Model Predictive Control

Chapter 11: Implementation

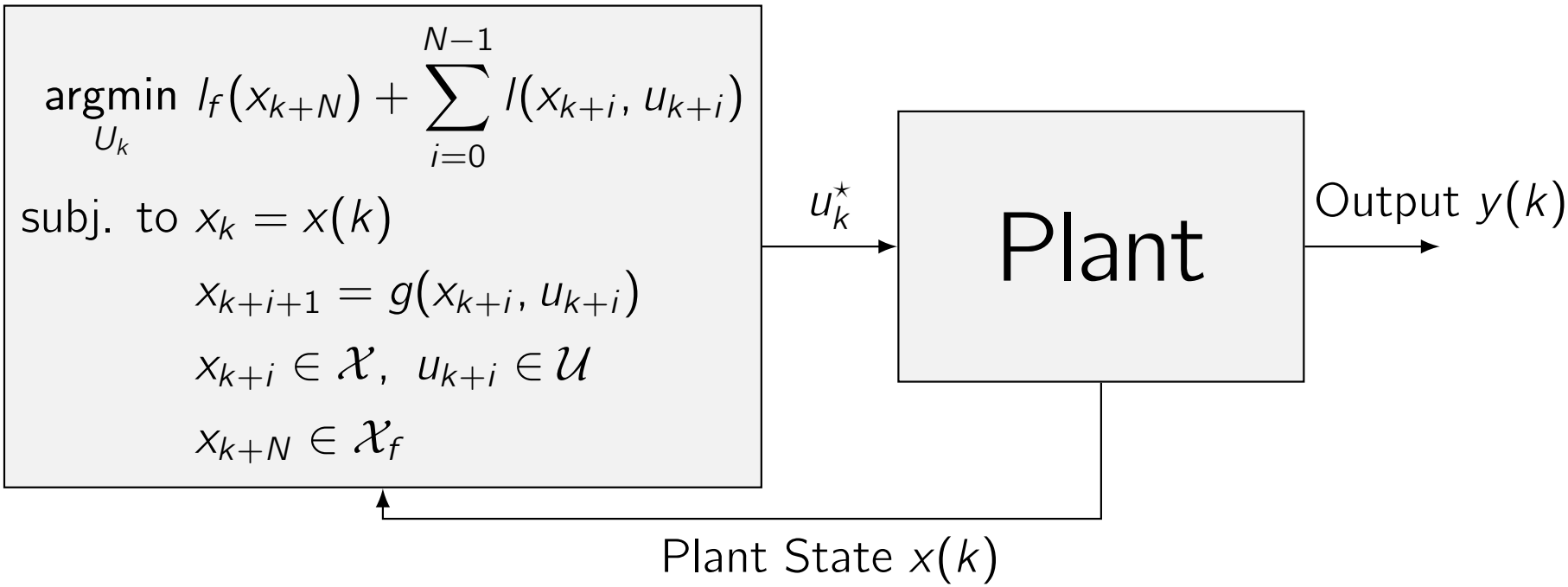
Prof. Melanie Zeilinger

ETH Zurich

Spring 2022

Coauthors: Dr. Alexander Domahidi, Embotech
Prof. Colin Jones, EPFL

Optimization in MPC



At each sample time:
Find the optimal input sequence for the entire planning window N:
$U_k^\star = \{u_k^\star, u_{k+1}^\star, \ldots, u_{k+N-1}^\star\}$

Need efficient optimization solvers. Two options:

- Iterative optimization methods
- **Explicit solution**

Introduction

OFFLINE

$$U^\star(x(k)) = \underset{U}{\text{argmin}}\ x_N^T P x_N + \sum_{i=0}^{N-1} x_i^\top Q x_i + u_i^\top R u_i$$

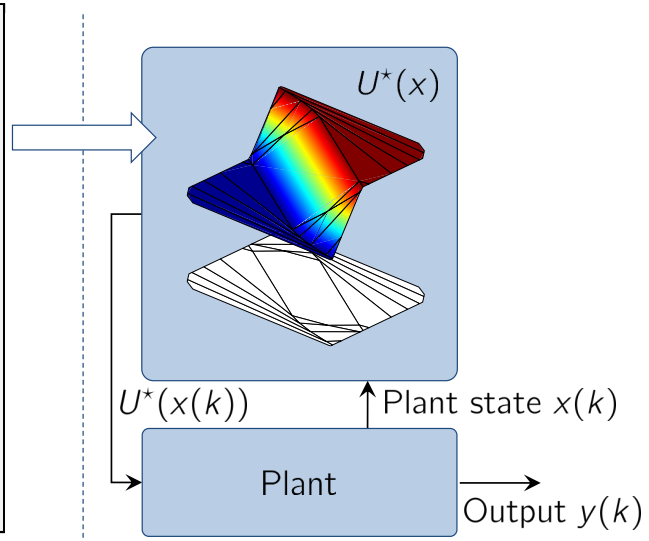
$$\text{subj. to } x_0 = x(k)$$

$$x_{i+1} = Ax_i + Bu_i,\ i = 0, \ldots, N-1$$

$$x_i \in \mathcal{X},\ u_i \in \mathcal{U},\ i = 0, \ldots, N-1$$

$$x_N \in \mathcal{X}_f$$

ONLINE



- Optimization problem is parameterized by state
- Pre-compute control law as function of state x
- Control law is piecewise affine for linear system/constraints

Result: Online computation dramatically reduced and **real-time**
Tool: **Parametric programming**

Recall: Quadratic Cost State Feedback Solution

$$J^\star(x(k)) = \min_U \quad [U^\top \; x(k)^\top] \begin{bmatrix} H & F^\top \\ F & Y \end{bmatrix} [U^\top \; x(k)^\top]^\top$$
$$\text{subj. to} \qquad GU \leq w + Ex(k)$$

The CFTOC problem is a **multiparametric quadratic program (mp-QP)** with the following solution properties:

- The first component of the optimal solution has the form

$$u_0^\star = \kappa(x(k)), \quad \forall x(k) \in \mathcal{X}_0,$$

  $\kappa : \mathbb{R}^n \to \mathbb{R}^m$, is continuous and PieceWise Affine on Polyhedra

$$\kappa(x) = F^j x + g^j \quad \text{if} \quad x \in CR^j, \quad j = 1, \ldots, N^r$$

- The polyhedral sets $CR^j = \{x \in \mathbb{R}^n | H^j x \leq K^j\}$, $j = 1, \ldots, N^r$ are a partition of the feasible polyhedron $\mathcal{X}_0$.
- The value function $J^\star(x(k))$ is convex and piecewise quadratic on polyhedra.

Active Set and Critical Region

Let $I := \{1, \ldots, m\}$ be the set of constraint indices.

Definition: Active Set

We define the active set at x, $A(x)$, and its complement, $NA(x)$, as

$$A(x) := \{j \in I : G_j z^*(x) - S_j x = w_j\}$$
$$NA(x) := \{j \in I : \ G_j z^*(x) - S_j x < w_j\}.$$

G_j, S_j and w_j are the j-th row of G, S and w, respectively.

Definition: Critical Region

CR_A is the set of parameters x for which the same set $A \subseteq I$ of constraints is active at the optimum. For a given $\bar{x} \in \mathcal{K}^*$ let $(A, NA) := (A(\bar{x}), NA(\bar{x}))$. Then,

$$CR_A := \{x \in \mathcal{K}^* : A(x) = A\}.$$

Example (1/2)

Consider the following problem:

$$\begin{aligned} J^*(x) = \min_z \quad & J(z,x) = \tfrac{1}{2}z^2 + 2xz + 2x^2 \\ \text{subj. to} \quad & z \leq 1 + x, \end{aligned}$$

$x \in \mathbb{R}$ is a parameter.
The goals:

1. find $z^*(x) = \text{argmin}_z \; J(z,x)$,
2. find all x for which the problem has a solution
3. compute the **value function** $J^*(x)$

Define Lagrangian: $L(z,x,\lambda) = f(z,x) + \lambda(z - x - 1)$
KKT conditions:

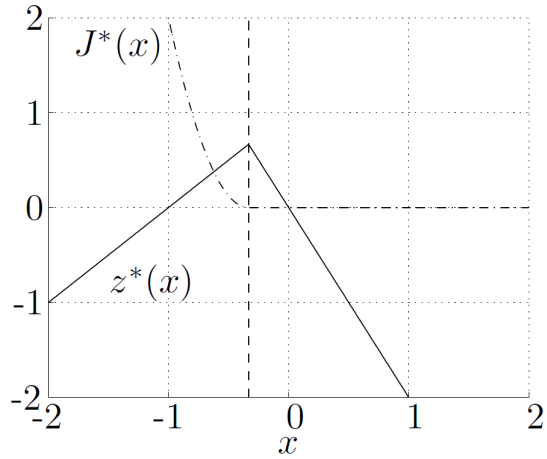
$$\begin{aligned} z + 2x + \lambda &= 0, \\ z - x - 1 &\leq 0, \\ \lambda(z - x - 1) &= 0, \\ \lambda &\geq 0. \end{aligned}$$

Example (2/2)

Consider the two strictly complementary cases:

1. $$\begin{matrix} z - x - 1 = 0 \\ \lambda \geq 0 \end{matrix} \Rightarrow \begin{cases} z^*(x) = x + 1, \\ x \leq -\frac{1}{3} \\ J^*(x) = \frac{9}{2}x^2 + 3x + \frac{1}{2} \end{cases}$$

2. $$\begin{matrix} z - x - 1 < 0 \\ \lambda = 0 \end{matrix} \Rightarrow \begin{cases} z^*(x) = -2x, \\ x > -\frac{1}{3} \\ J^*(x) = 0 \end{cases}$$



- $$\Rightarrow z^*(x) = \begin{cases} x + 1, & \text{if } x \leq -\frac{1}{3} \\ -2x, & \text{if } x \geq -\frac{1}{3} \end{cases},$$
  $$J^*(x) = \begin{cases} \frac{9}{2}x^2 + 3x + \frac{1}{2}, & \text{if } x \leq -\frac{1}{3} \\ 0, & \text{if } x \geq -\frac{1}{3} \end{cases}$$
- This problem has a solution for all x.

