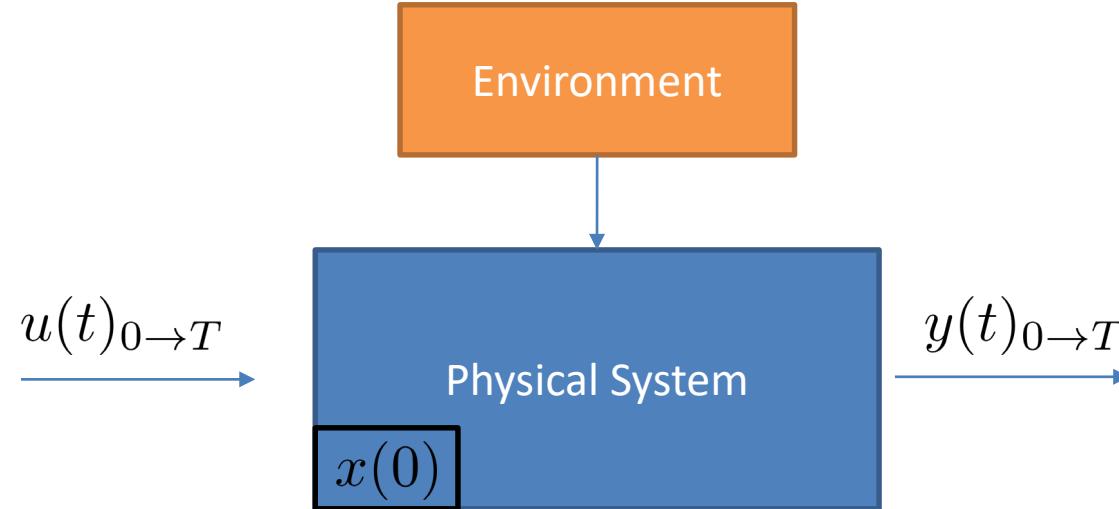


Introduction to Modeling

Outline

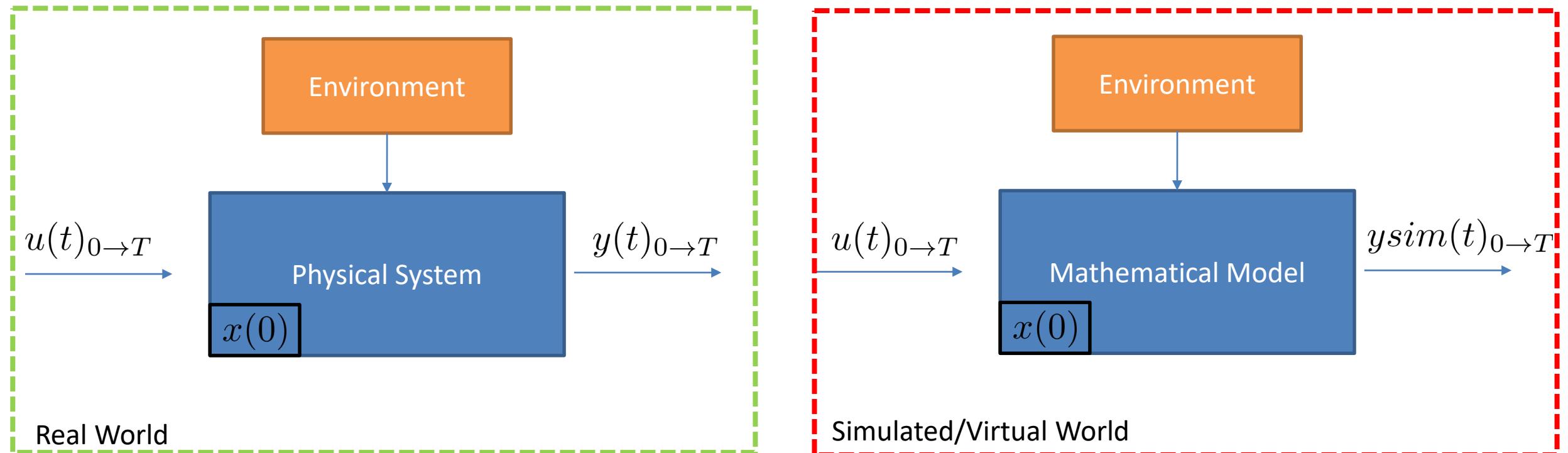
- What is a model?
- Need of Models
- High-fidelity/low-fidelity and Models for Control
- Key Concepts

Real Physical System - Block Abstraction



- $u(t)_{0 \rightarrow T}$ is the input to the system from time 0 to T
- $y(t)_{0 \rightarrow T}$ is the system output from time 0 to T
- $x(0)$ is the initial state
- The Environment affect the Physical System behavior

Mathematical Modeling

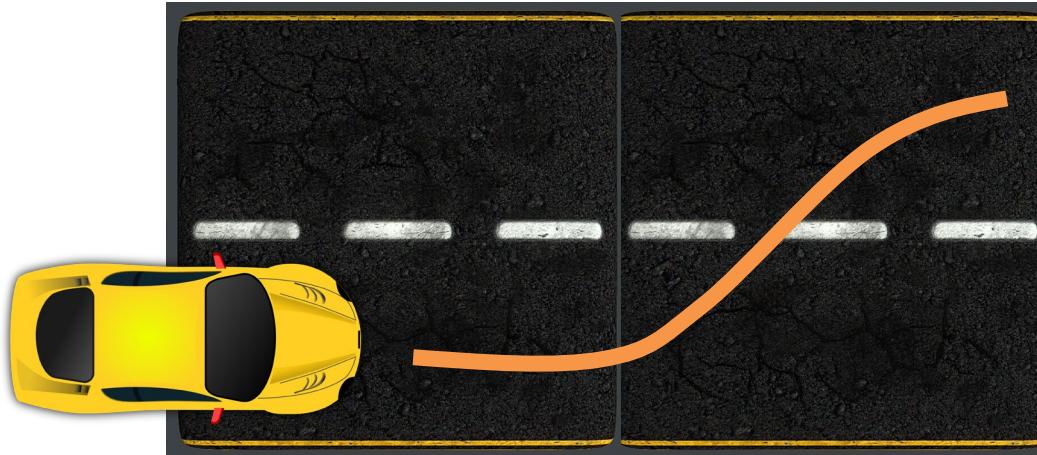


- In general, modelers have the objective

$$y(t)_{0 \rightarrow T} \simeq y_{sim}(t)_{0 \rightarrow T}$$

- For all possible, inputs, initial states and environments

Example: Build a Vehicle Model in Order to Perform a ...



- Think of three examples: ... lane change study ,comfort study,pedestrian detection study
- $u(t)_{0 \rightarrow T}$ is ...
- $y(t)_{0 \rightarrow T}$ is ...
- $x(0)$ is ...
- The Environment affecting is ...

Need of Models

High Fidelity

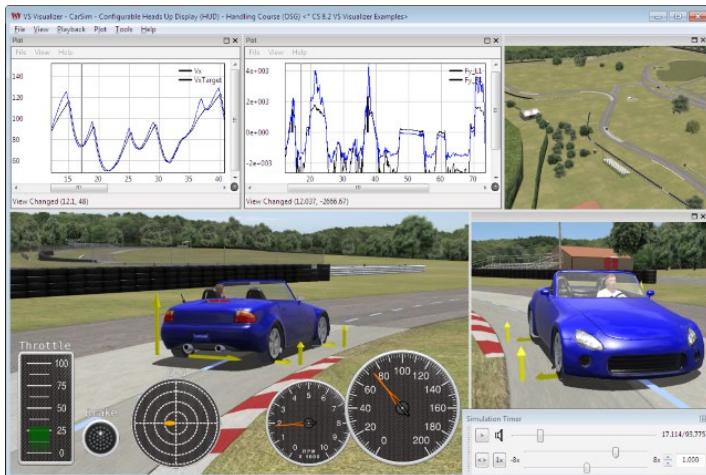
Cheaper and Safer

- System Design
- Control Design
- Any Algorithm Design

Low-Fidelity/Control Oriented Models

- Model-Based Control Design

High-fidelity/low-fidelity and Models for Control



CarSim

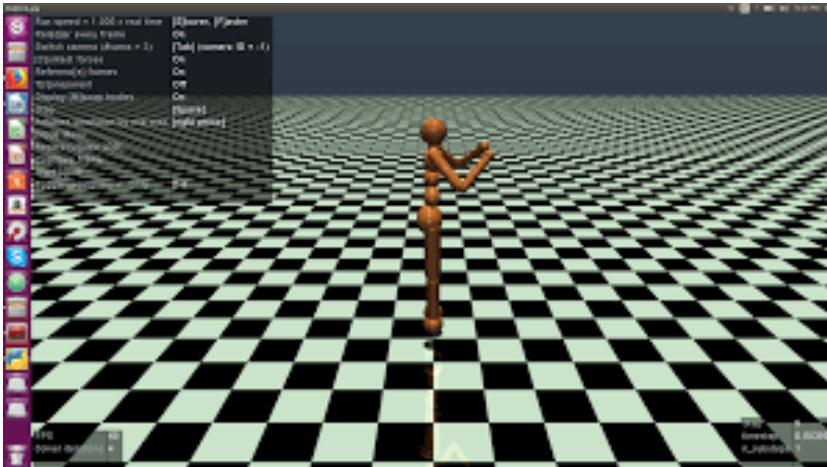


Carla

$$\begin{aligned}\dot{x} &= v \cos(\psi + \beta) \\ \dot{y} &= v \sin(\psi + \beta) \\ \dot{\psi} &= \frac{v}{l_r} \sin(\beta) \\ \dot{v} &= \textcolor{red}{a} \\ \beta &= \text{atan} \left(\frac{l_r}{l_f + l_r} \tan(\delta) \right)\end{aligned}$$

Kinematic bicycle model

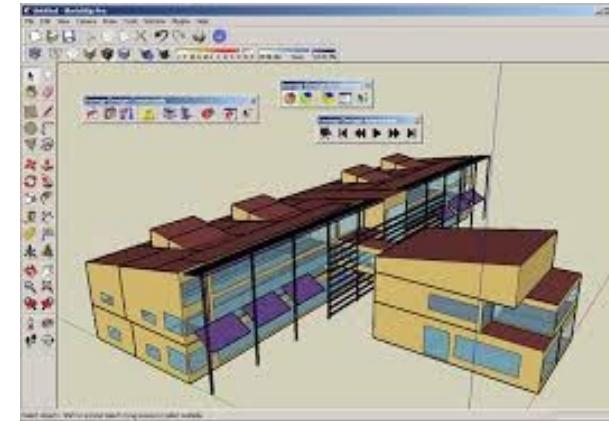
Same for Many Other Fields



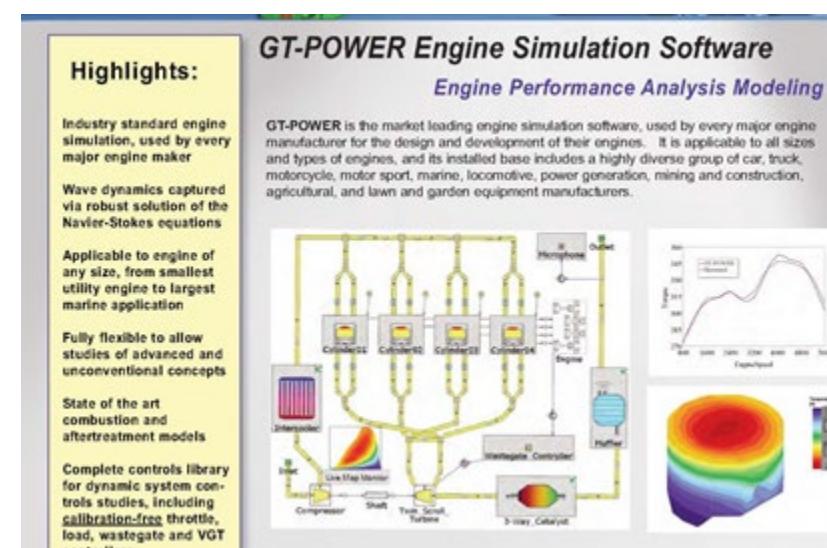
Mujoco, 3d Physic Engine



PVsyst for Photovoltaic Installations

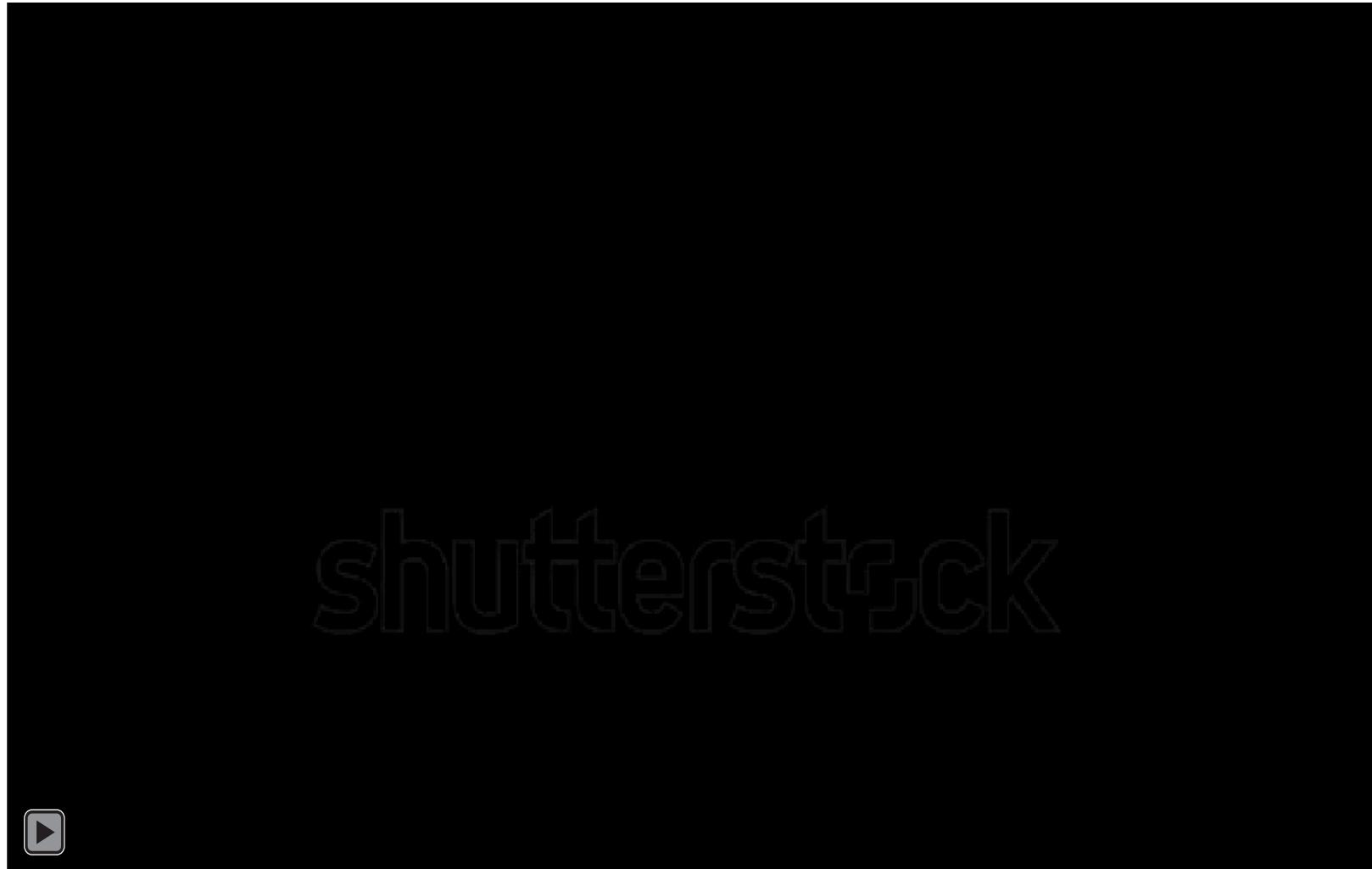


Energy plus for HVAC simulations



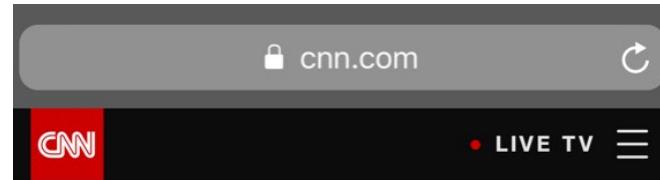
GT Power

Paper Towel Example

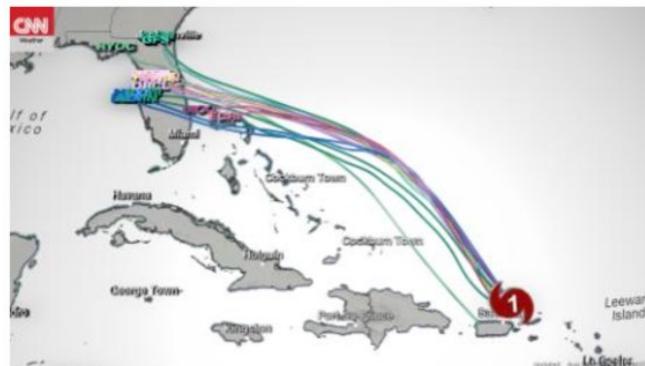


What is a good mathematical model of the 3D motion?

Summary Remark about “Modeling”



Making sense of the models



Spaghetti models show Dorian taking many destructive paths. Here's how you should read them

*The most important question:
“What do I need the model for?”*

Key Concepts

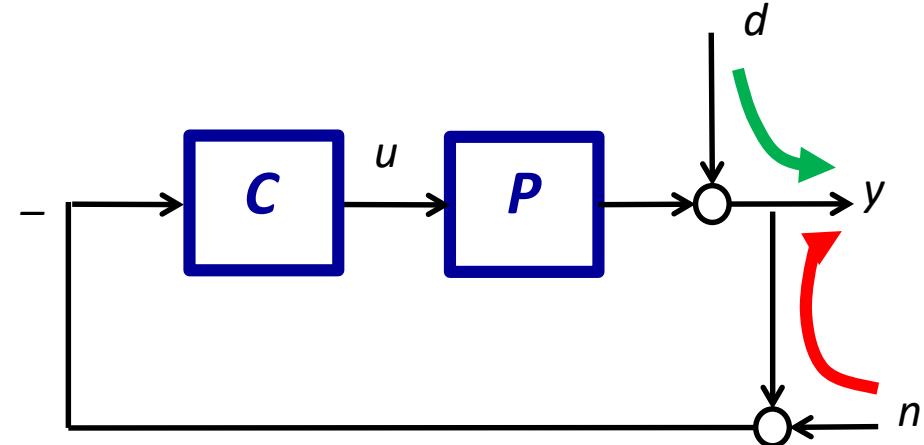
- *Notation*

- *State*

- Simulation Open loop and Closed-Loop

- Model-Based vs Model-Free

- Uncertainty



ODE Models

Francesco Borrelli, Manfred Morari

UC Berkeley

Institut für Automatik
ETH Zürich

Fall Semester 2020

Table of Contents

1. Use of ODE to Models Dynamical Systems
2. Basic Notation for ODE
3. Solving ODE
4. LTI Continuous-Time State Space Models
5. Discrete-Time Models

Table of Contents

1. Use of ODE to Models Dynamical Systems

2. Basic Notation for ODE

3. Solving ODE

4. LTI Continuous-Time State Space Models

5. Discrete-Time Models

Models of Dynamic Systems

- **Goal:** Introduce mathematical models to be used in Model Predictive Control (MPC) describing the behavior of dynamic systems
- Model classification: state space/transfer function, linear/nonlinear, time-varying/time-invariant, continuous-time/discrete-time, deterministic/stochastic
- If not stated differently, we use deterministic models
- Models of physical systems derived from first principles are mainly: nonlinear, time-variant, continuous-time (*)
- Target models for standard MPC are mainly: linear, time-invariant, discrete-time (†)
- Focus of this section is on how to 'transform' (*) to (†)

Table of Contents

1. Use of ODE to Models Dynamical Systems
2. Basic Notation for ODE
3. Solving ODE
4. LTI Continuous-Time State Space Models
5. Discrete-Time Models

Nonlinear, Time-Invariant, Continuous-Time, State Space ODE Model

$$\dot{x} = g(x, u)$$

$$y = h(x, u)$$

$x \in \mathbb{R}^n$	state vector	$g(x, u) : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$	system dynamics
$u \in \mathbb{R}^m$	input vector	$h(x, u) : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^p$	output function
$y \in \mathbb{R}^p$	output vector		

- Very general class of models
- Higher order ODEs can be easily brought to this form (next slide)
- Analysis and control synthesis generally hard → *linearization* to bring it to linear, time-invariant (LTI), continuous-time, state space form

Equivalence of one n -th order ODE and n 1-st order ODEs

$$x^{(n)} + g_n(x, \dot{x}, \ddot{x}, \dots, x^{(n-1)}) = 0$$

Define

$$x_{i+1} = x^{(i)}, \quad i = 0, \dots, n-1$$

Transformed system

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots && \vdots \\ \dot{x}_{n-1} &= x_n \\ \dot{x}_n &= -g_n(x_1, x_2, \dots, x_n)\end{aligned}$$

Pendulum Example

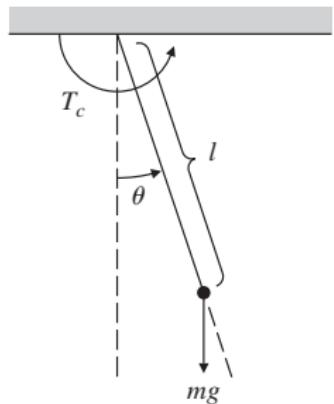
Example: Pendulum

Moment of inertia wrt. rotational axis $m l^2$

Torque caused by external force T_c

Torque caused by gravity $m g l \sin(\theta)$

System equation $m l^2 \ddot{\theta} = T_c - m g l \sin(\theta)$



Using $x_1 \triangleq \theta, x_2 \triangleq \dot{\theta} = \dot{x}_1$ and $u \triangleq T_c/m l^2$ the system can be brought to standard form

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{l} \sin(x_1) + u \end{bmatrix} = g(x, u)$$

Output equation depends on the measurement configuration, i.e. if θ is measured then $y = h(x, u) = x_1$.

Table of Contents

1. Use of ODE to Models Dynamical Systems

2. Basic Notation for ODE

3. Solving ODE

4. LTI Continuous-Time State Space Models

5. Discrete-Time Models

Solving ODE

- Covered in your Math/Numerical Analysis classes
- Interested in IVP
- Analytical vs Numerical Solution
- Ill-conditioned problems not addressed in this class

Pendulum Example

Use this example [!\[\]\(e664663439e6ace920117d2b3d75b910_img.jpg\) Open in Colab](#) to learn how to:

- Simulate a nonlinear ODE
- Control accuracy by using 'rtol' or feeding a fixed time vector
- Add friction to the model
- Simulate free response and forced response.
- Close the loop with a simple controller.
- Add measurement noise
- Add input uncertainty and constraints

You can also play with the double pendulum here! (to generate the animation will take some time on Colab) [!\[\]\(c6747d08ffcbb3c0701a343df825d2f1_img.jpg\) Open in Colab](#)

Table of Contents

1. Use of ODE to Models Dynamical Systems

2. Basic Notation for ODE

3. Solving ODE

4. LTI Continuous-Time State Space Models

5. Discrete-Time Models

LTI Continuous-Time State Space Models - Definition

$$\begin{aligned}\dot{x} &= A^c x + B^c u \\ y &= C^c x + D^c u\end{aligned}$$

$x \in \mathbb{R}^n$	state vector	$A^c \in \mathbb{R}^{n \times n}$	system matrix
$u \in \mathbb{R}^m$	input vector	$B^c \in \mathbb{R}^{n \times m}$	input matrix
$y \in \mathbb{R}^p$	output vector	$C^c \in \mathbb{R}^{p \times n}$	output matrix
		$D^c \in \mathbb{R}^{p \times m}$	throughput matrix

- Vast theory exists for the analysis and control synthesis of linear systems
- Exact solution (next slide)

LTI Continuous-Time State Space Models- Solution

Solution to linear ODEs

- Consider the ODE (written with explicit time dependence)
 $\dot{x}(t) = A^c x(t) + B^c u(t)$ with initial condition $x_0 \triangleq x(t_0)$, then its solution is given by

$$x(t) = e^{A^c(t-t_0)} x_0 + \int_{t_0}^t e^{A^c(t-\tau)} B^c u(\tau) d\tau$$

where $e^{A^c t} \triangleq \sum_{n=0}^{\infty} \frac{(A^c t)^n}{n!}$

LTI Continuous-Time State Space Models- Free and Forced Response

Solution to linear ODEs

- **free response**

$$x(t) = e^{A^c(t-t_0)}x_0$$

- **forced response**

$$x(t) = \int_{t_0}^t e^{A^c(t-\tau)} B^c u(\tau) d\tau$$

LTI Continuous-Time State Space Models- Example

Use this example [!\[\]\(a2f9594c2c856a03df90ec4016df4a10_img.jpg\) Open in Colab](#) to learn how to:

- Simulate a linear ODE
- Understand explicit solution
- Understand free plus forced response

Table of Contents

1. Use of ODE to Models Dynamical Systems
2. Basic Notation for ODE
3. Solving ODE
4. LTI Continuous-Time State Space Models
5. Discrete-Time Models

Nonlinear, Time-Invariant, Discrete-Time, State Space Models

- Nonlinear discrete-time systems are described by difference equations

$$\begin{aligned}x(k+1) &= g(x(k), u(k)) \\y(k) &= h(x(k), u(k))\end{aligned}$$

$x \in \mathbb{R}^n$	state vector	$g(x, u) : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$	system dynamics
$u \in \mathbb{R}^m$	input vector	$h(x, u) : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^p$	output function
$y \in \mathbb{R}^p$	output vector		

LTI Discrete-Time State Space Models - Definition

- Linear discrete-time systems are described by linear difference equations

$$\begin{aligned}x(k+1) &= Ax(k) + Bu(k) \\y(k) &= Cx(k) + Du(k)\end{aligned}$$

- Inputs and outputs of a discrete-time system are defined only at discrete time points, i.e. its inputs and outputs are sequences defined for $k \in \mathbb{Z}^+$
- Discrete time systems describe either
 - 1 Inherently discrete systems, eg. bank savings account balance at the k -th month
$$x(k+1) = (1 + \alpha)x(k) + u(k)$$
 - 2 'Transformed' continuous-time system

LTI Discrete-Time State Space Models - Solution

Linear discrete-time systems are described by linear difference equations

$$\begin{aligned}x(k+1) &= Ax(k) + Bu(k) \\y(k) &= Cx(k) + Du(k)\end{aligned}$$

Solution to LTI Discrete-Time Models

- free response

$$x(k) = A^k x_0$$

- forced response

$$x(k) = \sum_{i=0}^{k-1} A^{k-i-1} Bu(i)$$

LTI Discrete-Time State Space Models- Example

Use this example [!\[\]\(e51810ff30b37de53380ac76c06eed8d_img.jpg\) Open in Colab](#) to learn how to:

- Simulate a Linear and Nonlinear discrete-time system
- Check free and forced response formulas

ODE Models - Linearization

Francesco Borrelli, Manfred Morari

UC Berkeley

Institut für Automatik
ETH Zürich

Fall Semester 2020

Linearization Intro

- **Problem:** Most physical systems are nonlinear but linear systems are much better understood
- Nonlinear systems can be well approximated by a linear system in a 'small' neighborhood around a point in state space
- **Idea:** Control keeps the system around some operating point → replace nonlinear by a linearized system around operating point

Linearization the static case

First order Taylor expansion of $f(\cdot)$ around \bar{x}

$$f(x) \approx f(\bar{x}) + \left. \frac{\partial f}{\partial x} \right|_{x=\bar{x}} (x - \bar{x}), \text{ with } \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Definition of Equilibrium Point

Given

$$\begin{aligned}\dot{x} &= g(x, u) \\ y &= h(x, u)\end{aligned}$$

(x_s, u_s) is an equilibrium point (or pair) iff u_s keeps the system around stationary operating point x_s

$$\rightarrow \dot{x}_s = g(x_s, u_s) = 0, y_s = h(x_s, u_s)$$

Linearization Formula

$$\dot{x} = \underbrace{g(x_s, u_s)}_{=0} + \underbrace{\frac{\partial g}{\partial x} \Big|_{\substack{x=x_s \\ u=u_s}}}_{=A^c} (x - x_s) + \underbrace{\frac{\partial g}{\partial u} \Big|_{\substack{x=x_s \\ u=u_s}}}_{=B^c} (u - u_s)$$

$$\Rightarrow \dot{x} - \underbrace{\dot{x}_s}_{=0} = \Delta \dot{x} = A^c \Delta x + B^c \Delta u$$

$$y = \underbrace{h(x_s, u_s)}_{y_s} + \underbrace{\frac{\partial h}{\partial x} \Big|_{\substack{x=x_s \\ u=u_s}}}_{=C^c} (x - x_s) + \underbrace{\frac{\partial h}{\partial u} \Big|_{\substack{x=x_s \\ u=u_s}}}_{=D^c} (u - u_s)$$

$$\Rightarrow \underbrace{\Delta y}_{y - y_s} = C^c \Delta x + D^c \Delta u$$

Notation

- The linearized system is written in terms of *deviation* variables $\Delta x, \Delta u, \Delta y$
- Linearized system is only a good approximation for 'small' $\Delta x, \Delta u$
- Subsequently, instead of $\Delta x, \Delta u$ and Δy , x, u and y are used for brevity

Example: Linearization of pendulum equations

Recall the pendulum equations

$$\begin{aligned}\dot{x} &= \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{l} \sin(x_1) + u \end{bmatrix} = g(x, u) \\ y &= x_1 = h(x, u)\end{aligned}$$

Want to keep the pendulum around $x_s = (\pi/4, 0)'$ $\rightarrow u_s = \frac{g}{l} \sin(\pi/4)$

$$\begin{aligned}A^c &= \left. \frac{\partial g}{\partial x} \right|_{\substack{x=x_s \\ u=u_s}} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} \cos(\pi/4) & 0 \end{bmatrix}, \quad B^c = \left. \frac{\partial g}{\partial u} \right|_{\substack{x=x_s \\ u=u_s}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ C^c &= \left. \frac{\partial h}{\partial x} \right|_{\substack{x=x_s \\ u=u_s}} = [1 \ 0], \quad D^c = \left. \frac{\partial h}{\partial u} \right|_{\substack{x=x_s \\ u=u_s}} = 0\end{aligned}$$

Advanced Topics

- Linearization around a feasible trajectory
- Linearization around a desired trajectory
- Model Scheduling (or piecewise linear models)

ODE Models - Discretization

Francesco Borrelli, Manfred Morari

UC Berkeley

Institut für Automatik
ETH Zürich

Fall Semester 2020

In this Class We Work With Discrete Time Models

We will use:

- Nonlinear Discrete Time

$$\begin{aligned}x(k+1) &= g(x(k), u(k)) \\y(k) &= h(x(k), u(k))\end{aligned}$$

- or LTI Discrete Time

$$\begin{aligned}x(k+1) &= Ax(k) + Bu(k) \\y(k) &= Cx(k) + Du(k)\end{aligned}$$

Discretization

We call **discretization** the procedure of obtaining an “equivalent” DT model from a CT one.

Euler Discretization of Nonlinear Models

- 1 Given CT model

$$\begin{aligned}\dot{x}^c(t) &= g^c(x^c(t), u^c(t)) \\ y^c(t) &= h^c(x^c(t), u^c(t))\end{aligned}$$

- 2 Approximate $\frac{d}{dt}x^c(t) \simeq \frac{x^c(t+T_s) - x^c(t)}{T_s}$

- 3 T_s is the **sampling time**

- 4 Notation: $x(k) \triangleq x^c(t_0 + kT_s)$, $u(k) \triangleq u^c(t_0 + kT_s)$

- 5 Then DT model is

$$\begin{aligned}x(k+1) &= x(k) + T_s g^c(x(k), u(k)) = g(x(k), u(k)) \\ y(k) &= h^c(x(k), u(k)) = h(x(k), u(k))\end{aligned}$$

Under regularity assumptions, if T_s is small and CT and DT have “same” initial conditions and inputs, then outputs of CT and DT systems “will be close”

Euler Discretization of Linear Models

- 1 Given CT model

$$\begin{aligned}\dot{x}^c(t) &= A^c x(t) + B^c u(t) \\ y^c(t) &= C^c x(t) + D^c u(t)\end{aligned}$$

- 2 the DT model obtained with Euler discretization is

$$\begin{aligned}x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k)\end{aligned}$$

with $A = I + T_s A^c$, $B = T_s B^c$, $C = C^c$, $D = D^c$.

- This is the simplest one, there are a variety of discretization techniques. Only ZOH discussed next.

