Tutorial on numerical optimal control

Sébastien Kleff

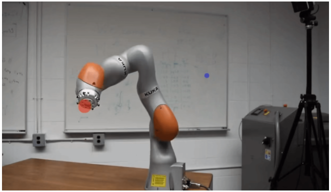
Oct. 3, 2024

Why optimal control?

Stairs climbing

https://youtu.be/v6MhPl2ICsc

Pick-and-place



https://youtu.be/ZtyCJYsGf4U

Parkour

https://youtu.be/tF4DML7FIWk

Obstacle avoidance

They all solve an **optimal control problem**

Tutorial objectives

Ideally, by the end of this tutorial :

- The relations between OC, MPC & DDP should be clear(er) to you
- You will understand words like "direct multiple shooting"
- You can implement your own MPC to control your favorite robot

To achieve these goals, I will provide

- Quick overview of Optimal Control
- Crash course on nonlinear optimization
- Tutorial using Crocoddyl and mim_solvers

Tutorial (will be) available on a dedicated repo :
`https://github.com/skleff1994/mpc_tutorial`

Tutorial plan

Tentative plan :

- Session 1 : Optimal control
- Session 2 : Numerical optimization
- Session 3 : MPC

Optimal control: Problem definition

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{1}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

Optimal control: Problem definition

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{1}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

Components

- State and control trajectories $x(.) : \mathbb{R}^+ \to \mathbb{R}^{n_x}, u(.) : \mathbb{R}^+ \to \mathbb{R}^{n_u}$

Optimal control: Problem definition

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{1}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

Components

- State and control trajectories $x(.) : \mathbb{R}^+ \rightarrow \mathbb{R}^{n_x}, u(.) : \mathbb{R}^+ \rightarrow \mathbb{R}^{n_u}$
- Running cost $\ell : \mathbb{R}^{n_x} : \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^+$ and terminal cost $\ell_T : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^+$

Optimal control: Problem definition

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{1}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

Components

- State and control trajectories $x(.) : \mathbb{R}^+ \to \mathbb{R}^{n_x}, u(.) : \mathbb{R}^+ \to \mathbb{R}^{n_u}$
- Running cost $\ell : \mathbb{R}^{n_x} : \times \mathbb{R}^{n_u} \to \mathbb{R}^+$ and terminal cost $\ell_T : \mathbb{R}^{n_x} \to \mathbb{R}^+$
- Dynamics constraint $f : \mathbb{R}^{n_x} : \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$

Optimal control: Problem definition

Continuous-time **Optimal Control Problem** (OCP)

$$V(x_0, t_0) = \min_{u(.)} \int_{t_0}^{T} \ell\big(x(t), u(t)\big)\,\mathrm{d}t + \ell_T(x(T)) \tag{1}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\big(x(t), u(t)\big) & \forall t \in [t_0, T] \\ c\big(x(t), u(t)\big) \le 0 & \forall t \in [t_0, T] \end{cases}$$

Components

- State and control trajectories $x(.) : \mathbb{R}^+ \rightarrow \mathbb{R}^{n_x}, u(.) : \mathbb{R}^+ \rightarrow \mathbb{R}^{n_u}$
- Running cost $\ell : \mathbb{R}^{n_x} : \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^+$ and terminal cost $\ell_T : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^+$
- Dynamics constraint $f : \mathbb{R}^{n_x} : \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$
- Path constraints $c : \mathbb{R}^{n_x} : \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_c}$ (possibly terminal c_T)

Optimal control: Problem definition

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{1}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

Components

- State and control trajectories $x(.) : \mathbb{R}^+ \to \mathbb{R}^{n_x}, u(.) : \mathbb{R}^+ \to \mathbb{R}^{n_u}$
- Running cost $\ell : \mathbb{R}^{n_x} : \times\mathbb{R}^{n_u} \to \mathbb{R}^+$ and terminal cost $\ell_T : \mathbb{R}^{n_x} \to \mathbb{R}^+$
- Dynamics constraint $f : \mathbb{R}^{n_x} : \times\mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$
- Path constraints $c : \mathbb{R}^{n_x} : \times\mathbb{R}^{n_u} \to \mathbb{R}^{n_c}$ (possibly terminal c_T)
- Initial condition $(x_0, t_0) \in \mathbb{R}^{n_x} \times \mathbb{R}^+$ and horizon $T > 0$ (can be $+\infty$)

Optimal control: Problem definition

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{1}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

Components

- State and control trajectories $x(.) : \mathbb{R}^+ \to \mathbb{R}^{n_x}, u(.) : \mathbb{R}^+ \to \mathbb{R}^{n_u}$
- Running cost $\ell : \mathbb{R}^{n_x} : \times \mathbb{R}^{n_u} \to \mathbb{R}^+$ and terminal cost $\ell_T : \mathbb{R}^{n_x} \to \mathbb{R}^+$
- Dynamics constraint $f : \mathbb{R}^{n_x} : \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$
- Path constraints $c : \mathbb{R}^{n_x} : \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_c}$ (possibly terminal c_T)
- Initial condition $(x_0, t_0) \in \mathbb{R}^{n_x} \times \mathbb{R}^+$ and horizon $T > 0$ (can be $+\infty$)
- Some technical assumptions

Optimal control: Manipulator example

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{2}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

A 7-DoF torque-controlled manipulator must reach an end-effector position $p^{\text{des}} \in \mathbb{R}^3$ while using mimium energy and satisfying operating constraints

Optimal control: Manipulator example

Continuous-time **Optimal Control Problem** (OCP)

$$V(x_0, t_0) = \min_{u(.)} \int_{t_0}^{T} \ell\big(x(t), u(t)\big)\,\mathrm{d}t + \ell_T(x(T)) \tag{2}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\big(x(t), u(t)\big) & \forall t \in [t_0, T] \\ c\big(x(t), u(t)\big) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

A 7-DoF torque-controlled manipulator must reach an end-effector position $p^{\text{des}} \in \mathbb{R}^3$ while using mimium energy and satisfying operating constraints

- What is x ?
- What is u ?
- What is f ?
- What is ℓ ?
- What is c ?

Optimal control: Manipulator example

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{2}$$

$$\text{s.t.} \quad \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

A 7-DoF torque-controlled manipulator must reach an end-effector position $p^{\text{des}} \in \mathbb{R}^3$ while using mimium energy and satisfying operating constraints

- We define the state variable $x = (q, \dot{q}) \in \mathbb{R}^{2n_q}$ and the control input $u = \tau \in \mathbb{R}^{n_q}$ with $n_q = 7$
- The dynamics constraint f is given by the robot forward dynamics

$$\underbrace{\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - h(q, \dot{q})) \end{bmatrix}}_{f(x,u)}$$

Optimal control: Manipulator example

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{2}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

A 7-DoF torque-controlled manipulator must reach an end-effector position $p^{\text{des}} \in \mathbb{R}^3$ while using mimium energy and satisfying operating constraints

- Path constraints are e.g. joint limits $[x^{\text{min}}, x^{\text{max}}]$ and torque limits $[u^{\text{min}}, u^{\text{max}}]$

$$c(x, u) = \begin{bmatrix} x - x^{\text{max}} \\ x^{\text{min}} - x \\ u - u^{\text{max}} \\ u^{\text{min}} - u \end{bmatrix}$$

Optimal control: Manipulator example

Continuous-time **Optimal Control Problem** (OCP)

$$V(x_0, t_0) = \min_{u(.)} \int_{t_0}^{T} \ell\big(x(t), u(t)\big)\,\mathrm{d}t + \ell_T(x(T)) \tag{2}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\big(x(t), u(t)\big) & \forall t \in [t_0, T] \\ c\big(x(t), u(t)\big) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

A 7-DoF torque-controlled manipulator must reach an end-effector position $p^{\text{des}} \in \mathbb{R}^3$ while using mimium energy and satisfying operating constraints

- Initial condition : robot at rest $x_0 = (q_0, 0)$ at $t_0 = 0$
- Cost function : penalizes the distance to the target and the energy

$$\ell(x, u) = \alpha \|u\|_2^2$$
$$\ell_T(x) = \|p(q) - p^{\text{des}}\|_2^2$$

where $\alpha > 0$ is a scalar parameter and the end-effector position $p(q)$ is given by the robot's forward kinematics

Optimal control: Solution

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{13}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

The $\min$ is the optimal cost or *value function* $V : \mathbb{R}^{n_x} \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$

Optimal control: Solution

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{13}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

The $\min$ is the optimal cost or *value function* $V : \mathbb{R}^{n_x} \times \mathbb{R}^{+} \rightarrow \mathbb{R}^{+}$

The $\arg\min$ is called the *optimal feedback control policy* $\pi : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_u}$

Optimal control: Solution

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{13}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

The $\min$ is the optimal cost or *value function* $V : \mathbb{R}^{n_x} \times \mathbb{R}^+ \to \mathbb{R}^+$

The $\arg\min$ is called the *optimal feedback control policy* $\pi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$

Very generic formulation

Optimal control: Solution

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{13}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

The $\min$ is the optimal cost or *value function* $V : \mathbb{R}^{n_x} \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$

The $\arg\min$ is called the *optimal feedback control policy* $\pi : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_u}$

Very generic formulation

Well-established framework

Optimal control: Solution

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{13}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

The $\min$ is the optimal cost or *value function* $V : \mathbb{R}^{n_x} \times \mathbb{R}^+ \to \mathbb{R}^+$

The $\arg\min$ is called the *optimal feedback control policy* $\pi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$

Very generic formulation

Well-established framework

But how to solve this?

