Introduction to Optimization

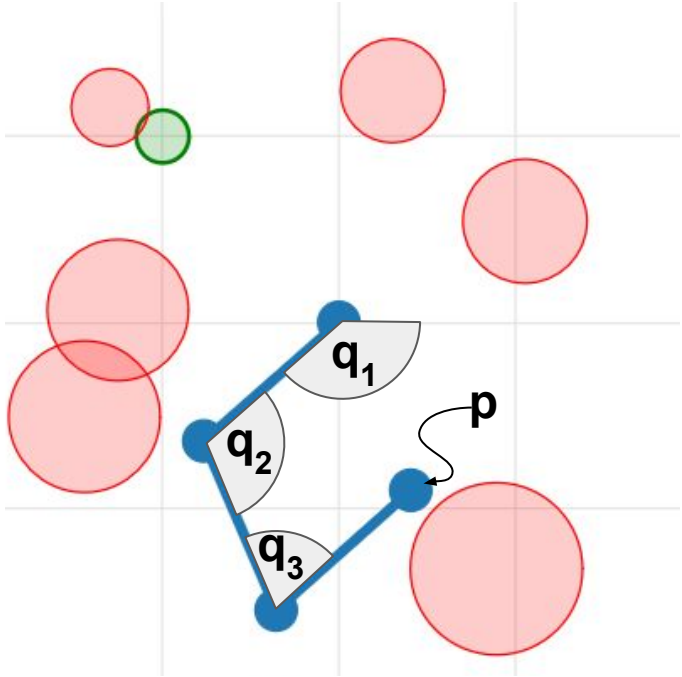
Lecture 7

Overview

- Introduction to and motivation for optimization
  - Reminder: Planning in continuous state spaces
  - What is optimization (“mathematical programming”)?
  - Demo: Practical application of optimization
- Formulating solutions to optimal control problems using optimization
- Classes of optimization problems
  - Convex programming
  - Linear programming
  - Mixed-integer linear programming
  - Quadratic programming

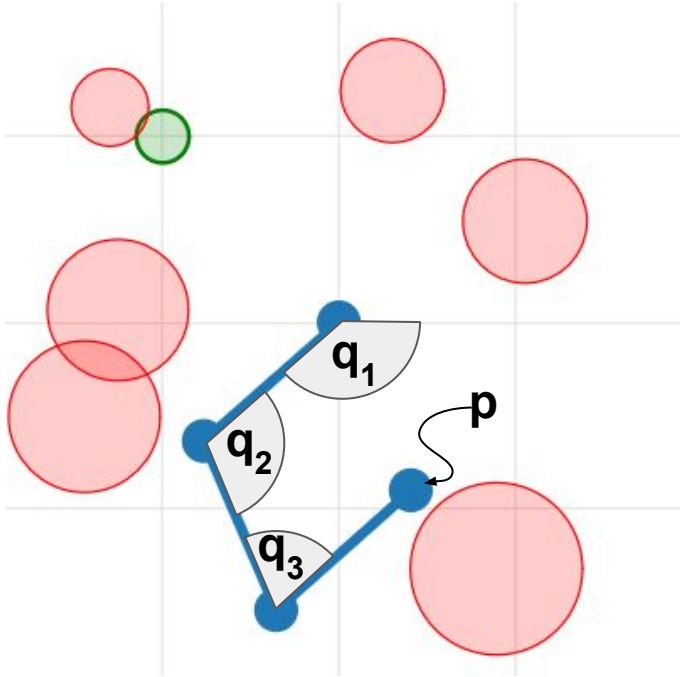
Example: Planar Manipulator Planning

- **Configuration space:**
  Joint angles (q_1, q_2, q_3)
- **Task:**
  p should reach the green goal region along a collision-free path
- **What tools have we seen in class so far to solve this problem?**



Example: Planar Manipulator Planning

- **Configuration space:**
  Joint angles (q_1, q_2, q_3)
- **Task:**
  p should reach the green goal region along a collision-free path
- **What tools have we seen in class so far to solve this problem?**
  - **RRT/PRM:** Good fit for continuous state space
  - **Dijkstra's/A*:** Optimality guarantees, but requires discretization



Demo: Planar Manipulator Planning using RRT

Live notebook demo

- What are the limitations we observed when using RRT to find a solution path?