ZOH Discretization (1/2)

Discretization of LTI continuous-time state space models

- Recall the solution of the ODE $x(t) = e^{A^c(t-t_0)}x_0 + \int_{t_0}^t e^{A^c(t-\tau)}B^cu(\tau)d\tau$
- Choose $t_0 = t_k$ (hence $x_0 = x(t_0) = x(t_k)$), $t = t_{k+1}$ and use $t_{k+1} - t_k = T_s$ and $u(t) = u(t_k) \quad \forall t \in [t_k, t_{k+1})$

$$\begin{aligned} x(t_{k+1}) &= e^{A^cT_s}x(t_k) + \int_{t_k}^{t_{k+1}} e^{A^c(t_{k+1}-\tau)}B^cu(\tau)d\tau \\ &= \underbrace{e^{A^cT_s}x(t_k)}_{\triangleq A} + \underbrace{\int_0^{T_s} e^{A^c(T_s-\tau')}B^cu(\tau')d\tau'}_{\triangleq B} \\ &= Ax(t_k) + Bu(t_k) \end{aligned}$$

- We found the exact discrete-time model predicting the state of the continuous-time system at time t_{k+1} given $x(t_k)$, $k \in \mathbb{Z}_+$ under the assumption of a constant $u(t)$ during a sampling interval
- $B = (A^c)^{-1}(A - I)B^c$, if A^c invertible

ZOH Discretization (2/2)

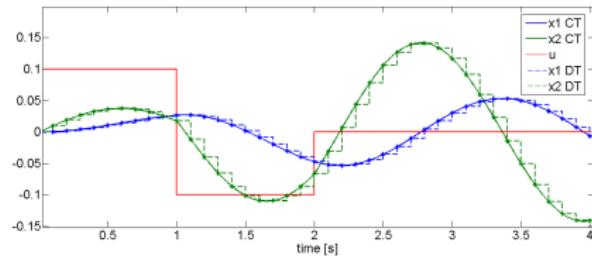
Example: Discretization of the linearized pendulum equations

Using $g/l = 10[s^{-2}]$ the pendulum equations linearized about $x_s = (\pi/4, 0)$ are given by

$$\dot{x}(t) = \begin{pmatrix} 0 & 1 \\ -10/\sqrt{2} & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t)$$

Discretizing the continuous-time system using the definitions of A and B , and $T_s = 0.1$ s, we get the following discrete-time system

$$x(k+1) = \begin{pmatrix} 0.965 & 0.099 \\ -0.699 & 0.965 \end{pmatrix} x(k) + \begin{pmatrix} 0.005 \\ 0.100 \end{pmatrix} u(k)$$



Convex and Non-Convex Optimization

Paul Goulart, Francesco Borrelli

Institut für Automatik
ETH Zürich

UC Berkeley

Fall Semester 2020

Table of Contents

1. Introduction
2. Describing and Solving Constrained Optimization
3. Properties, Terminology and Geometry of Optimization Problems
4. Software Tools for Optimization
5. Common Types of Optimization Problems
6. Examples

Why Study Optimization?

Optimization is about making **good decisions** or choices in a rigorous way, often subject to **constraints**. Applications appear everywhere in science, mathematics, and business:

- Managing a share portfolio
- Scheduling public transport
- Fitting a model to measured data
- Optimizing a supply chain
- Designing electronic circuit layouts
- Choosing worker shift patterns
- Shaping aerodynamic components
- Recovering images from raw MRI data

Our main application: **Control Design**

Describing an Optimization Problem

$$\begin{aligned} & \min_z f(z) \\ \text{subject to: } & z \in S \subseteq Z \end{aligned}$$

The problem has four ingredients:

- The vector z collects the **decision variables**
- The set Z is the **domain** of the decision variables
- The set $S \subseteq Z$ is the **constraint set**, and describes the **feasible** decisions.
- The **objective** function $f : Z \mapsto \mathbb{R}$ assigns a cost $f(z)$ to each decision z .

This problem can be written more compactly as

$$\min_{z \in S \subseteq Z} f(z)$$

Describing an Optimization Problem

$$\begin{aligned} \min_z \quad & f(z) \\ \text{subject to: } & z \in S \subseteq Z \end{aligned}$$

The problem has four ingredients:

- The vector z collects the **decision variables**
- The set Z is the **domain** of the decision variables
- The set $S \subseteq Z$ is the **constraint set**, and describes the **feasible** decisions.
- The **objective** function $f : Z \mapsto \mathbb{R}$ assigns a cost $f(z)$ to each decision z .

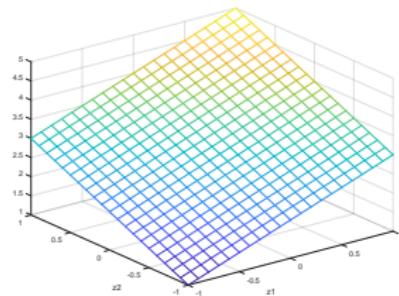
In unconstrained optimization $S = Z$.

Solving the optimization problem means to compute the least possible cost f^* and an associated optimal solution (or minimizer) z^*

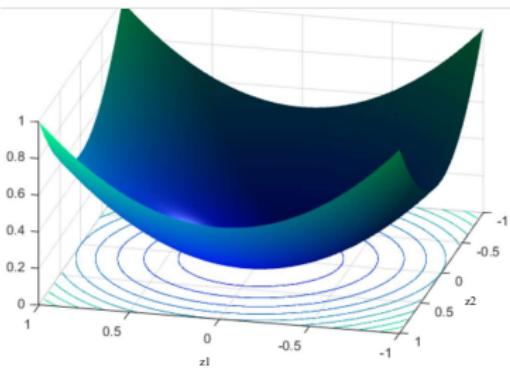
$$f(z^*) = f^*$$

A Few Classical Unconstrained Optimization Examples

Min of a linear function $c^T z + k$

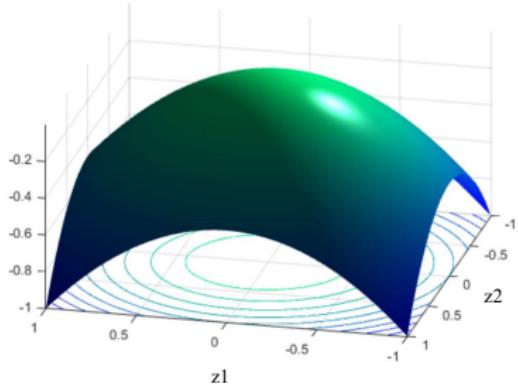


Min of a positive definite quadratic function
 $z^T H z + c^T z + k$

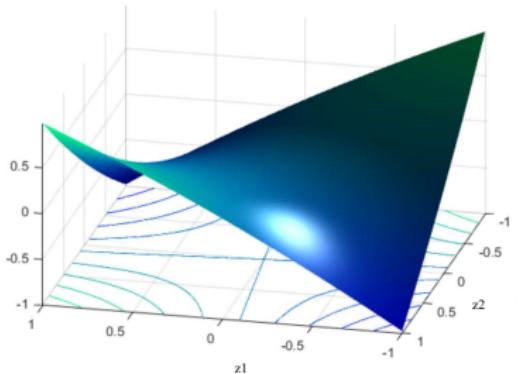


A Few Classical Unconstrained Optimization Examples

Min of a negative definite quadratic function
 $z^T H z + c^T z + k$



Min of an indefinite quadratic function
 $z^T H z + c^T z + k$



A Few Classical Unconstrained Optimization Examples

Min of **Wolfram** function $f(z) = \begin{vmatrix} z_1 - \sin(2z_1 + 3z_2) - \cos(3z_1 - 5z_2) \\ z_2 - \sin(z_1 - 2z_2) + \cos(z_1 + 3z_2) \end{vmatrix}$

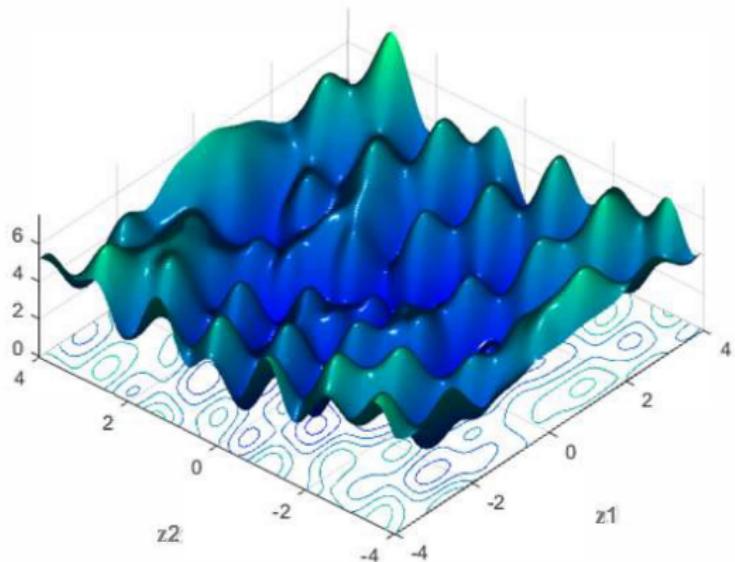


Table of Contents

1. Introduction
2. Describing and Solving Constrained Optimization
3. Properties, Terminology and Geometry of Optimization Problems
4. Software Tools for Optimization
5. Common Types of Optimization Problems
6. Examples

Describing an Optimization Problem

$$\begin{aligned} \min_z \quad & f(z) \\ \text{subject to: } & z \in S \subseteq Z \end{aligned}$$

The problem has four ingredients:

- The vector z collects the **decision variables**
- The set Z is the **domain** of the decision variables
- The set $S \subseteq Z$ is the **constraint set**, and describes the **feasible** decisions.
- The **objective** function $f : Z \mapsto \mathbb{R}$ assigns a cost $f(z)$ to each decision z .

This problem can be written more compactly as

$$\min_{z \in S \subseteq Z} f(z)$$

We call this a **nonlinear mathematical program** or just a **nonlinear program (NLP)**.

Describing an Optimization Problem

A more common problem format:

$$\begin{aligned} & \min_{z \in Z} f(z) \\ \text{subject to: } & g_i(z) \leq 0 \quad i = 1, \dots, m \\ & h_j(z) = 0 \quad j = 1, \dots, p \end{aligned}$$

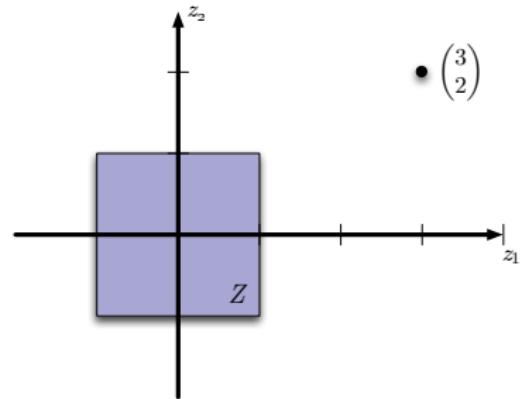
Defined by the following **problem data**:

- **Objective function** $f : Z \rightarrow \mathbb{R}$
- **Domain** $Z \subseteq \mathbb{R}^s$ of the objective function, from which the decision variable $z := (z_1; z_2; \dots; z_s)$ must be chosen.
- Optional **inequality constraint functions** $g_i : \mathbb{R}^s \rightarrow \mathbb{R}$, for $i = 1, \dots, m$
- Optional **equality constraint functions** $h_j : \mathbb{R}^s \rightarrow \mathbb{R}$, for $j = 1, \dots, p$

NB: Any *maximization* problem can be written this way using a change of sign.

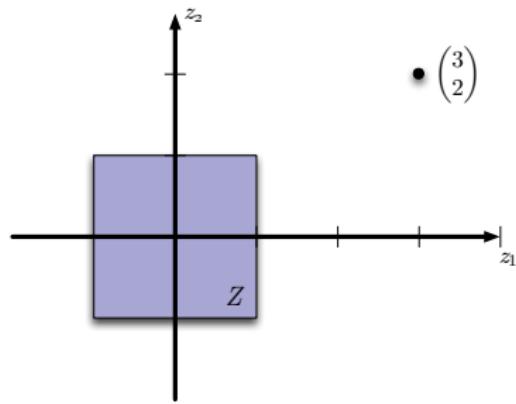
A Simple Example

Problem : In \mathbb{R}^2 , find the point in the unit box S closest to the point $(z_1, z_2) = (3, 2)$.



A Simple Example

Problem : In \mathbb{R}^2 , find the point in the unit box S closest to the point $(z_1, z_2) = (3, 2)$.

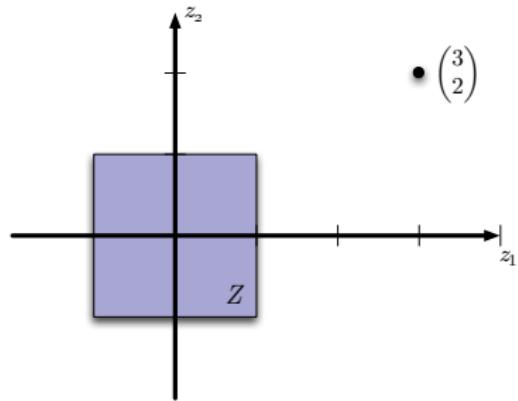


Three Major Steps

- How do I **transform** it into a formal mathematical optimization problem (next called “Mathematical Programming” problem)?
- How do I **solve** it the optimization problem?
- How do I know the solution is right (next called ==**certification** process)?

A Simple Example

Problem : In \mathbb{R}^2 , find the point in the unit box S closest to the point $(z_1, z_2) = (3, 2)$.



Same problem in standard format:

$$\min_{(z_1, z_2) \in \mathbb{R}^2} (z_1 - 3)^2 + (z_2 - 2)^2$$

subject to:

$z_1 \leq 1$
$-z_1 \leq 1$
$z_2 \leq 1$
$-z_2 \leq 1$

Table of Contents

1. Introduction
2. Describing and Solving Constrained Optimization
3. Properties, Terminology and Geometry of Optimization Problems
4. Software Tools for Optimization
5. Common Types of Optimization Problems
6. Examples

Properties of Optimization Problems

Consider the **Nonlinear Program** (NLP)

$$J^* = \min_{z \in \mathcal{S}} f(z)$$

Notation:

- If $J^* = -\infty$, then the problem is **unbounded below**.
- If the set \mathcal{S} is empty, then the problem is **infeasible** (and we set $J^* := +\infty$).
- If $\mathcal{S} = \mathbb{Z}$, the problem is **unconstrained**.
- There might be more than one solution. The set of solutions is:

$$\arg \min_{z \in \mathcal{S}} f(z) := \{z \in \mathcal{S} \mid f(z) = J^*\}$$

Terminology

Feasible point: A vector $z \in \mathcal{S}$ satisfying the inequality and equality constraints, i.e. $g_i(z) \leq 0$ for $i = 1, \dots, m$, $h_j(z) = 0$ for $j = 1, \dots, p$.

Strictly feasible point: A vector $z \in Z$ satisfying the inequality constraints strictly, i.e. $g_i(z) < 0$ for $i = 1, \dots, m$.

Optimal value: The lowest possible objective value, $f(z^*)$. Denoted by f^* (or p^* , or J^*).

Local optimality: z^* is locally optimal if there exists an $R > 0$ such that $z = z^*$ is optimal for

$$\min_{z \in Z} f(z)$$

subject to:

$g_i(z) \leq 0$	$i = 1, \dots, m,$
$h_j(z) = 0$	$j = 1, \dots, p$
$\ z - z^*\ _2 \leq R$	

Terminology

Optimal solution: Any *feasible* $z^* \in Z$ such that $f(z^*) \leq f(z)$ for all *feasible* $z \in Z$.

Local optimum: a point z_{local}^* that is optimal within a neighbourhood $\|z - z_{\text{local}}^*\| \leq R$.

Technical point: The optimal value is called the **infimum**. A vector z^* that achieves the optimal value is a **minimizer** or optimal solution. There might be more than one minimizer, or none at all.

What might go wrong?

It is possible that no minimizer will exist:

- If the constraints are inconsistent, then the problem is **infeasible**. Example:

$$\min_{z \in \mathbb{R}} z^2$$

$$\begin{aligned} \text{subject to: } z &\leq -1 \\ z &\geq 1 \end{aligned}$$

- It might be possible to make $f(z)$ arbitrarily negative without violating any of the constraints. Then the problem is referred to as **unbounded**. Example:

$$\min_{z \in \mathbb{R}} z$$

$$\text{subject to: } z \leq 0$$

- The value J^* might be finite, but there is no z that achieves it. Example:

$$\inf_{z \in \mathbb{R}} e^{-z}$$

$$\text{subject to: } z \geq 0$$

The optimal value $J^* = 0$ exists, but there are no optimal solutions.

Active, Inactive and Redundant Constraints

Consider the standard problem

$$\min_{z \in Z} f(z)$$

subject to:

$g_i(z) \leq 0$	$i = 1, \dots, m$
$h_j(z) = 0$	$j = 1, \dots, p$

- The i^{th} inequality constraint $g_i(z) \leq 0$ is **active** at \bar{z} if $g_i(\bar{z}) = 0$. Otherwise it is **inactive**.
- Equality constraints are always active.
- A **redundant** constraint is one that does not change the feasible set. This implies that removing a redundant constraint does not change the solution. Example:

$$\min_{z \in \mathbb{R}} f(z)$$

subject to:

$z \leq 1$	
$z \leq 2$	(redundant)

Implicit and Explicit Constraints

The constraints $g_i(z) \leq 0$, $i = 1, \dots, m$ and $h_j(z) = 0$, $j = 1, \dots, p$ are referred to as the **explicit constraints** of the optimization problem. However, the *domains* of the objective function f and constraint functions also define an **implicit constraint** on z :

$$z \in \text{dom}(f) \cap \bigcap_{i=1}^m \text{dom}(g_i) \cap \bigcap_{j=1}^p \text{dom}(h_j)$$

If a problem has $m = 0$ and $p = 0$, it is referred to as an **unconstrained problem**, although the limited domain of the *objective* function may still represent an implicit constraint.

Example:

$$\min_z f(z) = - \sum_{i=1}^k \log(a_i^\top z - b_i)$$

is unconstrained but still has the implicit constraint that $a_i^\top z > b_i$ for $i = 1, \dots, k$. In other words the constraint set $z \in Z = \{z \in \mathbb{R}^s \mid a_i^\top z > b_i, i = 1, \dots, k\}$ is implied by the domain of f .

Feasibility Problem

The “constraint satisfiability” problem

$$\begin{aligned} & \underset{z \in Z}{\text{find}} \quad z \\ \text{subject to: } & g_i(z) \leq 0 \quad i = 1, \dots, m \\ & h_j(z) = 0 \quad j = 1, \dots, p \end{aligned}$$

is a special case of the general optimization problem:

$$\begin{aligned} & \underset{z \in Z}{\min} \quad 0 \\ \text{subject to: } & g_i(z) \leq 0 \quad i = 1, \dots, m \\ & h_j(z) = 0 \quad j = 1, \dots, p \end{aligned}$$

- $p^* = 0$ if the constraints are feasible (consistent). Every feasible z is optimal.
- $p^* = \infty$ otherwise.

Geometry of an Optimization Problem

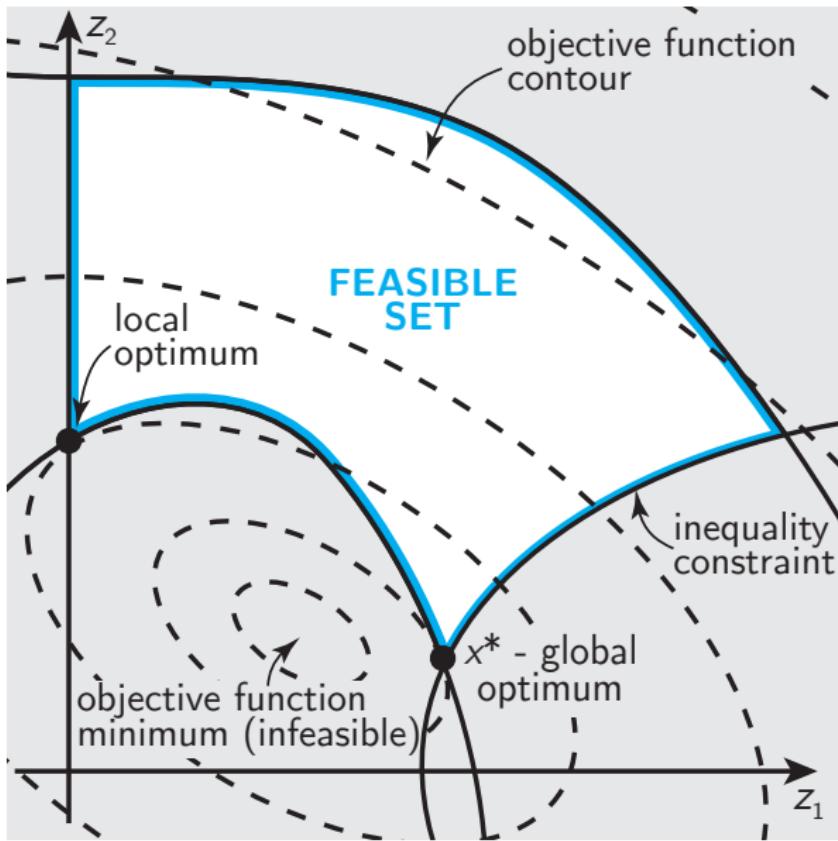


Table of Contents

1. Introduction
2. Describing and Solving Constrained Optimization
3. Properties, Terminology and Geometry of Optimization Problems
- 4. Software Tools for Optimization**
5. Common Types of Optimization Problems
6. Examples

Solving real-world problems using optimization:

- Model: Determine decision variables from real-world choices available
- Model: Choose an objective function
- Model: Identify constraints affecting choices
- (Try to) write the problem as a standard optimization problem
- Code problem
 - (A) in the format required by **solver** or
 - (B) in an environment (often called **math programming language**) interfaced to a **solver**
- Interpret the solution generated by the solver
- If the solver fails to provide a sensible solution:
 - Simplify model
 - Improve model
 - Reformulate the mathematical description of the model
 - Lower your expectations (e.g. fewer/looser constraints)

Solvers

Optimization problems only have an obvious or analytical solution in special or trivial cases. In general a solution can only be found using a piece of software called a **solver**:

Problem data → **Solver** → Solution + diagnostic information

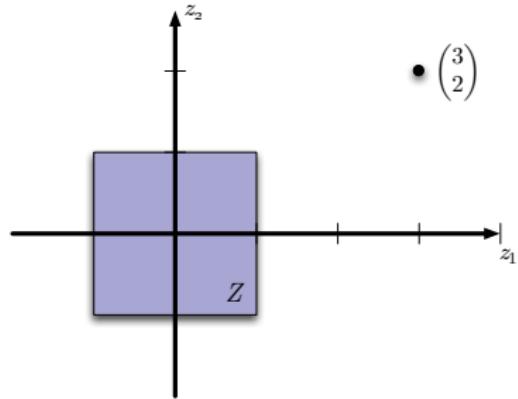
Solvers are typically written to deal with problems where the objective function f and the constraints g_i and h_j have specific forms. Choosing an appropriate solver is essential, although some are quite versatile.

Typical factors influencing the choice of solver:

- Are the objective and constraint functions **linear**?
- Are any of the decision variables restricted to **integers**?
- Is the problem **convex**? (*More on this later*)
- If the problem is large, does it have a special **structure**?

A Simple Example

Problem : In \mathbb{R}^2 , find the point in the unit box S closest to the point $(z_1, z_2) = (3, 2)$.



Same problem in standard format:

$$\min_{(z_1, z_2) \in \mathbb{R}^2} (z_1 - 3)^2 + (z_2 - 2)^2$$

subject to:

$$\begin{aligned} z_1 &\leq 1 \\ -z_1 &\leq 1 \\ z_2 &\leq 1 \\ -z_2 &\leq 1 \end{aligned}$$

Solving using a Solver or a Mathematical Programming Language

This  example will show how to solve the simple optimization problem by

- using a Mathematical Programming Solver
- using a Mathematical Programming Language

Software Tools for Optimization

A **modified version** of the simple optimization problem:

$$\min_{z_1, z_2} |z_1 + 5| + |z_2 - 3|$$

$$\begin{aligned}\text{subject to: } & 2.5 \leq z_1 \leq 5 \\ & -1 \leq z_2 \leq 1\end{aligned}$$

-
- This problem is equivalent to a linear program (more on this later).
 - Variety of **solvers** for solving LPs and QPs (and other standard types):
 - Examples: **OSQP, IPOPT, qpOASES, CPLEX, Gurobi, GLPK, XPRESS, qpOASES, OOQP, FORCES, SDPT3, Sedumi, MOSEK, MATLAB - linprog**
 - There is no standard interface to solvers – they are almost all different.
 - General purposes **programming languages** allow easy switching between solvers:
 - Examples: **Pyomo, CVX, Yalmip, GAMS, AMPL**

Example of Matlab Software Tools for Optimization

A simple optimization problem:

$$\min_{z_1, z_2} |z_1 + 5| + |z_2 - 3|$$

subject to: $2.5 \leq z_1 \leq 5$
 $-1 \leq z_2 \leq 1$

The YALMIP toolbox for Matlab (from ETH / Linköping):

```
%make variables
sdpvar z1 z2;
%define cost function
f = abs(z1 + 5) + abs(z2 - 3);
%define constraints
S = set(2.5 <= z1 <= 5) + ...
    set( -1 <= z2 <= 1);
%solve
solvesdp(S,f);
```

Example of Matlab Software Tools for Optimization

A simple optimization problem:

$$\min_{z_1, z_2} |z_1 + 5| + |z_2 - 3|$$

$$\begin{aligned}\text{subject to: } & 2.5 \leq z_1 \leq 5 \\ & -1 \leq z_2 \leq 1\end{aligned}$$

The CVX toolbox for Matlab (from Stanford):

```
cvx_begin
    %define cost function
    variables z1 z2
    %define constraints
    minimize(abs(z1 + 5) + abs(z2 - 3))
    subject to
        2.5 <= z1 <= 5
        -1 <= z2 <= 1
cvx_end      %solves automatically
```

Table of Contents

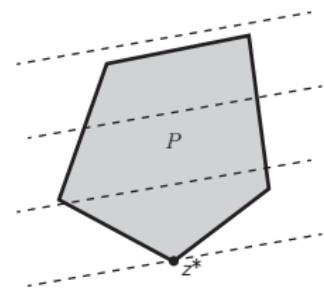
1. Introduction
2. Describing and Solving Constrained Optimization
3. Properties, Terminology and Geometry of Optimization Problems
4. Software Tools for Optimization
5. Common Types of Optimization Problems
6. Examples

“Easier” problems: Linear Programs

Linear Program (LP): Linear cost and constraint functions; feasible set is a polyhedron.

$$\min_z \quad c^\top z$$

$$\begin{aligned} \text{subject to: } & Gz \leq h \\ & Az = b \end{aligned}$$



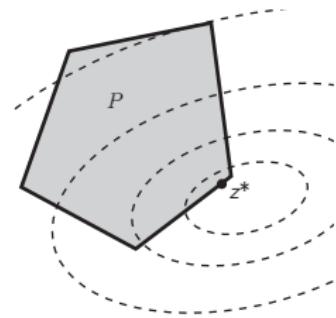
Linear optimization on a polytope.

“Easier” problems: Convex Quadratic Programs

Convex Quadratic Program (QP): Quadratic cost and linear constraint functions; feasible set is a polyhedron. Convex if $P \succeq 0$.

$$\min_z \quad \frac{1}{2} z^\top P z + q^\top z$$

$$\begin{aligned} \text{subject to: } & Gz \leq h \\ & Az = b \end{aligned}$$



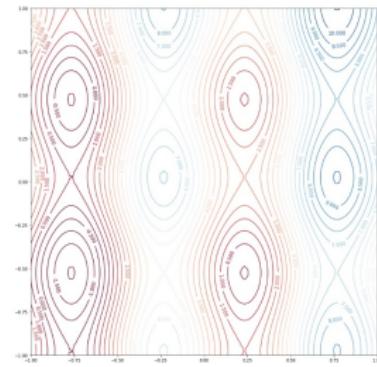
Convex quadratic optimization on a polytope.

“Harder” problems: Nonconvex Programs

Nonlinear Program:

$$\min_z f(z)$$

subject to: $g(z) \leq 0$
 $h(z) = 0$



$$\min_{z_1, z_2} 3 \sin(-2\pi z_1) + 2z_1 + 4 + \cos(2\pi z_2) + z_2$$

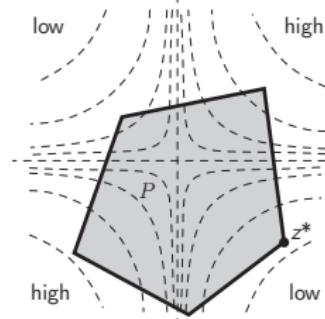
$$\text{s.t. } -1 \leq z_1 \leq 1 \\ -1 \leq z_2 \leq 1$$

“Harder” problems: Nonconvex Quadratic Programs

Nonconvex Quadratic Program: Quadratic cost and linear constraint functions; feasible set is a polyhedron.
 $P \preceq 0$.

$$\min_z \quad \frac{1}{2} z^\top P z + q^\top z$$

subject to: $Gz \leq h$
 $Az = b$



Nonconvex quadratic optimization
on a polytope. Contours represent a
saddle-shaped objective function.

“Harder” problems: Integer and Mixed Integer Programs

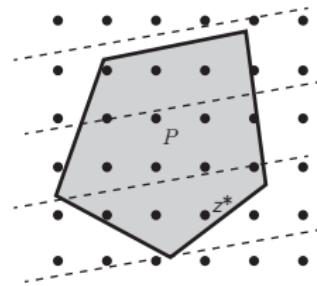
Mixed Integer Linear Program (MILP): Linear program with binary or integer constraints.

$$\min_z \quad c^\top z$$

$$\text{subject to: } Gz \leq h$$

$$Az = b$$

$$z \in \{0, 1\}^s \text{ or } z \in \mathbb{Z}^s$$



Linear optimization with integer constraints (dots).

Table of Contents

1. Introduction
2. Describing and Solving Constrained Optimization
3. Properties, Terminology and Geometry of Optimization Problems
4. Software Tools for Optimization
5. Common Types of Optimization Problems
6. Examples

More Examples

- Nonlinear Optimization [!\[\]\(63b2ec1c2337cdf23317147d254f74d7_img.jpg\) Open in Colab](#)
- Unit Commitment [!\[\]\(4bef0ca27945e5c640a1fb27438daaf3_img.jpg\) Open in Colab](#)
- Unit Replacement [!\[\]\(b164ecaaf201524150eff453b2c9f056_img.jpg\) Open in Colab](#)

Convex Optimization

Paul Goulart, Francesco Borrelli

Institut für Automatik
ETH Zürich

UC Berkeley

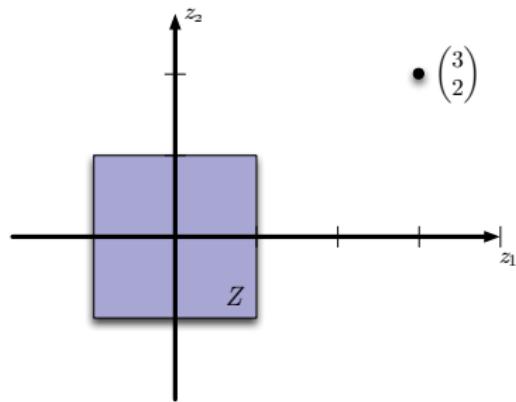
Fall Semester 2020

Table of Contents

1. Introduction
2. Convex Sets
3. Convex Functions
4. Convex Optimization Problems
5. Optimality Criteria for Unconstrained Problems
6. Optimality Criteria for Constrained Problems
7. Optimality Conditions and Duality

A Simple Example

Problem : In \mathbb{R}^2 , find the point in the unit box S closest to the point $(z_1, z_2) = (3, 2)$.



Three Major Steps

- How do I ~~transform~~ it into a formal mathematical optimization problem (next called “Mathematical Programming” problem)?
- How do I ~~solve~~ the optimization problem?
- How do I know the solution is right (next called ==~~certification~~ process)?

Starting from the Conclusions

Theorem

*For a convex optimization problem, **every** locally optimal solution is globally optimal.*

Theorem

Nice theory, analytic certificate of optimality, efficient algorithms.

Table of Contents

1. Introduction
2. Convex Sets
3. Convex Functions
4. Convex Optimization Problems
5. Optimality Criteria for Unconstrained Problems
6. Optimality Criteria for Constrained Problems
7. Optimality Conditions and Duality

Table of Contents

2. Convex Sets

2.1 Definition and Examples

2.2 Set Operations

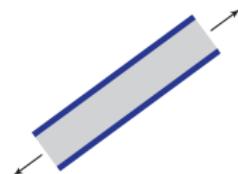
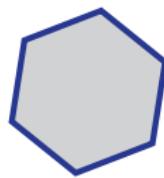
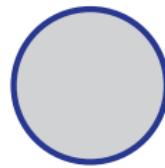
Definition (Convex Set)

A set Z is **convex** if and only if for any pair of points z and y in Z , any **convex combination** of z and y lies in Z :

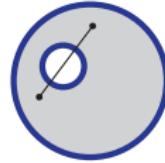
$$Z \text{ is convex} \Leftrightarrow \lambda z + (1 - \lambda)y \in Z, \forall \lambda \in [0, 1], \forall z, y \in Z$$

Interpretation: All line segments starting and ending in Z stay within Z .

Convex:



Non-convex:



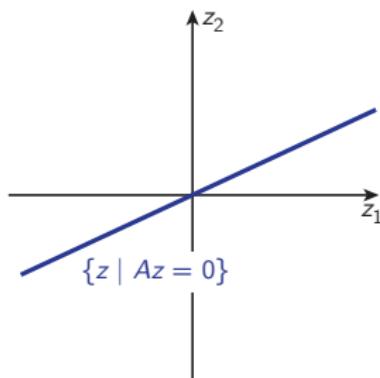
Definitions (Affine sets and Subspaces)

An **affine set** is a convex set defined by $Z = \{z \in \mathbb{R}^s \mid Az = b\}$. A **subspace** is an affine set with $b = 0$.

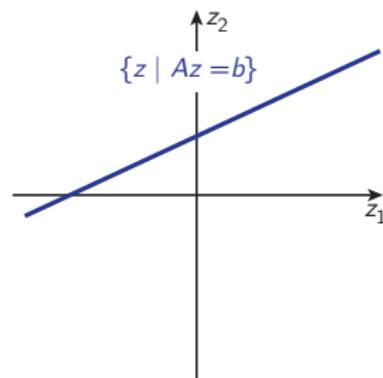
Verify convexity by definition — for all $z, y \in Z$, for all $\lambda \in [0, 1]$,

$$A(\lambda z + (1 - \lambda)y) = \lambda Az + (1 - \lambda)Ay = \lambda \cdot b + (1 - \lambda) \cdot b = b$$

This definition encompasses lines, planes and individual points.



