Convex Optimization & Optimization Tools

AA 203 Recitation #1

April 9th, 2021

Agenda

Preliminaries

- Why study Convex Optimization?
- Convex Sets & Convex Functions
- Convex Programming
- Linear Matrix Inequalities

Optimization Models and Tools

- Solvers
- Linear Programming
- Quadratic Programming

CVXPY: Convex Optimization in Python

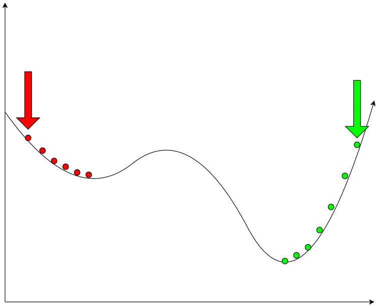
- Least Squares
- Discrete LQR

Preliminaries

Why study Convex Optimization?

Observation 1: Iterative methods like Gradient method and Newton Method can find local minima.

Observation 2: These methods can also get trapped in local minima and thus fail to converge to the *global* minima.



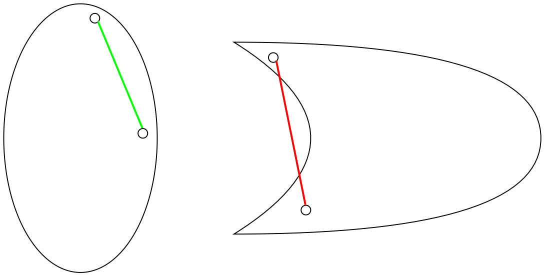
Observation 3: This issue doesn't show up for convex problems. For convex optimization problems, every locally optimal solution is also globally optimal.

Convex Sets

Definition (Convex Set)

A set $S \subset \mathbb{R}^d$ is convex if and only if: for any $x, y \in S$ and any $\alpha \in [0, 1]$, we also have $\alpha x + (1 - \alpha)y \in S$.

Examples:



Yes!   No

Convex Functions

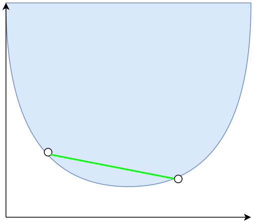
Definition (Convex Functions)

A function $f : S \to \mathbb{R}$ over a convex set $S \subset \mathbb{R}^d$ is convex if the set

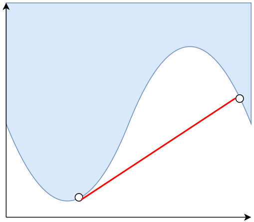
$$\text{epigraph}(f) := \left\{(x, y) \in \mathbb{R}^{d+1} : x \in S, y \in \mathbb{R} \text{ and } y \geq f(x)\right\} \text{ is convex.}$$

Equivalently: If the chord between $f(x_1)$ and $f(x_2)$ overestimates f between x_1 and x_2.

Examples:



Yes!



No

Convex Program

Definition (Convex Program)

A convex program (aka convex optimization problem) is a minimization problem of a convex function over a convex set:

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } x \in S \end{aligned}$$

where S is a convex set and $f : S \to \mathbb{R}$ is a convex function.

Definition (Local Minimum)

For an optimization problem $\min_{x \in S} f(x)$, a point x^* is a local minimum if there exists some $\epsilon > 0$ so that for every $x \in S$ with $||x - x^*||_2 \leq \epsilon$, $f(x^*) \leq f(x)$.

Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ *be a convex program. If* x^* *is a local minimum, then* $f(x^*) \leq f(x)$ *for every* $x \in S$. *In other words,* x^* *is a global minimum.*

Proof Idea: (by contradiction) Suppose x^* is a local but not global minimum.

Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ *be a convex program. If* x^* *is a local minimum, then* $f(x^*) \leq f(x)$ *for every* $x \in S$*. In other words,* x^* *is a global minimum.*

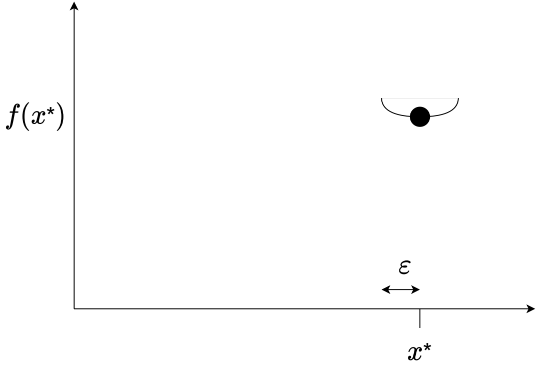
Proof Idea: (by contradiction) Suppose x^* is a local but not global minimum.

Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ *be a convex program. If* x^* *is a local minimum, then* $f(x^*) \leq f(x)$ *for every* $x \in S$*. In other words,* x^* *is a global minimum.*

Proof Idea: (by contradiction) Suppose x^* is a local but not global minimum.

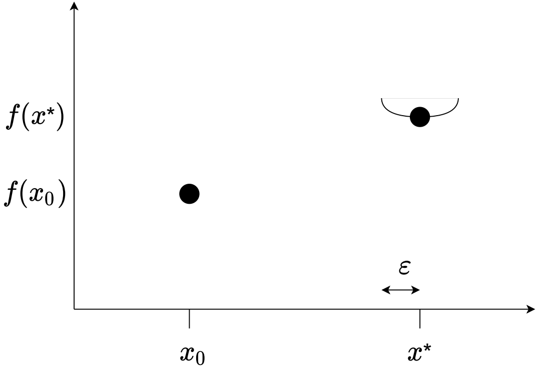


Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ *be a convex program. If* x^* *is a local minimum, then* $f(x^*) \leq f(x)$ *for every* $x \in S$*. In other words,* x^* *is a global minimum.*

Proof Idea: (by contradiction) Suppose x^* is a local but not global minimum.

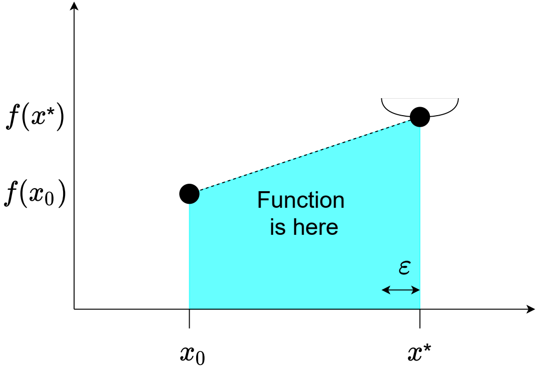


Convex Program: Local Optima are Global Optima

Theorem (Equivalence of Local and Global Optima)

Let $\min_{x \in S} f(x)$ *be a convex program. If* x^* *is a local minimum, then* $f(x^*) \leq f(x)$ *for every* $x \in S$*. In other words,* x^* *is a global minimum.*

Proof Idea: (by contradiction) Suppose x^* is a local but not global minimum.



Convex Program: Local Optima are Global Optima

Proof: (by contradiction) Suppose x^* is a local but not global minimum.

Since x^* is a local optima, there exists $\epsilon > 0$ so that $f(x^*) \leq f(x)$ for all $x \in S$, $||x - x^*||_2 \leq \epsilon$.

Since x^* is not a global minimum, we can find $x_0 \in S$ where $f(x_0) < f(x^*)$.

Since S is convex, $\alpha x^* + (1 - \alpha)x_0 \in S$ for every $\alpha \in [0, 1]$.

Note that $f((1 - \alpha)x^* + \alpha x_0) \leq (1 - \alpha)f(x^*) + \alpha f(x_0) < f(x^*)$.

Pick $\alpha' = \frac{\epsilon}{2||x^* - x_0||_2}$ and set $x' := (1 - \alpha')x^* + \alpha' x_0$.