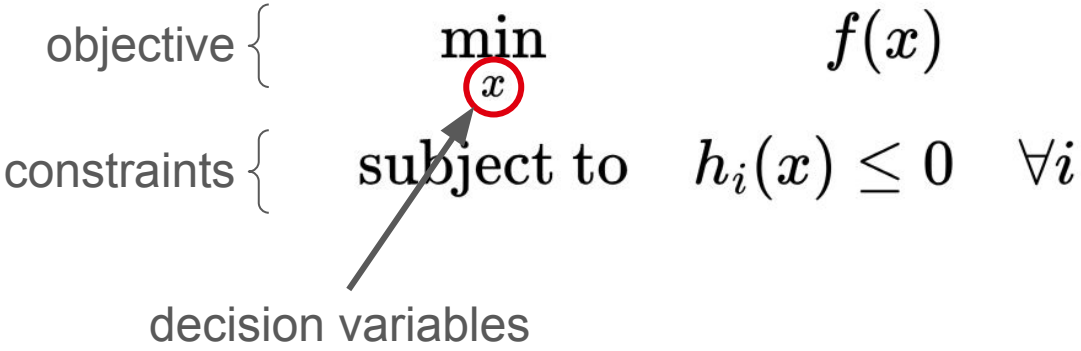
Mathematical Programming

- Goal: solve an optimization problem (selection of a best element) from some set according to some criteria

objective
$$\min_{x} \quad f(x)$$
constraints
$$\text{subject to} \quad h_i(x) \leq 0 \quad \forall i$$

Mathematical Programming

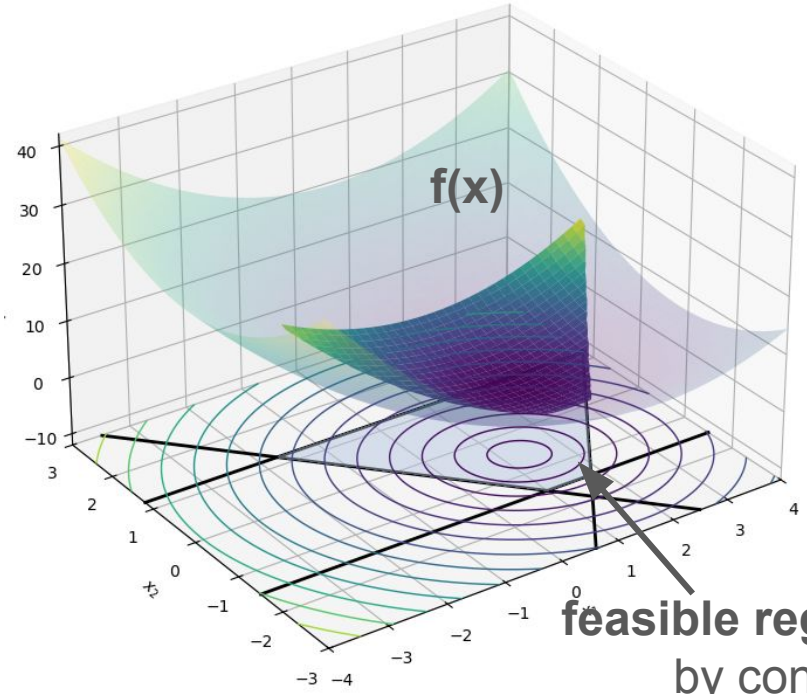
- Goal: solve an optimization problem (selection of a best element) from some set according to some criteria



Mathematical Programming

- Goal: solve an optimization problem (selection of a best element) from some set according to some criteria

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & h_i(x) \leq 0 \quad \forall i \end{aligned}$$



feasible region induced by constraints

Mathematical Programming

Let’s return to the manipulator motion planning problem.

- If we want **smooth motion**, what could be our objective function?
- How can we formulate an optimization problem to achieve this?
  - What is the objective function?
  - What are the constraints?

Demo: Planar Manipulator Planning using Optimization

Live notebook demo

Planar Manipulator Planning using Optimization

- **Objective:** minimize average torque
  - Implemented by minimizing each joint's acceleration at each timestep
- **Constraints**
  - **Two endpoints:** start at the current location and reach the goal
  - **Obstacles**
  - …anything else missing?

Planar Manipulator Planning using Optimization

- **Objective:** minimize energy usage
  - Implemented by minimizing acceleration at each timestep
- **Constraints**
  - **Two endpoints:** start at the current location and reach the goal
  - **Obstacles**
  - …anything else missing?
    - Joint angle limits
    - Motor torque limits
    - Joint speed limits

Planar Manipulator Planning using Optimization

- Other ways to formulate the optimization problem?
  For example:

  …could we choose a different set of decision variables?

  …is there a better way to parameterize the path?

