Lecture 8: Invariance

I. Basic Illustration of Invariance

- **Initial Remarks: Motivation and Concepts to start with**

Motivation for study on invariance

Again, recall the objectives of Constrained Optimal Control:

Given the dynamical system:

$$x^+ = f(x,u) \qquad (x,u) \in \mathcal{X},\mathcal{U}$$

Our objective is to design the control law $u = \kappa(x)$ such that system:

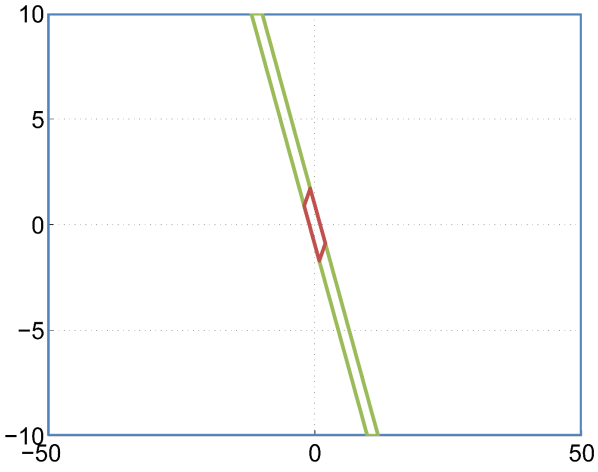
1. Satisfies constraints: $\{x_i\} \subset \mathcal{X}$, $\{u_i\} \subset \mathcal{U}$
2. Is asymptotically stable: $\lim_{i\to\infty} x_i = 0$
3. Optimizes "performance"
4. Maximizes the set $\{x_0 |$ Conditions $1-3$ are met $\}$

Invariance is the core of how to ensure #1: Satisfying constraints

To be more specific, consider the MPC example given in the last lecture:

1. Compute the optimal LQR controller and cost matrices: F_∞, P_∞
2. Compute the maximal invariant set $\mathcal{X}_f$ for the closed-loop linear system $x_{k+1} = (A + BF_\infty)x_k$ subject to the constraint:

$$\mathcal{X}_{cl}\left\{x \middle| \begin{bmatrix} A_x \\ A_u F_\infty \end{bmatrix} x \le \begin{bmatrix} b_x \\ b_u \end{bmatrix}\right\}$$



The blue region is the box constraints for the states, i.e. $A_x x \le b_x$, the green region is the input constraint represented by $A_u F_\infty x \le b_u$, and the **red region is invariant set, i.e. everything starting from the red region would end up in the red region.** The terminal region should be chosen as the red region because even if you start from the green region, it might still go out of the green region later!

This lecture would cover: **how can we calculate the invariant set to design our MPC controller?**

Autonomous Systems, Controlled systems, and Invariance

We consider the following two types of systems:

1. Autonomous system $x(k+1) = f_a(x(k))$
2. Controlled system $x(k+1) = f(x(k),u(k))$

Both systems are subject to state and input constraints: $x(k) \in \mathcal{X},\ u(k) \in \mathcal{U},\ \forall k \leq 0$

Two important concepts in this lecture:

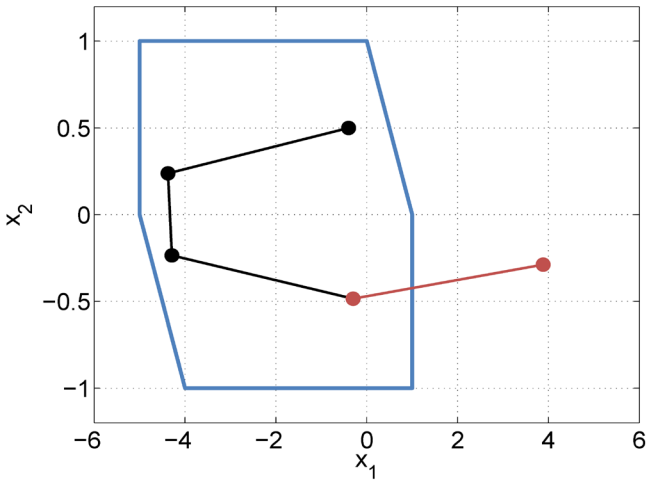
1. Invariance:

   The region in which an **autonomous** system will satisfy the constraints **for all time**

2. Controlled invariance

   Region for which there **exists a controller** so that the system satisfies the constraints **for all time**

- **Invariance: Which states are "good"?**



Consider the autonomous system:

$$\dot{x} = \begin{bmatrix} -2\zeta\omega & -\omega^2 \\ 1 & 0 \end{bmatrix} x(t)$$

Where $\omega = 10, \zeta = 0.01$, sampled at 10 Hz

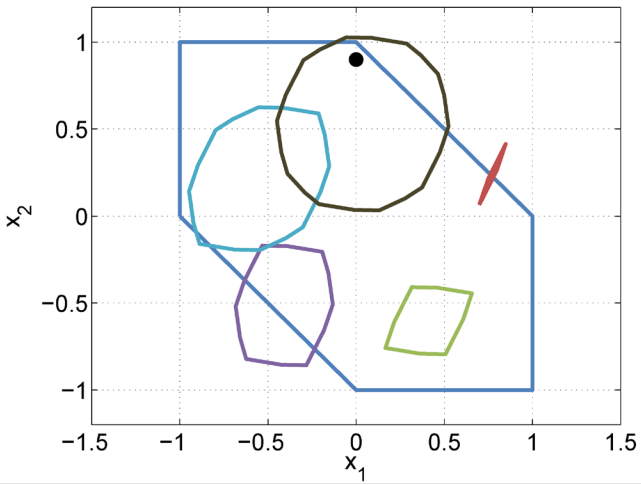
Constraints:

$$\mathcal{X} = \left\{ x \;\middle|\; \begin{array}{c} -5 \leq x_1 \leq 1 \\ -1 \leq x_2 \leq 1 \\ -5 \leq x_1 + x_2 \leq 1 \end{array} \right\}$$

The initial state is in the constraints. Is the next one? Yup, next one? Yup... Yup... No

Invariance: Look at an infinite distance into the future to determine if the trajectory beginning at the current state always remains in the constraints.