We have $f(x') < f(x^*)$ and $||x^* - x'||_2 \leq \epsilon$.

This contradicts the fact that x^* is a local minimum. □

Convex Program: Local Optima are Global Optima

The result relies on both S, f being convex.

S not convex examples: Optimal Control of Nonlinear Systems, Integer Programming.

f not convex examples: Maximum Likelihood for Gaussian Mixtures, Training Neural Networks.

Linear Matrix Inequalities (LMI)

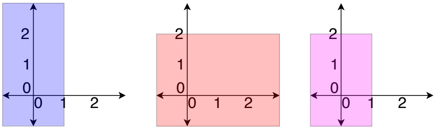
Goal: Introduce notation to efficiently express convex constraints.

Definition (Vector Inequality)

For $x, y \in \mathbb{R}^d$, we use $x \preceq y$ to denote that x is **element-wise less than** y. Concretely, $x \preceq y$ if for every $1 \leq i \leq d$, $x_i \leq y_i$.

Example: $x \succeq 0$ means all entries of x are non-negative.

We can also use inequalities to define sets: $\{x : x \preceq y\}$.



Example: $\left\{ x : x \preceq \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\}$ $\quad x_1 \leq 1 \quad x_2 \leq 2 \quad x \preceq (1, 2)^\top$

Matrix Inequalities

Definition (Positive Semidefinite Matrices)

We say a matrix $A \in \mathbb{R}^{d \times d}$ is positive semidefinite if $x^\top Ax \geq 0$ for every $x \in \mathbb{R}^d$. The relation $A \succeq 0$ is often used to denote positive semidefiniteness of A.

Definition (Matrix Inequalities)

We say $A \preceq B$ if $0 \preceq B - A$, i.e. $B - A$ is positive semidefinite.

The set $\{A : A \succeq 0\}$ is a convex set (in fact, it is a cone). Optimizations of convex functions over this set are **Semidefinite Programs** (SDP).

Applications of SDPs: Sum of Squares Programming, Lyapunov Stability analysis, approximation algorithms for combinatorial optimization.

Optimization Models and Tools

Optimization Models and Tools

Common Optimization Models

- Linear Programming (LP).
- Quadratic Programming (QP).
- Semidefinite Programming (SDP).
- Convex Programming (CP).
- Mixed-Integer Linear Programming (IP).

Optimization Software

- CVXPY (LP, QP, SDP, CP, IP).
- CPLEX (LP, QP, IP).

CVXPY Examples

- Least Squares.
- Discrete Linear Quadratic Regulator.

Linear Programming

Goal: Minimize a linear function subject to linear equality and inequality constraints. Mathematically,

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \ & c^\top x \\ \text{subject to} \ & Ax \preceq b \\ & A_{eq} x = b_{eq}. \end{aligned}$$

A linear programming instance is specified by $c \in \mathbb{R}^n, b \in \mathbb{R}^p, A \in \mathbb{R}^{p \times n}, b_{eq} \in \mathbb{R}^q, A_{eq} \in \mathbb{R}^{q \times n}$.

Software:

CPLEX: `x = cplexlp(c, A, b, Aeq, beq)`.

MATLAB: `x = linprog(c, A,b, Aeq, beq)`.

LP Example #1 - Multi-Robot Task Allocation

Consider a scenario where n robots $\{r_1, r_2, ..., r_n\}$ must collectively perform m tasks $\{t_1, t_2, ..., t_m\}$.

Each robot can perform at most 1 task.

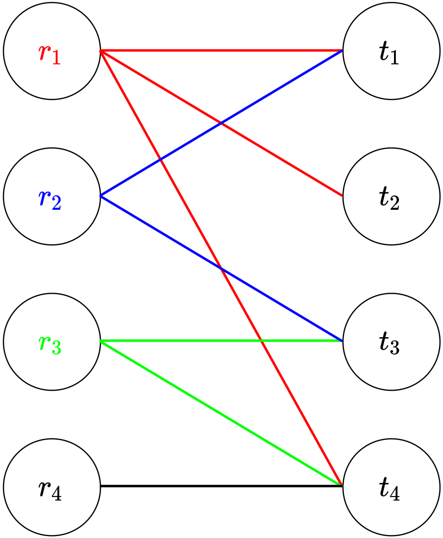
Each task requires only 1 robot.

u_{ij} is the utility achieved when r_i performs t_j.

