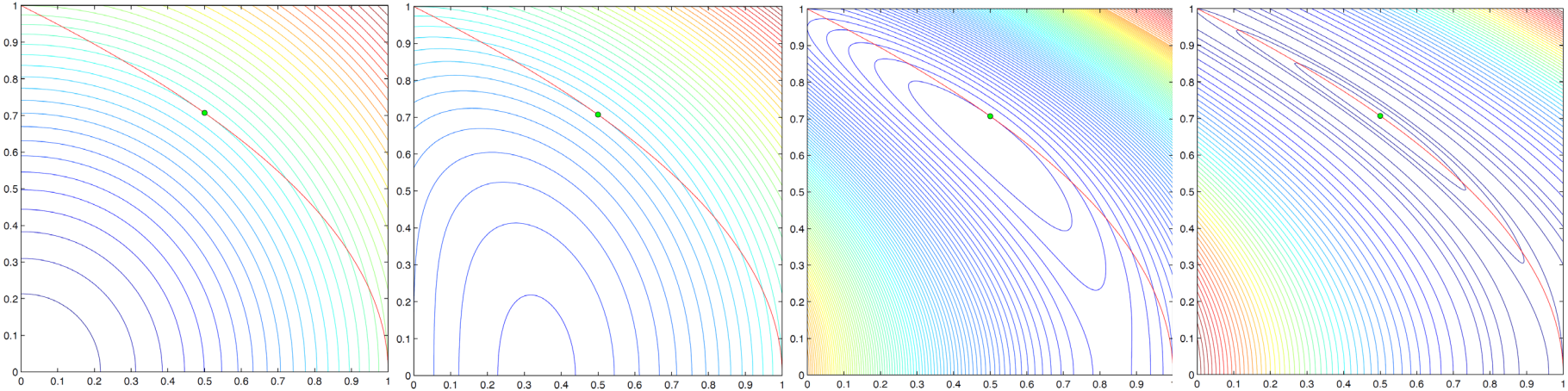
Penalty Method

Geometric Interpretation

An optimization problem and its penalized function:

$$\begin{aligned} \min_x \quad & x_1^2 + x_2^2 \\ \text{s.t.} \quad & x_1 + x_2^2 = 1 \end{aligned}$$

$$P_E(x, \sigma) = x_1^2 + x_2^2 + \frac{\sigma}{2}(x_1 + x_2^2 - 1)^2$$



$\sigma = 0.01$ $\sigma = 1$ $\sigma = 10$ $\sigma = 100$

The unconstrained minimum approaches the constrained one as penalty weight grows.

Penalty Method

L_2-Penalty Function: Inequality Constrained Case

A optimization problem with inequality constraints:

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

The penalized function of this optimization problem is :

$$P_I(x, \sigma) = f(x) + \frac{1}{2}\sigma \sum_{i \in \mathcal{I}} \max[c_i(x), 0]^2$$

The red part is also quadratic penalty function, while its 2nd order derivative is discontinuous.

Intuitively speaking,

$$\lim_{\sigma \to +\infty} \operatorname{argmin}_x P_I(x, \sigma) = \operatorname{argmin}_x f(x), \text{ s.t. } c_i(x) \leq 0, i \in \mathcal{I}$$

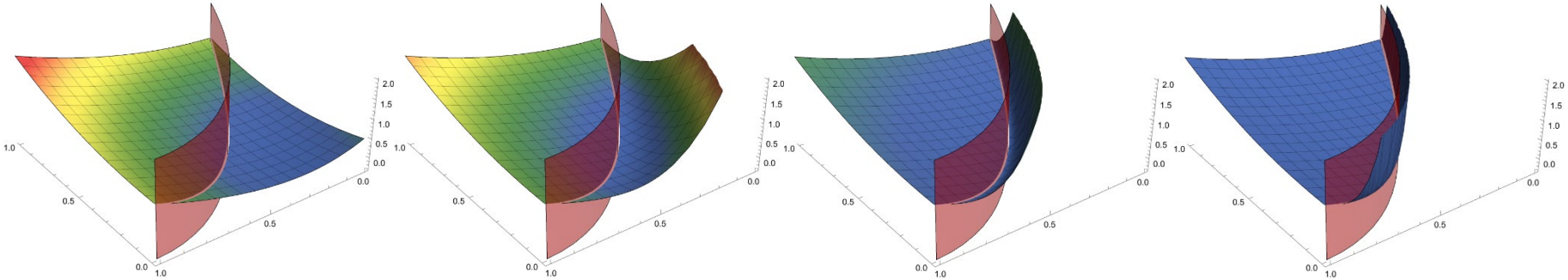
Penalty Method

Geometric Interpretation

An optimization problem and its penalized function:

$$\begin{aligned} \min_x \quad & x_1^2 + x_2^2 \\ \text{s.t.} \quad & (1 - x_2)^2 - x_1 \leq 0 \end{aligned}$$

$$P_I(x, \sigma) = x_1^2 + x_2^2 + \frac{\sigma}{2} \max[(1 - x_2)^2 - x_1, 0]^2$$



$\sigma = 1$ $\sigma = 10$ $\sigma = 100$ $\sigma = 1000$

The unconstrained minimum still approaches the constrained one as penalty weight grows.

Penalty Method

L_1-Penalty Function: Exactness

A optimization problem with general constraints:

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

The exactly penalized function of this problem is :

$$P(x, \sigma) = f(x) + \sigma \sum_{i \in \mathcal{E}} |c_i(x)| + \sigma \sum_{j \in \mathcal{I}} \max[c_j(x), 0]$$

The red part is L1 penalty function, while its derivative is discontinuous.

In return, we have

$$\exists M \in \mathbb{R}_{>0},\ \forall \sigma > M,\ \ \operatorname{argmin}_x P(x, \sigma) = \operatorname{argmin}_x f(x), \text{s.t. } c_i(x) = 0,\ i \in \mathcal{E},\ c_j(x) \leq 0,\ j \in \mathcal{I}$$

Intuitively, we only require a sufficiently large but finite weight for **exact** solutions.

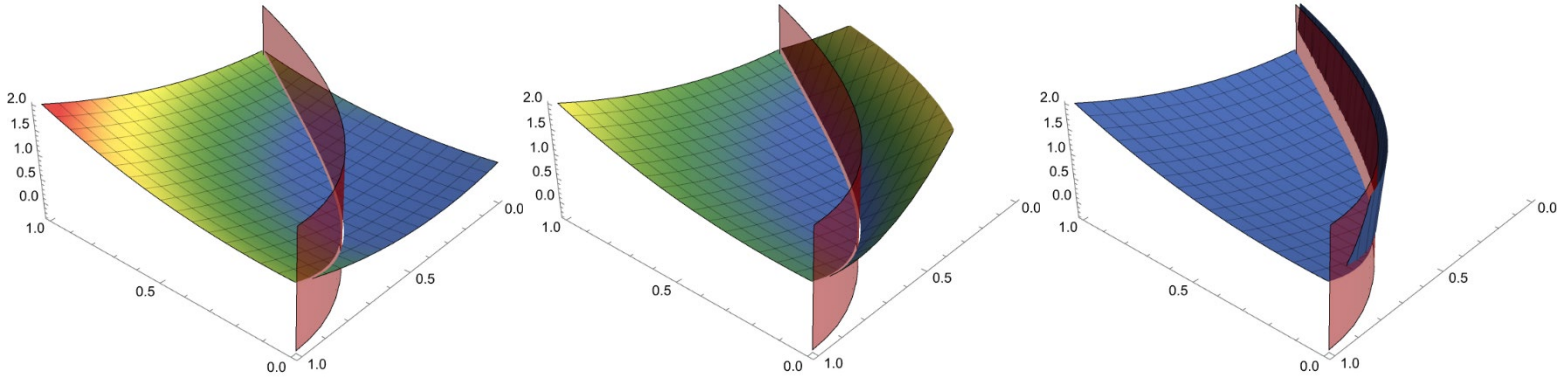
Penalty Method

Geometric Interpretation

An optimization problem and its exactly penalized function:

$$\begin{aligned} \min_x \quad & x_1^2 + x_2^2 \\ \text{s.t.} \quad & (1 - x_2)^2 - x_1 \leq 0 \end{aligned}$$

$$P(x, \sigma) = x_1^2 + x_2^2 + \sigma \max[(1 - x_2)^2 - x_1, 0]$$



$\sigma = 1$ $\sigma = 10$ $\sigma = 100$

The penalized function is nonsmooth, while $\sigma = 10$ suffices for the exact solution.