Planar Manipulator Planning using Optimization

- Other ways to formulate the optimization problem?
  For example:

  …could we choose a different set of decision variables?
  →motor torque or acceleration as decision variables

  …is there a better way to parameterize the path?
  →use continuous functions (e.g. polynomials/splines)

- In practice, there are often many different choices with different tradeoffs

Motivation: Optimal Control

Search methods like A* or RRT can efficiently search state spaces, however…

- Real-world robots often do not have the ability to directly control their state
  - e.g. for an autonomous car, we cannot directly control its position, instead, we control the accelerator pedal and the steering angle
- The motion of the robot follows some equations of motion
  - in the autonomous car example, the accelerator pedal may control e.g. the second derivative of the traversed distance, and the steering angle controls the derivative of the car's heading

Motivation: Optimal Control

Search methods like A* or RRT can efficiently search state spaces, however…

- Real-world robots often do not have the ability to directly control their state
  - e.g. for an autonomous car, we cannot directly control its position, instead, we control the accelerator pedal and the steering angle
- The motion of the robot follows some equations of motion
  - in the autonomous car example, the accelerator pedal may control e.g. the second derivative of the traversed distance, and the steering angle controls the derivative of the car's heading
- Therefore, it is generally useful to consider the **state equation**

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$$

time derivative of state state control input time

Formulating Optimal Control Problems Using Optimization

Mathematical programming provides a flexible way to specify optimal control problems

$$\begin{aligned} \min \quad & ??? \\ ??? & \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \quad \forall t \in [t_0, t_f] \leftarrow \text{dynamics} \end{aligned}$$

Formulating Optimal Control Problems Using Optimization

Mathematical programming provides a flexible way to specify optimal control problems

$$\min_{\mathbf{u}} \quad ???$$

$$\text{subject to} \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \quad \forall t \in [t_0, t_f] \leftarrow \text{dynamics}$$

Formulating Optimal Control Problems Using Optimization

Mathematical programming provides a flexible way to specify optimal control problems

$$\min_{\mathbf{u}} \quad ???$$

$$\text{subject to} \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \quad \forall t \in [t_0, t_f]$$

← dynamics

$$\mathbf{x}(t_0) = \mathbf{x}_0$$
$$\mathbf{x}(t_f) = \mathbf{x}_f$$

← endpoint constraints

Formulating Optimal Control Problems Using Optimization

Mathematical programming provides a flexible way to specify optimal control problems

$$\min_{\mathbf{u}} \quad ???$$

$$\text{subject to} \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \quad \forall t \in [t_0, t_f]$$

← dynamics

$$\mathbf{x}(t_0) = \mathbf{x}_0$$
$$\mathbf{x}(t_f) = \mathbf{x}_f$$

← endpoint constraints

$$\mathbf{x}(t) \in \mathcal{X}_{\text{safe}} \quad \forall t \in [t_0, t_f]$$

← additional state constraints (e.g. obstacle avoidance)

Formulating Optimal Control Problems Using Optimization

Mathematical programming provides a flexible way to specify optimal control problems

$$\min_{\mathbf{u}} \quad ???$$

$$\text{subject to} \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \quad \forall t \in [t_0, t_f]$$

← dynamics

$$\mathbf{x}(t_0) = \mathbf{x}_0$$
$$\mathbf{x}(t_f) = \mathbf{x}_f$$

← endpoint constraints

$$\mathbf{x}(t) \in \mathcal{X}_{\text{safe}} \quad \forall t \in [t_0, t_f]$$

← additional state constraints (e.g. obstacle avoidance)

$$\mathbf{u}(t) \in \mathcal{U}_{\text{feas}} \quad \forall t \in [t_0, t_f]$$

← control limits (e.g. actuator torque limits)

Formulating Optimal Control Problems Using Optimization

Mathematical programming provides a flexible way to specify optimal control problems

$$\min_{\mathbf{u}} \quad \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t))\, dt$$

cost
(e.g. energy)

$$\text{subject to} \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \quad \forall t \in [t_0, t_f]$$

dynamics

$$\mathbf{x}(t_0) = \mathbf{x}_0$$
$$\mathbf{x}(t_f) = \mathbf{x}_f$$

endpoint constraints

$$\mathbf{x}(t) \in \mathcal{X}_{\text{safe}} \quad \forall t \in [t_0, t_f]$$

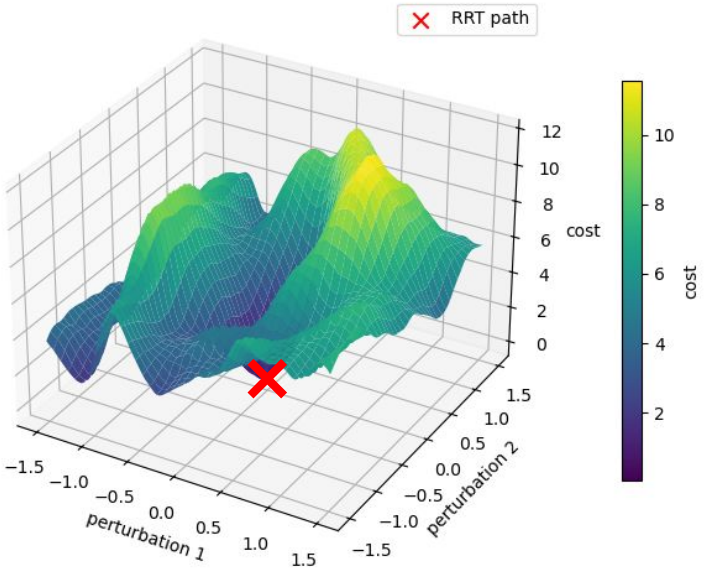
additional state constraints
(e.g. obstacle avoidance)