Example

Consider the double integrator

$$\begin{cases} x(t+1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \end{cases}$$

subject to constraints

$$-1 \leq u(k) \leq 1, \ k = 0, \dots, 5$$

$$\begin{bmatrix} -10 \\ -10 \end{bmatrix} \leq x(k) \leq \begin{bmatrix} 10 \\ 10 \end{bmatrix}, \ k = 0, \dots, 5$$

Compute the **state feedback** optimal controller $u^*(x(k))$ solving the CFTOC problem with $N = 6$, $Q = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$, $R = 0.1$, P the solution of the ARE, $\mathcal{X}_f = \mathbb{R}^2$.

Example

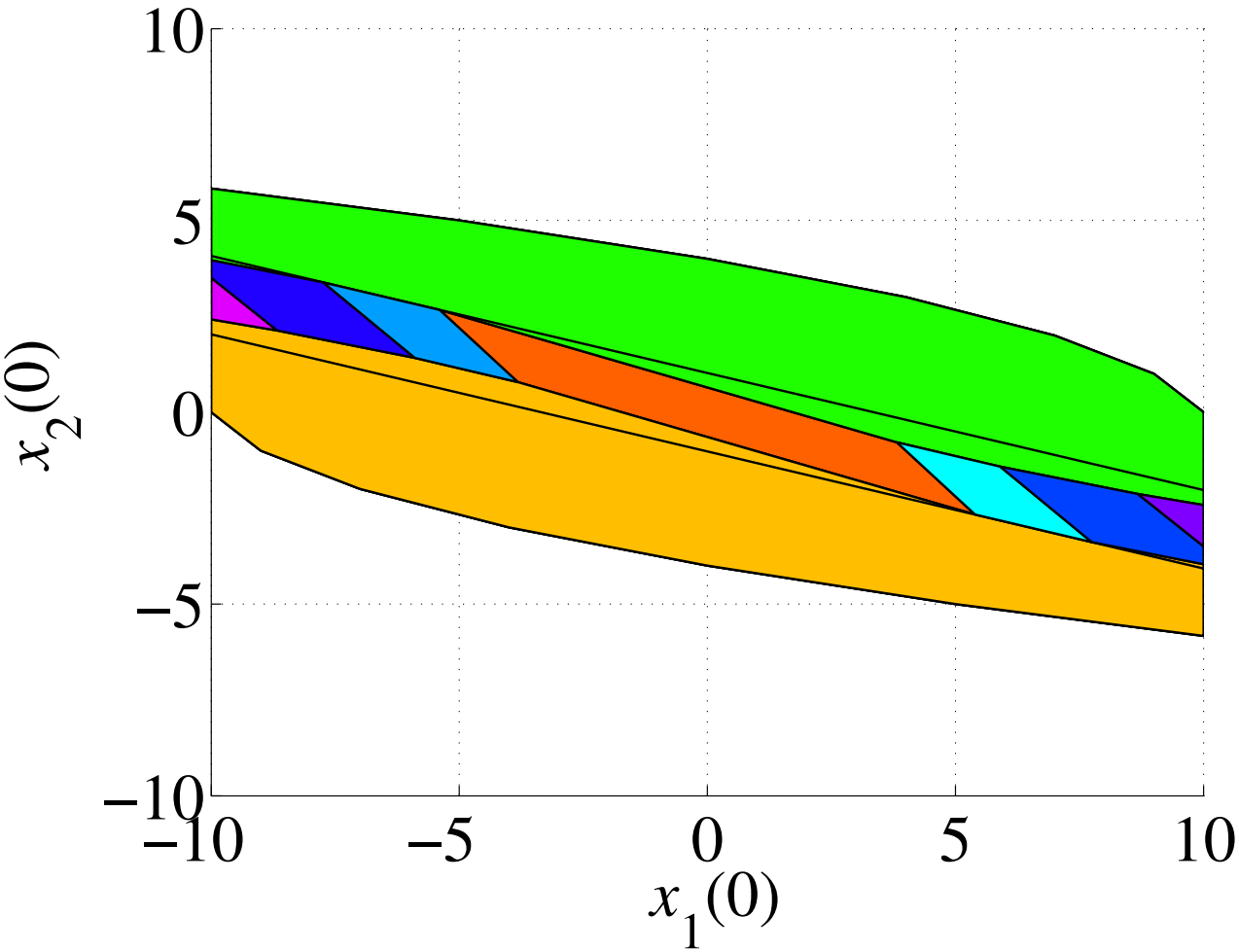


Figure: Partition of the state space for the affine control law $u^*(x)$ ($N_0^r = 13$)

Example

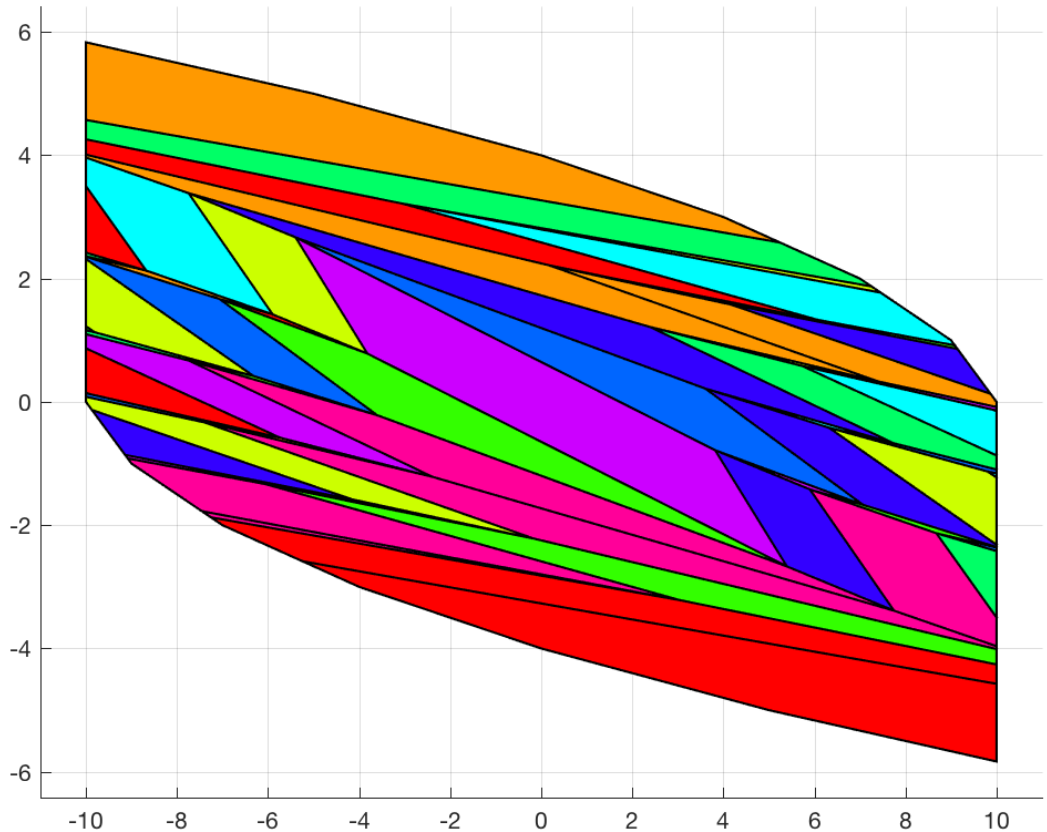


Figure: Partition of the state space for the affine control law $u^*(x)$ ($N_0^r = 61$)

Example

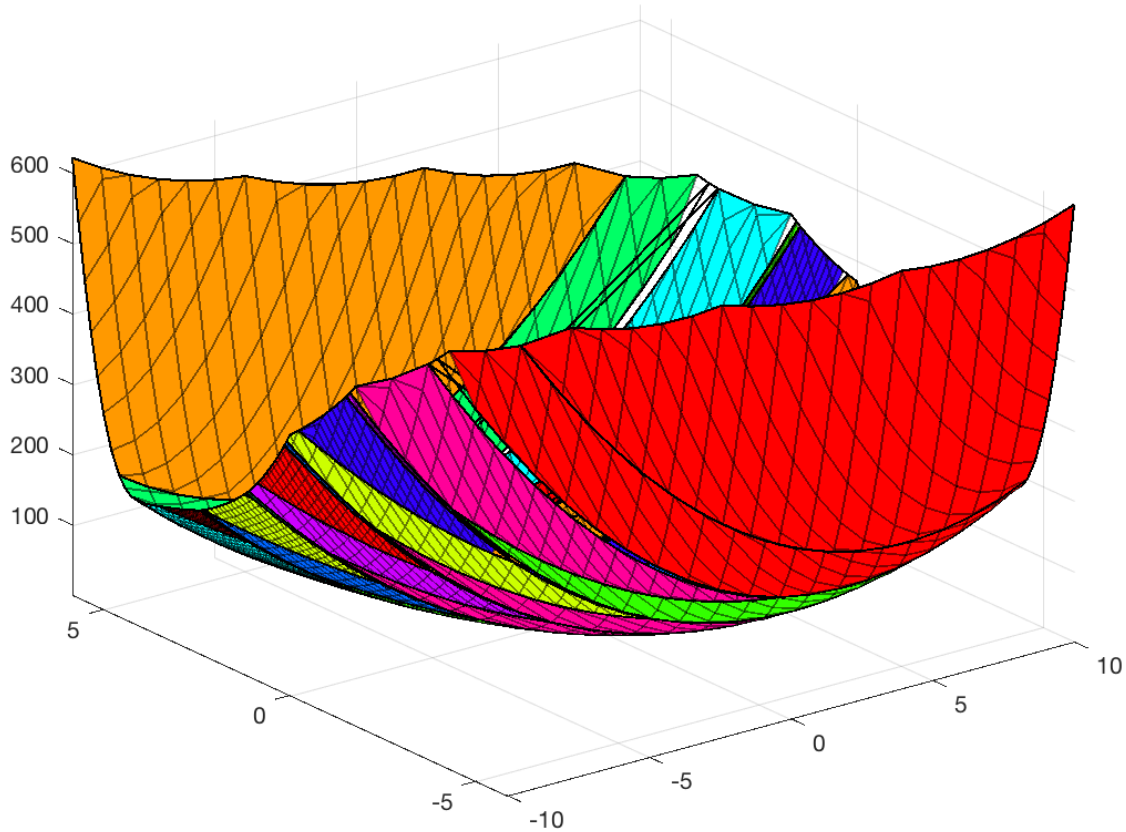


Figure: Value function for the affine control law $u^*(x)$ ($N_0^r = 61$)