Barrier Method

Barrier Function

A optimization problem with inequality constraints:

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

The formulation of barrier method is

logarithmic barrier $$B_{\ln}(x, \sigma) = f(x) - \sigma \sum_{i \in \mathcal{I}} \ln(-c_i(x))$$

inverse barrier $$B_{\text{inv}}(x, \sigma) = f(x) + \sigma \sum_{i \in \mathcal{I}} \text{inv}(-c_i(x)), \ \text{inv}(x) := 1/x \text{ if } x > 0$$

exponential barrier $$B_{\text{expi}}(x, \sigma) = f(x) + \sigma \sum_{i \in \mathcal{I}} \text{expi}(-c_i(x)), \ \text{expi}(x) := e^{1/x} \text{ if } x > 0$$

Intuitively speaking,

$$\lim_{\sigma \to 0^+} \text{argmin}_x \, B(x, \sigma) = \text{argmin}_x \, f(x), \text{s.t. } c_i(x) \leq 0, i \in \mathcal{I}$$

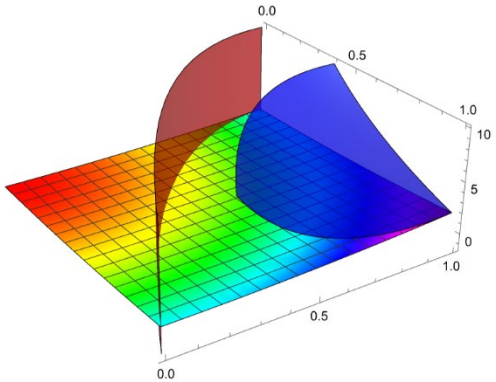
Barrier Method

Geometric Interpretation

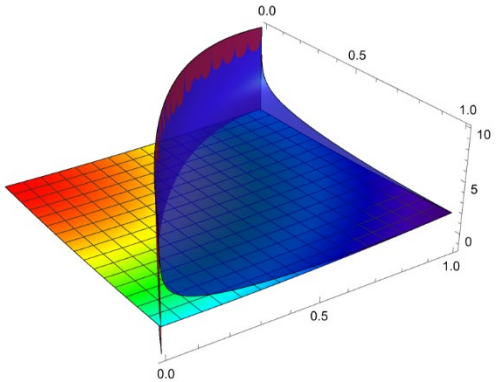
An optimization problem and barrier method formulation:

$$\begin{aligned} \min_x \quad & x_1^2 + x_2^2 \\ \text{s.t.} \quad & (1 - x_2)^2 - x_1 \leq 0 \end{aligned}$$

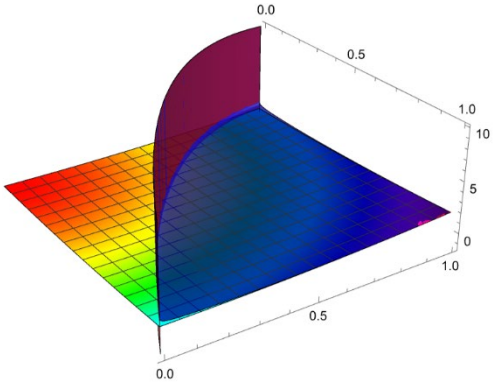
$$B(x, \sigma) = x_1^2 + x_2^2 - \sigma \ln[x_1 - (1 - x_2)^2]$$



$\sigma = 10$



$\sigma = 1$

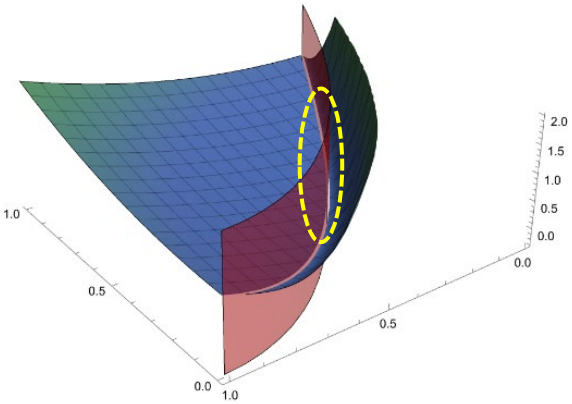


$\sigma = 0.1$

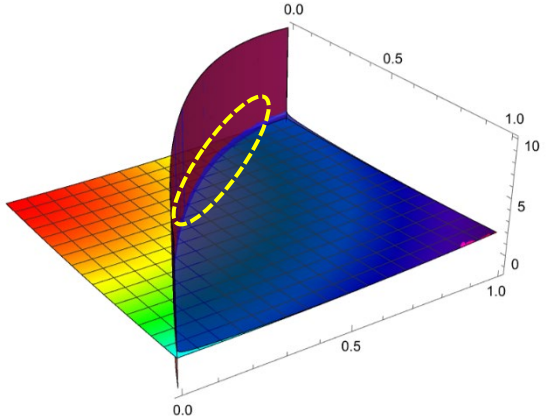
The unconstrained minimum approaches the constrained one as barrier weight decays.

Ill-Conditioning Penalty/Barrier Functions

The curvature blows up!



inexact/exact penalty



barrier

Largest singular value of Hessian is unbounded as penalty (barrier) weight increases (decays).

Lagrangian Relaxation

A convex optimization problem with only equality constraints:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & Ax = b \end{aligned}$$

The Lagrange function or Lagrangian of this problem is:

$$\mathcal{L}(x, \lambda) := f(x) + \langle \lambda, Ax - b \rangle$$

Obviously,

$$\max_{\lambda} f(x) + \langle \lambda, Ax - b \rangle = \begin{cases} f(x), Ax - b = 0 \\ \infty, \text{ otherwise} \end{cases}$$

Thus the optimization is equivalent to

$$\min_x f(x), \text{ s.t. } Ax = b \quad \Longleftrightarrow \quad \min_x \max_{\lambda} \mathcal{L}(x, \lambda)$$

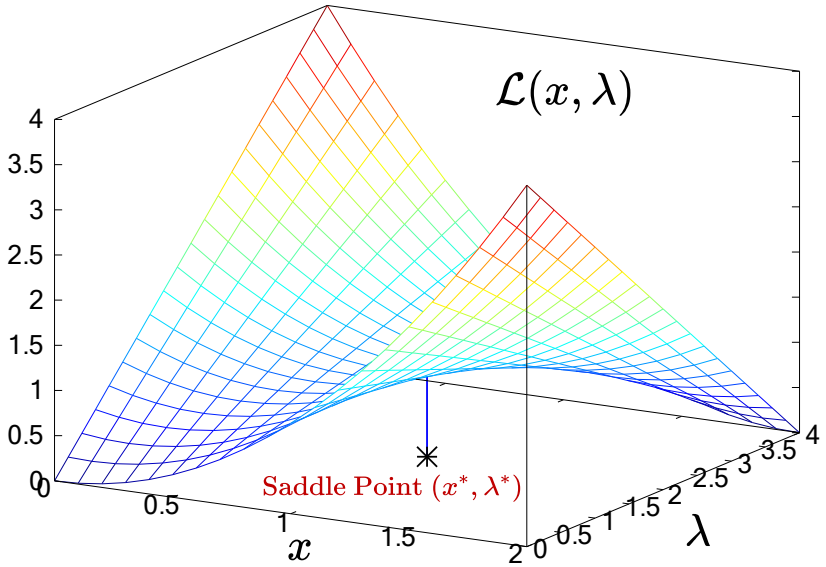
The constrained optimum is exactly the “saddle” point of the Lagrangian.

Lagrangian Relaxation

Geometric Interpretation

$$\begin{aligned} \min \quad & x^2 \\ \text{s.t.} \quad & 1 - x = 0 \end{aligned}$$

$$\mathcal{L}(x, \lambda) := x^2 + \lambda(1 - x)$$



How to compute the saddle point?

Lagrangian Relaxation

Uzawa's Method

$\min_x \max_\lambda \mathcal{L}(x, \lambda) := f(x) + \langle \lambda, Ax - b \rangle$ Solution of red subproblem is non-continuous.

$\max_\lambda \min_x \mathcal{L}(x, \lambda) := f(x) + \langle \lambda, Ax - b \rangle$ Solution of blue subproblem is easy if r.h.s. is ***strictly convex***.

Although $\max_\lambda \min_x \mathcal{L}(x, \lambda) \leq \min_x \max_\lambda \mathcal{L}(x, \lambda)$, von Neumann says it's tight for continuous convex $f(x)$.

Thus we directly maximize the dual function via gradient ascent.

$$d(\lambda) := \min_x f(x) + \langle \lambda, Ax - b \rangle$$

$$\begin{cases} x^{k+1} = \operatorname{argmin}_x \mathcal{L}(x, \lambda^k) & \text{Compute the dual function.} \\ \lambda^{k+1} = \lambda^k + \alpha(Ax^{k+1} - b) & \text{Gradient ascent.} \end{cases}$$

Fan K. Minimax theorems. Proceedings of the National Academy of Sciences. 1953 Jan;39(1):42-7.