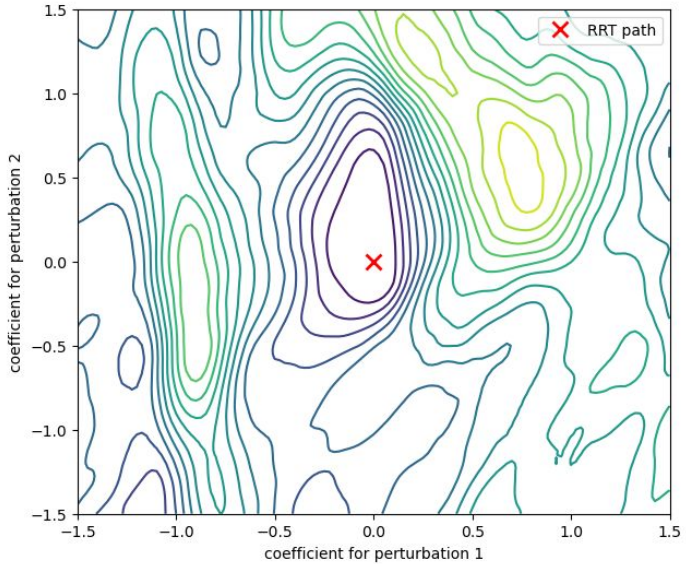
$$\mathbf{u}(t) \in \mathcal{U}_{\text{feas}} \quad \forall t \in [t_0, t_f]$$

control limits
(e.g. actuator torque limits)

Challenges with Nonlinear Programming

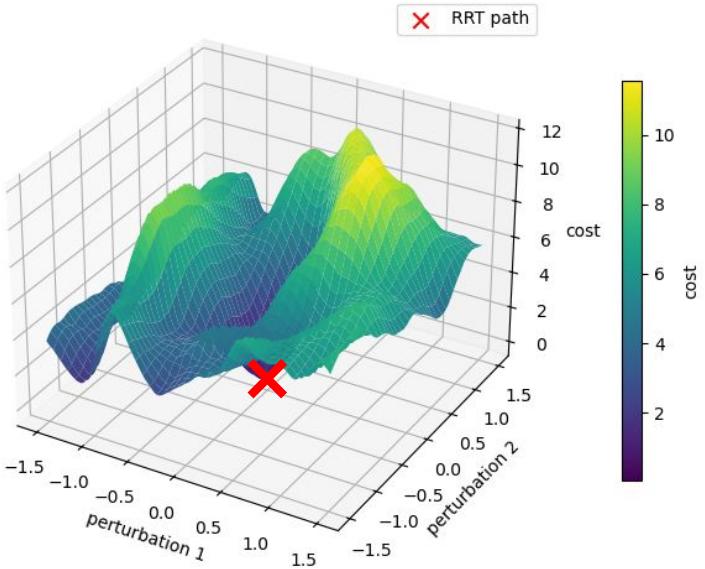
- We formulated our manipulator path smoothing optimization as a general **unconstrained, nonlinear** program (“NLP”)
- Our chosen cost function is very complicated with lots of bad local minima

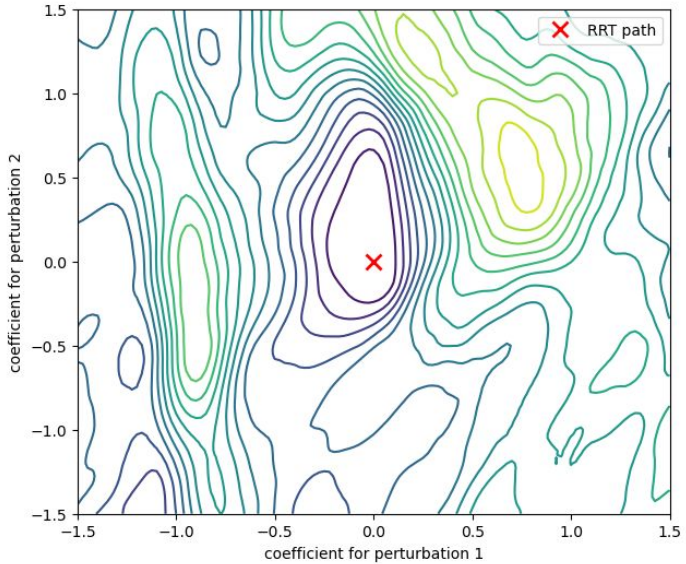




Challenges with Nonlinear Programming

- Our approach only succeeded because we initialized the optimization with a good **seed**: the trajectory computed via RRT.