Example

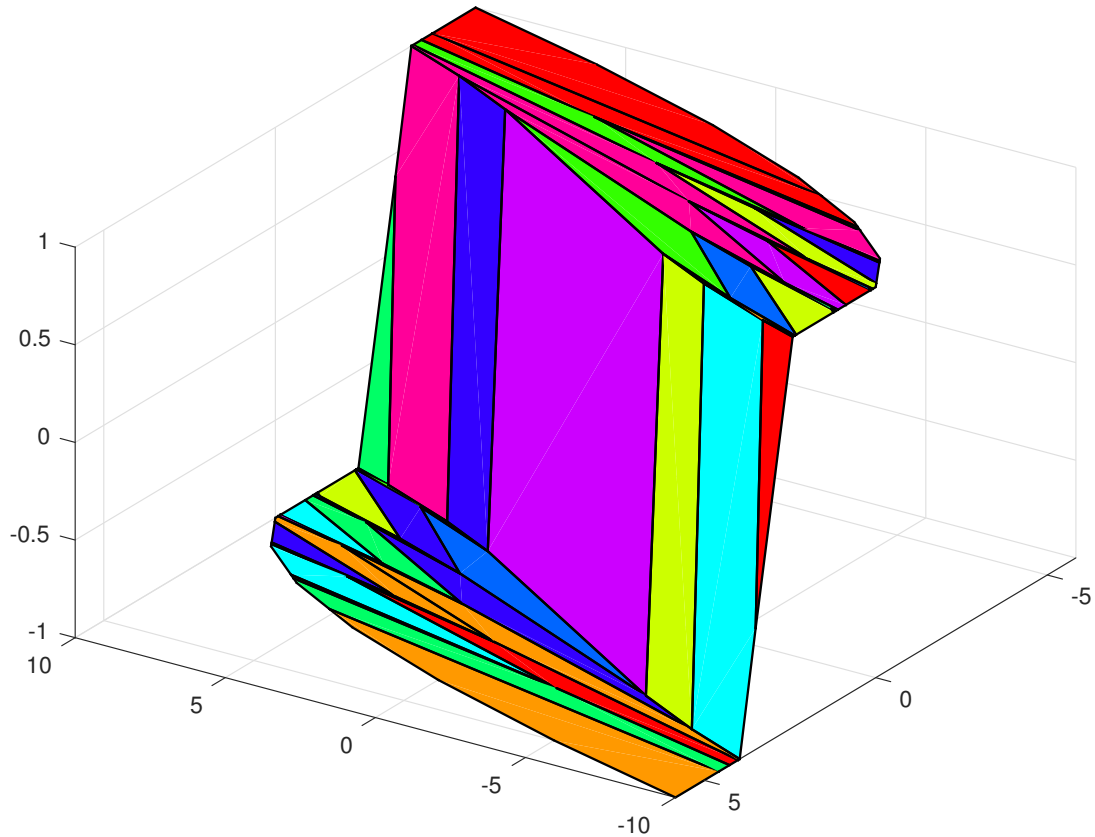


Figure: Optimal control input for the affine control law $u^*(0)$ ($N_0^r = 61$)

Recall: 1- /∞-Norm State Feedback Solution

$$\begin{array}{ll} \min\limits_{z} & c^\top z \\ \text{subj. to} & \bar{G}z \leq \bar{w} + \bar{S}x(k) \end{array}$$

The CFTOC problem is a **multiparametric linear program (mp-LP)** with the following solution properties:

- The first component of the multiparametric solution has the form

$$u_0^\star = \kappa(x(0)), \quad \forall x(0) \in \mathcal{X}_0,$$

  $\kappa : \mathbb{R}^n \to \mathbb{R}^m$, is continuous and PieceWise Affine on Polyhedra

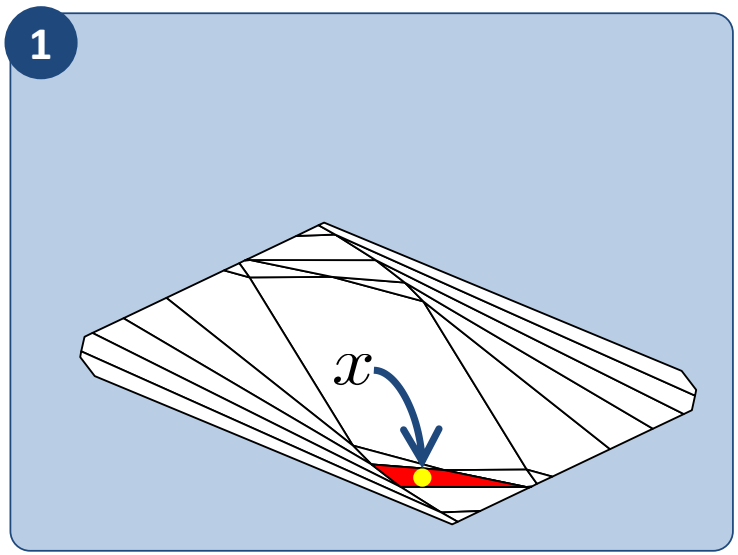
$$\kappa(x) = F^j x + g^j \quad \text{if} \quad x \in CR^j, \quad j = 1, \ldots, N^r$$

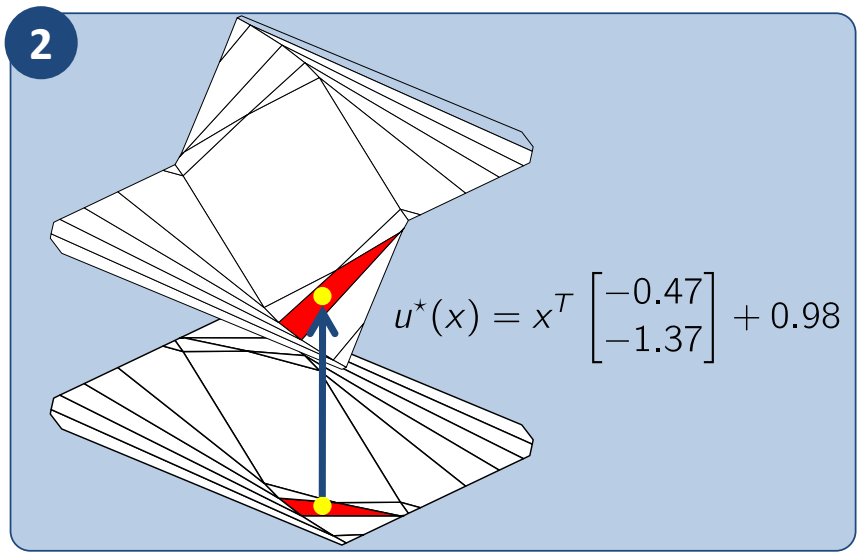
- The polyhedral sets $CR^j = \{x \in \mathbb{R}^n | H^j x \leq K^j\}$, $j = 1, \ldots, N^r$ are a partition of the feasible polyhedron $\mathcal{X}_0$.
- In case of multiple optimizers a PieceWise Affine control law exists.
- The value function $J^\star(x(0))$ is convex and piecewise linear on polyhedra.

Online evaluation: Point location

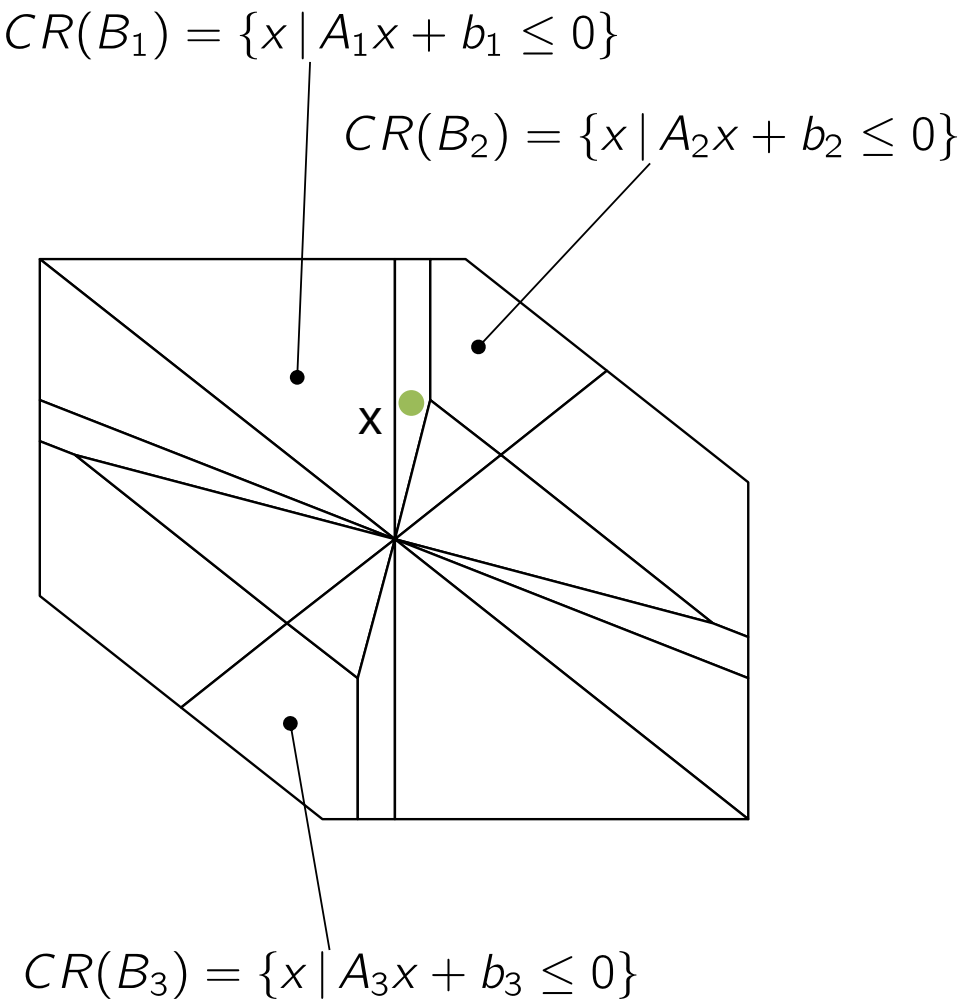
Calculation of piecewise affine function:

1. Point location
2. Evaluation of affine function





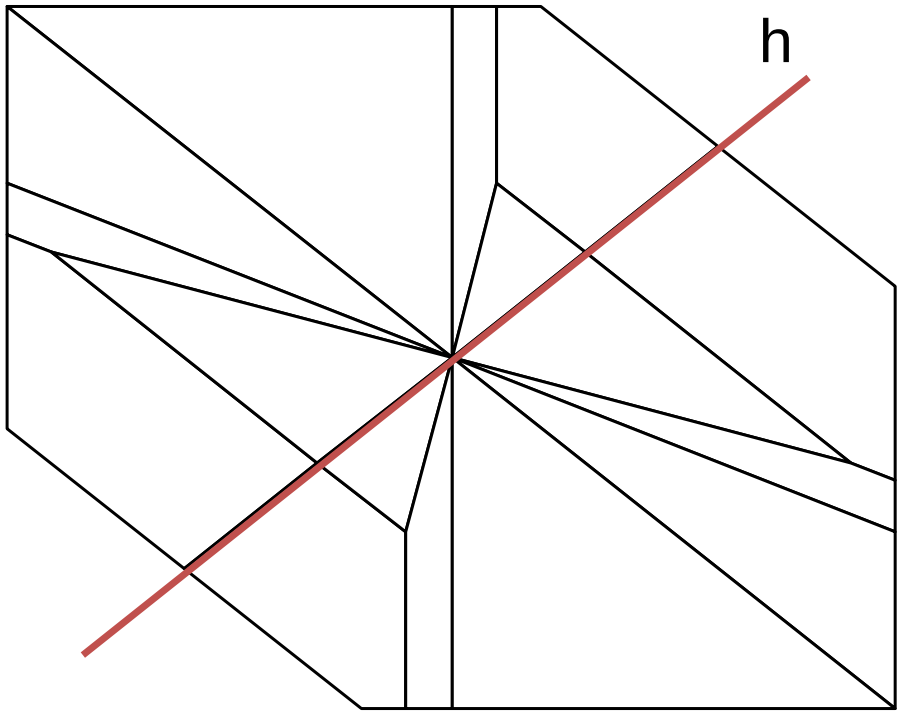
Sequential Search



Sequential search
for each j
  if $A_jx + b_j \leq 0$ then
  x is in region j

- Very simple
- Linear in number of regions

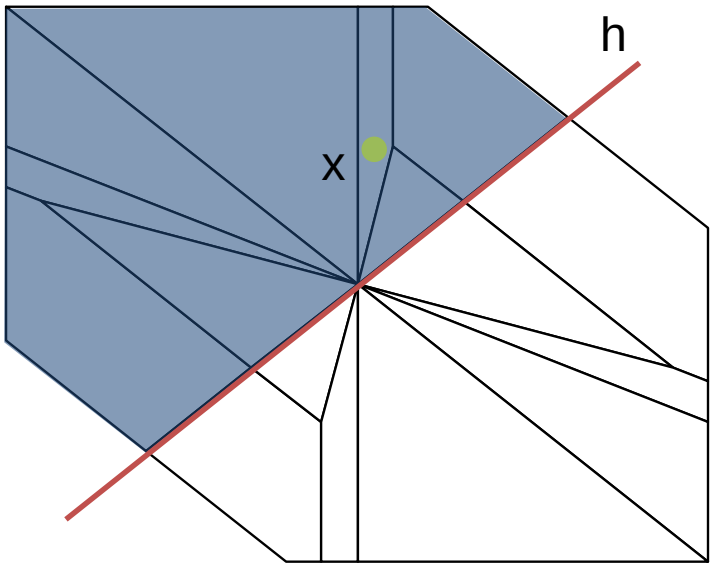
Logarithmic search (1/6)

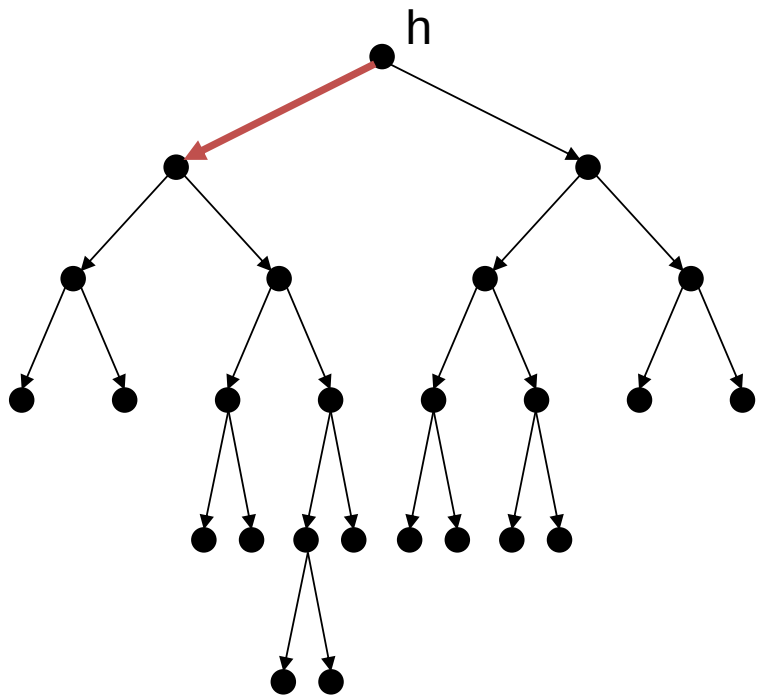


Offline construction of search tree

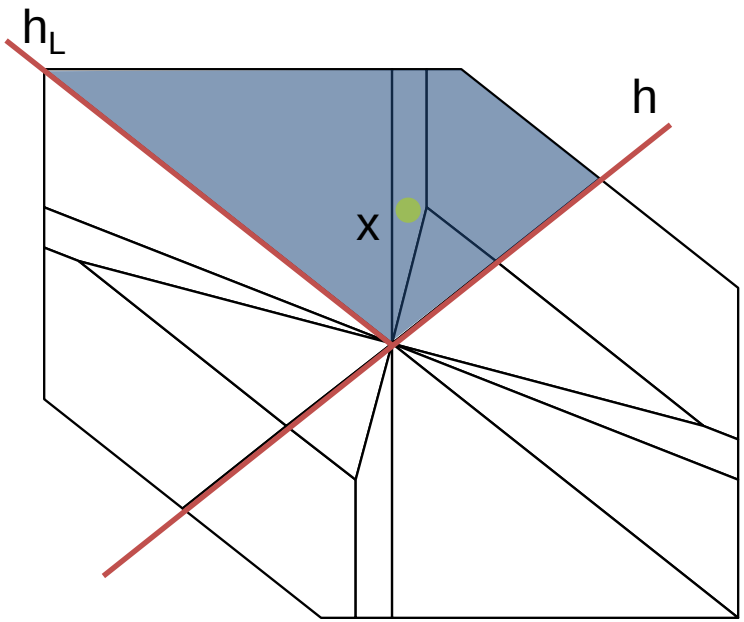
- Find hyperplane that separates regions into two equal sized sets
- Repeat for left and right sets

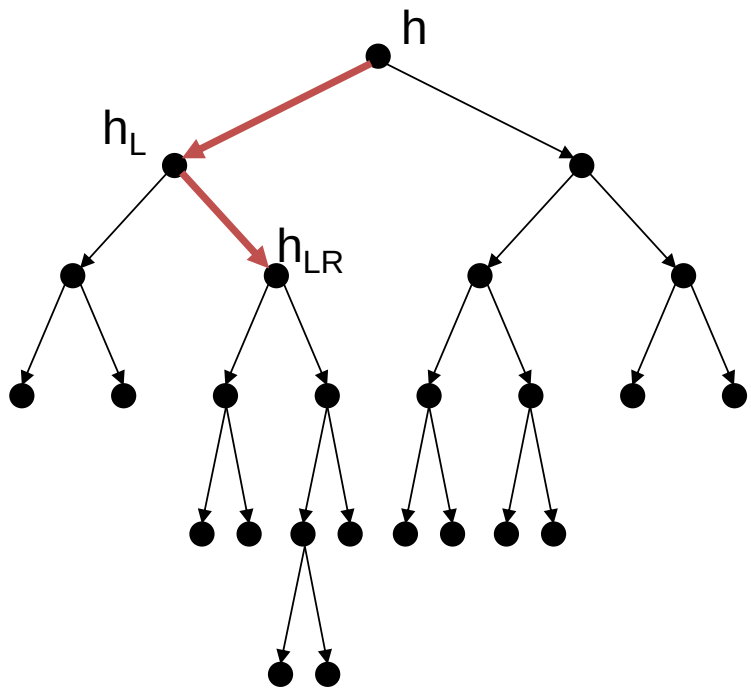
Logarithmic search (2/6)



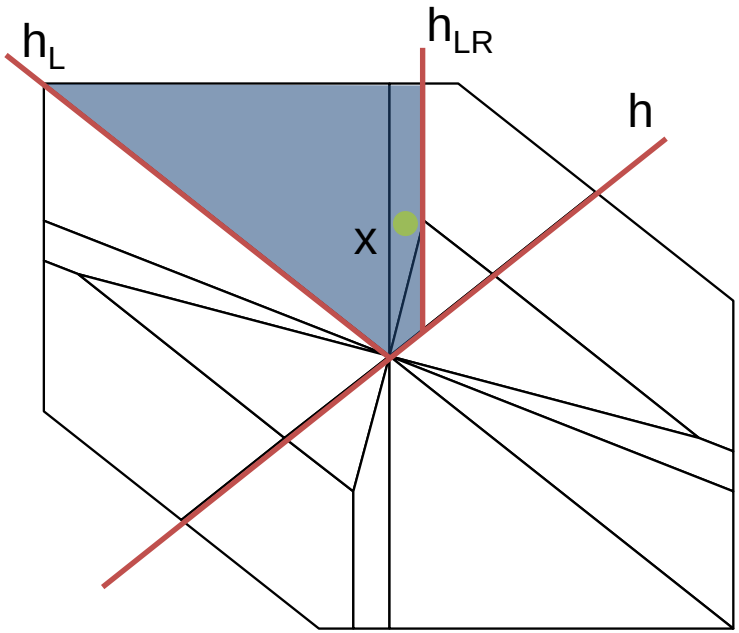


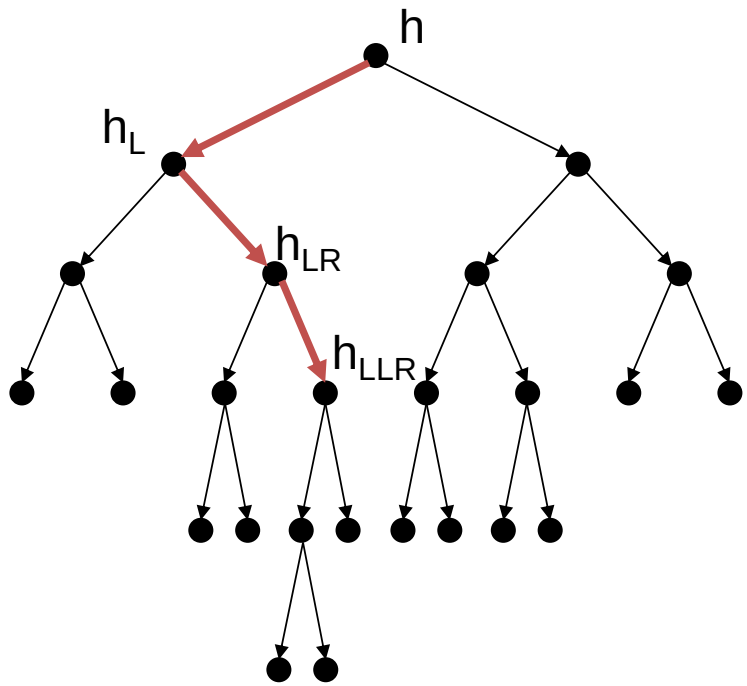
Logarithmic search (3/6)