A 1D subspace in \mathbb{R}^2



An affine space in \mathbb{R}^2

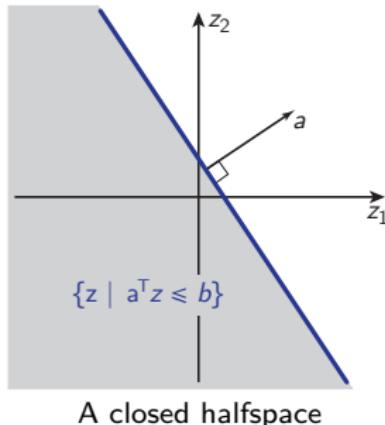
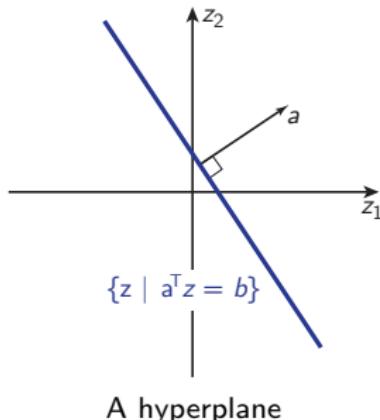
Definitions (Hyperplanes and halfspaces)

A **hyperplane** is defined by $\{z \in \mathbb{R}^s \mid a^\top z = b\}$ for $a \neq 0$, where $a \in \mathbb{R}^s$ is the normal vector to the hyperplane.

A **halfspace** is everything on one side of a hyperplane, i.e. $\{z \in \mathbb{R}^s \mid a^\top z \leq b\}$ for $a \neq 0$. It can either be **open** (strict inequality) or **closed** (non-strict inequality).

For $n = 2$, hyperplanes define lines. For $n = 3$, hyperplanes define planes. Compare to affine sets, which could define a line or a plane in \mathbb{R}^3 .

Hyperplanes and halfspaces are always convex.



Definitions (Polyhedra and polytopes)

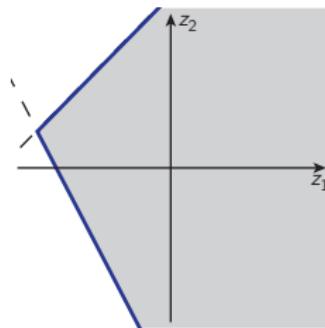
A **polyhedron** is the intersection of a finite number of closed halfspaces:

$$\begin{aligned} Z &= \{z \mid a_1^\top z \leq b_1, a_2^\top z \leq b_2, \dots, a_m^\top z \leq b_m\} \\ &= \{z \mid Az \leq b\} \end{aligned}$$

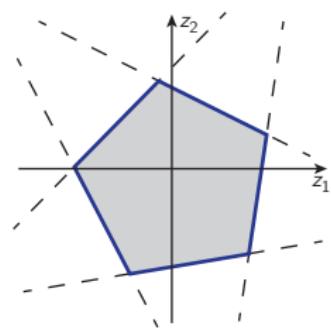
where $A := [a_1, a_2, \dots, a_m]^\top$ and $b := [b_1, b_2, \dots, b_m]^\top$.

A **polytope** is a bounded polyhedron.

Polyhedra and polytopes are always convex.



An (unbounded) polyhedron



A polytope

Definition (Vector norm)

A **norm** is any function $f : \mathbb{R}^s \rightarrow \mathbb{R}$ satisfying the following conditions:

- $f(z) \geq 0$ and $f(z) = 0 \Rightarrow z = 0$.
- $f(tx) = |t|f(z)$ for scalar t .
- $f(z + y) \leq f(z) + f(y)$, for all $z, y \in \mathbb{R}^s$.

A norm is denoted $\|z\|_{\bullet}$, where a symbol in place of the dot denotes the type of norm.
The notation $\|z\|$ refers to any arbitrary norm.

Definition (ℓ_p norm)

The ℓ_p norm on \mathbb{R}^s is denoted $\|z\|_p$, and is defined for any $p \geq 1$ by

$$\|z\|_p := \left[\sum_{i=1}^s |z_i|^p \right]^{1/p}$$

ℓ_p norms

By far the most common ℓ_p norms are:

- $p = 2$ (Euclidean norm):

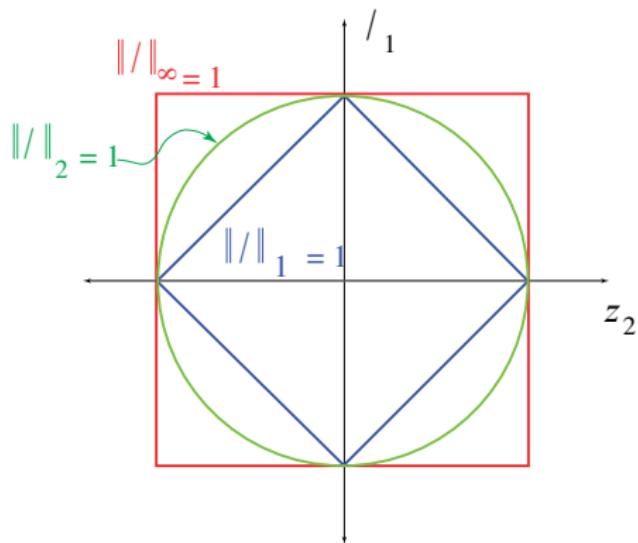
$$\|z\|_2 = \sqrt{\sum_i z_i^2}$$

- $p = 1$ (Sum of absolute values):

$$\|z\|_1 = \sum_i |z_i|$$

- $p = \infty$ (Largest absolute value):

$$\|z\|_\infty = \max_i |z_i|$$



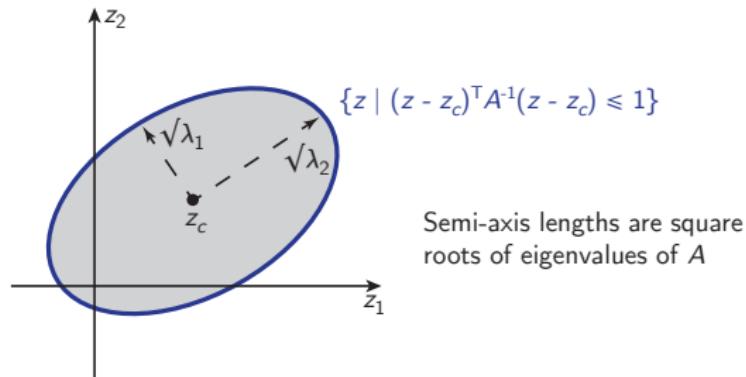
The **norm ball**, defined by $\{z \mid \|z - z_c\| \leq r\}$ where z_c is the centre of the ball and $r \geq 0$ is the radius, is always convex for any norm.

Definition (Ellipsoid)

An **ellipsoid** is a set defined as

$$\{z \mid (z - z_c)^\top A^{-1}(z - z_c) \leq 1\},$$

where z_c is the centre of the ellipsoid, and $A \succ 0$ (i.e. A is positive definite).



The **Euclidean ball** $B(z_c, r)$ is a special case of the ellipsoid, for which $A = r^2 I$, so that $B(z_c, r) := \{z \mid \|z - z_c\|_2 \leq r\}$.

Table of Contents

2. Convex Sets

2.1 Definition and Examples

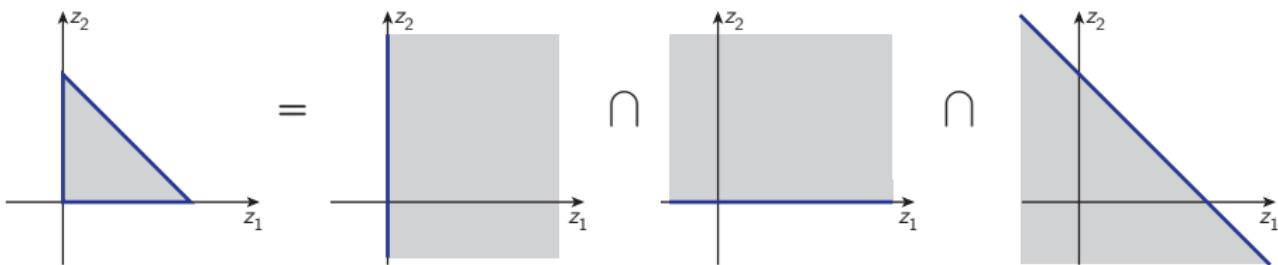
2.2 Set Operations

Intersection $Z \cap \mathcal{Y}$

Theorem

The intersection of two or more convex sets is itself convex.

Proof (for two sets): Consider any two points a and b which both lie in both of two convex sets Z and \mathcal{Y} . For any $\lambda \in [0, 1]$, $\lambda a + (1 - \lambda)b$ is in both Z and \mathcal{Y} . Therefore $\lambda a + (1 - \lambda)b \in Z \cap \mathcal{Y}, \forall \lambda \in [0, 1]$. This satisfies the definition of convexity for set $Z \cap \mathcal{Y}$.



Many sets can be written as the intersection of convex elements, and are therefore easily shown to be convex. Any convex set can be written as a (possibly infinite) intersection of halfspaces.

Convex Hull $\text{Co}(Z)$

The **convex hull** of a set Z is the set of all convex combinations of points in Z :

$$\text{Co}(Z) := \{z \mid z = \lambda a + (1 - \lambda)b, \lambda \in [0, 1], a, b \in Z\}$$

It is the smallest convex set that contains Z : for all convex sets $\mathcal{Y} \supseteq Z$, $\text{Co}(Z) \subseteq \mathcal{Y}$.



For a set $Z = \{z_1, z_2, \dots, z_q\}$ comprising q points, the convex hull can be written

$$\text{Co}(Z) = \left\{ \lambda_1 z_1 + \lambda_2 z_2 + \dots + \lambda_q z_q \mid \lambda_i \geq 0, i = 1, \dots, q, \sum_{i=1}^q \lambda_i = 1 \right\}$$

Union $Z \cup Y$

Note that the **union** of two sets is **not** convex in general, regardless of whether the original sets were convex!

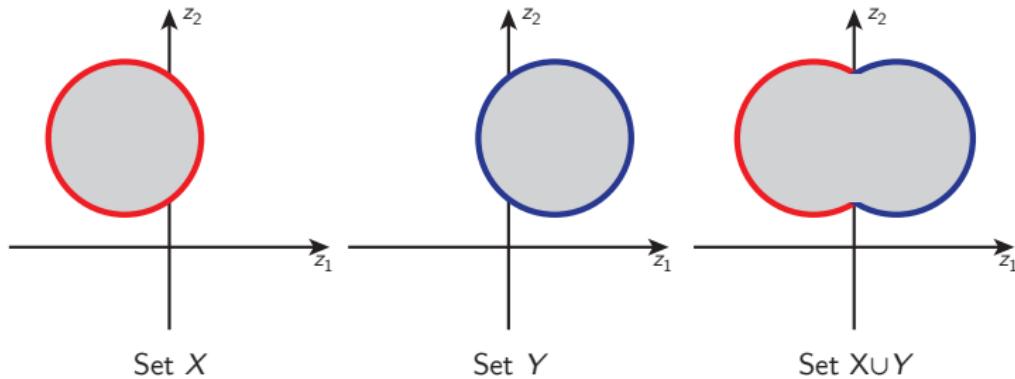


Table of Contents

1. Introduction
2. Convex Sets
- 3. Convex Functions**
4. Convex Optimization Problems
5. Optimality Criteria for Unconstrained Problems
6. Optimality Criteria for Constrained Problems
7. Optimality Conditions and Duality

Table of Contents

3. Convex Functions

3.1 Definitions

3.2 Examples

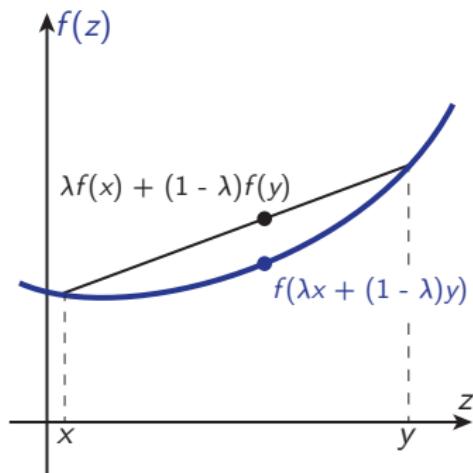
3.3 Properties

Definitions (Convex Function)

A function $f : \text{dom}(f) \rightarrow \mathbb{R}$ is **convex** iff¹ its domain $\text{dom}(f)$ is convex and

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y), \quad \forall \lambda \in (0, 1), \quad \forall x, y \in \text{dom}(f)$$

The function f is **strictly convex** if this inequality is strict.



f is **concave** iff the function $-f$ is convex.

¹"if and only if"

1st-order condition for convexity

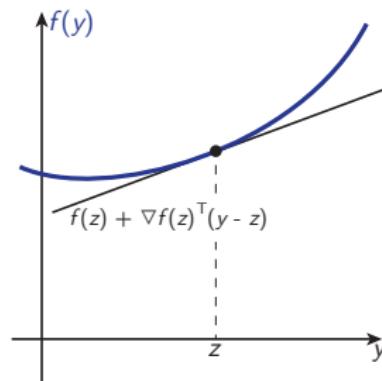
A differentiable function $f : \text{dom}(f) \rightarrow \mathbb{R}$ with a convex domain is **convex** iff

$$f(y) \geq f(z) + \nabla f(z)^T (y - z), \quad \forall z, y \in \text{dom}(f)$$

i.e. a first order approximator of f around any point z is a global underestimator of f .

The gradient $\nabla f(z)$ is given by

$$\nabla f(z) = \left[\frac{\partial f(z)}{\partial z_1}, \frac{\partial f(z)}{\partial z_2}, \dots, \frac{\partial f(z)}{\partial z_s} \right]^T$$



2nd-order condition for convexity

A twice-differentiable function $f : \text{dom}(f) \rightarrow \mathbb{R}$ is **convex** iff its domain $\text{dom}(f)$ is convex and

$$\nabla^2 f(z) \succeq 0, \quad \forall z \in \text{dom}(f),$$

where the Hessian $\nabla^2 f(z)$ is defined by

$$\nabla^2 f(z)_{ij} = \frac{\partial^2 f(z)}{\partial z_i \partial z_j}$$

If $\text{dom}(f)$ is convex and $\nabla^2 f(z) \succ 0$ for all $z \in \text{dom}(f)$, then f is **strictly convex**.

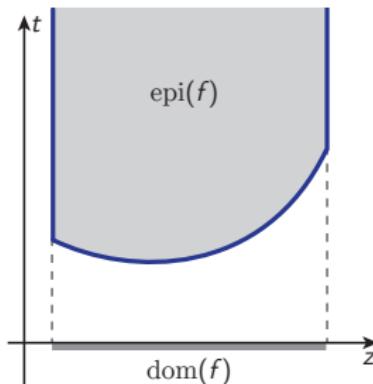
Epigraph of a Function

The **epigraph** of a function $f : \text{dom}(f) \rightarrow \mathbb{R}$ is the set

$$\text{epi}(f) = \left\{ \begin{bmatrix} z \\ t \end{bmatrix} \mid z \in \text{dom}(f), f(z) \leq t \right\} \subseteq \text{dom}(f) \times \mathbb{R}$$

It has dimension one higher than the domain of f .

A function is convex iff its epigraph is a convex set.



The epigraph of a convex function on a closed domain.

Level and sublevel sets

Definition (Level set)

The **level set** L_α of a function f for value α is the set of all $z \in \text{dom}(f)$ for which $f(z) = \alpha$:

$$L_\alpha = \{z \mid z \in \text{dom}(f), f(z) = \alpha\}$$

For $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ these are contour lines of constant “height”.

Definition (Sublevel set)

The **sublevel set** C_α of a function f for value α is defined by

$$C_\alpha = \{z \mid z \in \text{dom}(f), f(z) \leq \alpha\}$$

Function f is convex \Rightarrow sublevel sets of f are convex for all α . But not \Leftarrow !

Table of Contents

3. Convex Functions

3.1 Definitions

3.2 Examples

3.3 Properties

Examples of Convex Functions: $\mathbb{R} \rightarrow \mathbb{R}$

The following functions are **convex** (on domain \mathbb{R} unless otherwise stated):

- Affine: $ax + b$ for any $a, b \in \mathbb{R}$
- Exponential: e^{ax} for any $a \in \mathbb{R}$
- Powers: z^α on domain \mathbb{R}^+ , for $\alpha \geq 1$ or $\alpha \leq 0$
- Powers of absolute value: $|z|^p$, for $p \geq 1$

The following functions are **concave** (on domain \mathbb{R} unless otherwise stated):

- Affine: $ax + b$ for any $a, b \in \mathbb{R}$
- Powers: z^α on domain \mathbb{R}^+ , for $0 \leq \alpha \leq 1$
- Logarithm: $\log z$ on domain \mathbb{R}^+
- Entropy: $-z \log z$ on domain \mathbb{R}^+

Examples of Convex Functions: $\mathbb{R}^s \rightarrow \mathbb{R}$

Affine functions on \mathbb{R}^s are both convex and concave:

- On \mathbb{R}^s , for some $a \in \mathbb{R}^s$ and $b \in \mathbb{R}$:

$$f(z) = a^\top z + b$$

Vector Norms on \mathbb{R}^s are all convex:

- On \mathbb{R}^s , ℓ_p norms have the form, for $p \geq 1$,

$$\|z\|_p = \left(\sum_{i=1}^s |z_i|^p \right)^{1/p}, \quad \text{with } \|z\|_\infty = \max_i |z_i|$$

Examples of Convex Functions: $\mathbb{R}^{m \times n} \rightarrow \mathbb{R}$

Affine functions on $\mathbb{R}^{m \times n}$ are both convex and concave:

- On $\mathbb{R}^{m \times n}$, for some $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}$:

$$f(S) = \text{trace}(A^\top S) + b = \sum_{i=1}^m \sum_{j=1}^s A_{ij} S_{ij} + b$$

Matrix Norms on $\mathbb{R}^{m \times n}$ are all convex:

- On $\mathbb{R}^{m \times n}$ the **spectral**, or **maximum singular value** norm is

$$\|S\|_2 = \sigma_{\max}(S) = [\lambda_{\max}(S^\top S)]^{1/2}.$$

Table of Contents

3. Convex Functions

3.1 Definitions

3.2 Examples

3.3 Properties

Convexity-preserving Operations

Certain operations preserve the convexity of functions:

- Non-negative weighted sum
- Composition with affine function
- Pointwise maximum and supremum
- Partial minimization

and many other possibilities...

Convexity-preserving Operations

Theorem (Non-negative weighted sum)

If f is a function convex, then αf is convex for $\alpha \geq 0$. For several convex functions g_i , $\sum_i \alpha_i g_i$ is convex if all $\alpha_i \geq 0$.

Theorem (Composition with affine function)

If f is a convex function, then $f(Az + b)$ is convex.

Example: $\|Az - b\|$ is convex for any norm.

Theorem (Pointwise maximum)

If f_1, \dots, f_m are convex functions, then $f(z) = \max\{f_1(z), \dots, f_m(z)\}$ is convex.

Example: Piecewise linear functions $\max_{i=1,\dots,m} \{a_i^\top z + b\}$ are convex.

Convexity-preserving Operations (cont'd)

Theorem (Composition with scalar functions)

For $g : \mathbb{R}^s \rightarrow \mathbb{R}$ and $h : \mathbb{R} \rightarrow \mathbb{R}$, $f(z) = h(g(z))$ is convex if:

- g is convex, h is convex, h is non-decreasing
- g is concave, h is convex, h is non-increasing

Examples

- $\exp g(z)$ for convex g
- $1/g(z)$ for concave positive g

Convexity-preserving Operations (cont'd)

Theorem (Composition with vector functions)

For $g : \mathbb{R}^s \rightarrow \mathbb{R}^k$ and $h : \mathbb{R}^k \rightarrow \mathbb{R}$, $f(z) = h(g(z)) = h(g_1(z), g_2(z), \dots, g_k(z))$ is convex if:

- Each g_i is convex, h is convex, h is non-decreasing in each argument
- Each g_i is concave, h is convex, h is non-increasing in each argument

Examples

- $\log \sum_{i=1}^k \exp g_i(z)$ is convex if all g_i are positive
- $\sum_{i=1}^k \log g_i(z)$ is concave for concave positive g_i

Table of Contents

1. Introduction
2. Convex Sets
3. Convex Functions
4. Convex Optimization Problems
5. Optimality Criteria for Unconstrained Problems
6. Optimality Criteria for Constrained Problems
7. Optimality Conditions and Duality

Table of Contents

4. Convex Optimization Problems

- 4.1 Standard Convex Optimization Problem
- 4.2 Linear Programs
- 4.3 Quadratic Programs

Convex Optimization Problem

An optimization problem:

$$\min_{z \in Z} f(z)$$

subject to: $g_i(z) \leq 0 \quad i = 1, \dots, m$
 $h_j(z) = 0 \quad j = 1, \dots, p$

is convex if:

- The objective function f is a convex function on its domain Z .
- The feasible set S is a convex set.

Standard Form Convex Optimization Problem

A standard form **convex** optimization problem:

$$\min_{z \in Z} f(z)$$

$$\begin{aligned} \text{subject to: } & g_i(z) \leq 0 \quad i = 1, \dots, m \\ & a_j^\top z = b_j \quad j = 1, \dots, p \end{aligned}$$

This problem is convex if:

- The domain Z is a convex set.
- The objective function f is a convex function.
- The inequality constraint functions g_i are all convex.

Standard Form Convex Optimization Problem

The affine constraints are typically gathered into matrix form:

$$\begin{aligned} & \min_{z \in Z} f(z) \\ \text{subject to: } & g_i(z) \leq 0 \quad i = 1, \dots, m \\ & Az = b \quad A \in \mathbb{R}^{p \times m} \end{aligned}$$

Crucial Fact!

Theorem

*For a convex optimization problem, **every** locally optimal solution is globally optimal.*

NB: Writing or rewriting an optimization problem in convex form can be tricky, and is not always possible. It is always worth trying though.

Standard Form Convex Optimization Problem

The affine constraints are typically gathered into matrix form:

$$\begin{aligned} & \min_{z \in Z} f(z) \\ \text{subject to: } & g_i(z) \leq 0 \quad i = 1, \dots, m \\ & Az = b \quad A \in \mathbb{R}^{p \times m} \end{aligned}$$

Crucial Fact!(2)

Theorem

Nice theory, analytic certificate of optimality, efficient algorithms.

Local and Global Optimality for Convex Problems

Theorem

For a convex optimization problem, **every** locally optimal solution is globally optimal.

Proof:

- Assume that z is locally optimal, but not globally optimal.
- Therefore there is some other point y such that $f(y) < f(z)$.
- z locally optimal implies that there is some $R > 0$ such that

$$\|z - z\|_2 \leq R \Rightarrow f(z) \leq f(z)$$

- The problem can't be convex.

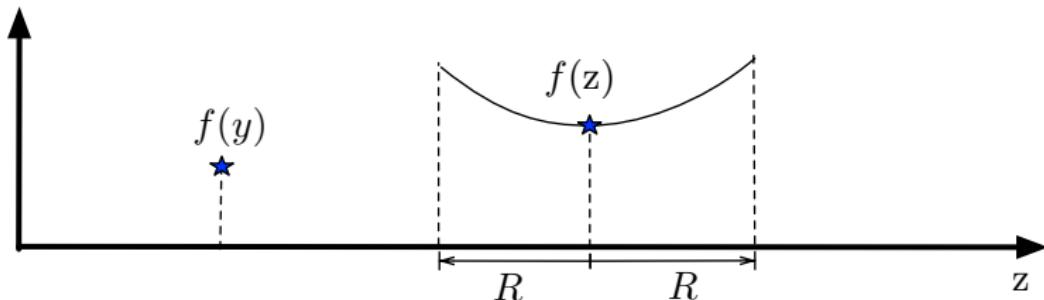


Table of Contents

4. Convex Optimization Problems

4.1 Standard Convex Optimization Problem

4.2 Linear Programs

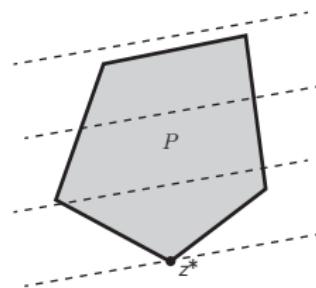
4.3 Quadratic Programs

General Linear Program (LP)

Affine cost and constraint functions:

$$\min_z \quad c^\top z + d$$

$$\begin{aligned} \text{subject to: } & Gz \leq h \\ & Az = b \end{aligned}$$



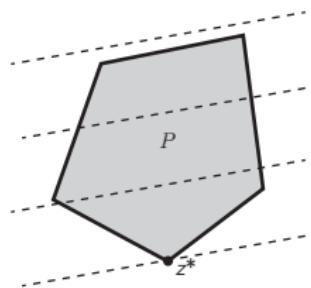
Linear optimization on a polytope.

- Feasible set is a polyhedron.
- Constant component d can be left out – it has no effect on the optimal solution.

General Linear Program (LP)

An alternative format:

$$\begin{array}{ll} \min_z & c^T z \\ \text{subject to: } & Az = b \\ & z \geq 0 \end{array}$$



Linear optimization on a polytope.

- All components of z are non-negative.
- Can easily convert previous format to this (using extra variables).

Many problems can be rewritten (with some effort!) into LPs.

Huge variety of solution methods and software are available.

Example Linear Programs

Cheapest cat-food problem:

- Choose quantities z_1, z_2, \dots, z_s of n different ingredients with unit cost c_j .
- Each ingredient j has nutritional content a_{ij} for nutrient i .
- Require for each nutrient i minimum level b_i .

In linear program form:

$$\begin{aligned} & \min_z \quad c^\top z \\ & \text{subject to: } Az \geq b \\ & \quad z \geq 0 \end{aligned}$$

This is an example of a resource allocation problem.

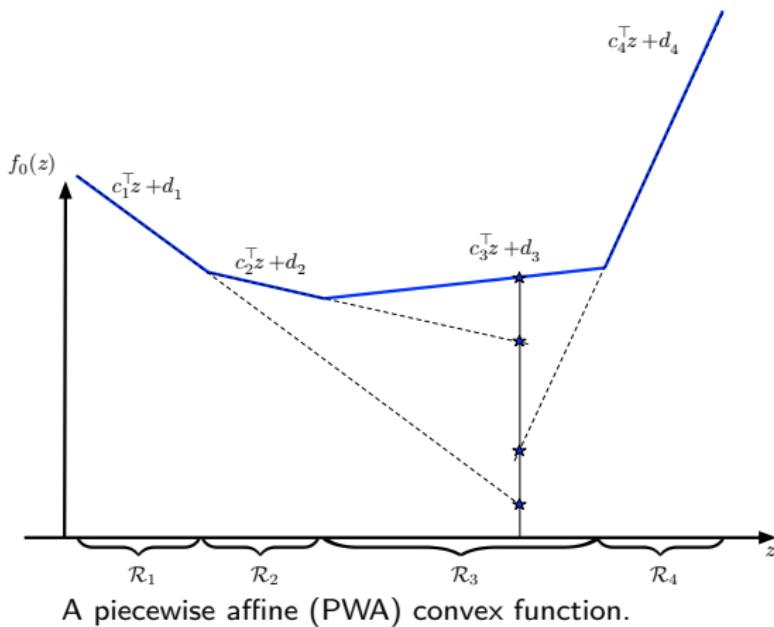
Kantorovich and Koopmans won the Nobel Prize in Economics in 1975 for their work on this problem (and its non-cat-food variants).

Example : Piecewise Affine Minimization

$$\min_z \quad \left[\max_{i=1,\dots,m} \{c_i^\top z + d_i\} \right]$$

subject to: $Gz \leq h$

- The function is affine on each region \mathcal{R}_i .
- Any convex and piecewise affine function can be written this way.
- Can be reformulated as an LP.



Example Linear Programs

Piecewise affine minimization:

$$\min_z \quad \left[\max_{i=1,\dots,m} \{c_i^\top z + d_i\} \right]$$

subject to: $Gz \leq h$

is equivalent to an LP:

$$\begin{aligned} & \min_{z,t} \quad t \\ & \text{subject to: } c_i^\top z + d_i \leq t \quad i = 1, \dots, m \\ & \quad Gz \leq h \end{aligned}$$

NB: trick was to add variables and write the problem in epigraph form.

ℓ_∞ minimization

Constrained ℓ_∞ (Chebyshev) minimization:

$$\begin{aligned} \min_{z \in \mathbb{R}^s} \quad & \|z\|_\infty \\ \text{subject to: } \quad & Fz \leq g \end{aligned}$$

Write this is a max of linear functions.

Equivalent to:

$$\begin{aligned} \min_{z \in \mathbb{R}^s} \quad & [\max \{z_1, \dots, z_s, -z_1, \dots, -z_s\}] \\ \text{subject to: } \quad & Fz \leq g \end{aligned}$$

ℓ_∞ minimization (cont'd)

Equivalent to:

$$\begin{array}{ll}
 \min_{z,t} & t \\
 \text{subject to:} & z_i \leq t \quad i = 1, \dots, n \\
 & -z_i \leq t \quad i = 1, \dots, n \\
 & Fz \leq g
 \end{array}
 \Rightarrow
 \begin{array}{ll}
 \min_{z,t} & t \\
 \text{subject to:} & -\mathbf{1}t \leq z \leq \mathbf{1}t \\
 & Fz \leq g
 \end{array}$$

- The notation ' $\mathbf{1}$ ' indicates a vector of ones.
- The constraint $-\mathbf{1}t \leq z \leq \mathbf{1}t$ bounds the absolute value of every element of z with a common scalar variable t .

ℓ_1 minimization

Constrained ℓ_1 minimization:

$$\min_{z \in \mathbb{R}^s} \|Az - b\|_1$$

subject to: $Fz \leq g$

Write this is a max of linear functions. Assume $A \in \mathbb{R}^{m \times n}$.

Equivalent to:

$$\min_{z \in \mathbb{R}^s} \left[\sum_{i=1}^m \max \{(Az - b)_i, -(Az - b)_i\} \right]$$

subject to: $Fz \leq g$

ℓ_1 minimization (cont'd)

Equivalent to:

$$\begin{array}{ll} \min_{z \in \mathbb{R}^s, t \in \mathbb{R}^m} & t_1 + \dots + t_m \\ \text{subject to:} & \begin{aligned} (Az - b)_i \leq t_i & i = 1, \dots, m \\ -(Az - b)_i \leq t_i & i = 1, \dots, m \\ Fz \leq g & \end{aligned} \end{array} \Rightarrow \begin{array}{ll} \min_{z \in \mathbb{R}^s, t \in \mathbb{R}^m} & \mathbf{1}^\top t \\ \text{subject to:} & \begin{aligned} -t \leq (Az - b) \leq t \\ Fz \leq g \end{aligned} \end{array}$$

- The notation ' $\mathbf{1}$ ' indicates a vector of ones.
- The constraint $-t \leq (Az - b) \leq t$ bounds the absolute value of each component of $(Az - b)$ with a component of the vector variable t .

Table of Contents

4. Convex Optimization Problems

- 4.1 Standard Convex Optimization Problem
- 4.2 Linear Programs
- 4.3 Quadratic Programs

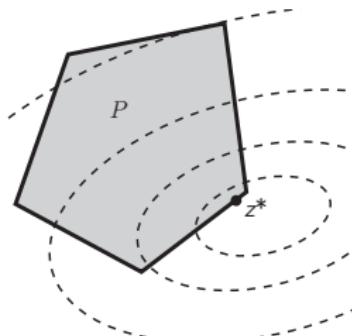
General Quadratic Program

Quadratic cost function with $P \in \mathbb{S}_+^s$, affine constraint functions. Feasible set is a polyhedron:

$$\min_z \quad \frac{1}{2} z^\top P z + q^\top z + r$$

$$\begin{aligned} \text{subject to: } & Gz \leq h \\ & Az = b \end{aligned}$$

- Constant component r can be left out, since it has no effect on the optimal solution.
- Maximization problems with a concave objective function ($-P \in \mathbb{S}_+^s$) are also quadratic programs.



Optimization of a quadratic objective function over a polytopic set P .
The level sets of the objective are shown as dotted lines.

Example Quadratic Programs - Least squares

Least squares:

$$\min_z \|Az - b\|_2^2$$

- Analytical solution $A^\dagger b$ (A^\dagger is the pseudo-inverse).
- Extra linear constraints $l \leq z \leq u$ can be added, although the QP would no longer have an analytical solution.

Example Quadratic Programs

Quadratic program with random cost:

$$\begin{aligned} \min_z \quad & \mathbb{E}[c^T z] + \gamma \text{var}(c^T z) = \bar{c}^T z + \gamma z^T \Sigma z \\ \text{subject to: } & Gz \leq h \\ & Az = b \end{aligned}$$

- Random cost function vector c with mean \bar{c} and covariance Σ , we wish to penalize expected cost plus a “risk premium” γ on the variance.
- Hence $c^T z$ is a random variable with mean $\bar{c}^T z$ and variance $z^T \Sigma z$.
- Large γ means large risk aversion — we prefer a small variance to the lowest expected cost.

Table of Contents

1. Introduction
2. Convex Sets
3. Convex Functions
4. Convex Optimization Problems
5. Optimality Criteria for Unconstrained Problems
6. Optimality Criteria for Constrained Problems
7. Optimality Conditions and Duality

$$f(z) : \mathbb{R}^s \rightarrow \mathbb{R}$$

$$H(z) = \begin{bmatrix} s \\ s \end{bmatrix}$$

$$H_{ij}(z) = \frac{\partial^2 f(z)}{\partial z_i \partial z_j} \Big|_{z=z^*}$$

Optimality Criterion for Differentiable f

Theorem (Necessary condition*)

$f : \mathbb{R}^s \rightarrow \mathbb{R}$ is differentiable at z^* . If z^* is a local minimizer, then $\nabla f(z^*) = 0$.

Theorem (Sufficient condition*)

Suppose that $f : \mathbb{R}^s \rightarrow \mathbb{R}$ is twice differentiable at z^* . If $\nabla f(z^*) = 0$ and the Hessian of $f(z)$ at z^* is positive definite, then z^* is a local minimizer.

Theorem (Necessary and sufficient condition*)

Suppose that $f : \mathbb{R}^s \rightarrow \mathbb{R} = z^\top H z + c^\top z + k$. If H is positive definite, then z^* is the global minimizer if and only if $\nabla f(z^*) = 0$.

*Proofs available in Chapter 4 of M.S. Bazaraa, H.D. Sherali, and C.M. Shetty.

Nonlinear Programming - Theory and Algorithms. John Wiley & Sons, Inc., New York, second edition, 1993.