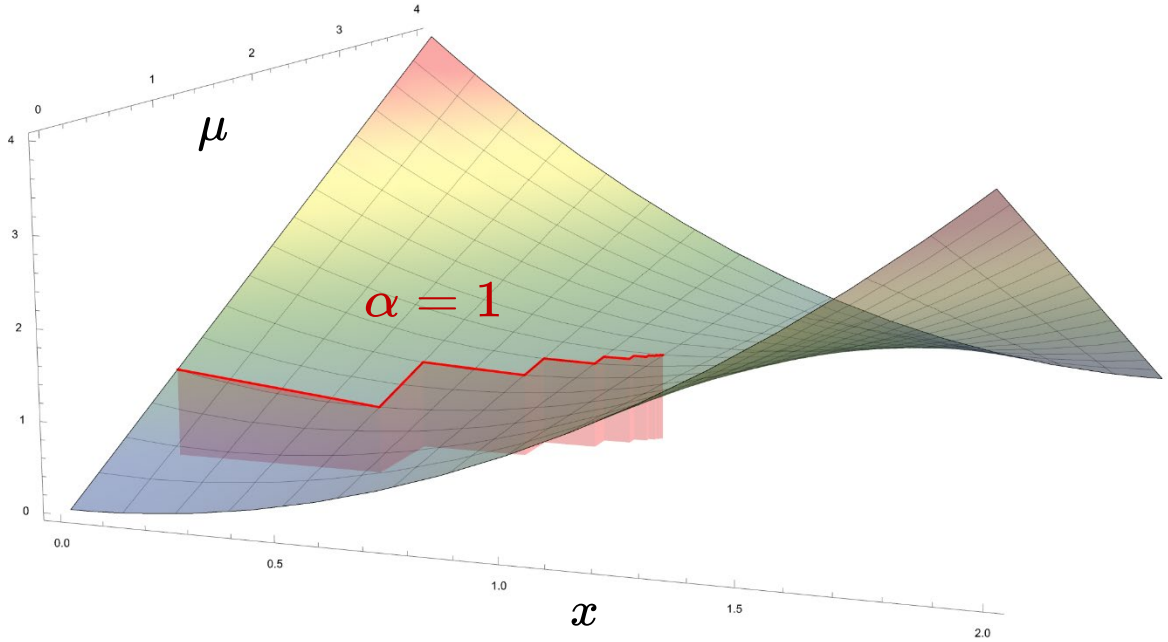
Lagrangian Relaxation

Geometric Interpretation of Uzawa's Method

$$\begin{aligned} \min \quad & x^2 \\ \text{s.t.} \quad & 1 - x = 0 \end{aligned}$$

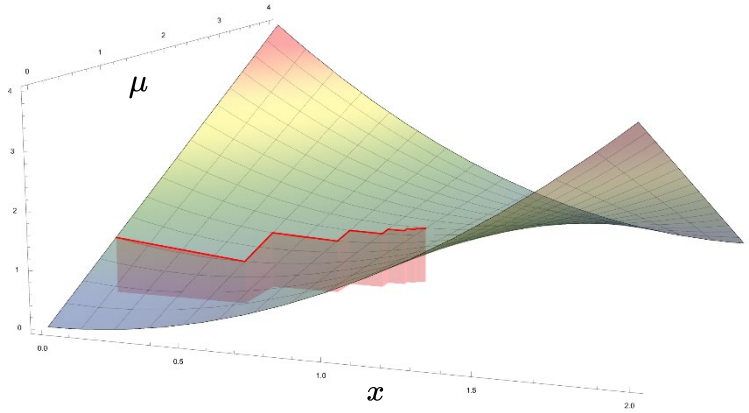
$$\mathcal{L}(x, \lambda) := x^2 + \lambda(1 - x)$$

$$\begin{cases} x^{k+1} = \underset{x}{\operatorname{argmin}}\, x^2 + \lambda^k (1 - x) \\ \lambda^{k+1} = \lambda^k + \alpha(1 - x^{k+1}) \end{cases}$$



Lagrangian Relaxation

Drawbacks of Uzawa's Method



- The primal optimization problem should be convex.
- Lagrangian should be strictly convex w.r.t. primal vars.
- Step size of dual ascent needs tuning.
- Convergence rate is not satisfactory.

$$\begin{cases} x^{k+1} = \underset{x}{\operatorname{argmin}}\, \mathcal{L}(x, \lambda^k) \\ \lambda^{k+1} = \lambda^k + \alpha(Ax^{k+1} - b) \end{cases}$$

Generally Constrained Optimization

KKT Conditions

Karush–Kuhn–Tucker (KKT) Conditions

Given general problem

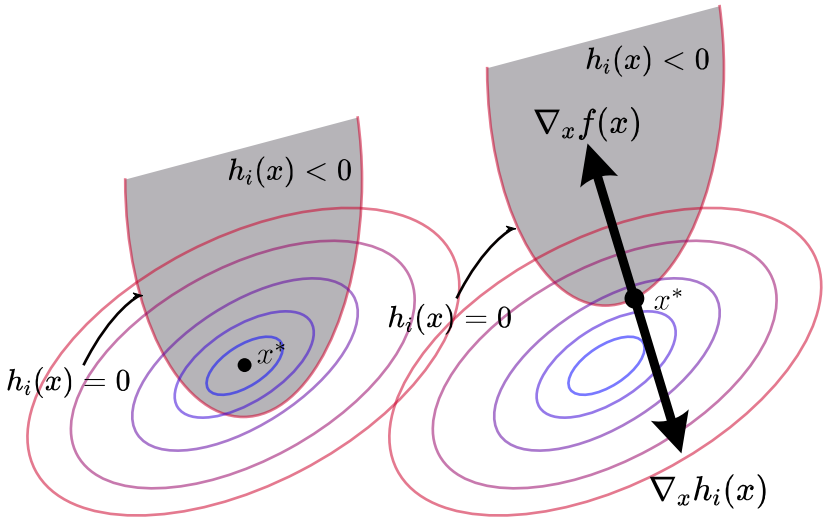
$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h_i(x) \le 0,\ i = 1, \dots, m \\ & \ell_j(x) = 0,\ j = 1, \dots, r \end{aligned}$$

If it is not degenerate (e.g. it is convex but without strictly feasible point), its optimal solution satisfies

- (stationarity) $0 \in \partial_x [f(x) + \sum_{i=1}^{m} u_i h_i(x) + \sum_{j=1}^{r} v_j \ell_j(x)]$
- $(\text{complementary slackness})$ $u_i \cdot h_i(x) = 0,\ i = 1, \dots, m$
- $(\text{primal feasibility})$ $h_i(x) \le 0,\ \ell_j(x) = 0,\ i = 1, \dots, m,\ j = 1, \dots, r$
- $(\text{dual feasibility})$ $u_i \ge 0,\ i = 1, \dots, m$

KKT Conditions

Geometric Interpretation



(stationarity) $0 \in \partial_x [f(x) + \sum_{i=1}^{m} u_i h_i(x) + \sum_{j=1}^{r} v_j \ell_j(x)]$

(complementary slackness) $u_i \cdot h_i(x) = 0,\ i = 1, \ldots, m$

(primal feasibility) $h_i(x) \leq 0,\ \ell_j(x) = 0,\ i = 1, \ldots, m,\ j = 1, \ldots, r$

(dual feasibility) $u_i \geq 0,\ i = 1, \ldots, m$

KKT Conditions

Example

A strictly convex QP with only equality constraints:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & \frac{1}{2} x^T Q x + c^T x \\ \text{s.t.} \ & Ax = b \end{aligned}$$

KKT conditions imply:

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ v^* \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix}$$

Only need to solve a linear system!

PHR Augmented Lagrangian Method

Powell-Hestenes-Rockafellar (PHR) Method for Equality-Constrained Cases

Consider an optimization with only equality constraints:

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

Uzawa's method is to conduct dual gradient ascent on the dual function

$$d(\lambda) := \min_x f(x) + \lambda^{\mathrm{T}} h(x)$$

If the Lagrangian is not strictly convex w.r.t. x, the dual function is then nonsmooth.
In this case, its gradient may not exist, making the dual gradient ascent problematic.

$$\nabla d(\lambda) \text{ does not necessarily exist!}$$

In PHR method, the minimax is smooothed by a "proximal point" term, penalizing deviations from a prior $\bar{\lambda}$ ($\rho > 0$).

$$\min_x \max_\lambda \ f(x) + \lambda^{\mathrm{T}} h(x) - \frac{1}{2\rho} \|\lambda - \bar{\lambda}\|^2$$

Now we can directly optimize the minimax problem instead of the interchanged one.