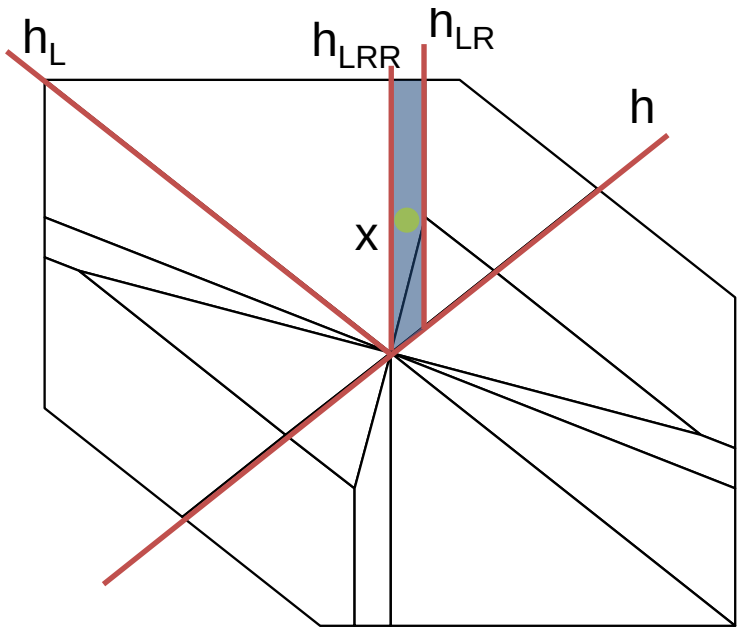


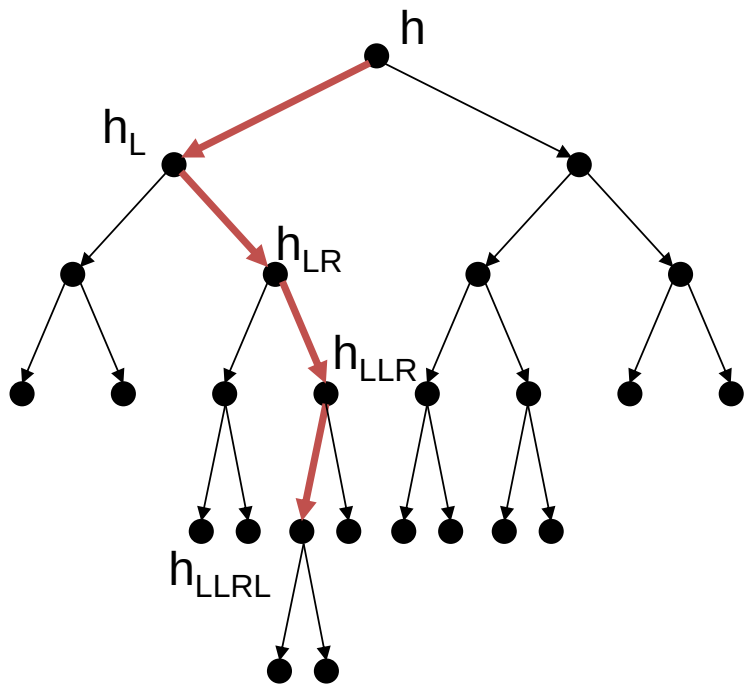
Logarithmic search (4/6)



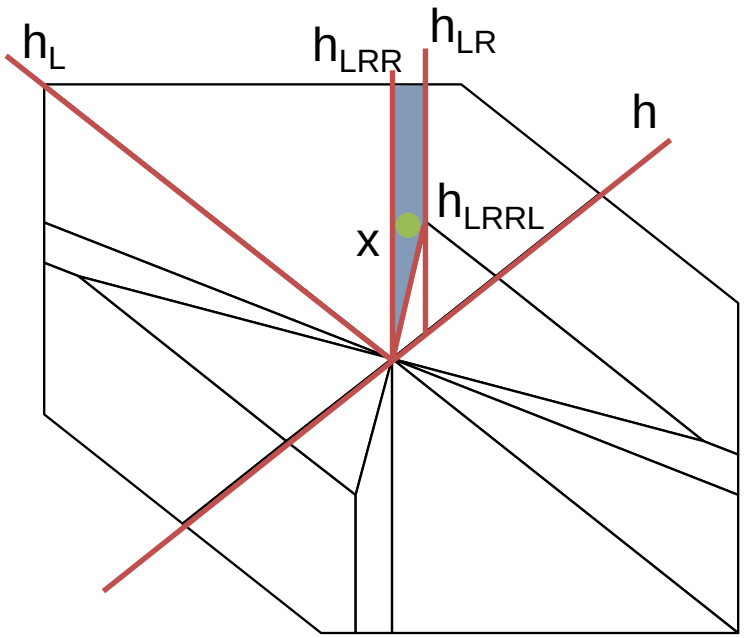


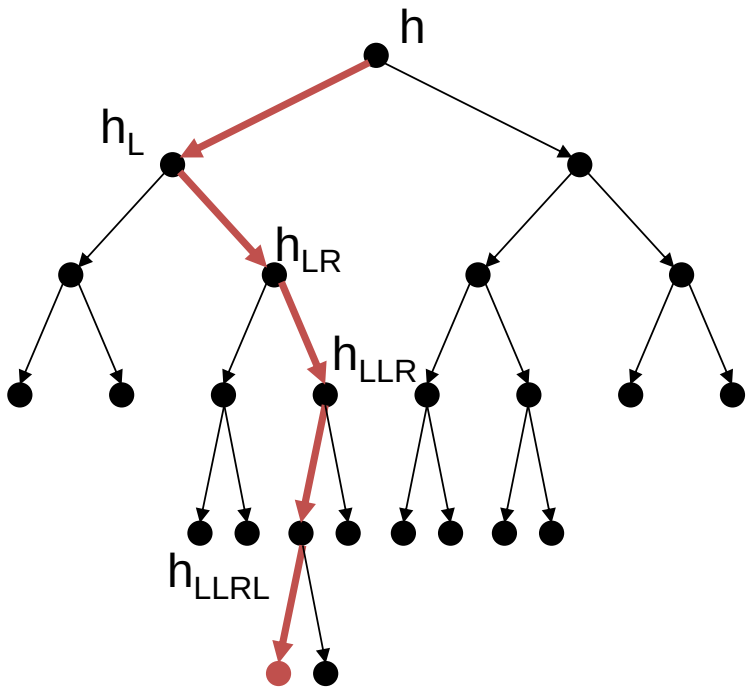
Logarithmic search (5/6)





Logarithmic search (6/6)





Explicit MPC - Summary

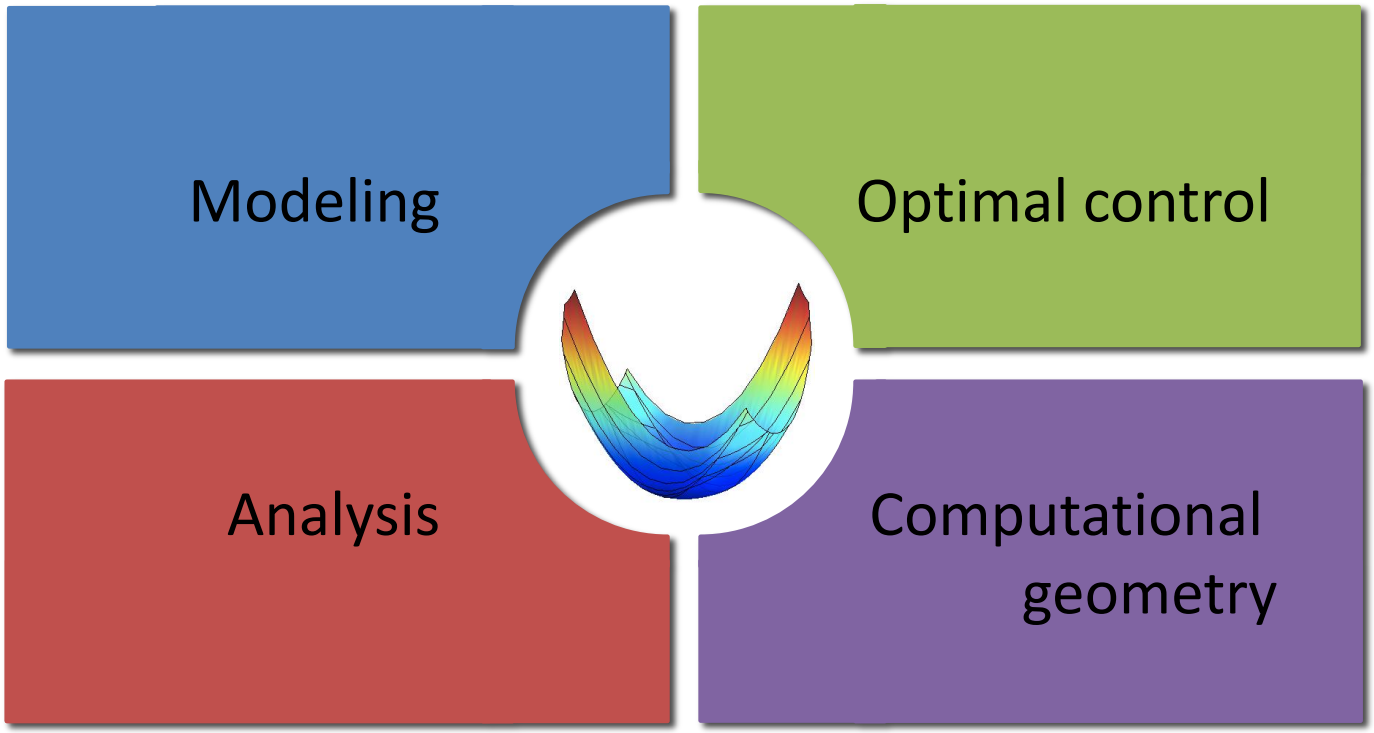
Point location:

- Sequential search
  - Very simple
  - Works for all problems
- Search tree
  - Potentially logarithmic
  - Significant offline processing (reasonable for $< 1'000$ regions)
- Many other options for special cases

Explicit MPC:

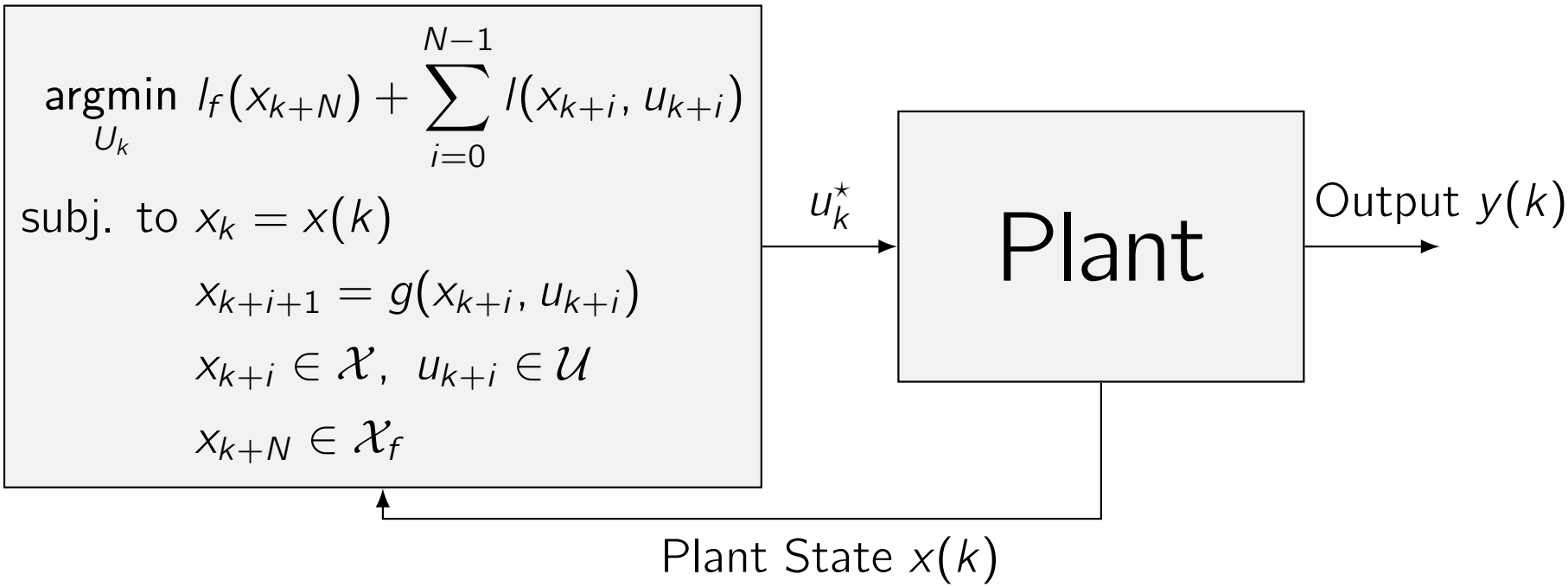
- Linear MPC + Quadratic or linear-norm cost $\Rightarrow$ Controller is PWA function
- We can pre-compute this function offline
- Online evaluation of a PWA function is very fast (ns - μs)
- We can only do this for very small systems! (3-6 states)

Multi-Parametric Toolbox



control.ee.ethz.ch/~mpt

Optimization in MPC



At each sample time:
Find the optimal input sequence for the entire planning window N:
$U_k^\star = \{u_k^\star, u_{k+1}^\star, \ldots, u_{k+N-1}^\star\}$

Need efficient optimization solvers. Two options:

- **Iterative optimization methods**
- Explicit solution

Recap: Convex Optimization Problems

The optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathbb{Q} \end{array} \tag{P}$$

is said to be convex, if $f : \mathbb{R}^n \to \mathbb{R}$ and the set $\mathbb{Q}$ are convex.

Most important examples:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax = b \\ & Gx \leq f \end{array} \tag{LP}$$

$$\begin{array}{ll} \min & \frac{1}{2} x^T H x + c^T x \\ \text{s.t.} & Ax = b \\ & Gx \leq f \end{array} \tag{QP}$$

(convex for $H \succeq 0$)

Numerical Optimization Methods

In all but the simplest cases, an analytical solution to (P),

$$x^* \in \arg \quad \min \quad f(x)$$
$$\text{s.t.} \quad x \in \mathbb{Q}$$

cannot be obtained.

$\Rightarrow$ Numerical computation of a solution that is "good enough" by

Iterative optimization methods:

Given an initial guess $x^{(0)}$, produce a sequence of iterates

$$x^{(i+1)} = \Psi(x^{(i)}, f, \mathbb{Q}), \quad i = 0, 1, \ldots, m-1$$

such that

$$|f(x^{(m)}) - f(x^*)| \leq \epsilon \quad \text{and} \quad \text{dist}(x^{(m)}, \mathbb{Q}) \leq \delta,$$

where ϵ and δ are user defined tolerances.

Outline

1. Unconstrained Minimization

2. Constrained Minimization

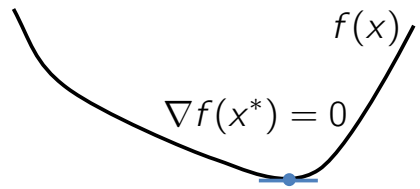
Unconstrained Minimization

$$\min_x \; f(x) \quad \text{with } f : \mathbb{R}^n \to \mathbb{R}$$

- f convex, twice continuously differentiable
- We assume optimal value $p^* = \min_x f(x)$ is finite

Optimality Conditions:

Theorem: Necessary and sufficient condition
Assume $f(\cdot)$ differentiable at x^*. If f is convex, then x^* is a global minimizer if and only if $\nabla f(x^*) = 0$.



Unconstrained minimization methods can be interpreted as iterative methods for solving optimality condition

$$\nabla f(x^*) = 0$$

(nonlinear set of equations, usually no analytical solution)

Descent Methods

$$x^{(i+1)} = x^{(i)} + h^{(i)}\Delta x^{(i)} \quad \text{with } f\big(x^{(i+1)}\big) < f\big(x^{(i)}\big)$$

- Δx is the **step** or **search direction**
- $h^{(i)}$ is the **step size** or **step length**
- $f\big(x^{(i+1)}\big) < f\big(x^{(i)}\big)$, i.e., $\Delta x^{(i)}$ is a **descent direction**
- There exists a $h^{(i)} > 0$ such that $f\big(x^{(i+1)}\big) < f\big(x^{(i)}\big)$ if $\nabla f\big(x^{(i)}\big)^T \Delta x^{(i)} < 0$

General descent method:

Input: starting point $x^{(0)} \in$ domain of f

repeat

1. Compute a *descent direction* $\Delta x^{(i)}$
2. *Line search*: Choose step size $h^{(i)} > 0$ such that $f\big(x^{(i)} + h^{(i)}\Delta x^{(i)}\big) < f\big(x^{(i)}\big)$
3. *Update* $x^{(i+1)} := x^{(i)} + h\Delta x^{(i)}$

until termination cond. (e.g. $f\big(x^{(m)}\big) - f\big(x^*\big) \leq \epsilon_1$ *or* $\|x^{(m)} - x^{(m-1)}\| \leq \epsilon_2$)

Outline

Descent Directions: Gradient Method

- **Gradient descent**
  Idea: Gradient ∇f gives direction of steepest local ascent
  $\Rightarrow$ Make steps of size h into anti-gradient direction $-\nabla f$:

$$\boxed{x^{(i+1)} = x^{(i)} - h^{(i)} \nabla f\big(x^{(i)}\big)} \tag{1}$$

Question: How to choose the step sizes $h^{(i)}$?

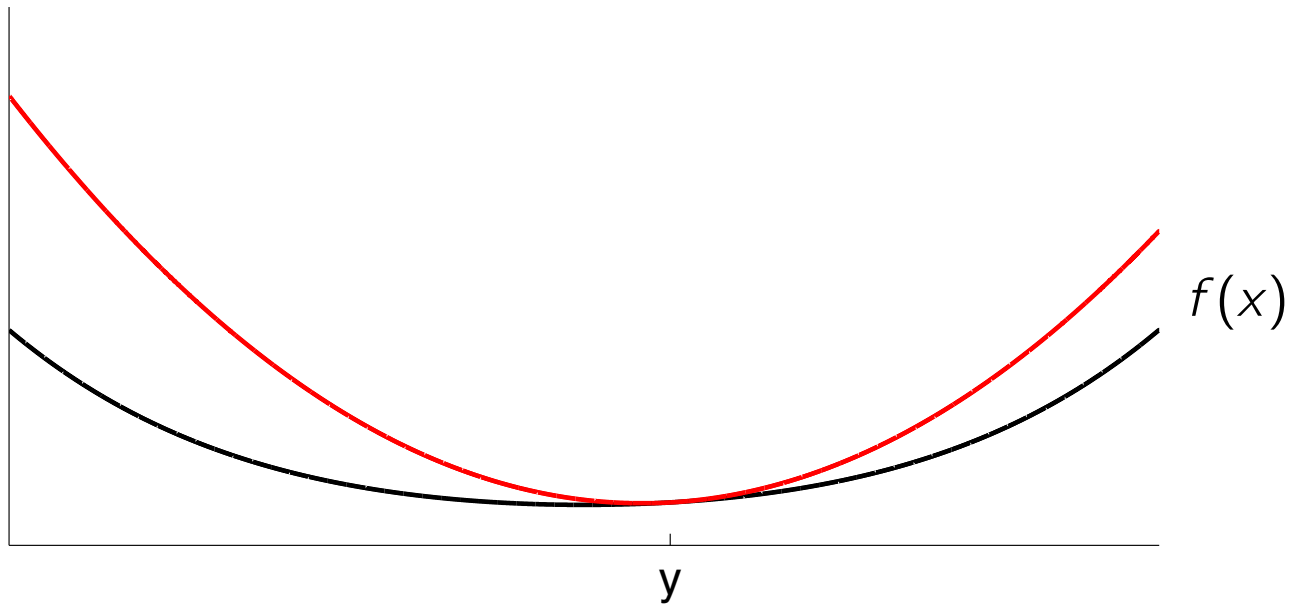
L-smoothness and Constant Step Size

Assumption: $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \; \forall x, y \in \mathbb{R}^n$