A Well Known Optimization Problem - Least Squares

Least squares:

$$\min_{z \in \mathbb{R}^s} \|Az - b\|_2^2$$

- Analytical solution $z^* = A^\dagger b$, if A full column rank
- $A^\dagger = (A^T A)^{-1} A^T$
- A^\dagger is often called the pseudo-inverse
- In Matlab $z^* = A \backslash b$, in Python “numpy.linalg.pinv(A)”
- Proof:

$$\min_z \|Az - b\|_2^2 = \min_z z^T (A^T A) z - z^T (2A^T b) + b^T b$$

If A full column rank then $A^T A$ is positive definite, from previous Theorem at z^*

$$\nabla f(z^*) = 0 \Rightarrow (2A^T A)z^* = (2A^T b)$$

Table of Contents

5. Optimality Criteria for Unconstrained Problems

5.1 Gradient Methods for Unconstrained Problems

Decent Direction for Differentiable f

Theorem (Descent Direction)

$f : \mathbb{R}^s \rightarrow \mathbb{R}$ differentiable at \bar{z} . If there exists a vector \mathbf{d} such that $\nabla f(\bar{z})' \mathbf{d} < 0$, then there exists a $\delta > 0$ such that $f(\bar{z} + \lambda \mathbf{d}) < f(\bar{z})$ for all $\lambda \in (0, \delta)$.

always decrease

- The vector \mathbf{d} in the theorem above is called **descent direction**.
- The direction of **steepest descent** \mathbf{d}_s at \bar{z} is defined as the normalized direction where $\nabla f(\bar{z})' \mathbf{d}_s < 0$ is minimized.
- The direction \mathbf{d}_s of steepest descent is $\mathbf{d}_s = -\frac{\nabla f(\bar{z})}{\|\nabla f(\bar{z})\|}$.

$$\boxed{-\frac{\nabla f(\bar{z})}{\|\nabla f(\bar{z})\|}}$$

① Definition of descent direction
 ② STEEPEST DESCENT \mathbf{d}_s is
 ③ NOTE LOCAL DEFINITION

Unconstrained Optimization Using Gradient Information

[Cauchy 1847]

Goal: Solve the unconstrained (i.e. $S = \mathbb{R}^n$) problem

$$\text{minimize } f(z)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex and continuously differentiable.

Idea: Gradient ∇f gives direction of steepest local ascent

⇒ Make steps of size h^k into anti-gradient direction $-\nabla f$:

$$z^{k+1} = z^k - h^k \nabla f(z^k) \quad (1)$$

Question: How to choose the step sizes h^k ? Not addressed in this class. See Basic and advanced optimization classes.

Table of Contents

1. Introduction
2. Convex Sets
3. Convex Functions
4. Convex Optimization Problems

5. Optimality Criteria for Unconstrained Problems

6. Optimality Criteria for Constrained Problems proof that the Z is the optimizer

-
7. Optimality Conditions and Duality

$$z_1^4 + z_2^4 + z_1^2 + 3z_2^2 + 4 = f(z_1, z_2)$$

$$\min_z f(z) = ?$$

$$z^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{ Proof?}$$

$$\textcircled{1} H(z^*) \geq 0, \nabla f(z^*) = 0$$

Optimality Conditions

Consider the problem

$$\begin{aligned} \min_z \quad & f(z) \\ \text{subject to: } \quad & g_i(z) \leq 0 \quad \text{for } i = 1, \dots, m \\ & h_j(z) = 0 \quad \text{for } j = 1, \dots, p \end{aligned}$$

- In general, an analytical solution does not exist.
- Solutions are usually computed by recursive algorithms which start from an initial guess z_0 and at step k generate a point z_k such that $\{f(z_k)\}_{k=0,1,2,\dots}$ converges to f^* .
- These algorithms recursively use and/or solve analytical **conditions for optimality**

KKT optimality conditions

$z^*, (u^*, v^*)$ of an optimization problem, with differentiable cost and constraints and zero duality gap, have to satisfy the following conditions:

$$0 = \nabla f(z^*) + \sum_{i=1}^m u_i^* \nabla g_i(z^*) + \sum_{j=1}^p v_j^* \nabla h_j(z^*), \quad (2a)$$

$$0 = u_i^* g_i(z^*), \quad i = 1, \dots, m \quad (2b)$$

$$0 \leq u_i^*, \quad i = 1, \dots, m \quad \begin{matrix} 1. \text{ ineq and eq} \\ (2c) \end{matrix}$$

$$0 \geq g_i(z^*), \quad i = 1, \dots, m \quad \begin{matrix} 2. \text{ involve additional var -> dual} \\ (2d) \end{matrix}$$

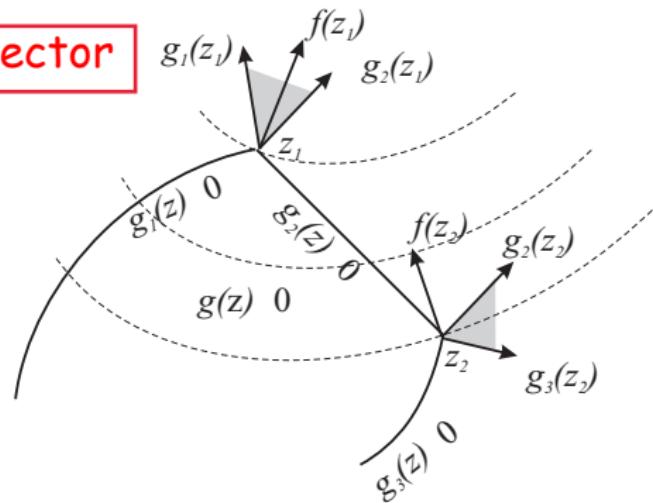
$$0 = h_j(z^*) \quad j = 1, \dots, p \quad \begin{matrix} 3. \text{ u var should be positive} \\ (2e) \end{matrix}$$

4. inactive $> u=0$; or active

- Conditions (7a)-(7e) are called the Karush-Kuhn-Tucker (KKT) conditions.
- KKT are necessary and sufficient conditions for feasible LPs and QPs.
- (u^*, v^*) called “dual variables” or “Lagrange multipliers”

KKT geometric interpretation

view in the vector



Rewrite the (7a), as

$$-\nabla f(z) = \sum_{i \in I} u_i \nabla g_i(z), \quad u_i \geq 0,$$

i.e., the direction of cost steepest descent belongs to the convex cone spanned by ∇g_i 's,

Table of Contents

1. Introduction
2. Convex Sets
3. Convex Functions
4. Convex Optimization Problems
5. Optimality Criteria for Unconstrained Problems
6. Optimality Criteria for Constrained Problems
7. Optimality Conditions and Duality **advanced topic**

Duality Theory. The Lagrange Function

Consider the **primal** optimization problem

$$\begin{aligned} \min_z \quad & f(z) \\ \text{subj. to} \quad & g_i(z) \leq 0 \quad \text{for } i = 1, \dots, m \\ & h_i(z) = 0 \quad \text{for } i = 1, \dots, p \\ & z \in Z = \mathbf{dom}(f) \cap \mathbf{dom}(g_i) \cap \mathbf{dom}(h_i) \end{aligned}$$

Any feasible point: upper bound of the optimal value

Lagrange dual problem: lower bound on optimal value

- Construct **Lagrange function**

$$\begin{aligned} L(z, u, v) = \quad & f(z) + u_1 g_1(z) + \dots + u_m g_m(z) + \\ & + v_1 h_1(z) + \dots + v_p h_p(z) \end{aligned}$$

- More compact

$$L(z, u, v) \triangleq f(z) + u' g(z) + v' h(z)$$

- u_i and v_i called **Lagrange multipliers** or dual variables
- objective is augmented with weighted sum of constraint functions

Duality Theory. The Lagrange Function

- Consider **Lagrange function**

$$L(z, u, v) \triangleq f(z) + u'g(z) + v'h(z)$$

Let $z \in S$ be feasible. For arbitrary vectors $u \geq 0$ and v trivially have

$$L(z, u, v) \leq f(z)$$

- After minimization we infer

$$\min_{z \in Z} L(z, u, v) \leq \min_{z \in Z, g(z) \leq 0, h(z)=0} f(z) = p^*$$

Duality Theory

Let

$$d(u, v) \triangleq \min_{z \in Z} L(z, u, v) \in [-\infty, +\infty] \quad (3)$$

Remarks

- Computing $d(u, v)$ is called **Lagrangian dual subproblem**.
- The (3) (Lagrangian dual subproblem) is an **unconstrained optimization problem**.
- $d(u, v)$ always **concave** (Requires proof)
- The point (u, v) is dual feasible if $u \geq 0$ and $(u, v) \in \text{dom}(d)$
- $d(u, v)$ is always a lowerbound for p^* for any feasible (u, v)

Lagrangian Dual Problem

Since $u \geq 0$ and v are arbitrary, any feasible choice gives a lowerbound. Best lower bound:

$$\max_{(u,v), u \geq 0} d(u, v) \quad (4)$$

Remarks

- Since $d(u, v)$ always **concave** \Rightarrow the dual problem (4) is convex, much easier to solve than the primal (non convex in general)
- The point (u, v) is dual feasible if $u \geq 0$ and $(u, v) \in \text{dom}(d)$
- **Weak duality** always holds:

$$\max_{(u,v), u \geq 0} d(u, v) \leq \min_{z \in Z, g(z) \leq 0, h(z)=0} f(z)$$

Duality Theory. Duality Gap and Certificate of Optimality

Let:

$$d^* = \max_{(u,v), u \geq 0} d(u, v)$$

$$f^* = \min_{z \in Z, g(z) \leq 0, h(z)=0} f(z)$$

then

- we always have $d^* \leq f^*$
- $d^* - f^*$ is called **optimal duality gap**
- **Strong duality** if $d^* = f^*$
- In case of strong duality u^* and v^* serve as **certificate of optimality**

Strong Duality

- It is **sometimes** true that $d^* = p^*$.
- Strong duality usually holds for convex problems. In general need **Constraint qualifications**. Conditions on the constraint functions implying strong duality for convex problems. Satisfied for feasible LPs and QPs.
- Strong duality usually does not hold for non-convex problems.

Duality Theory. Duality Gap and Certificate of Optimality

Let:

$$d^* = \max_{(u,v), u \geq 0} d(u, v)$$

$$f^* = \min_{z \in Z, g(z) \leq 0, h(z)=0} f(z)$$

then

- we always have $d^* \leq f^*$
- $d^* - f^*$ is called **optimal duality gap**
- **Strong duality** if $d^* = f^*$
- In case of strong duality u^* and v^* serve as **certificate of optimality**

Table of Contents

7. Optimality Conditions and Duality

7.1 Complementarity slackness

7.2 KKT conditions

Complementary slackness

Suppose that z^*, u^*, v^* are primal, dual feasible with zero duality gap (hence, they are primal, dual optimal)

$$\begin{aligned} f(z^*) &= d(u^*, v^*) \\ &= \inf_z (f(z) + u^{*\prime} g(z) + v^{*\prime} h(z)) \\ &\leq f(z^*) + u^{*\prime} g(z^*) + v^{*\prime} h(z^*) \end{aligned}$$

hence we have $\sum_{i=1}^m u_i^* g_i(z^*) = 0$ and so

$$u_i^* g_i(z^*) = 0, \quad i = 1, \dots, m$$

- called **complementary slackness** condition
- i -th constraint inactive at optimum $\implies u_i = 0$
- $u_i^* > 0$ at optimum $\implies i$ -th constraint active at optimum

Table of Contents

7. Optimality Conditions and Duality

7.1 Complementarity slackness

7.2 KKT conditions

KKT optimality conditions

Suppose

- f, g_i, h_i are differentiable
- z^*, u^*, v^* are (primal, dual) optimal, with zero duality gap

by complementary slackness we have

$$f(z^*) = f(z^*) + \sum_i u_i^* g_i(z^*) + \sum_j v_j^* h_j(z^*) \quad (5)$$

and by definition of dual function we have

$$\begin{aligned} f(z^*) + \sum_i u_i^* g_i(z^*) + \sum_j v_j^* h_j(z^*) &= \\ \min_z \left(f(z) + \sum_i u_i^* g_i(z) + \sum_j v_j^* h_j(z) \right) & \end{aligned} \quad (6)$$

i.e., z^* minimizes $L(z, u^*, v^*)$ therefore

$$\nabla f(z^*) + \sum_i u_i^* \nabla g_i(z^*) + \sum_j v_j^* \nabla h_j(z^*) = 0$$

KKT optimality conditions

$z^*, (u^*, v^*)$ of an optimization problem, with differentiable cost and constraints and zero duality gap, have to satisfy the following conditions:

$$0 = \nabla f(z^*) + \sum_{i=1}^m u_i^* \nabla g_i(z^*) + \sum_{j=1}^p v_j^* \nabla h_j(z^*), \quad (7a)$$

$$0 = u_i^* g_i(z^*), \quad i = 1, \dots, m \quad (7b)$$

$$0 \leq u_i^*, \quad i = 1, \dots, m \quad (7c)$$

$$0 \geq g_i(z^*), \quad i = 1, \dots, m \quad (7d)$$

$$0 = h_j(z^*) \quad j = 1, \dots, p \quad (7e)$$

Conditions (7a)-(7e) are called the Karush-Kuhn-Tucker (KKT) conditions.

KKT optimality conditions

Consider a NLP in standard form.

Theorem (Necessary and sufficient condition)

Suppose that the NLP is convex and that cost and constraints f , g_i and h_i are differentiable at a feasible z^ . If the NLP satisfies constraint qualification conditions then z^* is optimal if and only if there are (u^*, v^*) that, together with z^* , satisfy the KKT conditions.*

Convex Optimization

Paul Goulart, Francesco Borrelli

Institut für Automatik
ETH Zürich

UC Berkeley

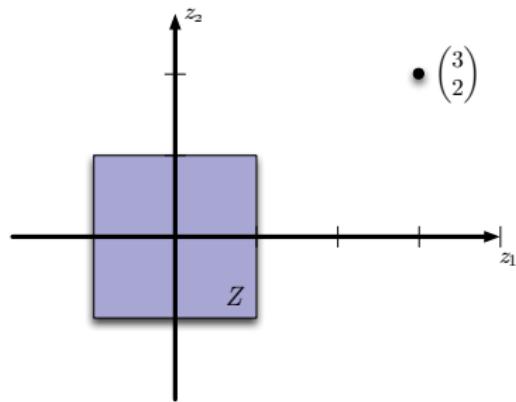
Fall Semester 2020

Table of Contents

1. Introduction
2. Convex Sets
3. Convex Functions
4. Convex Optimization Problems
5. Optimality Criteria
6. Optimality Conditions and Duality

A Simple Example

Problem : In \mathbb{R}^2 , find the point in the unit box S closest to the point $(z_1, z_2) = (3, 2)$.



Three Major Steps

- How do I ~~transform~~ it into a formal mathematical optimization problem (next called “Mathematical Programming” problem)?
- How do I ~~solve~~ it the optimization problem?
- How do I know the solution is right (next called ==~~certification~~ process)?

Starting from the Conclusions

Theorem

*For a convex optimization problem, **every** locally optimal solution is globally optimal.*

Theorem

Nice theory, analytic certificate of optimality, efficient algorithms.

Table of Contents

1. Introduction
2. Convex Sets
3. Convex Functions
4. Convex Optimization Problems
5. Optimality Criteria
6. Optimality Conditions and Duality

Table of Contents

2. Convex Sets

2.1 Definition and Examples

2.2 Set Operations

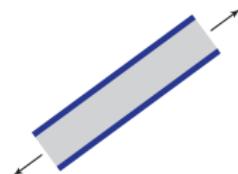
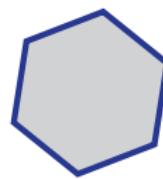
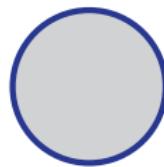
Definition (Convex Set)

A set Z is **convex** if and only if for any pair of points z and y in Z , any **convex combination** of z and y lies in Z :

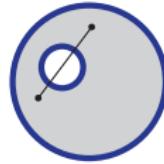
$$Z \text{ is convex} \Leftrightarrow \underline{\lambda z + (1 - \lambda)y \in Z, \forall \lambda \in [0, 1], \forall z, y \in Z}$$

Interpretation: All line segments starting and ending in Z stay within Z .

Convex:



Non-convex:



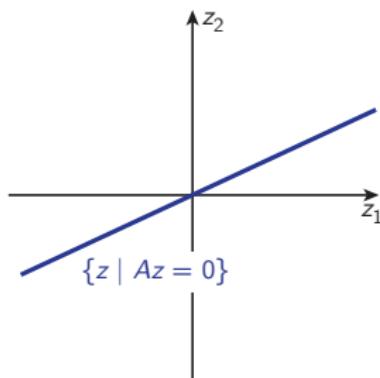
Definitions (Affine sets and Subspaces)

An **affine set** is a convex set defined by $Z = \{z \in \mathbb{R}^s \mid Az = b\}$. A **subspace** is an affine set with $b = 0$.

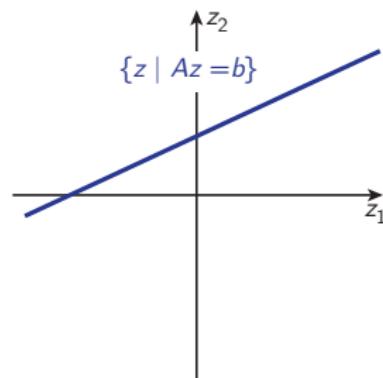
Verify convexity by definition — for all $z, y \in Z$, for all $\lambda \in [0, 1]$,

$$\rightarrow A(\lambda z + (1 - \lambda)y) = \lambda Az + (1 - \lambda)Ay = \lambda \cdot b + (1 - \lambda) \cdot b = b$$

This definition encompasses lines, planes and individual points.



A 1D subspace in \mathbb{R}^2



An affine space in \mathbb{R}^2

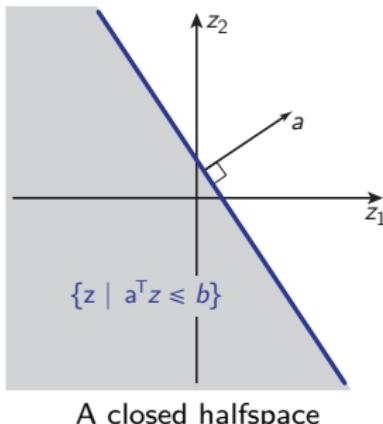
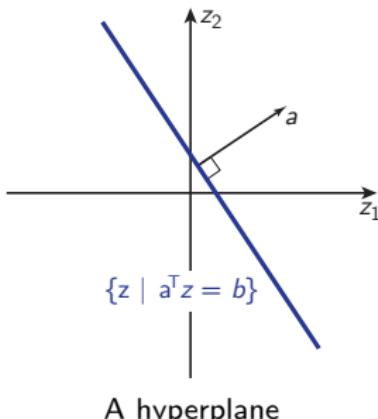
Definitions (Hyperplanes and halfspaces)

A **hyperplane** is defined by $\{z \in \mathbb{R}^s \mid a^\top z = b\}$ for $a \neq 0$, where $a \in \mathbb{R}^s$ is the normal vector to the hyperplane.

A **halfspace** is everything on one side of a hyperplane, i.e. $\{z \in \mathbb{R}^s \mid a^\top z \leq b\}$ for $a \neq 0$. It can either be **open** (strict inequality) or **closed** (non-strict inequality).

For $n = 2$, hyperplanes define lines. For $n = 3$, hyperplanes define planes. Compare to affine sets, which could define a line or a plane in \mathbb{R}^3 .

Hyperplanes and halfspaces are always convex.



Definitions (Polyhedra and polytopes)

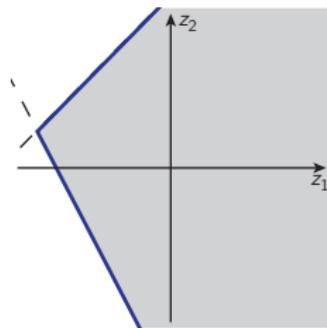
A **polyhedron** is the intersection of a finite number of closed halfspaces:

$$\begin{aligned} Z &= \{z \mid a_1^\top z \leq b_1, a_2^\top z \leq b_2, \dots, a_m^\top z \leq b_m\} \\ &= \{z \mid Az \leq b\} \end{aligned}$$

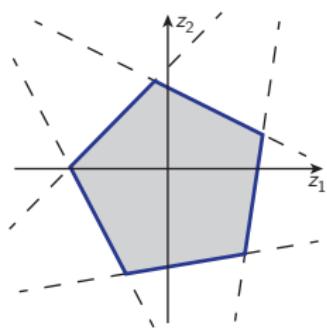
where $A := [a_1, a_2, \dots, a_m]^\top$ and $b := [b_1, b_2, \dots, b_m]^\top$.

A **polytope** is a bounded polyhedron.

Polyhedra and polytopes are always convex.



An (unbounded) polyhedron



A polytope

Definition (Vector norm)

A **norm** is any function $f : \mathbb{R}^s \rightarrow \mathbb{R}$ satisfying the following conditions:

- $f(z) \geq 0$ and $f(z) = 0 \Rightarrow z = 0$.
- $f(tx) = |t|f(z)$ for scalar t .
- $f(z + y) \leq f(z) + f(y)$, for all $z, y \in \mathbb{R}^s$.

A norm is denoted $\|z\|_{\bullet}$, where a symbol in place of the dot denotes the type of norm.
The notation $\|z\|$ refers to any arbitrary norm.

Definition (ℓ_p norm)

The ℓ_p norm on \mathbb{R}^s is denoted $\|z\|_p$, and is defined for any $p \geq 1$ by

$$\|z\|_p := \left[\sum_{i=1}^s |z_i|^p \right]^{1/p}$$

ℓ_p norms

By far the most common ℓ_p norms are:

- $p = 2$ (Euclidean norm):

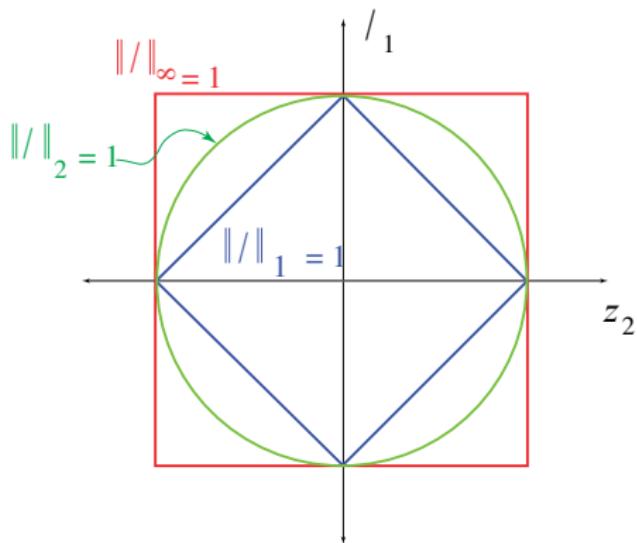
$$\|z\|_2 = \sqrt{\sum_i z_i^2}$$

- $p = 1$ (Sum of absolute values):

$$\|z\|_1 = \sum_i |z_i|$$

- $p = \infty$ (Largest absolute value):

$$\|z\|_\infty = \max_i |z_i|$$



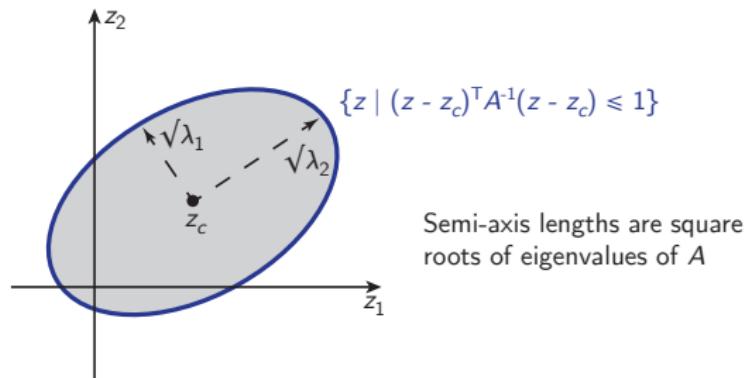
The **norm ball**, defined by $\{z \mid \|z - z_c\| \leq r\}$ where z_c is the centre of the ball and $r \geq 0$ is the radius, is always convex for any norm.

Definition (Ellipsoid)

An **ellipsoid** is a set defined as

$$\{z \mid (z - z_c)^\top A^{-1}(z - z_c) \leq 1\},$$

where z_c is the centre of the ellipsoid, and $A \succ 0$ (i.e. A is positive definite).



The **Euclidean ball** $B(z_c, r)$ is a special case of the ellipsoid, for which $A = r^2 I$, so that $B(z_c, r) := \{z \mid \|z - z_c\|_2 \leq r\}$.

Table of Contents

2. Convex Sets

2.1 Definition and Examples

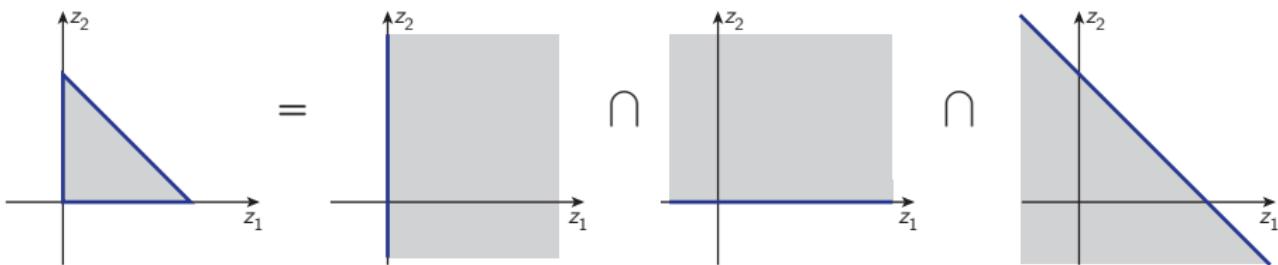
2.2 Set Operations

Intersection $Z \cap \mathcal{Y}$

Theorem

The intersection of two or more convex sets is itself convex.

Proof (for two sets): Consider any two points a and b which both lie in both of two convex sets Z and \mathcal{Y} . For any $\lambda \in [0, 1]$, $\lambda a + (1 - \lambda)b$ is in both Z and \mathcal{Y} . Therefore $\lambda a + (1 - \lambda)b \in Z \cap \mathcal{Y}, \forall \lambda \in [0, 1]$. This satisfies the definition of convexity for set $Z \cap \mathcal{Y}$.



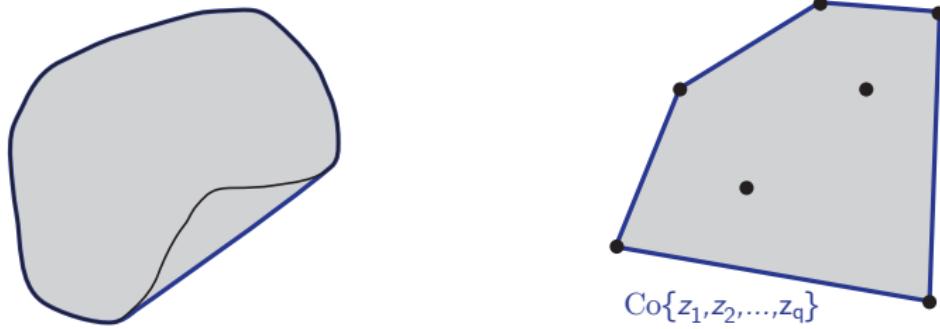
Many sets can be written as the intersection of convex elements, and are therefore easily shown to be convex. Any convex set can be written as a (possibly infinite) intersection of halfspaces.

Convex Hull $\text{Co}(Z)$

The **convex hull** of a set Z is the set of all convex combinations of points in Z :

$$\text{Co}(Z) := \{z \mid z = \lambda a + (1 - \lambda)b, \lambda \in [0, 1], a, b \in Z\}$$

It is the smallest convex set that contains Z : for all convex sets $\mathcal{Y} \supseteq Z$, $\text{Co}(Z) \subseteq \mathcal{Y}$.



For a set $Z = \{z_1, z_2, \dots, z_q\}$ comprising q points, the convex hull can be written

$$\text{Co}(Z) = \left\{ \lambda_1 z_1 + \lambda_2 z_2 + \dots + \lambda_q z_q \mid \lambda_i \geq 0, i = 1, \dots, q, \sum_{i=1}^q \lambda_i = 1 \right\}$$

Union $Z \cup Y$

Note that the union of two sets is **not** convex in general, regardless of whether the original sets were convex!

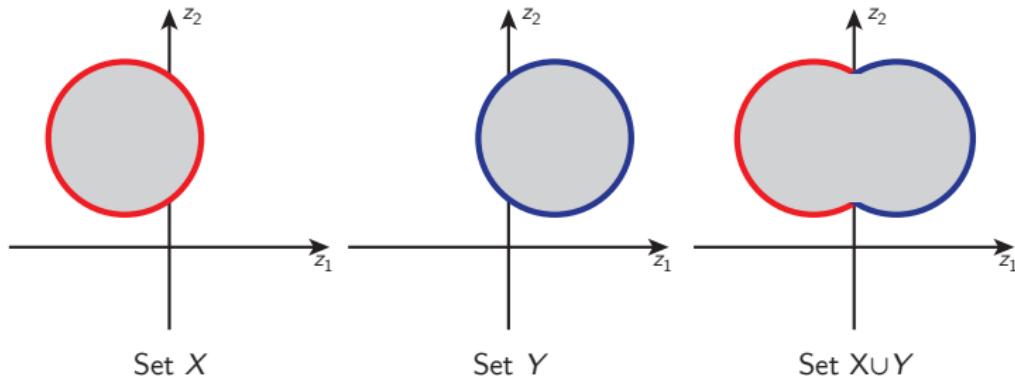


Table of Contents

1. Introduction
2. Convex Sets
3. Convex Functions
4. Convex Optimization Problems
5. Optimality Criteria
6. Optimality Conditions and Duality

Table of Contents

3. Convex Functions

3.1 Definitions

3.2 Examples

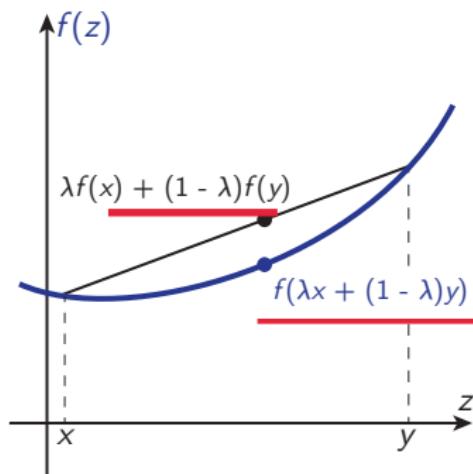
3.3 Properties

Definitions (Convex Function)

A function $f : \text{dom}(f) \rightarrow \mathbb{R}$ is **convex** iff¹ its domain $\text{dom}(f)$ is convex and

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y), \quad \forall \lambda \in (0, 1), \quad \forall x, y \in \text{dom}(f)$$

The function f is **strictly convex** if this inequality is strict.



f is **concave** iff the function $-f$ is convex.

¹"if and only if"

1st-order condition for convexity

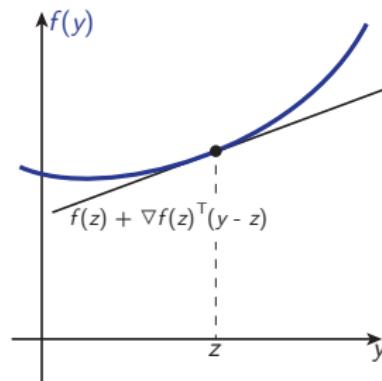
A differentiable function $f : \text{dom}(f) \rightarrow \mathbb{R}$ with a convex domain is **convex** iff

$$f(y) \geq f(z) + \nabla f(z)^T (y - z), \quad \forall z, y \in \text{dom}(f)$$

i.e. a first order approximator of f around any point z is a global underestimator of f .

The gradient $\nabla f(z)$ is given by

$$\nabla f(z) = \left[\frac{\partial f(z)}{\partial z_1}, \frac{\partial f(z)}{\partial z_2}, \dots, \frac{\partial f(z)}{\partial z_s} \right]^T$$



2nd-order condition for convexity

A twice-differentiable function $f : \text{dom}(f) \rightarrow \mathbb{R}$ is **convex** iff its domain $\text{dom}(f)$ is convex and

$$\nabla^2 f(z) \succeq 0, \quad \forall z \in \text{dom}(f),$$

where the Hessian $\nabla^2 f(z)$ is defined by

$$\nabla^2 f(z)_{ij} = \frac{\partial^2 f(z)}{\partial z_i \partial z_j}$$

If $\text{dom}(f)$ is convex and $\nabla^2 f(z) \succ 0$ for all $z \in \text{dom}(f)$, then f is **strictly convex**.

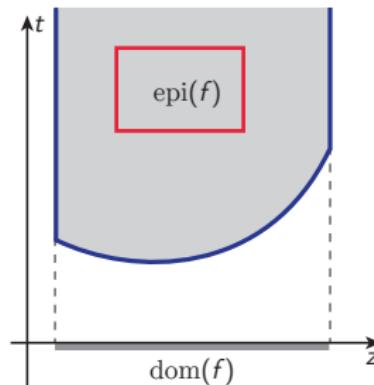
Epigraph of a Function

The **epigraph** of a function $f : \text{dom}(f) \rightarrow \mathbb{R}$ is the set

$$\text{epi}(f) = \left\{ \begin{bmatrix} z \\ t \end{bmatrix} \mid z \in \text{dom}(f), f(z) \leq t \right\} \subseteq \text{dom}(f) \times \mathbb{R}$$

It has dimension one higher than the domain of f .

A function is convex iff its epigraph is a convex set.



The epigraph of a convex function on a closed domain.

Level and sublevel sets

Definition (Level set)

The **level set** L_α of a function f for value α is the set of all $z \in \text{dom}(f)$ for which $f(z) = \alpha$:

$$L_\alpha = \{z \mid z \in \text{dom}(f), f(z) = \alpha\}$$

For $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ these are contour lines of constant “height”.