PHR Augmented Lagrangian Method

PHR Method for Equality-Constrained Cases

$$\max_{\lambda} \; f(x) + \lambda^{\mathrm{T}} h(x) - \frac{1}{2\rho}\|\lambda - \bar{\lambda}\|^2$$

The inner problem of the smoothed minimax problem becomes trivial

$$\lambda^*(\bar{\lambda}) = \bar{\lambda} + \rho h(x)$$

Now the outer problem can be directly conducted via unconstrained optimization

$$\begin{aligned}
&\min_{x}\max_{\lambda} \; f(x) + \lambda^{\mathrm{T}} h(x) - \frac{1}{2\rho}\|\lambda - \bar{\lambda}\|^2 \\
=&\min_{x} \; f(x) + \lambda^*(\bar{\lambda})^{\mathrm{T}} h(x) - \frac{1}{2\rho}\|\lambda^*(\bar{\lambda}) - \bar{\lambda}\|^2 \\
=&\min_{x} \; f(x) + (\bar{\lambda} + \rho h(x))^{\mathrm{T}} h(x) - \frac{\rho}{2}\|h(x)\|^2 \\
=&\min_{x} \; f(x) + \bar{\lambda}^{\mathrm{T}} h(x) + \frac{\rho}{2}\|h(x)\|^2
\end{aligned}$$

This only solves an approximation of the minimax (original problem). How to increase precision?

1. Reduce the approx. weight $1/\rho$
2. Update the prior value $\bar{\lambda} \leftarrow \lambda^*(\bar{\lambda})$

PHR Augmented Lagrangian Method

PHR Method for Equality-Constrained Cases

$$
\begin{aligned}
x &\leftarrow \arg\min_x \; f(x) + \bar{\lambda}^{\mathrm{T}} h(x) + \frac{\rho}{2}\|h(x)\|^2 \\
\bar{\lambda} &\leftarrow \bar{\lambda} + \rho h(x)
\end{aligned}
$$

This new procedure yields the Uzawa's method with fixed stepsize for

$$
\min_x \max_\lambda \; f(x) + \frac{\rho}{2}\|h(x)\|^2 + \lambda^{\mathrm{T}} h(x)
$$

Obviously, the corresponding primal problem becomes

$$
\begin{aligned}
\min_{x\in\mathbb{R}^n} \;& f(x) + \frac{\rho}{2}\|h(x)\|^2 \\
\text{s.t.} \;& h(x) = 0
\end{aligned}
$$

This problem is fully equivalent to the original one even if the optimization is nonconvex!

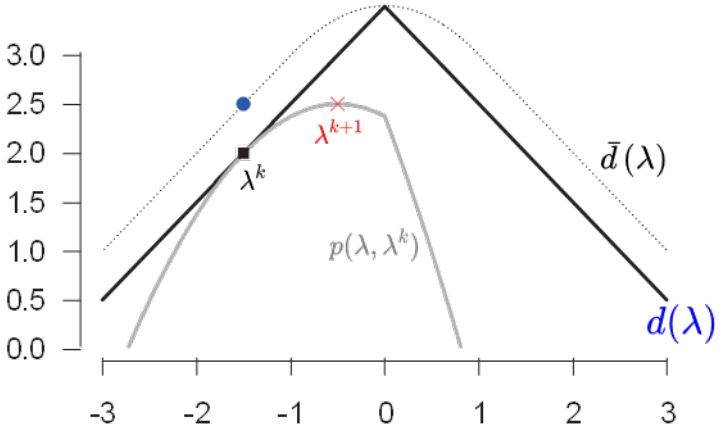
Its Lagrangian, also called the Augmented Lagrangian of the original problem, is

$$
\mathcal{L}(x,\lambda;\rho) := \underbrace{f(x) + \lambda^{\mathrm{T}} h(x)}_{\text{Lagrangian}} + \underbrace{\frac{\rho}{2}\|h(x)\|^2}_{\text{Augmentation}}
$$

PHR Augmented Lagrangian Method

PHR Method as Dual Smoothing

Consider the PHR-ALM in a convex problem's interchanged (dual) problem $d(\lambda) := \min_x f(x) + \lambda^{\mathrm{T}} h(x)$



$$p(\gamma, \lambda) = d(\gamma) - \frac{1}{2\rho}\|\gamma - \lambda\|^2$$ Maximizing it yields updating the dual variable.

$$\bar{d}(\lambda) = \max_{\gamma} p(\gamma, \lambda)$$ A smooth surrogate dual function with invariant maximum.

The update of dual variable in PHR method is indeed maximizing a smooth surrogate dual function!

PHR Augmented Lagrangian Method

PHR Method for Equality-Constrained Cases

As for the original nonconvex equality-constrained optimization

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

A more frequently used equivalent form of its PHR Augmented Lagrangian is

$$\mathcal{L}_\rho(x,\lambda) := f(x) + \frac{\rho}{2}\left\| h(x) + \frac{\lambda}{\rho} \right\|^2 - \frac{1}{2\rho}\|\lambda\|^2$$

The KKT solution can be solved via

$$\begin{cases} x^{k+1} = \operatorname{argmin}_x \mathcal{L}_{\rho^k}(x,\lambda^k) \\ \lambda^{k+1} = \lambda^k + \rho^k h\left(x^{k+1}\right) \\ \rho^{k+1} = \min[(1+\gamma)\rho^k, \beta] \end{cases}$$

- ρ^k can be any nondecreasing positive sequence, implying $\gamma \geq 0, \beta > 0, \rho^0 > 0$.
- Inner-loop unconstrained opt is well-behaved and can be solved inexactly.

PHR Augmented Lagrangian Method

PHR Method for Inequality-Constrained Cases

Consider the nonconvex inequality constrained problem

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

We rewrite it as

$$\begin{aligned} \min_{x\in\mathbb{R}^n, s\in\mathbb{R}^m} \ & f(x) \\ \text{s.t.} \ & g(x) + [s]^2 = 0 \end{aligned}$$

$[\cdot]^2$ implies element-wise squaring

The primal descent of augmented Lagrangian implies

$$\min_{x\in\mathbb{R}^n, s\in\mathbb{R}^m} f(x) + \frac{\rho}{2}\left\| g(x) + [s]^2 + \frac{\lambda}{\rho}\right\|^2 = \min_{x\in\mathbb{R}^n}\min_{s\in\mathbb{R}^m} f(x) + \frac{\rho}{2}\left\| g(x) + [s]^2 + \frac{\lambda}{\rho}\right\|^2 = \min_{x\in\mathbb{R}^n} f(x) + \frac{\rho}{2}\left\| \max\left[g(x) + \frac{\lambda}{\rho}, 0\right]\right\|^2$$

Thus we can directly define the augmented Lagrangian for inequality-constrained problem as:

$$\mathcal{L}_\rho(x,\mu) := f(x) + \frac{\rho}{2}\left\| \max\left[g(x) + \frac{\mu}{\rho}, 0\right]\right\|^2 - \frac{1}{2\rho}\|\mu\|^2$$

The dual ascent $\mu^{k+1} = \mu^k + \rho\big(g(x^{k+1}) + [s^{k+1}]^2\big)$ can also be conducted in closed form:

$$\mu^{k+1} = \max\left[\mu^k + \rho g\big(x^{k+1}\big),\ 0\right]$$

PHR Augmented Lagrangian Method

PHR Augmented Lagrangian Method (PHR-ALM) for General Nonconvex Cases

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

Its PHR Augmented Lagrangian is defined as

$$\mathcal{L}_\rho(x,\lambda,\mu) := f(x) + \frac{\rho}{2}\left\{ \left\| h(x) + \frac{\lambda}{\rho} \right\|^2 + \left\| \max\left[g(x) + \frac{\mu}{\rho}, 0 \right] \right\|^2 \right\} - \frac{1}{2\rho}\left\{ \|\lambda\|^2 + \|\mu\|^2 \right\}$$

where $\rho > 0, \mu \succeq 0$. The PHR-ALM is simply repeating the primal descent + dual ascent iterations.

$$\begin{cases} x \leftarrow \operatorname{argmin}_x \mathcal{L}_\rho(x,\lambda,\mu) \\ \lambda \leftarrow \lambda + \rho h(x) \\ \mu \leftarrow \max[\mu + \rho g(x), 0] \\ \rho \leftarrow \min[(1+\gamma)\rho, \ \beta] \end{cases}$$