Objective: Match robots to tasks to maximize the total utility.

LP Example #1 - Multi-Robot Task Allocation

Graph Representation:
Construct a graph where the vertices are $\{r_1, ..., r_n, t_1, ..., t_m\}$.
Include an edge between r_i and t_j of weight u_{ij} if $u_{ij} > 0$.
Finding the maximum utility matching becomes a maximum weight bipartite matching problem in this graph!



LP Example #1 - Multi-Robot Task Allocation

Cast the maximum weight bipartite matching problem as a linear program:

Decision variable: $x \in \mathbb{R}^{mn}$, where x_{ij} determines whether or not r_i will perform t_j.

LP Example #1 - Multi-Robot Task Allocation

Cast the maximum weight bipartite matching problem as a linear program:

Decision variable: $x \in \mathbb{R}^{mn}$, where x_{ij} determines whether or not r_i will perform t_j.

LP Example #1 - Multi-Robot Task Allocation

Cast the maximum weight bipartite matching problem as a linear program:

Decision variable: $x \in \mathbb{R}^{mn}$, where x_{ij} determines whether or not r_i will perform t_j.

$$\underset{x \in \mathbb{R}^{mn}}{\text{maximize}} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij} x_{ij} \tag{1}$$

LP Example #1 - Multi-Robot Task Allocation

Cast the maximum weight bipartite matching problem as a linear program:

Decision variable: $x \in \mathbb{R}^{mn}$, where x_{ij} determines whether or not r_i will perform t_j.

$$\underset{x \in \mathbb{R}^{mn}}{\text{maximize}} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \sum_{j=1}^{m} x_{ij} \leq 1 \text{ for all } 1 \leq i \leq n \tag{2}$$

LP Example #1 - Multi-Robot Task Allocation

Cast the maximum weight bipartite matching problem as a linear program:

Decision variable: $x \in \mathbb{R}^{mn}$, where x_{ij} determines whether or not r_i will perform t_j.

$$\underset{x \in \mathbb{R}^{mn}}{\text{maximize}} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \sum_{j=1}^{m} x_{ij} \leq 1 \text{ for all } 1 \leq i \leq n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1 \text{ for all } 1 \leq j \leq m \tag{3}$$

LP Example #1 - Multi-Robot Task Allocation

Cast the maximum weight bipartite matching problem as a linear program:

Decision variable: $x \in \mathbb{R}^{mn}$, where x_{ij} determines whether or not r_i will perform t_j.

$$\underset{x \in \mathbb{R}^{mn}}{\text{maximize}} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \sum_{j=1}^{m} x_{ij} \leq 1 \text{ for all } 1 \leq i \leq n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1 \text{ for all } 1 \leq j \leq m \tag{3}$$

$$x \succeq 0.$$

LP Example #1 - Multi-Robot Task Allocation

Cast the maximum weight bipartite matching problem as a linear program:

Decision variable: $x \in \mathbb{R}^{mn}$, where x_{ij} determines whether or not r_i will perform t_j.

$$\underset{x \in \mathbb{R}^{mn}}{\text{maximize}} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{m} x_{ij} \leq 1 \text{ for all } 1 \leq i \leq n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1 \text{ for all } 1 \leq j \leq m \tag{3}$$

$$x \succeq 0.$$

(2) ensures each robot performs at most one task, (3) ensures that no task is assigned to more than 1 robot.