Definition (Sublevel set)

The **sublevel set** C_α of a function f for value α is defined by

$$C_\alpha = \{z \mid z \in \text{dom}(f), f(z) \leq \alpha\}$$

Function f is convex \Rightarrow sublevel sets of f are convex for all α . But not \Leftarrow !

Table of Contents

3. Convex Functions

3.1 Definitions

3.2 Examples

3.3 Properties

Examples of Convex Functions: $\mathbb{R} \rightarrow \mathbb{R}$

The following functions are **convex** (on domain \mathbb{R} unless otherwise stated):

- Affine: $ax + b$ for any $a, b \in \mathbb{R}$
- Exponential: e^{ax} for any $a \in \mathbb{R}$
- Powers: z^α on domain \mathbb{R}^+ , for $\alpha \geq 1$ or $\alpha \leq 0$
- Powers of absolute value: $|z|^p$, for $p \geq 1$

The following functions are **concave** (on domain \mathbb{R} unless otherwise stated):

- Affine: $ax + b$ for any $a, b \in \mathbb{R}$
- Powers: z^α on domain \mathbb{R}^+ , for $0 \leq \alpha \leq 1$
- Logarithm: $\log z$ on domain \mathbb{R}^+
- Entropy: $-z \log z$ on domain \mathbb{R}^+

Examples of Convex Functions: $\mathbb{R}^s \rightarrow \mathbb{R}$

Affine functions on \mathbb{R}^s are both convex and concave:

- On \mathbb{R}^s , for some $a \in \mathbb{R}^s$ and $b \in \mathbb{R}$:

$$f(z) = a^\top z + b$$

Vector Norms on \mathbb{R}^s are all convex:

- On \mathbb{R}^s , ℓ_p norms have the form, for $p \geq 1$,

$$\|z\|_p = \left(\sum_{i=1}^s |z_i|^p \right)^{1/p}, \quad \text{with } \|z\|_\infty = \max_i |z_i|$$


Examples of Convex Functions: $\mathbb{R}^{m \times n} \rightarrow \mathbb{R}$

Affine functions on $\mathbb{R}^{m \times n}$ are both convex and concave:

- On $\mathbb{R}^{m \times n}$, for some $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}$:

$$f(S) = \text{trace}(A^\top S) + b = \sum_{i=1}^m \sum_{j=1}^s A_{ij} S_{ij} + b$$

Matrix Norms on $\mathbb{R}^{m \times n}$ are all convex:

- On $\mathbb{R}^{m \times n}$ the **spectral**, or **maximum singular value** norm is

$$\|S\|_2 = \sigma_{\max}(S) = [\lambda_{\max}(S^\top S)]^{1/2}.$$

Table of Contents

3. Convex Functions

3.1 Definitions

3.2 Examples

3.3 Properties

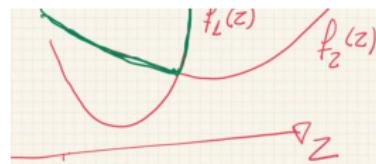
Convexity-preserving Operations

Certain operations preserve the convexity of functions:

- Non-negative weighted sum
- Composition with affine function
 $f(x) > f(ax+b)$
- Pointwise maximum and supremum
- Partial minimization

and many other possibilities...

$$\begin{aligned} & f_1(z) \quad f_2(z) \quad f_3(z) \\ & \text{CONVEX} \\ & \alpha f_1(z) + \beta f_2(z) + \gamma f_3(z) \\ & \text{ALSO CONVEX IF } \alpha, \beta, \gamma \geq 0 \end{aligned}$$



$$\max \left\{ f_1(z), f_2(z) \right\} = f_3(z)$$

↑
IF CONVEX
↑
THEN ALSO CONVEX

Convexity-preserving Operations

Theorem (Non-negative weighted sum)

If f is a function convex, then αf is convex for $\alpha \geq 0$. For several convex functions g_i , $\sum_i \alpha_i g_i$ is convex if all $\alpha_i \geq 0$.

Theorem (Composition with affine function)

If f is a convex function, then $f(Az + b)$ is convex.

Example: $\|Az - b\|$ is convex for any norm.

Theorem (Pointwise maximum)

If f_1, \dots, f_m are convex functions, then $f(z) = \max\{f_1(z), \dots, f_m(z)\}$ is convex.

Example: Piecewise linear functions $\max_{i=1,\dots,m} \{a_i^\top z + b\}$ are convex.

~~Convexity-preserving Operations (cont'd)~~

Theorem (Composition with scalar functions)

For $g : \mathbb{R}^s \rightarrow \mathbb{R}$ and $h : \mathbb{R} \rightarrow \mathbb{R}$, $f(z) = h(g(z))$ is convex if:

- g is convex, h is convex, h is non-decreasing
- g is concave, h is convex, h is non-increasing

Examples

- $\exp g(z)$ for convex g
- $1/g(z)$ for concave positive g

~~Convexity-preserving Operations (cont'd)~~

Theorem (Composition with vector functions)

For $g : \mathbb{R}^s \rightarrow \mathbb{R}^k$ and $h : \mathbb{R}^k \rightarrow \mathbb{R}$, $f(z) = h(g(z)) = h(g_1(z), g_2(z), \dots, g_k(z))$ is convex if:

- Each g_i is convex, h is convex, h is non-decreasing in each argument
- Each g_i is concave, h is convex, h is non-increasing in each argument

Examples

- $\log \sum_{i=1}^k \exp g_i(z)$ is convex if all g_i are positive
- $\sum_{i=1}^k \log g_i(z)$ is concave for concave positive g_i

Table of Contents

1. Introduction

2. Convex Sets

3. Convex Functions

4. Convex Optimization Problems

5. Optimality Criteria

6. Optimality Conditions and Duality

1. assess if MPC convex or not
2. not sure > cvxpy
3. start with LP,QP > convex
> fit MPC problem

Table of Contents

4. Convex Optimization Problems

- 4.1 Standard Convex Optimization Problem
- 4.2 Linear Programs
- 4.3 Quadratic Programs

Convex Optimization Problem

An optimization problem:

$$\min_{z \in Z} f(z)$$

subject to: $g_i(z) \leq 0 \quad i = 1, \dots, m$
 $h_j(z) = 0 \quad j = 1, \dots, p$

is convex if:

- The objective function f is a **convex function** on its domain Z .
- The feasible set S is a **convex set**.

Standard Form Convex Optimization Problem

A standard form **convex** optimization problem:

$$\min_{z \in Z} f(z)$$

subject to: $g_i(z) \leq 0 \quad i = 1, \dots, m$
 $a_j^\top z = b_j \quad j = 1, \dots, p$



equality cons should be linear

This problem is convex if:

- The domain Z is a convex set.
- The objective function f is a convex function.
- The inequality constraint functions g_i are all convex.

Standard Form Convex Optimization Problem

The affine constraints are typically gathered into matrix form:

$$\begin{aligned} & \min_{z \in Z} f(z) \\ \text{subject to: } & g_i(z) \leq 0 \quad i = 1, \dots, m \\ & Az = b \quad A \in \mathbb{R}^{p \times m} \end{aligned}$$

Crucial Fact!

Theorem

*For a convex optimization problem, **every** locally optimal solution is globally optimal.*

NB: Writing or rewriting an optimization problem in convex form can be tricky, and is not always possible. It is always worth trying though.

Standard Form Convex Optimization Problem

The affine constraints are typically gathered into matrix form:

$$\begin{aligned} & \min_{z \in Z} f(z) \\ \text{subject to: } & g_i(z) \leq 0 \quad i = 1, \dots, m \\ & Az = b \quad A \in \mathbb{R}^{p \times m} \end{aligned}$$

Crucial Fact!(2)

Theorem

Nice theory, analytic certificate of optimality, efficient algorithms.

Local and Global Optimality for Convex Problems

Theorem

For a convex optimization problem, **every** locally optimal solution is globally optimal.

Proof:

- Assume that z is locally optimal, but not globally optimal.
- Therefore there is some other point y such that $f(y) < f(z)$.
- z locally optimal implies that there is some $R > 0$ such that

$$\|z - z\|_2 \leq R \Rightarrow f(z) \leq f(z)$$

- The problem can't be convex.

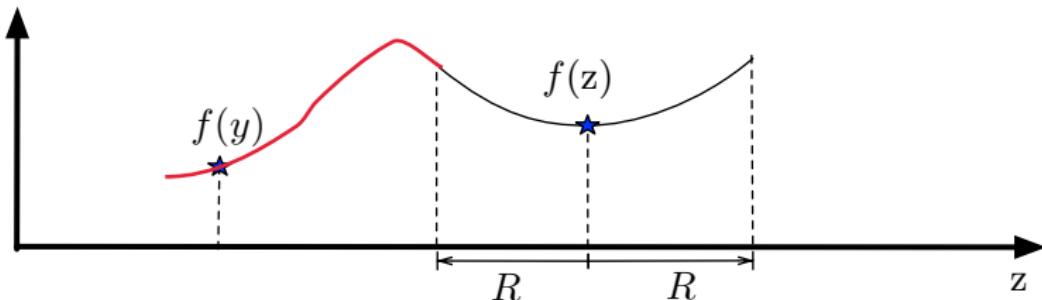


Table of Contents

4. Convex Optimization Problems

4.1 Standard Convex Optimization Problem

4.2 Linear Programs

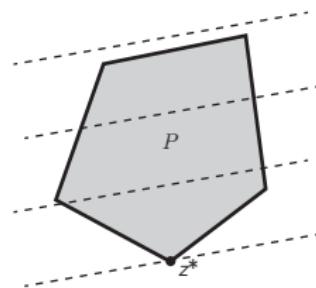
4.3 Quadratic Programs

General Linear Program (LP)

Affine cost and constraint functions:

$$\min_z \quad c^\top z + d$$

$$\begin{aligned} \text{subject to: } & Gz \leq h \\ & Az = b \end{aligned}$$



Linear optimization on a polytope.

- Feasible set is a polyhedron.
- Constant component d can be left out – it has no effect on the optimal solution.

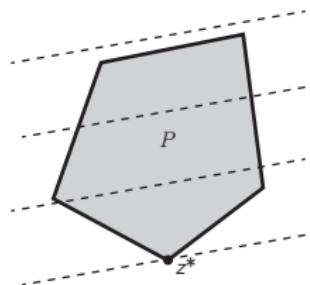
General Linear Program (LP)

An alternative format:

$$\min_z \quad c^T z$$

$$\text{subject to: } \begin{aligned} Az &= b \\ z &\geq 0 \end{aligned}$$

equal to the one before



Linear optimization on a polytope.

- All components of z are non-negative.
- Can easily convert previous format to this (using extra variables).

Many problems can be rewritten (with some effort!) into LPs.

Huge variety of solution methods and software are available.

Example Linear Programs

Cheapest cat-food problem:

- Choose quantities z_1, z_2, \dots, z_s of n different ingredients with unit cost c_j .
- Each ingredient j has nutritional content a_{ij} for nutrient i .
- Require for each nutrient i minimum level b_i .

In linear program form:

$$\begin{aligned} & \min_z \quad c^\top z \\ & \text{subject to: } Az \geq b \\ & \quad z \geq 0 \end{aligned}$$

This is an example of a resource allocation problem.

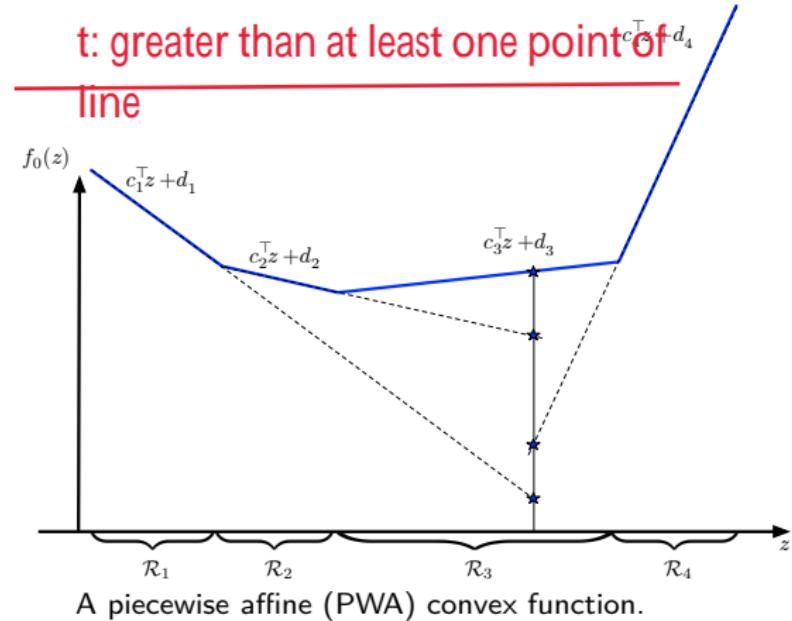
Kantorovich and Koopmans won the Nobel Prize in Economics in 1975 for their work on this problem (and its non-cat-food variants).

Example : Piecewise Affine Minimization

$$\min_z \quad \left[\max_{i=1,\dots,m} \{ c_i^\top z + d_i \} \right]$$

subject to: $Gz \leq h$

- The function is affine on each region \mathcal{R}_i .
- Any convex and piecewise affine function can be written this way.
- Can be reformulated as an LP.



Example Linear Programs

Piecewise affine minimization:

$$\min_z \quad \left[\max_{i=1,\dots,m} \{c_i^\top z + d_i\} \right]$$

subject to: $Gz \leq h$

is equivalent to an LP:

$$\min_{z,t} \quad t$$

subject to: $c_i^\top z + d_i \leq t \quad i = 1, \dots, m$

$Gz \leq h$

$$\begin{aligned} & \min_z \quad \max \{z+L, -z+5\} \\ \text{s.t. } & 5 \leq z \leq 7 \\ & z \in \mathbb{R} \\ & \min_{z,t} \quad t \\ \text{s.t. } & z+L \leq t \\ & -z+5 \leq t \\ & 5 \leq z \leq 7 \\ & \bar{z} = [z, t] \quad \min_{\bar{z}} \underline{c}^\top \bar{z} \\ & \underline{c}^\top = [0, 1] \quad \text{s.t. } \underline{A}\bar{z} \leq \underline{b} \\ & A = \begin{bmatrix} I & -I \\ -I & I \\ I & 0 \\ -I & 0 \end{bmatrix}, b = \begin{bmatrix} L \\ 5 \\ 7 \\ -5 \end{bmatrix} \end{aligned}$$

NB: trick was to add variables and write the problem in epigraph form.

ℓ_∞ minimizationConstrained ℓ_∞ (Chebyshev) minimization:

$$\min_{z \in \mathbb{R}^s} \|z\|_\infty$$

subject to: $Fz \leq g$

Write this is a max of linear functions.

Equivalent to:

$$\min_{z \in \mathbb{R}^s} [\max \{z_1, \dots, z_s, -z_1, \dots, -z_s\}]$$

subject to: $Fz \leq g$

same trick as before

ℓ_∞ minimization (cont'd)

Equivalent to:

$$\begin{array}{ll}
 \min_{z,t} & t \\
 \text{subject to:} & z_i \leq t \quad i = 1, \dots, n \\
 & -z_i \leq t \quad i = 1, \dots, n \\
 & Fz \leq g
 \end{array}
 \Rightarrow
 \begin{array}{ll}
 \min_{z,t} & t \\
 \text{subject to:} & -\mathbf{1}t \leq z \leq \mathbf{1}t \\
 & Fz \leq g
 \end{array}$$

- The notation ' $\mathbf{1}$ ' indicates a vector of ones.
- The constraint $-\mathbf{1}t \leq z \leq \mathbf{1}t$ bounds the absolute value of every element of z with a common scalar variable t .

ℓ_1 minimization**Constrained ℓ_1 minimization:**

$$\min_{z \in \mathbb{R}^s} \|Az - b\|_1$$

subject to: $Fz \leq g$ Write this is a max of linear functions. Assume $A \in \mathbb{R}^{m \times n}$.

Equivalent to:

$$\min_{z \in \mathbb{R}^s} \left[\sum_{i=1}^m \max \{(Az - b)_i, -(Az - b)_i\} \right]$$

subject to: $Fz \leq g$

$$\begin{aligned}
 & \min_{z \in \mathbb{R}^s} \left\| \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} \right\|_1 = \\
 & \min_z \left(|z_1| + |z_2| + \dots + |z_m| \right) = \\
 & \min_z \left[\max \{z_1, -z_1\} + \right. \\
 & \quad \left. \max \{z_2, -z_2\} + \dots + \right. \\
 & \quad \left. \max \{z_m, -z_m\} \right] \\
 & \min_z \frac{t_1 + t_2 + \dots + t_m}{t_i \geq 0}
 \end{aligned}$$

ℓ_1 minimization (cont'd)

Equivalent to:

$$\min_{z \in \mathbb{R}^s, t \in \mathbb{R}^m} t_1 + \dots + t_m$$

subject to:

$$(Az - b)_i \leq t_i \quad i = 1, \dots, m$$

$$-(Az - b)_i \leq t_i \quad i = 1, \dots, m$$

$$Fz \leq g$$

$$\begin{aligned} & \min_{z, t_1, t_2} t_1 + t_2 \\ \text{s.t. } & t_1 \geq z_1 + 3 \\ & t_1 \geq -z_1 - 3 \\ & t_2 \geq z_2 + 5 \\ & t_2 \geq -z_2 - 5 \end{aligned}$$

$$\begin{aligned} & \min_{z \in \mathbb{R}^s, t \in \mathbb{R}^m} \mathbf{1}^\top t \\ \text{subject to: } & -t \leq (Az - b) \leq t \\ & Fz \leq g \end{aligned}$$

- The notation ' $\mathbf{1}$ ' indicates a vector of ones.
- The constraint $-t \leq (Az - b) \leq t$ bounds the absolute value of each component of $(Az - b)$ with a component of the vector variable t .

Table of Contents

4. Convex Optimization Problems

- 4.1 Standard Convex Optimization Problem
- 4.2 Linear Programs
- 4.3 Quadratic Programs

General Quadratic Program

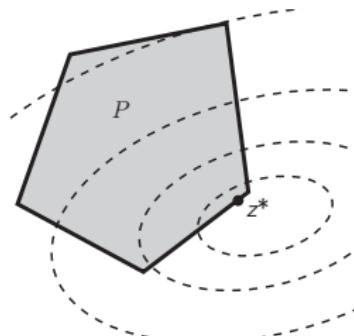
Quadratic cost function with $P \in \mathbb{S}_+^s$, affine constraint functions. Feasible set is a polyhedron:

$$\min_z \quad \frac{1}{2} z^\top P z + q^\top z + r$$

$$P > 0$$

$$\begin{aligned} \text{subject to: } & Gz \leq h \\ & Az = b \end{aligned}$$

- Constant component r can be left out, since it has no effect on the optimal solution.
- Maximization problems with a concave objective function ($-P \in \mathbb{S}_+^s$) are also quadratic programs.



Optimization of a quadratic objective function over a polytopic set P .
The level sets of the objective are shown as dotted lines.

Example Quadratic Programs - Least squares

Least squares:

$$\min_z \|Az - b\|_2^2$$

- Analytical solution $A^\dagger b$ (A^\dagger is the pseudo-inverse).
- Extra linear constraints $\underline{l} \leq z \leq \bar{u}$ can be added, although the QP would no longer have an analytical solution.

qp based model predictive control

Linear system dynamics

quadratic cost functions

Example Quadratic Programs

Quadratic program with random cost:

$$\begin{aligned} \min_z \quad & \mathbb{E}[c^T z] + \gamma \text{var}(c^T z) = \bar{c}^T z + \gamma z^T \Sigma z \\ \text{subject to: } & Gz \leq h \\ & Az = b \end{aligned}$$

- Random cost function vector c with mean \bar{c} and covariance Σ , we wish to penalize expected cost plus a “risk premium” γ on the variance.
- Hence $c^T z$ is a random variable with mean $\bar{c}^T z$ and variance $z^T \Sigma z$.
- Large γ means large risk aversion — we prefer a small variance to the lowest expected cost.

Example Quadratic Programs

Tikhonov Regularization: Least squares with extra penalty for nonzero terms.

$$\min_{z \in \mathbb{R}^s} \|Az - b\|_2^2 + \gamma \cdot \|z\|_1$$

Equivalent to:

$$\min_{z \in \mathbb{R}^s, t \in \mathbb{R}^s} \|Az - b\|_2^2 + \gamma \cdot \mathbf{1}^\top t$$

$$\text{subject to: } -t \leq z \leq t$$

- A larger penalty γ will tend to produce sparser solutions.
- Note that we have converted an unconstrained problem into a larger constrained one to get it into standard QP form.
- Requires $\gamma \geq 0$ for convexity.

Table of Contents

1. Introduction
2. Convex Sets
3. Convex Functions
4. Convex Optimization Problems
5. Optimality Criteria
6. Optimality Conditions and Duality

Optimality Criterion for Differentiable f

Theorem (Necessary condition*)

$f : \mathbb{R}^s \rightarrow \mathbb{R}$ is differentiable at z^* . If z^* is a local minimizer, then $\nabla f(z^*) = 0$.

Theorem (Sufficient condition*)

Suppose that $f : \mathbb{R}^s \rightarrow \mathbb{R}$ is twice differentiable at z^* . If $\nabla f(z^*) = 0$ and the Hessian of $f(z)$ at z^* is positive definite, then z^* is a local minimizer.



Theorem (Necessary and sufficient condition*)

Suppose that $f : \mathbb{R}^s \rightarrow \mathbb{R} = z^\top Hz + c^\top z + k$. If H is positive definite, then z^* is the global minimizer if and only if $\nabla f(z^*) = 0$.



*Proofs available in Chapter 4 of M.S. Bazaraa, H.D. Sherali, and C.M. Shetty.

Nonlinear Programming - Theory and Algorithms. John Wiley & Sons, Inc., New York, second edition, 1993.

A Well Known Optimization Problem - Least Squares

Least squares:

$$\min_{z \in \mathbb{R}^s} \|Az - b\|_2^2$$

- Analytical solution $z^* = A^\dagger b$, if A full column rank
- $A^\dagger = (A^T A)^{-1} A^T$
- A^\dagger is often called the pseudo-inverse
- In Matlab $z^* = A \backslash b$
- Proof:

$$\min_z \|Az - b\|_2^2 = \min_z z^T (A^T A) z - z^T (2A^T b) + b^T b$$

If A full column rank then $A^T A$ is positive definite, from previous Theorem at z^*

$$\nabla f(z^*) = 0 \Rightarrow (2A^T A)z^* = (2A^T b)$$

$$\begin{aligned}
 & \min_{z \in \mathbb{R}^s} \|Az - b\|_2^2 \\
 &= (Az - b)^T (Az - b) = \\
 &= z^T A^T A z - 2b^T A z + b^T b = f(z) \\
 & H = A^T A \rightarrow A^\dagger \text{ (sols)} A \quad \text{IDENTITY MATRIX} \\
 & \boxed{A^T} \quad \boxed{I} \quad \boxed{A} \\
 & \boxed{H \succeq 0}, \quad H \succ 0 \text{ if } \text{rank}(A) = s \\
 & \text{A full column rank}
 \end{aligned}$$

A Well Known Optimization Problem - Least Squares

Least squares:

$$\min_{z \in \mathbb{R}^s} \|Az - b\|_2^2$$

- Analytical solution $z^* = A^\dagger b$, if A full column rank
- $A^\dagger = (A^T A)^{-1} A^T$
- A^\dagger is often called the pseudo-inverse
- In Matlab $z^* = A \backslash b$ **python : numpy.linalg.pinv(A)**
- Proof:

$$\min_z \|Az - b\|_2^2 = \min_z z^T (A^T A) z - z^T (2A^T b) + b^T b$$

If A full column rank then $A^T A$ is positive definite, from previous Theorem at z^*

$$\nabla f(z^*) = 0 \Rightarrow (2A^T A)z^* = (2A^T b)$$

Notes:

- More details in Packard's online lecture "Least Squares: Part 8 (of 9)"
- We use $\|x\|_2$ for 2-norm of a vector x . Packard notes simply use $\|x\|$
- A minimizer z^* of the above problem is also a minimizer of

$$\min_{z \in \mathbb{R}^s} \|Az - b\|_2$$

Table of Contents

continue by new
slides

5. Optimality Criteria

5.1 Unconstrained Problems

5.2 Constrained Problems

Decent Direction for Differentiable f

Theorem (Descent Direction)

$f : \mathbb{R}^s \rightarrow \mathbb{R}$ differentiable at \bar{z} . If there exists a vector \mathbf{d} such that $\nabla f(\bar{z})' \mathbf{d} < 0$, then there exists a $\delta > 0$ such that $f(\bar{z} + \lambda \mathbf{d}) < f(\bar{z})$ for all $\lambda \in (0, \delta)$.

-
- The vector \mathbf{d} in the theorem above is called **descent direction**.
 - The direction of **steepest descent** \mathbf{d}_s at \bar{z} is defined as the normalized direction where $\nabla f(\bar{z})' \mathbf{d}_s < 0$ is minimized.
 - The direction \mathbf{d}_s of steepest descent is $\mathbf{d}_s = -\frac{\nabla f(\bar{z})}{\|\nabla f(\bar{z})\|}$.

Optimality Criterion for Differentiable f

Theorem (Necessary condition*)

$f : \mathbb{R}^s \rightarrow \mathbb{R}$ is differentiable at \bar{z} . If \bar{z} is a local minimizer, then $\nabla f(\bar{z}) = 0$.

Theorem (Sufficient condition*)

Suppose that $f : \mathbb{R}^s \rightarrow \mathbb{R}$ is twice differentiable at \bar{z} . If $\nabla f(\bar{z}) = 0$ and the Hessian of $f(z)$ at \bar{z} is positive definite, then \bar{z} is a local minimizer.

Theorem (Necessary and sufficient condition*)

Suppose that $f : \mathbb{R}^s \rightarrow \mathbb{R}$ is differentiable at \bar{z} . If f is convex, then \bar{z} is a global minimizer if and only if $\nabla f(\bar{z}) = 0$.

*Proofs available in Chapter 4 of M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. Nonlinear Programming - Theory and Algorithms. John Wiley & Sons, Inc., New York, second edition, 1993.

Table of Contents

5. Optimality Criteria

5.1 Unconstrained Problems

5.2 Constrained Problems

Optimality Conditions

Consider the problem

$$\begin{aligned} \min_z \quad & f(z) \\ \text{subject to: } \quad & g_i(z) \leq 0 \quad \text{for } i = 1, \dots, m \\ & h_j(z) = 0 \quad \text{for } j = 1, \dots, p \end{aligned}$$

- In general, an analytical solution does not exist.
- Solutions are usually computed by recursive algorithms which start from an initial guess z_0 and at step k generate a point z_k such that $\{f(z_k)\}_{k=0,1,2,\dots}$ converges to f^* .
- These algorithms recursively use and/or solve analytical **conditions for optimality**

KKT optimality conditions

$z^*, (u^*, v^*)$ of an optimization problem, with differentiable cost and constraints and zero duality gap, have to satisfy the following conditions:

$$0 = \nabla f(z^*) + \sum_{i=1}^m u_i^* \nabla g_i(z^*) + \sum_{j=1}^p v_j^* \nabla h_j(z^*), \quad (1a)$$

$$0 = u_i^* g_i(z^*), \quad i = 1, \dots, m \quad (1b)$$

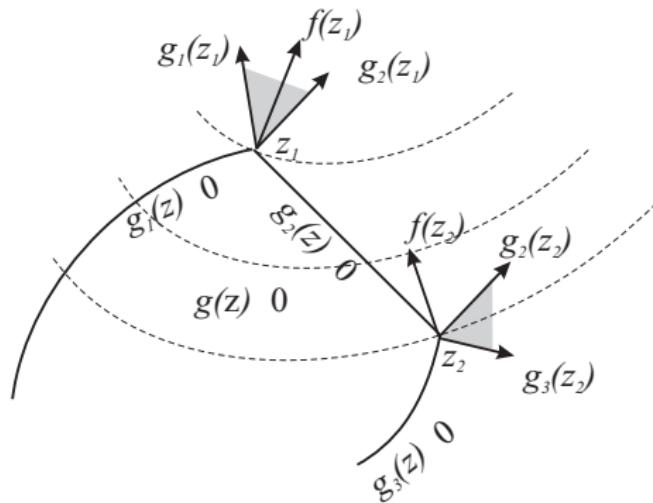
$$0 \leq u_i^*, \quad i = 1, \dots, m \quad (1c)$$

$$0 \geq g_i(z^*), \quad i = 1, \dots, m \quad (1d)$$

$$0 = h_j(z^*) \quad j = 1, \dots, p \quad (1e)$$

- Conditions (6a)-(6e) are called the Karush-Kuhn-Tucker (KKT) conditions.
- KKT are necessary and sufficient conditions for feasible LPs and QPs.
- (u^*, v^*) called “dual variables” or “Lagrange multipliers”.

KKT geometric interpretation



Rewrite the (6a), as

$$-\nabla f(z) = \sum_{i \in I} u_i \nabla g_i(z), \quad u_i \geq 0,$$

i.e., the direction of cost steepest descent belongs to the convex cone spanned by ∇g_i 's,

Table of Contents

1. Introduction
2. Convex Sets
3. Convex Functions
4. Convex Optimization Problems
5. Optimality Criteria
6. Optimality Conditions and Duality

Duality Theory. The Lagrange Function

Consider the **primal** optimization problem

$$\begin{aligned} \min_z \quad & f(z) \\ \text{subj. to} \quad & g_i(z) \leq 0 \quad \text{for } i = 1, \dots, m \\ & h_i(z) = 0 \quad \text{for } i = 1, \dots, p \\ & z \in Z = \mathbf{dom}(f) \cap \mathbf{dom}(g_i) \cap \mathbf{dom}(h_i) \end{aligned}$$

Any feasible point: upper bound of the optimal value

Lagrange dual problem: lower bound on optimal value

- Construct **Lagrange function**

$$\begin{aligned} L(z, u, v) = \quad & f(z) + u_1 g_1(z) + \dots + u_m g_m(z) + \\ & + v_1 h_1(z) + \dots + v_p h_p(z) \end{aligned}$$

- More compact

$$L(z, u, v) \triangleq f(z) + u' g(z) + v' h(z)$$

- u_i and v_i called **Lagrange multipliers** or dual variables
- objective is augmented with weighted sum of constraint functions

Duality Theory. The Lagrange Function

- Consider **Lagrange function**

$$L(z, u, v) \triangleq f(z) + u'g(z) + v'h(z)$$

Let $z \in S$ be feasible. For arbitrary vectors $u \geq 0$ and v trivially have

$$L(z, u, v) \leq f(z)$$

- After minimization we infer

$$\min_{z \in Z} L(z, u, v) \leq \min_{z \in Z, g(z) \leq 0, h(z)=0} f(z) = p^*$$

Duality Theory

Let

$$d(u, v) \triangleq \min_{z \in Z} L(z, u, v) \in [-\infty, +\infty] \quad (2)$$

Remarks

- Computing $d(u, v)$ is called **Lagrangian dual subproblem**.
- The (2) (Lagrangian dual subproblem) is an **unconstrained optimization problem**.
- $d(u, v)$ always **concave** (Requires proof)
- The point (u, v) is dual feasible if $u \geq 0$ and $(u, v) \in \text{dom}(d)$
- $d(u, v)$ is always a lowerbound for p^* for any feasible (u, v)

Lagrangian Dual Problem

Since $u \geq 0$ and v are arbitrary, any feasible choice gives a lowerbound. Best lower bound:

$$\max_{(u,v), u \geq 0} d(u, v) \quad (3)$$

Remarks

- Since $d(u, v)$ always **concave** \Rightarrow the dual problem (3) is convex, much easier to solve than the primal (non convex in general)
- The point (u, v) is dual feasible if $u \geq 0$ and $(u, v) \in \text{dom}(d)$
- **Weak duality** always holds:

$$\max_{(u,v), u \geq 0} d(u, v) \leq \min_{z \in Z, g(z) \leq 0, h(z)=0} f(z)$$

Duality Theory. Duality Gap and Certificate of Optimality

Let:

$$d^* = \max_{(u,v), u \geq 0} d(u, v)$$

$$f^* = \min_{z \in Z, g(z) \leq 0, h(z)=0} f(z)$$

then

- we always have $d^* \leq f^*$
- $d^* - f^*$ is called **optimal duality gap**
- **Strong duality** if $d^* = f^*$
- In case of strong duality u^* and v^* serve as **certificate of optimality**

Strong Duality

- It is **sometimes** true that $d^* = p^*$.
- Strong duality usually holds for convex problems. In general need **Constraint qualifications**. Conditions on the constraint functions implying strong duality for convex problems. Satisfied for feasible LPs and QPs.
- Strong duality usually does not hold for non-convex problems.