- **Controlled Invariance: Does a good input exist?**



Consider the controlled system:

$$x(k+1) = 0.9 \begin{bmatrix} \sin(0.3) & \cos(0.3) \\ -\cos(0.3) & \sin(0.3) \end{bmatrix} x(k) + 0.25 \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} u(k)$$

Constraints:

$$\|u(k)\|_\infty \leq 0.1$$

$$\|x(k)\|_\infty \leq 1$$

$$\|[1 \quad 1]x(k)\|_\infty \leq 1$$

Since we have a set of possible inputs, a point would lead to a set of possible next states (depending on the choices of control inputs), so the evolution of a single point would result in a series of region progress (red → green → purple → blue → black). And we can see that the black point is an invariant point since after four steps, it is included in the black region. i.e. There exists a control input to make it come back to the original place.

II. Invariance

- **Definitions and Conditions**

Consider the constraint satisfaction, for an autonomous $x(k+1)=f_a(x(k))$, or closed-loop system $x(k+1)=f(x(k),u(k))$ for a given controller κ.

Definition (Positive Invariant Set):

A set $\mathcal{O}$ is said to be a positive invariant set for the autonomous system $x(k+1)=f_a(x(k))$ if:

$$x(k)\in\mathcal{O}\Rightarrow x(k+1)\in\mathcal{O},\ \forall k\in\{0,1,\ldots\}$$

Definition (Maximal Positive Invariant Set):

The set $\mathcal{O}_\infty\subset\mathcal{X}$ is the maximal invariant set with respect to $\mathcal{X}$ if $0\in\mathcal{O}_\infty$, $\mathcal{O}_\infty$ is invariant and $\mathcal{O}_\infty$ contains all invariant sets that contain the origin.

Note:

1. From the definition, we know that if the invariant set is within the constraints, it provides a set of initial states from which the trajectory will never violate the system constraints.
2. The maximal invariant set is the set of all states for which the system will remain feasible if it starts in $\mathcal{O}_\infty$

Definition (Pre Set):

Given a set $\mathcal{S}$ and the dynamic system $x(k+1)=f(x(k))$, the pre-set of $\mathcal{S}$ is the set of states that evolve into the target set S in one time step:

$$\mathrm{pre}(\mathcal{S})=\{x|\ f(x)\in\mathcal{S}\}$$

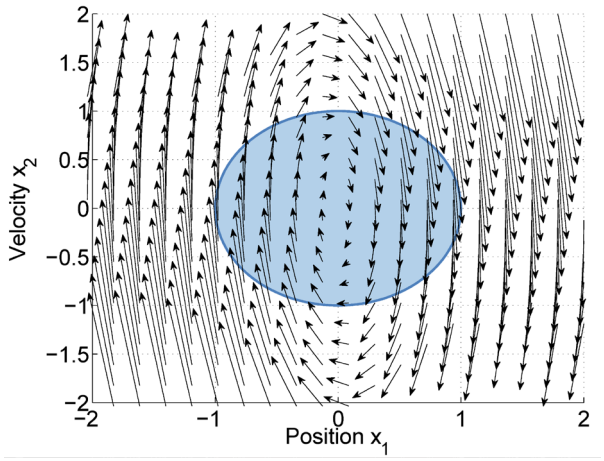
Pre-Set Example: Pendulum

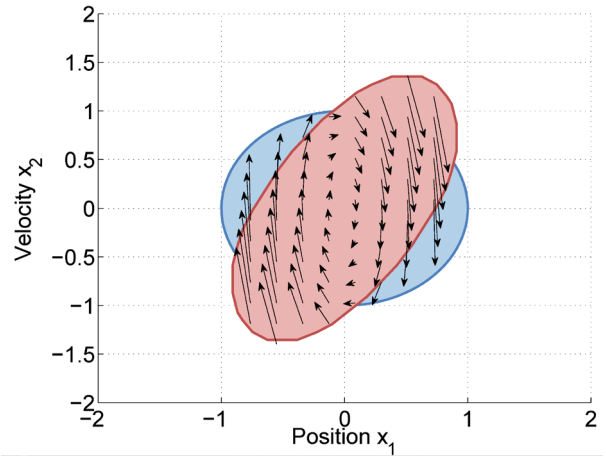
Consider the dynamical system:

$$x(k+1)=x(k)+\begin{bmatrix}x_2(k)\\ -9.8\sin x_1(k)-x_2(k)\end{bmatrix}$$

(Discretized with forward Euler at 1Hz) Target set: $\mathcal{T}=\{x|\|x\|_2\le 1\}$

Which states will be in the target set at the next point in time? ⇒ Consider the phase diagram.





Note:

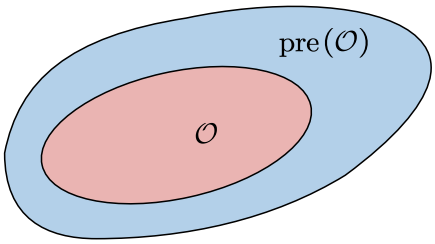
1. The phase diagram in the discrete case is constructed by connecting the current state and the next state. Here, we use the brute force method (sample every grid) to judge, keep the arrows with ends in the blue region, and the start of those arrows form the pink pre-set.
2. This is just a 2D toy example. Generally, it is extremely difficult to compute the pre-set

Invariant Set Conditions

Theorem (Geometric condition for invariance):

A set $\mathcal{O}$ is a positive invariant set if and only if

$$\mathcal{O} \subseteq \text{pre}(\mathcal{O})$$



We can prove from contradiction for both the necessary and sufficient conditions:

(Necessity $\Rightarrow$):

If $\mathcal{O} \nsubseteq \text{pre}(\mathcal{O})$, then $\exists\, \bar{x} \in \mathcal{O}$ such that $\bar{x} \notin \mathcal{O}$. From the definition of $\text{pre}(\mathcal{O})$, $f(\bar{x}) \notin \mathcal{O}$ and thus $\mathcal{O}$ is not a positive invariant set.