Optimal control: Problem definition

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{13}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

The decision variable $u(.) \in \left(\mathbb{R}^{n_u}\right)^{[0,T]}$ is infinite dimensional !

Optimal control: Problem definition

Continuous-time **Optimal Control Problem** (OCP)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{13}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

The decision variable $u(.) \in \left(\mathbb{R}^{n_u}\right)^{[0,T]}$ is infinite dimensional !

V is a *functional* characterized by the Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE)

$$-\frac{\partial V(x,t)}{\partial t} = \min_{u} \left[\ell(x,u) + \mathcal{I}_c(x,u) + \frac{\partial V(x,t)}{\partial x} f(x,u)\right]$$

Optimal control: Problem definition

Continuous-time **Optimal Control Problem** (OCP)

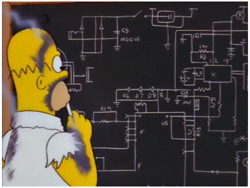
$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{13}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [t_0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [t_0, T] \end{cases}$$

The decision variable $u(.) \in (\mathbb{R}^{n_u})^{[0,T]}$ is infinite dimensional !

V is a *functional* characterized by the Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE)

$$-\frac{\partial V(x,t)}{\partial t} = \min_{u} \left[\ell(x,u) + \mathcal{I}_c(x,u) + \frac{\partial V(x,t)}{\partial x} f(x,u)\right]$$



No analytical solution except in very specific cases !

Analytic solution: Special case of LQR

Particular case : Linear-Quadratic Regulator (LQR)

When f is linear, ℓ is quadratic (no path constraints)

$$V(x_0, t_0) = \min_{u(.)} \int_{t_0}^{T} \left(x(t)^\top Q x(t) + u(t)^\top R u(t) \right) + x(T)^\top Q_T x(T) \tag{3}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 \\ \dot{x}(t) = Ax(t) + Bu(t) \qquad \forall t \in [t_0, T] \end{cases}$$

where $Q, Q_T, R \succ 0$ and (A, B) is controllable.

Analytic solution: Special case of LQR

Particular case : Linear-Quadratic Regulator (LQR)

When f is linear, ℓ is quadratic (no path constraints)

$$V(x_0, t_0) = \min_{u(.)} \int_{t_0}^{T} \left(x(t)^\top Q x(t) + u(t)^\top R u(t) \right) + x(T)^\top Q_T x(T) \tag{3}$$

$$\text{s.t.} \quad \begin{cases} x(t_0) = x_0 \\ \dot{x}(t) = Ax(t) + Bu(t) \qquad \forall t \in [t_0, T] \end{cases}$$

where $Q, Q_T, R \succ 0$ and (A, B) is controllable.

The value function V is quadratic, i.e. $V(x,t) = x^\top P(t) x$ where $P(t) \succ 0$ is the solution the Riccati differential equation

$$-\dot{P}(t) = A^\top P(t) + P(t)A - P(t)BR^{-1}B^\top P(t) + Q \tag{4}$$

Analytic solution: Special case of LQR

Particular case : Linear-Quadratic Regulator (LQR)

When f is linear, ℓ is quadratic (no path constraints)

$$V(x_0, t_0) = \min_{u(.)} \int_{t_0}^{T} \left(x(t)^\top Q x(t) + u(t)^\top R u(t) \right) + x(T)^\top Q_T x(T) \tag{3}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 \\ \dot{x}(t) = Ax(t) + Bu(t) \qquad \forall t \in [t_0, T] \end{cases}$$

where $Q, Q_T, R \succ 0$ and (A, B) is controllable.

The value function V is quadratic, i.e. $V(x,t) = x^\top P(t) x$ where $P(t) \succ 0$ is the solution the Riccati differential equation

$$-\dot{P}(t) = A^\top P(t) + P(t)A - P(t)BR^{-1}B^\top P(t) + Q \tag{4}$$

The optimal policy is linear $\pi(x(t)) = K(t)x(t)$ where $K(t) = -R^{-1}B^\top P(t)$

Analytic solution: Special case of LQR

Particular case : Linear-Quadratic Regulator (LQR)

When f is linear, ℓ is quadratic (no path constraints)

$$V\left(x_0, t_0\right) = \min_{u(.)} \int_{t_0}^{T} \left(x(t)^\top Q x(t) + u(t)^\top R u(t)\right) + x(T)^\top Q_T x(T) \tag{3}$$

$$\text{s.t.} \begin{cases} x(t_0) = x_0 \\ \dot{x}(t) = Ax(t) + Bu(t) \qquad \forall t \in [t_0, T] \end{cases}$$

where $Q, Q_T, R \succ 0$ and (A, B) is controllable.

The value function V is quadratic, i.e. $V(x,t) = x^\top P(t) x$ where $P(t) \succ 0$ is the solution the Riccati differential equation

$$-\dot{P}(t) = A^\top P(t) + P(t)A - P(t)BR^{-1}B^\top P(t) + Q \tag{4}$$

The optimal policy is linear $\pi(x(t)) = K(t)x(t)$ where $K(t) = -R^{-1}B^\top P(t)$

In the general case, HJB must be solved numerically

Numerical solution: Simple Pendulum

Example : Pendulum swing-up task

State $x = (\theta, \dot{\theta})$, control $u = \tau$ and dynamics model f

$$ml^2\ddot{\theta} + mgl\sin\theta = \tau \tag{5}$$

Running cost $\ell(x, u) = \alpha\|u\|^2$ and terminal cost $\ell_T(x) = \|x\|^2$

Numerical solution: Simple Pendulum

Example : Pendulum swing-up task

State $x = (\theta, \dot{\theta})$, control $u = \tau$ and dynamics model f

$$ml^2\ddot{\theta} + mgl\sin\theta = \tau \tag{5}$$

Running cost $\ell(x, u) = \alpha\|u\|^2$ and terminal cost $\ell_T(x) = \|x\|^2$

This is not LQR (f is nonlinear) so we must solve the HJB PDE numerically

Numerical solution: Simple Pendulum

Example : Pendulum swing-up task

State $x = (\theta, \dot{\theta})$, control $u = \tau$ and dynamics model f

$$ml^2\ddot{\theta} + mgl\sin\theta = \tau \tag{5}$$

Running cost $\ell(x, u) = \alpha\|u\|^2$ and terminal cost $\ell_T(x) = \|x\|^2$

This is not LQR (f is nonlinear) so we must solve the HJB PDE numerically

We **discretize** the state and control spaces into finite meshes

We solve the PDE numerically to compute V explicitly for every $(\theta, \dot{\theta})$

Check-out the pendulum example and play with it : `pendulum_bellman.py`

Numerical solution: Simple Pendulum

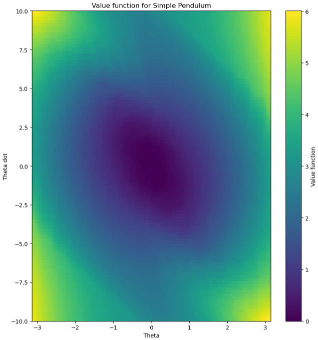
Example : Pendulum swing-up task

State $x = (\theta, \dot{\theta})$, control $u = \tau$ and dynamics model f

$$ml^2\ddot{\theta} + mgl\sin\theta = \tau \tag{5}$$

Running cost $\ell(x,u) = \alpha\|u\|^2$ and terminal cost $\ell_T(x) = \|x\|^2$

This is not LQR (f is nonlinear) so we must solve the HJB PDE numerically



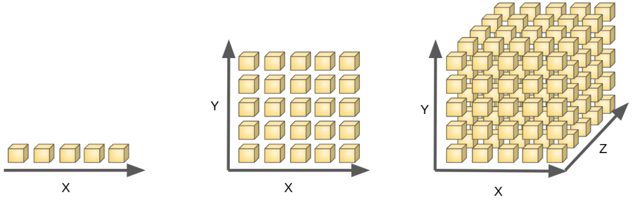
We **discretize** the state and control spaces into finite meshes

We solve the PDE numerically to compute V explicitly for every $(\theta, \dot{\theta})$

Check-out the pendulum example and play with it : `pendulum_bellman.py`

Numerical solution: Curse of dimensionality

Major problem : the number of points N required to maintain the same sampling density increases **exponentially** with the state space dimension n_x



100 points per dimension with $n_x = 2$: $N = 10^4$ points

100 points per dimension with $n_x = 6$: $N = 10^8$ points

Our 7-DoF torque-controlled manipulator has $n_x = 14$...