Duality Theory. Duality Gap and Certificate of Optimality

Let:

$$d^* = \max_{(u,v), u \geq 0} d(u, v)$$

$$f^* = \min_{z \in Z, g(z) \leq 0, h(z)=0} f(z)$$

then

- we always have $d^* \leq f^*$
- $d^* - f^*$ is called **optimal duality gap**
- **Strong duality** if $d^* = f^*$
- In case of strong duality u^* and v^* serve as **certificate of optimality**

Table of Contents

6. Optimality Conditions and Duality

6.1 Complementarity slackness

6.2 KKT conditions

Complementary slackness

Suppose that z^*, u^*, v^* are primal, dual feasible with zero duality gap (hence, they are primal, dual optimal)

$$\begin{aligned} f(z^*) &= d(u^*, v^*) \\ &= \inf_z (f(z) + u^{*\prime} g(z) + v^{*\prime} h(z)) \\ &\leq f(z^*) + u^{*\prime} g(z^*) + v^{*\prime} h(z^*) \end{aligned}$$

hence we have $\sum_{i=1}^m u_i^* g_i(z^*) = 0$ and so

$$u_i^* g_i(z^*) = 0, \quad i = 1, \dots, m$$

- called **complementary slackness** condition
- i -th constraint inactive at optimum $\implies u_i = 0$
- $u_i^* > 0$ at optimum $\implies i$ -th constraint active at optimum

Table of Contents

6. Optimality Conditions and Duality

6.1 Complementarity slackness

6.2 KKT conditions

KKT optimality conditions

Suppose

- f, g_i, h_i are differentiable
- z^*, u^*, v^* are (primal, dual) optimal, with zero duality gap

by complementary slackness we have

$$f(z^*) = f(z^*) + \sum_i u_i^* g_i(z^*) + \sum_j v_j^* h_j(z^*) \quad (4)$$

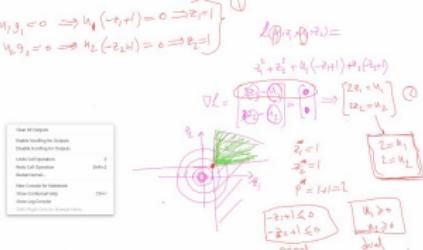
and by definition of dual function we have

$$\begin{aligned} f(z^*) + \sum_i u_i^* g_i(z^*) + \sum_j v_j^* h_j(z^*) &= \\ \min_z \left(f(z) + \sum_i u_i^* g_i(z) + \sum_j v_j^* h_j(z) \right) \end{aligned} \quad (5)$$

i.e., z^* minimizes $L(z, u^*, v^*)$ therefore

$$\nabla f(z^*) + \sum_i u_i^* \nabla g_i(z^*) + \sum_j v_j^* \nabla h_j(z^*) = 0$$

KKT optimality conditions



$z^*, (u^*, v^*)$ of an optimization problem, with differentiable cost and constraints and zero duality gap, have to satisfy the following conditions:

$$0 = \nabla f(z^*) + \sum_{i=1}^m u_i^* \nabla g_i(z^*) + \sum_{j=1}^p v_j^* \nabla h_j(z^*), \quad L() = 0 \quad (6a)$$

$$0 = u_i^* g_i(z^*), \quad i = 1, \dots, m \quad \text{complexity slackness} \quad (6b)$$

$$0 \leq u_i^*, \quad i = 1, \dots, m \quad \text{dual} \quad (6c)$$

$$0 \geq g_i(z^*), \quad i = 1, \dots, m \quad \text{primal} \quad (6d)$$

$$0 = h_j(z^*) \quad j = 1, \dots, p \quad (6e)$$

Conditions (6a)-(6e) are called the Karush-Kuhn-Tucker (KKT) conditions.

KKT optimality conditions

Consider a NLP in standard form.

Theorem (Necessary and sufficient condition)

Suppose that the NLP is convex and that cost and constraints f , g_i and h_i are differentiable at a feasible z^ . If the NLP satisfies constraint qualification conditions then z^* is optimal if and only if there are (u^*, v^*) that, together with z^* , satisfy the KKT conditions.*

Unconstrained and Constrained Optimal Control

F. Borrelli, M. Morari, C. Jones

UC Berkeley

ETH Zürich

EPFL Lausanne

Fall Semester 2020

Table of Contents

1. Optimal Control

1.1 Introduction

1.2 Batch Approach

1.3 A Note on Continuous-Time CFTOC

2. Linear Quadratic Optimal Control

2.1 Batch Approach Solution

3. Constrained Linear Optimal Control

3.1 Problem formulation

3.2 Feasible Sets

4. Constrained Optimal Control: 2-Norm

4.1 Problem Formulation

4.2 Construction of the QP with substitution

4.3 Construction of the QP without substitution

5. Constrained Optimal Control: 1-Norm and ∞ -Norm

5.1 Problem Formulation

5.2 Construction of the LP with substitution

Table of Contents

1. Optimal Control

1.1 Introduction

1.2 Batch Approach

1.3 A Note on Continuous-Time CFTOC

Optimal Control Introduction (1/2)

- Discrete-time *optimal control* is concerned with choosing an optimal input sequence $U_{0 \rightarrow N} \triangleq [u'_0, u'_1, \dots]'$ (as measured by some objective function), over a finite or infinite time horizon, in order to apply it to a system with a given initial state $x(0)$.
- The objective, or cost, function is often defined as a sum of *stage costs* $q(x_k, u_k)$ and, when the horizon has finite length N , a *terminal cost* $p(x_N)$:

$$J_{0 \rightarrow N}(x_0, U_{0 \rightarrow N}) \triangleq p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k)$$

stage
terminal

- The states $\{x_k\}_{k=0}^N$ must satisfy the system dynamics

$$x_{k+1} = g(x_k, u_k), \quad k = 0, \dots, N-1$$

$$x_0 = x(0) \quad \text{initial constraint}$$

and there may be state and/or input constraints

$$h(x_k, u_k) \leq 0, \quad k = 0, \dots, N-1.$$

Optimal Control Introduction (2/2)

- In the finite horizon case, there may also be a constraint that the final state x_N lies in a set \mathcal{X}_f

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

- A general finite horizon optimal control formulation for discrete-time systems is therefore

opt input seq 0-N

$$J_{0 \rightarrow N}^*(x(0)) \triangleq \min_{U_{0 \rightarrow N}} J_{0 \rightarrow N}(x(0), U_{0 \rightarrow N})$$

cost
 subject to $x_{k+1} = g(x_k, u_k), k = 0, \dots, N - 1$ state constraint
 $h(x_k, u_k) \leq 0, k = 0, \dots, N - 1$
 $x_N \in \mathcal{X}_f$ terminal
 $x_0 = x(0)$ known

Regulation Problem Introduction

Consider the nonlinear time-invariant system **discrete time**

$$x(t+1) = g(x(t), u(t)),$$

subject to the constraints

origin is an eq. point

$$h(x(t), u(t)) \leq 0, \forall t \geq 0$$

with $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ the state and input vectors. **Assume that** $g(0, 0) = 0, h(0, 0) \leq 0$

origin is feasible

The control objectives are $x_f = 0$

- control the system to the origin (also referred to "regulate") in N Steps
- satisfying state and input constraints
- minimizing a cost function over the control horizon

Solution Ingredients

- A prediction model

$$x_{k+1} = g(x_k, u_k), \quad k = 1, \dots, N-1$$

and $x_0 = x(0)$, (note no model mismatch with real system)

- The time *horizon* N (often called prediction horizon or control horizon)
- The input sequence over the horizon:

$$U_{0 \rightarrow N} := [u'_0, \dots, u'_{N-1}]' \in \mathbb{R}^s, \quad s := mN$$

- The cost function

$$J_{0 \rightarrow N}(x_0, U_{0 \rightarrow N}) := p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k)$$


where $q(x_k, u_k)$ and $p(x_N)$ are the *stage cost* and *terminal cost*, respectively.

- $\mathcal{X}_f \subseteq \mathbb{R}^n$ is a *terminal region*, that we want the system states to reach at the end of the horizon. 0 in case of a regulation problem
- In standard regulation problems $q(x, u) \succeq 0$ and $p(x) \succeq 0$

semi PD

The CFTOC problem

The **Constrained Finite Time Optimal Control (CFTOC)** problem.

$$\begin{aligned}
 J_{0 \rightarrow N}^*(x_0) = & \min_{U_{0 \rightarrow N}} J_{0 \rightarrow N}(x_0, U_{0 \rightarrow N}) \\
 \text{subj. to } & x_{k+1} = g(x_k, u_k), \quad k = 0, \dots, N-1 \\
 & h(x_k, u_k) \leq 0, \quad k = 0, \dots, N-1 \\
 \text{OPT. VAR:} & x_N \in \mathcal{X}_f \\
 & x_0 = x(0) \\
 & \underline{u_0, u_1, \dots, u_{N-1}} \\
 & \underline{x_0, x_1, \dots, x_N}
 \end{aligned}$$

- $\mathcal{X}_{0 \rightarrow N} \subseteq \mathbb{R}^n$ to is the set of feasible initial conditions $x(0)$
- the optimal cost $J_{0 \rightarrow N}^*(x_0)$ is called *value function*,
- assume that there exists a minimum
- denote by $U_{0 \rightarrow N}^*$ one of the minima ($U_{0 \rightarrow N}^* = [u_0^*, \dots, u_{N-1}^*]'$)
- We apply to the system

$$\underline{u(t) = u_t^*, \quad t = 0, \dots, N-1}$$

Objectives

■ **Finite Time Solution**

- 1 a general nonlinear programming problem (*batch approach*)
- 2 recursively by invoking Bellman's Principle of Optimality (*recursive approach*)
- 3 discuss in details the linear system case dynamic program

■ **Infinite Time Solution.** We will investigate

- 1 if a solution exists as $N \rightarrow \infty$
- 2 the properties of this solution

■ **Uncertainty.** We will not discuss how to extend the problem description and consider uncertainty. See advanced MPC class

Table of Contents

1. Optimal Control

1.1 Introduction

1.2 Batch Approach

1.3 A Note on Continuous-Time CFTOC

Solution via Batch Approach. NLP formulation

Write the equality constraints from system constraints as

$$\begin{aligned}x_1 &= g(x(0), u_0) \\x_2 &= g(x_1, u_1) \\&\vdots \\x_N &= g(x_{N-1}, u_{N-1})\end{aligned}$$

then the optimal control problem

$$\begin{aligned}J_{0 \rightarrow N}^*(x_0) = \min_{U_{0 \rightarrow N}} \quad & p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) \\ \text{subj. to} \quad & x_1 = g(x_0, u_0) \\ & x_2 = g(x_1, u_1) \\ & \vdots \\ & x_N = g(x_{N-1}, u_{N-1}) \\ & h(x_k, u_k) \leq 0, \quad k = 0, \dots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(0)\end{aligned}$$

constraints

is a general Non Linear Programming (NLP) problem with variables u_0, \dots, u_{N-1} and x_1, \dots, x_N .

Batch Approach Solution

- In this class we will “trust” a general-purpose nonlinear solver
- One could design more efficient solvers for tailored problem (will not do this)
- Sometimes work to do in formulating cost and constraints (see “Practice and Theory” slide)

Batch Approach Example

This [Open in Colab](#) example will show how to use CFTOC to control a Unicycle. Try to

- Play with input constraints
- Add state constraints
- Change horizon
- Change cost function

Practice and Theory

The CFTOC formulation in this set of slides is the simplest possible. In Practice:

- Reference Tracking
- Time-Varying Constraints
- Rate constraints
- Prediction of other “agents”
- Time-varying terminal set
- Cost Shaping and Tuning
- Soft Constraints *hard~*
- Time vs Space formulations
- Obstacle avoidance constraints

You will learn (some of) the above through Labs, HW examples and in your project.

In the next lectures we will focus on the theory/design principles, learning:

- Feedback Policy vs Open-Loop Predictions
- Analytical Solution for the linear system case
- LP and QP formulation
- Controller Feasible Sets
- Dynamic Programming

Table of Contents

1. Optimal Control

1.1 Introduction

1.2 Batch Approach

1.3 A Note on Continuous-Time CFTOC

Batch Approach Solution

The equivalent continuous-time version is well studied in the literature as well. Let $T_f = NT_s$ where T_s is the sampling time.

c.t d.t

integral

$$J_{0 \rightarrow T_f}^*(\bar{x}_0) = \min_{U_0 \rightarrow T_f} p(x(T_f)) + \int_0^{T_f} q(x(t), u(t)) \, dt$$

subj. to

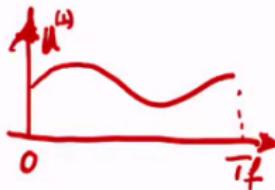
ODE

$$\dot{x} = g_c(x(t), u(t)), \quad t \in [0, T_f]$$

$$h(x(t), u(t)) \leq 0, \quad t \in [0, T_f]$$

$$x(T_f) \in \mathcal{X}_f$$

$$x(0) = \bar{x}_0$$



- It is an infinite-dimensional optimization problem
- We will not focus on these type of problems
- Pyomo allows to formulate and solve problems also in CT. (See second part of the Unicycle problem as example)
- Pyomo has different methods implemented to transform it into a finite-dimensional optimization problem

Outline

1. Optimal Control
2. Linear Quadratic Optimal Control linear sys
 - 2.1 Batch Approach Solution
3. Constrained Linear Optimal Control
4. Constrained Optimal Control: 2-Norm
5. Constrained Optimal Control: 1-Norm and ∞ -Norm

Linear Quadratic Optimal Control

finite

- In this section, only *linear* discrete-time time-invariant systems

$$x(k+1) = Ax(k) + Bu(k)$$

and *quadratic* cost functions ABPQR are given

$$J_0(x_0, U_0) \triangleq x'_N P x_N + \sum_{k=0}^{N-1} [x'_k Q x_k + u'_k R u_k] \quad (1)$$

are considered, and we consider only the problem of regulating the state to the origin, *without state or input constraints*.

- The two most common solution approaches will be described here
 - Batch Approach*, which yields a series of *numerical values* for the input
 - Recursive Approach*, which uses Dynamic Programming to compute control *policies* or *laws*, i.e. functions that describe how the control decisions depend on the system states.

Unconstrained Finite Horizon Control Problem

- **Goal:** Find a sequence of inputs $U_0 \triangleq [u'_0, \dots, u'_{N-1}]'$ that minimizes the objective function

$$J_0^*(x(0)) \triangleq \min_{U_0} \quad \begin{array}{c} \text{state} \\ x'_N P x_N + \sum_{k=0}^{N-1} [x'_k Q x_k + u'_k R u_k] \end{array}$$

subject to $x_{k+1} = Ax_k + Bu_k, k = 0, \dots, N-1$ linear
 $x_0 = x(0)$

opt is unique

- $P \succeq 0$, with $P = P'$, is the *terminal weight*
- $Q \succeq 0$, with $Q = Q'$, is the *state weight* **cost is convex**
- $R \succ 0$, with $R = R'$, is the *input weight*
- N is the horizon length
- Note that $x(0)$ is the current state, whereas x_0, \dots, x_N and u_0, \dots, u_{N-1} are *optimization variables* that are constrained to obey the system dynamics and the initial condition.

Table of Contents

2. Linear Quadratic Optimal Control

2.1 Batch Approach Solution

Final Result

- The problem is unconstrained
- Setting the gradient to zero:

$$U_0^*(x(0)) = \mathbf{K}x(0)$$

control law

- which implies

$$u^*(0)(x(0)) = K_0x(0), \dots, u^*(N-1)(x(0)) = K_{N-1}x(0)$$

which is a linear, open-loop controller function of the initial state $x(0)$.

- The optimal cost is

$$J_0^*(x(0)) = x(0)'P_0x(0)$$

which is a positive definite quadratic function of the initial state $x(0)$.

The diagram shows a handwritten derivation of the optimal control law. It starts with the expression $U_{0 \rightarrow N}^* = \begin{bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{N-1}^* \end{bmatrix} = \begin{bmatrix} \square \\ \square \\ \vdots \\ \square \end{bmatrix} = \begin{bmatrix} K_0 \\ K_1 \\ \vdots \\ K_{N-1} \end{bmatrix} \cdot \begin{bmatrix} x(0) \end{bmatrix}$. Below this, it is noted that $K \in \mathbb{R}^{S \times n}$, $K_0 \in \mathbb{R}^{n \times n}$, $S = N \cdot m$, and $K_L \in \mathbb{R}^{n \times n}$.

Solution approach 1: Batch Approach (1/4)

- The batch solution explicitly represents all future states x_k in terms of initial condition x_0 and inputs u_0, \dots, u_{N-1} .
- Starting with $x_0 = x(0)$, we have $x_1 = Ax(0) + Bu_0$, and $x_2 = Ax_1 + Bu_1 = A^2x(0) + ABu_0 + Bu_1$, by substitution for x_1 , and so on. Continuing up to x_N we obtain:

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} I \\ A \\ \vdots \\ A^N \end{bmatrix} x(0) + \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ A^{N-1}B & \cdots & AB & B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

- The equation above can be represented as

$$\mathcal{X} \triangleq \mathcal{S}^x x(0) + \mathcal{S}^u U_0. \quad (2)$$

Solution approach 1: Batch Approach (2/4)

- Define

$\begin{bmatrix} Q \\ & \ddots \\ & & P \end{bmatrix}$

P: terminal

$Rbar = np.kron(np.eye(N), R)$

$$\longrightarrow \overline{Q} \triangleq \text{blockdiag}(Q, \dots, Q, P) \quad \text{and} \quad \overline{R} \triangleq \text{blockdiag}(R, \dots, R)$$

$Qbar = \text{block_diag}(np.kron(np.eye(N), Q), PN)$

Then the finite horizon cost function (1) can be written as

$$J_0(x(0), U_0) = \mathcal{X}' \overline{Q} \mathcal{X} + U_0' \overline{R} U_0 . \quad (3)$$

- Eliminating \mathcal{X} by substituting from (2), equation (3) can be expressed as:

$$\begin{aligned} J_0(x(0), U_0) &= (\mathcal{S}^x x(0) + \mathcal{S}^u U_0)' \overline{Q} (\mathcal{S}^x x(0) + \mathcal{S}^u U_0) + U_0' \overline{R} U_0 \\ &= U_0' H U_0 + 2x(0)' F U_0 + x(0)' \mathcal{S}^x' \overline{Q} \mathcal{S}^x x(0) \end{aligned}$$

where $H \triangleq \mathcal{S}^u' \overline{Q} \mathcal{S}^u + \overline{R}$ and $F \triangleq \mathcal{S}^x' \overline{Q} \mathcal{S}^u$.

- Note that $H \succ 0$, since $R \succ 0$ and $\mathcal{S}^u' \overline{Q} \mathcal{S}^u \succeq 0$.

Y mat

Solution approach 1: Batch Approach (3/4)

- Since the problem is unconstrained and $J_0(x(0), U_0)$ is a positive definite quadratic function of U_0 we can solve for the optimal input U_0^* by setting the gradient with respect to U_0 to zero:

$$\begin{aligned}\nabla_{U_0} J_0(x(0), U_0) &= 2HU_0 + 2F'x(0) = 0 \\ \Rightarrow U_0^*(x(0)) &= -H^{-1}F'x(0) \\ &= -(\mathcal{S}^{u'}\bar{Q}\mathcal{S}^u + \bar{R})^{-1}\mathcal{S}^{u'}\bar{Q}\mathcal{S}^x x(0) \\ &= \mathbf{K}x(0),\end{aligned}$$

which is a linear function of the initial state $x(0)$.

Note H^{-1} always exists, since $H \succ 0$ and therefore has full rank.

- The optimal cost can be shown (by back-substitution) to be

$$\begin{aligned}J_0^*(x(0)) &= -x(0)'FHF'x(0) + x(0)'\mathcal{S}^x\bar{Q}\mathcal{S}^x x(0) \\ &= x(0)'(\mathcal{S}^x\bar{Q}\mathcal{S}^x - \mathcal{S}^x\bar{Q}\mathcal{S}^u(\mathcal{S}^{u'}\bar{Q}\mathcal{S}^u + \bar{R})^{-1}\mathcal{S}^{u'}\bar{Q}\mathcal{S}^x)x(0),\end{aligned}$$

P0

Solution approach 1: Batch Approach (4/4)

Summary

- The Batch Approach expresses the cost function in terms of the initial state $x(0)$ and input sequence U_0 by eliminating the states x_k .
- Because the cost $J_0(x(0), U_0)$ is a strictly convex quadratic function of U_0 , its minimizer U_0^* is unique and can be found by setting $\nabla_{U_0} J_0(x(0), U_0) = 0$. This gives the optimal input sequence U_0^* as a linear function of the initial state $x(0)$:

$$\begin{aligned} U_0^*(x(0)) &= -(\mathcal{S}^{u'} \bar{\mathcal{Q}} \mathcal{S}^u + \bar{R})^{-1} \mathcal{S}^{u'} \bar{\mathcal{Q}} \mathcal{S}^x x(0) \\ &= \mathbf{K}x(0) \end{aligned}$$

- The optimal cost is a quadratic function of the initial state $x(0)$

$$J_0^*(x(0)) = x(0)' (\mathcal{S}^{x'} \bar{\mathcal{Q}} \mathcal{S}^x - \mathcal{S}^{x'} \bar{\mathcal{Q}} \mathcal{S}^u (\mathcal{S}^{u'} \bar{\mathcal{Q}} \mathcal{S}^u + \bar{R})^{-1} \mathcal{S}^{u'} \bar{\mathcal{Q}} \mathcal{S}^x) x(0)$$

- If there are state or input constraints, solving this problem by matrix inversion is not guaranteed to result in a feasible input sequence

Outline

1. Optimal Control

2. Linear Quadratic Optimal Control

3. Constrained Linear Optimal Control

3.1 Problem formulation

3.2 Feasible Sets

4. Constrained Optimal Control: 2-Norm

5. Constrained Optimal Control: 1-Norm and ∞ -Norm

Table of Contents

3. Constrained Linear Optimal Control

3.1 Problem formulation

3.2 Feasible Sets

Constrained Linear Optimal Control

Cost function

$$J_0(x(0), U_0) = p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k)$$



- $U_0 \triangleq [u'_0, \dots, u'_{N-1}]'$
- Squared Euclidian norm: $p(x_N) = x'_N P x_N$ and $q(x_k, u_k) = x'_k Q x_k + u'_k R u_k$.
- $p = 1$ or $p = \infty$: $p(x_N) = \|P x_N\|_p$ and $q(x_k, u_k) = \|Q x_k\|_p + \|R u_k\|_p$.

Constrained Finite Time Optimal Control problem (CFTOC)

$$\begin{aligned} J_0^*(x(0)) = & \min_{U_0} \quad J_0(x(0), U_0) \\ \text{subj. to } & x_{k+1} = A x_k + B u_k, \quad k = 0, \dots, N-1 \\ & x_k \in \mathcal{X}, \quad u_k \in \mathcal{U}, \quad k = 0, \dots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(0) \end{aligned} \tag{4}$$

N is the time horizon and $\mathcal{X}, \mathcal{U}, \mathcal{X}_f$ are polyhedral regions.

may be infeasible with some constraint -> no input

Table of Contents

3. Constrained Linear Optimal Control

3.1 Problem formulation

3.2 Feasible Sets

Feasible Sets

Set of initial states $x(0)$ for which the optimal control problem (4) is feasible:

$$\mathcal{X}_0 = \{x_0 \in \mathbb{R}^n \mid \exists(u_0, \dots, u_{N-1}) \text{ such that } x_k \in \mathcal{X}, u_k \in \mathcal{U}, \\ k = 0, \dots, N-1, x_N \in \mathcal{X}_f, \text{ where } x_{k+1} = Ax_k + Bu_k\}$$

In general \mathcal{X}_i is the set of states x_i at time i for which (4) is feasible:

$$\mathcal{X}_i = \{x_i \in \mathbb{R}^n \mid \exists(u_i, \dots, u_{N-1}) \text{ such that } x_k \in \mathcal{X}, u_k \in \mathcal{U}, \\ k = i, \dots, N-1, x_N \in \mathcal{X}_f, \text{ where } x_{k+1} = Ax_k + Bu_k\},$$

The sets \mathcal{X}_i for $i = 0, \dots, N$ play an important role in the solution of the CFTOC problem. They are independent of the cost.

Outline

1. Optimal Control

2. Linear Quadratic Optimal Control

3. Constrained Linear Optimal Control

4. Constrained Optimal Control: 2-Norm

4.1 Problem Formulation

4.2 Construction of the QP with substitution

4.3 Construction of the QP without substitution

5. Constrained Optimal Control: 1-Norm and ∞ -Norm

Table of Contents

4. Constrained Optimal Control: 2-Norm

4.1 Problem Formulation

4.2 Construction of the QP with substitution

4.3 Construction of the QP without substitution

Problem Formulation

Quadratic cost function

$$J_0(x(0), U_0) = x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k \quad (5)$$

with $P \succeq 0$, $Q \succeq 0$, $R \succ 0$.

Constrained Finite Time Optimal Control problem (CFTOC).

$$\begin{aligned} J_0^*(x(0)) = \min_{U_0} \quad & J_0(x(0), U_0) \\ \text{subj. to} \quad & x_{k+1} = Ax_k + Bu_k, \quad k = 0, \dots, N-1 \\ & x_k \in \mathcal{X}, \quad u_k \in \mathcal{U}, \quad k = 0, \dots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(0) \end{aligned} \quad (6)$$

N is the time horizon and \mathcal{X} , \mathcal{U} , \mathcal{X}_f are polyhedral regions.

Table of Contents

4. Constrained Optimal Control: 2-Norm

4.1 Problem Formulation

4.2 Construction of the QP with substitution

4.3 Construction of the QP without substitution

Construction of the QP with substitution

- **Step 1:** Rewrite the cost as (see previous slides)

$$\begin{aligned} J_0(x(0), U_0) &= U_0' H U_0 + 2x(0)' F U_0 + x(0)' Y x(0) \\ &= [U_0' \ x(0)'] \begin{bmatrix} H & F' \\ F & Y \end{bmatrix} [U_0' \ x(0)']' \end{aligned}$$

- **Step 2:** Rewrite the constraints compactly as (details provided on the next slide)

$$G_0 U_0 \leq w_0 + E_0 x(0)$$

- **Step 3:** Rewrite the optimal control problem as

$$\boxed{\begin{aligned} J_0^*(x(0)) &= \min_{U_0} \quad [U_0' \ x(0)'] \begin{bmatrix} H & F' \\ F & Y \end{bmatrix} [U_0' \ x(0)']' \\ \text{subj. to} \quad G_0 U_0 &\leq w_0 + E_0 x(0) \end{aligned}}$$

Solution

$$\begin{aligned} J_0^*(x(0)) = \min_{U_0} \quad & [U_0' \ x(0)'] \begin{bmatrix} H & F' \\ F & Y \end{bmatrix} [U_0' \ x(0)']' \\ \text{subj. to} \quad & G_0 U_0 \leq w_0 + E_0 x(0) \end{aligned}$$

For a given $x(0)$ U_0^* can be found via a QP solver.

cvx.qp

IMP:
b and Z is diff!!!!

Construction of QP constraints with substitution

If \mathcal{X} , \mathcal{U} and \mathcal{X}_f are given by:

ABPQR given

$$G_0 U_0 \leq w_0 + E_0 x(0)$$

$$\mathcal{X} = \{x \mid A_x x \leq b_x\} \quad \mathcal{U} = \{u \mid A_u u \leq b_u\} \quad \mathcal{X}_f = \{x \mid A_f x \leq b_f\}$$

Then G_0 , E_0 and w_0 are defined as follows

$$G_0 = \begin{bmatrix} A_u & 0 & \dots & 0 \\ 0 & A_u & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_u \\ \hline 0 & 0 & \dots & 0 \\ A_x B & 0 & \dots & 0 \\ A_x A B & A_x B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_f A^{N-1} B & A_f A^{N-2} B & \dots & A_f P \end{bmatrix}, \quad E_0 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -A_x \\ -A_x A \\ -A_x A^2 \\ \vdots \\ -A_f A^N \end{bmatrix}, \quad w_0 = \begin{bmatrix} b_u \\ b_u \\ \vdots \\ b_u \\ b_x \\ b_x \\ b_x \\ \vdots \\ b_f \end{bmatrix}$$

Table of Contents

4. Constrained Optimal Control: 2-Norm

4.1 Problem Formulation

4.2 Construction of the QP with substitution

4.3 Construction of the QP without substitution

Construction of the QP without substitution

To obtain the QP problem

$$\begin{aligned} J_0^*(x(0)) = \min_{U_0} \quad & [U_0' \ x(0)'] \begin{bmatrix} H & F' \\ F & Q \end{bmatrix} [U_0' \ x(0)']' \\ \text{subj. to} \quad & G_0 U_0 \leq w_0 + E_0 x(0) \end{aligned}$$

we have substituted the state equations

$$x_{k+1} = Ax_k + Bu_k$$

into the state constraints $x_k \in \mathcal{X}$.

It is often more efficient to keep the explicit equality constraints.

Construction of the QP without substitution

We transform the CFTOC problem into the QP problem

$$\begin{aligned}
 J_0^*(x(0)) = \min_z & [z' \ x(0)']' \begin{bmatrix} \bar{H} & 0 \\ 0 & Q \end{bmatrix} [z' \ x(0)']' \\
 \text{subj. to } G_{0,\text{in}} z & \leq w_{0,\text{in}} + E_{0,\text{in}} x(0) \\
 G_{0,\text{eq}} z & = E_{0,\text{eq}} x(0)
 \end{aligned}$$

- Define variable:

$$z = [x'_1 \ \dots \ x'_N \ \ u'_0 \ \dots \ u'_{N-1}]'$$

- Equalities from system dynamics $x_{k+1} = Ax_k + Bu_k$:

$$G_{0,\text{eq}} = \begin{bmatrix} I & & & & & \\ -A & I & & & & \\ & -A & I & & & \\ & & \ddots & \ddots & & \\ & & & -A & I & \end{bmatrix}, E_{0,\text{eq}} = \begin{bmatrix} A \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Construction of the QP without substitution

If \mathcal{X} , \mathcal{U} and \mathcal{X}_f are given by:

$$\mathcal{X} = \{x \mid A_x x \leq b_x\} \quad \mathcal{U} = \{u \mid A_u u \leq b_u\} \quad \mathcal{X}_f = \{x \mid A_f x \leq b_f\}$$

Then matrices $G_{0,\text{in}}$, $w_{0,\text{in}}$ and $E_{0,\text{in}}$ are:

$$G_{0,\text{in}} = \begin{bmatrix} 0 & \dots & 0 & 0 & \dots & 0 \\ A_x & & & 0 & & \\ & A_x & & & 0 & \\ & & \ddots & & & \\ & & & A_x & & 0 \\ & & & & A_f & 0 \\ 0 & & & & & A_u \\ & 0 & & & & A_u \\ & & \ddots & & & \\ & & & 0 & & A_u \\ & & & & A_u & A_u \end{bmatrix} \quad w_{0,\text{in}} = \begin{bmatrix} b_x \\ b_x \\ b_x \\ \vdots \\ b_x \\ b_f \\ \bar{b}_u \\ b_u \\ \vdots \\ b_u \\ b_u \end{bmatrix}$$

$$E_{0,\text{in}} = [-A'_x \ 0 \ \cdots \ 0]'$$

Construction of the QP without substitution

Build cost function from MPC cost $x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k$

$$\bar{H} = \begin{bmatrix} Q & & & \\ \ddots & & & \\ & Q & & \\ & & P & \\ \hline & & & \bar{R} \\ & & & \ddots \\ & & & R \end{bmatrix}$$

Outline

general case, nonlinear sys, cost, constraints

1. Optimal Control

linear sys, quadratic cost, no constraint

2. Linear Quadratic Optimal Control

linear sys, a quadratic cost

3. Constrained Linear Optimal Control

linear constraints

4. Constrained Optimal Control: 2-Norm

5. Constrained Optimal Control: 1-Norm and ∞ -Norm

5.1 Problem Formulation

5.2 Construction of the LP with substitution

linear sys, linear constraint

piecewise linear

convex cost function

Table of Contents

5. Constrained Optimal Control: 1-Norm and ∞ -Norm

5.1 Problem Formulation

5.2 Construction of the LP with substitution

Problem Formulation

Piece-wise linear cost function

p can be 1 or ∞

$$J_0(x(0), U_0) := \|Px_N\|_p + \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p \quad (7)$$

with $p = 1$ or $p = \infty$, P, Q, R full column rank matrices

Constrained Finite Time Optimal Control Problem (CFTOC)

$$\begin{aligned} J_0^*(x(0)) = \min_{U_0} \quad & J_0(x(0), U_0) \\ \text{subj. to} \quad & x_{k+1} = Ax_k + Bu_k, \quad k = 0, \dots, N-1 \\ & x_k \in \mathcal{X}, \quad u_k \in \mathcal{U}, \quad k = 0, \dots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(0) \end{aligned} \quad (8)$$

N is the time horizon and $\mathcal{X}, \mathcal{U}, \mathcal{X}_f$ are polyhedral regions.

Table of Contents

5. Constrained Optimal Control: 1-Norm and ∞ -Norm

5.1 Problem Formulation

5.2 Construction of the LP with substitution

batch sol to CFTOC of type (7) (8) can be computed with
linear programs

Construction of the LP with substitution

The problem results in the following standard LP

$$p = \inf$$

$$\begin{aligned} & \min_{z_0} && c'_0 z_0 \\ & \text{subj. to} && \bar{G}_0 z_0 \leq \bar{w}_0 + \bar{S}_0 x(0) \end{aligned}$$

where $z_0 := \{\varepsilon_0^x, \dots, \varepsilon_N^x, \varepsilon_0^u, \dots, \varepsilon_{N-1}^u, u'_0, \dots, u'_{N-1}\} \in \mathbb{R}^s$,

$s \triangleq (m+1)N + N + 1$ and

$$J_0(x(0), U_0) := \|Px_N\|_p + \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p$$

$$\bar{G}_0 = \begin{bmatrix} G_\varepsilon & 0 \\ 0 & G_0 \end{bmatrix}, \quad \bar{S}_0 = \begin{bmatrix} S_\varepsilon \\ S_0 \end{bmatrix}, \quad \bar{w}_0 = \begin{bmatrix} w_\varepsilon \\ w_0 \end{bmatrix}$$

For a given $x(0)$ U_0^* can be obtained via an LP solver (the 1-norm case is similar).