(Sufficiency $\Leftarrow$):

If $\mathcal{O}$ is not a positive invariant set, then $\exists\, \bar{x} \in \mathcal{O}$ such that $f(\bar{x}) \notin \mathcal{O}$. This implies that $\bar{x} \in \mathcal{O}$ and $\bar{x} \notin \text{pre}(\mathcal{O})$ and thus $\mathcal{O} \nsubseteq \text{pre}(\mathcal{O})$.

Also not that $\mathcal{O} \subseteq \text{pre}(\mathcal{O}) \Leftrightarrow \text{pre}(\mathcal{O}) \cap \mathcal{O} = \mathcal{O}$

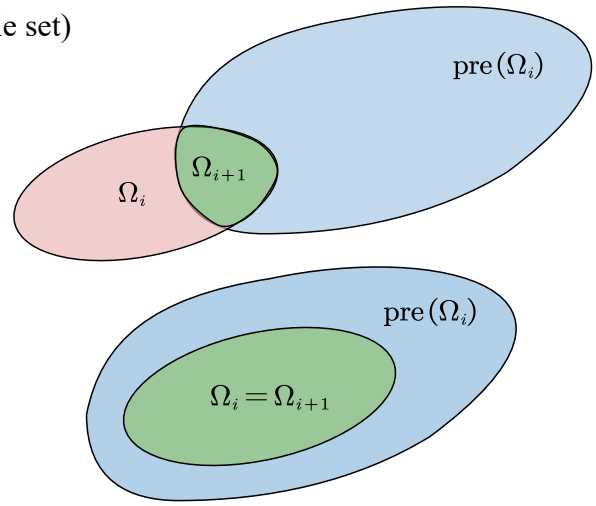
- **Computing Invariant Sets**

Algorithm (Conceptual Algorithm to Compute Invariant Set):

Input: $f, \mathcal{X}$ (system dynamics and feasible set)

Output: $\mathcal{O}_\infty$ (maximum invariant set)

$\Omega_0 \leftarrow \mathcal{X}$
loop
  $\Omega_{i+1} \leftarrow \text{pre}(\Omega_i) \cap \Omega_i$
  if $\Omega_{i+1} = \Omega_i$ **then**
    return $\mathcal{O}_\infty = \Omega_i$
  end if
end loop



The algorithm generates the set sequence $\{\Omega_i\}$ satisfying $\Omega_{i+1} \subseteq \Omega_i$ for all $i \in \mathbb{N}$ and it terminates when $\Omega_{i+1} = \Omega_i$ so that Ω_i is the maximal positive invariant set $\mathcal{O}_\infty$ for $x(k+1) = f(x(k))$.

Note: In practical numerical implementations, we can sometimes never get perfectly set equivalence, and the loop would generally run infinitely many times, so some small number ε should be set to give a proper cut-off of the algorithm.

Computing Invariant Sets: example

Given system:

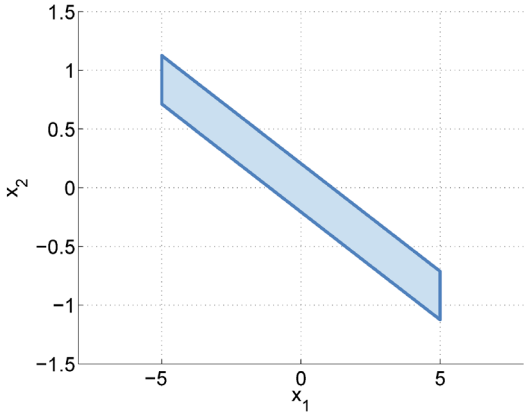
$$x(k+1)=\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}x(k)+\begin{bmatrix}1\\ 0.5\end{bmatrix}u(k)$$