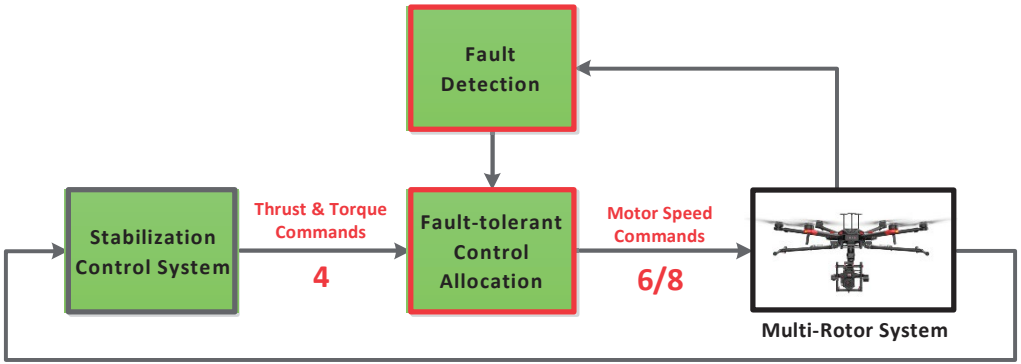
$\gamma = 0$ or keep a constant ρ is OK

- How to choose parameters? $\rho_{\text{ini}} = 1, \lambda_{\text{ini}} = \mu_{\text{ini}} = 0, \gamma = 1, \beta = 10^3$
- What is the inner opt prec.? $\|\nabla_x \mathcal{L}_\rho(x,\lambda,\mu)\|_\infty < \xi^k \min\left[1, \max\left[\|h(x)\|_\infty, \left\| \max\left[g(x), -\frac{\mu}{\rho}\right] \right\|_\infty\right]\right]$ with positive ξ^k converging to 0
- What is the stop criterion? $\max\left[\|h(x)\|_\infty, \left\| \max\left[g(x), -\frac{\mu}{\rho}\right] \right\|_\infty\right] < \epsilon_{\text{cons}}, \ \left\|\nabla_x \mathcal{L}_\rho(x,\lambda,\mu)\right\|_\infty < \epsilon_{\text{prec}}$

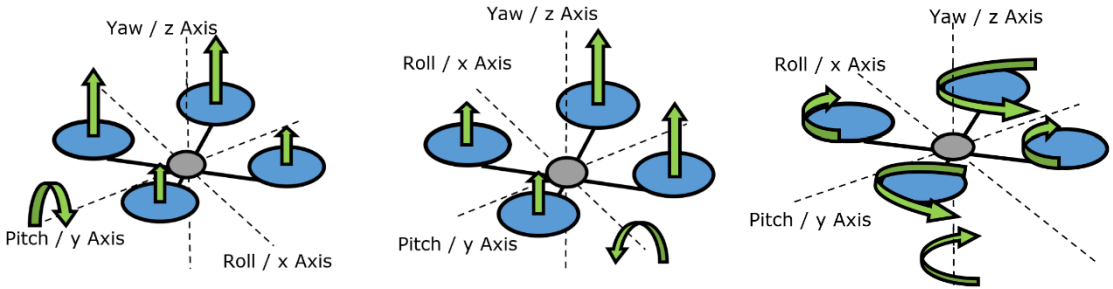
Application 1:
Control Allocation Problem

Control Allocation Problem

Control allocation for over-actuated system



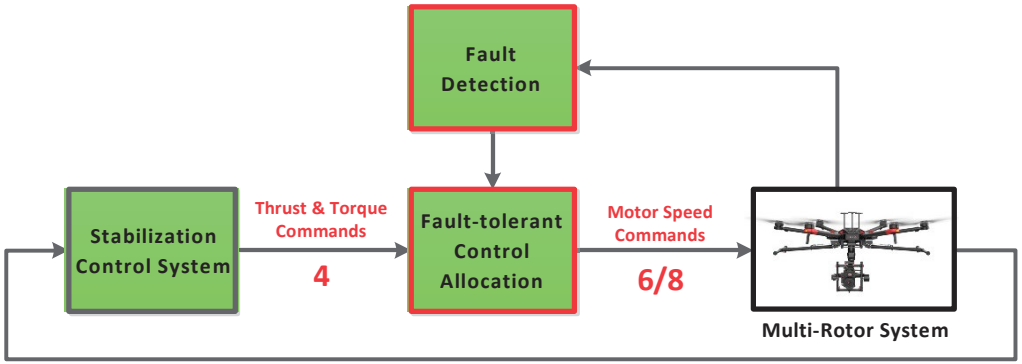
How does a quadrotor produce total thrust + 3D torque?

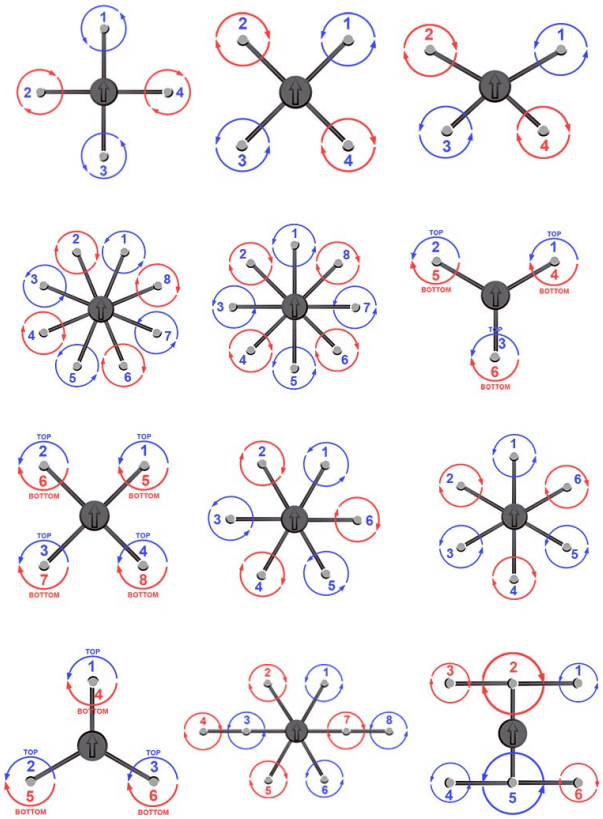


Four rotors suffice for total thrust + roll/pitch/yaw torque!

Control Allocation Problem

Control allocation for over-actuated system





Why more rotors?

In case of rotor failure!

How to compute redundant inputs?

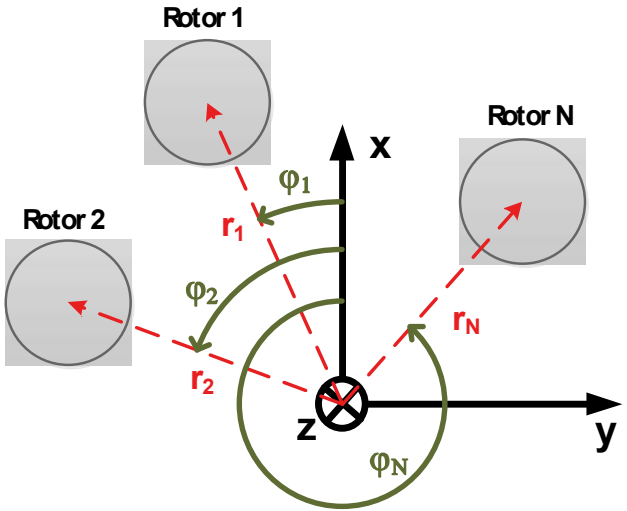
Control allocation!

Determine the best rotor speed for the expected thrust/torque.

How to formulate this into an opt. ?

Control Allocation Problem

Control allocation as constrained optimization



Thrust and yaw torque of the i-th rotor where no failure occurs:

$$F_i = \eta_i \cdot \mu \cdot \omega_i^2$$
$$T_i = -\eta_i \cdot \mathrm{sign}(\omega_i) \cdot \kappa \cdot \omega_i^2$$

The actual total thrust and roll/pitch/yaw torque is computed as

$$\begin{bmatrix} f_{\text{total}} \\ \tau_{\text{roll}} \\ \tau_{\text{pitch}} \\ \tau_{\text{yaw}} \end{bmatrix} = \begin{bmatrix} t_1 & \cdots & t_N \\ l_1 & \cdots & l_N \\ m_1 & \cdots & m_N \\ n_1 & \cdots & n_N \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \vdots \\ \omega_N^2 \end{bmatrix}$$

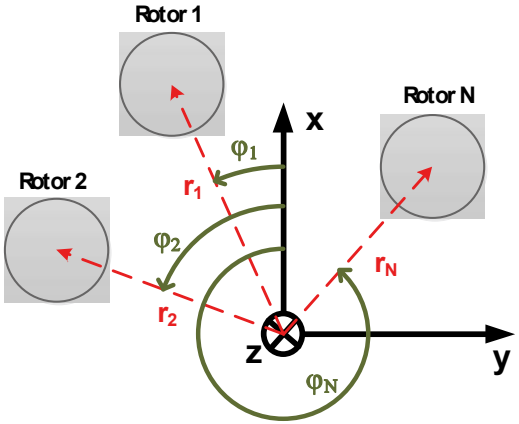
where the constant matrix is

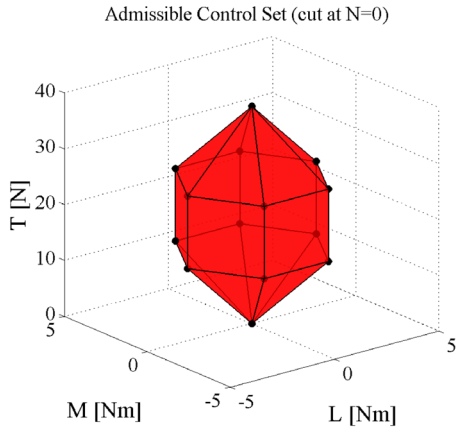
$$t_i = \eta_i \cdot \mu$$
$$l_i = \eta_i \cdot \mu \cdot r_i \cdot \sin(\varphi_i)$$
$$m_i = \eta_i \cdot \mu \cdot r_i \cdot \cos(\varphi_i)$$
$$n_i = -\eta_i \cdot \kappa \cdot \mathrm{sign}(\omega_i)$$