Construction of the LP with substitution - details

Recall that the ∞ -norm problem can be equivalently formulated as

$$\begin{aligned}
 \min_{z_0} \quad & \varepsilon_0^x + \dots + \varepsilon_N^x + \varepsilon_0^u + \dots + \varepsilon_{N-1}^u \\
 \text{subj. to} \quad & -\mathbf{1}_n \varepsilon_k^x \leq \pm Q \left[A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j} \right], \\
 & -\mathbf{1}_r \varepsilon_N^x \leq \pm P \left[A^N x_0 + \sum_{j=0}^{N-1} A^j B u_{N-1-j} \right], \\
 & -\mathbf{1}_m \varepsilon_k^u \leq \pm R u_k, \\
 & A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j} \in \mathcal{X}, \quad u_k \in \mathcal{U}, \\
 & A^N x_0 + \sum_{j=0}^{N-1} A^j B u_{N-1-j} \in \mathcal{X}_f, \\
 & k = 0, \dots, N-1 \\
 & x_0 = x(0)
 \end{aligned}$$

midterm: main idea
solution for linear, unconstrained systems

Unconstrained and Constrained Optimal Control Dynamic Programming

F. Borrelli, M. Morari, C. Jones

UC Berkeley

ETH Zürich

EPFL Lausanne

Fall Semester 2020

Outline

1. Introduction to the Concepts of Feedback and Policy
2. Recursive Approach

Summary From Batch Approach

- The solution to CFTOC problems using batch approach is an **Open-Loop** solution (always)
- For the case of unconstrained linear systems with quadratic cost, the solution can be computed analytically

$$u_1^* =$$

$$u_0^* = K_0 x(0), K_1 x(0), \dots, u_{N-1}^* = K_{N-1} x(0)$$

- For all other cases, the solution requires a optimization solver

sq of opt $U_0^* = u_0^*, \dots, u_{N-1}^* = \text{solver}(\text{cost}(x(0)), \text{constraints}(x(0)))$

- inputs**
In the specific case of linear systems and quadratic cost with linear constraints, $\text{solver}(\text{cost}(x(0)), \text{constraints}(x(0)))$ is the QP

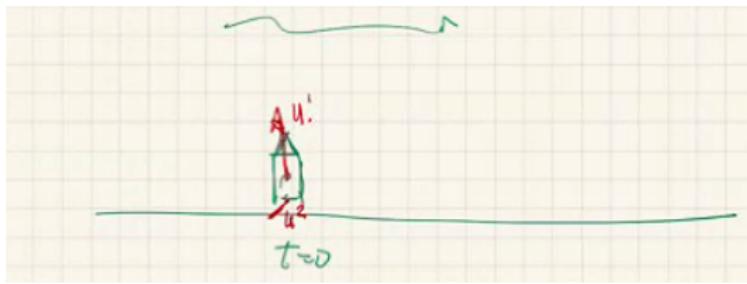
$$\begin{aligned} \min_{U_0} \quad & [U_0' \ x(0)'] \begin{bmatrix} H & F' \\ F & Y \end{bmatrix} [U_0' \ x(0)']' \\ \text{subj. to} \quad & G_0 U_0 \leq w_0 + E_0 x(0) \end{aligned}$$

Summary From Batch Approach

- Note that in the linear/unconstrained case one can first compute the **policy** K_0, K_1, \dots, K_{N-1} ("off-line")
- and then evaluate it during the implementation stage (often referred to "real-time'"):

$$\underline{u(0) = u_0^* = K_0 x(0), \dots, u(N-1) = u_{N-1}^* = K_{N-1} x(0)}$$

- Evaluation is a **simple** matrix vector multiplication.
- In all other cases the **policy** is implicitly defined by the optimization problem `solver(cost(x(0)), constraints(x(0)))` and cannot be precomputed for all initial conditions.
- A solver provides **the value of the policy** for a given $x(0)$.



$t=0$ MEASURE $x(0)$:

$$\text{MULTIPLY } K_0 \cdot x(0) \rightarrow u_0^* = []$$

$$\text{APPLY } u(0) = u_0^*$$

$$t=L \text{ MULTIPLY } K_L \cdot x(L) \rightarrow u_L^*$$

$$\text{APPLY } u(L) = u_L^*$$

SOLVE CFTOC($N, x(0)$)

$$\text{IMPLEMENT } u(0) = u_0^*$$

$t=L$ MEASURE $x(L)$

SOLVE CFTOC($N-L, x(L)$)

Addressing Two Elements: Feedback and Policies

- Can I compute the solution to CFTOC problems as a **Closed-Loop** solution (i.e., as feedback)?
- Can I compute the solution to CFTOC problems as feedback **policies**?

dynamic programming solves both

first introduce a 'simpler' approach

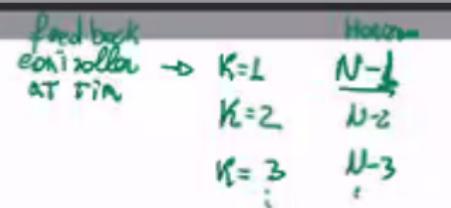
Shrinking Horizon Approach

QP with exception of simple cases requires a solver

Denote $CFTOC(N, \bar{x})$ a routine which solves a CFTOC problem with horizon N and initial condition \bar{x}

- At time $t = 0$ solve $CFTOC(N, x(0))$ to get u_0^*, \dots, u_{N-1}^*
- Implement $u(0) = u_0^*$, discard the remaining inputs
- At time $t = 1$ measure current state $x(1)$
- At time $t = 1$ solve $CFTOC(N - 1, x(1))$ to get u_1^*, \dots, u_{N-1}^* no u_0^*
- Implement $u(1) = u_1^*$, discard the remaining inputs
- ...
- ...
- At time $t = N$, **STOP**

Policy Approximation



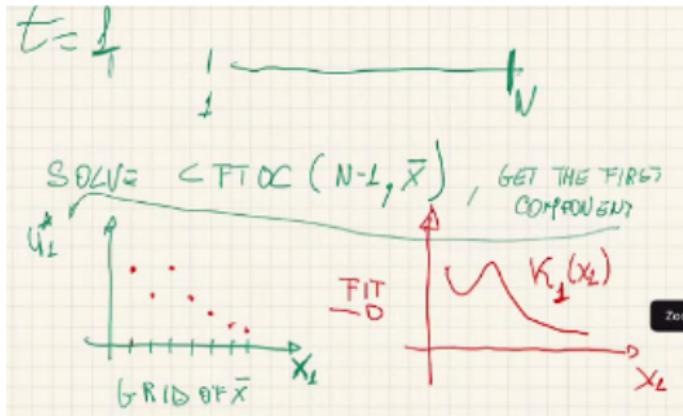
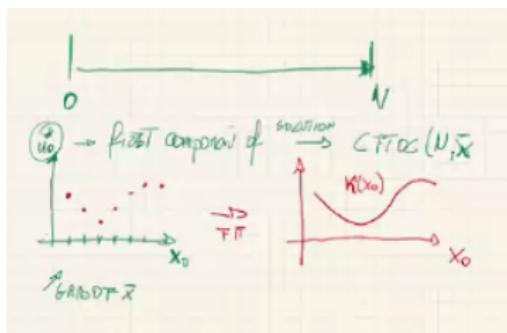
To avoid using a solver during the implementation process one can pre-compute an approximated policy as follows

- Grid the state space x in a fine set of points $\mathcal{G} = \{x^{(1)}, \dots, x^{(v)}\}$
- Solve $CFTOC(k, \bar{x})$ for all $\bar{x} \in \mathcal{G}$ to get $\{u_k^{*(1)}, \dots, u_k^{*(v)}\}$
- Use your preferred interpolation method (spline/neural networks,...) to find the map $\kappa(x)$ which best fits

$$\{x^{(1)}, \dots, x^{(v)}\} \rightarrow \{u_k^{*(1)}, \dots, u_k^{*(v)}\}$$

- The function $\kappa(x)$ is the approximated feedback control policy for time k
- The evaluation of $\kappa(\cdot)$ in real-time does not require any optimization solver (but is an approximation and might lead to infeasible inputs)





AT END , BY USING SHIFTING HORIZON + TITING I

SOLVE THE ORIGINAL CFTDC AS A SET OF TIME VARYING FEED-

BACK POLICIES

$K_0(x_0), K_1(x_1), K_2(x_2), \dots, K_{N-1}(x_{N-1})$

IN REAL TIME IMPLEMENT

$$u(t) = K_t(x(t))$$

$$t = 0, \dots, N-1$$

nonlinear, time-varying , feedback

Outline

1. Introduction to the Concepts of Feedback and Policy
2. Recursive Approach
 - 2.1 Introduction
 - 2.2 The Dynamic Programming Algorithm or recursive approach
 - 2.3 A Few Implementation Details
 - 2.4 Linear Quadratic Optimal Control: Solution via Recursive Approach
 - 2.5 Infinite Horizon Optimal Control

Table of Contents

2. Recursive Approach

2.1 Introduction

2.2 The Dynamic Programming Algorithm

2.3 A Few Implementation Details

2.4 Linear Quadratic Optimal Control: Solution via Recursive Approach

2.5 Infinite Horizon Optimal Control

Optimal Control Solution via Recursive Approach

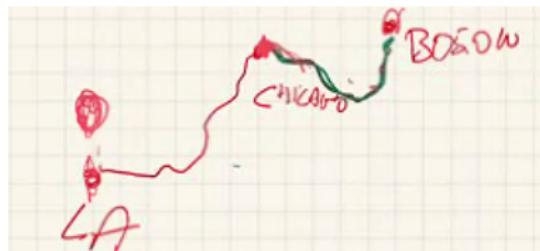
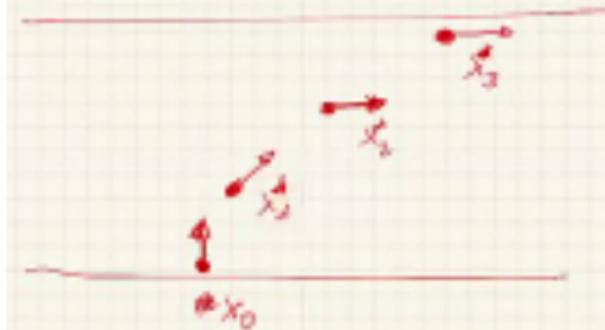
=dynamic programming

Principle of optimality

For a trajectory x_0, x_1^*, \dots, x_N^* to be optimal, the trajectory starting from any intermediate point x_j^* , i.e. $x_j^*, x_{j+1}^*, \dots, x_N^*$, $0 \leq j \leq N - 1$, must be optimal.

Suppose that the fastest route from Los Angeles to Boston passes through Chicago. The principle of optimality formalizes the obvious fact that the Chicago to Boston portion of the route is also the fastest route for a trip that starts from Chicago and ends in Boston.

$N=3$



General Problem Formulation

Recall the **Constrained Finite Time Optimal Control (CFTOC)** problem.

$$\begin{aligned}
 J_{0 \rightarrow N}^*(x_0) = & \min_{U_{0 \rightarrow N}} J_{0 \rightarrow N}(x_0, U_{0 \rightarrow N}) \\
 \text{subj. to} & x_{k+1} = g(x_k, u_k), \quad k = 0, \dots, N-1 \\
 & h(x_k, u_k) \leq 0, \quad k = 0, \dots, N-1 \\
 & x_N \in \mathcal{X}_f \\
 & x_0 = x(0)
 \end{aligned}$$

- $\mathcal{X}_f \subseteq \mathbb{R}^n$ is a *terminal region*,
- $\mathcal{X}_{0 \rightarrow N} \subseteq \mathbb{R}^n$ is the set of feasible initial conditions $x(0)$ 
- the optimal cost $J_{0 \rightarrow N}^*(x_0)$ is called *value function*,
- assume that there exists a minimum
- denote by $U_{0 \rightarrow N}^*$ one of the minima

Solution via Recursive Approach

Principle of optimality

For a trajectory x_0, x_1^*, \dots, x_N^* to be optimal, the trajectory starting from any intermediate point x_j^* , i.e. $x_j^*, x_{j+1}^*, \dots, x_N^*$, $0 \leq j \leq N - 1$, must be optimal.

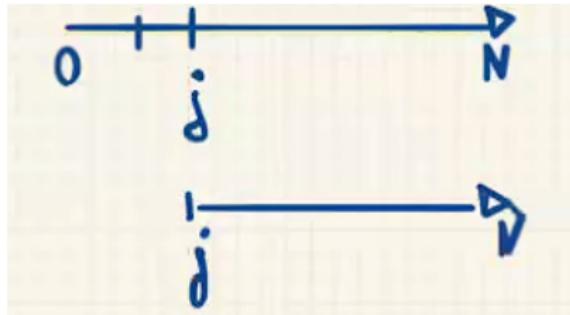
Define the cost from j to N

$$\underline{J}_{j \rightarrow N}(x_j, u_j, u_{j+1}, \dots, u_{N-1}) := p(x_N) + \sum_{k=j}^{N-1} q(x_k, u_k),$$

also called the j -step cost-to-go. Then the *optimal cost-to-go* $J_{j \rightarrow N}^*$ is

$$\begin{aligned} J_{j \rightarrow N}^*(x_j) := & \min_{u_j, u_{j+1}, \dots, u_{N-1}} J_{j \rightarrow N}(x_j, u_j, u_{j+1}, \dots, u_{N-1}) \\ \text{subj. to } & x_{k+1} = g(x_k, u_k), \quad k = j, \dots, N-1 \\ & h(x_k, u_k) \leq 0, \quad k = j, \dots, N-1 \\ & x_N \in \mathcal{X}_f \end{aligned} \quad (1)$$

Note that $J_{j \rightarrow N}^*(x_j)$ depends only on the initial state x_j .



$j=N:$

$$\mathcal{J}_{N \rightarrow N}(x_N) = p(x_N)$$

$$x_{N \rightarrow N} = x_f$$

INITIALIZATION

$j=N:$

$$\mathcal{J}_{N \rightarrow N}(x_N) = p(x_N)$$

$$x_{N \rightarrow N} = x_f$$

INITIALIZATION

idea: solve CFTOC going back, one step at time

SKETCH OF EVOLUTION IN 2D ($x \in \mathbb{R}^2$)

SET OF STATES

X_{N-1} WHERE $N-1 \rightarrow N$
FEASIBLE

$N-1$

N



$X_{0 \rightarrow N}$



$X_{N-1 \rightarrow N}$

\bullet
 x_f

$\mathcal{D}_{0 \rightarrow N}(x_0)$



$\mathcal{D}_{N-1 \rightarrow N}(x_{N-1})$



$p(x)$
 x

$X_{0 \rightarrow N}(x_0)$

$X_{N-1 \rightarrow N}(x_{N-1})$

$\mathcal{D}_{N \rightarrow N}(x_N)$

Solution via Recursive Approach

By the ***principle of optimality*** the cost $J_{j-1 \rightarrow N}^*$ can be found by solving

$$\begin{aligned}
 J_{j-1 \rightarrow N}^*(x_{j-1}) = & \min_{u_{j-1}} \underbrace{q(x_{j-1}, u_{j-1})}_{\text{stage cost}} + \underbrace{J_{j \rightarrow N}^*(x_j)}_{\text{optimal cost-to-go}} \\
 \text{subj. to } & x_j = g(x_{j-1}, u_{j-1}) \\
 & h(x_{j-1}, u_{j-1}) \leq 0 \\
 & x_j \in \mathcal{X}_{j \rightarrow N}.
 \end{aligned} \tag{2}$$

Note that

- the only decision variable is u_{j-1} ,
- the inputs u_j^*, \dots, u_{N-1}^* have already been selected optimally to yield the optimal cost-to-go $J_{j \rightarrow N}^*(x_j)$.
- in $J_{j \rightarrow N}^*(x_j)$, the state x_j can be replaced by $g(x_{j-1}, u_{j-1})$
- The set $\mathcal{X}_{j \rightarrow N}$ is the set of states x_j for which (1) is feasible. We will study these sets later in this class.

Table of Contents

2. Recursive Approach

2.1 Introduction

2.2 The Dynamic Programming Algorithm

2.3 A Few Implementation Details

2.4 Linear Quadratic Optimal Control: Solution via Recursive Approach

2.5 Infinite Horizon Optimal Control

Solution via Recursive Approach

The following (recursive) ***dynamic programming*** algorithm can be used to compute the optimal control law.

$$\begin{aligned} J_{N \rightarrow N}^*(x_N) &= p(x_N) \\ \mathcal{X}_{N \rightarrow N} &= \mathcal{X}_f, \end{aligned}$$

cost to go

$$\begin{aligned} J_{N-1 \rightarrow N}^*(x_{N-1}) &= \min_{u_{N-1}} q(x_{N-1}, u_{N-1}) + J_{N \rightarrow N}^*(g(x_{N-1}, u_{N-1})) \\ \text{subj. to } & h(x_{N-1}, u_{N-1}) \leq 0, \\ & g(x_{N-1}, u_{N-1}) \in \mathcal{X}_{N \rightarrow N} \end{aligned}$$

\vdots

$$\begin{aligned} J_{0 \rightarrow N}^*(x_0) &= \min_{u_0} q(x_0, u_0) + J_{1 \rightarrow N}^*(g(x_0, u_0)) \\ \text{subj. to } & h(x_0, u_0) \leq 0, \\ & g(x_0, u_0) \in \mathcal{X}_{1 \rightarrow N} \\ & x_0 = x(0). \end{aligned}$$

Table of Contents

2. Recursive Approach

2.1 Introduction

2.2 The Dynamic Programming Algorithm

2.3 A Few Implementation Details

2.4 Linear Quadratic Optimal Control: Solution via Recursive Approach

2.5 Infinite Horizon Optimal Control

Solution via Recursive Approach: Comments

- DP algorithm is appealing because at each step j only u_j is computed.
- Need to construct the optimal cost-to-go $J_{N-j}^*(x_j)$, a *function* defined over $\mathcal{X}_{j \rightarrow N}$.
- In a few special cases we know the type of function and we can find it efficiently.
- “brute force” approach. Construct $J_{j-1 \rightarrow N}$ by gridding the set $\mathcal{X}_{j-1 \rightarrow N}$ and computing the optimal cost-to-go function on each grid point.
- A nonlinear *feedback* (time varying) control law is implicitly defined:

$$\begin{aligned} u_j^*(x_j) = & \arg \min_{u_j} q(x_j, u_j) + J_{j+1 \rightarrow N}^*(g(x_j, u_j)) \\ \text{subj. to } & h(x_j, u_j) \leq 0, \\ & g(x_j, u_j) \in \mathcal{X}_{j+1 \rightarrow N} \end{aligned}$$

Notation

For the sake of simplicity we will use the following shorter notation

set of state

$$J_j^*(x_j) := J_{j \rightarrow N}^*(x_j), \quad j = 0, \dots, N$$

$$J_\infty^*(x_0) := J_{0 \rightarrow \infty}^*(x_0)$$

$$\mathcal{X}_j := \mathcal{X}_{j \rightarrow N}, \quad j = 0, \dots, N$$

$$\mathcal{X}_\infty := \mathcal{X}_{0 \rightarrow \infty}$$

$$U_0 := U_{0 \rightarrow N}$$

->N is removed

Table of Contents

2. Recursive Approach

2.1 Introduction

2.2 The Dynamic Programming Algorithm

2.3 A Few Implementation Details

2.4 Linear Quadratic Optimal Control: Solution via Recursive Approach

2.5 Infinite Horizon Optimal Control

Recall: Finite Horizon LQR

goal: $x = 0, u = 0$

linear quad regulator

- **Goal:** Find a sequence of inputs $U_0 \triangleq [u'_0, \dots, u'_{N-1}]'$ that minimizes the objective function

$$J_0^*(x(0)) \triangleq \min_{U_0} \quad x'_N P x_N + \sum_{k=0}^{N-1} [x'_k Q x_k + u'_k R u_k]$$

quad convex

subject to $x_{k+1} = Ax_k + Bu_k, k = 0, \dots, N-1$
 $x_0 = x(0)$ linear, no constraint

- $P \succeq 0$, with $P = P'$, is the *terminal* weight
- $Q \succeq 0$, with $Q = Q'$, is the *state* weight
- $R \succ 0$, with $R = R'$, is the *input* weight
- N is the horizon length
- Note that $x(0)$ is the current state, whereas x_0, \dots, x_N and u_0, \dots, u_{N-1} are *optimization variables* that are constrained to obey the system dynamics and the initial condition.

midterm:

PQRABN are given

1. $x(0)$ is given \rightarrow ? optimal approach

2. $x^*(2)$ is given \rightarrow ? $u^*(2)$ -: shrinking approach $\rightarrow N' = N-2$
or: compute $F^*(2)$

Final Result

- The problem is unconstrained
- Using the Dynamic Programming Algorithm we have

$$u^*(k) = F_k x(k)$$

which is a linear, time-varying state-feedback controller.

- the optimal cost-to-go $k \rightarrow N$ is

$$J_k^*(x(k)) = x(k)' P_k x(k)$$

k to N

which is a positive definite quadratic function of the state at time k

- F_k is computed by using P_{k+1}
- Each P_k is related to P_{k+1} by a recursive equation (Riccati Difference Equation)



Solution approach 2: Recursive Approach (1/8)

- Alternatively, we can use dynamic programming to solve the same problem in a recursive manner.
- Define the “ j -step optimal cost-to-go” as the *optimal* cost attainable for the step j problem:

$$J_j^*(x(j)) \triangleq \min_{u_j, \dots, u_{N-1}} x'_N P x_N + \sum_{k=j}^{N-1} [x'_k Q x_k + u'_k R u_k]$$

subject to $x_{k+1} = Ax_k + Bu_k, k = j, \dots, N-1$
 $x_j = x(j)$

This is the minimum cost attainable for the remainder of the horizon after step j

Solution approach 2: Recursive Approach (2/8)

- Consider the 1-step problem (solved at time $N - 1$)

$$J_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} \{ x'_{N-1} Q x_{N-1} + u'_{N-1} R u_{N-1} + x'_N P_N x_N \} \quad (3)$$

subject to $x_N = Ax_{N-1} + Bu_{N-1}$

$$P_N = P \quad (4)$$

where we introduced the notation P_j to express the optimal cost-to-go $x'_j P_j x_j$. In particular, $P_N = P$.

- Substituting (4) into (3)

$$\begin{aligned} J_{N-1}^*(x_{N-1}) = & \min_{u_{N-1}} \{ x'_{N-1} (A' P_N A + Q) x_{N-1} \\ & + u'_{N-1} (B' P_N B + R) u_{N-1} \\ & + 2x'_{N-1} A' P_N B u_{N-1} \} \end{aligned}$$

unconstrained quad convex -> $J' = 0$, u is var, x is not

REMEMBER

OPT. VAR IS U_{N-1}

X_{N-1} IS A PARAMETER

$$2(B^T P_N B + R) U_{N-1} + 2 B^T P_N A X_{N-1} = 0$$

$$\overset{\star}{U_{N-1}} = - \left(\bar{B}^T \bar{P}_N \bar{B} + \bar{R} \right)^{-1} \cdot \bar{B}^T \bar{P}_N \bar{A} X_{N-1}$$

$$\overset{\star}{J}_{N-1}(X_{N-1}) = X_{N-1}^T (A^T P_N A + Q) X_{N-1} +$$

$$X_{N-1}^T A^T \bar{P}_N \bar{B} (B^T P_N B + R)^{-1} (\cancel{B^T \bar{P}_N \bar{B} + R}) \cdot$$

$$(\cancel{B^T \bar{P}_N \bar{B} + R})^{-1} B^T \bar{P}_N \bar{A} \cdot X_{N-1}$$

$$- 2 X_{N-1}^T (A^T P_N A) \cdot (B^T P_N B + R)^{-1} B^T P_N A X_{N-1}$$

Solution approach 2: Recursive Approach (3/8)

- Solving again by setting the gradient to zero leads to the following optimality condition for u_{N-1}

$$2(B'P_NB + R)u_{N-1} + 2B'P_NAx_{N-1} = 0$$

Optimal 1-step input:

$$\begin{aligned} u_{N-1}^* &= -(B'P_NB + R)^{-1}B'P_NAx_{N-1} \\ &\triangleq F_{N-1}x_{N-1} \end{aligned}$$

1-step cost-to-go:

linear gain(A,B,P_N,R)

$$J_{N-1}^*(x_{N-1}) = x'_{N-1}P_{N-1}x_{N-1},$$

quadratic

where

$$P_{N-1} = A'P_NA + Q - A'P_NB(B'P_NB + R)^{-1}B'P_NA.$$

Solution approach 2: Recursive Approach (4/8)

- The recursive solution method used from here relies on Bellman's **Principle of Optimality**
- For any solution for steps 0 to N to be optimal, any solution for steps j to N with $j \geq 0$, taken from the 0 to N solution, must itself be optimal for the j -to- N problem
- Therefore we have, for any $j = 0, \dots, N$

$$\begin{aligned} J_j^*(x_j) &= \min_{u_j} \{ J_{j+1}^*(x_{j+1}) + x_j' Q x_j + u_j' R u_j \} \\ \text{s.t. } x_{j+1} &= A x_j + B u_j \end{aligned}$$

- Suppose that the fastest route from Los Angeles to Boston passes through Chicago. Then the principle of optimality formalizes the obvious fact that the Chicago to Boston portion of the route is also the fastest route for a trip that starts from Chicago and ends in Boston.

Solution approach 2: Recursive Approach (5/8)

- Now consider the 2-step problem, posed at time $N - 2$

$$J_{N-2}^*(x_{N-2}) = \min_{u_{N-1}, u_{N-2}} \left\{ \sum_{k=N-2}^{N-1} x_k' Q x_k + u_k' R u_k + x_N' P x_N \right\}$$

s.t. $x_{k+1} = Ax_k + Bu_k, \quad k = N-2, N-1$

- From the Principle of Optimality, the cost function is equivalent to

$$\begin{aligned} J_{N-2}^*(x_{N-2}) &= \min_{u_{N-2}} \{ J_{N-1}^*(x_{N-1}) && \text{cost to go} \\ &\quad + x_{N-2}' Q x_{N-2} + u_{N-2}' R u_{N-2} \} && \text{stage cost} \\ &= \min_{u_{N-2}} \{ x_{N-1}' P_{N-1} x_{N-1}' \\ &\quad + x_{N-2}' Q x_{N-2} + u_{N-2}' R u_{N-2} \} \end{aligned}$$

Solution approach 2: Recursive Approach (6/8)

- As with 1-step solution, solve by setting the gradient with respect to u_{N-2} to zero

Optimal 2-step input

similar with u^*_{N-1} , replace the P_N with P_{N-1}

$$\begin{aligned} u_{N-2}^* &= -(B'P_{N-1}B + R)^{-1}B'P_{N-1}Ax_{N-2} \\ &\triangleq F_{N-2}x_{N-2} \end{aligned}$$

2-step cost-to-go

$$J_{N-2}^*(x_{N-2}) = x_{N-2}'P_{N-2}x_{N-2},$$

where

$$P_{N-2} = A'P_{N-1}A + Q - A'P_{N-1}B(B'P_{N-1}B + R)^{-1}B'P_{N-1}A$$

- We now recognize the recursion for P_j and u_j^* , $j = N-1, \dots, 0$.

GO BACKWARD

$N \rightarrow INITIALIZE \quad \vec{P}_0 = \vec{P}$

$N-1 \quad COMPUTE \quad \vec{T}_{N-1} \leftarrow DATA, \vec{P}_N$
 $\vec{P}_{N-1} \leftarrow DATA, \vec{P}_0$

$N-2 \quad COMPUTE \quad \vec{T}_{N-2} \leftarrow DATA, \vec{P}_{N-1}$
 $\vec{P}_{N-2} \leftarrow DATA, \vec{P}_{N-2}$

:

:

$\emptyset \quad COMPUTE \quad \vec{T}_0 \leftarrow DATA \vec{P}$

Solution approach 2: Recursive Approach (7/8)

- We can obtain the solution for any given time step k in the horizon

$$\begin{aligned} u^*(k) &= -(B'P_{k+1}B + R)^{-1}B'P_{k+1}Ax(k) \\ &\triangleq F_kx(k) \quad \text{for } k = 0, \dots, N-1 \end{aligned}$$

where we can find any P_k by recursive evaluation from $P_N = P$, using

$$P_k = A'P_{k+1}A + Q - A'P_{k+1}B(B'P_{k+1}B + R)^{-1}B'P_{k+1}A \quad (5)$$

This is called the *Discrete Time Riccati Equation* or *Riccati Difference Equation (RDE)*.

- Evaluating down to P_0 , we obtain the N -step cost-to-go

$$J_0^*(x(0)) = x(0)'P_0x(0)$$

Solution approach 2: Recursive Approach (8/8)

Summary

- From the Principle of Optimality, the optimal control policy for any step j is then given by

$$u^*(k) = -(B'P_{k+1}B + R)^{-1}B'P_{k+1}Ax(k) = F_kx(k)$$

and the optimal cost-to-go is

$$J_k^*(x(k)) = x'_k P_k x(k)$$

- Each P_k is related to P_{k+1} by the Riccati Difference Equation

$$P_k = A'P_{k+1}A + Q - A'P_{k+1}B(B'P_{k+1}B + R)^{-1}B'P_{k+1}A,$$

which can be initialized with $P_N = P$, the given terminal weight

Comparison of Batch and Recursive Approaches (1/2)

- Fundamental difference: Batch optimization returns a sequence $U_0^*(x(0))$ of numeric values depending only on the initial state $x(0)$, while dynamic programming yields feedback policies $u^*(k) = F_k x(k)$, $k = 0, \dots, N - 1$ depending on each $x(k)$.
- If the state evolves exactly as modelled, then the sequences of control actions obtained from the two approaches are identical.
- The recursive solution should be more robust to disturbances and model errors, because if the future states later deviate from their predicted values, the exact optimal input can still be computed.
- The Recursive Approach is computationally more attractive because it breaks the problem down into single-step problems. For large horizon length, the Hessian H in the Batch Approach, which must be inverted, becomes very large.

Comparison of Batch and Recursive Approaches (2/2)

- Without any modification, both solution methods will break down when inequality constraints on x_k or u_k are added.
- The Batch Approach is far easier to adapt than the Recursive Approach when constraints are present: just perform a constrained minimization for the current state.
- Doing this at *every* time step within the time available, and then using only the first input from the resulting sequence, amounts to *receding horizon control*.

Table of Contents

2. Recursive Approach

- 2.1 Introduction
- 2.2 The Dynamic Programming Algorithm
- 2.3 A Few Implementation Details
- 2.4 Linear Quadratic Optimal Control: Solution via Recursive Approach
- 2.5 Infinite Horizon Optimal Control

Infinite Horizon Control Problem: Optimal Sol'n (1/2)

- In some cases we may want to solve the same problem with an infinite horizon:

$$J_{\infty}(x(0)) = \min_{u(\cdot)} \left\{ \sum_{k=0}^{\infty} [x'_k Q x_k + u'_k R u_k] \right\}$$

subject to $x_{k+1} = Ax_k + Bu_k, \quad k = 0, 1, 2, \dots, \infty,$
 $x_0 = x(0)$

- As with the Dynamic Programming approach, the optimal input is of the form

$$u^*(k) = -(B' P_{\infty} B + R)^{-1} B' P_{\infty} A x(k) \triangleq F_{\infty} x(k)$$

time invariant gain

and the infinite-horizon cost-to-go is

$$J_{\infty}(x(k)) = x(k)' P_{\infty} x(k).$$

Infinite Horizon Control Problem: Optimal Sol'n (2/2)

- The matrix P_∞ comes from an infinite recursion of the RDE, from a point infinitely far into the future.
- Assuming the RDE does converge to some constant matrix P_∞ , it must satisfy the following (from (5), with $P_k = P_{k+1} = P_\infty$)

$$P_\infty = A'P_\infty A + Q - A'P_\infty B(B'P_\infty B + R)^{-1}B'P_\infty A,$$

which is called the **Algebraic Riccati Equation (ARE)**

- The constant feedback matrix F_∞ is referred to as the asymptotic form of the **Linear Quadratic Regulator (LQR)**.
- In fact, if (A, B) is controllable and (Q, A) is observable, then the RDE (initialized with Q at $k = \infty$ and solved for $k \searrow 0$) converges to the unique positive definite solution P_∞ of the ARE.

Stability of Infinite-Horizon LQR

- In addition, the closed-loop system with $u(k) = F_\infty x(k)$ is guaranteed to be asymptotically stable, under the stabilizability and detectability assumptions of the previous slide.
- The latter statement can be proven by substituting the control law $u(k) = F_\infty x(k)$ into $x(k+1) = Ax(k) + Bu(k)$, and then examining the properties of the system

always stable ($Q, R > 0$)

$$x(k+1) = (A + BF_\infty)x(k). \quad (6)$$

- The asymptotic stability of (6) can be proven by showing that the infinite horizon cost $J_\infty^*(x(k)) = x(k)'P_\infty x(k)$ is actually a Lyapunov function for the system, i.e. $J_\infty^*(x(k)) > 0$, $\forall k \neq 0$, $J_\infty^*(0) = 0$, and $J_\infty^*(x(k+1)) < J_\infty^*(x(k))$, for any $x(k)$. This implies that

$$\lim_{k \rightarrow \infty} x(k) = 0.$$

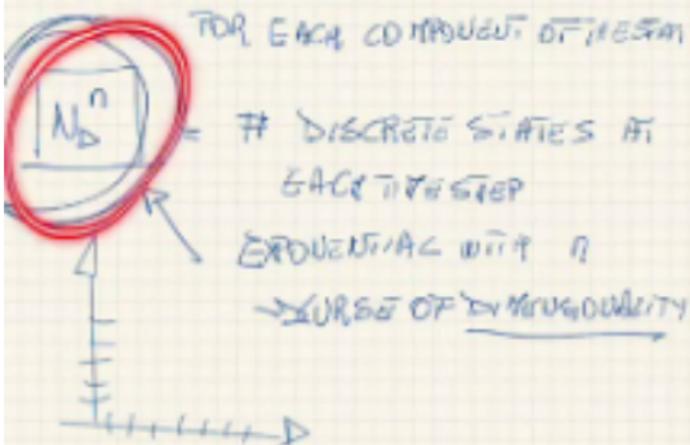
NEED TO GRID STATE SPACE

Say $n = \#$ states.

N = HORIZON

N_b : # DISCRETIZED POINT

FOR EACH COMPONENT OF STATE



1. why infinite
2. DP, batch cannot be used
3. for LQR, unconstrained, solution exists under mild conditions on ABQR
4. solution is obtained by solving ARE
5. solution always leads to stable close loop system

Numerical Optimization Methods

Alexander Domahidi, Stefan Richter, Fabian Ullmann,
Melanie Zeilinger, Francesco Borrelli, Colin Jones and Manfred Morari

UC Berkeley,
Institut für Automatik
ETH Zürich

Fall Semester 2016

Table of Contents

- 1. Introduction
- 2. Unconstrained Optimization
 - 2.1 Gradient Methods
 - 2.2 Newton's Method
- 3. Constrained Optimization
 - 3.1 Gradient Methods
 - 3.2 Interior Point Methods
 - 3.3 Active Set Methods
- 4. Software
 - 4.1 Modeling
 - 4.2 Solvers for Desktop PCs
 - 4.3 Solvers for Embedded Platforms

Table of Contents

1. Introduction

2. Unconstrained Optimization

2.1 Gradient Methods

2.2 Newton's Method

3. Constrained Optimization

3.1 Gradient Methods

3.2 Interior Point Methods

3.3 Active Set Methods

4. Software

4.1 Modeling

4.2 Solvers for Desktop PCs

4.3 Solvers for Embedded Platforms

Numerical Optimization Methods

In all but the simplest cases, an analytical solution to,

$$\begin{aligned} z^* \in \operatorname{argmin}_{\text{s.t.}} & f(z) \\ & z \in S \end{aligned}$$

cannot be obtained.