$\Leftrightarrow$ ∇f is Lipschitz-continuous with constant L

$\Leftrightarrow$ f can be upper bounded by a quadratic function:

$$f(x) \leq f(y) + \nabla f(y)^T (x - y) + \tfrac{L}{2}\|x - y\|^2 \; \forall x, y \in \mathbb{R}^n$$



$\Rightarrow$ Choose constant step size: $h^{(i)} = \frac{1}{L}$

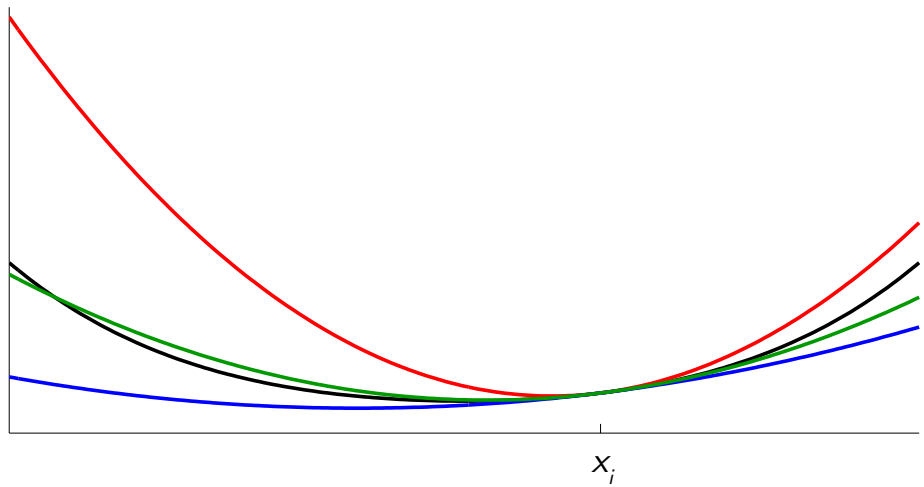
Outline

Descent Directions: Newton's Method

$$x^{(i+1)} = x^{(i)} + h^{(i)} \Delta x^{(i)} \quad \text{with } f(x^{(i+1)}) < f(x^{(i)})$$

Idea: Minimize second-order approximation of f at current iterate $x^{(i)}$:

$$x^{(i+1)} = \arg\min_x \underbrace{f(x^{(i)}) + \nabla f(x^{(i)})^T (x - x^{(i)}) + \frac{1}{2}(x - x^{(i)})^T \nabla^2 f(x^{(i)})(x - x^{(i)})}_{\triangleq \tilde{f}(x, x^{(i)})}$$



Note: $\tilde{f}$ is not necessarily an upper bound on f

Descent Directions: Newton's Method

Idea: Minimize second-order approximation of f at current iterate x_i:

$$x^{(i+1)} = \arg\min_x f(x^{(i)}) + \nabla f(x^{(i)})^T (x - x^{(i)}) + \frac{1}{2}(x - x^{(i)})^T \nabla^2 f(x_i)(x - x^{(i)})$$

$$\nabla_x \left(f(x^{(i)}) + \nabla f(x^{(i)})^T (x - x^{(i)}) + \frac{1}{2}(x - x^{(i)})^T \nabla^2 f(x^{(i)})(x - x^{(i)}) \right) \Bigg|_{x=x^{(i+1)}} = 0$$

$$\Leftrightarrow \nabla f(x^{(i)}) + \nabla^2 f(x^{(i)})(x^{(i+1)} - x^{(i)}) = 0$$

$$\Leftrightarrow x^{(i+1)} = x^{(i)} \underbrace{- \left(\nabla^2 f(x^{(i)}) \right)^{-1} \nabla f(x^{(i)})}_{\text{Newton direction } \Delta x_{nt}}$$

Since $\tilde{f}$ is not an upper bound on f, full Newton step does not necessarily yield descent (i.e. $f(x^{(i+1)}) > f(x^{(i)})$ might occur)

Idea: Use step size $h^{(i)} > 0$ such that Newton step yields descent

$$\text{Newton step: } \boxed{x^{(i+1)} = x^{(i)} - h^{(i)} \left(\nabla^2 f(x^{(i)}) \right)^{-1} \nabla f(x^{(i)})} \tag{2}$$

Line Search

Newton step: $\boxed{x^{(i+1)} = x^{(i)} + h^{(i)} \Delta x_{nt}}$

Problem: Find $h^{(i)} > 0$ s.t. $f(x^{(i)} + h^{(i)} \Delta x_{nt}) < f(x^{(i)})$

Line search (LS) methods:

- **Exact**: Compute **best** $h^{(i)}$:

$$h^{(i)*} = \arg\min_{h>0} f(x^{(i)} + h^{(i)} \Delta x_{nt})$$

  Optimization in 1 variable $\rightarrow$ solve by bisection
  Time consuming (requires many evaluations of f)

- **Inexact**: Find $h^{(i)}$ that decreases f by some amount.
  Example: **Backtracking**[1] line search.
  For $\alpha \in (0,\, 0.5)$ and $\beta \in (0,\, 1)$:
  Initialize $h^{(i)} = 1$.
  while $f(x^{(i)} + h^{(i)} \Delta x_{nt}) > f(x^{(i)}) + \alpha h^{(i)} \nabla f(x^{(i)})^T \Delta x_{nt}$ **do** $h^{(i)} \leftarrow \beta h^{(i)}$

[1] More details in e.g. *[Boyd & Vandenberghe, Convex Optimization, 2004]*

Equality constraints in Newton's method

Consider the equality constrained problem (with matrix $A \in \mathbb{R}^{m \times n}$)

$$\text{minimize } f(x)$$
$$\text{subject to } Ax = b$$

Newton: Minimize quadratic model of f around $x^{(i)}$ to obtain descent direction

$$\Delta x_{nt}(x_i) \in \arg\min_{\Delta x} \frac{1}{2}\Delta x^T \nabla^2 f(x^{(i)})\Delta x + \nabla f(x^{(i)})\Delta x \qquad (**)$$
$$\text{s.t. } A\Delta x = -Ax^{(i)} + b$$

Notice that if $Ax^{(i)} = b$, then $\Delta x_{nt} \in \text{Null}(A)$

$$\Rightarrow Ax^{(i+1)} = Ax^{(i)} + h^{(i)} A\Delta x_{nt}(x^{(i)}) = b + 0 \quad \forall h^{(i)}$$

Hence if initial iterate satisfies $Ax^{(0)} = b$, then $Ax^{(i)} = b \; \forall i$

Computation: Amounts to solving a linear system.

Optimality conditions of (**) (with multipliers $\lambda \in \mathbb{R}^m$):

$$\nabla^2 f(x^{(i)})\Delta x + \nabla f(x^{(i)}) + A^T\lambda = 0 \\ A\Delta x = 0 \quad \Leftrightarrow \quad \boxed{\begin{bmatrix} \nabla^2 f(x^{(i)}) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x^{(i)}) \\ 0 \end{bmatrix}}$$

Outline

1. Unconstrained Minimization

2. Constrained Minimization

Common Classes of Constrained Optimization Methods

- Gradient descent methods:

  Idea: Gradient gives direction of steepest local ascent
  $\rightarrow$ Make steps into anti-gradient direction

- Interior-point methods:

  Idea: Solve relaxed KKT system using Newton's method

- Active set methods:

  Idea: Iteratively identify set of active constraints

Outline

Constrained Minimization Using Gradient Methods

Consider the following constrained convex optimization problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathbb{Q} \end{array} \tag{P}$$

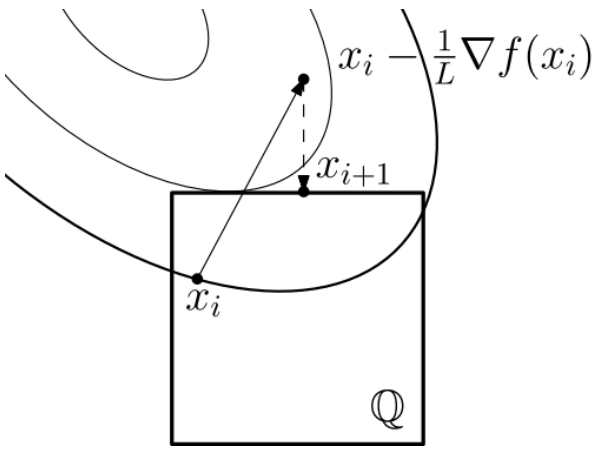
where $\mathbb{Q}$ is convex and f is convex and L-smooth.

$\Rightarrow$ Incorporate constraints in gradient step:

$$\boxed{x^{(i+1)} = \pi_{\mathbb{Q}}\left(x^{(i)} - h^{(i)} \nabla f(x^{(i)})\right)}$$

where $\pi_{\mathbb{Q}}$ is a **projection**:

$$\pi_{\mathbb{Q}}(y) \triangleq \arg\min_{x} \ \frac{1}{2}\|x - y\|_2^2 \quad \text{s.t. } x \in \mathbb{Q}$$



Can similarly choose $h^{(i)} = 1/L$,
convergence rates are as in the unconstrained case.