Computing V explicitly is not tractable if $n_x \geq 4$ or 5

Recap

Analytic resolution of HJB PDE : *"Explicit formula for V"*

- Exact *global* solution (i.e. compute $V(x)$ for all x)
- Feedback (closed-loop) policy $\pi(x)$
- Only works in very specific cases (e.g. LQR)

Recap

Analytic resolution of HJB PDE : *"Explicit formula for V"*

- Exact *global* solution (i.e. compute $V(x)$ for all x)
- Feedback (closed-loop) policy $\pi(x)$
- Only works in very specific cases (e.g. LQR)

Numerical resolution of HJB PDE : *"Integrate the HJB PDE"*

- Approximate global solution
- Feedback (closed-loop) policy $\pi(x)$
- Curse of dimensionality

Recap

Analytic resolution of HJB PDE : *"Explicit formula for V"*

- Exact *global* solution (i.e. compute $V(x)$ for all x)
- Feedback (closed-loop) policy $\pi(x)$
- Only works in very specific cases (e.g. LQR)

Numerical resolution of HJB PDE : *"Integrate the HJB PDE"*

- Approximate global solution
- Feedback (closed-loop) policy $\pi(x)$
- Curse of dimensionality

Luckily there is an alternative : direct optimal control

Direct Optimal Control

1. "First discretize" : Transform OCP into Nonlinear Program
2. "Then optimize" : Solve Nonlinear Program

Direct Optimal Control

1. "First discretize" : Transform OCP into Nonlinear Program
2. "Then optimize" : Solve Nonlinear Program

- Can deal with large, difficult problems

Direct Optimal Control

1. "First discretize" : Transform OCP into Nonlinear Program
2. "Then optimize" : Solve Nonlinear Program

- Can deal with large, difficult problems
- Many solvers available

Direct Optimal Control

1. "First discretize" : Transform OCP into Nonlinear Program
2. "Then optimize" : Solve Nonlinear Program

- Can deal with large, difficult problems
- Many solvers available
- Widely used in robotics

Direct Optimal Control

1. "First discretize" : Transform OCP into Nonlinear Program
2. "Then optimize" : Solve Nonlinear Program

- Can deal with large, difficult problems
- Many solvers available
- Widely used in robotics
- Relies on well-established numerical optimization tools

Direct Optimal Control

1. "First discretize" : Transform OCP into Nonlinear Program
2. "Then optimize" : Solve Nonlinear Program

- Can deal with large, difficult problems
- Many solvers available
- Widely used in robotics
- Relies on well-established numerical optimization tools
- Local solutions only (only valid around some given $x(t)$)

Direct Optimal Control

1. "First discretize" : Transform OCP into Nonlinear Program
2. "Then optimize" : Solve Nonlinear Program

- Can deal with large, difficult problems
- Many solvers available
- Widely used in robotics
- Relies on well-established numerical optimization tools
- Local solutions only (only valid around some given $x(t)$)
- Control trajectories $u(t)$ (open-loop policy)*

***Spoiler** : *MPC will essentially use direct optimal control to approximate the closed-loop policy* $\pi(x)$ *through repeated open-loop solutions* $u(t)$

Direct approach: "First discretize"

Transcription : Parametrize the infinite-dimensional OCP (13)

Direct approach: "First discretize"

Transcription : Parametrize the infinite-dimensional OCP (13)

1. **Discretization** of the time horizon $t_0, ..., t_N$ with $t_k = k\delta$ for $k \in [0, N]$

Direct approach: "First discretize"

Transcription : Parametrize the infinite-dimensional OCP (13)

1. **Discretization** of the time horizon $t_0, ..., t_N$ with $t_k = k\delta$ for $k \in [0, N]$
2. **Parametrization** e.g. zero-order hold $u(t) = u(t_k)$ for $t \in [t_k, t_{k+1})$

Direct approach: "First discretize"

Transcription : Parametrize the infinite-dimensional OCP (13)

1. **Discretization** of the time horizon $t_0, ..., t_N$ with $t_k = k\delta$ for $k \in [0, N]$
2. **Parametrization** e.g. zero-order hold $u(t) = u(t_k)$ for $t \in [t_k, t_{k+1})$
3. **Integration** of the state dynamics $x(.)$, e.g. using Euler and midpoint rule

$$x_{k+1} = x_k + \underbrace{F(x_k, u_k)}_{\text{continuous-time dynamics}} \delta \tag{6}$$

$$\ell(x_k, u_k) = \underbrace{L(x_k, u_k)}_{\text{continuous-time cost}} \delta \tag{7}$$

Direct approach: "First discretize"

Transcription : Parametrize the infinite-dimensional OCP (13)

1. **Discretization** of the time horizon $t_0, ..., t_N$ with $t_k = k\delta$ for $k \in [0, N]$
2. **Parametrization** e.g. zero-order hold $u(t) = u(t_k)$ for $t \in [t_k, t_{k+1})$
3. **Integration** of the state dynamics $x(.)$, e.g. using Euler and midpoint rule

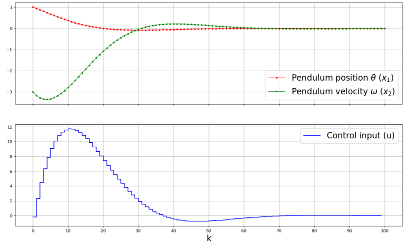
$$x_{k+1} = x_k + \underbrace{F(x_k, u_k)}_{\text{continuous-time dynamics}} \delta \tag{6}$$

$$\ell(x_k, u_k) = \underbrace{L(x_k, u_k)}_{\text{continuous-time cost}} \delta \tag{7}$$

Pendulum with semi-explicit Euler

$$\dot{\theta}_{k+1} = \dot{\theta}_k - \overbrace{\left(\frac{-g\sin(\theta)}{l} + u\right)}^{\text{continuous-time acceleration}} \delta$$

$$\theta_{k+1} = \theta_k + \delta\dot{\theta}_{k+1}$$



Direct approach: "First discretize"

This leads to a *discrete-time OCP* with *finite-dimensional* decision variables

$$\min_{u_0,\dots,u_{T-1}} \sum_{k=0}^{T-1} \ell(x_k, u_k) + \ell_T(x_T) \tag{8}$$

$$\text{s.t.} \begin{cases} x_0 = \hat{x} \\ x_{k+1} = f(x_k, u_k) \\ c\,(x_k, u_k) \leq 0 \end{cases}$$

Remarks

Direct approach: "First discretize"

This leads to a *discrete-time OCP* with *finite-dimensional* decision variables

$$\min_{u_0,\ldots,u_{T-1}} \sum_{k=0}^{T-1} \ell(x_k, u_k) + \ell_T(x_T) \tag{8}$$

$$\text{s.t.} \begin{cases} x_0 = \hat{x} \\ x_{k+1} = f(x_k, u_k) \\ c\,(x_k, u_k) \leq 0 \end{cases}$$

Remarks

- Discrete-time problems are much easier to study (see Bertsekas)

Direct approach: "First discretize"

This leads to a *discrete-time OCP* with *finite-dimensional* decision variables

$$\min_{u_0,\ldots,u_{T-1}} \sum_{k=0}^{T-1} \ell(x_k, u_k) + \ell_T(x_T) \tag{8}$$

$$\text{s.t.} \begin{cases} x_0 = \hat{x} \\ x_{k+1} = f(x_k, u_k) \\ c\,(x_k, u_k) \leq 0 \end{cases}$$

Remarks

- Discrete-time problems are much easier to study (see Bertsekas)
- Most of continuous-time optimal control theory applies in discrete-time

Direct approach: "First discretize"

This leads to a *discrete-time OCP* with *finite-dimensional* decision variables

$$\min_{u_0,\dots,u_{T-1}} \sum_{k=0}^{T-1} \ell(x_k,u_k) + \ell_T(x_T) \tag{8}$$

$$\text{s.t.} \quad \begin{cases} x_0 = \hat{x} \\ x_{k+1} = f(x_k,u_k) \\ c\left(x_k,u_k\right) \le 0 \end{cases}$$

Remarks

- Discrete-time problems are much easier to study (see Bertsekas)
- Most of continuous-time optimal control theory applies in discrete-time
- For instance, the Bellman equation is the discrete-time equivalent of HJB

$$V_j(x) = \min_u \; V_{j+1}\left(f(x,u)\right) + \ell(x,u) + \mathcal{I}_c(x,u) \tag{9}$$

where V_j is the optimal cost-to-go at stage j

Discrete-time OCP: Special case of LQR

Particular case : Linear-Quadratic Regulator (LQR)

When f is linear, ℓ is quadratic (no path constraints)

$$V_0\left(x_0\right) = \min_{u_0,\dots,u_{T-1}} \sum_{k=0}^{T-1} \left(x_k^\top Q x_k + u_k^\top R u_k\right) + x_T^\top Q_T x_T \tag{10}$$

$$\text{s.t.} \quad \begin{cases} x_0 = \hat{x} \\ x_{k+1} = A x_k + B u_k \qquad \forall k \in \{0, \dots, N-1\} \end{cases}$$

where $Q, Q_T, R \succ 0$ and (A, B) is controllable.

Discrete-time OCP: Special case of LQR

Particular case : Linear-Quadratic Regulator (LQR)

When f is linear, ℓ is quadratic (no path constraints)

$$V_0\left(x_0\right) = \min_{u_0,\dots,u_{T-1}} \sum_{k=0}^{T-1} \left(x_k^\top Q x_k + u_k^\top R u_k\right) + x_T^\top Q_T x_T \tag{10}$$

$$\text{s.t.} \begin{cases} x_0 = \hat{x} \\ x_{k+1} = A x_k + B u_k & \forall k \in \{0, \dots, N-1\} \end{cases}$$

where $Q, Q_T, R \succ 0$ and (A, B) is controllable.