LP Example #1 - Multi-Robot Task Allocation

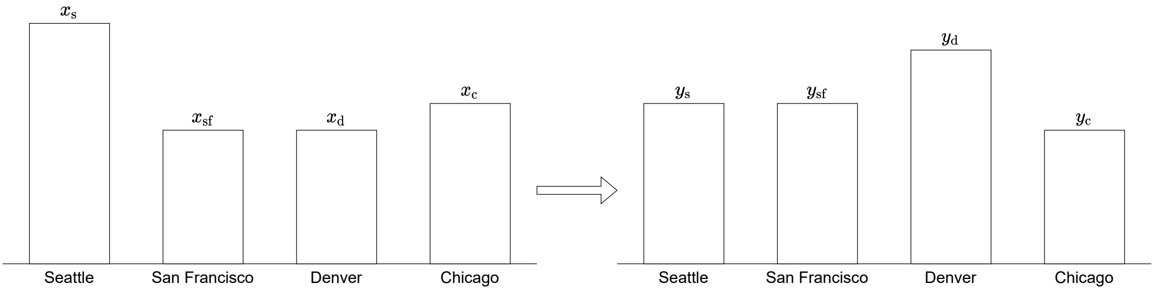
Even though fractional solutions are feasible for (1), we can always find an optimal solution which is integral $x^* \in \{0, 1\}^{mn}$!

If $x_{ij}^* = 1$, have robot r_i perform task t_j.

LP Example #2 - Discrete Optimal Transport

Goal: Re-distribute supplies between warehouses $w_1, w_2, \ldots, w_n$ to align with regional demand. Perform redistribution with minimum cost (cost = volume $\times$ distance).

Initial supply distribution: $x \in \mathbb{R}^n_+$,
Target supply distribution: $y \in \mathbb{R}^n_+$,
with $||x||_1 = ||y||_1 = 1$.



LP Example #2 - Discrete Optimal Transport

Decision variable: $P \in \mathbb{R}^{n \times n}$ where P_{ij} is the volume of supplies we send from w_i to warehouse w_j.

LP Example #2 - Discrete Optimal Transport

Decision variable: $P \in \mathbb{R}^{n \times n}$ where P_{ij} is the volume of supplies we send from w_i to warehouse w_j.

	Seattle	San Francisco	Denver	Chicago
Seattle	$S \to S$	$S \to SF$	$S \to D$	$S \to C$
San Francisco	$SF \to S$	$SF \to SF$	$SF \to D$	$SF \to C$
Denver	$D \to S$	$D \to SF$	$D \to D$	$D \to C$
Chicago	$C \to S$	$C \to SF$	$C \to D$	$C \to C$

LP Example #2 - Discrete Optimal Transport

Decision variable: $P \in \mathbb{R}^{n \times n}$ where P_{ij} is the volume of supplies we send from w_i to warehouse w_j.

	Seattle	San Francisco	Denver	Chicago	Row Sum
Seattle	$S \to S$	$S \to SF$	$S \to D$	$S \to C$	x_{s}
San Francisco	$SF \to S$	$SF \to SF$	$SF \to D$	$SF \to C$	x_{sf}
Denver	$D \to S$	$D \to SF$	$D \to D$	$D \to C$	x_{d}
Chicago	$C \to S$	$C \to SF$	$C \to D$	$C \to C$	x_{c}
Column Sum	y_{s}	y_{sf}	y_{d}	y_{c}	

LP Example #2 - Discrete Optimal Transport

Denote $d(w_i, w_j)$ as the distance between w_i and w_j.
The Optimal Transport problem is the following LP:

$$\begin{aligned} \underset{P \in \mathbb{R}^{n \times n}}{\text{minimize}} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} d(w_i, w_j) P_{ij} \\ \text{s.t.} \quad & P\mathbb{1} = x && (4) \\ & P^\top \mathbb{1} = y && (5) \\ & P_{ij} \geq 0 \text{ for all } 1 \leq i, j \leq n. && (6) \end{aligned}$$

Where (4) and (5) enforce initial and terminal conditions respectively.