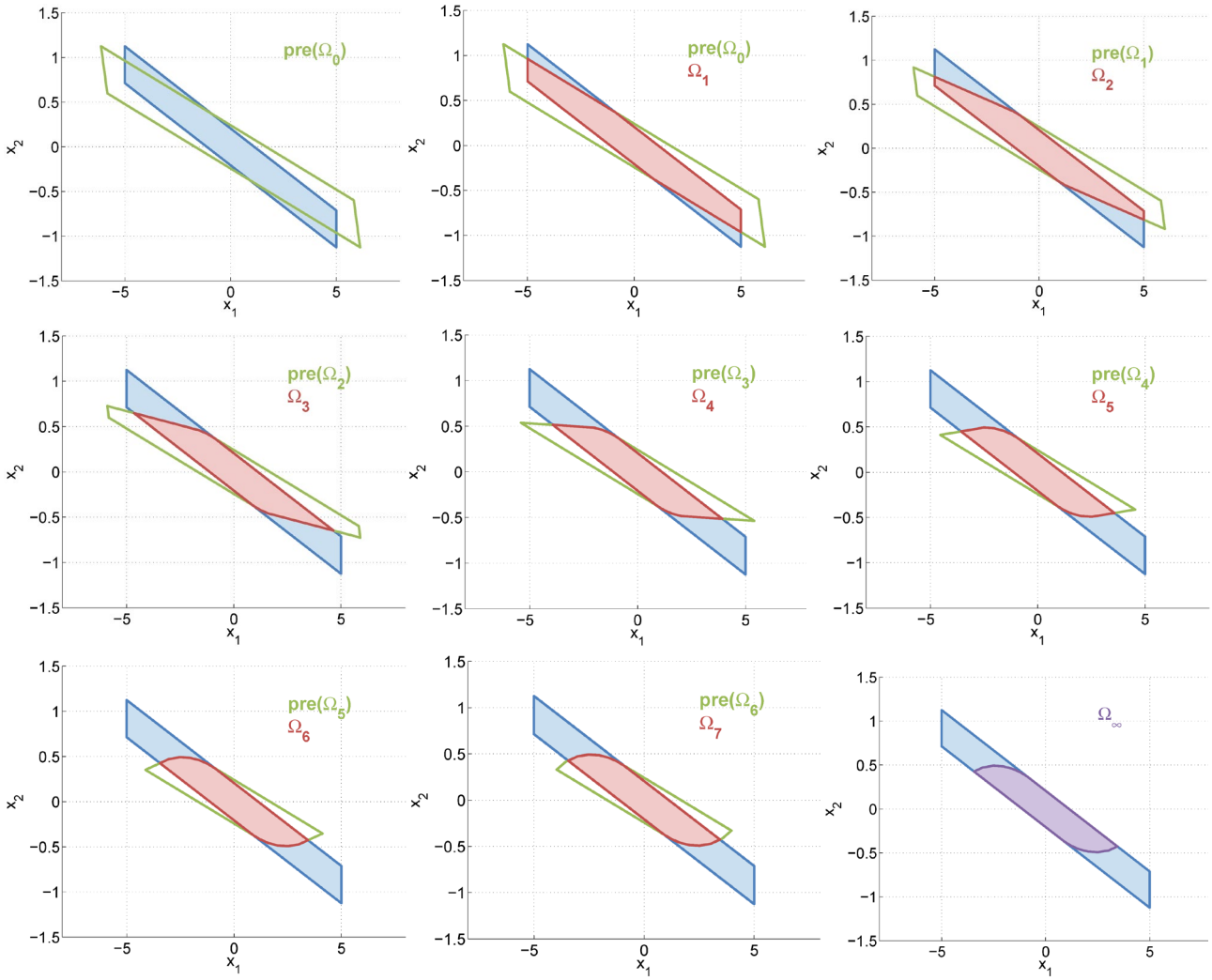
With constraints:

$$\begin{bmatrix}-5\\ -10\end{bmatrix}\le x(k)\le\begin{bmatrix}5\\ 10\end{bmatrix}$$

$$-0.1\le u(k)\le 0.1$$

Where $u(k) = Kx(k)$, with K the optimal LQR controller for $Q=I, R=90$





Note:

1. Later, we will show how to calculate the pre-set, here we just assume we know how to calculate it.
2. The algorithm, as long as we choose to use polytope computation, is not conservative. If we start from the whole feasible set, the maximal invariant set can be achieved exactly.
3. This is the invariant set with a given LQR controller. i.e. we have assigned a specific controller, so

this is not the maximal controlled invariant set but just a maximal system invariant set for the closed loop autonomous system. This also implies that with different given controllers, we will get different maximal invariant sets. In real applications, however, we generally would not explicitly decide the form of the controller to try to maximize the invariant set since it is not intuitive when the dimension scales up.

III. Controlled Invariance

- **Definitions, Conditions, and Computing**

Definition (Control Invariant Set):

A set $\mathcal{C} \subseteq \mathcal{X}$ is said to be a control invariant set if for controlled system $x(k+1) = f(x(k), u(k))$:

$$x(k) \in \mathcal{C} \Rightarrow \exists\, u \in \mathcal{U} \text{ such that } f(x(k), u(k)) \in \mathcal{C} \text{ for all } k \in \mathbb{N}^+$$

Definition (Maximal Control Invariant Set):

The set $\mathcal{C}_\infty$ is said to be the maximal control invariant set for the system $x(k+1) = f(x(k), u(k))$ subject to the constraints $(x(k), u(k)) \in \mathcal{X} \times \mathcal{U}$ if it is control invariant and contains all control invariant sets contained in $\mathcal{X}$

Note:

1. From the definition, we know that the control invariant set defines the states for which there exists a controller that will satisfy constraints for all time.
2. For all states contained in the maximal control invariant set $\mathcal{C}_\infty$ there exists a control law, such that the system constraints are never violated.

Control Invariant Set Conditions

Luckily, the concept of a pre-set extends to systems with exogenous inputs, and the same geometric condition holds for control invariant sets. As a result, the same conceptual algorithm can be used:

Definition (Pre-Set for Control Invariance):

$$\text{pre}(\mathcal{S}) = \{x \mid \exists\, u \in \mathcal{U} \text{ s.t. } f(x,u) \in \mathcal{S}\}$$

Theorem (Geometric Condition for Control Invariance):

$$\text{A set } \mathcal{C} \text{ is a control invariant set if and only if } \mathcal{C} \subseteq \text{pre}(\mathcal{C})$$

Algorithm (Conceptual Algorithm to Compute Control Invariant Set):

$\Omega_0 \leftarrow \mathcal{X}$
loop
  $\Omega_{i+1} \leftarrow \text{pre}(\Omega_i) \cap \Omega_i$
  if $\Omega_{i+1} = \Omega_i$ **then**
    return $\mathcal{C}_\infty = \Omega_i$
    end if
  end loop

Note: Everything generalized smoothly and we can prove them using the same scheme. The only drawback and biggest challenge in real practice are that the calculation of pre-set for controlled systems is much more complicated.