The value function V is quadratic, i.e. $V_k(x) = x^\top P_k x$ where $P_k \succ 0$ is the solution the Riccati differential equation

$$P_k = A^\top P_k A - (A^\top P_k B)(R + B^\top P_k B)^{-1} B^\top P_k A + Q \tag{11}$$

Discrete-time OCP: Special case of LQR

Particular case : Linear-Quadratic Regulator (LQR)

When f is linear, ℓ is quadratic (no path constraints)

$$V_0\left(x_0\right) = \min_{u_0,\dots,u_{T-1}} \sum_{k=0}^{T-1} \left(x_k^\top Q x_k + u_k^\top R u_k\right) + x_T^\top Q_T x_T \tag{10}$$

$$\text{s.t.} \begin{cases} x_0 = \hat{x} \\ x_{k+1} = A x_k + B u_k \qquad \forall k \in \{0, \dots, N-1\} \end{cases}$$

where $Q, Q_T, R \succ 0$ and (A, B) is controllable.

The value function V is quadratic, i.e. $V_k(x) = x^\top P_k x$ where $P_k \succ 0$ is the solution the Riccati differential equation

$$P_k = A^\top P_k A - (A^\top P_k B)(R + B^\top P_k B)^{-1} B^\top P_k A + Q \tag{11}$$

The optimal policy is linear $\pi_k(x) = K_k x$ where $K_k = -R^{-1} B^\top P_k$

Discrete-time OCP: Special case of LQR

Particular case : Linear-Quadratic Regulator (LQR)

When f is linear, ℓ is quadratic (no path constraints)

$$V_0\left(x_0\right) = \min_{u_0,\ldots,u_{T-1}} \sum_{k=0}^{T-1} \left(x_k^\top Q x_k + u_k^\top R u_k\right) + x_T^\top Q_T x_T \tag{10}$$

$$\text{s.t.} \quad \begin{cases} x_0 = \hat{x} \\ x_{k+1} = A x_k + B u_k \qquad \forall k \in \{0, \ldots, N-1\} \end{cases}$$

where $Q, Q_T, R \succ 0$ and (A, B) is controllable.

The value function V is quadratic, i.e. $V_k(x) = x^\top P_k x$ where $P_k \succ 0$ is the solution the Riccati differential equation

$$P_k = A^\top P_k A - (A^\top P_k B)(R + B^\top P_k B)^{-1} B^\top P_k A + Q \tag{11}$$

The optimal policy is linear $\pi_k(x) = K_k x$ where $K_k = -R^{-1} B^\top P_k$

In the general case, there is no analytic solution for V, π

Direct approach: "Then optimize"

The discrete-time OCP

$$V_0(x_0) = \min_{u_0,\dots,u_{T-1}} \sum_{k=0}^{T-1} \ell(x_k, u_k) + \ell_T(x_T) \tag{8}$$

$$\text{s.t.} \begin{cases} x_0 = \hat{x} \\ x_{k+1} = f(x_k, u_k) \\ c\left(x_k, u_k\right) \leq 0 \end{cases}$$

can in theory be solved **globally** in the general case by solving Bellman's equation, but it suffers from the curse of dimensionality (like HJB).

Direct approach: "Then optimize"

The discrete-time OCP

$$V_0(x_0) = \min_{u_0,\dots,u_{T-1}} \sum_{k=0}^{T-1} \ell(x_k, u_k) + \ell_T(x_T) \tag{8}$$

$$\text{s.t.} \begin{cases} x_0 = \hat{x} \\ x_{k+1} = f(x_k, u_k) \\ c\left(x_k, u_k\right) \leq 0 \end{cases}$$

can in theory be solved **globally** in the general case by solving Bellman's equation, but it suffers from the curse of dimensionality (like HJB).

Instead, we seek a **local** solution by solving (8) as a Nonlinear Program (NLP)

$$\min_z \; L(z) \tag{12}$$

$$\text{s.t. } G(z) \leq 0$$

where $z \in \mathbb{R}^n$, $L : \mathbb{R}^n \to \mathbb{R}^+$ and $G : \mathbb{R}^n \to \mathbb{R}^{n_c}$.

Direct approach: "Then optimize"

The discrete-time OCP

$$V_0(x_0) = \min_{u_0,\ldots,u_{T-1}} \sum_{k=0}^{T-1} \ell(x_k, u_k) + \ell_T(x_T) \tag{8}$$

$$\text{s.t.} \begin{cases} x_0 = \hat{x} \\ x_{k+1} = f(x_k, u_k) \\ c\,(x_k, u_k) \leq 0 \end{cases}$$

can in theory be solved **globally** in the general case by solving Bellman's equation, but it suffers from the curse of dimensionality (like HJB).

Instead, we seek a **local** solution by solving (8) as a Nonlinear Program (NLP)

$$\min_{z} \; L(z) \tag{12}$$

$$\text{s.t. } G(z) \leq 0$$

where $z \in \mathbb{R}^n$, $L : \mathbb{R}^n \to \mathbb{R}^+$ and $G : \mathbb{R}^n \to \mathbb{R}^{n_c}$.

This is a standard nonlinear optimization problem that can be solved using textbook numerical optimization

Recap

- Optimal control is a generic framework
- OCPs are challenging to solve globally
- We can seek local solutions instead
- This requires to transform the original OCP into an NLP

Next time

Session 2 will focus on the NLP resolution

- What are the main techniques to solve a generic NLP
- How our NLP has an special structure
- How this structure can be exploited to solve it efficiently

Session 3 will focus on MPC implementation and introduction of the existing tools (Crocoddyl, mim_solvers)

Previously

In Session 1 : the continuous-time optimal control problem (OCP)

$$V(x_0) = \min_{u(.)} \int_0^T \ell\left(x(t), u(t)\right) \mathrm{d}t + \ell_T(x(T)) \tag{13}$$

$$\text{s.t.} \begin{cases} x(0) = x_0 & \\ \dot{x}(t) = f\left(x(t), u(t)\right) & \forall t \in [0, T] \\ c\left(x(t), u(t)\right) \leq 0 & \forall t \in [0, T] \end{cases}$$

was transcripted into a *discrete-time* OCP

$$V_0(x_0) = \min_{u_0, \dots, u_{T-1}} \sum_{k=0}^{T-1} \ell(x_k, u_k) + \ell_T(x_T) \tag{8}$$

$$\text{s.t.} \begin{cases} x_0 = \hat{x} & \\ x_{k+1} = f(x_k, u_k) & \forall k \in \{0, T-1\} \\ c\left(x_k, u_k\right) \leq 0 & \forall k \in \{0, T-1\} \end{cases}$$

which has now **finite-dimensional** decision variables.

Today

The discrete-time OCP

$$V_0(x_0) = \min_{u_0,\dots,u_{T-1}} \sum_{k=0}^{T-1} \ell(x_k, u_k) + \ell_T(x_T) \tag{8}$$
$$\text{s.t. } \begin{cases} x_0 = \hat{x} \\ x_{k+1} = f(x_k, u_k) \\ c\,(x_k, u_k) \leq 0 \end{cases}$$

is still intractable to solve globally.

But we said it could be solved **locally as a Nonlinear Program (NLP)**

$$\min_z \; L(z) \tag{12}$$
$$\text{s.t. } G(z) \leq 0$$

using standard numerical optimization techniques (see Nocedal)

Today

The discrete-time OCP

$$V_0(x_0) = \min_{u_0,\dots,u_{T-1}} \sum_{k=0}^{T-1} \ell(x_k, u_k) + \ell_T(x_T) \tag{8}$$

$$\text{s.t.} \begin{cases} x_0 = \hat{x} \\ x_{k+1} = f(x_k, u_k) \\ c\left(x_k, u_k\right) \leq 0 \end{cases}$$

is still intractable to solve globally.

But we said it could be solved **locally as a Nonlinear Program (NLP)**

$$\min_z \; L(z) \tag{12}$$

$$\text{s.t. } G(z) \leq 0$$

using standard numerical optimization techniques (see Nocedal)

Today I will show you how to solve NLPs !

Today

Today, we will study the NLP

$$\begin{aligned} \min_{z} \ & F(z) \\ \text{s.t. } & G(z) = 0 \\ & H(z) \leq 0 \end{aligned} \tag{14}$$

Today

Today, we will study the NLP

$$\begin{aligned} \min_{z} \ & F(z) \\ \text{s.t. } & G(z) = 0 \\ & H(z) \leq 0 \end{aligned} \tag{14}$$

Notations warning :

- The NLP cost is denoted by F (vs L last time)
- The NLP constraint is split into equality (G) and inequality (H)

Unconstrained optimization optimization

Consider the unconstrained optimization problem

$$\min_{z \in \mathbb{R}^n} F(z) \tag{15}$$

Unconstrained optimization optimization

Consider the unconstrained optimization problem

$$\min_{z \in \mathbb{R}^n} F(z) \tag{15}$$

Definition (Global minimizer)

z^* is called a global minimizer if for any $z \in \mathbb{R}^n$, $F(z^*) \leq F(z)$

Unconstrained optimization optimization

Consider the unconstrained optimization problem

$$\min_{z \in \mathbb{R}^n} F(z) \tag{15}$$

Definition (Global minimizer)

z^* is called a global minimizer if for any $z \in \mathbb{R}^n$, $F(z^*) \leq F(z)$

Definition (Local minimizer)

z^* is called a global minimizer if there exist a neighborhood $\mathcal{N}$ of z^* such that for any $z \in \mathcal{N}$, $F(z^*) \leq F(z)$