LP Example #2 - Discrete Optimal Transport

Remarks:

Can view $P \in \mathbb{R}^{n \times n}$ as a joint distribution over (W_1, W_2), where

the marginal distribution of W_1 is x,

the marginal distribution of W_2 is y.

The optimal transportation cost is a distance function for distributions, known as the *Wasserstein Distance*.

Linear Programming - Properties

Linear programs can be solved efficiently (millions of variables and constraints); They are among the easiest convex optimization problems to solve.

There are many applications: Pattern planning, minimum weight matching, multi-commodity maximum flow, production planning, etc.

Definition (Extreme Point)

Given a convex set S, a point x is called extreme if it cannot be written as a convex combination of other points in S.

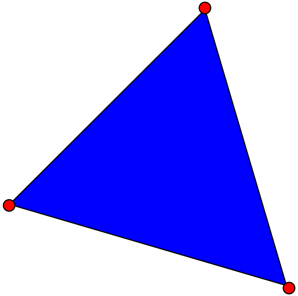
As a consequence, all points in S can be written as convex combinations of the extreme points of S.

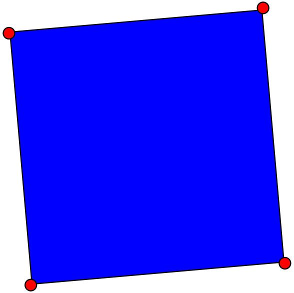
Linear Programming - Properties

For a linear program, the constraint set is comprised of linear equality and inequality constraints.

This means the constraint set is a polyhedron.

Extreme points of polyhedra are the corners.





Linear Programming - Properties

Theorem (Extreme Solutions of Linear Programs)

If a linear program $\min_{x \in P} c^\top x$ *has a finite optimal value (i.e. it has a non-empty solution set), then the solution set contains at least one extreme point of* P.

Proof: Let $x^* \in P$ be an optimal solution.

Let E_P be the set of extreme points of P.

Since $x^* \in P$, we can write it as a convex combination of points in E_P.

Thus $x^* = \sum_{x \in E_P} \alpha_x x$ where $\sum_{x \in E_P} \alpha_x = 1$ and $\alpha_x \geq 0$.

Thus $c^\top x^* = \sum_{x \in E_P} \alpha_x c^\top x \geq \min_{x \in E_P} c^\top x$, since the minimum is always at most the average.

So there is some $x' \in E_P$ with $c^\top x' \leq c^\top x^*$.

Since x^* is a minimizer, x' must also be a minimizer.

Quadratic Programming

Goal: Minimize a quadratic function subject to linear constraints. Mathematically,

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \ & \frac{1}{2} x^\top H x + f^\top x \\ \text{subject to} \ & Ax \preceq b \\ & A_{eq} x = b_{eq} \end{aligned}$$

where $H \succeq 0$.

A quadratic programming instance is specified by
$f \in \mathbb{R}^n, H \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^p, A \in \mathbb{R}^{p \times n}, b_{eq} \in \mathbb{R}^q, A_{eq} \in \mathbb{R}^{q \times n}$.

Software:

CPLEX: `x = cplexqp(H, f, A, b, Aeq, beq)`.

MATLAB: `x = quadprog(H, f, A,b, Aeq, beq)`.

QP Example: Discrete LQR

Given a discrete linear dynamical system

$$x_{t+1} = Ax_t + Bu$$

The goal is to efficiently drive the state from x_0 to the origin. We incur a large cost if (a) the state is far from the origin or (b) we use a lot of control effort.

$$\frac{1}{2}x_T^\top Q_T x_T + \frac{1}{2}\sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t$$

There are also control effort constraints.