Formulating control allocation as optimization:

- minimize error between ref/actual thrust/torque
- reduce the discontinuous jump of the rotor speeds
- allocation must be admissible and satisfy actuator limits

Control Allocation Problem





Control allocation as constrained optimization

Actuator limits $\omega_{\min}^2 \leq \omega_i^2 \leq \omega_{\max}^2,\ \forall\, i \in \{1, \ldots, N\}$

Error between ref/actual thr./torq.

$$\left\| \begin{bmatrix} \bar{f}_{\text{total}} \\ \bar{\tau}_{\text{roll}} \\ \bar{\tau}_{\text{pitch}} \\ \bar{\tau}_{\text{yaw}} \end{bmatrix} - \begin{bmatrix} t_1 & \cdots & t_N \\ l_1 & \cdots & l_N \\ m_1 & \cdots & m_N \\ n_1 & \cdots & n_N \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \vdots \\ \omega_N^2 \end{bmatrix} \right\|_W^2$$

Jump between last/current speed

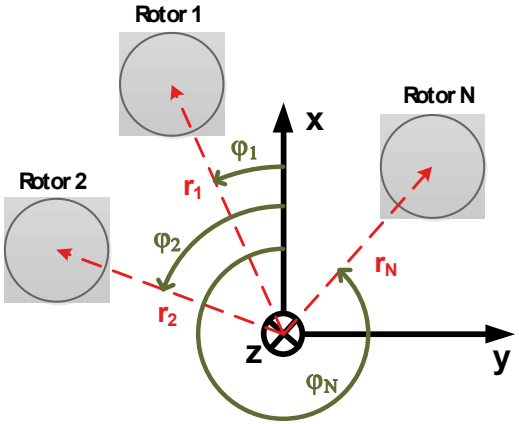
$$\left\| \begin{bmatrix} \bar{\omega}_1^2 \\ \vdots \\ \bar{\omega}_N^2 \end{bmatrix} - \begin{bmatrix} \omega_1^2 \\ \vdots \\ \omega_N^2 \end{bmatrix} \right\|^2$$

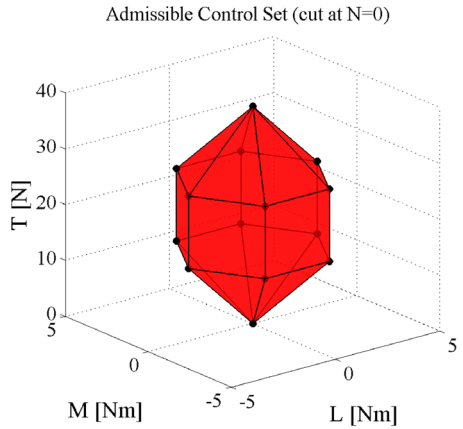
Actual thr./torq. must be admissible

$$A \begin{bmatrix} t_1 & \cdots & t_N \\ l_1 & \cdots & l_N \\ m_1 & \cdots & m_N \\ n_1 & \cdots & n_N \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \vdots \\ \omega_N^2 \end{bmatrix} \leq b$$

We use the incremental value as decision variables: $\omega_i^2 = \bar{\omega}_i^2 + \nu_i$

Control Allocation Problem





Control allocation as constrained optimization

$$\min_{\nu_1,\ldots,\nu_N} \left\| \begin{bmatrix} \bar{f}_{\text{total}} \\ \bar{\tau}_{\text{roll}} \\ \bar{\tau}_{\text{pitch}} \\ \bar{\tau}_{\text{yaw}} \end{bmatrix} - \begin{bmatrix} t_1 & \cdots & t_N \\ l_1 & \cdots & l_N \\ m_1 & \cdots & m_N \\ n_1 & \cdots & n_N \end{bmatrix} \begin{bmatrix} \bar{\omega}_1^2 + \nu_1 \\ \vdots \\ \bar{\omega}_N^2 + \nu_N \end{bmatrix} \right\|_W^2 + \rho \left\| \begin{bmatrix} \nu_1 \\ \vdots \\ \nu_N \end{bmatrix} \right\|^2$$

$$\text{s.t.} \quad \begin{bmatrix} \omega_{\min}^2 - \bar{\omega}_1^2 \\ \vdots \\ \omega_{\min}^2 - \bar{\omega}_N^2 \end{bmatrix} \leq \begin{bmatrix} \nu_1 \\ \vdots \\ \nu_N \end{bmatrix} \leq \begin{bmatrix} \omega_{\max}^2 - \bar{\omega}_1^2 \\ \vdots \\ \omega_{\max}^2 - \bar{\omega}_N^2 \end{bmatrix}$$

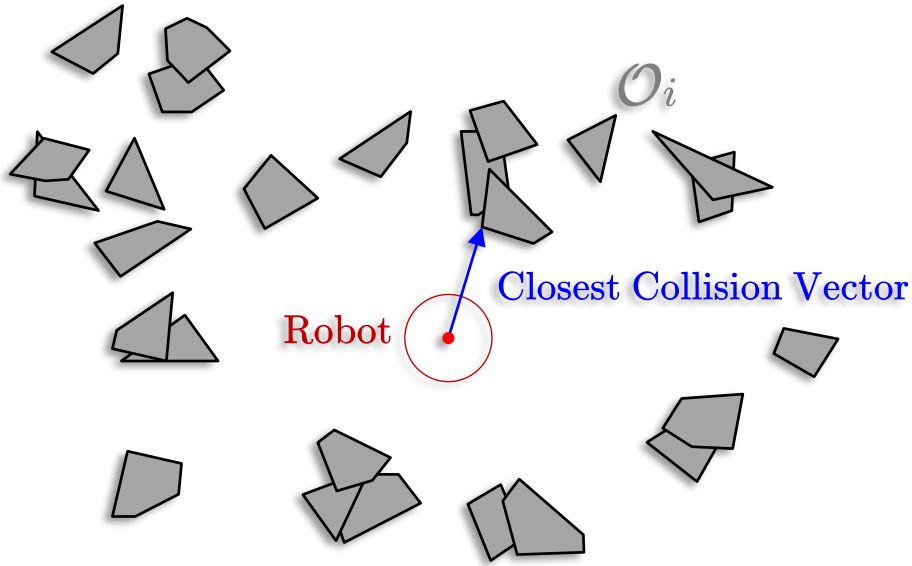
$$A \begin{bmatrix} t_1 & \cdots & t_N \\ l_1 & \cdots & l_N \\ m_1 & \cdots & m_N \\ n_1 & \cdots & n_N \end{bmatrix} \begin{bmatrix} \bar{\omega}_1^2 + \nu_1 \\ \vdots \\ \bar{\omega}_N^2 + \nu_N \end{bmatrix} \leq b$$

This is a strictly convex low-dim QP as long as N<10, whose exact solution can be obtain in linear time.

Application 2:
Collision Distance Computation

Collision Distance Computation

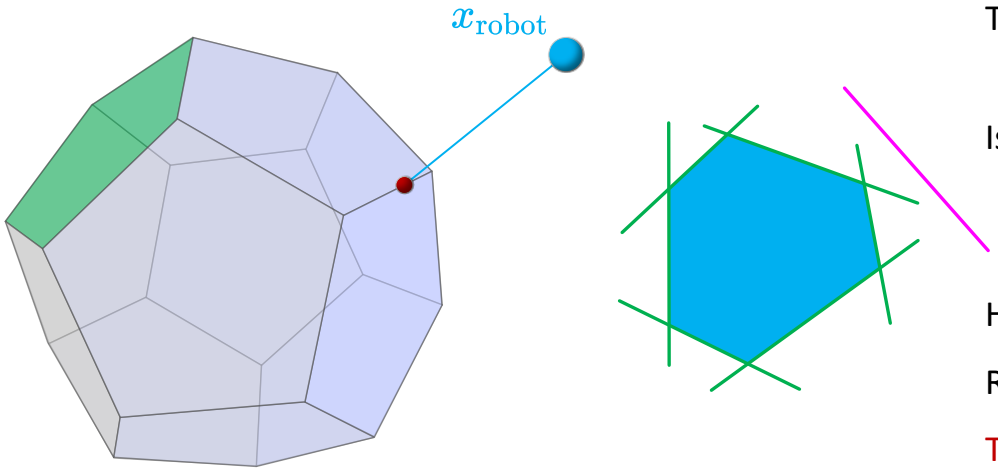
Efficiently computing collision distance is a key function in robot navigation.