Unconstrained optimization optimization

Consider the unconstrained optimization problem

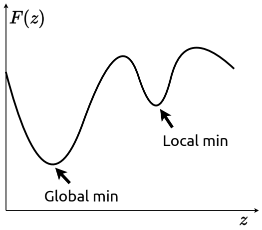
$$\min_{z \in \mathbb{R}^n} F(z) \tag{15}$$

Definition (Global minimizer)

z^* is called a global minimizer if for any $z \in \mathbb{R}^n$, $F(z^*) \leq F(z)$

Definition (Local minimizer)

z^* is called a global minimizer if there exist a neighborhood $\mathcal{N}$ of z^* such that for any $z \in \mathcal{N}$, $F(z^*) \leq F(z)$



Unconstrained optimization optimization

Consider the unconstrained optimization problem

$$\min_{z \in \mathbb{R}^n} F(z) \tag{15}$$

Definition (Global minimizer)

z^* is called a global minimizer if for any $z \in \mathbb{R}^n$, $F(z^*) \leq F(z)$

Definition (Local minimizer)

z^* is called a global minimizer if there exist a neighborhood $\mathcal{N}$ of z^* such that for any $z \in \mathcal{N}$, $F(z^*) \leq F(z)$

- Global minimizer $\implies$ local minimizer (converse not true)
- Global minimizers are often difficult to find

Unconstrained optimization optimization

Consider the unconstrained optimization problem

$$\min_{z \in \mathbb{R}^n} F(z) \tag{15}$$

Definition (Global minimizer)

z^* is called a global minimizer if for any $z \in \mathbb{R}^n$, $F(z^*) \leq F(z)$

Definition (Local minimizer)

z^* is called a global minimizer if there exist a neighborhood $\mathcal{N}$ of z^* such that for any $z \in \mathcal{N}$, $F(z^*) \leq F(z)$

- Global minimizer $\implies$ local minimizer (converse not true)
- Global minimizers are often difficult to find

Theorem (First-order necessary conditions)

If z^* is a local minimizer then it is a stationary point, i.e. $\nabla F(z^*) = 0$.

Unconstrained optimization optimization

Consider the unconstrained optimization problem

$$\min_{z \in \mathbb{R}^n} F(z) \tag{15}$$

Definition (Global minimizer)

z^* is called a global minimizer if for any $z \in \mathbb{R}^n$, $F(z^*) \leq F(z)$

Definition (Local minimizer)

z^* is called a global minimizer if there exist a neighborhood $\mathcal{N}$ of z^* such that for any $z \in \mathcal{N}$, $F(z^*) \leq F(z)$

- Global minimizer $\implies$ local minimizer (converse not true)
- Global minimizers are often difficult to find

Theorem (First-order necessary conditions)

If z^* is a local minimizer then it is a stationary point, i.e. $\nabla F(z^*) = 0$.

- We typically look for stationary points
- Most of the interesting results hold when F is convex

Unconstrained optimization optimization

We look for **local** minimizers by constructing a sequence of iterates $(z^k)_{k\in\mathbb{N}}$ converging to a local minimum of F

$$z^{k+1} = z^k + \alpha^k \Delta z^k$$

where $\Delta z^k \in \mathbb{R}^n$ is the **search direction** and $\alpha^k \geq 0$ is the **step size**.

Unconstrained optimization optimization

We look for **local** minimizers by constructing a sequence of iterates $(z^k)_{k\in\mathbb{N}}$ converging to a local minimum of F

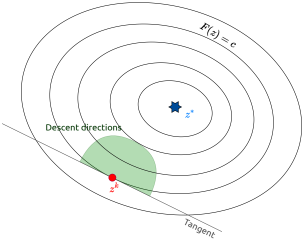
$$z^{k+1} = z^k + \alpha^k \Delta z^k$$

where $\Delta z^k \in \mathbb{R}^n$ is the **search direction** and $\alpha^k \geq 0$ is the **step size**.

Definition (Descent direction)

The search direction Δz^k is a valid descent direction if

$$\nabla F(z^k)^\top \Delta z^k < 0$$



Unconstrained optimization optimization

We look for **local** minimizers by constructing a sequence of iterates $(z^k)_{k\in\mathbb{N}}$ converging to a local minimum of F

$$z^{k+1} = z^k + \alpha^k \Delta z^k$$

where $\Delta z^k \in \mathbb{R}^n$ is the **search direction** and $\alpha^k \geq 0$ is the **step size**.

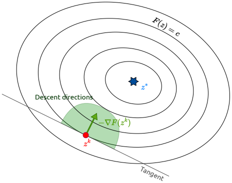
Definition (Descent direction)

The search direction Δz^k is a valid descent direction if

$$\nabla F(z^k)^\top \Delta z^k < 0$$

Example (Steepest descent)

The steepest descent direction $\Delta z = -\nabla F(z)$ is a valid descent direction.



Unconstrained optimization optimization

We look for **local** minimizers by constructing a sequence of iterates $(z^k)_{k\in\mathbb{N}}$ converging to a local minimum of F

$$z^{k+1} = z^k + \alpha^k \Delta z^k$$

where $\Delta z^k \in \mathbb{R}^n$ is the **search direction** and $\alpha^k \geq 0$ is the **step size**.

Definition (Descent direction)

The search direction Δz^k is a valid descent direction if

$$\nabla F(z^k)^\top \Delta z^k < 0$$

Example (Steepest descent)

The steepest descent direction $\Delta z = -\nabla F(z)$ is a valid descent direction.

Example (Newton's method)

The Newton direction defined by

$$\Delta z = -\nabla^2 F(z)^{-1} F(z) \tag{16}$$

is a valid descent direction (for $\nabla^2 F(z) \succ 0$).

Unconstrained optimization optimization

Once a valid descent direction Δz has been chosen, we need to make sure that the step corresponds to an actual decrease in the objective F, i.e. find α s.t.

$$F(z + \alpha \Delta z) < F(z)$$

Unconstrained optimization optimization

Once a valid descent direction Δz has been chosen, we need to make sure that the step corresponds to an actual decrease in the objective F, i.e. find α s.t.

$$F(z + \alpha \Delta z) < F(z)$$

Example (Exact line search)

Select the step size that benefits the most from Δz

$$\alpha = \arg\min_{\alpha > 0} F(z + \alpha \Delta z)$$

Unconstrained optimization optimization

Once a valid descent direction Δz has been chosen, we need to make sure that the step corresponds to an actual decrease in the objective F, i.e. find α s.t.

$$F(z + \alpha\Delta z) < F(z)$$

Example (Exact line search)

Select the step size that benefits the most from Δz

$$\alpha = \arg\min_{\alpha > 0} F(z + \alpha\Delta z)$$

Example (Backtracking line search (BLTS))

Try and reduce α until some "sufficient decrease" condition is met

---

Result: step size α

Choose $\alpha, \beta, \gamma \in \,]0, 1[$

while $F(z + \alpha\Delta z) > F(z) + \alpha\beta\nabla F(z)^\top \Delta z$ **do**

  $\alpha \leftarrow \gamma\alpha$

Numerical optimization: Steepest descent

Gradient descent minimizes the 1^{st}-order Taylor expansion of F around the current iterate

$$F(z + \Delta z) \approx F(z) + \nabla F(z)^\top \Delta z + o(\|\Delta z^2\|)$$

by searching along the negative gradient $\Delta z = -\nabla F(z)$

Numerical optimization: Steepest descent

Gradient descent minimizes the 1^{st}-order Taylor expansion of F around the current iterate

$$F(z + \Delta z) \approx F(z) + \nabla F(z)^\top \Delta z + o(\|\Delta z^2\|)$$

by searching along the negative gradient $\Delta z = -\nabla F(z)$

Theorem (Global linear convergence of gradient descent)

When f is strongly convex, gradient descent with BLTS converges linearly to the global mimimum $F^* \in \mathbb{R}$, i.e. there exist $\eta > 0$ s.t.

$$F(z^{k+1}) - F^* \leq \eta \left(F(z^{k+1}) - F^* \right)$$

Numerical optimization: Steepest descent

Gradient descent minimizes the 1^{st}-order Taylor expansion of F around the current iterate

$$F(z + \Delta z) \approx F(z) + \nabla F(z)^\top \Delta z + o(\|\Delta z^2\|)$$

by searching along the negative gradient $\Delta z = -\nabla F(z)$

Theorem (Global linear convergence of gradient descent)

When f is strongly convex, gradient descent with BLTS converges linearly to the global mimimum $F^* \in \mathbb{R}$, i.e. there exist $\eta > 0$ s.t.

$$F(z^{k+1}) - F^* \leq \eta \left(F(z^{k+1}) - F^* \right)$$

- Global linear convergence
- The linear convergence rate η depends on the conditioning $\nabla^2 F(z)$
- In practice, convergence can be ridiculously slow
- Play with the code : `mpc_tutorial/optimization/unconstrained.py`