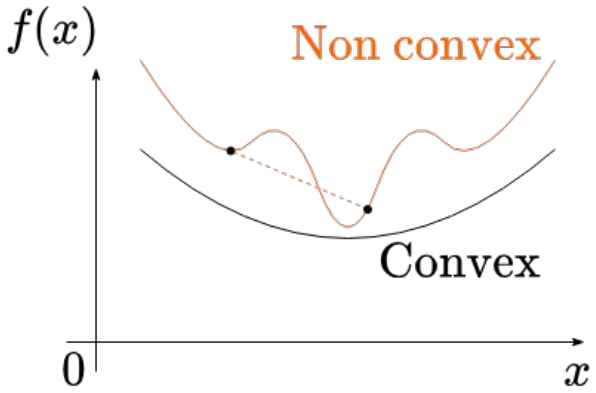


Classes of Optimization Problems

- As seen on the manipulator example, general nonlinear programs can have extremely complicated cost landscapes and solution techniques may lack any optimality or safety guarantees
- Knowledge about certain **structure** in the optimization problem can enable more efficient solvers and allow making statements about optimality of solutions

Classes of Optimization Problems

- **Convex** vs. **nonconvex**
  If the objective is convex, a local optimum is also a global optimum
  Otherwise, need to differentiate between local and global optima



Classes of Optimization Problems

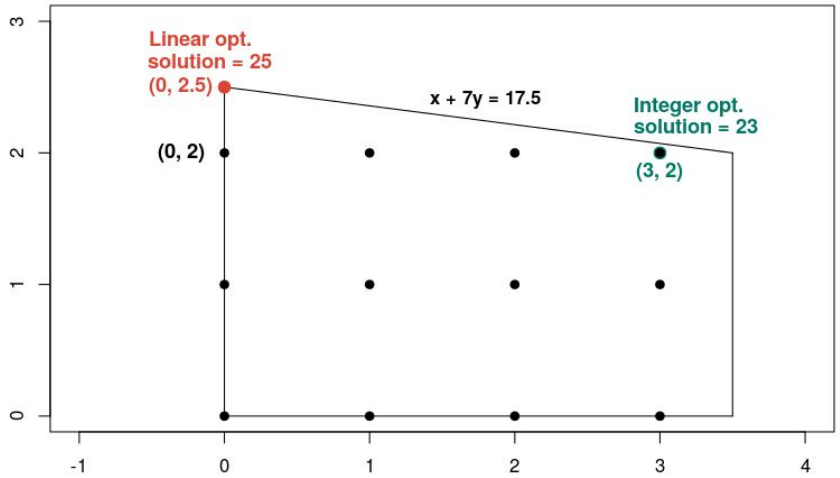
- **Convex** vs. **nonconvex**
  If the objective is convex, a local optimum is also a global optimum
  Otherwise, need to differentiate between local and global optima

- **Continuous** vs. **discrete** (or **mixed**)
  Solution techniques vary or can be combined to solve problems involving both discrete decisions and continuous variables

Classes of Optimization Problems

- **Convex** vs. **nonconvex**
  If the objective is convex, a local optimum is also a global optimum
  Otherwise, nee *optima*

- **Continuous** vs
  Solution techniq *blems involving both*
  discrete decisio



Classes of Optimization Problems

- **Convex** vs. **nonconvex**
  If the objective is convex, a local optimum is also a global optimum
  Otherwise, need to differentiate between local and global optima

- **Continuous** vs. **discrete** (or **mixed**)
  Solution techniques vary or can be combined to solve problems involving both discrete decisions and continuous variables

- **Constrained** vs. **unconstrained**
  Hard constraints lead to different solutions compared to unconstrained problem. Solution techniques can explicitly consider constraints or translate from a constrained formulation to an unconstrained one.

Classes of Optimization Problems

- **Convex** vs. **nonconvex**

  If the objective is convex, a local optimum is also a global optimum

- **C** r **mixed**)