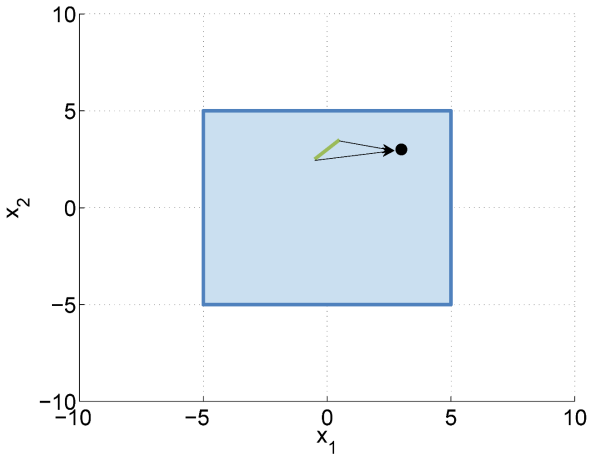
Computing Control Invariant Sets: example

Given system:

$$x(k+1)=\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}x(k)+\begin{bmatrix}1\\ 0.5\end{bmatrix}u(k)$$

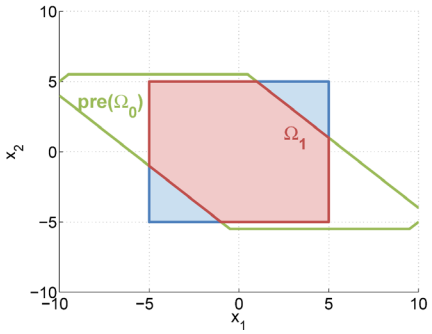
With constraints:

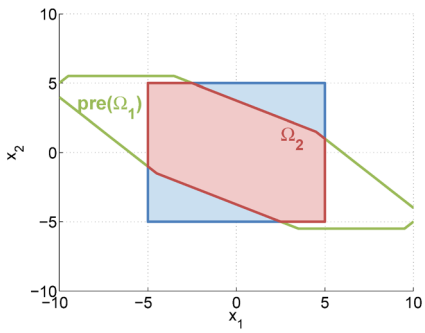
$\|x(k)\|_\infty \le 5, \|u(k)\|_\infty \le 1$

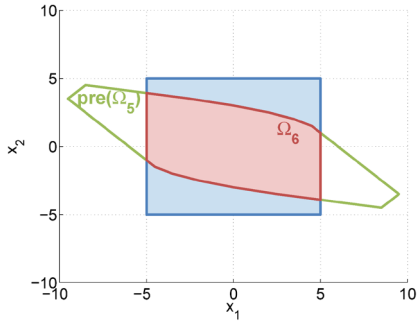


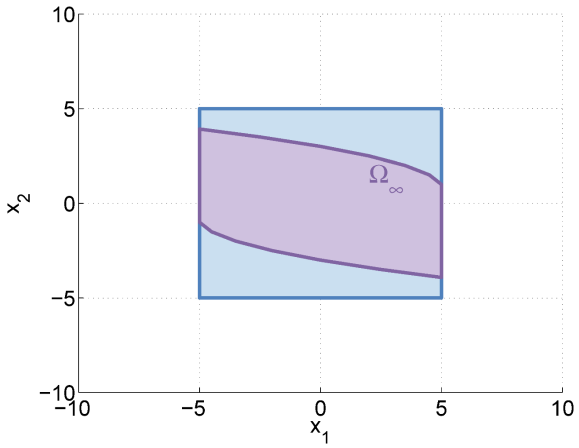
Note:

1. Here we do not know the exact form of the controller, unlike the example above, which is a closed-loop autonomous system
2. The big difference is that an entire set of states can map into a single point, as shown on the right









Note:

1. Later, we will show how to calculate the preset for the controlled system, here we just assume we know how to calculate it.
2. We can see that the pre-set is a lot larger, but much more difficult to compute
3. The maximum control invariant set is the best any controller can do

- **From Control Invariant Set to Control Law**

Let $\mathcal{C}$ be a control invariant set for the system $x(k+1)=f(x(k),u(k))$. A control law $\kappa(x(k))$ will guarantee that the system $x(k+1)=f(x(k),u(k))$ will satisfy the constraints for all time if:

$$f(x,\kappa(x))\in\mathcal{C} \quad \text{for all} \quad x\in\mathcal{C}$$

We can use this fact to **synthesize a control law** from a control invariant set by solving an optimization problem as below:

$$\kappa(x)=\operatorname{argmin}\{g(x,u)\mid f(x,u)\in\mathcal{C}\}$$

where g is any function (including $g(x,u)=0$)

Note: **This does not ensure that the system will converge,** but it will satisfy constraints.

Relation to MPC

A control invariant set is a powerful object

If one can compute the control invariant set, it provides a direct method for synthesizing a control law that will satisfy constraints, and the maximal control invariant set is the best any controller can do!

So why don't we always compute them?

$\Rightarrow$ We in fact can not compute them! For constrained linear systems, it is already often too complex

For constrained nonlinear systems, it is almost always too complex.

Therefore, we use **MPC**: A method of **implicitly describing a control invariant set** such that it is easy to represent and compute!

Note: See the summary part for more detailed descriptions of this idea.

IV. Polytopes and Polytopic Computation

- **Basic Polytope Operations and Definitions**

Recall the algorithm to compute the invariant set

$\Omega_0 \leftarrow \mathcal{X}$
loop
  $\Omega_{i+1} \leftarrow \text{pre}(\Omega_i) \cap \Omega_i$
  if $\Omega_{i+1} = \Omega_i$ **then**
    return $\mathcal{C}_\infty = \Omega_i$
  end if
end loop

We can see that there are several operations that needed to be done in this algorithm:

1. Represent the set Ω_i
2. Intersection of the set
3. Pre-set computation
4. Equality test of the set (bi-directional subset)

And we focus on the (convex) polytopes, here first we introduce some basic concepts to handle **# 1 the representation of the set**.

Definition (Polyhedron):

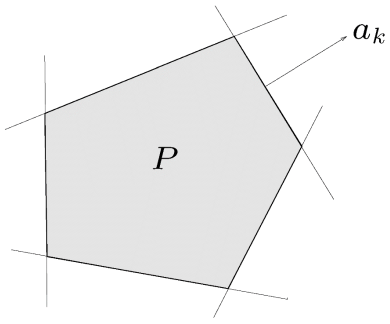
A polyhedron is the intersection of a finite number of halfspaces:

$$\mathcal{P} = \{x \mid a_i^{\mathrm{T}} x \leq b_i, i = 1 \ldots, n\}$$

A polytope is a **bounded polyhedron.**

Note: it is also often written as $\mathcal{P} = \{x \mid Ax \leq b\}$, for matrix $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, where the inequality is understood row-wise.