Numerical optimization: Newton direction

Newton method minimizes the 2^{nd}-order Taylor expansion

$$F(z + \Delta z) \approx F(z) + \nabla F(z)^\top \Delta z + \frac{1}{2}\Delta z^\top \nabla^2 F(z) \Delta z + o(\|\Delta z^3\|)$$

by searching along $\Delta z = -\left(\nabla^2 F(z)\right)^{-1} \nabla F(z)$

Numerical optimization: Newton direction

Newton method minimizes the 2^{nd}-order Taylor expansion

$$F(z+\Delta z) \approx F(z) + \nabla F(z)^\top \Delta z + \frac{1}{2}\Delta z^\top \nabla^2 F(z) \Delta z + o(\|\Delta z^3\|)$$

by searching along $\Delta z = -\left(\nabla^2 F(z)\right)^{-1} \nabla F(z)$

Theorem (Local quadratic convergence of Newton)

When f is strongly convex, Newton method with BLTS converges quadratically to $F^* \in \mathbb{R}$, i.e. there exist $\eta > 0$ s.t.

$$\|\nabla F(z^{k+1})\| \leq \left(\eta \|\nabla F(z^{k+1})\|\right)^2$$

provided that z^0 is close enough to z^*

Numerical optimization: Newton direction

Newton method minimizes the 2^{nd}-order Taylor expansion

$$F(z+\Delta z) \approx F(z) + \nabla F(z)^\top \Delta z + \frac{1}{2}\Delta z^\top \nabla^2 F(z) \Delta z + o(\|\Delta z^3\|)$$

by searching along $\Delta z = -\left(\nabla^2 F(z)\right)^{-1} \nabla F(z)$

Theorem (Local quadratic convergence of Newton)

When f is strongly convex, Newton method with BLTS converges quadratically to $F^* \in \mathbb{R}$, i.e. there exist $\eta > 0$ s.t.

$$\|\nabla F(z^{k+1})\| \leq \left(\eta \|\nabla F(z^{k+1})\|\right)^2$$

provided that z^0 is close enough to z^*

- Local quadratic convergence : z^0 must be "close enough" to z^*
- Convergence is *much faster* than gradient descent
- $\nabla^2 F(z)$ is expensive to compute and must be $\succ 0$
- Play with the code : `mpc_tutorial/optimization/unconstrained.py`

Unconstrained optimization (recap)

In order to solve the unconstrained minimization problem

$$\min_{z \in \mathbb{R}^n} \; F(z)$$

Unconstrained optimization (recap)

In order to solve the unconstrained minimization problem

$$\min_{z \in \mathbb{R}^n} F(z)$$

we are looking for **local minimizers** of F by iteratively computing a **descent direction** and searching for a sufficient decrease along this direction.

Unconstrained optimization (recap)

In order to solve the unconstrained minimization problem

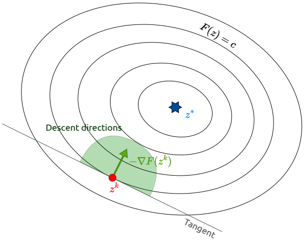
$$\min_{z \in \mathbb{R}^n} \; F(z)$$

we are looking for **local minimizers** of F by iteratively computing a **descent direction** and searching for a sufficient decrease along this direction.

Gradient descent

1^{st}-order method leveraging ∇F

Global linear convergence rate



Unconstrained optimization (recap)

In order to solve the unconstrained minimization problem

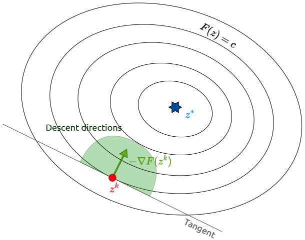
$$\min_{z \in \mathbb{R}^n} F(z)$$

we are looking for **local minimizers** of F by iteratively computing a **descent direction** and searching for a sufficient decrease along this direction.

Gradient descent

1^{st}-order method leveraging ∇F

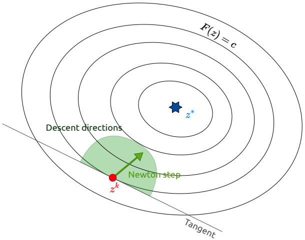
Global linear convergence rate



Newton method

2^{nd}-order method leveraging $\nabla F, \nabla^2 F$

Local quadratic convergence rate



Unconstrained optimization (recap)

In order to solve the unconstrained minimization problem

$$\min_{z \in \mathbb{R}^n} \; F(z)$$

we are looking for **local minimizers** of F by iteratively computing a **descent direction** and searching for a sufficient decrease along this direction.

Gradient descent

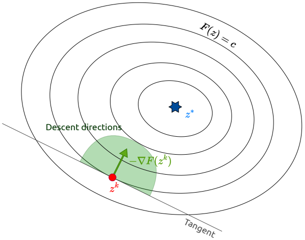
1^{st}-order method leveraging ∇F

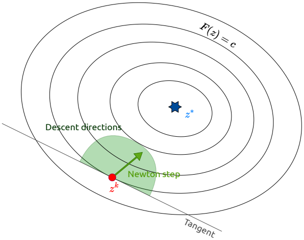
Global linear convergence rate

Newton method

2^{nd}-order method leveraging $\nabla F, \nabla^2 F$

Local quadratic convergence rate





Example

Unconstrained optimization (recap)

In order to solve the unconstrained minimization problem

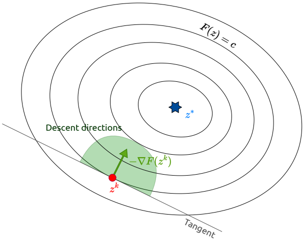
$$\min_{z\in\mathbb{R}^n} \; F(z)$$

we are looking for **local minimizers** of F by iteratively computing a **descent direction** and searching for a sufficient decrease along this direction.

Gradient descent

1^{st}-order method leveraging ∇F

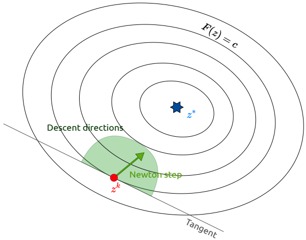
Global linear convergence rate



Newton method

2^{nd}-order method leveraging $\nabla F, \nabla^2 F$

Local quadratic convergence rate



What about constraints?

Constrained optimization: Problem

Consider now the problem with **constraints** $G : \mathbb{R}^n \mapsto \mathbb{R}^{n_e}, H : \mathbb{R}^n \mapsto \mathbb{R}^{n_i}$

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \; & F(z) \\ \text{s.t. } & G(z) = 0 \\ & H(z) \leq 0 \end{aligned}$$

Constrained optimization: Problem

Consider now the problem with **constraints** $G : \mathbb{R}^n \mapsto \mathbb{R}^{n_e}, H : \mathbb{R}^n \mapsto \mathbb{R}^{n_i}$

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \; & F(z) \\ \text{s.t. } & G(z) = 0 \\ & H(z) \leq 0 \end{aligned}$$

Definition (Feasible set)

The feasible set is defined as

$$\Omega = \{z \in \mathbb{R}^n \mid G(z) = 0 \wedge H(z) \leq 0\}$$

A minimizer z^* is a feasible point $z^* \in \Omega$ that minimizes F.

Constrained optimization: Problem

Consider now the problem with **constraints** $G : \mathbb{R}^n \mapsto \mathbb{R}^{n_e}, H : \mathbb{R}^n \mapsto \mathbb{R}^{n_i}$

$$\begin{aligned} &\min_{z \in \mathbb{R}^n} \ F(z) \\ \text{s.t. } &G(z) = 0 \\ &H(z) \leq 0 \end{aligned}$$

Definition (Feasible set)

The feasible set is defined as

$$\Omega = \{z \in \mathbb{R}^n \mid G(z) = 0 \wedge H(z) \leq 0\}$$

A minimizer z^* is a feasible point $z^* \in \Omega$ that minimizes F.

- Same notions of global vs local minimizers
- We are usually looking for a *local* solution $z^* \in \Omega$
- Easier (i.e. many results) when F is convex *and* G, H are linear

Constrained optimization: Problem

Consider now the problem with **constraints** $G : \mathbb{R}^n \mapsto \mathbb{R}^{n_e}, H : \mathbb{R}^n \mapsto \mathbb{R}^{n_i}$

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \ & F(z) \\ \text{s.t. } & G(z) = 0 \\ & H(z) \leq 0 \end{aligned}$$

Definition (Feasible set)

The feasible set is defined as

$$\Omega = \{z \in \mathbb{R}^n \mid G(z) = 0 \wedge H(z) \leq 0\}$$

A minimizer z^* is a feasible point $z^* \in \Omega$ that minimizes F.

- Same notions of global vs local minimizers
- We are usually looking for a *local* solution $z^* \in \Omega$
- Easier (i.e. many results) when F is convex *and* G, H are linear

Like in the unconstrained case, we will look for **local optima** by enforcing some first-order **necessary** conditions

Constrained optimization: KKT conditions

Definition (Lagrangian)

We define the Lagrangian as

$$\mathcal{L}(z, \lambda, \mu) = F(z) + \lambda^\top G(z) + \mu^\top H(z)$$

where $(\lambda, \mu) \in \mathbb{R}^{n_e} \times \mathbb{R}^{n_i}$ are the Lagrange multipliers associated (G, H). (λ, μ) are also called the *dual variables* and z the *primal variable*.

Constrained optimization: KKT conditions

Definition (Lagrangian)

We define the Lagrangian as

$$\mathcal{L}(z, \lambda, \mu) = F(z) + \lambda^\top G(z) + \mu^\top H(z)$$

where $(\lambda, \mu) \in \mathbb{R}^{n_e} \times \mathbb{R}^{n_i}$ are the Lagrange multipliers associated (G, H). (λ, μ) are also called the *dual variables* and z the *primal variable*.