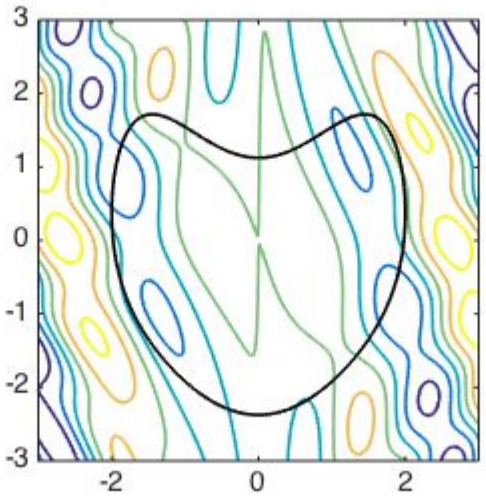
  S can be cor oth

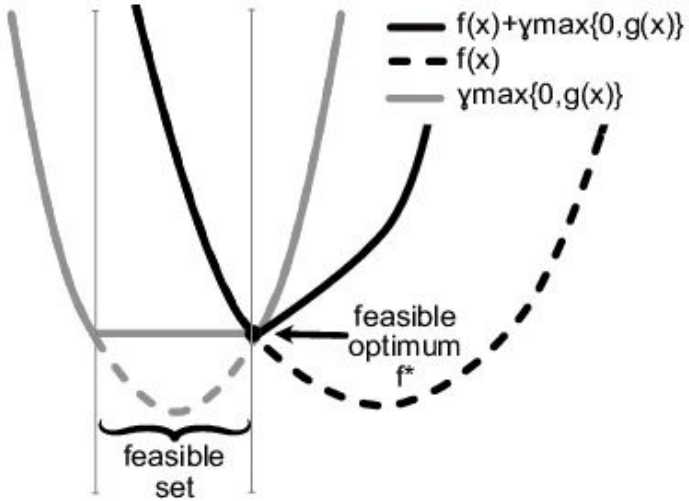
  a nuous vari

- **C** ined

  H erent solutions compared to unconstrained

  problem. Solution techniques can explicitly consider constraints or translate from a constrained formulation to an unconstrained one.





Convex Programming

objective
$$\min_x \quad f(x)$$

Inequality constraints
$$\text{subject to} \quad h_i(x) \leq 0 \quad \forall i$$

Equality constraints
$$a_j^T x = b_j \quad \forall j$$

- The cost and constraints functions are **convex**

$$h_i(\alpha x + \beta y) \leq \alpha h_i(x) + \beta h_i(y)$$

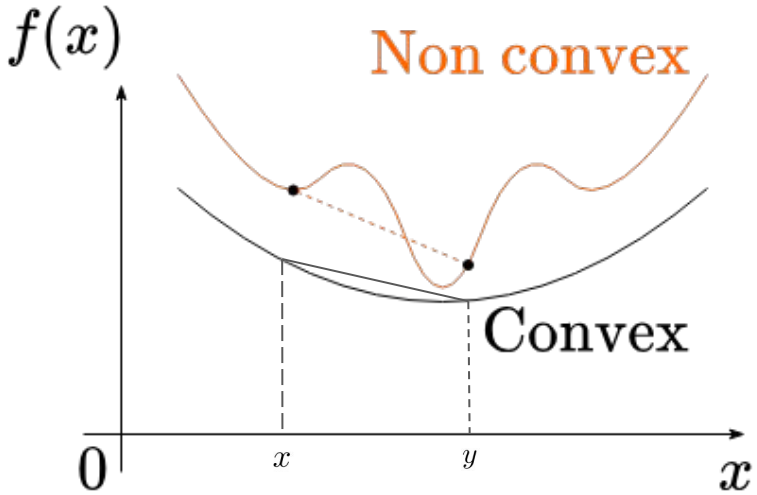
$$\alpha + \beta = 1, \ \alpha \geq 0, \ \beta \geq 0$$

- Feasible set is also **convex**

Convex Programming

$$h_i(\alpha x + \beta y) \leq \alpha h_i(x) + \beta h_i(y)$$

$$\alpha + \beta = 1,\ \alpha \geq 0,\ \beta \geq 0$$

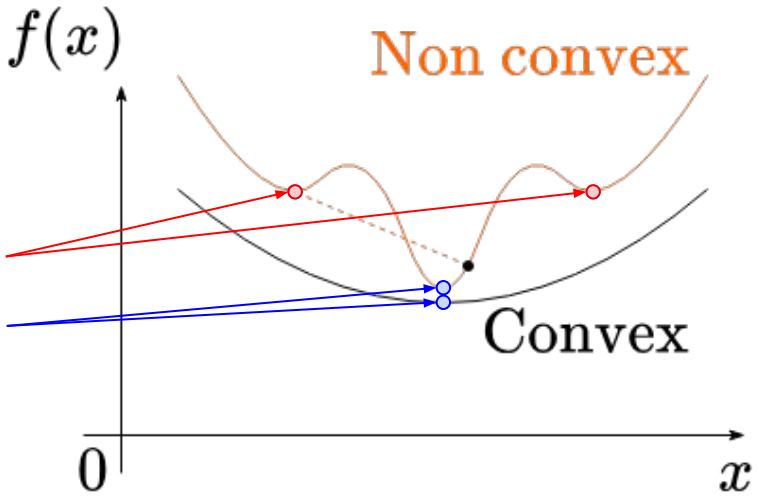


Convex Programming

- At convex programming, **any locally optimal point** is also **(globally) optimal.**