Examples of polyhedron: 2D and 3D



Definition (Convex hull):

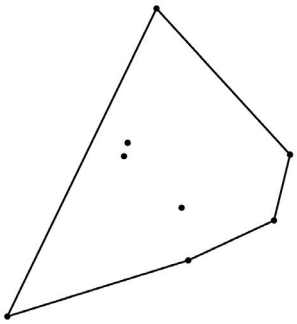
For any subset $\mathcal{S}$ of $\mathbb{R}^d$, the convex hull $\mathrm{co}(\mathcal{S})$ of $\mathcal{S}$ is the intersection of all convex sets containing $\mathcal{S}$. Since the intersection of two convex sets is convex, it is the smallest convex set containing $\mathcal{S}$.

Proposition (Convex hull):

The convex hull of a set $\mathcal{S} \subseteq \mathbb{R}^d$ is Given a set of points $\{v_1, v_2, \ldots, v_k\}$ in $\mathbb{R}^d$, their convex hull is:

$$\mathrm{co}(\{v_1, \ldots, v_k\}) = \left\{ x \;\middle|\; x = \sum_i \lambda_i v_i,\ \lambda_i \geq 0,\ \sum_i \lambda_i = 1,\ \forall i = 1 \ldots, k \right\}$$

Examples of convex hulls: 2D and 3D



Definition (Polytope Reduction):

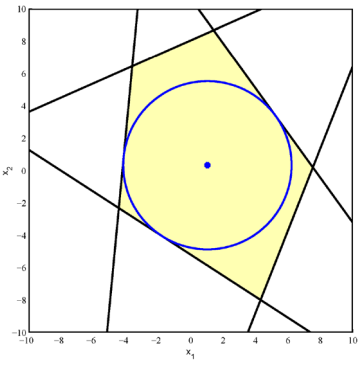
The computation of the minimal representation of a polytope $\mathcal{P}$.

A polytope $\mathcal{P} \subset \mathbb{R}^n, \mathcal{P} = \{x \in \mathbb{R}^n : Ax \leq b\}$ is in a minimal representation if the removal of any row in $Ax \leq b$ would change it (i.e., if there are no redundant constraints).

Definition (Chebyshev Ball):

The Chebyshev Ball of a polytope $\mathcal{P}$ corresponds to the largest radius ball $\mathcal{B}(x_c, R)$ with the center x_c such that $\mathcal{B}(x_c, R) \subset \mathcal{P}$

Note: The calculation of the Chebyshev Ball (largest ball inscribing the polytope) can be done efficiently. However, calculating the smallest ball surrounding the polytope is hard.



Definition (Polytope Projection):

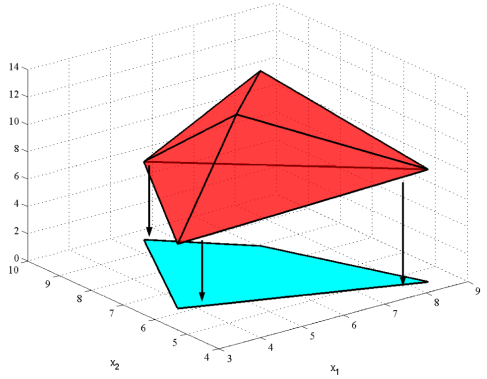
Given a polytope:

$\mathcal{P} = \{[x^{\mathrm{T}} \quad y^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{n+m} : A^x x + A^y y \leq b\} \subset \mathbb{R}^{n+m}$, the projection onto the x-space $\mathbb{R}^n$ is defined as:

$$\mathrm{proj}_x(\mathcal{P}) = \{x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^m : A^x x + A^y y \leq b\}$$

In MATLAB with polyhedron created using MPT toolbox, we can call: Q = projection(P, dim) to get the projection.

Note: Projection operation plays the most important role in finding the control invariant set



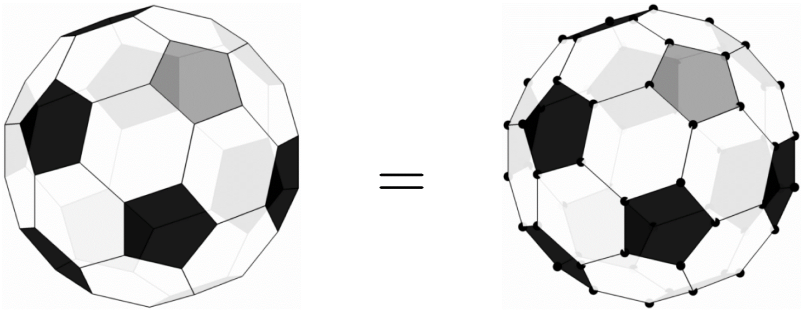
- **Polytopes Representations and Constraints in MPC**

Definition (Minkowski-Weyl Theorem):

For $\mathcal{P} \subset \mathbb{R}^d$, the following statements are equivalent:

$\mathcal{P}$ is a polytope, that is to say, $\mathcal{P}$ is bounded and there exist $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$ such that $\mathcal{P} = \{x \mid Ax \leq b\}$ (Also called the **inequality representation** of polytopes)

$\mathcal{P}$ is finitely generated, that is to say, there exist a finite set of vectors $\{v_i\} = \{v_1, \ldots, v_s\}$ such that $\mathcal{P} = \mathrm{co}(\{v_1, \ldots, v_s\})$ (Also called the **vertex representation** of polytopes)



Constraints in MPC

There are four types of most commonly used constraints in MPC:

1. Input Saturation:

$$u_{lb} \leq u(k) \leq u_{ub} \Rightarrow \begin{bmatrix} 1 \\ -1 \end{bmatrix} u(k) \leq \begin{bmatrix} u_{ub} \\ -u_{lb} \end{bmatrix}$$

2. Magnitude constraints

$$\| Cx(k) \|_\infty \leq \alpha \Rightarrow \begin{bmatrix} C \\ -C \end{bmatrix} x(k) \leq \mathbf{1}\alpha$$

3. Rate constraints

$$\| x(k) - x(k+1) \|_\infty \leq \alpha \Rightarrow \begin{bmatrix} I & -I \\ -I & I \end{bmatrix} \begin{bmatrix} x(k) \\ x(k+1) \end{bmatrix} \leq \mathbf{1}\alpha$$

4. Integral constraints

$$\| x(k) \|_1 \leq \alpha \Rightarrow x(k) \in \mathrm{co}(e_i\alpha, -e_i\alpha)$$

We can see that constraints in MPC are commonly described as a set of inequalities, and hence are polytopes. This would be our standing assumption in the following contents.

Note: We can also see that we would mostly discuss polytopes using their inequality representation. Only the integral constraints would require transformations of vertex representation

- **Intersection, Pre-Set Computation and Equality Test**

Now we can discuss in detail the other three operations needed to compute invariant sets.

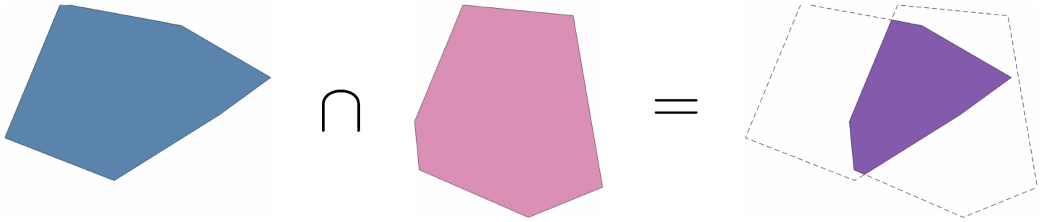
Intersection of Polytopes

The intersection $\mathcal{I} \subseteq \mathbb{R}^n$ of sets $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{T} \subseteq \mathbb{R}^n$ is:

$$\mathcal{I} = \mathcal{S} \cap \mathcal{T} = \{x \mid x \in \mathcal{S} \text{ and } x \in \mathcal{T}\}$$

And we use the intersection of polytopes in inequality form since it is easy:

$$\begin{matrix}\mathcal{S}\colon \{x|\ Cx\le c\}\\ \mathcal{T}\colon \{x|\ Dx\le d\}\end{matrix} \Rightarrow \mathcal{S}\cap\mathcal{T}=\left\{x\middle|\begin{bmatrix}C\\D\end{bmatrix}x\le\begin{bmatrix}c\\d\end{bmatrix}\right\}$$



Note: Intersection of polytopes in vertex form is difficult (exponential complexity)

Pre-Set Computation for Autonomous System

Recall our definition for pre-set: Given a set $\mathcal{S}$ and the autonomous system $x(k+1)=Ax(k)$. the pre-set of $\mathcal{S}$ is the set of states that evolve into the target set $\mathcal{S}$ in one time step:

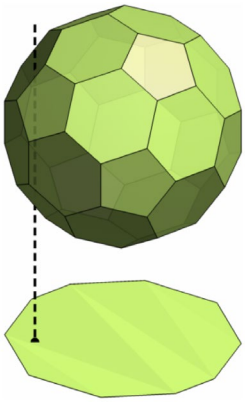
$$\text{pre}(\mathcal{S})=\{x|\ Ax\in\mathcal{S}\}$$

Therefore, if $\mathcal{S}\colon \{x|\ Fx\le f\}$, then $\text{pre}(\mathcal{S})=\{x|\ FAx\le f\}$

Pre-Set Computation for Controlled System

Consider the controlled system $x(k+1)=Ax(k)+Bu(k)$ under the constraints $u(k)\in\mathcal{U}=\{Gu\le g\}$ and the set $\mathcal{S}=\{x|\ Fx\le f\}$, we have:

$$\begin{aligned}\text{pre}(\mathcal{S})&=\{x|\ \exists u\in\mathcal{U},\ Ax+Bu\in\mathcal{S}\}\\&=\{x|\ \exists u\in\mathcal{U},\ F(Ax+Bu)\le f\}\\&=\left\{x\middle|\ \exists u,\ \begin{bmatrix}FA & FB\\0 & G\end{bmatrix}\begin{bmatrix}x\\u\end{bmatrix}\le\begin{bmatrix}f\\g\end{bmatrix}\right\}\end{aligned}$$



Recall the definition mentioned above. **This is a projection operation**

Remarks on Polytopic Projection

Given a polytope $\mathcal{P}=\{(x,y)\in\mathbb{R}^n\times\mathbb{R}^d|\ Cx+Dy\le b\}$ find a matrix E and vector e such that the polytope:

$$\mathcal{P}_\pi=\{x|\ Ex\le e\}=\{x|\ \exists y,(x,y)\in\mathcal{P}\}$$

is called the polytopic projection

Computing projections in inequality form is computationally complex.

Note: If $C\in\mathbb{R}^{m\times n}$ and $E\in\mathbb{R}^{q\times n}$:

1. q can be an exponential function of m (worst case)
2. Standard algorithms take time and space doubly exponential in m and q
3. The best algorithm to date is polynomial time in m and linear in q, but it requires $\mathcal{P}$ to have a special structure, which is a form of general position (Colin Jones' PhD Thesis).
4. We do not need to know the details about the algorithm here.