Theorem (First-order necessary conditions for optimality)

If z^* be a local minimizer (+ some assumptions), then there exist $\lambda^* \in \mathbb{R}^{n_e}$ and $\mu^* \in \mathbb{R}^{n_i}$ such that the Karush-Kuhn-Tucker (KKT) conditions hold

$$\nabla_z \mathcal{L}(z^*, \lambda^*, \mu^*) = 0 \quad \textit{Stationarity} \tag{17}$$

$$G(z^*) = 0 \ \wedge \ H(z^*) \leq 0 \quad \textit{Primal feasibility} \tag{18}$$

$$\mu_i \geq 0 \quad \textit{Dual feasibility} \tag{19}$$

$$\mu_i H_i(z^*) = 0 \quad \textit{Complementary slackness} \tag{20}$$

- In the unconstrained case we were looking for stationary points
- We are now looking for **KKT points**

Constrained optimization: Newton method

Let's focus on the equality-constrained problem first (no H)

$$\begin{bmatrix} \nabla_z \mathcal{L}(z, \lambda) \\ G(z) \end{bmatrix} = 0 \tag{21}$$

Constrained optimization: Newton method

Let's focus on the equality-constrained problem first (no H)

$$\begin{bmatrix} \nabla_z \mathcal{L}(z,\lambda) \\ G(z) \end{bmatrix} = 0 \tag{21}$$

Now recall a fundamental result from undergrad analysis.

Theorem (Newton-Raphson method for nonlinear equations)

We can find a root of the nonlinear equation

$$r(w) = 0$$

iteratively using the Newton-Raphson algorithm

Result: *Root* w
while $\|r(w)\| > tol$ **do**
  Compute the step direction Δw *s.t.* $\nabla r(w)^\top \Delta w = -r(w)$
  Take the step $w \leftarrow w + \Delta w$

Constrained optimization: Newton method

Let's focus on the equality-constrained problem first (no H)

$$\begin{bmatrix} \nabla_z \mathcal{L}(z, \lambda) \\ G(z) \end{bmatrix} = 0 \tag{21}$$

Now recall a fundamental result from undergrad analysis.

Theorem (Newton-Raphson method for nonlinear equations)

We can find a root of the nonlinear equation

$$r(w) = 0$$

iteratively using the Newton-Raphson algorithm

Result: *Root* w
while $\|r(w)\| > tol$ **do**
  Compute the step direction Δw *s.t.* $\nabla r(w)^\top \Delta w = -r(w)$
  Take the step $w \leftarrow w + \Delta w$

Example : Newton-Raphson to solve $r(w) = 0$ with $r(w) = w^4 - 3w^3 + 2$

Play with it : `mpc_tutorial/optimization/newton_raphson.py`

Constrained optimization: Newton method

Let's focus on the equality-constrained problem first (no H)

$$\begin{bmatrix} \nabla_z \mathcal{L}(z, \lambda) \\ G(z) \end{bmatrix} = 0 \tag{21}$$

Idea: Apply Newton-Raphson to find a root of (21)

Solve $r(z, \lambda) = 0$ with

$$r(z, \lambda) = \begin{bmatrix} \nabla_z \mathcal{L}(z, \lambda) \\ G(z) \end{bmatrix}$$

Constrained optimization: Newton method

Let's focus on the equality-constrained problem first (no H)

$$\begin{bmatrix} \nabla_z \mathcal{L}(z,\lambda) \\ G(z) \end{bmatrix} = 0 \tag{21}$$

The Newton-Raphson step $\Delta w \triangleq (\Delta z, \Delta\lambda)$ is given by solving

$$\underbrace{\begin{bmatrix} \nabla^2_{zz} \mathcal{L}(z,\lambda) & \nabla G(z) \\ \nabla G(z)^\top & 0 \end{bmatrix}}_{\text{"}\nabla r(w)^\top\text{"}} \underbrace{\begin{bmatrix} \Delta z \\ \Delta\lambda \end{bmatrix}}_{\text{"}\Delta w\text{"}} = - \underbrace{\begin{bmatrix} \nabla_z \mathcal{L}(z,\lambda) \\ \nabla G(z) \end{bmatrix}}_{\text{"}r(w)\text{"}} \tag{22}$$

and it is called the **Newton direction**

Constrained optimization: Newton method

Let's focus on the equality-constrained problem first (no H)

$$\begin{bmatrix} \nabla_z \mathcal{L}(z,\lambda) \\ G(z) \end{bmatrix} = 0 \tag{21}$$

The Newton-Raphson step $\Delta w \triangleq (\Delta z, \Delta\lambda)$ is given by solving

$$\underbrace{\begin{bmatrix} \nabla^2_{zz}\mathcal{L}(z,\lambda) & \nabla G(z) \\ \nabla G(z)^\top & 0 \end{bmatrix}}_{"\nabla r(w)^\top"} \underbrace{\begin{bmatrix} \Delta z \\ \Delta\lambda \end{bmatrix}}_{"\Delta w"} = -\underbrace{\begin{bmatrix} \nabla_z \mathcal{L}(z,\lambda) \\ \nabla G(z) \end{bmatrix}}_{"r(w)"} \tag{22}$$

and it is called the **Newton direction**

The primal-dual variables (z,λ) are then updated

$$\begin{bmatrix} z^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} z^k \\ \lambda^k \end{bmatrix} + \begin{bmatrix} \Delta z \\ \Delta\lambda \end{bmatrix} \tag{23}$$

We need $\nabla^2_{zz}\mathcal{L}(z,\lambda) \succ 0$ and $\nabla G(z)$ full row rank

Constrained optimization: Newton method

Let's focus on the equality-constrained problem first (no H)

$$\begin{bmatrix} \nabla_z \mathcal{L}(z, \lambda) \\ G(z) \end{bmatrix} = 0 \tag{21}$$

The Newton-Raphson step $\Delta w \triangleq (\Delta z, \Delta\lambda)$ is given by solving

$$\underbrace{\begin{bmatrix} \nabla^2_{zz} \mathcal{L}(z, \lambda) & \nabla G(z) \\ \nabla G(z)^\top & 0 \end{bmatrix}}_{\text{"}\nabla r(w)^\top\text{"}} \underbrace{\begin{bmatrix} \Delta z \\ \Delta\lambda \end{bmatrix}}_{\text{"}\Delta w\text{"}} = - \underbrace{\begin{bmatrix} \nabla_z \mathcal{L}(z, \lambda) \\ \nabla G(z) \end{bmatrix}}_{\text{"}r(w)\text{"}} \tag{22}$$

and it is called the **Newton direction**

The primal-dual variables (z, λ) are then updated

$$\begin{bmatrix} z^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} z^k \\ \lambda^k \end{bmatrix} + \begin{bmatrix} \Delta z \\ \Delta\lambda \end{bmatrix} \tag{23}$$

We need $\nabla^2_{zz} \mathcal{L}(z, \lambda) \succ 0$ and $\nabla G(z)$ full row rank

Example : $F(z) = z^\top A z$ and $G(z) = z^\top B z - c$

Play with it : `mpc_tutorial/optimization/constrained.py`

Constrained optimization: Newton method

Let's focus on the equality-constrained problem first (no H)

$$\begin{bmatrix} \nabla_z \mathcal{L}(z, \lambda) \\ G(z) \end{bmatrix} = 0 \tag{21}$$

The Newton-Raphson step $\Delta w \triangleq (\Delta z, \Delta \lambda)$ is given by solving

$$\underbrace{\begin{bmatrix} \nabla^2_{zz} \mathcal{L}(z, \lambda) & \nabla G(z) \\ \nabla G(z)^\top & 0 \end{bmatrix}}_{"\nabla r(w)^\top"} \underbrace{\begin{bmatrix} \Delta z \\ \Delta \lambda \end{bmatrix}}_{"\Delta w"} = - \underbrace{\begin{bmatrix} \nabla_z \mathcal{L}(z, \lambda) \\ \nabla G(z) \end{bmatrix}}_{"r(w)"} \tag{22}$$

and it is called the **Newton direction**

The primal-dual variables (z, λ) are then updated

$$\begin{bmatrix} z^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} z^k \\ \lambda^k \end{bmatrix} + \begin{bmatrix} \Delta z \\ \Delta \lambda \end{bmatrix} \tag{23}$$

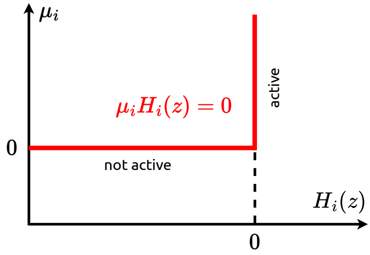
We need $\nabla^2_{zz} \mathcal{L}(z, \lambda) \succ 0$ and $\nabla G(z)$ full row rank

What about inequalities now ?