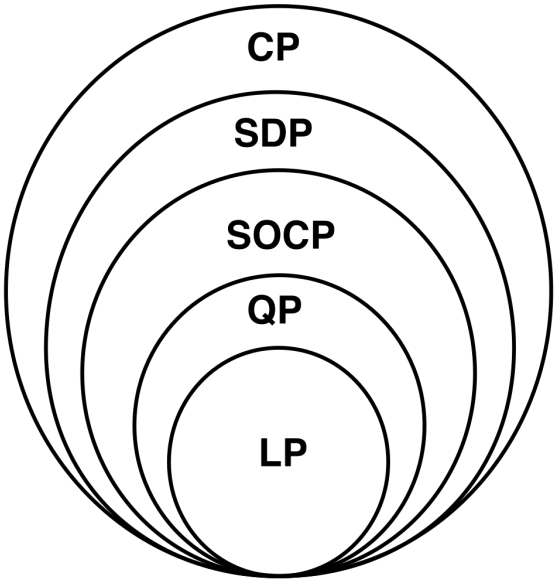
Local optimal point

Global optimum point



Convex Programming

- Classes of Convex Programming
  - **Linear Programming (LP)**
  - **Quadratic Programming (QP)**
  - Second Order Cone Programming (SOCP)
  - Semidefinite Programming (SDP)
  - Conic Programming (CP)
- We are going to overview **LP** and **QP**!



Linear Programming (LP)

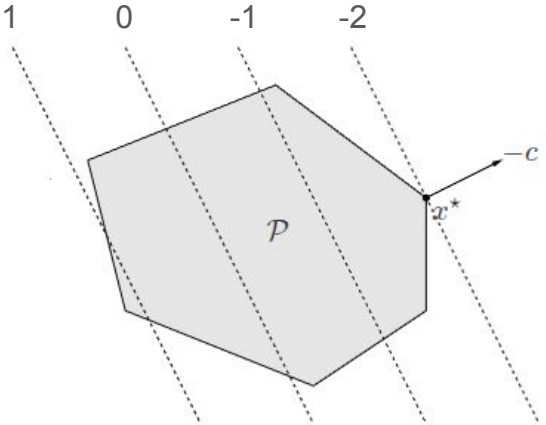
objective
$$\min_x \quad c^T x + d$$
Inequality constraints
$$\text{subject to} \quad Gx \leq h$$
Equality constraints
$$Ax = b$$

- Objective and all constraints are linear
- Linear Programming $\subset$ Convex Programming

Linear Programming (LP)

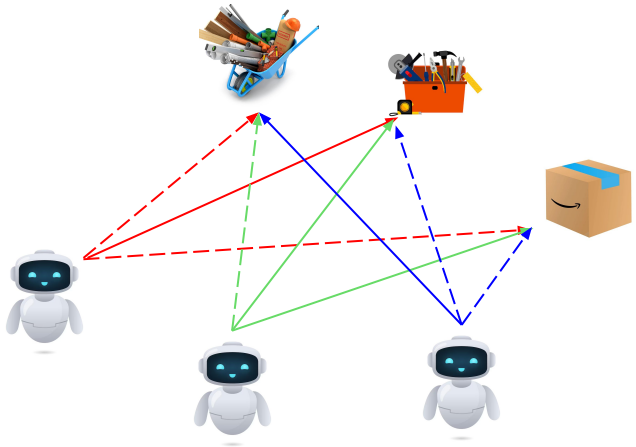
- The optimal solution is one of the intersections of constraints
- The set satisfying constraints is polyhedron, and the level curves are hyperplanes orthogonal to c
- We can find optimal solutions by searching intersection
  - (Simplex method) fast and scalable

$$\begin{aligned} \min_{x} \quad & c^T x + d \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$



Mixed-Integer Linear Programming (MILP)

- Example: Multi-robot task assignment
  - Goal: minimize total cost
  - # of tasks: m, # of robots: n
  - Each task is assigned by max. P robots
  - Each robot can do max. Q tasks



Mixed-Integer Linear Programming (MILP)

decision variables: $x_{ij} = \begin{cases} 1 & \text{robot i performs task j} \\ 0 & \text{otherwise} \end{cases}$ c_{ij} :cost for robot i to perform task j

Total cost: $\min_x \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$

subject to

Max. # of robots per task: $\sum_{i=1}^{n} x_{ij} \leq P$

Max. # of tasks per robot: $\sum_{j=1}^{m} x_{ij} \leq Q$

Decision variable bound: $0 \leq x_{ij} \leq 1 \quad \forall i, j$

Mixed-Integer Linear Programming (MILP)

What is different to **LP**?

decision variables: $x_{ij} = \begin{cases} 1 & \text{robot i performs task j} \\ 0 & \text{otherwise} \end{cases}$ c_{ij} :cost for robot i to perform task j

Total cost

$$\min_{x} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$$

Max. # of robots per task

$$\text{subject to} \quad \sum_{i=1}^{n} x_{ij} \leq P$$

Max. # of tasks per robot

$$\sum_{j=1}^{m} x_{ij} \leq Q$$

Decision variable bound

$$0 \leq x_{ij} \leq 1 \quad \forall i, j$$

Mixed-Integer Linear Programming (MILP)

What is different to **LP**?