Compute collision vectors for necessary obstacles then choise the shortest one!

Collision Distance Computation

Collision vector from a robot to a polytope obstacle: ***H-rep*** cases



What is a polytope?

The ***H*-representation**, i.e., finite halfspace intersection

$$\mathcal{P} = \left\{ x \in \mathbb{R}^d \mid A_{\mathcal{P}} x \leq b_{\mathcal{P}} \right\}$$

Is the halfspace always tight? Not necessary!

A tight halfspace: $x_1 + x_2 + x_3 \leq 1$

A redundant halfspace: $x_1 + x_2 + x_3 \leq 2$

How to compute collision vector?

Remove redundance? Compute Convex Hull? GJK?

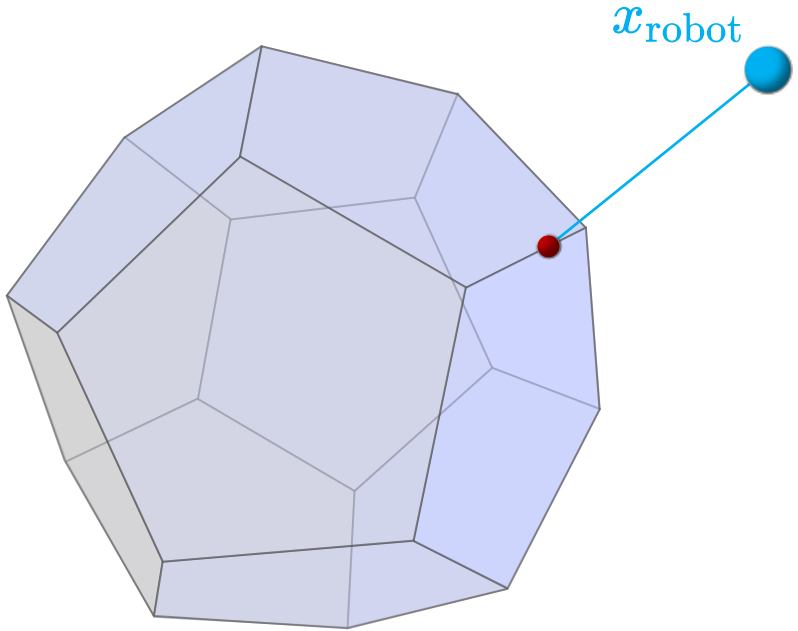
Trivially use the low-dim QP!

$$\min_{x \in \mathbb{R}^d} \|x - x_{\text{robot}}\|^2$$
$$\text{s.t. } A_{\mathcal{P}} x \leq b_{\mathcal{P}}$$

The lowest complexity, exact solution, applicability to redundant cases, efficiency in 2/3/4 dimension.

Collision Distance Computation

Collision vector from a robot to a polytope obstacle: ***V-rep*** cases

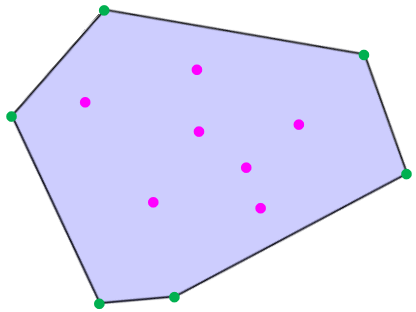


What if?

The ***V*-representation**, i.e., implicit convex hull of vertices

$$\mathcal{P} = \text{conv}\{v_1, \ldots, v_m\}$$

Redundant vertices are still possible!



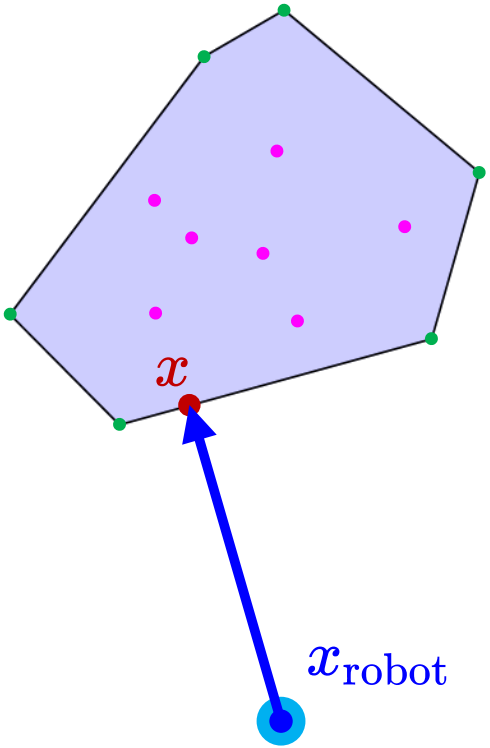
How to compute collision vector?

Need to explicitly compute their convex hull?

Still QP! Two different formulations!

Collision Distance Computation

Collision vector from a robot to a polytope obstacle: ***V-rep*** cases



One frequently used scheme is to employ barycentric coordinate:

$$x \in \mathcal{P} := \text{conv}\{v_1, \ldots, v_m\}$$

$$\text{if and only if there are } \lambda_i \text{ and } \Sigma_{i=1}^m \lambda_i = 1$$

$$\text{such that } x = \Sigma_{i=1}^m \lambda_i v_i$$

Directly optimizing the coordinates yields

$$\min_{\lambda \in \mathbb{R}^m} \|(v_1, \ldots, v_m)\lambda - x_{\text{robot}}\|^2$$

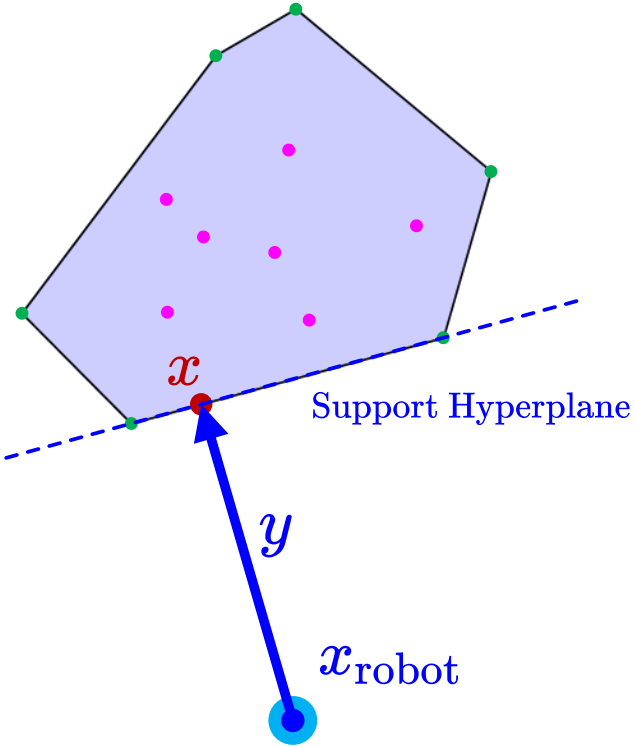
$$\text{s.t. } \lambda \geq 0,\ 1^{\mathrm{T}}\lambda = 1$$

This is a QP but with m-dimensional variables and (m+1) constraints.
When there are a lot of redundant vertices, this QP has high dim.
PHR-ALM can definitely be applied to solve it iteratively.
It is still possible but complicated to solve it exactly via solution sparsity.

Is there any better scheme?

Collision Distance Computation

Collision vector from a robot to a polytope obstacle: ***V-rep*** cases



Any collision vector is the normal of its support hyperplane.

Due to the convexity, the following two problems are equivalent

Minimize length of the V-polytope's collision vector

Maximize length of the hyperplane's normal vector

Obviously, the separating halfspace is $\{x \in \mathbb{R}^d \mid y^{\mathrm{T}}(x - x_{\text{robot}}) \leq y^{\mathrm{T}}y\}$

$$\begin{aligned} &\max_{y \in \mathbb{R}^d} y^{\mathrm{T}}y, \\ &\text{s.t. } (v_i - x_{\text{robot}})^{\mathrm{T}}y \geq y^{\mathrm{T}}y,\ \forall i \in \{1, \ldots, m\} \end{aligned}$$

Notice that if we use $z = y/(y^{\mathrm{T}}y)$ or equivalently $y = z/(z^{\mathrm{T}}z)$

$$\begin{aligned} &\min_{z \in \mathbb{R}^d} z^{\mathrm{T}}z, \\ &\text{s.t. } (v_i - x_{\text{robot}})^{\mathrm{T}}z \geq 1,\ \forall i \in \{1, \ldots, m\} \end{aligned}$$

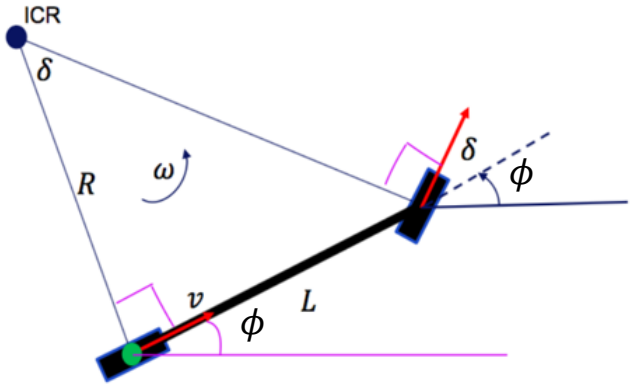
Infeasibility implies collision occurs, else $x = y + x_{\text{robot}} = z/(z^{\mathrm{T}}z) + x_{\text{robot}}$

This is a low-dim QP again !