Implications for MPC

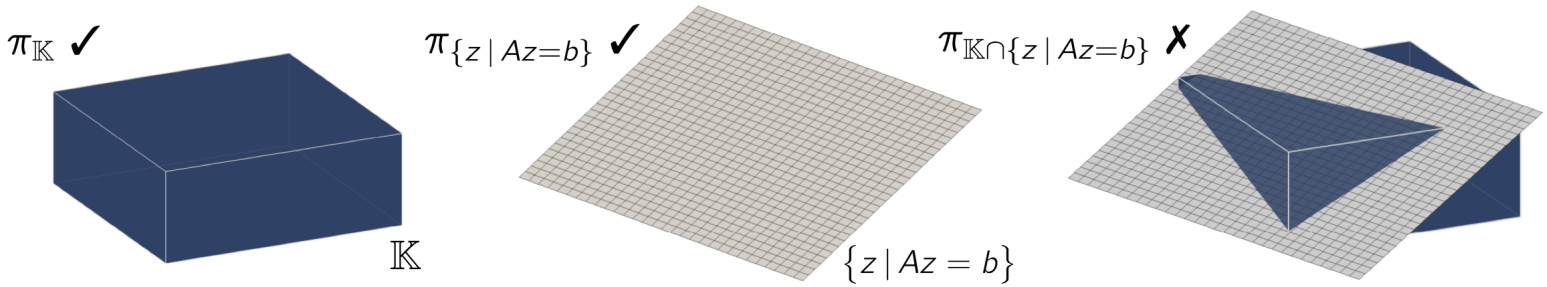
- MPC with simple input constraint set $\mathbb{U}$ (using "condensed" formulation, i.e. eliminate states):

$$\begin{aligned} \min \ & \frac{1}{2} u^T H u + x_0^T F u \\ \text{s.t. } & u \in \mathbb{U} \end{aligned}$$

- MPC with state constraints: Individual projections are easy, projection on intersection is not $\rightarrow$ dualize

$$\begin{aligned} \min \ & 1/2 \left(\sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i + x_N^T P x_N \right) \\ \text{s.t. } & x_{i+1} = A x_i + B u_i \\ & u_i \in \mathbb{U}_i, \quad x_i \in \mathbb{X}_i \end{aligned}$$

Illustration: $z \triangleq (x, u)$, $\mathbb{K} \triangleq \mathbb{X} \times \mathbb{U}$



Outline

2. Constrained Minimization
    - Projected Gradient Method
    - Interior-Point Methods
    - Software

Constrained Minimization Problem

Consider the following problem with inequality constraints

$$\begin{aligned} \text{min} \;\; & f(x) \\ \text{s.t.} \;\; & g_i(x) \leq 0, \; i = 1, \ldots, m \end{aligned}$$

Assumptions:

- f, g_i convex, twice continuously differentiable
- $f(x^*)$ is finite and attained
- strict feasibility: there exists a $\tilde{x}$ with

$$\tilde{x} \in \text{ dom } f, \quad g_i(\tilde{x}) < 0, i = 1, \ldots, m$$

- feasible set is closed and compact

Idea: There exist many methods for unconstrained minimization

$\Rightarrow$ Reformulate problem as an unconstrained problem

Primal-Dual Interior-point Methods

Idea: Iteratively solve the **relaxed** KKT system

$$\begin{aligned} Cx^* &= d \\ g_i(x^*) + s_i^* &= 0, \quad i = 1, \dots, m \\ \nabla f(x^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(x^*) + C^T \nu^* &= 0 \\ \lambda_i^* g_i(x^*) &= -\kappa, \quad i = 1, \dots, m \\ \lambda_i^*, s_i^* &\geq 0, \quad i = 1, \dots, m \end{aligned} \qquad (**)$$

where we introduced slack $s \in \mathbb{R}^m$.

Idea: leave dual multipliers λ_i^* as variables (before, they were implicitly defined by primal log barrier)[2]:

- Solve the primal and dual problem simultaneously via (**)
- Primal-dual central path $\triangleq \{(x, \nu, \lambda, s) \mid (**) \text{ holds}\}$
- Follow central path to solution by reducing κ to zero
- Solve (**) by Newton method (with additional "safeguards" & line search)

[2] See e.g. *[Stephen Wright, Primal-dual Interior-point Methods, SIAM 1997]*

Primal-Dual Search Direction Computation

At each iteration, linearize (**) and solve

$$\begin{bmatrix} H(x,\lambda) & C^T & G(x)^T & 0 \\ C & 0 & 0 & 0 \\ G(x) & 0 & 0 & I \\ 0 & 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\nu \\ \Delta\lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla f(x) + C^T\nu + G(x)^T\lambda \\ Cx - d \\ g(x) + s \\ S\lambda - v \end{bmatrix}$$

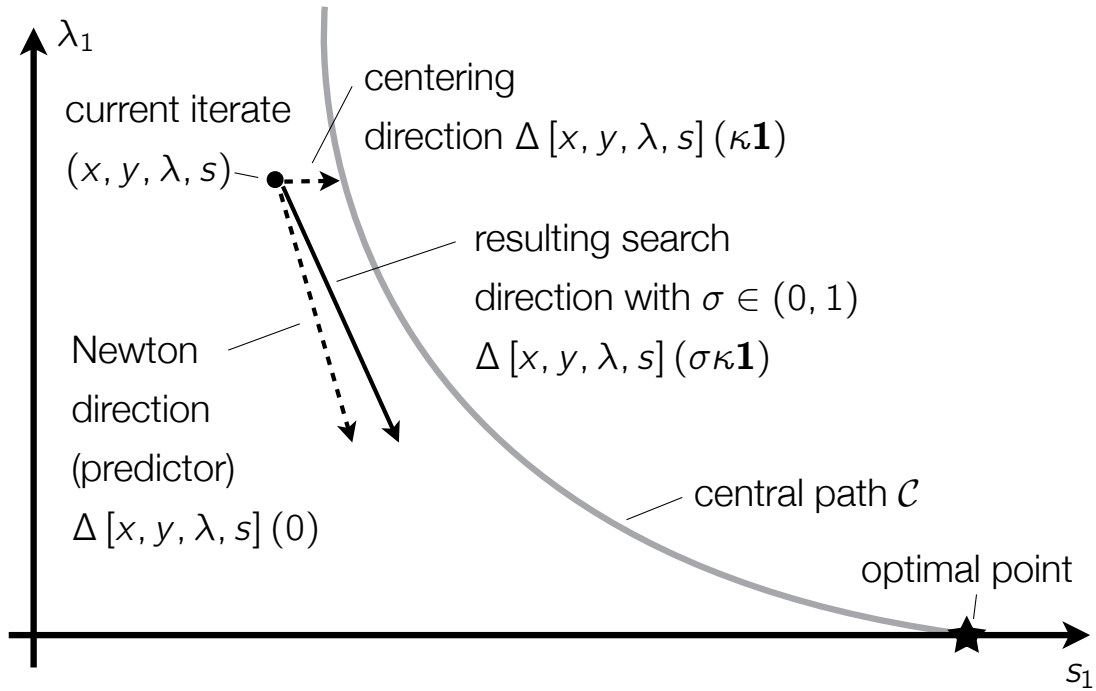
where $S \triangleq \text{diag}(s_1, \ldots, s_m)$ and $\Lambda \triangleq \text{diag}(\lambda_1, \ldots, \lambda_m)$, the (1,1) block in the coefficient matrix is

$$H(x,\lambda) \triangleq \nabla^2 f(x) + \sum_{i=1}^{m} \lambda_i \nabla^2 g_i(x)$$

and the vector $v \in \mathbb{R}^m$ allows for a modification of the right-hand side. Call resulting direction $\Delta[x, \nu, \lambda, s](v)$.

Search Directions in Primal-Dual Methods

Can generate different directions $\Delta[x, \nu, \lambda, s](\nu)$ depending on ν:



- $\nu = 0$: pure Newton direction ("predictor" or "affine-scaling")
- $\nu = \kappa\mathbf{1}$: centering direction, approach central path

$\Rightarrow$ Using linear combination via **centering parameter** $\sigma \in (0, 1)$ ensures fast convergence in theory & practice by tracking central path to solution

Outline

2. Constrained Minimization
    - Projected Gradient Method
    - Interior-Point Methods
    - Software

Software for Modeling Optimization Problems

Formulate optimization problem in mathematical language and pass to solver:

- **CVX** (cvxr.com/cvx): Matlab software for modeling convex problems
- **YALMIP** (users.isy.liu.se/johanl/yalmip): Matlab software for modeling convex and some non-convex optimization problems. MPC-Example:

```
u = sdpvar(repmat(nu,1,N),repmat(1,1,N));
constraints = []; objective = 0; x = x0;
for k = 1:N
    x = A*x + B*u{k};
    objective = objective + norm(Q*x,1) + norm(R*u{k},1);
    constraints = [constraints, -1 <= u{k}<= 1, -5<=x<=5];
end
```

- **AMPL** (www.ampl.com): industry standard, proprietary software. Supports basically all solvers.
- **GAMS** (www.gams.com): commercial high-level modeling system for large-scale optimization. Supports many different types of problems (LPs, QCQPs, MILPs, MINLPs, ...) and solvers

Software for Solving Convex Problems on Desktop PCs

General purpose solvers

- **SeDuMi** (`sedumi.ie.lehigh.edu`): widely used free solver, with Matlab interface
- **SDPT3** (`www.math.nus.edu.sg/~mattohkc/sdpt3`): Matlab software, free (GPL)
- **CVXOPT** (`abel.ee.ucla.edu/cvxopt`): free Python solver, allows customization of linear system solvers
- **IBM CPLEX**: industry standard for (MI)LPs and (MI)QCQPs (commercial)
- **Gurobi** (`www.gurobi.com`): commercial (MI)SOCP solver, by creators of CPLEX, strong Python support
- **MOSEK** (`www.mosek.com`): fastest commercial solver for second-order cone programs
- **OOQP** (`pages.cs.wisc.edu/~swright/ooqp`): object-oriented QP solver (needs LAPACK/BLAS)

All listed solvers are based on interior-point methods (6 out of 7 primal-dual)

Convex Optimization Solvers for Embedded Platforms

- **qpOASES** (`www.kuleuven.be/optec/software/qpOASES`): active set solver (LGPL)
- **HPIPM** (`https://github.com/giaf/hpipm`): structure-exploiting primal-dual IPM (BSD)
- **ECOS** (`github.com/embotech/ecos`): Sparse SOCP solver, 800 lines of library free C code, Python & Matlab interface

Code generation: – generate problem-specific C-code:

- **CVXGEN** (`cvxgen.com`): code generation for small QPs, extremely fast, code can get large
- **FORCES PRO** (`www.embotech.com`): Code generation for interior-point, gradient methods, ADMM, non-convex solvers (NLPs), problems with binary variables