Constrained optimization: Inequalities

The complementary slackness condition $\mu_i H_i(z^*) = 0$ is **non-smooth**

We **cannot** apply Newton-Raphson directly in the presence of inequalities



Constrained optimization: Inequalities

Key: re-interpret the Newton method for equality-constrained problems

Theorem (Quadratic model interpretation)

The Newton-Raphson direction is ***also*** *given by solving a linear-quadratic approximation of the original NLP*

$$\begin{aligned} \min_{\Delta z} \ & \frac{1}{2}\Delta z^\top \nabla_{zz}\mathcal{L}(z,\lambda)\Delta z + \nabla F(z)^\top \Delta z \\ \text{s.t. } & \nabla G(z)^\top \Delta x + G(z) = 0 \qquad (\lambda^{QP}) \end{aligned} \tag{24}$$

In particular we have $\lambda^{QP} = \lambda^{k+1}$

Constrained optimization: Inequalities

Key: re-interpret the Newton method for equality-constrained problems

Theorem (Quadratic model interpretation)

*The Newton-Raphson direction is **also** given by solving a linear-quadratic approximation of the original NLP*

$$\begin{aligned} \min_{\Delta z} \ & \frac{1}{2}\Delta z^\top \nabla_{zz}\mathcal{L}(z,\lambda)\Delta z + \nabla F(z)^\top \Delta z \\ \textit{s.t. } & \nabla G(z)^\top \Delta x + G(z) = 0 \qquad (\lambda^{QP}) \end{aligned} \tag{24}$$

In particular we have $\lambda^{QP} = \lambda^{k+1}$

Constrained optimization: Inequalities

Key: re-interpret the Newton method for equality-constrained problems

Theorem (Quadratic model interpretation)

*The Newton-Raphson direction is **also** given by solving a linear-quadratic approximation of the original NLP*

$$\min_{\Delta z} \frac{1}{2}\Delta z^\top \nabla_{zz}\mathcal{L}(z,\lambda)\Delta z + \nabla F(z)^\top \Delta z \tag{24}$$

$$\text{s.t. } \nabla G(z)^\top \Delta x + G(z) = 0 \qquad (\lambda^{QP})$$

In particular we have $\lambda^{QP} = \lambda^{k+1}$

We were just solving a QP !

Constrained optimization: Inequalities

Key: re-interpret the Newton method for equality-constrained problems

Theorem (Quadratic model interpretation)

*The Newton-Raphson direction is **also** given by solving a linear-quadratic approximation of the original NLP*

$$\begin{aligned} \min_{\Delta z} \quad & \frac{1}{2}\Delta z^\top \nabla_{zz}\mathcal{L}(z,\lambda)\Delta z + \nabla F(z)^\top \Delta z \\ \textit{s.t.} \quad & \nabla G(z)^\top \Delta x + G(z) = 0 \qquad (\lambda^{QP}) \end{aligned} \tag{24}$$

In particular we have $\lambda^{QP} = \lambda^{k+1}$

We were just solving a QP !

Based on this key observation, there are 2 main ways to handle inequalities :

- Interior Point (IP) : "Get rid of inequality"
- Sequential-Quadratic Programming (SQP) : "Give it to the QP"

Constrained optimization: SQP

Idea : relegate the inequalities to the QP

Idea : relegate the inequalities to the QP

We are solving a **sequence of linear-quadratic approximations** of the original NLP around each iterate (z^k, λ^k)

$$\begin{aligned} \min_{\Delta z}\ & \frac{1}{2}\Delta z^\top \nabla_{zz}\mathcal{L}(z^k,\lambda^k)\Delta z + \nabla F(z^k)^\top \Delta z \\ \text{s.t. } & \nabla G(z^k)^\top \Delta x + G(z^k) = 0 \\ & \nabla H(z^k)^\top \Delta x + H(z^k) \leq 0 \end{aligned} \tag{25}$$

Constrained optimization: SQP

Idea : relegate the inequalities to the QP

We are solving a **sequence of linear-quadratic approximations** of the original NLP around each iterate (z^k, λ^k)

$$\begin{aligned} \min_{\Delta z}\ & \frac{1}{2}\Delta z^\top \nabla_{zz}\mathcal{L}(z^k,\lambda^k)\Delta z + \nabla F(z^k)^\top \Delta z \\ \text{s.t. }\ & \nabla G(z^k)^\top \Delta x + G(z^k) = 0 \\ & \nabla H(z^k)^\top \Delta x + H(z^k) \leq 0 \end{aligned} \tag{25}$$

Use any QP solver you like to compute the Newton direction $(\Delta z, \Delta\lambda, \Delta\mu)$

Constrained optimization: SQP

Idea : relegate the inequalities to the QP

We are solving a **sequence of linear-quadratic approximations** of the original NLP around each iterate (z^k, λ^k)

$$\begin{aligned} \min_{\Delta z} \ & \frac{1}{2}\Delta z^\top \nabla_{zz}\mathcal{L}(z^k, \lambda^k)\Delta z + \nabla F(z^k)^\top \Delta z \\ \text{s.t. } & \nabla G(z^k)^\top \Delta x + G(z^k) = 0 \\ & \nabla H(z^k)^\top \Delta x + H(z^k) \leq 0 \end{aligned} \tag{25}$$

Use any QP solver you like to compute the Newton direction $(\Delta z, \Delta \lambda, \Delta \mu)$

Typically perform a **line-search** using a *merit function* [Nocedal]

$$\begin{aligned} z^{k+1} &= z^k + \alpha \Delta z \\ \lambda^{k+1} &= \lambda^k + \alpha \Delta \lambda \\ \mu^{k+1} &= \mu^k + \alpha \Delta \mu \end{aligned}$$

Constrained optimization: SQP

Idea : relegate the inequalities to the QP

We are solving a **sequence of linear-quadratic approximations** of the original NLP around each iterate (z^k, λ^k)

$$\begin{aligned} \min_{\Delta z}\ & \frac{1}{2}\Delta z^\top \nabla_{zz}\mathcal{L}(z^k,\lambda^k)\Delta z + \nabla F(z^k)^\top \Delta z \\ \text{s.t. }\ & \nabla G(z^k)^\top \Delta x + G(z^k) = 0 \\ & \nabla H(z^k)^\top \Delta x + H(z^k) \leq 0 \end{aligned} \tag{25}$$

Use any QP solver you like to compute the Newton direction $(\Delta z, \Delta\lambda, \Delta\mu)$

Typically perform a **line-search** using a *merit function* [Nocedal]

$$\begin{aligned} z^{k+1} &= z^k + \alpha\Delta z \\ \lambda^{k+1} &= \lambda^k + \alpha\Delta\lambda \\ \mu^{k+1} &= \mu^k + \alpha\Delta\mu \end{aligned}$$

SQP has local quadratic convergence !

Constrained optimization: Recap

Key take-aways :

- Constrained optimization is much harder

Constrained optimization: Recap

Key take-aways :

- Constrained optimization is much harder
- KKT conditions are the 1^{st}-order necessary conditions for optimality

Constrained optimization: Recap

Key take-aways :

- Constrained optimization is much harder
- KKT conditions are the 1^{st}-order necessary conditions for optimality
- We can solve equality-constrained problems using Newton's method

Constrained optimization: Recap

Key take-aways :

- Constrained optimization is much harder
- KKT conditions are the 1^{st}-order necessary conditions for optimality
- We can solve equality-constrained problems using Newton's method
- This is equivalent to solving a sequence of equality-constrained QPs

Constrained optimization: Recap

Key take-aways :

- Constrained optimization is much harder
- KKT conditions are the 1^{st}-order necessary conditions for optimality
- We can solve equality-constrained problems using Newton's method
- This is equivalent to solving a sequence of equality-constrained QPs
- Inequality constraints are handled based on this idea

Constrained optimization: Recap

Key take-aways :

- Constrained optimization is much harder
- KKT conditions are the 1^{st}-order necessary conditions for optimality
- We can solve equality-constrained problems using Newton's method
- This is equivalent to solving a sequence of equality-constrained QPs
- Inequality constraints are handled based on this idea
- In particular, SQP solves a sequence of inequality-constrained QPs

Constrained optimization: Recap

Key take-aways :

- Constrained optimization is much harder
- KKT conditions are the 1^{st}-order necessary conditions for optimality
- We can solve equality-constrained problems using Newton's method
- This is equivalent to solving a sequence of equality-constrained QPs
- Inequality constraints are handled based on this idea
- In particular, SQP solves a sequence of inequality-constrained QPs
- SQP converges quadratically to KKT points

This was a long detour from our original problem

Now how to use SQP to solve our discrete-time OCP ?

Next time

In Session 3 we will address 2 questions

1. How to apply SQP to the solve the OCP ? i.e. what is z in terms of the OCP decision variables u_k, x_k ?
   - Single-shooting $z \triangleq (u_0, ..., u_{T-1})$. Only controls are decision variables
   - Multiple-shooting $z \triangleq (x_0, u_0, ..., x_{T-1}, u_{T-1}, x_T)$. Both controls *and* states are decisions variables.
2. How to solve the SQP (14) *efficiently* by exploiting the underlying structure of the OCP ?
3. What is Model-Predictive Control (MPC) and how to achieve it in practice ?

Single vs multiple shooting

What is z w.r.t. our optimal control variables x_k, u_k ?

We consider 2 options :

1. Single-shooting $z \triangleq (u_0, ..., u_{T-1})$. Only controls are decision variables, states are recovered from controls through integration of the dynamics

$$x_{k+1} = f(x_k, u_k) \tag{26}$$

2. Multiple-shooting $z \triangleq (x_0, u_0, ..., x_{T-1}, u_{T-1}, x_T)$. Both controls *and* states are decisions variables.