QP Example: Discrete LQR

The discrete Linear Quadratic Regulator (LQR) with control effort constraints u_{LB}, u_{UB} can be formulated as a QP.

$$
\begin{aligned}
\underset{u \in \mathbb{R}^T}{\text{minimize}} \ & \frac{1}{2} x_T^\top Q_T x_T + \frac{1}{2} \sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t \\
\text{subject to} \ & x_{t+1} = A x_t + B u_t \text{ for all } 0 \leq t \leq T-1 && (7) \\
& x_0 = \text{initial condition} && (8) \\
& u_{LB} \preceq u_t \preceq u_{UB} \text{ for all } 0 \leq t \leq T-1. && (9)
\end{aligned}
$$

CVXPY: Convex Optimization in Python

Problem Objects in CVXPY

Instantiate by specifying an objective function and constraints.

```
prob = cvx.Problem(objective, constraints)
```

Specify a decision variable `x = cvx.Variable(n)`.

The `objective` is an expression, i.e. a function of the decision variable.

The `constraints` is a list of constraint objects.

Use `prob.solve()` to solve the problem.

Use `prob.status` to see if the optimization was successful.

The solution can then be found at `x.value`

The objective value of the solution can be found at `prob.value`

Least Squares in CVXPY

Recall the Least squares problem:

$$\min_{x \in \mathbb{R}^m} ||Ax - b||_2^2$$

where $A \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^n$.

Problem setup

```
import numpy as np
import cvxpy as cvx

n = 10
m = 5

A = np.random.normal(0,1,(n,m))
b = np.random.normal(0,1,(n,))
```

Least Squares in CVXPY

Solving the problem

```
x = cvx.Variable(m)

objective = cvx.Minimize(cvx.sum_squares(A @ x - b))
constraints = []

prob = cvx.Problem(objective, constraints)
prob.solve()

print(prob.status)
print(prob.value) # optimal objective value
print(x.value) # get the optimal solution
```

Discrete LQR in CVXPY

Recall the Discrete LQR problem:

$$
\begin{aligned}
\underset{u \in \mathbb{R}^T}{\text{minimize}} \;& \frac{1}{2} x_T^\top Q_T x_T + \frac{1}{2} \sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t \\
\text{subject to } \;& x_{t+1} = A x_t + B u_t \text{ for all } 0 \le t \le T-1 \\
& x_0 = \text{initial condition} \\
& u_{LB} \preceq u_t \preceq u_{UB} \text{ for all } 0 \le t \le T-1.
\end{aligned}
$$

Discrete LQR in CVXPY

Problem setup

```
import numpy as np
import cvxpy as cvx

n = 5 # state dimension (x)
m = 5 # control dimenion (u)
T = 20 # number of timesteps in planning horizon
u_bound = 1.0 # bound on control effort

Q = np.eye(n) # state deviation cost
R = 2*np.eye(m) # control effort cost
A = np.random.normal(0,1,(n,n)) # dynamics
B = np.random.normal(0,1,(n,m))

x_0 = np.random.normal(0,1,(n,)) # initial condition
```

Discrete LQR in CVXPY

Iterative building of objective and constraints

```
X = {}
U = {}
cost_terms = []
constraints = []
```

Discrete LQR in CVXPY

Iterative building of objective and constraints

```
for t in range(T):
    X[t] = cvx.Variable(n) # state variable for time t
    U[t] = cvx.Variable(m) # control variable for time t
    cost_terms.append( cvx.quad_form(X[t],Q) ) # state cost
    cost_terms.append( cvx.quad_form(U[t],R) ) # control cost
    constraints.append( cvx.norm(U[t],"inf") <= u_bound ) # control effort

    if (t == 0):
        constraints.append( X[t] == x_0) # initial condition

    if (t < T-1 and t > 0):
        # dynamics constraint
        constraints.append( A @ X[t-1] + B @ U[t-1] == X[t] )
```

Discrete LQR in CVXPY

Solving the Problem

```
objective = cvx.Minimize(cvx.sum(cost_terms))

prob = cvx.Problem(objective, constraints)
prob.solve()
print(prob.status) # optimal, infeasible, etc.
print(prob.value) # optimal objective value
print(U[0].value) # optimal control
```