Application 3:
Nonlinear Model Predictive Control

Nonlinear Model Predictive Control

Example: Control longitudinal acceleration and steering angle of the vehicle simultaneously for autonomous driving of tracking a reference trajectory



$$\begin{cases} \dot{x} = v\cos(\phi) \\ \dot{y} = v\sin(\phi) \\ \dot{\phi} = v\tan(\delta)/L \\ \dot{v} = a \end{cases}$$

$$\dot{s} = f(s, u)$$

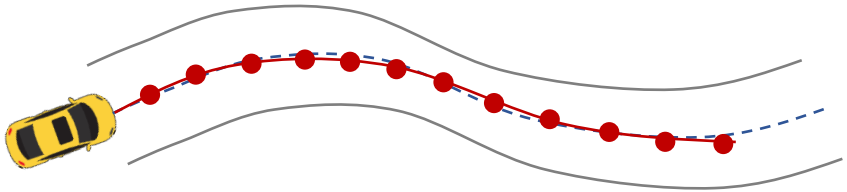
Input: $\begin{bmatrix} a \\ \delta \end{bmatrix}$ u

State: $\begin{bmatrix} x \\ y \\ \phi \\ v \end{bmatrix}$ s

$[x, y]$	cartesian position of rear wheel	v	velocity at rear wheel
ϕ	vehicle orientation	a	longitudinal acceleration
L	vehicle length	δ	steering input

Nonlinear Model Predictive Control

Example: Control longitudinal acceleration and steering angle of the vehicle simultaneously for autonomous driving of tracking a reference trajectory



Time-discretized state and control input:

$$s_{k+1} = F(s_k, u_k) := f(s_k, u_k)\tau + s_k$$

where

$$s_k = \begin{bmatrix} x_k \\ y_k \\ \phi_k \\ v_k \end{bmatrix}, \; u_k = \begin{bmatrix} a_k \\ \delta_k \end{bmatrix}, \; \tau \text{ is a constant duration}$$

Our objective function is to minimize tracking error and energy:

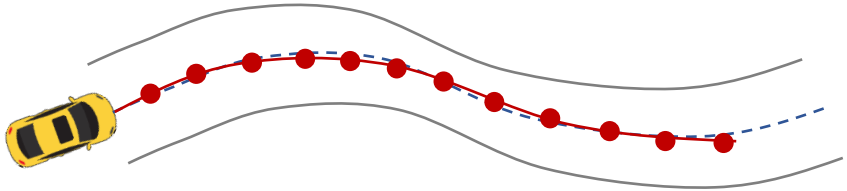
$$J(s_1, \ldots, s_N, u_0, \ldots, u_N) := \sum_{k=1}^{N} \left[(x_k - x_k^{\text{ref}})^2 + (y_k - y_k^{\text{ref}})^2 + w_v (a_k - a_{k-1})^2 + w_\delta (\delta_k - \delta_{k-1})^2 \right]$$

Physical limits requires

$$\begin{aligned} &\forall\, k \in \{0, \ldots, N\} \\ &a_{\min} \le a_k \le a_{\max} \\ &\delta_{\min} \le \delta_k \le \delta_{\max} \\ &v_{\min} \le v_k \le v_{\max} \end{aligned} \qquad G(s_k, u_k) \le 0$$

Nonlinear Model Predictive Control

Trajectory tracking can thus be formulated as a constrained optimization



$$\begin{aligned} \min_{s_1,\ldots,s_N,u_0,\ldots,u_N} \quad & J(s_1,\ldots,s_N,u_0,\ldots,u_N) \\ \text{s.t.} \quad & F(s_k,u_k)=s_{k+1}, \ \forall\, i\in\{0,\ldots,N\} \\ & G(s_k,u_k)\le 0, \quad \forall\, i\in\{0,\ldots,N\} \end{aligned}$$

If we view all discretized states as functions of controls, the equalities can be avoided.

$$u_{0:N}=(u_0,\ldots,u_N)$$

$$F(s_k,u_k)=s_{k+1},\ \forall\, i\in\{0,\ldots,N\} \quad \Longleftrightarrow \quad s_k(u_{0:N}),\ \forall\, i\in\{1,\ldots,N\}$$

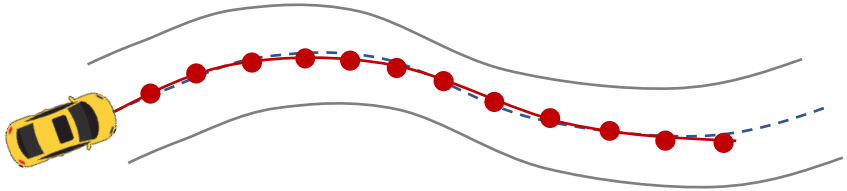
Consequently, we have

$$\begin{aligned} \min_{u_{0:N}} \ & J(s_1(u_{0:N}),\ldots,s_N(u_{0:N}),u_{0:N}) \\ \text{s.t. } & G(s_k(u_{0:N}),u_k)\le 0,\ \forall\, i\in\{0,\ldots,N\} \end{aligned}$$

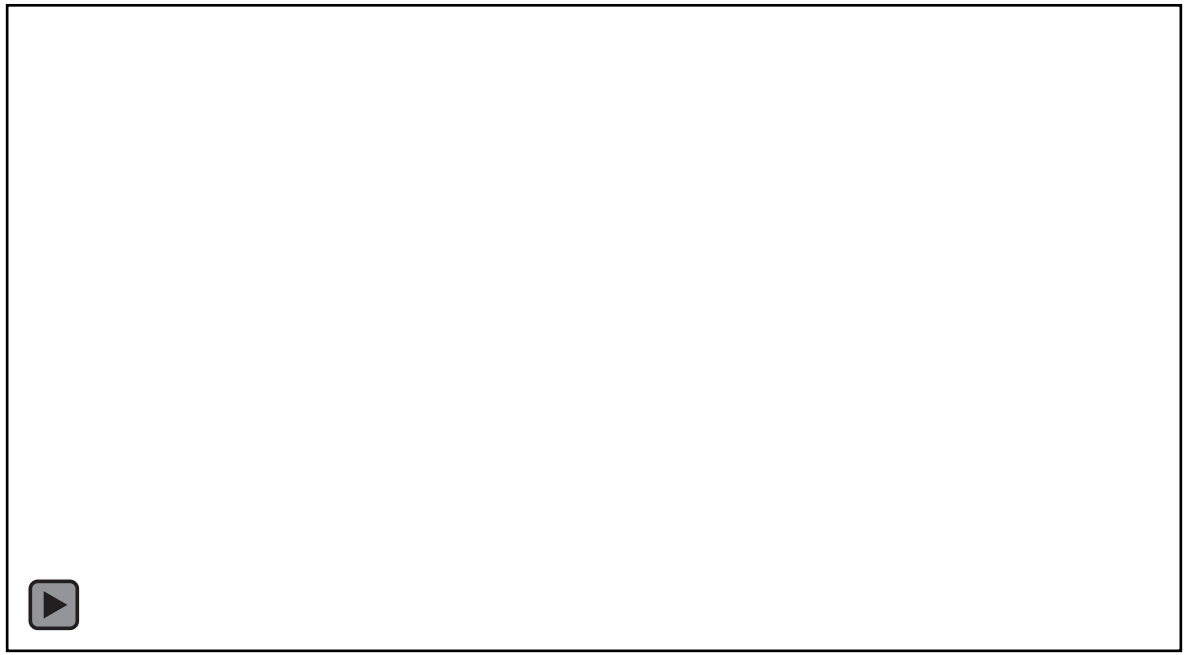
Both formulations can be efficiently solved by PHR-ALM.

Nonlinear Model Predictive Control

Trajectory tracking can thus be formulated as a constrained optimization



$$\min_{u_{0:N}} J(s_1(u_{0:N}), \ldots, s_N(u_{0:N}), u_{0:N})$$
$$\text{s.t. } G(s_k(u_{0:N}), u_k) \leq 0, \ \forall\, i \in \{0, \ldots, N\}$$



Thanks for Listening!

Ack.: The slide is prepared by Zhepei Wang and Fei Gao, improved by Lingfeng Wang, Song Zhao, and Shuo Yang.