⇒ Numerical computation of a solution that is “good enough” by

Iterative optimization methods:

Given an initial guess z^0 , produce a sequence of iterates

$$z^{k+1} = \Psi(z^k, f, S), \quad k = 0, 1, \dots, k_{max}$$

such that

$$|f(z^{k_{max}}) - f(z^*)| \leq \epsilon \quad \text{and} \quad \operatorname{dist}(z^{k_{max}}, S) \leq \delta,$$

where ϵ and δ are user defined tolerances.

Numerical Optimization Methods

repeat $z^{k+1} = \Psi(z^k, f, S)$, $k = 0, 1, \dots, k_{max} - 1$
until $|f(z^{k_{max}}) - f(z^*)| \leq \epsilon$ and $\text{dist}(z^{k_{max}}, S) \leq \delta$

Important aspects of optimization algorithms:

- Convergence: is k_{max} finite for every $\epsilon, \delta > 0$?
- Convergence speed: dependence of errors $f(z^{k_{max}}) - f(z^*)$ and $\text{dist}(z^{k_{max}}, S)$ on iteration counter
- Feasibility: for some methods $\delta = 0$, but in general $\delta \neq 0$
- Numerical robustness in presence of finite precision arithmetics
- Warm-starting: can the method take advantage of z^0 being close to z^* ?
- Preconditioning: equivalence transformation of (P) into a similar problem (P') that can be solved in fewer iterations?

Table of Contents

1. Introduction

2. Unconstrained Optimization

2.1 Gradient Methods

2.2 Newton's Method

3. Constrained Optimization

3.1 Gradient Methods

3.2 Interior Point Methods

3.3 Active Set Methods

4. Software

4.1 Modeling

4.2 Solvers for Desktop PCs

4.3 Solvers for Embedded Platforms

Table of Contents

2. Unconstrained Optimization

2.1 Gradient Methods

2.2 Newton's Method

Unconstrained Optimization Using Gradient Information

[Cauchy 1847]

Goal: Solve the *unconstrained* (i.e. $S = \mathbb{R}^n$) problem

$$\text{minimize } f(z)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex and continuously differentiable.

Idea: Gradient ∇f gives direction of steepest local ascent

\Rightarrow Make steps of size h^k into anti-gradient direction $-\nabla f$:

$$z^{k+1} = z^k - h^k \nabla f(z^k) \tag{1}$$

Question: How to choose the step sizes h^k ?

A Useful Reformulation of the Gradient Update

Gradient update:

$$z^{k+1} = z^k - h^k \nabla f(z^k)$$

Idea: This update rule results from minimizing a quadratic function:

$$z^{k+1} = \arg \min_z f(z^k) + \nabla f(z^k)^T (z - z^k) + \frac{1}{2h^k} \|z - z^k\|^2$$

$$\Leftrightarrow \nabla_z \left(f(z^k) + \nabla f(z^k)^T (z - z^k) + \frac{1}{2h^k} \|z - z^k\|^2 \right) \Big|_{z=z^{k+1}} = 0$$

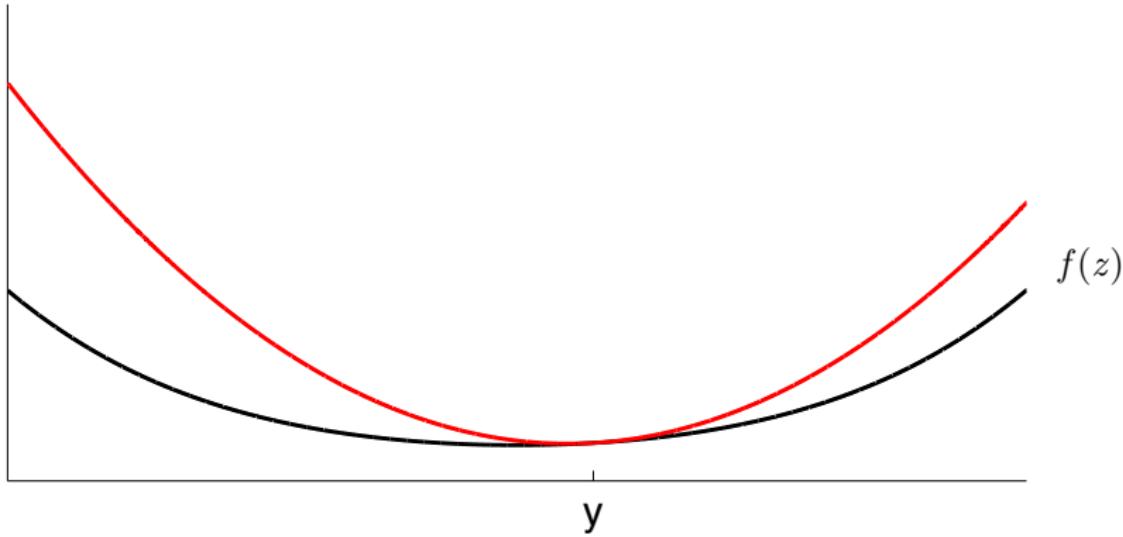
$$\Leftrightarrow \nabla f(z^k) + \frac{1}{h^k} (z^{k+1} - z^k) = 0$$

$$\Leftrightarrow z^{k+1} = z^k - h^k \nabla f(z^k)$$

L -smoothness

Assumption: f can be upper bounded by a quadratic function:

$$f(z) \leq f(y) + \nabla f(y)^T (z - y) + \frac{L}{2} \|z - y\|^2 \quad \forall z, y \in \mathbb{R}^n$$



The Gradient Method Converges with a Constant Step Size

Gradient update \Leftrightarrow minimization of quadratic function:

$$\begin{aligned} z^{k+1} &= z^k - h^k \nabla f(z^k) \\ &= \arg \min_z f(z^k) + \nabla f(z^k)^T (z - z^k) + \frac{1}{2h^k} \|z - z^k\|^2 \end{aligned}$$

L -smoothness $\Leftrightarrow f$ can be upper bounded by quadratic function:

$$f(z) \leq f(z^k) + \nabla f(z^k)^T (z - z^k) + \frac{L}{2} \|z - z^k\|^2 \triangleq \bar{f}(z, z^k)$$

Results:

- 1 Constant step size:
$$h^k = \frac{1}{L}$$
- 2 Convergence: $f(z^{k+1}) \leq \bar{f}(z^{k+1}, z^k) \leq \bar{f}(z^k, z^k) = f(z^k)$

Gradient Method - Summary

Problem: minimize $f(z)$

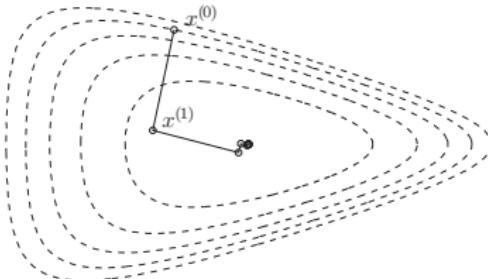
Requirements: f is L -smooth and convex

Gradient method:

Set z^0 .
Repeat $z^{k+1} = z^k - \frac{1}{L} \nabla f(z^k)$ for $k = 0, \dots, k_{max}$
until $f(z^{k_{max}}) - f(z^*) \leq \epsilon_1$ or $\|z^{k_{max}} - z^{k_{max}}\| \leq \epsilon_2$.

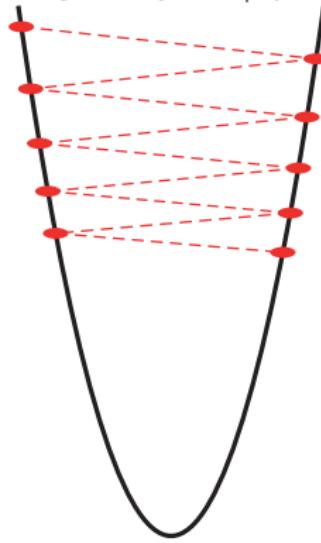
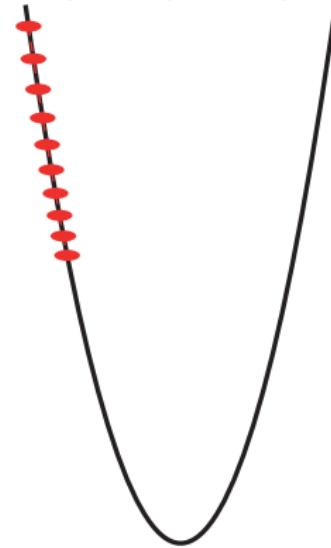
Convergence: The GM converges with $k_{max} \sim \mathcal{O}(L\|z^0 - z^*\|^2/\epsilon_1)$.

Example: $\min e^{z^1 + 3z^2 - 0.1} + e^{z^1 - 3z^2 - 0.1} + e^{-z^1 - 0.1}$



[Boyd & Vandenberghe, 2004]

The Effect of Inexact L

GM with $L + \delta$ GM with $L - \delta$ 

Inexact information on L yields slower convergence. Convergence can even be lost if $L < L_{\text{true}}/2$.

Figures by courtesy of Dr. Michel Baes, IFOR, ETH Zurich.

Table of Contents

2. Unconstrained Optimization

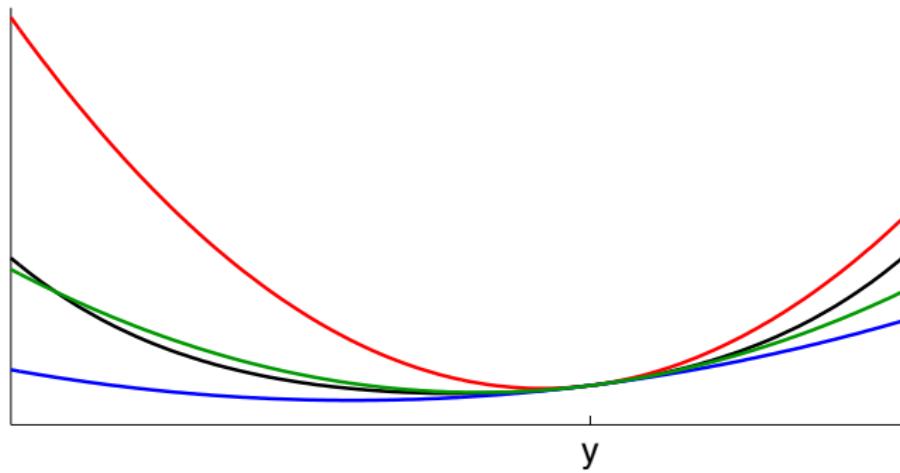
2.1 Gradient Methods

2.2 Newton's Method

Newton's Method

Idea: Minimize second-order approximation of f at point z^k :

$$z^{k+1} = \arg \min_z \underbrace{f(z^k) + \nabla f(z^k)^T (z - z^k) + \frac{1}{2}(z - z^k)^T \nabla^2 f(z^k)(z - z^k)}_{\triangleq \tilde{f}(z, z^k)}$$



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

Newton's Method

Idea: Minimize second-order approximation of f at point z^k :

$$\begin{aligned} z^{k+1} &= \arg \min_z f(z^k) + \nabla f(z^k)^T (z - z^k) + \frac{1}{2} (z - z^k)^T \nabla^2 f(z^k) (z - z^k) \\ \nabla_z \left(f(z^k) + \nabla f(z^k)^T (z - z^k) + \frac{1}{2} (z - z^k)^T \nabla^2 f(z^k) (z - z^k) \right) \Big|_{z=z^{k+1}} &= 0 \\ \Leftrightarrow \nabla f(z^k) + \nabla^2 f(z^k)(z^{k+1} - z^k) &= 0 \\ \Leftrightarrow z^{k+1} &= z^k - \underbrace{(\nabla^2 f(z^k))^{-1} \nabla f(z^k)}_{\text{Newton direction } d_N(z^k)} \end{aligned}$$

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

Idea: Use step size $h^k > 0$ such that Newton step yields descent

Newton step:
$$z^{k+1} = z^k - h^k (\nabla^2 f(z^k))^{-1} \nabla f(z^k)$$

(2)

Line Search

$$\text{Newton step: } z^{k+1} = z^k + h^k d_N(z^k)$$

Problem: Find $h^k > 0$ s.t. $f(z^k + h^k d_N(z^k)) < f(z^k)$

Line search (LS) methods:

- *Exact:* Compute best h^k :

$$h^{k*} = \arg \min_{h>0} f(z^k + h^k d_N(z^k))$$

Optimization in 1 variable \rightarrow solve by bisection

Time consuming (requires many evaluations of f)

- *Inexact:* Find h^k that decreases f by some per cent. Example:

Backtracking¹ line search. For $\alpha \in (0, 0.5)$ and $\beta \in (0, 1)$:

Initialize $h^k = 1$.

while $f(z^k + h^k d_N(z^k)) > f(z^k) + \alpha h^k \nabla f(z^k)^T d_N(z^k)$ do $h^k \leftarrow \beta h^k$

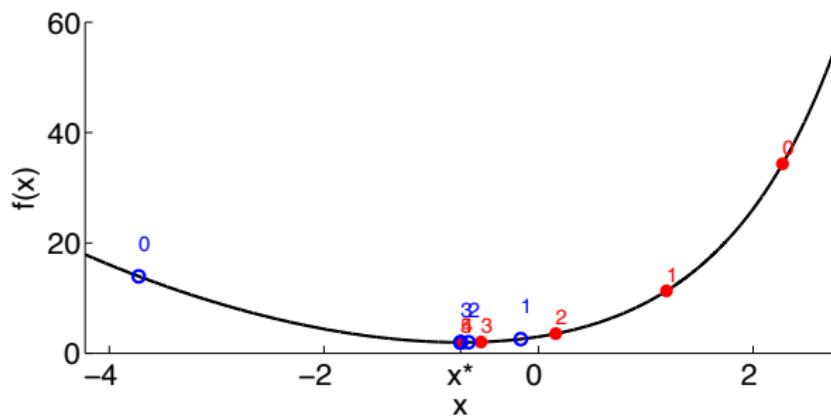
¹More details in e.g. [Boyd & Vandenberghe, Convex Optimization, 2004]

Newton's Method: Example

(Without line search)

Minimizing $f(z) = 3e^z + z^2$

Initial values: $z^{0^1} = z^* + 3$, $z^{0^2} = z^* - 3$, $z^* = -0.726$



k	$ z^{k^1} - z^* $	$ z^{k^2} - z^* $
0	3.0e+00	3.0e+00
1	1.9e+00	5.6e-01
2	8.9e-01	7.4e-02
3	1.9e-01	1.2e-03
4	7.9e-03	2.8e-07
5	1.3e-05	1.7e-14
6	3.7e-11	3.3e-16
7	3.3e-16	3.3e-16

z_0^1 is outside quadratic convergence zone

Table of Contents

1. Introduction

2. Unconstrained Optimization

2.1 Gradient Methods

2.2 Newton's Method

3. Constrained Optimization

3.1 Gradient Methods

3.2 Interior Point Methods

3.3 Active Set Methods

4. Software

4.1 Modeling

4.2 Solvers for Desktop PCs

4.3 Solvers for Embedded Platforms

Table of Contents

3. Constrained Optimization

3.1 Gradient Methods

3.2 Interior Point Methods

3.3 Active Set Methods

Constrained Minimization Using Gradient Methods

Consider the following constrained convex optimization problem:

$$\begin{aligned} & \text{minimize} && f(z) \\ & \text{subject to} && z \in S \end{aligned} \tag{P}$$

where S is convex and f is convex and L -smooth.

Recap: We can solve the *unconstrained* problem $S \equiv \mathbb{R}^n$ efficiently by the gradient method:

Set z^0 .
Repeat $z^{k+1} = z^k - \frac{1}{L} \nabla f(z^k)$ for $k = 0, \dots, k_{max} - 1$
until $f(z^{k_{max}}) - f(z^*) \leq \epsilon_1$ or $\|z^{k_{max}} - z^{k_{max}-1}\| \leq \epsilon_2$.

Question: How to handle constraints?

A Useful Reformulation of the Gradient Update

- **Unconstrained case:** Gradient update results from minimizing a quadratic function:

$$\begin{aligned} z^{k+1} &= \arg \min_z f(z^k) + \nabla f(z^k)^T (z - z^k) + \frac{1}{2h^k} \|z - z^k\|^2 \\ \Leftrightarrow z^{k+1} &= z^k - h^k \nabla f(z^k) \end{aligned}$$

- **Constrained case:** Incorporate constraints in minimization:

$$\begin{aligned} z^{k+1} &= \arg \min_{z \in S} f(z^k) + \nabla f(z^k)^T (z - z^k) + \frac{1}{2h^k} \|z - z^k\|^2 \\ \Leftrightarrow z^{k+1} &= \boxed{\pi_S(z^k - h^k \nabla f(z^k))} \end{aligned}$$

where π_S is a **projection**:

$$\begin{aligned} \pi_S(y) &\triangleq \arg \min_z \frac{1}{2} \|z - y\|_2^2 \\ \text{s.t. } z &\in S \end{aligned}$$

(Proof by equivalence of optimality conditions and strong convexity.)

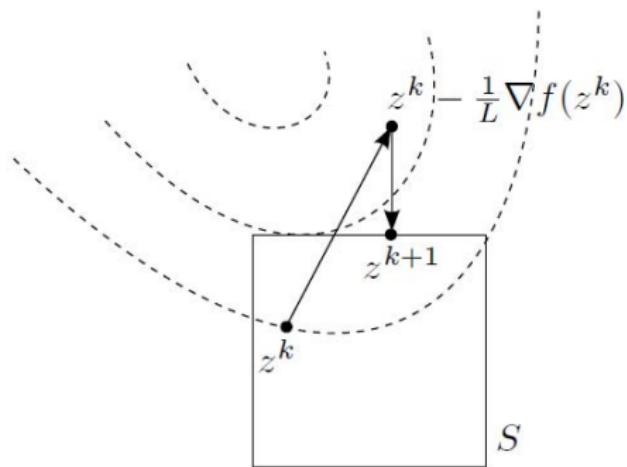
Gradient Methods for Constrained Optimization

Gradient method

Set z^0 .

Repeat $z^{k+1} = \pi_S(z^k - \frac{1}{L} \nabla f(z^k))$ $k = 0, \dots, k_{max}$

until $f(z^{k_{max}}) - f(z^*) \leq \epsilon_1$ or $\|z^{k_{max}} - z^{k_{max}-1}\| \leq \epsilon_2$.



- If the projection is “easy” to compute: efficient algorithm

Sets that Allow Inexpensive Projections

Euclidean norm projection operators of closed convex sets in \mathbb{R}^n :

Set Name	Set Definition	Projection
hyperplane	$\mathbb{Q} = \left\{ z \in \mathbb{R}^n \mid a^T z = b \right\}$ with $(a, b) \in \mathbb{R}^n \times \mathbb{R}, a \neq 0$	$\pi_{\mathbb{Q}}(z) = z + \frac{b - a^T z}{\ a\ ^2} a$
affine set	$\mathbb{Q} = \left\{ z \in \mathbb{R}^n \mid Az = b \right\}$ with $(A, b) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m, A \neq 0$	$\pi_{\mathbb{Q}}(z) = \begin{cases} z + A^T (AA^T)^{-1} (b - Az) & \text{if rank } A = m, \\ z + A^T A^T \dagger (A^{\dagger} b - z) & \text{otherwise} \end{cases}$
halfspace	$\mathbb{Q} = \left\{ z \in \mathbb{R}^n \mid a^T z \leq b \right\}$ with $(a, b) \in \mathbb{R}^n \times \mathbb{R}, a \neq 0$	$\pi_{\mathbb{Q}}(z) = \begin{cases} z + \frac{b - a^T z}{\ a\ ^2} a & \text{if } a^T z > b, \\ z & \text{otherwise} \end{cases}$
nonnegative orthant	$\mathbb{Q} = \left\{ z \in \mathbb{R}^n \mid z \geq 0 \right\}$	$\left(\pi_{\mathbb{Q}}(z) \right)_k = \begin{cases} z_k & \text{if } z_k \geq 0, \\ 0 & \text{otherwise} \end{cases} \quad k = 1, \dots, n$
rectangle (e.g. ∞ -norm ball)	$\mathbb{Q} = \left\{ z \in \mathbb{R}^n \mid l \leq z \leq u \right\}$ with $(l, u) \in \mathbb{R}^n \times \mathbb{R}^n$	$\left(\pi_{\mathbb{Q}}(z) \right)_k = \begin{cases} l_k & \text{if } z_k < l_k, \\ z_k & \text{if } l_k \leq z_k \leq u_k, \\ u_k & \text{if } z_k > u_k \end{cases} \quad k = 1, \dots, n$

Sets that Allow Inexpensive Projections

Euclidean norm projection operators of closed convex sets in \mathbb{R}^n :

Set Name	Set Definition	Projection
2-norm ball	$\mathbb{Q} = \left\{ z \in \mathbb{R}^n \mid \ z\ \leq r \right\}$ with $r \geq 0$	$\pi_{\mathbb{Q}}(z) = \begin{cases} r \frac{z}{\ z\ } & \text{if } \ z\ > r, \\ z & \text{otherwise} \end{cases}$
simplex	$\mathbb{Q} = \left\{ z \in \mathbb{R}^n \mid z \geq 0, \sum_{k=1}^n z_k = r \right\}$ with $r \geq 0$	Easy to implement $\mathcal{O}(n^2)$ algorithm or a $\mathcal{O}(n \log n)$ algorithm based on sorting. Both algorithms compute the projection exactly.
1-norm ball	$\mathbb{Q} = \left\{ z \in \mathbb{R}^n \mid \ z\ _1 \leq r \right\}$ with $r \geq 0$	The projection on the 1-norm ball can be reduced to a projection on a simplex, so that the complexity is $\mathcal{O}(n^2)$ or $\mathcal{O}(n \log n)$, see above.
second-order cone	$\mathbb{Q} = \left\{ (z, s) \in \mathbb{R}^n \times \mathbb{R}_+ \mid \ z\ \leq s \right\}$	$\pi_{\mathbb{Q}}((z, s)) = \begin{cases} (z, s) & \text{if } \ z\ \leq s, \\ \frac{s + \ z\ }{2\ z\ } \begin{bmatrix} z \\ \ z\ \end{bmatrix} & \text{if } -\ z\ < s < \ z\ , \\ (0, 0) & \text{if } \ z\ \leq -s \end{cases}$

Table of Contents

3. Constrained Optimization

3.1 Gradient Methods

3.2 Interior Point Methods

3.3 Active Set Methods

Constrained Minimization Problem

Consider the following problem with inequality constraints

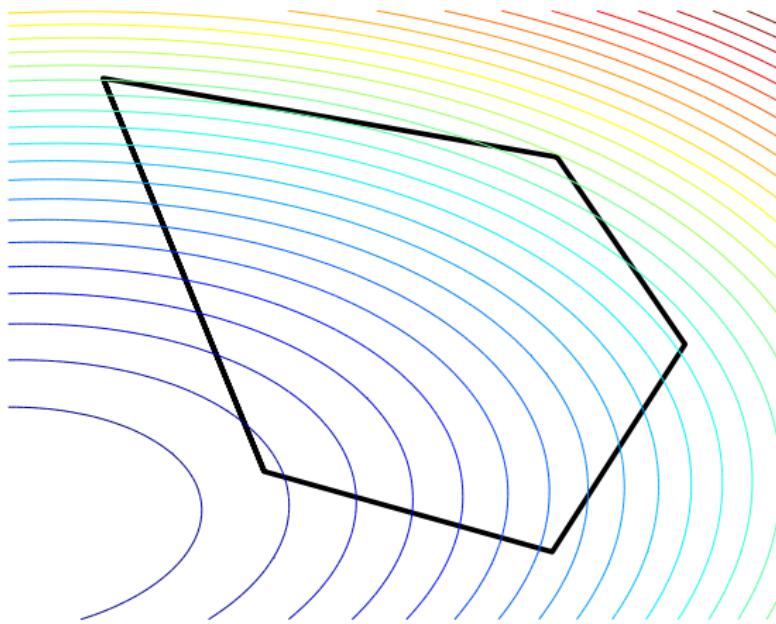
$$\begin{aligned} & \min f(z) \\ \text{s.t. } & g_i(z) \leq 0, i = 1, \dots, m \end{aligned}$$

- f, g_i convex, twice continuously differentiable
- We assume $f(z^*)$ is finite and attained
- We assume problem is strictly feasible: there exists a \tilde{z} with

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

Idea: There exist many methods for unconstrained minimization
⇒ Reformulate problem as an unconstrained problem

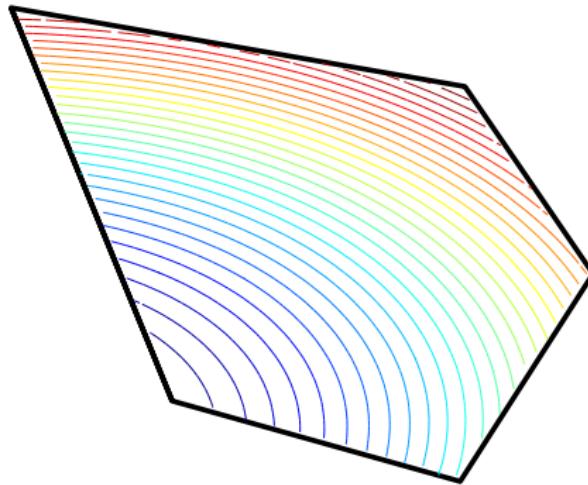
Graphical Illustration



Minimize a function over a set

Graphical Illustration

Define function as ∞ if constraints violated.



Minimize this function over \mathbb{R}^n

Barrier Method

$$\begin{array}{ll} \min & f(z) \\ \text{s.t.} & g_i(z) \leq 0, i = 1, \dots, m \end{array} \Leftrightarrow \min f(z) + \mu \phi(z)$$

Constraints have been moved to objective via *indicator function*:

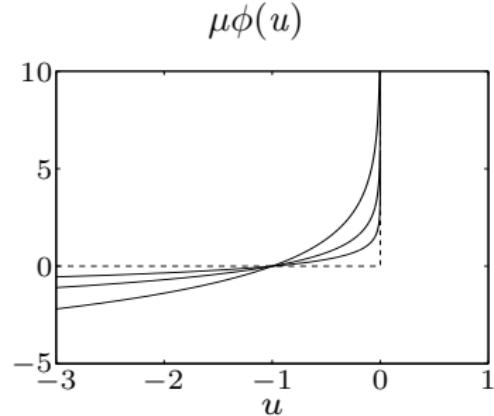
$$\phi(z) = \sum_{i=1}^m I_-(g_i(z)), \quad \mu = 1$$

where $I_-(u) = 0$ if $u \leq 0$ and $I_- = \infty$ otherwise

- Augmented cost is not differentiable
→ approximation by *logarithmic barrier*:

$$\phi(z) = - \sum_{i=1}^m \log(-g_i(z))$$

- For $\mu > 0$ smooth approximation of indicator function
- Approximation improves as $\mu \rightarrow 0$



Logarithmic Barrier Function

$$\phi(z) = - \sum_{i=1}^m \log(-g_i(z)), \quad \text{dom } \phi = \{z \mid g_1(z) < 0, \dots, g_m(z) < 0\}$$

- Convex, smooth on its domain
- $\phi(z) \rightarrow \infty$ as z approaches boundary of domain
- $\arg \min_z \phi(z)$ is called **analytic center** of the set defined by inequalities $g_1 < 0, \dots, g_m < 0$
- Twice continuously differentiable with derivatives

$$\nabla \phi(z) = \sum_{i=1}^m \frac{1}{-g_i(z)} \nabla g_i(z)$$

$$\nabla^2 \phi(z) = \sum_{i=1}^m \frac{1}{g_i(z)^2} \nabla g_i(z) \nabla g_i(z)^T + \frac{1}{-g_i(z)} \nabla^2 g_i(z)$$

Central Path

- Define $z^*(\mu)$ as the solution of

$$\min_z f(z) + \mu \phi(z)$$

(assume minimizer exists and is unique for each $\mu > 0$)

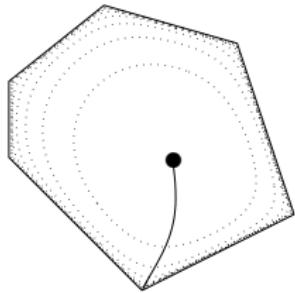
- Barrier parameter μ determines relative weight between objective and barrier
- Barrier 'traps' $z(\mu)$ in strictly feasible set
- *Central path* is defined as $\{z^*(\mu) \mid \mu > 0\}$
- For given μ can compute $z^*(\mu)$ by solving smooth unconstrained minimization problem
- Intuitively $z^*(\mu)$ converges to optimal solution as $\mu \rightarrow 0$
(easy to prove under mild conditions)

Example: Central Path for an LP

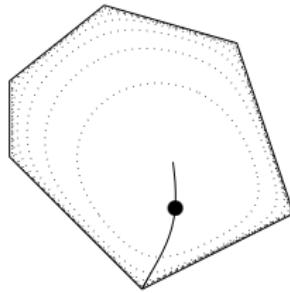
$$\begin{array}{ll} \min & c^T z \\ \text{s.t.} & a_i^T z \leq b_i, i = 1, \dots, 6 \end{array}$$

$z \in \mathbb{R}^2$, c points upward

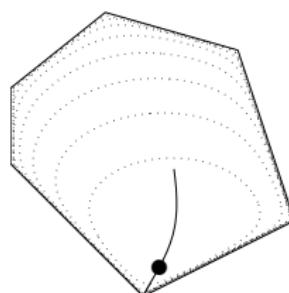
$$\mu = 1000$$



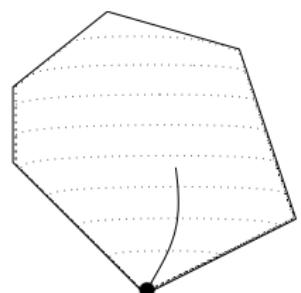
$$\mu = 1$$



$$\mu = 1/5$$



$$\mu = 1/100$$



Path-following Method

Idea: Follow central path to the optimal solution

Solve sequence of smooth unconstrained problems:

$$z^*(\mu) = \arg \min_z f(z) + \mu \phi(z)$$

- Assume current solution is on the central path $z^i = z^*(\mu^i)$
- Obtain μ^{i+1} by decreasing μ^i by some amount
- Solve for $z^*(\mu^{i+1})$ starting from $z^*(\mu^i)$ (unconstrained optimization)
- Method converges to the optimal solution, i.e., $z^i \rightarrow z^*$ for $\mu \rightarrow 0$

Table of Contents

3. Constrained Optimization

3.1 Gradient Methods

3.2 Interior Point Methods

3.3 Active Set Methods

Active Set Idea

Consider

$$\begin{aligned} \min_z \quad & f(z) \\ \text{subj. to} \quad & z \in S, \end{aligned}$$

where the feasible set $S \subset \mathbb{R}^s$ is a polyhedron, i.e. a set defined by linear equalities and inequalities, and the objective f is a linear function (*linear programming* (LP)) or a convex quadratic function (*quadratic programming* (QP)).

- Active set methods aim to identify the set of active constraints at the solution. Once this set is known, a solution to the problem can be easily identified.
- Since the number of potentially visited active sets depends combinatorially on the number of decision variables and constraints, these methods have a worst case complexity that is exponential in the problem size (as opposed to first-order and interior point methods). However, active set methods have proved to work quite well in practice.

Active Set for LPs

$$\begin{array}{ll} \min_z & c' z \\ \text{subj. to} & Gz \leq w \end{array} \quad (3)$$

- Central to active set methods for LP is the observation that a solution is always attained at a vertex of the polyhedral feasible set.
- A simple strategy would be to enumerate all vertices of the polyhedron and declare the vertex with the smallest cost as solution.
- However, one can do better by only visiting those vertices that improve the cost at the previous ones.
- This is the main idea behind active set methods for LP.

Table of Contents

1. Introduction

2. Unconstrained Optimization

2.1 Gradient Methods

2.2 Newton's Method

3. Constrained Optimization

3.1 Gradient Methods

3.2 Interior Point Methods

3.3 Active Set Methods

4. Software

4.1 Modeling

4.2 Solvers for Desktop PCs

4.3 Solvers for Embedded Platforms

Table of Contents

4. Software

4.1 Modeling

4.2 Solvers for Desktop PCs

4.3 Solvers for Embedded Platforms

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; z = z0;
for k = 1:N
    z = A*z + B*u{k};
    objective = objective + norm(Q*z,1) + norm(R*u{k},1);
    constraints = [constraints, -1 <= u{k}<= 1, -5<=z<=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

Table of Contents

4. Software

4.1 Modeling

4.2 Solvers for Desktop PCs

4.3 Solvers for Embedded Platforms

Software for Solving Convex Problems on Desktop PCs

Standalone solvers (tedious to use without modeling language)

- **Ipopt** (projects.coin-or.org/ipopt): widely used free solver for large-scale nonlinear optimization.
- **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

All listed solvers are based on interior-point methods

Table of Contents

4. Software

4.1 Modeling

4.2 Solvers for Desktop PCs

4.3 Solvers for Embedded Platforms

Convex Optimization Solvers for Embedded Platforms

Solvers: – open source:

- **qpOASES** (www.kuleuven.be/optec/software/qpOASES): active set solver, free (LGPL)
- **OOQP** (pages.cs.wisc.edu/~swright/ooqp): object-oriented QP solver (needs LAPACK/BLAS)
- **ECOS** (github.com/ifa-ethz/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
- **FiOrdOs** (fiordos.ethz.ch): code generation for gradient methods. Supports certification, automatic preconditioning, etc.
- **FORCES** (forces.ethz.ch): code generation for interior point methods. Super fast for MPC-like optimization problems.
- **FORCES PRO** (www.embotech.com): Professional version of FORCES