Subset test and Equality (Bi-Directional Subset) Test

The equality test (terminal condition) is **essentially the subset test of the polytopes, and the subset test can be concluded using a series of bi-directional tests:**

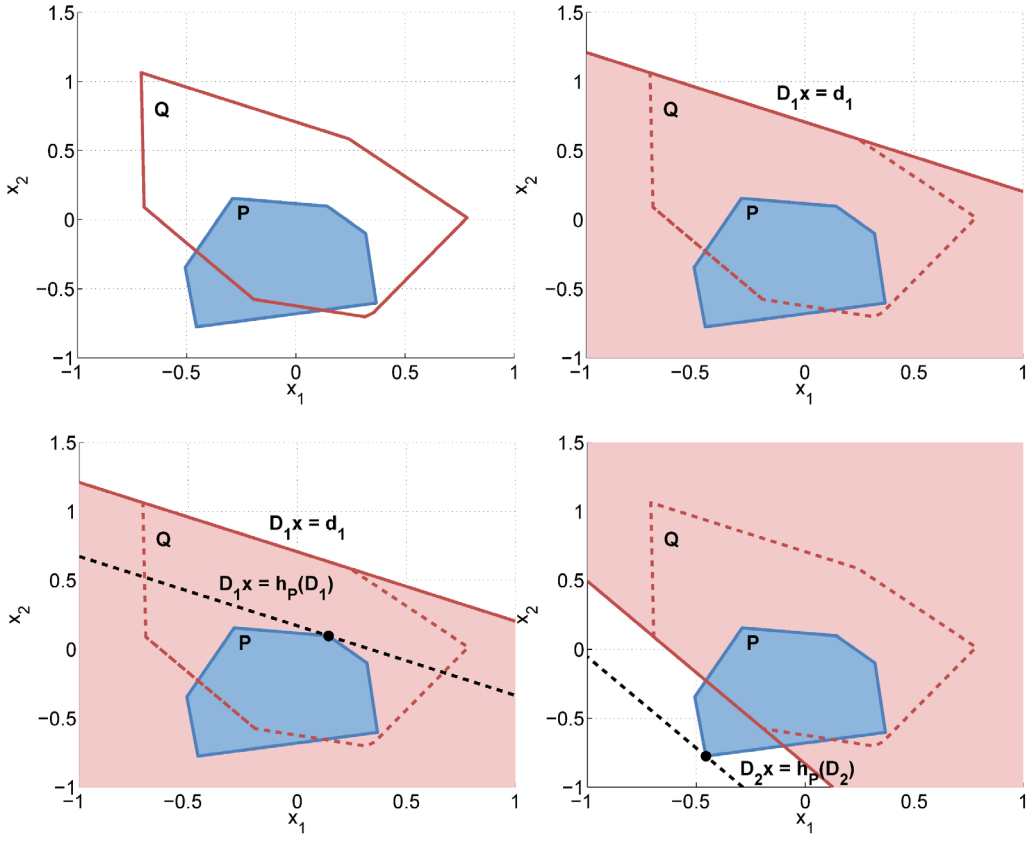
Is $\mathcal{P}=\{x|\ Cx\leq c\}$ contained in $\mathcal{Q}=\{x|\ Dx\leq d\}$?

$\Rightarrow$ if $\mathcal{P}\subset\{x|\ Dx\leq d\}$ for each row D_i of D $\Rightarrow$ Solve a series of linear programming.

How? Define the support function of the set $\mathcal{P}$:

$$h_p(D_i)=\max_x\ D_i x$$

$$\text{subj. to. } Cx\leq c$$



$h_P(D_1)\leq d_1$ ✔ $h_P(D_2)>d_2$ ✖

V. Summary

- **Summary For Linear Systems / Polyhedral Constraint**

Polyhedral invariant set

1. Can represent the maximum invariant set
2. Can be complex (many inequalities) for more than ~ 5-10 states
3. The resulting MPC optimization will be a quadratic program

Ellipsoidal invariant set

1. Smaller than polyhedral (not the maximal invariant set)
2. Easy to compute for large dimensions

3. Fixed complexity
4. The resulting MPC optimization will be a quadratically constrained quadratic program

(See extra slides at the end of the lecture to learn more about ellipsoidal invariant sets)

- **MPC and Control Invariance**

Calculating the invariant set is **very difficult, very complex but very useful, using MPC turns an invariant set into a control invariant set with tractable computation. Why?**

1. Controlled invariance means that we can ensure the system to satisfy constraints all the time. That is, the **system's feasibility is guaranteed all the time.**
2. Recall that in last lecture, when terminal set and cost are introduced, we can get **recursively feasibility, and that makes the MPC problem always feasible** (of course, we need the initial feasibility and some assumptions to hold). In this sense, setting the terminal set and terminal cost (for a terminal autonomous but not controlled system) mimics the effect of the control invariant set since they **both ensure feasibility in the long run**.
3. The big difference lies in that MPC is online computation, and the exact calculation of the control invariant set is offline computation. Even if we have time for offline computation, it is usually not tractable for complex systems.

- **Possible Generalizations and Related Topics**

Generalization of the Simple Cases

Basically, there are two directions to generalize: high dimensional case and non-convex case

Nonconvex case:

Need to split the non-convex set into convex polytopes, i.e. we will have a set of convex sets (polytopes), and that would be much more complicated in numerical realization.

High-dimensional case:

Using ellipsoids instead of polytopes would be the best choice, but as is mentioned above, ellipsoids are somewhat conservative. Sum-of-Squares programming using LMI may be the intermediate choice However, we still need to make tradeoffs.

Related Research Topics

A related concept is control barrier function (CBF), which also utilizes the power of control invariance. There are two main differences between CBF and the contents we discuss today:

1. CBF mostly used for nonlinear systems. For linear systems, the results should be very similar.
2. CBF usually explicitly defines a Lyapunov-like function and adds constraints for states to be pushed back on boundaries, while MPC implicitly uses the idea of Lyapunov stability. This makes these two methods, though utilizing the same idea, in fact design/synthesis controllers in a different ways, both conceptually and computationally