Decision variables are not continuous

decision variables: $x_{ij} = \begin{cases} 1 & \text{robot i performs task j} \\ 0 & \text{otherwise} \end{cases}$

c_{ij} :cost for robot i to perform task j

Total cost

$$\min_x \quad \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$$

Max. # of robots per task

$$\text{subject to} \quad \sum_{i=1}^{n} x_{ij} \leq P$$

Max. # of tasks per robot

$$\sum_{j=1}^{m} x_{ij} \leq Q$$

Decision variable bound

$$0 \leq x_{ij} \leq 1 \quad \forall i, j$$

Quadratic Programming (QP)

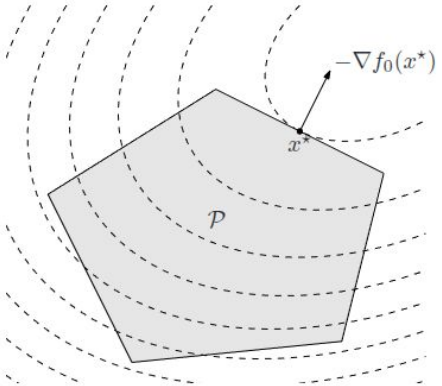
Quadratic Objective

Linear inequality constraints

Linear Equality Constraints

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T Q x + c^T x \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$

- Polyhedron feasible set



Quadratic Programming (QP)

Quadratic Objective

Linear inequality constraints

Linear Equality Constraints

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^T Q x + c^T x \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$

- Simplest example: least squares

$$\begin{aligned} \|Ax - b\|^2 &= (Ax - b)^T (Ax - b) \\ &= x^T A^T A x - 2b^T A x + b^T b \\ &= x^T \underbrace{A^T A}_{Q} x + \underbrace{(-2A^T b)^T}_{c^T} x + \underbrace{b^T b}_{\text{const.}} \end{aligned}$$

Quadratic Programming (QP)

- (Very) simplified optimal control problem
  - Goal: input sequence of dynamics to reach goal state + to minimize system effort (energy)
  - Initial state is given
  - State and input is bounded

$$
\begin{aligned}
\min_{u_0,\ldots,u_{N-1}} \quad & \sum_{k=0}^{N-1} u_k^T R u_k + (x_N - x_{\text{goal}})^T Q_f (x_N - x_{\text{goal}}) \\
\text{subject to} \quad & x_{k+1} = A x_k + B u_k, \quad k = 0, \ldots, N-1 \\
& x_{\min} \leq x_k \leq x_{\max} \\
& u_{\min} \leq u_k \leq u_{\max} \\
& x_0 = \bar{x}
\end{aligned}
$$

Quadratic Programming (QP)

$$
\begin{aligned}
\min_{u_0,\ldots,u_{N-1}} \quad & \sum_{k=0}^{N-1} u_k^T R u_k + (x_N - x_{\text{goal}})^T Q_f (x_N - x_{\text{goal}}) \\
\text{subject to} \quad & x_{k+1} = A x_k + B u_k, \quad k = 0, \ldots, N-1 \\
& x_{\min} \leq x_k \leq x_{\max} \\
& u_{\min} \leq u_k \leq u_{\max} \\
& x_0 = \bar{x}
\end{aligned}
$$

Quadratic Programming (QP)

$$\min_{u_0,\ldots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^T R u_k + (x_N - x_{\text{goal}})^T Q_f (x_N - x_{\text{goal}})$$

Dynamics: subject to $x_{k+1} = Ax_k + Bu_k, \quad k = 0, \ldots, N-1$

State & input bound:

$$x_{\min} \leq x_k \leq x_{\max}$$

$$u_{\min} \leq u_k \leq u_{\max}$$

Initial state: $x_0 = \bar{x}$

Quadratic Programming (QP)

Energy

Distance to final state

$$\min_{u_0,\ldots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^T R u_k + (x_N - x_{\text{goal}})^T Q_f (x_N - x_{\text{goal}})$$

Dynamics

$$\text{subject to} \quad x_{k+1} = Ax_k + Bu_k, \quad k = 0, \ldots, N-1$$

State & input bound

$$x_{\min} \leq x_k \leq x_{\max}$$

$$u_{\min} \leq u_k \leq u_{\max}$$

Initial state

$$x_0 = \bar{x}$$